OXUF922
DATA SHEET

FEATURES

- 1394B Phy Interface and Link
- Backward compatible with 1394A
- 800Mb/s Support
- SBP2 Bus Mastering
- USB2.0 Phy and Link Layer
- Backward Compatible with USB1.1
- 480Mb/s Support
- Bulk Transfer USB Slave receiver
- 100MHz IDE interface
- Full support for AT A6 Drives
- LBA support for large drives
- Full Master and Slave support
- Supports Data Rates up to 80MBytes/s
- Simple DMA master scheme
- ORB accelerator
- Groups orbs and provides high speed response for hard coded commands
- High-Performance, Low Power ARM7TDMI Processor
- 50MHz Clock Rate
- 32Bit
- 8KByte Closely Coupled RAM
- 12Mb/s Async UART
- Extended 128Byte Buffer
- Local Bus Support
- Generic bus support for external peripherals
- 8 GPIO pins
- Programmable IO
- 6KByte Cache for USB or 1394 data
- 0.18µm advanced CMOS process
- 160 LQFP or 176 VFBGA
- I/O Supply Voltage 3.3V +/- 0.3V
- Core Supply Voltage 1.8V +/- 0.15V

DESCRIPTION

The OXUF922 is a combined USB2.0 and 1394B bridge to any IDE device or DMA based SRAM architecture. Optimised for performance the OXUF922 has a flexible ARM7 embedded processor, which can be programmed for next generation Computer Peripheral and Consumer Applications.

- 1394B and USB2.0 PC and Apple Storage Devices for HDD, DVD, CD, CF and Tape or Dual LUN Combinations of the above.
- Consumer Appliance (STB, PVR) Storage solutions for HDD and DVD.
- Digital Camera Companions
- Digital Camcorder Storage Devices
- Advanced Compressed Audio Players (MP3)
- Printers and Scanners
- 1394B Raid Array with UART Back-channel monitor support.

The embedded ARM7TDMI processor enables a new set of innovative products to be supported through custom firmware development. Program code can be programmed 'In-System' through the 1394 port simplifying manufacture. The 1394 Link supports A, B or Beta Phys and is fully backward compatible with earlier 1394 standards. The Link interface supports data rates up to S800 (800Mb/s) and has a rich complement of 1394 second layer functionality.

The OXUF922 fully supports 1394 Peer-To-Peer operation enabling PC-less communication over the 1394 Bus for file copying or manipulation. In addition the OXUF922 is backward compatible with the Oxford FW900 and FW911 enabling a limited Peer-To-Peer operation by SBP2 mastering the 1394 bus.

The OXUF922 has an ORB data accelerator which, without processor intervention, significantly increases the performance of the bridge device when transferring many small files.

.

CONTENTS

REVISION HISTORY

REV	DATE	REASON FOR CHANGE / SUMMARY OF CHANGE
0.4	15/4/02	Output pin drive strengths added. Package drawing clarified
0.5	29/4/02	active polarity of pin 41 (FFLASH) changed
0.6	23/7/02	added reset / PLL en info
0.7	29/7/02	Added VFBGA info. 1394 Phy Link timings updated. Revision updated
0.8	19/9/02	Added clarification to PLL_TEST and PLL_EN pins. Front page updated
0.9	6/11/02	Added block descriptions and register details
0.10	11/11/02	General formatting changes, addition of CPU related functions and registers
0.11	18/11/02	Added Static Interface details and timing Added ATA bypass details
0.12	19/11/02	Final review. Missing references fixed. GPIO tables fixed
1.0	20/11/02	First Release
1.1	06/12/02	Add more detail into sections 7.4 Link-Core, 7.5 FIFO Manager., 7.10 Serial Controller and 7.11 Serial Audio.

1 OXUF922 DESCRIPTION

Description

The OXUF922 is a combined USB2.0 and 1394 bridge to IDE device. The 1394 Link supports A or B PHYs and is fully backward compatible with earlier 1394 standards. The OXUF922 also includes an on-chip USB2.0 PHY allowing up to 480Mb/s data transfer.

The 1394 Link interface supports data rates up to S800 (800Mb/s) and has a rich complement of 1394 second layer functionality. The OXUF922 fully supports 1394 Peer-To-Peer operation enabling PC-less communication over the 1394 Bus for file copying or manipulation. The chip has an ORB data accelerator which, without processor intervention, significantly increases the performance of the bridge device when transferring many small files, and frees the processor for other tasks.

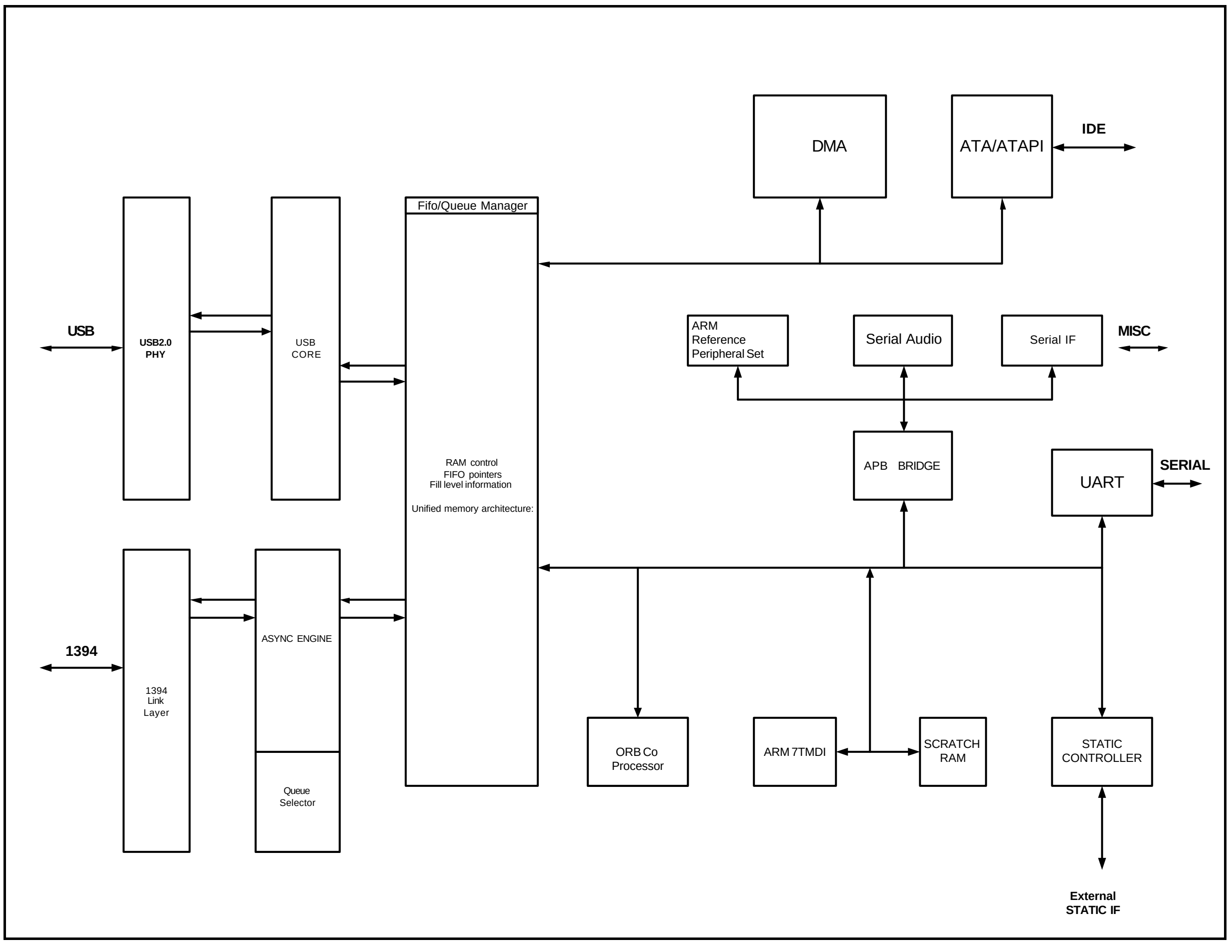
The OXUF922 integrates an ARM7TDMI operating at 50 MHz with a closely coupled zero wait state 2Kx32 SRAM provided for local program, stack or data storage. The closely coupled ram supports byte, word and quadlet access. The embedded ARM7TDMI processor enables a new set of innovative products to be supported through custom firmware development. Program code can be programmed 'In-System' through the 1394 port simplifying manufacture.

By combining the ARM7TDMI processor core with on-chip SRAM and a wide range of peripheral functions including timers, serial communication controllers and Oxford Semiconductor extensive knowledge of 1394 and USB applications the OXUF922 provides a highly flexible and cost-effective solution to many compute-intensive applications requiring connectivity and storage on top of normal micro-controller features.

Development Support

The OXUF922 is be supported by an open source 'C' compiler (GCC) coupled with a free UNIX environment (Cygwin) to enable the compiling and debugging of the software. Free Uploader software allows quick and easy method of uploading software to the device in-circuit.

1.1 Block Diagram



2 PIN INFORMATION

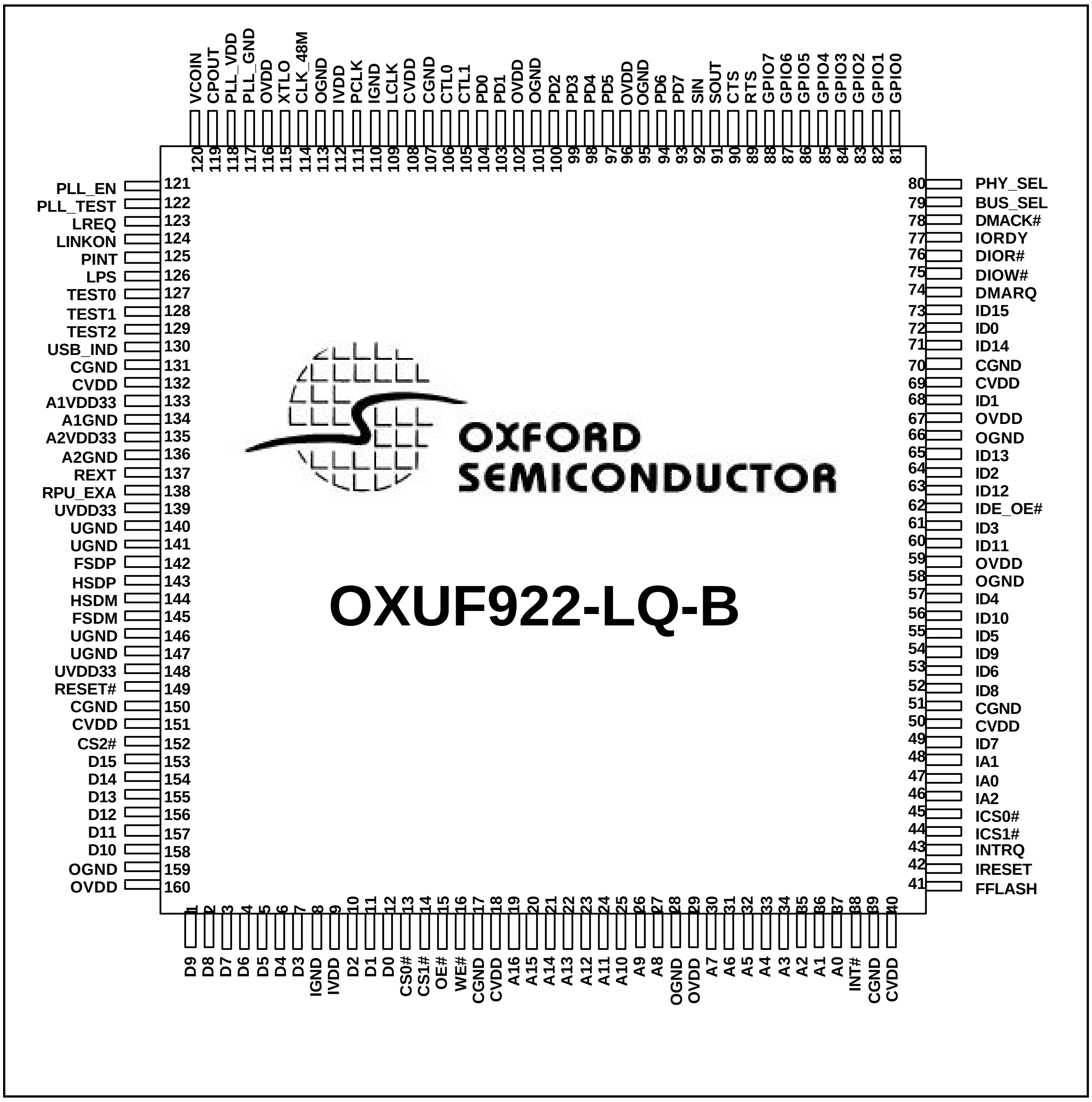
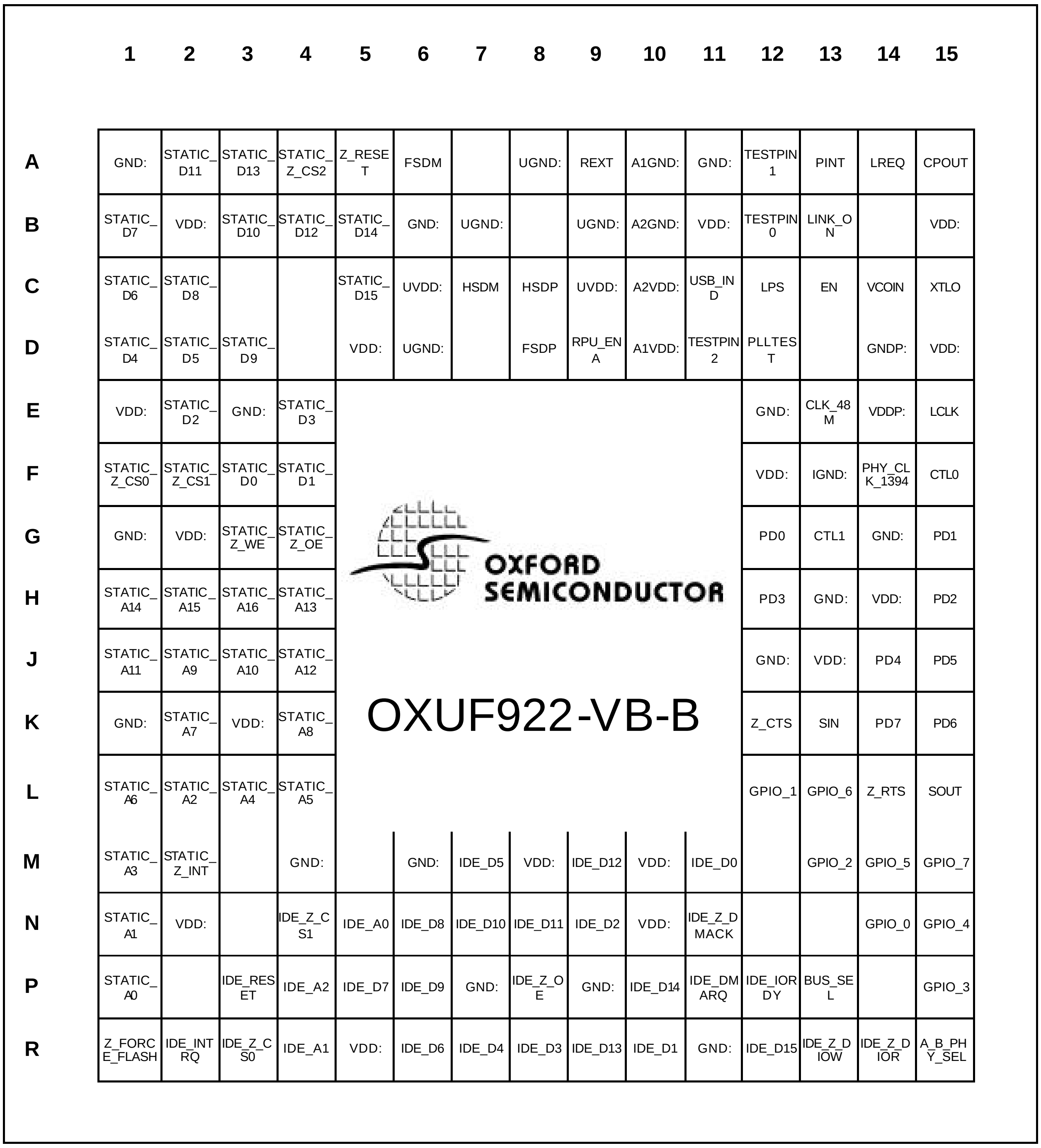


Figure 1 – LQFP Pin Information



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	GND:	STATIC_D11	STATIC_D13	STATIC_Z_CS2	Z_RESET	FSDM		UGND:	REXT	A1GND:	GND:	TESTPIN1	PINT	LREQ	CPOUT
B	STATIC_D7	VDD:	STATIC_D10	STATIC_D12	STATIC_D14	GND:	UGND:		UGND:	A2GND:	VDD:	TESTPIN0	LINK_ON		VDD:
C	STATIC_D6	STATIC_D8			STATIC_D15	UVDD:	HSDM	HSDP	UVDD:	A2VDD:	USB_IND	LPS	EN	VCOIN	XTLO
D	STATIC_D4	STATIC_D5	STATIC_D9		VDD:	UGND:		FSDP	RPU_ENA	A1VDD:	TESTPIN2	PLLTEST		GNDP:	VDD:
E	VDD:	STATIC_D2	GND:	STATIC_D3								GND:	CLK_48M	VDDP:	LCLK
F	STATIC_Z_CS0	STATIC_Z_CS1	STATIC_D0	STATIC_D1								VDD:	IGND:	PHY_CLK_1394	CTL0
G	GND:	VDD:	STATIC_Z_WE	STATIC_Z_OE								PD0	CTL1	GND:	PD1
H	STATIC_A14	STATIC_A15	STATIC_A16	STATIC_A13								PD3	GND:	VDD:	PD2
J	STATIC_A11	STATIC_A9	STATIC_A10	STATIC_A12								GND:	VDD:	PD4	PD5
K	GND:	STATIC_A7	VDD:	STATIC_A8				OXUF922-VB-B				Z_CTS	SIN	PD7	PD6
L	STATIC_A6	STATIC_A2	STATIC_A4	STATIC_A5								GPIO_1	GPIO_6	Z_RTS	SOUT
M	STATIC_A3	STATIC_Z_INT		GND:		GND:	IDE_D5	VDD:	IDE_D12	VDD:	IDE_D0		GPIO_2	GPIO_5	GPIO_7
N	STATIC_A1	VDD:		IDE_Z_CS1	IDE_A0	IDE_D8	IDE_D10	IDE_D11	IDE_D2	VDD:	IDE_Z_DMACK			GPIO_0	GPIO_4
P	STATIC_A0		IDE_RESET	IDE_A2	IDE_D7	IDE_D9	GND:	IDE_Z_OE	GND:	IDE_D14	IDE_DMARQ	IDE_IORDY	BUS_SEL		GPIO_3
R	Z_FORCE_FLASH	IDE_INTRQ	IDE_Z_CS0	IDE_A1	VDD:	IDE_D6	IDE_D4	IDE_D3	IDE_D13	IDE_D1	GND:	IDE_D15	IDE_Z_DIOW	IDE_Z_DIOR	A_B_PHY_SEL

Figure 2 – VFBGA Pin Information
(Top view)

