yet another PlayStationPortable Documentation

(not quite worth printing - yet)

December 23, 2006

this is the result of myself pasting together various freely available documents aswell as adding some of my own findings. have fun... additions and corrections welcome :)

THIS IS WORK IN PROGRESS! INFORMATION CONTAINED IN THIS DOCUMENT MAY BE MISSING, INCOMPLETE OR EVEN PLAIN WRONG! NO F**N' WARRANTY IMPLIED! IF THE USE OF THE INFORMATION CONTAINED HERE RESULTS IN ULTRA REALISTIC SMOKE EFFECTS, BRAIN DAMAGE OR LOSS OF PHYSICAL AND/OR MENTAL HEALTH PLEASE DON'T COME BACK AND SAY YOU HAVEN'T BEEN WARNED! YOU SHOULDN'T BE USING THIS IN THE FIRST PLACE!**

groepaz/hitmen (groepaz@gmx.net)
Hitmen-Console http://www.hitmen-console.org

Contents

1 Introductional Rant

If you don't know what programming a machine down to the metal is all about, go away! no really, this document is not for you! if you are seeking for advice on using existing solutions, such as SDKs or libraries, you will find little to none information that is of any use for you and you might only become frustrated by figuring out how little you know. If you however aren't afraid of numbers and want to dare jumping into the snake-pit of semi-accurate information based on guesswork done by a bunch of freaks - feel invited. this was made to give you what you need in the most compressed and visually pleasing form possible. *Stuff that matters.*

1.1 Things that are in this document

just about everything explicitly and specifically related to the PSP hard- and software internals and its programming. everything inside the box is subject to be documented, may it be relevant for actual programming or not. its ment as a reference for everyone who wants to know in all possible detail what makes this thing tick.

one more thing: please notice that this is a technical documentation which is presented for pure educational purposes and higher learning, and not a moral lesson. i have decided against leaving out any information since i believe that information by itself should not be crippled in any way. if you choose to abuse this information for any kind of illegal activities (**PLEASE DON'T!**) so be it, but don't bother me with it.

1.2 Things that are *not* in this document

several things were decided to not being put into this document because they didn't fit into the 'technical documentation' type of concept. They may be documented seperatly some time but not now and not here. These things are:

- ▷ Tips on Emulating the PSP on another Host system (this kind of information is only useful for a very limited number of people, and additionally might be highly confusing and/or misleading for those who are writing actual PSP programs)
- ▷ Instructions on using any tools that let you upload and execute code on the PSP, or any other development related tools **except** anything related to setting up and using gcc as a cross-compiler targeted to the PSP.
- ▷ anything related to gaming, cheat-codes and the like. (this is a tech-doc not a gaming FAQ!)
- ▷ detailed and/or complete sourcecode, except when a formal explanation would just over-complicate things. (this is a documentation, not a code library)
- ▷ anything related to playing/booting/copying pirated games (as you may have noticed, **we do not support piracy!**)

some of these may be arguable, so if you think they should be here - probably along the lines of the appendix - don't hesitate to write the chapter in question and send it to me. i might include it if you write it, but other than that i wont care (there is still enough other stuff to complete).