3 PIN DESCRIPTIONS

LQFP Pin	VFBGA	#	Type[1]	Name	Description
1394 LINK					
93,94,97-100,103,104	K14,K15,J15, J14,H12,H15 G15,G12	8	B_4	PD[7:0]	Phy-Link Data Bus
105,106	G13,F15	2	B_4	CTL[1:0]	Phy-Link Control Bus
111	F14	1	I	PCLK	49.152 (A)/ 98.304 (B) MHz clock sourced by PHY
123	A14	1	O_4	LREQ	Link Request
124	B13	1	IU	LINKON	Requests link to power up when in a low power mode
126	C12	1	O_4	LPS	Indicates to phy that link is powered and ready
109	E15	1	O_4	LCLK	B Only – PCLK returned to PHY
125	A13	1	I	PINT	B Only – PHY Interrupt
STATIC IF					
153-158,1,2	C5,B5,A3,B4, A2,B3,D3,C2	8	B_8	D[15:8]	Static IF external upper data bus when TEST[2:0] = 000 or 010
153-157	C5,B5,A3,B4, A2	0	B_8	TDI,TDO, TMS, TCK, TRST	JTAG Bus when TEST[2:0] =001
158,1,2	B3,D3,C2	0	O_8	A[19:17]	Extended Static IF external address bus when TEST[2:0] =001
156-158,1,2	B4,A2,B3,D3, C2	0	O_8	A[21:17]	Extended Static IF external address bus when TEST[2:0] =101
3-7,10-12	B1,C1,D2,D1, E4,E2,F4,F3	8	T_B_8	D[7:0]	Static IF external lower data bus
19-27,30-37	H3,H2,H1,H4, J4,J1,J3,J2,K 4,K2,L1,L4,L3 ,M1,L2,N1,P1	17	T_O_8	A[16:0]	Static IF external address bus
152,14,13	A4,F2,F1	3	T_O_8	CS#[2:0]	Static IF external chip selects. CS0# is always used for Flash.
15	G4	1	T_O_8	OE#	External output enable. Active when reading data from external devices including Flash
16	G3	1	T_O_8	WE#	Write Enable. Active when writing to external devices
38	M2	1	T_B_8	INT#	External CPU interrupt – Defined as input by default
38	M2	0	T_B_8	CLK_50	50MHz clock output – pin function controlled by firmware
79	P13	1	I	BUS_SEL	High for 8 bit external Flash device, low for 16 bit external Flash device
USB					
143	C8	1	USB_B	HSDP	High Speed Data+ Connect to USB D+ line
144	C7	1	USB_B	HSDM	High Speed Data - Connect to USB D- line
137	A9	1	USB_O	REXT	200 uA fixed reference bias current pin. Connect to ground via the external resister Rext 12.5K$\Omega \pm$ 1%
142	D8	1	USB_B	FSDP	Full Speed Data + Connect to USB D+ line via external resister (39Ω $\pm$2 %)
145	A6	1	USB_B	FSDM	Full Speed Data - Connect to USB D- line via external resister (39Ω $\pm$2 %)

138	D9	1	USB_O	RPU_ENA	In FS mode and HS chirp mode, this pin is set to H to supply Vdd to external resister Rup (1.5KΩ ±5 %). In HS operation, it is tri-stated.
IDE					
73,71,65,63,60,56,54,52, 49,53,55,57,61,64,68,72	R12,P10,R9, M9,N8,N7,P6, N6,P5,R6,M7, R7,R8,N9,R1 0,M11	16	T_B_4	ID[15:0]	IDE data bus
46,48,47	P4,R4,N5	3	T_O_4	IA[2:0]	IDE address bus
44,45	N4,R3	2	T_O_4	ICS#[1:0]	IDE chip select. Selects IDE drive 0 or 1
62	P8	1	T_O_8	IDE_OE#	IDE output enable. Only used when external buffering is required to drive IDE data bus
42	P3	1	T_O_4	IRESET	IDE interface reset
74	P11	1	T_I	DMARQ	
75	R13	1	T_O_4	DIOW#	IDE interface write strobe
76	R14	1	T_O_4	DIOR#	IDE interface read strobe
77	P12	1	T_I	IORDY	
78	N11	1	T_O_4	DMACK#	
43	R2	1	T_I	INTRQ	
UART					
91	L15	1	O_4	SOUT	Transmitter serial data output.
89	L14	1	O_4	RTS#	Active-low Request-To-Send output.
92	K13	1	I	SIN	Receiver serial data input.
90	K12	1	I	CTS#	Active-low Clear-To-Send input.
PLL + OSC					
114 115	E13 C15	1 1	I O	XTLI XTLO	OSC input from 48 MHz crystal (or crystal oscillator) OSC output
122	D12	1	I	PLL_TEST	Test Enable (tie low for normal operation)
121	C13	1	I	PLL_EN	VCO Enable (tie high for normal operation)
120	C14	1	I	VCOIN	Loop filter in
119	A15	1	O	CPOUT	Loop filter out
MISC					
88	M15	1	T_B_4	GPIO 7	General purpose IO / DTR# / Serial Audio driver word
87	L13	1	T_B_4	GPIO 6	General purpose IO / DSR# / Serial Audio driver data
86	M14	1	T_B_4	GPIO 5	General purpose IO / DCD# / Serial Audio driver clk
85	N15	1	T_B_4	GPIO 4	General purpose IO / RI
84	P15	1	T_B_4	GPIO 3	General purpose IO
83	M13	1	T_B_4	GPIO 2	General purpose IO / STATIC WE2# (upper byte write enable)
82	L12	1	T_B_4	GPIO 1	General purpose IO / Serial Data
81	N14	1	T_B_4	GPIO 0	General purpose IO / Serial Clk
129-127	D11,A12,B12	3	I	TEST[2:0]	Test pins to select functional modes. See section 3.3 Functional Modes
149	A5	1	I	RESET#	Active low reset
41	R1	1	I	FFLASH	While low the device is configured in force_flash mode.
80	R15	1	I	PHY_SEL	Selects A or B Link PHY
130	C11	1	I	USB_IND	High to indicate USB

Power and ground[2]					
18,40,50,69,108,132,151	G2,N2,R5,N10,F12,B11,D5	7	CVDD	digital VDD	1.8V Core voltage
17,39,51,70,107,131,150	G1,M4,M6,R11,G14,A11,B6	7	CGND	digital GND	
9,29,59,67,96,102,112,116,160	E1,K3,M8,M10,J13,H14,D15,B15,B2	9	VDD	digital VDD	3.3V IO voltage
8,28,58,66,95,101,110,113,159	E3,K1,P7,P9, J12,H13,F13, E12,A1	9	GND	digital GND	
133,135,139,148	D10,C10,C9, C6	4	UVDD	USB VDD	3.3V IO voltage
134,136,140,141,146,147	A10,B10,A8,B9,B7,D6	6	UGND	USB GND	
118	E14	1	PVDD	PLL VDD	1.8V PLL voltage
117	D14	1	PGND	PLL GND	

Table 1 - Pin Description

Note 1: Type key – (w_) x (y)(_z) – All IO are CMOS levels

w	Tolerance
T	5V tolerant
	3.3V

X	Direction
I	input
O	output
B	Bi-directional

y	Pull-up/down
U	pull-up
D	pull-down
	none

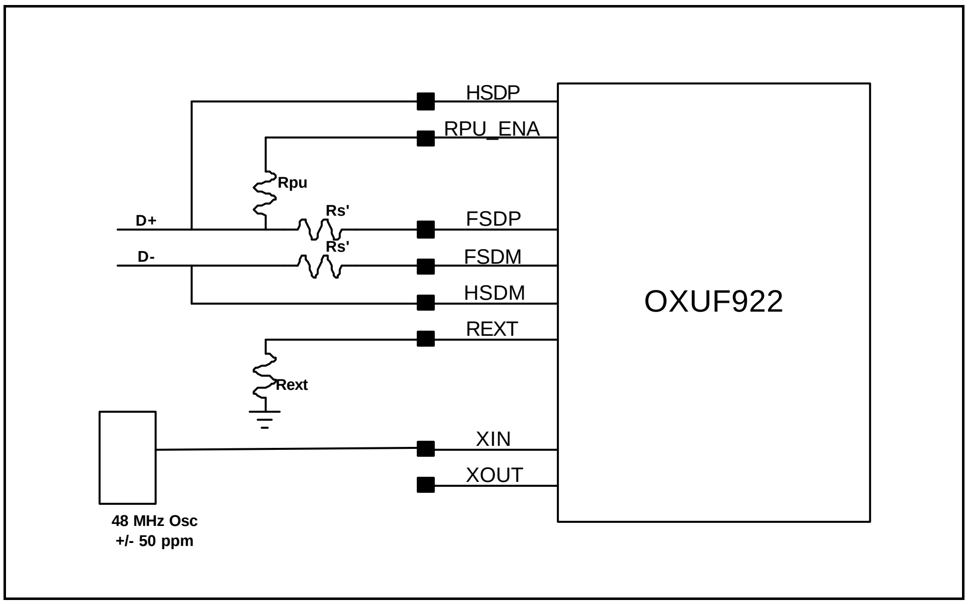
z	Output drive capability
4	4 mA drive
8	8 mA drive

USB_O	Dedicated USB output
USB_B	Dedicated USB bi-directional

CVDD	1.8V Digital Core Power
CGND	Core Ground
VDD	3.3V IO Digital Power
GND	IO Ground
PVDD	1.8V Digital Core Power
PGND	PLL Ground
UVDD	3.3V Analogue Power
UGND	Analogue Ground

Note 2: Power and ground
Separate supplies are recommended for the digital and analogue power supplies.

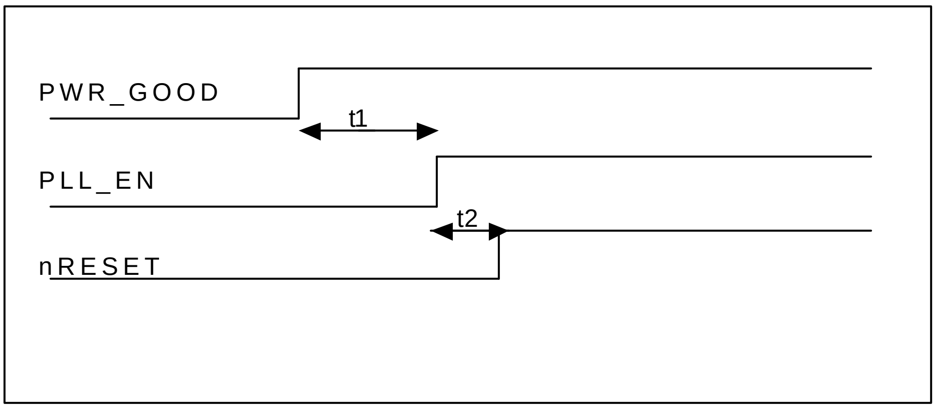
3.1 USB Pin Configuration Requirement



(Note: see Table 1 - Pin Description for resistor values)

3.2 Power on / reset sequence

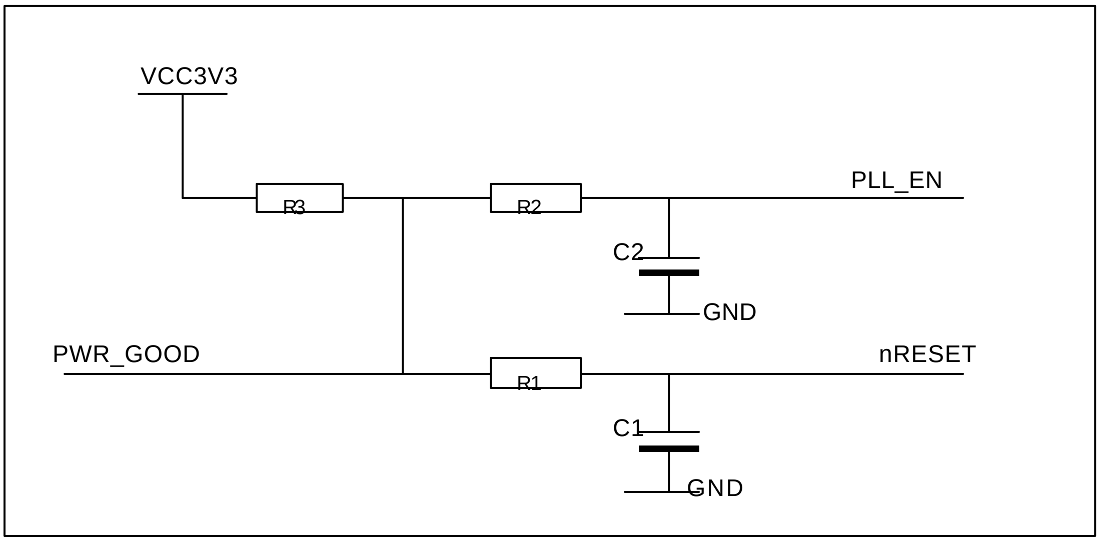
The OXUF922 Requires reset timing as follows:



t1: The oscillator needs time to stabilise before the OXUF922's PLL is enabled (by pulling high PLL_EN). t1 is the time required from power being stable (within limits) for the oscillator to stabilize.

t2: >2mS. This is the minimum delay required for the PLL to lock. The PLL must be locked prior to taking the OXUF922 out of reset.

The OXUF922 PLL Enable and Reset inputs have Schmitt type inputs to allow the use of RC delay elements as shown in the example below.



In calculating values for R1, R2, R3, C1 and C2, 'PWR_GOOD' is assumed to be an open drain output from the local power conditioning circuitry and that it can sink at least 1mA with an output low voltage of 0.3v thereby setting R3 at 3K3 (as nearest convenient value).

For this example let the requirement for T1 be 40mS (the actual requirement will depend on the oscillator used which is application dependant.)

The lowest value of input high voltage for the OXUF922 Schmitt inputs is 0.9 Volts (Min Vil + Min Hysteresis = 0.5 + 0.4 V).

Using R = T / (C * ln[(Vaim – Vstart)/(Vaim – Vth)])

Calculate value for R2 given C2 = 1uF
Set C = 1uF
T = 50mS

Vaim = 3.3 V	(Supply voltage)
Vstart = 0.3 V	(PWR_GOOD Output Low)
Vth = 0.9 V	(Input high voltage)

Gives R = 224K therefore let R2 = 220K as nearest standard value.

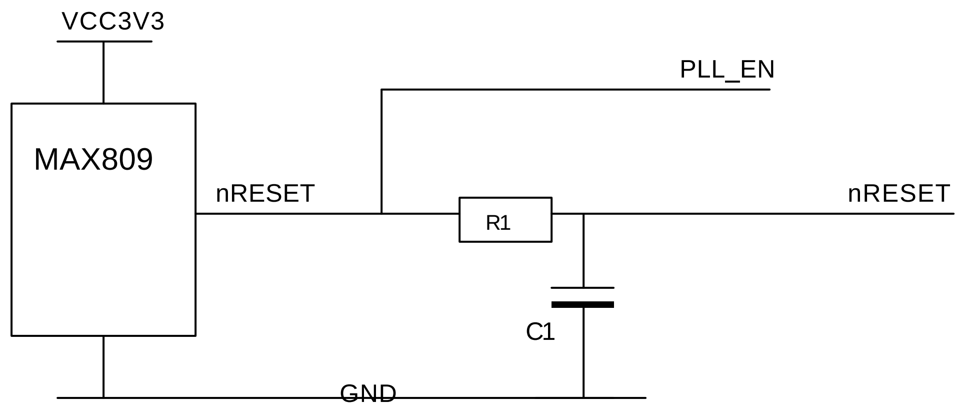
When tolerances, thresholds and supply range are accounted for, C = +20%, R = +5%, Vaim = 3.0, Vth = 2.3
T becomes 374 ms

Now calculate values for R1 given C1 = 1uF.
Set C = 1uF –20%
T = 52mS

Vaim = 3.3 V	(Supply voltage)
Vstart = 0.3 V	(PWR_GOOD Output Low)
Vth = 0.9 V	(Input high voltage)

Gives R = 291.3K therefore let R1 = 330K to use standard values and account for 5% tolerance

The 3K3 pull-up will extend these times slightly but is insignificant compared to capacitor tolerance.



R1 = 69K ohms C1 = 330nF makes t2 ~= 4.5mS.

C1 is determined from the minimum Vih threshold of 0.9V, a starting Vil of 0.3v. By making R1 large enough not to load the output say 69K. This leads to a value of 240nF. This has been rounded up to 330nF as the nearest standard value and provides some margin.

The MAX809 provides a reset low pulse of 140mS, easily accommodating the requirements for t1.

Why use the MAX809?

A reset controller is defined for this circuit because not using one requires knowledge of the end user systems power supply characteristics, specifically the supply rise time.

3.2.1 Lockup Hazard

If an OXUF922 /PHY combination is powered up attached to a 1394 bus, there is a potential hazard where the OXUF922 misses the node ID from the PHY and hence fails to operate correctly. This is only an issue when trying to access the flash port when no firmware is loaded. When firmware is loaded one of the first things it does is cause a bus reset which sorts everything out.

Avoidance action: Ensure the OXUF922 comes out of reset before the PHY.

3.3 Functional Modes

Test Mode[1]	Description
000	Normal mode – 8 or 16 bit data bus, 128Kbyte address range
001	JTAG bond out on upper data bus, 8 bit data bus only, 1Mbyte address range
010	1394 Only, no USB clock source required
011	Reserved
100	Reserved
101	Extended address on upper data bus, 8 bit data bus only, 4 MByte address range
110	Reserved
111	Reserved

Notes
1. Test Mode = TEST [2:0] pins

3.4 External Devices Required

8/16-bit Flash
48 MHz $\pm$50 ppm crystal oscillator for USB (and system clock).
1394 PHY (A or B) and corresponding crystal.

4 MEMORY ORGANISATION

4.1 Memory Organisation

Block	CPU Base Address	Decode Size (Bytes)
External Chip Selects	0x00000000	4 x 4M
Repeat of above	0x01000000	4 x 4M
Repeat of above	0x02000000	4 x 4M
Repeat of above	0x03000000	4 x 4M
Unused	0x04000000	
Unused	0x05000000	
FIFO Manager	0x06000000	n/a
UART	0x07000000	
Unused	0x08000000	
Unused	0x09000000	
Logic Registers	0x0A000000	n/a
Logic Registers (Repeat)	0x0B000000	n/a
Unused	0x0C000000	
Unused	0x0D000000	
Static If	0x0E000000	n/a
Unused	0x0F000000	
SCRATCH RAM Base address	0x80002000	8K

4.2 Register Set and Base Addresses of Hardware Devices

Block	Base Address (31:16)	Location
Chip Select 0 address (External Flash)	0x0000	See section 5.2 Static RAM Controller
Chip Select 1	0x0040	See section 5.2 Static RAM Controller
Chip Select 2	0x0080	See section 5.2 Static RAM Controller
not used	0x00C0	
UART	0x0700	See section 6 UART
RPS	0x0A00	See section 5.3 RPS Block
Link	0x0A10	See section 7.4 Link-Core
Async Engine	0x0A20	See section 7.8 Async Engine
Queue Selector	0x0A28	See section 7.7 Queue Selector
USB	0x0A30	See section 7.9 USB2 core
FIFOMAN	0x0A40	See section 7.5 FIFO Manager.
not used	0x0A50	
DMA	0x0A60	See section 7.1 DMA Engine
ATA	0x0A70	See section 7.2 ATA Block
OCP	0x0A80	See section 7.6 ORB Co-Processor.
not used	0x0A90	
serial control	0x0AA0	See section 7.10 Serial Controller
clock control	0x0AB0	See section 5.1 Clock Block
Serial Audio	0x0AC0	See section 7.11 Serial Audio
not used	0x0AD0 – 0AF0	
Static Registers	0x0E00	See section 5.2 Static RAM Controller

Note : register set repeats, only bits 27:24 decoded

5 CPU FUNCTIONS

5.1 Clock Block

The clock block contains all the clock generation and gating logic required. There are four registers to allow the control of the serial clock, the clock to the USB PHY for suspend and power saving control and the stop and start controls for the clocks to the majority of the logic. In addition to stopping clocks it is also possible to slow all the logic clocks, except the USB PHY clock to half speed. This is done via bit 30 of the start / stop registers.

The table below details which clocks run at what speed, which are controllable and their state after reset.
The normal application of the device would have the 48 MHz source being used as the system clock and USB clock, and a separate crystal being used for the 1394 PHY. There is an additional mode where the 1394 clock can be used to replace the 48 MHz, but this then precludes the use of USB. See section 3.3 Functional Modes for details on pin assignment for this.

Block	Frequency (MHz)	Clock Stop	Reset Value
Fifoman – CPU bus	50	Yes	Stopped
Fifoman – DMA bus	100	Yes	Stopped
OCP – AHB master	50	Yes	Stopped
OCP – function	100	Yes	Stopped
Static	50	Yes	Running
Serial	50	Yes	Stopped
UART – AHB	50	Yes	Stopped
UART – sys_clk	50	Yes	Stopped
Link / Async / Queue Sel	100	Yes	Running
USB	100	Yes	Running
DMA / ATA / DMA AHB subsys	100	Yes	Stopped
Serial Audio	100	Yes	Stopped

5.1.1 Clock Control Registers

Clock control registers can be found starting at location 0AB00000

Register	Offset	Reset value	Description
Serial Clk Control	0	0x020	Divider value to generate serial clock
UTMI Clk Control	4	0x01	UTMI PHY clock enable (1=enabled)
Clock Stop	8	0x01DC	Write to stop various clocks
Clock Start	C	0x01DC	Write to start various clocks

5.1.2 Serial Clk Register

Set one bit to select serial clk frequency when using serial if controller

Bit	Serial frequency (MHz)	Reset read value
0	100 / 8	0
1	100 / 16	0
2	100 / 32	0
3	100 / 64	0
4	100 / 128	0
5	100 / 256	1
6	100 / 512	0
7	100 / 1024	0
31-8	reserved	0

5.1.3 UTMI Clk Register

Writing to this register will enable / disable the UTMI PHY's clock. This can be used in conjunction with the suspend indication in the USB core to put the USB PHY into power save mode.

Bit	UTMI Clk enable	Reset read value
0	1 to enable clk, 0 to disable	1
31-1	reserved	0

5.1.4 Clock Stop Register

Write 1 to stop the clocks (bits 0 to 9)
Write 1 to slow clocks below to half rate (bit 30)

Clock Stop bit	Clock(s)	Reset read value
0	Link system clock Async clock Queue sel clock	0 = started, 1 = stopped
1	USB system clock	0
2	FM system clock FM AHB CPU clock	1
3	OCP system clock OCP AHB CPU clock	1
4	DMA system clock ATA system clock	1
5	Static system clock	0
6	Serial system clock	1
7	Serial Audio system clock	1
8	UART system clock UART AHB clock	1
29:9	Not defined	0
30	Slow clock select	0 = normal, 1= slow
31	Reserved	0

5.1.5 Clock Start Register

Write 1 to start the clocks (bits 0 to 9)
Write 1 to bit 30 to return clocks below to full rate

Clock Start bit	Clock(s)	Reset / read value
0	Link system clock Async clock Queue sel clock	0 = started, 1 = stopped
1	USB system clock	0
2	FM system clock FM AHB CPU clock	1
3	OCP system clock OCP AHB CPU clock	1
4	DMA system clock ATA system clock	1
5	Static system clock	0
6	Serial system clock	1
7	Serial Audio system clock	1
8	UART system clock UART AHB clock	1
29:9	Not defined	0
30	Slow clock select	0 = normal, 1= slow
31	reserved	0

5.2 Static RAM Controller

The controller supports up to three chip select banks with independent timing control. This allows the external connection of flash and peripherals to the OXUF922, so the chip can take on the responsibility of application host.

- Three separate chip select banks (CS0, CS1 & CS2).
- 8bit or 16bit memory support (pin select for cs0 boot, register select for cs1 & cs2)
- Bank size up to 4MBytes (A21:0) when the chip is in 8 bit mode
- Bank size up to 1MBytes (A19:0) when the chip is in 8 bit mode with JTAG debug
- Bank size up to 128KBytes(A16:1) when the chip is in 16 bit mode
- Programmable burst fetch.
- Supports external 245 buffer ICs for glue-less data bus expansion.
- Configurable pin for an interrupt input from a static peripheral or 50 MHz output for use by a synchronous peripheral on the static bus.
- Second write enable pin can be selected to be used for an upper byte write enable for 16 bit SRAM on the static bus.

The way each chip select bank is accessed and controlled is via its own register. The register for each bank (address space) controls the following:-

- Whether it is an 8 bit or 16 bit bus. Note that when in 8 bit mode the bank size is 4Mbytes, when in 16 bit mode it is 128Kbytes.
- A delay of up to 3 clocks can be added to /WE strobing low after /CS strobes low in a write cycle.
- The number of clocks into a write cycle that /WE goes high can be controlled.
- The number of clocks /CS is low in a write cycle (from 1 up to 64).
- The number of clocks /CS is low in a read cycle (from 1 up to 64).
- /OE can be delayed by 1 Clk after /CS strobes low in a read cycle.

Note: Setting the test pins may constrain the data width of connected memory devices, regardless of the register settings.

5.2.1 Register Set

All addresses are stated as an offset from the static ram controller register set base address, 0x0E000000.

Address offset from STATIC_REG_BASE	R/W	Reset Value	Register
0x00	R	0x00000001	Version ID register
0x04	R/W	0xXFFFFFFF	Static Bank 0 register (Boot Bank)
0x08	R/W	0x0FFFFFFF	Static Bank 1 register
0x0C	R/W	0x0FFFFFFF	Static Bank 2 register
0x10			Not used

Note X values in Boot bank reset. See below.

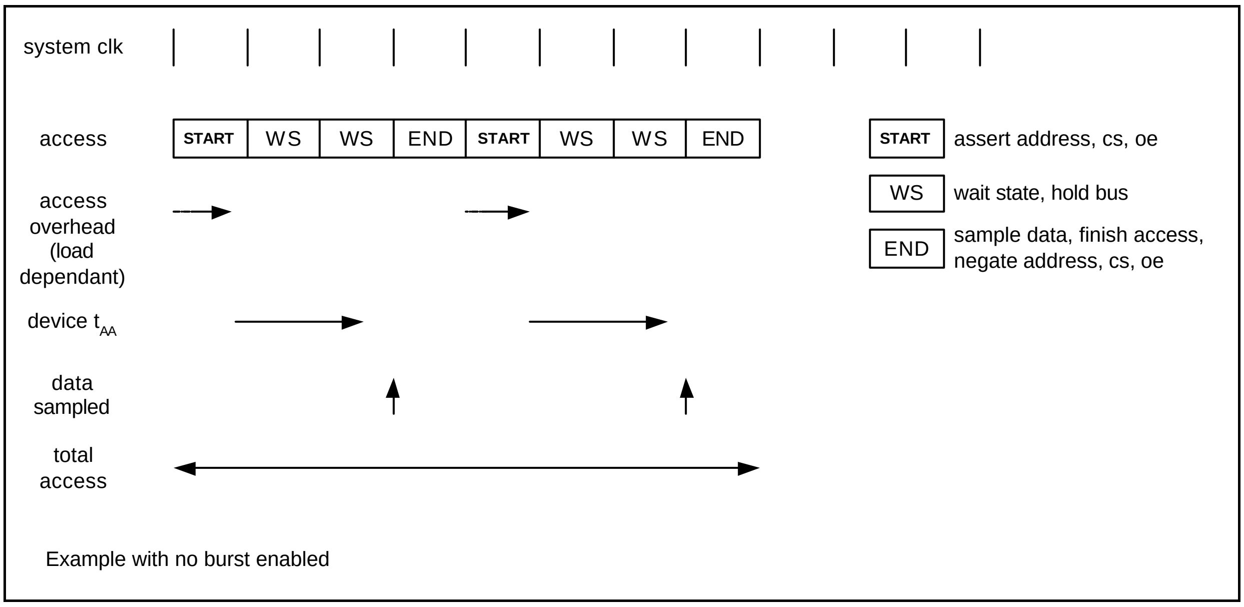
5.2.2 Static Bank registers 0 to 3

Bits	Dir	Name	Reset	Function
31:30	R/W	Width	00*	External memory width "00" 8bit, "01" 16bit, "10" 32bit
29	R/W	Read Burst Enable	0	Enable read bursts speeds access by not raising /CS between accesses
28	R/W	Buffer Present	0	External 245 buffer required for this bank
27:26		write_start	11	The number of clocks into the write cycle that the /WE strobes fall.
25:24	R/W	turn_cycle	11	Turn around time in CPU clocks. /CS high time.
23:22		Reserved	11	
21:16	R/W	write_pulse	111111	The number of clocks into the write cycle that the /WE strobes rise.
15:14		Reserved	11	
13:8		write_cycle	111111	/CS is low for write_cycle+1 for a write access
7	R/W	Delay OE	1	Specifies whether OE should be delayed 1 clock from CS
6		Reserved	1	
5:0	R/W	read_cycle	111111	/CS is low for read_cycle+1 for a read access.

The reset value is such that the memory works in the slowest possible mode.
*The Width configuration for the boot block is set by pin 79 **BUS_SEL**, so that the boot block can be accessed correctly from reset. This should be set in conjunction with the test pins.
Section 12 External Bus Timing Diagrams gives examples of various configurations.

5.2.3 Example Bus Timing with Wait States

Assuming an 8-bit Flash device on CS(0), the following diagram and tables gives example values to set in the bank registers depending on the access time of the device (t_{AA}).


system clk
access
START
WS
WS
END
START
WS
WS
END
access overhead (load dependant)
device t_{AA}
data sampled
total access
START assert address, cs, oe
WS wait state, hold bus
END sample data, finish access, negate address, cs, oe
Example with no burst enabled

48 MHz system clock (USB and/or 1394)

			Total Access Time (ns)			
	Max t_{AA} (ns) [3]		32bit Read (ARM)		16bit Read (Thumb)	
Read_cycle[1]	15pF load [2]	25pF load [2]	No Burst [4]	Burst [5]	No Burst	Burst
0	4.6	3.6	167	104	83	63
1	25.4	24.4	250	188	125	104
2	46.3	45.3	333	271	167	146
3	67.1	66.1	417	354	208	188
4	87.9	86.9	500	438	250	229

49.152 MHz system clock (1394 only)

			Total Access Time (ns)			
	Max t_{AA} (ns) [3]		32bit Read (ARM)		16bit Read (Thumb)	
Read_cycle[1]	15pF load [2]	25pF load [2]	No Burst [4]	Burst[5]	No Burst	Burst
0	4.1	3.1	163	102	81	61
1	24.4	23.4	244	183	122	102
2	44.8	43.8	326	264	163	142
3	65.1	64.1	407	346	203	183
4	85.5	84.5	488	427	244	224

Notes:
1. read_cycle as defined by bits (5:0) in bank register
2. Load seen by address, cs (chip select) and oe (output enable). Affects access overhead. (16.25ns for 15pF, 17.25ns for 25pF)
3. Max t_{AA} is calculated from ((1 + read_cycle) * system clk period) – access overhead.
4. Total access time = (1 + read_cycle + 1) * system clk period * number of required memory accesses
5. Total access time = ((1 + read_cycle) * system clk period * number of required memory accesses) + 1 * system clk period

5.2.4 General Address Decoding

The following address decodes are used by this block. Note the processor must have the boot code at address 0x00000000.

ARM address offset from STATIC_BASE	Block	Function
0x00000000	External FLASH chip select (0)	External Flash
0x00400000	External chip select (1)	SRAM
0x00800000	External chip select (2)	LED Bank

5.2.5 Additional Functions

To provide a means to directly interface to an external device running synchronously to the OXUF922, there is the facility to output the CPU clock. This is controlled from the GPIO data register detailed below. Timing diagrams showing the timing relationship is shown in section 12 External Bus Timing Diagrams.

Additionally when interfacing to a 16bit SRAM a second write enable maybe required for byte write accesses. This function is also provided through the GPIO register below.

GPIO data register - location 0x0A0003C0 – note this will also control the direction change of the pin.

GPIO data bit	Write	Read	Reset value
17	Set to 1 to mux static_wen(1) onto GPIO(2) pin for 16 bit SRAM writes	Register write value	0
18	Set to 1 to output static_clk (50 MHz) on pin 38 (static_int)	Register write value	0

5.3 RPS Block

The ARM Reference Peripheral Set RPS provides a basic set of standard APB peripherals. The reference peripheral set (or RPS) is intended to provide useful common functionality that is required in many embedded systems. This block implements counter timer, interrupt controller and addre ss map controller. These peripherals are fully defined in ARM's RPS specification. In addition to these, this RPS includes the following extra system peripherals:

- A watchdog timer for providing a system reset should software hang, or if no software is present
- A GPIO controller to provide flexible digital IO
- A Block reset controller which allows individual design sub-blocks to be reset independently of one another under software control

5.3.1 Block descriptions

Brief description of the blocks is given below with more detailed descriptions following

Interrupt Controller (see section 5.4)
The interrupt controller provides a means by which the processors enable, disable, set and clear the various interrupts in the system. The actual in terrupts seen by the processor are routed from this block. This block is also fully specified in the ARM RPS specification.

Watchdog Timer (see section 5.6)
This block provides a counter which is continually counting (incremen ting every CPU clock cycle). The count is reset every time its register is read (returning the count immediately prior to the read), and a timeout output goes active if the terminal count is ever reached. This timeout may be used to reset the system processor. This provides a useful facility for software crash protection.

In addition, the watchdog has two timeout modes, Fast (default) and slow. In fast timeout mode, the terminal count is 16,384 clock cycles (327uS with a 50MHz clock). In slow timeout mode (which may be entered by writing a special value to the watchdog register), the timeout is a longer programmable period. (This is defined in more detail in the register block description).

The watchdog switch on timeout output is a programmable output (reset to ACTIVE) for interfacing to a "Force Flash" controller. The intended use is to cause a transition to force flash mode if a watchdog timeout occurs while this bit is set.

Programmable Counters (see section 5.7)
These are programmable 16-bit down counting interval (re-loading) or timeout (one-shot) counters used for generating timed interrupts to the processor. These timers are fully specified in the ARM RPS specification.

GPIO (see section 5.8)
This block provides user I/O. GPIOs can be programmed as inputs or outputs. Additionally, inputs can be set to cause an interrupt upon a transition. Each GPIO has its own direction control (output enable), so open drain outputs can be emulated easily by setting the output value to zero, and writing the inverse of the data to the OE bit instead of the output.

Block Reset Controller (see section 5.9)
This block simply defines a writable register which generates several reset outputs which can be routed to the various sub-blocks in the system, in order to provide more flexible resetting. All reset outputs are activated by the global system reset input, or by setting the appropriate bit in the reset control register. In this case the bit is self clearing, and the reset is activated for 4 CPU clock cycles.

5.4 ARM RPS Register Set Summary

IRQ Interrupt Control

BASE	READ	WRITE
0x0000	Masked IRQ Source Status	*none*
0x0004	Raw IRQ Source Status	*none*
0x0008	IRQ Enable Mask	IRQ Enable bits
0x000C	*none*	IRQ Disable bits
0x0010	*none*	Software Interrupt

FIQ Interrupt Control

BASE	READ	WRITE
0x0104	Masked FIQ Source Status	*none*
0x0108	Raw FIQ Source Status	*none*
0x010C	FIQ Enable Mask	FIQ Enable bits
0x0110	*none*	FIQ Disable bits

Timer One

BASE	READ	WRITE
0x0200	Timer 1 Load	
0x0204	Timer 1 Current Count	none
0x0208	Timer 1 Control	
0x020C	none	Timer 1 Clear
0x0210	none	none

Timer Two

BASE	READ	WRITE
0x0220	Timer 2 Load	
0x0224	Timer 2 Current Count	*none*
0x0228	Timer 2 Control	
0x022C	*none*	Timer 2 Clear
0x0230	*none*	*none*

Reset & Remap

BASE	READ	WRITE
0x0300	*none*	Pause
0x0304	Identification	*none*
0x0308	*none*	Clear Reset Map
0x030C	Reset Status	Reset Status Set
0x0310	*none*	Reset Status Clear

Note: Reset & Remap registers are NOT implemented in the OXUF922.

5.5 Interrupt Operation

The interrupt controller provides a means by which the processor can enable, disable, set and clear the various interrupts in the system. The actual interrupts seen by the processor are routed from this block. This block is fully specified in the ARM RPS specification.

5.5.1 Interrupt Bit Mapping

The interrupt registers can be found starting at location 0A000000. The table below shows the OXUF922 interrupt 32-bit assignment for the Interrupt Controller.

Interrupt bit	Write / Read	Reset value
0	FIQ – SERIAL AUDIO	0
1	Software Irq (SWI)	0
2	Reserved	0
3	Reserved	0
4	Timer 0	0
5	Timer 1	0
6	ATA	0
7	DMA	0
8	Link	0
9	UART	0
10	Async	0
11	FIFOMAN	0
12	Reserved	0
13	OCP	0
14	USBCORE	0
15	Static	0
31	GPIO interrupt	0
16:30	Reserved	0

5.6 Watchdog Timer

This block provides a counter which is continually counting (incrementing every CPU clock cycle). The count is reset every time its register is read (returning the count immediately prior to the read) under normal operation the watchdog timer must be cleared at a regular interval. If this is not done a timeout output goes active if the terminal count is ever reached. This timeout may be used to reset the system processor. This provides a useful facility for software crash protection providing a system reset should software hang.

In addition, the watchdog has two timeout modes, Fast (default) and slow. In fast timeout mode, the terminal count is 16,384 clock cycles (327uS with a 50MHz clock). In slow timeout mode (which may be entered by writing a special value to the watchdog register), the timeout is a longer programmable period. (This is defined in more detail in the register block description below).

The watchdog switch on timeout output is a programmable output (reset to ACTIVE) for interfacing to a "Force Flash" controller. The intended use is to cause a transition to force flash mode if a watchdog timeout occurs while this bit is set.

5.6.1 Watchdog timer - location 0x0A000380

Offset **Watchdog Control Register**

0x0380	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD	Number of APB clock cycles sincle last read (or reset)																															
WR1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	0	0	0	1	F	S
WR2	0	0	1	0	0	0	0	0	1	1	0	1	0	Timeout			0	1	0	1	1	1	1	1	1	0	0	0	1	0	0	1

Reading the watchdog returns the number of CPU clock cycles since the last time it was read (or reset) and then resets the count. The count is also reset when a timeout occurs, or the ARM reset input signal is asserted. Writing to this register with values from 0x1068AFC5 to 0x1068AFC7 allows the F and S bits to be set (according to the bottom two bits of the value written).

Switch on timeout bit (S) is intended for use in conjunction with an external "Force flash" controller, allowing it to enter force flash mode upon a watchdog timeout when this bit is set. This bit is set by default.

Fast Timeout bit (F) is used to cause watchdog timeout after just 16,384 clock cycles. This is the default state, hence software should clear this bit and the S bit as soon as possible after booting (By writing **0x1068AFC4)**

5.6.2 WDT PERIOD

Writing this register with 0x20D15F89 OR'ed with (timeout << 16) allows setting of the long timeout value (F = '0'). The timeouts defined are given in the following table:

F Bit	Timeout	Clocks to timeout	Period – 50MHz clock
1	x	16 x 1024	327 us
0	0	4096 x 1024	84 ms
0	1 (default)	8192 x 1024	168 ms
0	2	16384 x 1024	336 ms
0	3	32768 x 1024	671 ms
0	4	65536 x 1024	1.34 s
0	5-7	Reserved	Undefined

The watchdog may be completely **disabled** by writing the value **0x20D15F88**. It should be noted that once the watchdog has been disabled the counter is stopped. Writing with the timeout value specified above will re-enable the watchdog.