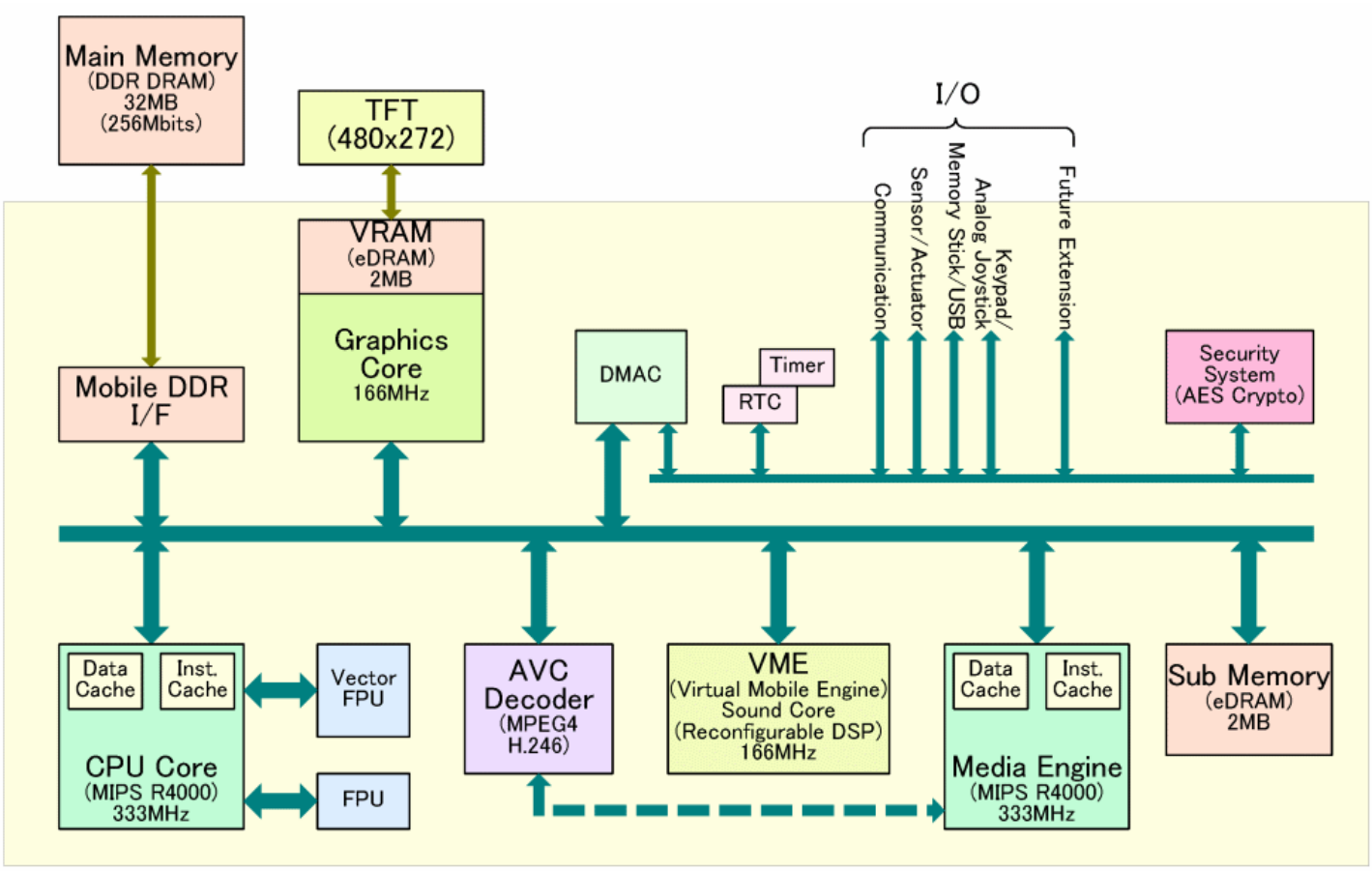
1.3 Conventions

- ▷ we count bits starting from 0, the most significant bit of a byte is bit 7. when visualising a byte the most significant bit comes first (left), and the least significant bit comes last (right).
- ▷ when dealing with 16- or 32 byte values all figures are in big endian byte order. this means that the most significant byte comes first (left), and the least significant byte comes last (right). notice that this is **not** the way values are actually handled by the allegrex cpu (since it is little endian).
- ▷ if known (from patents or other freely available sources) we use the same terminology as Sony does, in particular we try to use the same names and abbreviations for hardware registers, signals and the like as a weak attempt of providing consistency with other existing documentation.
- ▷ absolute memory addresses are shown as used in real world PSP Programs. For this matter we dont use physical adresses to avoid confusion for the majority of our readers.
- ▷ code snippets are in either real or pseudo C language. any logical or arithmetic expressions outside code snippets are loosely simelar to C notation according to the following table:

Description	Symbol
logical or bitwhise AND	&
logical or bitwhise OR	|
logical or bitwhise exclusive OR	^
logical or bitwhise NOT (inverse)	!
equality or assignment	=
addition	+
substraction	-
multiplication	*
division	/

please notice that -outside code- we do not make a difference between logical and bitwhise operations. if in doubt the operation is bitwhise, it should however be clearly visible from the context.

2 System Overview



2.1 Playstation Portable Main Unit

- ▷ Main CPU (System clock frequency 1~333MHz), MIPS32R2 'Allegrex' core (little endian)
- ▷ Media Engine CPU (System clock frequency 1~333MHz), MIPS32R2 core (little endian)
- ▷ Main Memory 32MB (DDR SDRAM)
- ▷ Flash Memory 32MB
- ▷ Embedded DRAM 4MB
- ▷ 4.3 inch wide 16:9 high resolution TFT LCD screen, 480 x 272 pixel, 16.77 million colors, backlight, Maximum luminance 180 / 130 / 80cd/m2 (when using battery pack), 200 / 180 / 80cd/m2 (when using AC adaptor)
- ▷ custom 'Universal Media Disc' (UMD), 60mm optical secured ROM disc with cartridge (1.8GB)
- ▷ Stereo Sound, two builtin Speakers
- ▷ Wireless LAN (IEEE802.11b, WiFi), a maximum of 16 PSP systems can be connected wirelessly through the ad-hoc mode, Typical indoor range of approx. 30m at 11Mbps and approx. 91m at 1Mbps. Typical outdoor range of approx. 120m at 11Mbps and approx. 460m at 1Mbps.
- ▷ USB 2.0 (mini-B)
- ▷ Memory Stick PRO Duo
- ▷ IrDA
- ▷ IR Remote (SIRCS)
- ▷ Main Connectors: Memory Stick Duo Slot, DC IN 5V connector, DC OUT connector, Headset connector, USB connector