5.7 Counter Timers

5.7.1 Overview

The OXUF922 has two timer modules which can be either a programmable 16-bit down counting interval (re-loading) or a timeout (one-shot) counter used for generating timed interrupts to the processor. Each channel can be independently programmed to perform a wide range of functions, including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse-width modulation.

Each timer is a 16 bit wide down counter with selectable pre-scalar. The pre-scalar allows either the CPU clock (50 MHz) to be used directly, or the clock divided by 16 or 256 may be used. This is provided by 0, 4 or 8 stages of pre-scale.

Two modes of operation are available, free-running and periodic timer. In periodic timer mode the counter will generate an interrupt at a constant interval. In free-running mode the timer will overflow after reaching its zero value and continue to count down from the maximum value.

5.7.2 Timer Operation

Each of the timers is loaded by writing a value to there respective *load register* and then, if enabled by setting the enable bit in the timer *control register* , the timer will count down to zero. On reaching a count of zero an interrupt will be generated. The interrupt may be cleared by writing to the respective Timer *clear register*.

After reaching a zero count, if the timer is operating in free-running mode then the timer will continue to decrement from its maximum value. If periodic timer mode is selected then the timer will reload from the *load register* and continue to decrement. In this mode the timer will effectively generate a periodic interrupt. The mode is selected by a bit in the *Control register*. At any point the current timer value may be read from the *Value (Current count) register.* The timer is enabled by a bit in the *control register*.

At reset the timer will be disabled, the interrupt will be cleared and the *Load register* will be undefined. The mode and pre-scale value will also be undefined.

5.8 GPIO Block

5.8.1 Overview

The OXUF922 has 8 user I/O pins provided by the GPIO block. These pins can be programmed as either inputs or outputs. Additionally, when set-up as inputs they can be set to cause an interrupt upon a transition. Each GPIO pin has its own direction control (output enable), so open drain outputs can be emulated easily by setting the output value to zero, and writing the inverse of the data to the OE (output enable) bit instead of the output.

The interrupt enable mask allows input transition interrupts to be generated. Setting an interrupt mask bit for a given input will cause an interrupt event to be flagged in the *interrupt event register* every time the state of that input changes. Setting an interrupt mask bit for a GPIO which is defined as an output (in the [OE] output enable register) will have no affect.

Interrupt events can be read from the *Input Interrupt events register* and individual inputs events can be cleared by writing to this register with the appropriate bit or bits set to clear the event.

The GPIO internal interrupt will be set whenever the *Input Interrupt Events register* is non-zero (i.e. any events are set). This interrupt can be read from the IRQ interrupt controller in the same way as any other interrupt, and must be enabled as such in order to generate an actual interrupt to the CPU.

GPIO registers can be found starting at location 0x0A0003C0

Register	Offset	Reset value	Description
GPIO data	0	#	This allows the setting and reading of the GPIO pins. Additional functions related to GPIO are controlled here.
GPIO output enable	4	0	GPIO direction setting
GPIO Interrupt enable mask	8	0	This allows the GPIO inputs to be used as interrupts, by enabling the ones required.
GPIO Interrupt event	C	0	Interrupt status and clear register

5.8.2 GPIO Data Register - location 0x0A0003C0

GPIO data bit	Write	Read	Reset value
7:0	Output value	Pin value	App specific
10:8	Pin multiplexing control - see below section 5.8.6	Pin mux setting	000
11	N/A	A_B_PHY_SEL pin	App specific
12	N/A	USB_IND pin	App specific
13	N/A	BUS_SEL pin	App specific
14	N/A	TEST0 pin	App specific
15	N/A	TEST1 pin	App specific
16	N/A	TEST2 pin	App specific
17	mux static_wen(1) onto GPIO(2) pin for 16 bit SRAM writes	Register write value	0
18	Output static_clk (50 MHz) on pin 38 (static_int)	Register write value	0
31:19	Reserved	0	0

5.8.3 GPIO Output Enable register - location 0x0A0003C4

This register determines whether the GPIO pin will be an input or an output dependent on what is written to it. For example
Write value to register

GPIO oen bit	Write	Read	Reset value
7:0	output enable (oen), 0 = input, 1 = output	reg write value	0

5.8.4 GPIO Interrupt Enable (mask) register - location 0x0A0003C8

The value written to this register determines whether the corresponding input will generate an interrupt when a transition occurs on the input. Setting an interrupt mask bit for a given input will cause an interrupt event to be flagged in the *interrupt event register* every time the state of that input changes.

As can be seen it is important that the value used here corresponds to the value loaded into the GPIO Output Enable register.

GPIO interrupt enable mask bit	Write	Read	Reset value
7:0	interrupt enable (int en), 0 = no interrupt, 1= enable interrupt	reg write value	0

5.8.5 GPIO interrupt event register - location 0x0A0003CC

This register can be used to be read from to determine which input has generated an interrupt. Conversely this register can then be written to wih the appropriate bit or bits set to clear the interrupt(s).

GPIO interrupt mask bit	Write	Read	Reset value
7:0	0 = no effect, 1 = clear interrupt	0 = no interrupt, 1= interrupt event	0

5.8.6 GPIO Pin Multiplexing

By setting gpio_data bits 17, 10, 9, 8 the GPIO pins can be reconfigured to use other functions as detailed below. Note only GPIO (3) is always a GPIO pin.

GPIO data(10:8)	GPIO(7)	GPIO(6)	GPIO(5)	GPIO(4)	GPIO(3)	GPIO(2)	GPIO(1)	GPIO(0)
000	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
001	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	serial_data	serial_clk
010	Serial Audio_word	Serial Audio_data	Serial Audio_clk	GPIO	GPIO	GPIO	GPIO	GPIO
011	Serial Audio_word	Serial Audio_data	Serial Audio_clk	GPIO	GPIO	GPIO	serial_data	serial_clk
100	DTR	DSR	DCD	RI	GPIO	GPIO	GPIO	GPIO
101	DTR	DSR	DCD	RI	GPIO	GPIO	serial_data	serial_clk
110	DTR	DSR	DCD	RI	GPIO	GPIO	GPIO	GPIO
111	DTR	DSR	DCD	RI	GPIO	GPIO	serial_data	serial_clk

5.9 Reset Control

5.9.1 Overview

The reset strategy is divided into three sections, firstly synchronisation of the system reset, secondly reset control of the ARM, and thirdly control of the resets to each major block of logic.

The system reset is asynchronous and is synchronised to the internal 50 MHz clock before being distributed around the chip. When in force flash mode, setting the device reset bit in the force flash port will generate a sync reset, causing the chip to come out of force flash mode and negating the reset.

The negation of the sync reset enables a 20 us counter which times out and brings the ARM out of reset. This ensures that all system interfaces are stable prior to the start of code execution. The ARM can also be reset by a watchdog timeout or exiting force flash mode, both events also kick off the 20 us timer.

The sync reset is used by the block reset controller which allows software to control the resets the major logic blocks as detailed below. This function is found in the RPS block.

5.9.2 Block Reset Mapping - location 0A000340

The block reset register can be found starting at location 0A000340. Writing a 1 to the relevant bit generates a 3 cycle reset pulse at 50 MHz.

Block reset bit	Write / Read	Reset value
0	ATA	0
1	DMA	0
2	Reserved	0
3	Static	0
4	Link core	0
5	Async_engine	0
6	queue selector	0
7	FIFOMAN	0
8	Reserved	0
9	OCP	0
10	USB core	0
11	Serial IF	0
12	UART	0
13	Serial Audio	0
31:14	Reserved	0

6 UART

The UART operates in synchronous mode only clocked by the 50 MHz clock. Note: exact frequency will be either 48 MHz or 49.152 MHz depending on configuration. The UART is based on a 950 UART with receive and transmit FIFO's of 128x8 deep. The internal UART is located at address 0x07000000. The following pins are permanently available on the chip I/O, SIN, SOUT, CTS and RTS. For information on the UART refer to Oxford products incorporating UART's.

Indexed Control Register Set

Register	Offset from ICR
acr	0x00
cpr	0x01
tcr	0x02
cks	0x03
ttl	0x04
rtl	0x05
fcl	0x06
fch	0x07
Id1	0x08
Id2	0x09
Id3	0x0A
rev	0x0B
csr	0x0C
nmr	0x0d
mdm	0x0e
rfc	0x0f
gds	0x10
dms	0x11
pidx	0x12
cka	0x13

The SPR offset column indicates the value that must be written into SPR prior to reading / writing any of the Indexed Control Registers via ICR.

UART 550 Registers

Register	Address Offset
thr	0x00
ier	0x01
fcr	0x02
lcr	0x03
mcr	0x04
lsr	0x05
msr	0x06
spr	0x07

UART 650 Registers

Register	Address Offset
efr	0x02
Xon1	0x04
Xon2	0x05
Xoff1	0x06
Xoff2	0x07

UART 950 Registers

Register	Address Offset
asr	0x00
rfl	0x03
tfl	0x04
icr	0x05

If hardware flow control is required then there are GPIO pins available to do this. (DTR, DSR, DCD and RI)

GPIO data(10:8)	GPIO(7)	GPIO(6)	GPIO(5)	GPIO(4)	GPIO(3)	GPIO(2)	GPIO(1)	GPIO(0)
000	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
001	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	serial_data	serial_clk
010	Serial Audio_word	Serial Audio_data	Serial Audio_clk	GPIO	GPIO	GPIO	GPIO	GPIO
011	Serial Audio_word	Serial Audio_data	Serial Audio_clk	GPIO	GPIO	GPIO	serial_data	serial_clk
100	**DTR**	**DSR**	**DCD**	**RI**	GPIO	GPIO	GPIO	GPIO
101	**DTR**	**DSR**	**DCD**	**RI**	GPIO	GPIO	serial_data	serial_clk
110	**DTR**	**DSR**	**DCD**	**RI**	GPIO	GPIO	GPIO	GPIO
111	**DTR**	**DSR**	**DCD**	**RI**	GPIO	GPIO	serial_data	serial_clk

7 INTERFACE & PERIPHERAL FUNCTIONS

7.1 DMA Engine

The OXUF922 includes a single-channel high speed DMA controller supporting the requesting DMA sources. The DMA controller is capable of transferring 8, 16 and 32-bit data between FIFO manager and ATA blocks.

DMA registers can be found starting at location 0x0A600000

Register	Offset	Reset value	Description
control and status Register	00		See below
base source address register	04		Holds the base-Source address for DMA transfers
base destination address register	08		Stores the current Source address of the DMA transfer. This value is transfer-width aligned and an increment by the number of bytes transferred per cycle or it is fixed.
byte count register	0C		Holds the Destination-base address for DMA transfers
current source address register	10		Stores the current destination address of the DMA transfer. This value is transfer-width aligned and an increment by the number of bytes transferred per cycle or it is fixed.
current destination address register	14		See below
current byte count register	18		Indicates number of bytes read from the source at any point during the current DMA transaction. It is byte aligned and decrements on each cycle by the number of bytes transferred. Indicates number of bytes read from source.
interrupt register and DMA version	1C		See below

Control and status Register			
0	FAIR_SHARE_ARB	R/W	Indicates which Stage the channel belongs to. 1: Stage 1, higher group (reset value) 0: Stage 2, lower group
1	DMA_IN_PROGRESS	R	This bit indicates if a DMA channel is idle or not. 0: DMA idle. 1: DMA busy, transfer is not completed (current-byte-count not zero).
5,2	SFT_S_DREQ	R/W	CPU uses these bits to indicate the valid source request for the channel. DMA always reads from the source. 0000: req0 (reset value) 0001: req1 0010: req2 …. 1111: memory
9,6	SFT_D_DREQ	R/W	CPU uses these bits to indicate the valid destination request for the channel. DMA always writes to destination. 0000: req0 (reset value) 0001: req1 0010: req2 …. 1111: memory
10	INT	R	When set interrupt is enabled, indicates completion of a DMA transfer (when current byte count register is zero). This bit is cleared when CPU writes to BYTE_COUNT register for a new transfer or BASE_X_ADDR registers (if not expecting a new transfer).
11	NEXT_FREE	R	When high, this bit indicates that the address- and byte-count registers are ready to be programmed. DMA-controller will set this bit high after it loads the current registers. It deactivates when CPU writs to BYTE_COUNT register.
12	CH_RESET	R/W	Setting this bit to "1" will disable the next bus-cycle transfer and reset the channel. Minimum pulse Width High is 30ns (3x Hclk) .CPU needs to set it back to "0" when starting a new transfer.
14,13	DIRECTION	R/W	Indicates the data transfer direction between interfaces A and B. DMA-controller always reads from the source and writes to the destination. 00: Source = A, Destination = A (reset value) 01: Source = B, Destination = A 10: Source = A, Destination = B 11: Source = B, Destination = B
15	INC_ADDR_S	R/W	Determines whether the Source address will be fixed or incremented during a transfer. 0: Fixed address during the DMA transfer, only lowest 4-bits will increment (reset value). 1: Address will increment as current byte count increments.
16	INC_ADDR_D	R/W	Determines whether the Destination address will be fixed or incremented during a transfer. 0: Fixed address during the DMA transfer, only lowest 4-bits will increment (reset value) 1: Address will increment as current byte count increments.
17	MODE_A	R/W	Determines the data transfer mode for Interface A. 0: Single transfer mode (reset value) 1: Burst transfer mode.
18	MODE_B	R/W	Determines the data transfer mode for Interface B. 0: Single transfer mode (reset value) 1: Burst transfer mode.
21-19	DEVICE_TYPE_S	R/W	This determines the maximum data transfer width for source device.

			000: 8-bits Device (reset value) 001: 16-bits Device 010: 32-bits Device Others: Unused.
24-22	DEVICE_TYPE_D	R/W	This determines the maximum data transfer width for destination device. 000: 8-bits Device (reset value) 001: 16-bits Device 010: 32-bits Device Others: Unused.
25	PAUSE_DMA	R/W	Writing one to this bit will pause the next bus-cycle transfer, until it is set to zero.
26	INT_ENABLE	R/W	Setting this bit to "0" will disable the interrupt for this channel. Reset value is "1".
27	FIXED_ADDR_S	R/W	0: reset value, no effect, see bit 13. 1: Source address is really fixed (all bits); bit 13 must be '0'.
28	FIXED_ADDR_D	R/W	0: reset value, no effect, see bit 14. 1: Destination address is really fixed (all bits); bit 14 must be '0'.
29	STARVE_LOW_PRIO	R/W	0: reset value 1: Starve the low priority channels. Low priority channels get executed when high priority is not requesting. It has to be cleared by software when not needed.
31-30	UNUSED	R/W	Writing to these bits has no effect. Read will return all zeros.

Byte count register			
31	RD_EOT	R/W	Setting this bit to '1' will enable the end of read transfer signal for this channel. Reset is '0'.
30	WR_EOT	R/W	Setting this bit to '1' will enable the end of write transfer signal for this channel. Reset is '0'.
20 – 0 (Byte_count_size -1 : 0)	BYTE_COUNT	R/W	Byte_count_size is programmable. Byte_count holds the block size in bytes for the next DMA transfer.

Channel I Interrupt Register and DMA-version			
Channels	INT_REG	R	Indicates what channel is interrupting, where bit zero indicates channel zero, bit one is channel one and etc.
23-16	NUM_OF_CH	R	Returns number of channels.
31-24	Version	R	Highest 8-bits is DMA version number = 0x01.

7.2 ATA Block

The ATA(PI) controller core is a dedicated hardware block performin g the function of controlling one or two IDE devices, either ATA or ATAPI. The core provides the full ATA interface, and core communication is via a CPU interface and a high speed DMA data interface. Commands are sent to the attached devices in the form of SBP-2 compliant ORBs, via the CPU interface. Any data associated with the command can be transferred to or from the ATA device via either interface. The core is responsible for all protocols, data transfers and commands handling of the ATA or ATAPI device, and is fully compliant with all mandatory features of the ATA/ATAPI-6 spec. In addition to interfacing to IDE drives, the core contains a bypass mode which allows the systems DMA engine to interface with an external DMA capable device such as a SCSI controller. This interface uses almost the same pins as the IDE interface does (see section 7.3 ATA Bypass).

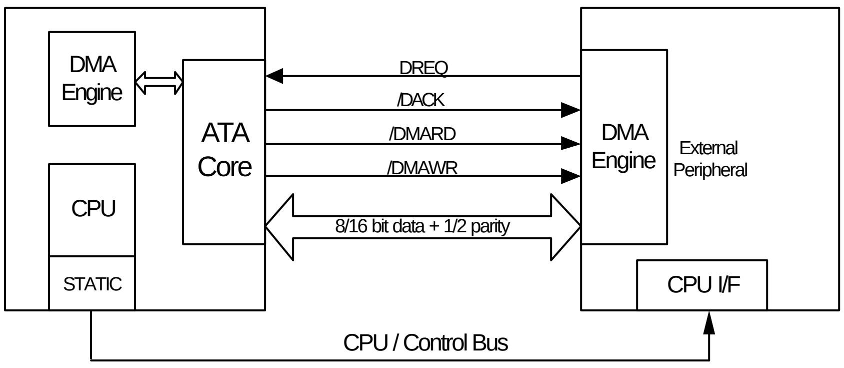
Register	Offset	Description
ORB1	0x00	First of four registers used to write ATA/ATAPI commands into the device
ORB2	0x04	Second of four registers used to write ATA/ATAPI commands into the device
ORB3	0x08	Third of four registers used to write ATA/ATAPI commands into the device
ORB4	0x0C	Fourth of four registers used to write ATA/ATAPI commands into the device
Command & Status 0	0x10	Contains command and status information for the MASTER ATA device
Command & Status 1	0x14	Contains command and status information for the SLAVE ATA device (If present)
Dev Ctrl	0x18	Provides the interface for setting ATA core operation parameters
Burst buffer port	0x1C	CPU read / write access to the DMA FIFO
RBC1	0x20	First of three registers used to write RBC CDBs into the device that require translation.
RBC2	0x24	Second of three registers used to write RBC CDBs into the device that requires translation.
RBC3	0x28	Third of three registers used to write RBC CDBs into the device that require translation.
IDE access	0x2C	Direct IDE register access
Interrupt status	0x30	Read to determine interrupt status, write to clear active interrupt
Interrupt enable set	0x34	Active high interrupt enable set register – write only
Interrupt enable clear	0x38	Active high interrupt enable clear register – write only
Interrupt enable	0x34/8	Active high interrupt enable register – read only
Version	0x3C	Read only version register
Burst Buffer Control	0x48	Software force EOT to complete DMA transfer when required

7.3 ATA Bypass

The ATA Bypass function allows the IDE bus and specific protocols to be bypassed, which allows the DMA engine to use the burst buffer FIFO and the IDE pins of the ATA block to drive an external DMA bus. It can be used in a variety of different modes to enable interfacing to various external peripherals and DMA controllers.

- Master FIFO control. As a master the Bypass Control drives the transfers to the DMA peripherals.
- 8 quadlet deep FIFO
- Programmable control signal polarity.
- Programmable parity.
- Programmable strobe timing.
- Configurable data bus size 8 or 16 bit
- Asynchronous interface.
- Single, burst and continuous transfer modes.

It has two main compatibility modes, Am53CF94/6 and FAS compatible mode.



7.3.1 Pin Mapping

Pin number	IDE Pin Name	Bypass Pin
73,71,65,63,60,56,54,52,49,53,55,57,61,64,68,72	ID[15:0]	data(15:0)
44,45	ICS#[1:0]	parity(1:0)
74	DMARQ	DREQ
75	DIOW#	DMAWR
76	DIOR#	DMARD
78	DMACK	DACK

7.3.2 Operation

A transfer is started by the External Peripheral asserting DREQ. This is acknowledged by the OXUF922 asserting /DACK.

For Reads (data being transferred to the OXUF922), the peripheral drives data onto the Data lines and this is latched into the 922 on the edge of /DACK or /DMARD being de-asserted. For Writes (data being transferred from the OXUF922), the data is driven onto the data lines by the OXUF922 and should be latched into the peripheral on the edge of /DMAWR being de-asserted.

There are two types of DMA operation that can be used – 'Valid on DREQ' and 'Start on DREQ'. The former mode requires DREQ to be asserted for the whole of the transfer. If DREQ goes inactive during the transfer, the transfer of data is suspended until it is asserted again. 'Start on DREQ' mode requires DREQ to be asserted only until the transfer starts.

The operation of the /DACK signal is determined by the DACK type selected. It either remains asserted until the transfer completes, or it follows the active strobe signal.

Bypass mode is entered by setting the **enable** bit in the **bypass config** register. This disables the ATA specific blocks, and multiplexes the IDE pins to allow the DMA engine to pass data through the **burst buffer** to an external DMA capable device or for the DMA engine to receive data from the external device through the **burst buffer.** The many parameters associated with the timing and configuration of the interface is controlled through the bypass configuration register. Once the ATA core and the DMA engine have been configured, the **start** bit may be set in the **bypass control** register to allow data transfers to proceed. The size of the transfer is controlled by the DMA engine. Once the transfer has completed an interrupt shall be generated.

Bypass Registers

Address Offset	Name	Description
0x40	Bypass Configuration	Configure bypass control to match external DMA interface chip
0x44	Bypass Control	Write only to start bypass transfer

7.3.3 Bypass Configuration

Address	Name
0x40	Read /Write Bypass Config parameters See section 7.3.5 Configuration Parameters
Bits	Description
0	Direction: 0 = DMA_IN, 1 = DMA_OUT
1	Parity: 0 = Odd parity, 1 = Even parity
2	Data bus: 0 = 8 bit bus, 1 = 16 bit bus
3	Byte lane: 0 = upper byte lane, 1 = lower byte lane
4	dreq_lvl: 0 = active low, 1 = active high
5	dack_lvl: 0 = active low, 1 = active high
6	dmard_lvl: 0 = active low, 1 = active high
7	dmawr_lvl: 0 = active low, 1 = active high
8	dma_type : 0 = Valid on Dreq, 1 = Start on Dreq
9	dack_type: 0 = cselect, 1 = strobe
15:10	Burst size – 0 to 63
18:16	tsaw -
21:19	tsnw
24:22	tndack
27:25	tsndn
30:28	tdacks
31	enable

7.3.4 Bypass Control / Status

Address	Name
0x44	Read/ Write Bypass Control and status
Bits	Description
0	Start – Set to start DMA transfer to/ from external DMA interface - self-clearing bit. Write only
1	Parity Error – Read only
2	Dreq – resampled Dreq pin – Read only
31:3	Reserved (0x000000)

7.3.5 Configuration Parameters

Parameter	Setting		Default
Direction	DMA_IN	DMA_OUT	DMA_IN
Parity	Odd	Even	Odd
Data Bus	Byte	Word	Word
Byte Location	Upper	Lower	Upper
Dreq active level	High	Low	High
Dack active level	High	Low	Low
Dmard active level	High	Low	Low
Dmawr active level	High	Low	Low
DMA Type see note 1	Valid on Dreq	Start on Dreq	Valid on Dreq
Dack Type see note 2	Chip Select	Strobe	Chip Select
Burst Size see note 3	0 - 63		0

Notes:
1. DMA Type
Start on Dreq only requires Dreq to be asserted until the transfer starts. The transfer shall only be valid for burst sizes of 1 to 63 transfers. Not allowed for continuous transfers when burst size = 0.
Valid on Dreq requires Dreq to remain asserted to validate the transfer. If during a strobe Dreq negates, that transfer remains valid, but the state machine must see Dreq asserted again before any more strobes will be generated. Continuous transfers only.

2. Dack Type
Chip Select type causes the Dack signal to remain asserted until the transfer completes.
Strobe type caused the Dack signal to follow the active strobe signal.

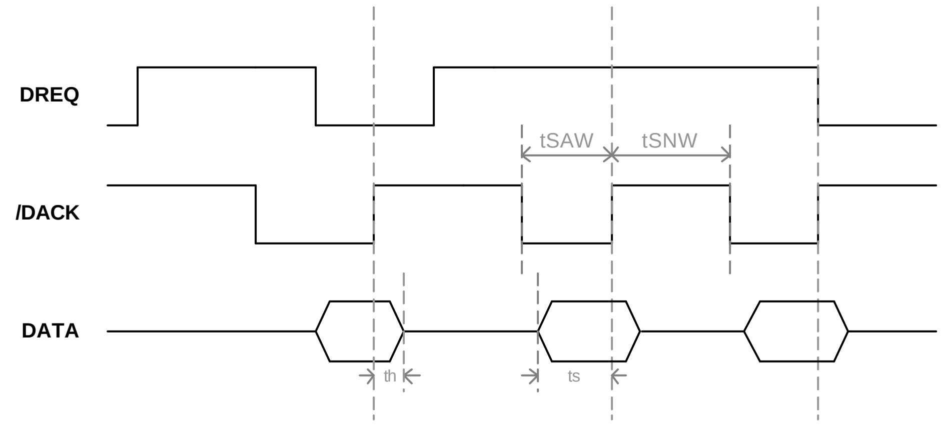
3. Burst Size
0 = continuous transfer, transfers started and stopped via Dreq. Must use Valid on Dreq DMA Type.
1 – 63 = number of bus transfers per assertion of dreq. Must use Start on Dreq DMA Type.

7.3.6 Timing Parameters

Parameter	Symbol	Option	Notes
Strobe Active Width	tSAW	0 – 7	clk_period * (tSAW+1)
Strobe Negated Width	tSNW	0 – 7	clk_period * (tSNW+1) ns
Dack Negated to Dack Active	tNDACK	0 – 7	clk_period * (tNDACK+1) ns **
Strobe Negated to Dack Negated	tSNDN	0 – 7	clk_period * (tSNDN+1) ns **
Dack asserted to Strobe asserted	tDACKS	0 – 7	clk_period * (tDACKS+1) ns **

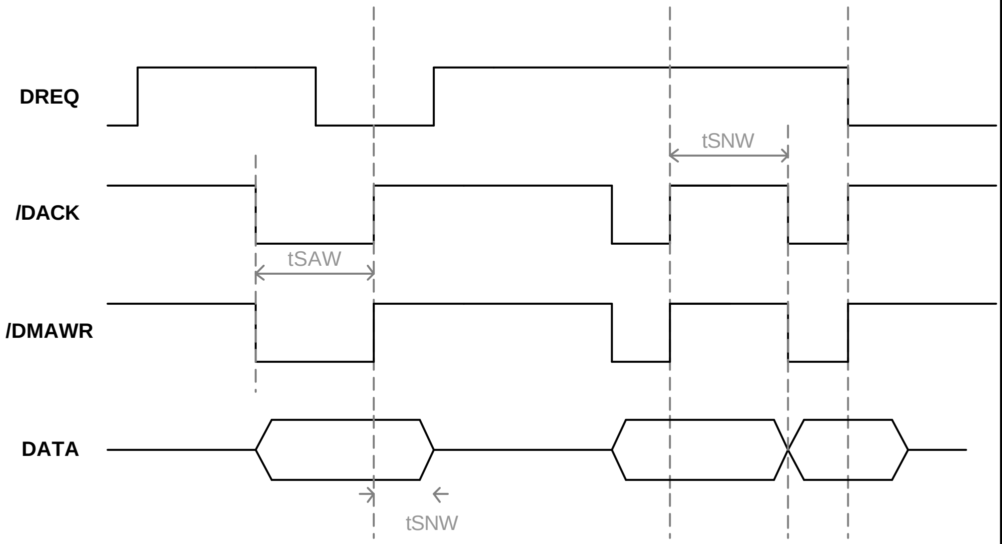
** Dack_type=CSELECT only

Am53CF94/6 Compatible Read w/o Byte Control



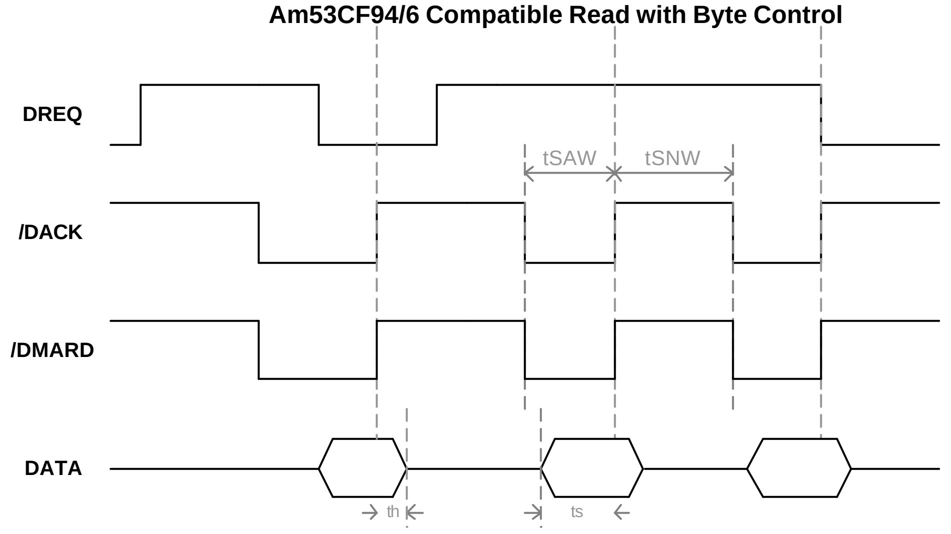
Bypass Configuration Parameters:
Direction = DMA_IN, Parity = Odd,
Data_bus = Word, Dreq_lvl = High,
Dack_lvl = Low, DMA Mode = Valid on DREQ,
DACK Mode = Strobe, Burst Size = 1

Am53CF94/6 Compatible Write w/o Byte Control



Bypass Configuration Parameters:
Direction = DMA_OUT, Parity = Odd,
Data_bus = Word, Dreq_lvl = High,
Dack_lvl = Low, DMAWR_lvl = Low,
DMA Mode = Valid on DREQ, DACK Mode = Strobe, Burst Size = 1

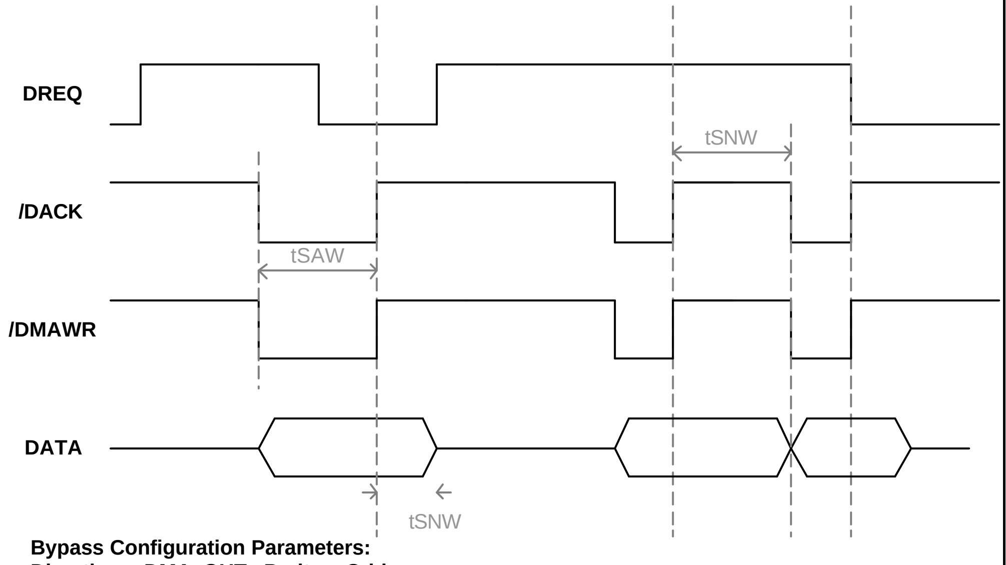
Am53CF94/6 Compatible Read with Byte Control



Bypass Configuration Parameters:
Direction = DMA_IN, Parity = Odd,
Dreq_lvl = High, Dack_lvl = Low, DMARD_lvl = Low,
DMA Mode = Valid on DREQ,
DACK Mode = Strobe, Burst Size = 1

Data_bus and Byte_Lane must be configured to match AS0 and BHE setting on Am53CF94/6

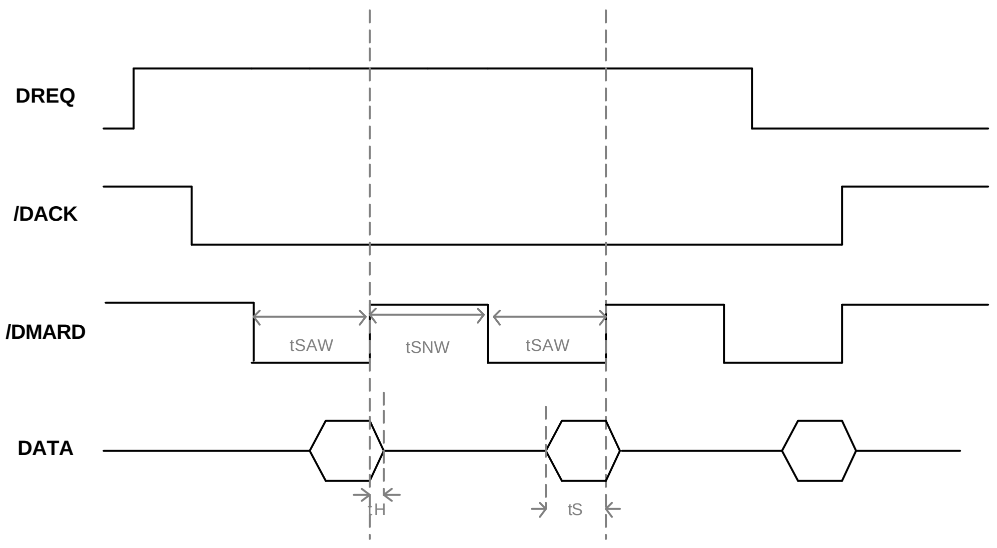
Am53CF94/6 Compatible Write with Byte Control



Bypass Configuration Parameters:
Direction = DMA_OUT, Parity = Odd,
Dreq_lvl = High, Dack_lvl = Low, DMAWR_lvl = Low,
DMA Mode = Valid on DREQ, DACK Mode = Strobe, Burst Size = 1

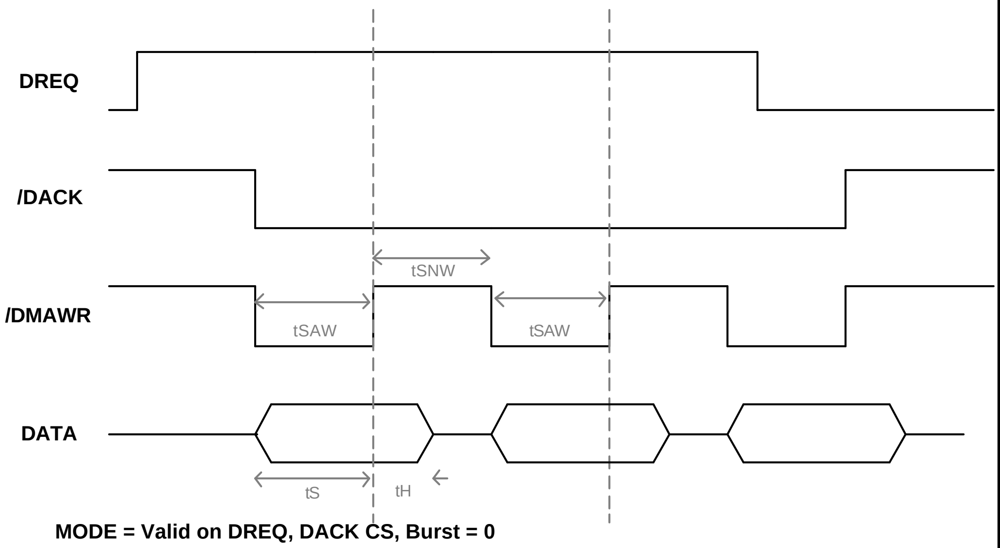
Data_bus and Byte_Lane must be configured to match AS0 and BHE setting on Am53CF94/6

FAS Compatible Read



Bypass Configuration Parameters:
Direction = DMA_IN, Parity = Odd / Even,
Dreq_lvl = High, Dack_lvl = Low, DMARD_lvl = Low,
DMA Mode = Valid on DREQ,
DACK Mode = CSELECT, Burst Size = 0

FAS Compatible Write



MODE = Valid on DREQ, DACK CS, Burst = 0

Valid only when DREQ active during strobe, number of bytes/words = burst size. /DACK used as chip select, data valid on DMAWR rising edge and DREQ active (at any point during strobe)

7.3.7 Bypass Limitations

No termination state

None of the interfaces have a terminate state that can be distinguished from a pause therefore when the DMA engine finishes (burst_buffer_empty=TRUE) the state machine will return to the start state resetting all internal pointers. The burst buffer will do likewise.

burst_buffer_empty shall only be asserted when the burst buffer FIFO is empty and the EOT signal from the DMA engine has been asserted.

Two situations exist which may cause problems.
1/ External device does not return enough data when reading or stops before all write data has been sent. CPU must work with some timeout value then check registers in the external device to determine what action to take. Note, no ATA interrupts in this case.
2/ External device sends more valid data than the DMA engine is programmed for. Up to two quads could be received in this block but lost when the DMA engine finishes. Possible solution would be to not programme EOT enable in DMA engine and let CPU determine action after DMA engine has finished programmed transfer. Note, no ATA interrupts in this case. This does not affect writes.

The logical state of dreq is passed back to the register block for the CPU to examine if necessary.

The size of transfers is limited to multiples of a quadlet in this version.

Bypass mode does not work with 8bit mode when data is on upper byte lane

Lower byte lane OK. Possible work-around is to route the data bus to the lower byte lane.

Bypass mode bandwidth limitations

Fastest setting for 16bit write transfer, tsaw=0, tsnw=1 = 66 MB/s
Fastest setting for 16bit read transfer, tsaw=0, tsnw=0 plus quad overhead of 1 cycle = 80 MB/s
(Assuming 100 MHz input clock)

7.3.8 Bypass Mode in a streaming implementation

An implementation could stream data continuously from the 1394 interface to the external DMA interface.

This operation would set the bypass into a continuous data stream mode. An EOT is never sent to the ATA core when the DMA is done, but an EOT needs to done to the FIFO Manager in order to empty its queues so that subsequent 1394 packets can be received.

7.4 Link-Core

The Link provides the interface between the 1394 phy on one side and the internal sub-blocks on the other (Async Engine and or isochronous AV Block). It may be implemented as either a 1394b or a 1394-1995/ 1394a-2000 Link, selectable by an external pin. The phy-link interface is parallel, operating up to 800Mb/s (S800) for 1394b or 400Mb/s (S400) for 1394a. The Link core is capable of transmitting and receiving asynchronous and isochronous data. The Link is designed to be a re-usable core, which passes 1394 packet information to the internal interface in quadlet format at varying bit rates. It performs no processing on the contents of valid packets. Packets with corrupt headers are discarded. It is Cycle Master capable, Isochronous Resource Manager capable and Bus Manager capable. The Link is capable of operating in multiple clock domains. The Phy side uses a Phy generated 50MHz (49.152) or 100MHz (98.304) clock, for legacy or beta respectively.

7.4.1 Register Set

Register	Offset	Description
Link0	0x00	Function enable register
Link1	0x04	PHY register access and bus notifi cation status
Link2	0x08	Value of the current cycle timer
Link3	0x0C	Power up, power down and reset control
Link4	0x10	Link core version
Link5	0x14	Isochronous and B format controls
Link6	0x18	Enables specific doorbells
Link7	0x1C	Indicates which enabled doorbells have been received.
Link8	0x20	Isoch channel available 31 downto 0
Link9	0x24	Isoch channel available 63 down to 32
Link10	0x28	Bus timer
Link11	0x2C	PHY packet transmision
	0x30	reserved
	0x34	reserved
	0x38	reserved
Link12	0x3C	General enables and status
Link13	0x40	Interrupt Status
Link14	0x44	Interrupt Enable
Link15	0x48	Isoch channel status 31 downto 0
Link16	0x4C	Isoch channel status 63 down to 32
	0x50	reserved
	0x54	reserved

7.5 FIFO Manager.

The asynchronous 1394 rx/tx FIFOs and the USB endpoints are maintained by the FIFO manager. This block allows dynamic allocation of FIFO sizes and provides CPU and high speed DMA access to them. The FIFO Manager registers can be found starting at location 0x0A400000.

The FIFO manager functionality can be broken down into three distinct sections: the data plane ram access unit, the context ram unit and the AHB slave units.

7.5.1 Data plane

The data plane RAM function provides arbitration and timing control allowing multiple read and write sources to access a shared memory resource. The data plane only supports four quadlet incrementing address accesses from odd or even start addresses. The reader and writer arbitrate for the ram and once granted writers provide a four clock burst of write data controlled by an enable signal, readers are provided with read data accompanied with a valid signal. A function within the chip may use the FIFO manager data plane unit without using any of the other functions just as it would a ram. The data plane arbitration is implemented in a simple round robin inspecting high priority requests then low priority requests. Back to back requests if granted allow continuous bursting.

7.5.2 Context RAM

The context ram unit allows readers, writers and the controlling CPU to define FIFO structures in the data plane ram and maintain state e.g. sizes, locations, read/write pointers and fill level. Therefore it is this unit plus the distributed control in the read and write functions, which maintains the logical FIFOs in FIFO manager memory. The position and length of FIFOs is programmable along with many other characteristics.

Not all the readers and writers to the data plane will use this resource. Access to the context is controlled via arbitration and a series of locks so that two agents cannot write to the same FIFO at the same time. The writing and reading of context about the FIFOs will require time which the agents must allow in latency calculations due to the arbitration and current activity within the context unit.

7.5.3 AHB slave units and APB interface

There are two AHB slave interface units, one for interfacing to the data DMA bus the other for interfacing to the CPU interface. Both interfaces allow access to the FIFOs supporting read and write operations through FIFO context, as any other reader and writer would expect. There is an APB CPU interface which deals with configuration registers and context ram configuration. Debug access to the data plane ram is provided by overlaying a FIFO structure over the ram at the addresses required and accessing as a FIFO.

7.5.4 Register Set

Register	Offset	Description
control	0x00	CPU access to queue number and version register
scratch_control	0x04	for future use
spare1	0x08	for future use
spare2	0x0C	for future use
ctx_win[4]	0x10 – 0x1C	read and write pointers for context windows
clear write	0x20	specify queue number
clear read	0x24	specify queue number
set write	0x28	specify queue number
set read	0x2C	specify queue number
cpu_lock[4]	0x30 – 0x3C	set / clear cpu lock on queue
cpu_mask[4]	0x40 – 0x4C	set / clear cpu mask on queue
cpu_int[4]	0x50 – 0x5C	queue interrupt status
read_lock[4]	0x60 – 0x6C	read lock status of queues
write_lock[4]	0x70 – 0x7C	write lock status of queues
not_empty[4]	0x80 – 0x8C	fill status of queues
thresh_exceed[4]	0x90 – 0x9C	threshold exceeded status of queues
cpu ahb control	0x100	various control bits for fm cpu if
cpu ahb rw_ptr	0x104	read only read pointer / write pointer register for fm cpu if
cpu ahb base_top	0x108	read only top and base pointer register
cpu ahb new_read_ptr	0x10C	read only new_read_ptr register
cpu ahb new_write_ptr	0x110	read only new_write_ptr register
cpu ahb fill_level	0x114	for future use
dma ahb control	0x200	various control bits for fm dma if
dma ahb rw_ptr	0x204	read only read pointer / write pointer register for fm dma if
dma ahb base_top	0x208	read only top and base pointer register
dma ahb new_read_ptr	0x20C	read only new_read_ptr register
dma ahb new_write_ptr	0x210	read only new_write_ptr register
dma ahb fill_level	0x214	for future use

7.6 ORB Co-Processor.

The ORB Coprocessor (OCP) is an acceleration unit designed to facilitate efficient 1394 SBP-2, SBP-3 and USB Bulk transfer mode operations at hardware speeds in the system.

The ORB Coprocessor unloads the task of packet based DMA and header building for ORBs. The unit interacts with the FIFO manager and DMA engine, to move data into and out of the FIFO manager. The destination and source of the data stream is either memory mapped SDRAM or FIFO based i.e. ATA.

Support for USB transfers is essentially a subset of that required for 1394 as no header building is required for USB.

OCP Registers start from 0x0A200000

Address Offset	Quadlet offset	Register Name	Reset Value
0x000	0	OCP_AHB_CONTROL	0x00000000
0x004 to 0x0FF	-	Unused	-
0x100	64	OCP_STATUS	0x00000003
0x104	65	OCP_ORB_CONTROL	0x00000000
0x108	66	OCP_DMA_CONTROL	0x00000000
0x10C	67	OCP_FM_QUEUE	0x00000000
0x110	68	Unused	-
0x114	69	OCP_NODE_ID	0x00000000
0x118	70	OCP_ORB_LEN	0x00000000
0x11C	-	Unused	-
0x120	-	Unused	-
0x124	-	Unused	-
0x128	74	OCP_DMA_SOURCE	0x00000000
0x12C	75	OCP_DMA_DEST	0x00000000
0x130	76	OCP_DATADESC_1A	0x00000000
0x134	77	OCP_DATADESC_1B	0x00000000
0x138	78	OCP_DATADESC_2A	0x00000000
0x13C	79	OCP_DATADESC_2B	0x00000000
0x140 to 0x1F8	-	Unused	-
0x1FC	127	OCP_VERSION	0x4F435031 “OCP1”

7.7 Queue Selector

The queue selector resides between the 1394 Async Link receive and the Async Engine management sub-system in order to examine incoming packets and route them into appropriate queues. The design takes the form of a very simple microcode engine for maximum flexibility and reusability.

Queue Selector registers start from 0x0A280000.

Address Offset	Register Name
0x000	QS_CSR
0x004 to 0x014	Unused
0x018	QS_HB_DEBUG
0x01C	QS_CPU_DEBUG
0x020 to 0x0F8	Unused
0x0FC	QS_VERSION
0x100	QS_RAM_ACCESS_BASE
0x100 to 0x1FC	QS_RAM_ACCESS registers

7.8 Async Engine

The Asynchronous Engine provides the data handling stage between the IEEE-1394 Link Core and the FIFO Manager. Asynchronous Engine provides hardware support for the following:

- 1394 Packet Receive
- 1394 Packet Transmit
- Packet Transmission Error Handling
- Received Packet Queue Selector
- Interface to the FIFO Manager

Previous bridge implementations have a limited amount of queue selection, designed expressly for an efficient SPB-2 implementation. In order to support other protocols and make the solution more generic a better queue selection scheme is required.

Async Engine registers start from 0x0A200000

Address Offset	Quadlet offset	Register Name
0x000	0	AE_CONFIG
0x004	1	AE_SPLIT_TRANS
0x008	2	AE_CONTROL
0x00C	3	AE_STATUS
0x010	4	AE_RETRANS
0x014	5	AE_INTR_ENABLE
0x018	6	AE_INTR_STATUS
0x01C	7	AE_RESET
0x020	8	AE_TX_QUEUES_0
0x030	12	AE_ACK_DISCARD
0x034	13	AE_STREAM_CHANNEL_EN_0
0x038	14	AE_STREAM_CHANNEL_EN_1
0x03C	15	AE_RX_QUAD_COUNT_CONFIG
0x040	16	AE_INTR_0
0x044	17	AE_INTR_1
0x048	18	AE_INTR_2
0x04C to 0x0FF		Unused
0x100	64	AE_RX_DEBUG
0x104	65	AE_RXFIFO_DEBUG
0x108	66	AE_RX_IF_DEBUG
0x10C	67	AE_RX_QUAD_COUNT_DEBUG
0x110	68	AE_TX_DEBUG
0x114	69	AE_TXFIFO_DEBUG
0x118	70	AE_TX_IF_DEBUG
0x11C to 0x1F8		Unused
0x1FC	127	AE_VERSION

7.9 USB2 core

The USB core interfaces a USB2.0 internal physical layer (PHY) to the FIFO manager in the system. The core performs all the functions required to efficiently manage end points in generic USB applications including bulk transfer protocols.

The USB core has the base address of 0x0A300000.

Due to the different clock domains in the USB design, a method of safe clock crossing must be used to read and write the registers. The following table defines the registers that must be used. When writing to the USB the data and write address must be written then the control register polled to ensure the write has completed. For reads the read address register should be written first, the control register polled until the data is valid, then it can be read from the data register.

Address Offset	Register	Operation
00h	DATA register	R/W
04h	READ address register	W
08h	WRITE address register	W
0Ch	CONTROL register	R/W

The Endpoint and other registers are defined in the address map below.

Address	Targeted Register
00000h	Endpoint 0 OUT (Control)
00004h	Endpoint 1 OUT
00040h	Endpoint 0 IN (Control)
00044h	Endpoint 1 IN
00048h	Endpoint 2 IN
00088h	PHY Status & Control
000CCh	General Purpose I/O Register
00300h	HS Timeouts
00304h	FS Timeouts
003F0h	Function Enables
003F4h	Mask Register
003FCh	Universal Register

7.10 Serial Controller

The OXUF922 provides a two wire method of transmitting and receiving serial data.

Serial Controller registers can be found starting at location 0x0AA00000.

Offset	Read	Write
0	ADDRESS/CONTROL	ADDRESS/CONTROL
4	READ DATA REG	WRITE DATA REG
8	SW CONTROL OUT	SW CONTROL OUT
C	SW CONTROL IN	SW CONTROL IN

Serial address and control register

Register bit	Read	Write
0	SERIAL TRANSACTION NOT DONE	SERIAL TRANSACTION GO
1	SERIAL TRANSACTION FAIL	
2	SERIAL READ/WRITE	SERIAL READ/WRITE
3	TRANSACTION TYPE	TRANSACTION TYPE
4	SERIAL CONTROLLER RESET	SERIAL CONTROLLER RESET
15 to 5	ADDRESS[10:0]	ADDRESS[10:0]
31 to 16	Undefined	

SERIAL TRANSCATION GO:

When set to 1 the chosen transaction as indicated by the other bits in this registers are executed. This bit is cleared by hardware when the transaction has completed. This bit may be polled to indicate that a transaction is in progress.
Note: That for a write transaction the WRITE DATA REG in the Serial read DATA register must be programmed in advance.

SERIAL TRANSACTION DONE:

Cleared to 0 by hardware and indicates the completion of a transaction.

SERIAL TRANSACTION FAIL:

When 1 indicates that the completed transaction failed. This is likely due to no response from the serial bus slave. This bit is cleared by hardware when the transaction is successful.

SERIAL READ WRITE:

When the TRANSACTION TYPE is 0 then this bit controls whether a read or a write cycle is performed.
WRITE CYCLE = 1 READ CYCLE = 0

TRANSACTION TYPE:

When 0 the transaction will be as per SERIAL READ WRITE
When 1 the serial controller will perform the RESET TRANSACTION on the bus.
NOTE: the GO bit must be set to execute this transaction.
The reset transaction should be performed after power up to ensure the serial bus is properly reset. It should also be performed if successive failures are noticed.

SERIAL CONTROLLER RESET:

The serial bus software should implement a timeout to ensure that the serial bus is not locked up there are no timeouts within the serial controller logic. If a time out occurs then the software should perform a SERIAL CONTROLLER RESET by setting this bit. It should then clear this bit and execute the RESET TRANSACTION. Timeout periods will be a function of the selected clock rate.

ADDRESS:

The 11 bit address is mapped directly to the serial data bus.

Serial read DATA register

Register bit	Read	Write
7 to 0	SERIAL READ DATA	SERIAL WRITE DATA
31 to 8	Undefined	undefined

Serial software control out register

Register bit	Read	Write
0	SERIAL CLOCK	SERIAL CLOC K
1	SERIAL DATA OUT	SERIAL DATA OUT
2	SERIAL DATA ENABLE	SERIAL DATA ENABLE
3	SERIAL SW CONTROL	SERIAL SW CONTROL

In order to allow more flexibility over the serial bus it is possible for the software to directly control the serial bus. The serial bus has been multiplexed with GPIO pins see **Table 2 - Serial IF** control in register.

SERIAL CLOCK and DATA:
These map directly to the SERIAL CLOCK and SERIAL DATA pins (if enabled through the GPIO) as illustrated below:

GPIO data(10:8)	GPIO(7)	GPIO(6)	GPIO(5)	GPIO(4)	GPIO(3)	GPIO(2)	GPIO(1)	GPIO(0)
000	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
001	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	serial_data	serial_clock
010				GPIO	GPIO	GPIO	GPIO	GPIO
011				GPIO	GPIO	GPIO	serial_data	serial_clock
100	DTR	DSR	DCD	RI	GPIO	GPIO	GPIO	GPIO
101	DTR	DSR	DCD	RI	GPIO	GPIO	serial_data	serial_clock
110	DTR	DSR	DCD	RI	GPIO	GPIO	GPIO	GPIO
111	DTR	DSR	DCD	RI	GPIO	GPIO	serial_data	serial_clock

Table 2 - Serial IF control in register

SERIAL DATA ENABLE:
When set to 1 this enables the tri-state SERIAL DATA pin to act as an output.

SERIAL SW CONTROL:
Must be set to 1 to enable the SW control mode and allow software to drive the serial bus.

Serial control software in register

Register bit	Read	Write
0	SERIAL DATA IN	NOT APPLICABLE

A single bit is provided to allow the software to read the serial input data bit.

7.11 Serial Audio

Serial Audio is a serial bus (path) design for digital audio devices and technologies such as compact disc CD players, digital sound processors, and digital TV DTV sound. The Serial Audio design handles audio separately from clock signals. The Serial Audio bus design consists of three serial bus lines: a line with two time-division multiplexing (TDM)

- A data line (SDA): single direction
- A clock line (SCL): minimal frequency is 2*Fs*(number of bits/sample).
- A 'Word Select' line (SWS): determines between left and right channel samples

Data is transferred MSB first. Two channels per sample period are transferred: left channel first. For the left channels, the SWS signal has a low level. See figure 3.

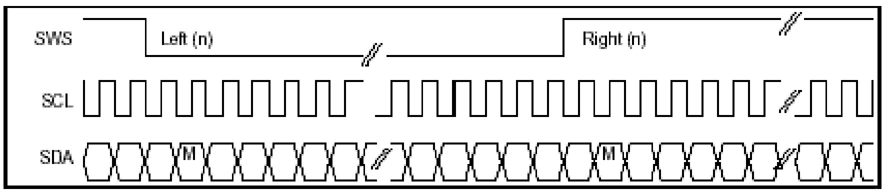
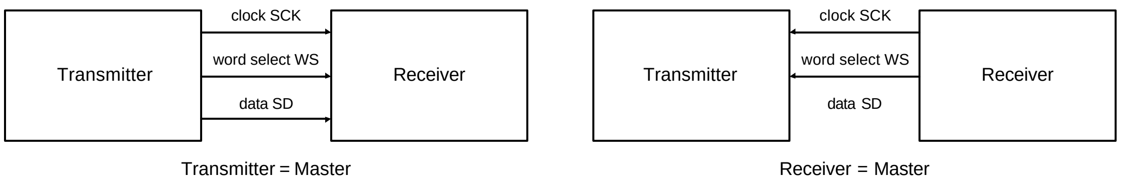
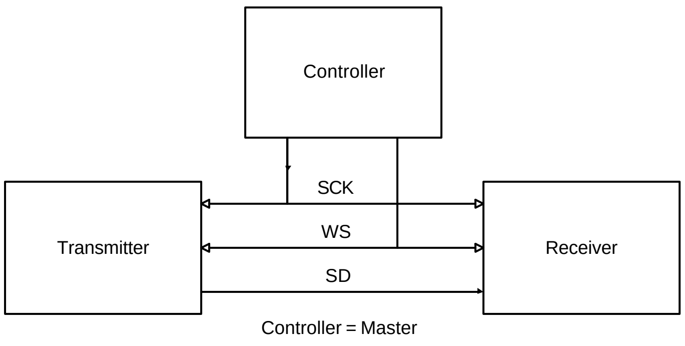


Figure 3





7.11.1 Register Set

The following tables show the definition and functions of the 3 wire serial audio interface status and control register starting at location 0x0AC00000.

Offset	Read	Write
0	Clock Register	Clock Register
4	Control Register	Control Register
8	Status Register	-
C	-	FIFO Register

SERIAL AUDIO Clock Register

BIT	NAME	R/W	DESCRIPTION
10-0	DTO_TERMINAL_COUNT	R/W	Defines the DTO roll-over.
23-11	CLOCK_ACCUMULATION	R/W	Defines the accumulation value, used to generate the DTO increment signal.

SERIAL AUDIO Control Register

BIT	NAME	R/W	DESCRIPTION
0	SERIAL AUDIO_ENABLE	R/W	3 wire serial audio bus enable (1 = active) reset - inactive
1	INT	R/W	When set interrupts are enabled, when reset interrupts are disabled – reset disabled Interrupt only pulses for 1 clock cycle.
2	DELAY_BIT	R/W	When set 3 wire serial audio is Philips format, when reset EIAJ format – reset Philips format.
4-3	DATA_WIDTH	R/W	00 = 32 bit audio data 01 = 24 bit audio data 10 = 16 bit audio data
6-5	INPUT_DATA_WIDTH	R/W	00 = 24 bit audio data 01 = 20 bit audio data 10 = 18 bit audio data 11 = 16 bit audio data
7	INVERT WORD SELECT		Used to convert between Philips (set bit to 0) and Sony mode (bit to 1)
8	RIGHT JUSTIFY		When set the 3 wire serial audio data will be right justified when the output is on the 3 wire serial audio bus Philips mode bit = 0, Sony bit = 1
9	MSB_FIRST	R/W	Defines which bit is sent across the 3 wire serial audio bus first. (1 = MSB first, 0 = LSB first) – reset = 1. Only used if major problem

SERIAL AUDIO Status Register

BIT	NAME	R/W	DESCRIPTION
0	CTS	R	When set input FIFO has at least 1 quadlet of space.
1	INT_PENDING	R	Set when interrupt triggered, reset when the 3 wire serial audio status register read.
2	UNDERFLOW	R	Set when input FIFO has underrun, reset when the 3 wire serial audio status register read.

SERIAL AUDIO FIFO register

BIT	NAME	R/W	DESCRIPTION
31-0	C_REG_DATA	W	Audio data input FIFO

7.11.2 Feature list

- 8 x 3byte input FIFO
- Two methods of serial audio clock generation; one that maintains exact lock to input 1394 stream but will have short term jitter and one that never varies and needs software support to ensure a constant data stream.
- Supports 16, 24 and 32 bit data for Philips and EIAJ formats.
- Supports a range of 61883 audio formats; generic, AM824 (raw, IEC60958). However ALL checking of labels MUST be done by the processor. (Note it is not necessary for the processor to check all samples).
- Supports output only.
- In the event of an underflow of the input FIFO, the last data will be re-transmitted.
- Supports stereo sample rates of 48 KHz, 44.1 KHz, 32 KHz, 24 KHz and 22.05 KHz.

	Quick Summary of Audio DACs		
TI			
	PCM1742, 24-Bit, 96kHZ Sampling Enhanced Multilevel, Delta-Sigma, Audio DAC		tolerent to clock jitter, but no PLL
	PCM1741 - same as 1742, but single supply		tolerent to clock jitter, but no PLL
	PCM1772 same as 1742, but low voltage		not yet available
Philips			
	TA1311, stereo continuous calibration		NOT tolerent to clock jitter
	UDA1320ATS; Low-cost stereo filter DAC		tolerent to clock jitter, but no PLL
	UDA1330 - same as above		tolerent to clock jitter, but no PLL
	UDA1350/1351 - IEC958 audio dac		Has PLL - low jitter
Analog Devices	1866 - dual dac		NOT tolerent to clock jitter
	1865 - dual 18b dacs		NOT tolerent to clock jitter
	1851 - single dac		NOT tolerent to clock jitter
	1862 low noise dac		NOT tolerent to clock jitter
	1859 Stero 18bit DAC		Has PLL - low jitter
Wolfson			
	WM8740 - high performance dac		NOT tolerent to clock jitter

7.11.3 Limitations

- Only simple clock generator is implemented. When locked to the 1394 stream this implies that the chip connecting to the serial audio will need to have tolerance to the clock jitter, or in less expensive applications (fixed frequency method) data might be lost / replicated in the event of underflows / overflows.
- Does not cope with audio data clustered into quadlets where UNIT_SIZE /= UNIT_DIMENSION (see 61883 spec for details – 24-bit * 4 audio pack not supported). This means that each sample occupies a complete quadlet, regardless of actual sample size.
- Input samples will be 24-bit or 16-bit only.
- Not MIDI data or 32-bit floating as defined in 61883.
- Input is controlled by the processor and triggered from an interrupt that is set every 40 us.

8 FORCE FLASH V 1.1

To program the flash via the 1394 bus the device must be in Force Flash mode and the host must issue Write Quadlets to the FLASH PORT CSR which the Link interprets, sends ACK complete and puts the data into the Flash Port register.

Flash Port register

Bit	Function	External Mapping
31	reset	
30	overlay address	
29	cs	cs#(0) (flash)
28	we	we#
27	rd	data bus output enable
26	oe	oe#
25	cs1	cs#(1)
24	cs2	cs#(2)
23:17	not used	
16 : 0 *	address	addr
15 : 0	data	data

* Addr selected when bit 30 (overlay addr) true

The force flash controller generates the flash_loading signal which controls the pin muxing, to enable the Flash Port register data onto the external static bus, hence it will take 3 accesses to do one write cycle e.g.

Address, overlay address, cs
Data, not overlay address, cs, we, data bus output enable
Data, not overlay address, cs, not we, not data bus output enable.

Multiple write cycles may be required to program each location of the Flash depending on its programming algorithm. The time between host accesses must be sufficient to meet timing requirements of the particular Flash device. Note that there is currently no way for the host to read back data from the Flash via this method.

The whole Flash image may be programmed using the above method or a downloader program may be programmed and executed to program the Flash. This downloader would run from internal RAM and allow the host to send Write Block packets containing on the Flash image. This would allow a more efficient and faster software controlled programming sequence.

To enter Force Flash mode, the flash_loading signal is generated under the following conditions

1. If the force_flash pin is held active. Highest priority.
2. If the host writes the enable value to the UPLOAD_MODE_CTRL CSR, the Link will generate the flash_loading signal.
3. If the watchdog times out when the switch_on_timeout signal is true. This is the default start position with a blank flash.

Force Flash mode shall be exited under the following conditions

1. The chip is reset either through a PON reset or the setting of the reset bit in the Flash Port register

The switch_on_timeout is enabled at reset, or when force flash mode is entered via the pin, or when the CPU writes to the SOT register enabling it. This requires a 31 bit data match to ensure no accidental setting, and a bit to enable or disable the feature.

When flash_loading is true, in addition to the pin muxing control, the following conditions are also true

1. The CPU is held in reset
2. Sync_reset is generated 2 clock cycles after the reset bit is set in the flash_port. This takes the chip out of force flash mode and enables the CPU.
3. 1394 Bus Reset shall not cause the chip to exit Force Flash mode

Function	CSR Address	Enable Value
UPLOAD_MODE_CTRL		
FLASH_PORT	F0080000	n/a

Function	CPU Address	Value
Enable switch-on-timeout	0x4A000380	1068AFC5*
Disable switch-on-timeout	0x4A000380	1068AFC4*

*bit (1) = fast_timeout and must be set appropriately

9 OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
V_{DDH}	DC I/O supply voltage	-0.3	4.0	V
V_{DD}	DC Core supply voltage	-0.3	2.2	V
V_{IN}	DC input voltage (3.3V I/O)	-0.3	V_{DDH} + 0.3	V
V_{IN}	DC input voltage (5V tolerant I/O)	-0.3	5.6	V
I_{out}	DC output current		+/- 30	mA
T_{STG}	Storage temperature	-55	125	°C

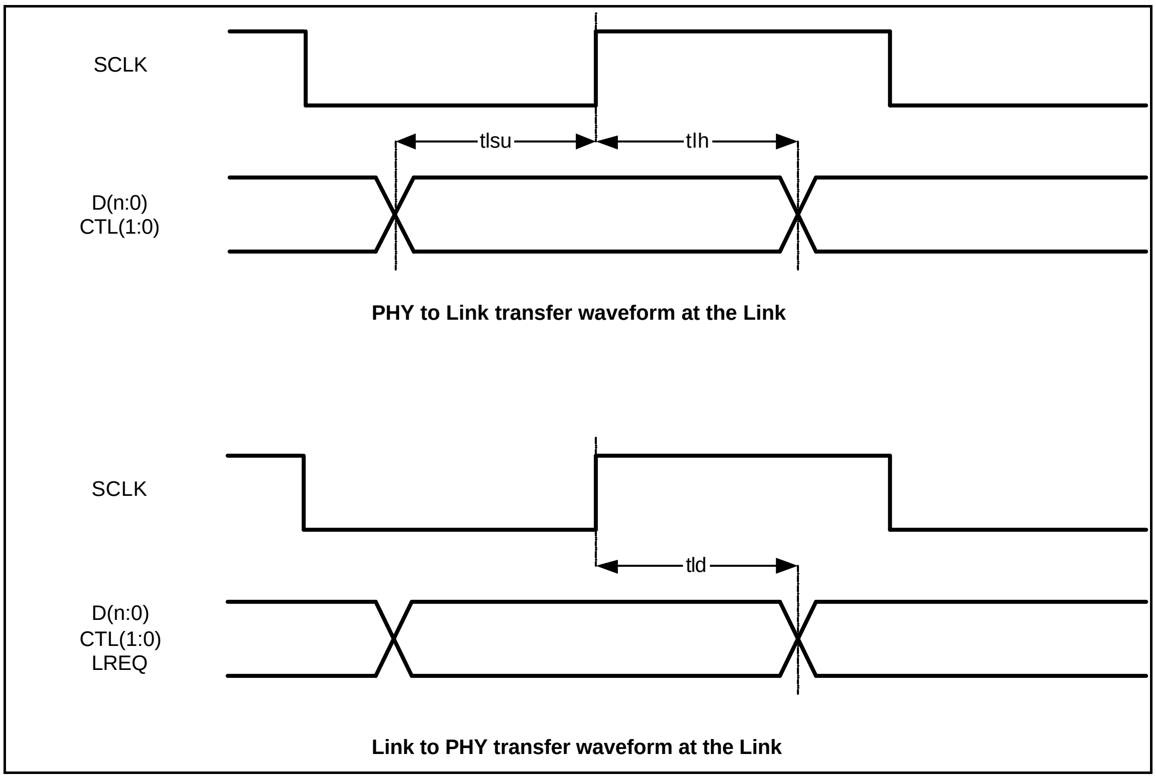
Symbol	Parameter	Min	Max	Units
V_{DDH}	DC I/O supply voltage	3.3	3.6	V
V_{DD}	DC Core supply voltage	1.65	1.95	V
T_A	Ambient Temperature	0	70	°C

10 DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Condition	Min	Max	Units
V_{IH}	Input high voltage	CMOS Interface CMOS Schmitt trig 5V tolerant	2.0		V
V+	Input high voltage	CMOS Schmitt trig	TYP 1.8	2.3	V
V_{IL}	Input low voltage	CMOS Interface[1] CMOS Schmitt trig 5V tolerant		0.8	V
V-	Input low voltage	CMOS Schmitt trig	0.5	TYP 0.9	V
V_H	Hysteresis voltage	Schmitt	0.4		
I_{IH}	Input high leakage current	$V_{in} = V_{DD}$	-10	10	µA
I_{IL}	Input low leakage current	$V_{in} = V_{SS}$	-10	10	µA
V_{OH}	Output high voltage	$I_{OH} = -1$ µA	2.4		V
V_{OH}	Output high voltage	$I_{OH} = -1$mA to –24mA	2.4		V
V_{OL}	Output low voltage	$I_{OL} = 1$ µA		0.4	V
V_{OL}	Output low voltage	$I_{OL} = 1$mA to 24mA		0.4	V
I_{OZ}	3-state output leakage current		-10	10	µA

11 PHY-LINK INTERFACE TIMING DIAGRAMS

11.1 A Mode

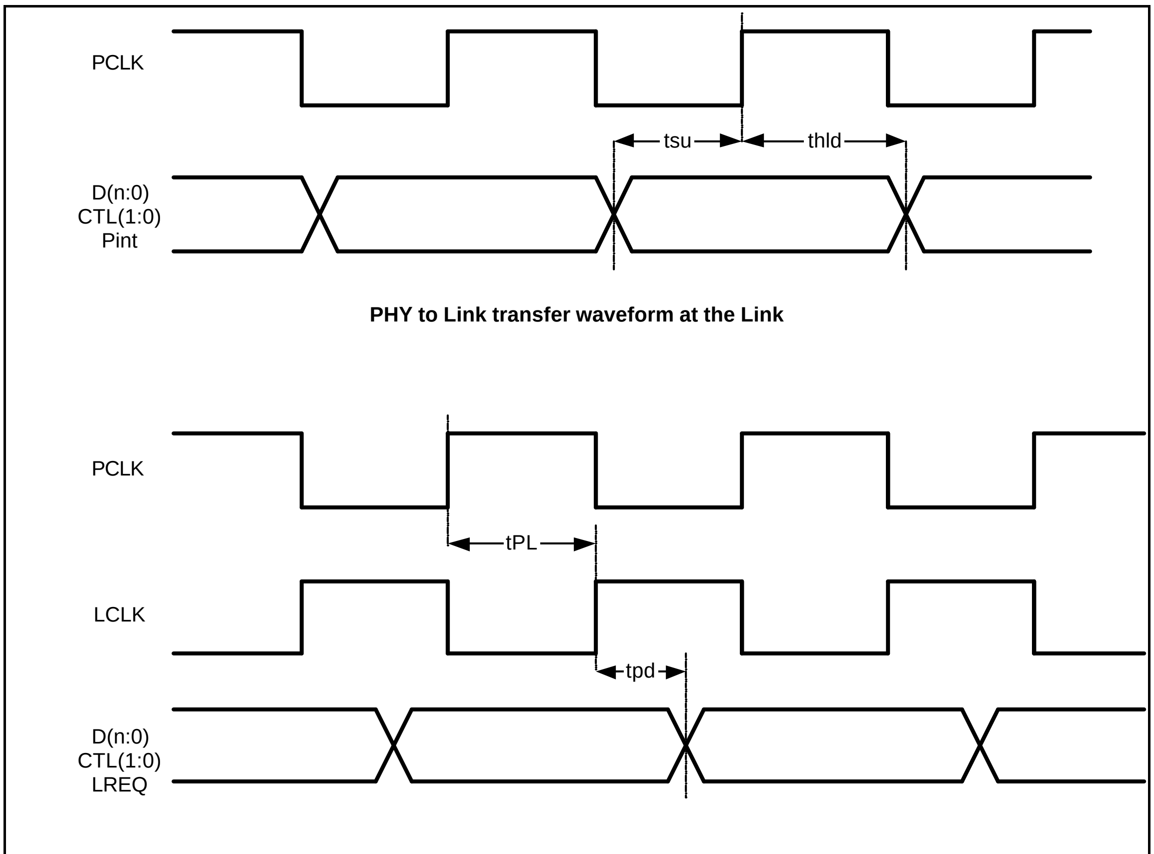

SCLK
tlsu
tlh
D(n:0)
CTL(1:0)
PHY to Link transfer waveform at the Link
SCLK
tld
D(n:0)
CTL(1:0)
LREQ
Link to PHY transfer waveform at the Link

A Mode timing

Name	1394a-2000 specification		OXUF922	
	Min	Max	Min	Max
tlsu	6		2.5	
tlh	0		0	
tld(1/2/3)	1	10	2	8

Cap load on outputs from link = 10pF

11.2 B Mode



PHY to Link transfer waveform at the Link

Link to PHY transfer waveform at the Link

B Mode timing

Name	**1394b**		**OXUF922**	
	Min	**Max**	**Min**	**Max**
tsu	2.5		2.5	
thld	0		0	
tPL		5		5
tpd(1/2/3)	0.5	7.0	0.5	3.0

Capacitance load on outputs seen from link = 10pF

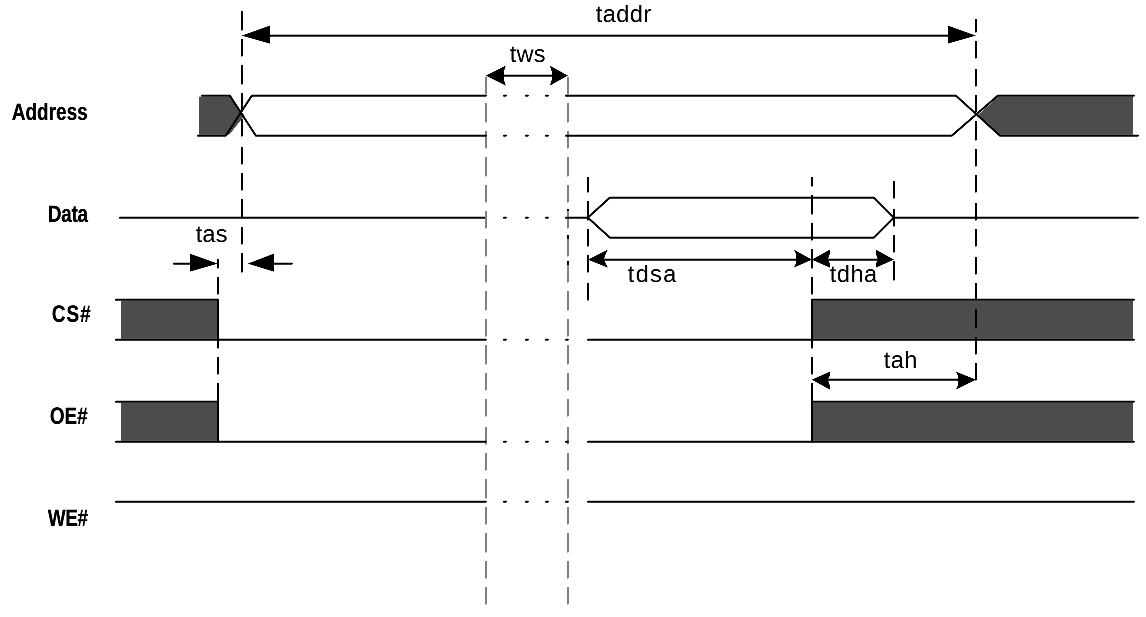
12 EXTERNAL BUS TIMING DIAGRAMS

Asynchronous timing

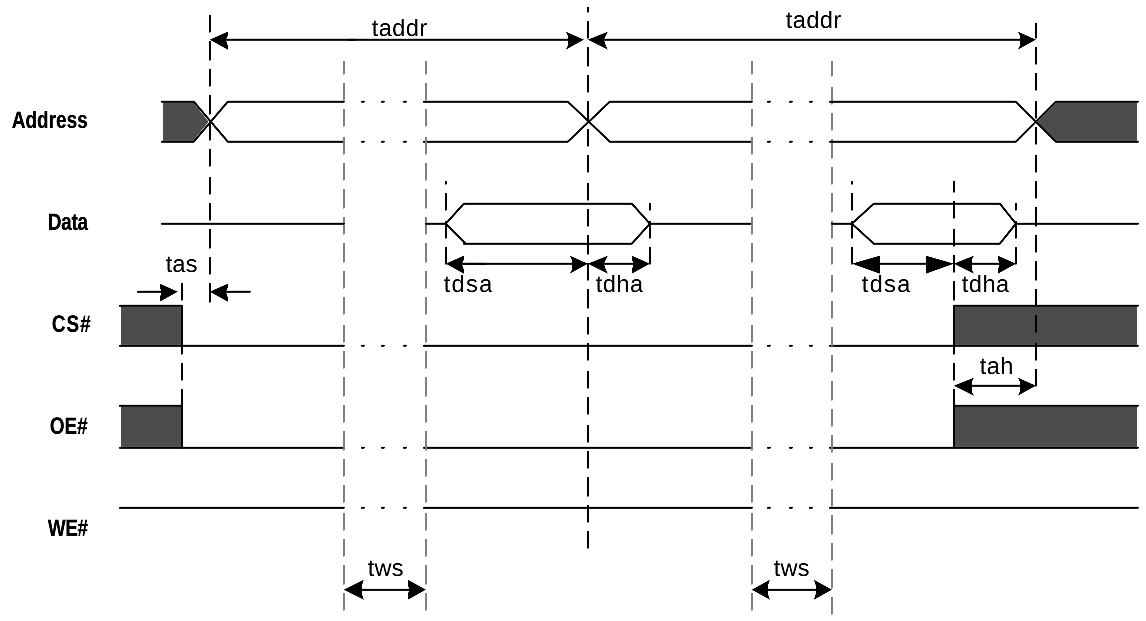
Name	Description	min (ns)	max (ns)
	Common Timings		
tas	CSx# and OE# falling to valid Address	0	2
tws	Wait State Additional Delay (number of wait states (WS) + 1 * 20 ns)	20	5120 ns
taddr	Address Valid	40	20+tws
	Common Read Timings		
tah	Address hold after CSx# or OE# rising	0	-
Taa	Access Time of external device	25	5125
tdsa	Data setup to CSx# and OE# rising	17.5	-
tdha	Data hold after CSx# and OE# rising	0	-
	Common Write Timings		
twds	Data valid to WE# rising	20	20+(tws-20)
twdh	Data hold after WE# rising	17	20
tcsw	CS# setup before WE# valid	0	2
tadw	Address setup before WE# valid	0	2
twe	WE# valid	20	20+tws

Synchronous timing

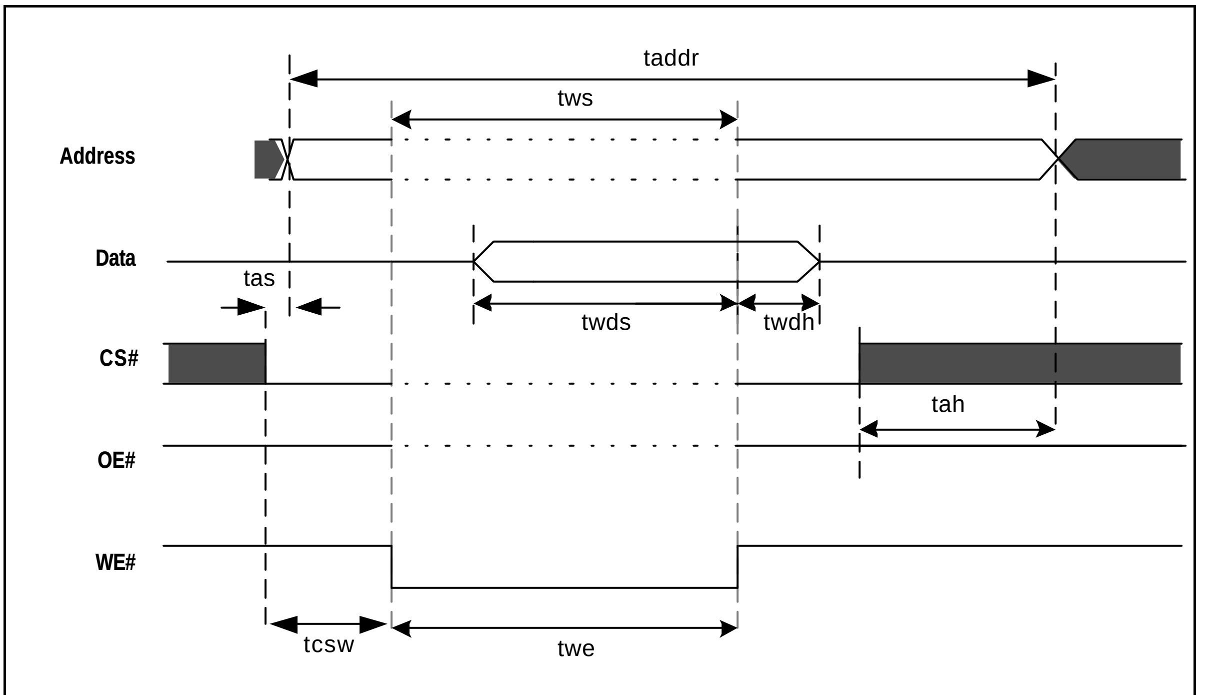
Name	Description	min (ns)	max (ns)
	Common Timings		
tcycle	clock period	20.3	20.8
tcych	clock high	57.7 %	
tcycl	clock low	42.5 %	
tsaa	address valid after rising clock edge	0.8	2.6
tsan	address invalid after rising clock edge	tcycle	tcycle
tscsa	cs asserted after rising clock edge	0.6	1.8
tscsn	cs negated after rising clock edge	0.2	0.6
	Common Read Timings		
tsoea	oe asserted after rising clock edge	-0.1	-0.5
tsoen	oe negated after rising clock edge	-0.1	-0.3
	Common Write Timings		
tsd	data valid after rising clock edge	0.6	1.6
tsdh	data hold after rising clock edge	0.0	-0.2
tswa	we asserted after rising clock edge	0.4	1.6
tswn	we negated after rising clock edge	0.1	0.4

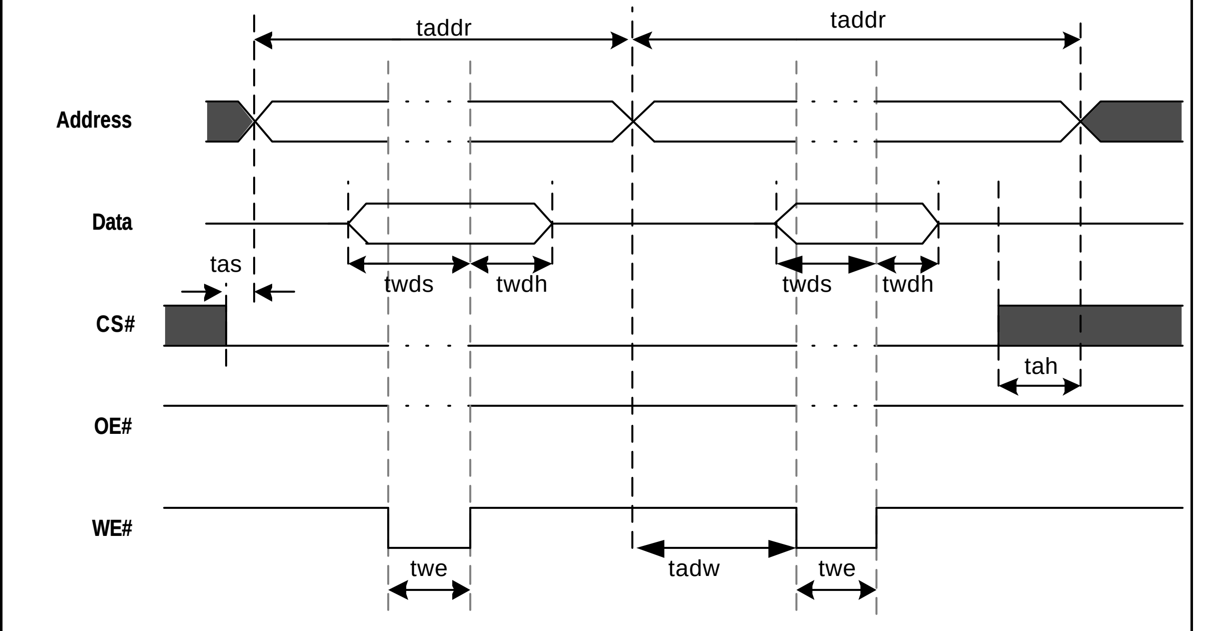


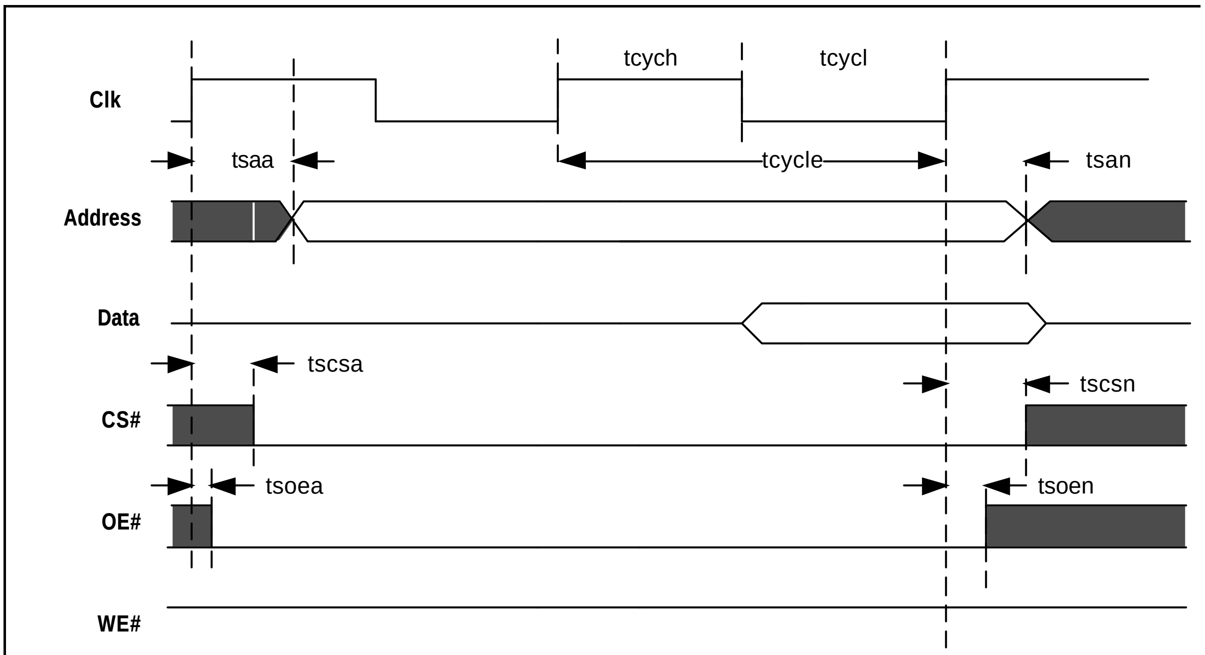
taddr
tws
Address
Data
tas
tdsa
tdha
CS#
tah
OE#
WE#

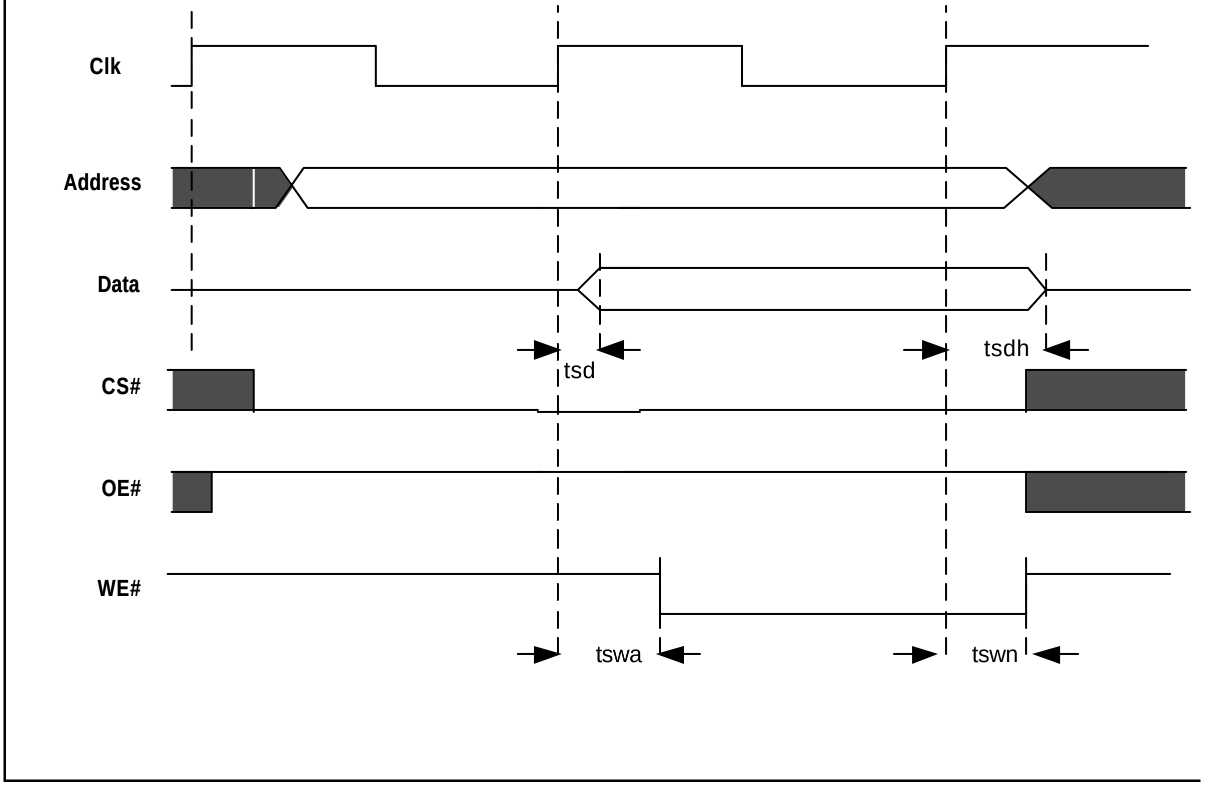


taddr
taddr
Address
Data
tas
tdsa
tdha
tdsa
tdha
CS#
tah
OE#
WE#
tws
tws


taddr
tws
Address
Data
tas
twds
twdh
CS#
tah
OE#
WE#
tcsw
twe


taddr
taddr
Address
Data
tas
twds
twdh
twds
twdh
CS#
tah
OE#
WE#
twe
tadw
twe


Clk
Address
Data
CS#
OE#
WE#
tcych
tcycl
tsaa
tcycle
tsan
tscsa
tscsn
tsoea
tsoen


Clk
Address
Data
CS#
OE#
WE#
tsd
tsdh
tswa
tswn

13 POWER CONSUMPTION

Current in mA.

USB2	1.8 v	3.3 v	Total mA
Running(idle)	136	37	173
Running(Working)	136	43	179

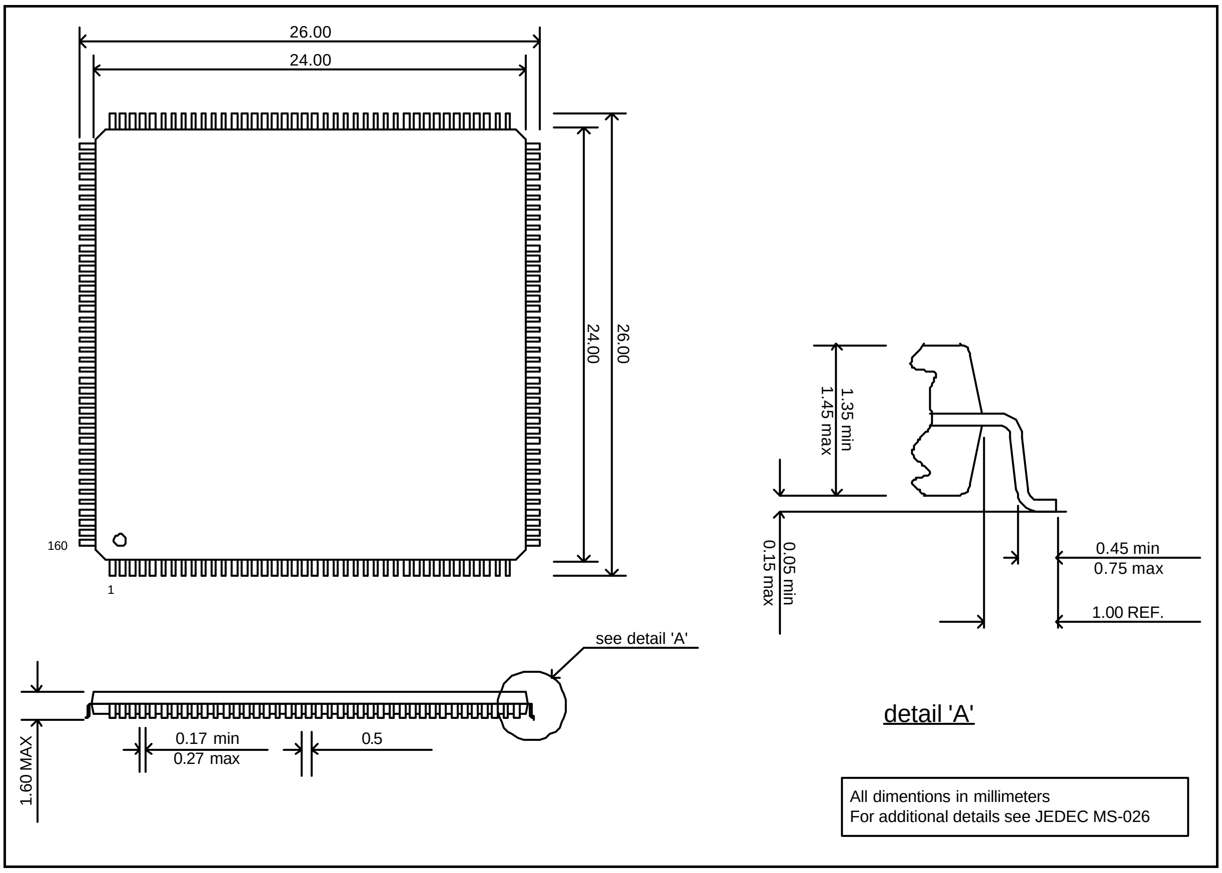
USB1	1.8 v	3.3 v	Total mA
Running(idle)	136	37	173
Running(Working)	136	40	176

1394 'A' Mode	1.8 v	3.3 v	Total mA
Running(idle)	122	44	166
Running(Working)	142	44	186

1394 'B' Mode	1.8 v	3.3 v	Total mA
Running(idle)	129	50	179
Running(Working)	TBD	TBD	TBD

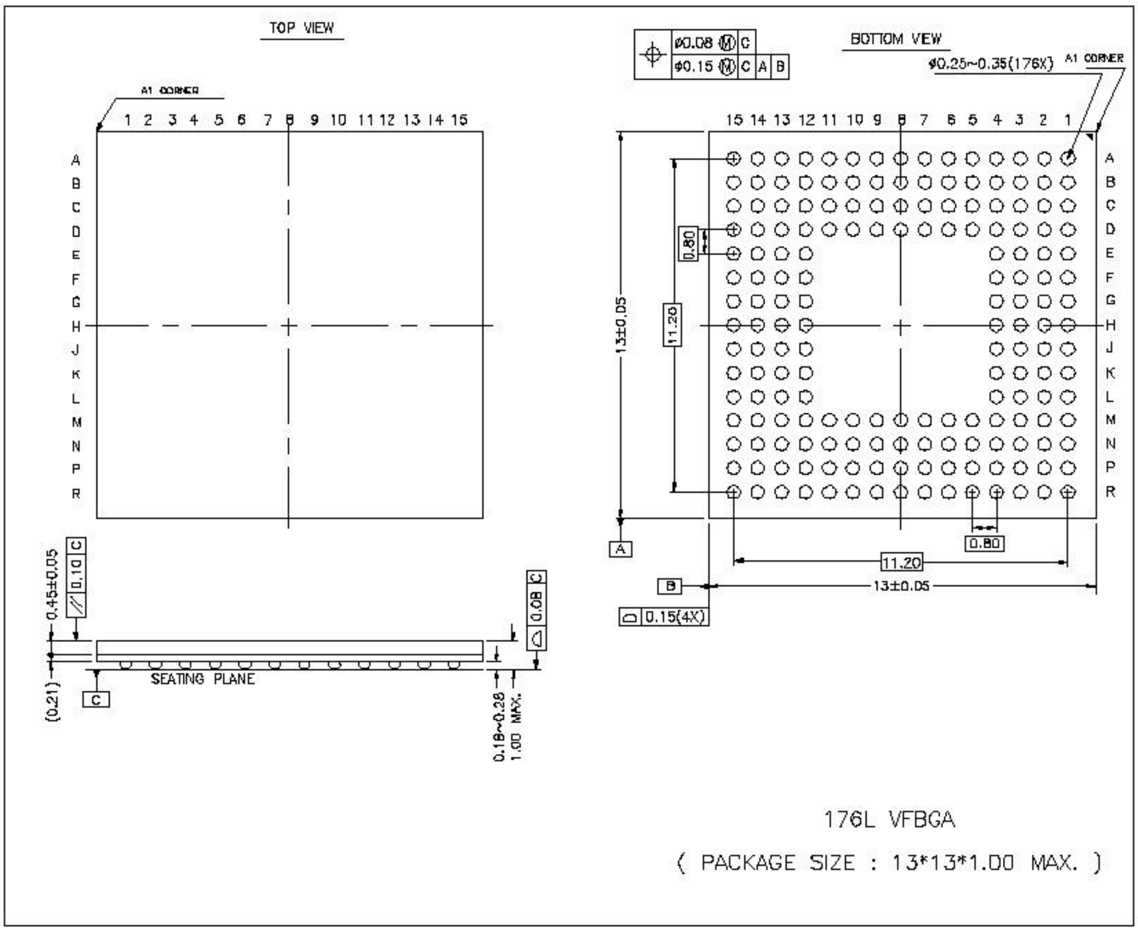
14 PACKAGE INFORMATION

14.1 160 LQFP Package



All dimentions in millimeters
For additional details see JEDEC MS-026

14.2 176 BGA Package



15 ORDERING INFORMATION

OXUF922 - LQ – B

Revision

Package Type –
LQ = 160 LQFP
VB = 176 VFBGA

Part Number

NOTES

This page has been intentionally left blank

CONTACT DETAILS

Oxford Semiconductor Ltd.
25 Milton Park
Abingdon
Oxfordshire
OX14 4SH
United Kingdom

Telephone:	+44 (0)1235 824900
Fax:	+44 (0)1235 821141
Sales e-mail:	sales@oxsemi.com
Web site:	http://www.oxsemi.com

DISCLAIMER