- ▷ Keys/Switches: Directional buttons (Up/Down/Right/Left) , Analog Stick, Enter keys (Triangle, Circle, Cross, Square), Left, Right buttons, START button, SELECT button, HOME button, POWER/HOLD switch x, Display button, Sound button, Volume +/- buttons, Wireless LAN switch (ON/OFF), OPEN latch (UMD)
- ▷ Power Lithium-ion Battery
- ▷ AC Adaptor
- ▷ Recommended Retail Price 19,800 yen (20,790 yen tax inclusive), 249euro
- ▷ Dimensions Approximately 170mm (W) x 23mm (H) x 74mm (D)
- ▷ Weight Approximately 280g (including battery)

2.1.1 Modells/Revisions

- ▷ PSP1000 - Japan - Released December 12, 2004
- ▷ PSP1000K - Japan - Value Pack - Released December 12, 2004
- ▷ PSP1001 - US - Released March 24, 2005
- ▷ PSP1001K - US Value Pack
- ▷ PSP1002 - Australia/New Zealand - released September 1, 2005
- ▷ PSP1002K - EU Value Pack
- ▷ PSP1003 - UK - released September 1, 2005
- ▷ PSP1004 - Europe, Middle East & Africa - released September 1, 2005
- ▷ PSP1005 - Korea - Released May 10, 2005
- ▷ PSP1006 - Hong Kong/Singapore
- ▷ PSP1007 - Taiwan
- ▷ PSP1008 - Russia
- ▷ PSP1009 - China

2.1.1.1 Box Code

on the Box is a label looking like this:

```
PSP-1001 K
    120V
       A
```

the Letter in the 3rd Line indicates the Firmware that is preinstalled:

Boxcode	Firmware	Board
A	1.50	
B	1.51	
C,D,E	1.52	
F	2.00	
G	2.01	
H	2.50	
I	2.60	
J		
K		
L	2.81	TA-086

2.2 Game Specifications

- ▷ UMD Audio (profile name TBD), UMD Video (profile name TBD)
- ▷ Video Codec: H.264 / AVC MP Level3
- ▷ Audio Codec: ATRAC3plus, MP3
- ▷ Security (Encryption) 128bit AES
- ▷ Access control Region, Parental Control

2.3 Supplied accessories

- ▷ AC adaptor (PSP-100)
- ▷ Battery pack (PSP-110)

2.4 Separately Sold Accessories

2.4.1 Memory Stick Duo (PSP-M32)

- ▷ Copyright protection technology : MagicGateTM
- ▷ Capacity: 32MB to 32GB supported
- ▷ Recommended Retail Price 2,800 yen (2,940 yen tax inclusive)
- ▷ Dimensions: Approximately 20mm (W) x 1.6mm (H) x 31mm (D)
- ▷ Weight: Approximately 2g

2.4.2 AC adaptor (PSP-100)

- ▷ Specifications Rated input voltage : 100V - 240V 50/60Hz
- ▷ Rated voltage/electrical current output : 5V / 2.0A
- ▷ Recommended Retail Price 3,500 yen (3,675 yen tax inclusive)
- ▷ Dimensions: Approximately 76mm (W) x 22mm (H) x 46mm (D)
- ▷ Weight: Approximately 44g

2.4.3 Battery pack (PSP-110)

- ▷ Specifications Voltage/Capacity : 3.6V/1800mAh
- ▷ Recommended Retail Price 4,800 yen (5,040 yen tax inclusive)
- ▷ Dimensions: Approximately 52mm (W) x 12.5mm (H) x 36mm (D)
- ▷ Weight: Approximately 44g

2.4.4 Headphone with remote control (PSP-140(W))

- ▷ Remote Control : Play/Pause, FF, FR, Volume +/-, Hold switch
- ▷ Headphone : In-the-ear type headphone
- ▷ Recommended Retail Price 2,800 yen (2,940 yen tax inclusive)

2.4.5 Soft case and hand strap (PSP-170(B))

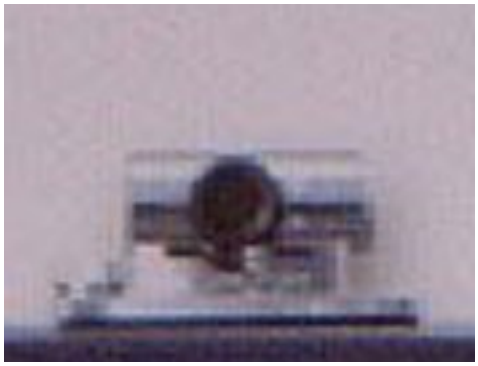
- ▷ Recommended Retail Price 2,000 yen (2,100 yen tax inclusive)
- ▷ Soft case: Dimensions: Approximately 195mm (W) x 7.5mm (H) x 108mm (D)
- ▷ Hand strap: Dimensions: Approximately 189mm (W) x 3.3mm (H) x 9mm (D)

2.4.6 USB microphone (PSP-240(X))

- ▷ monaural condenser microphone
- ▷ weight approximately 6 grams
- ▷ Dimensions: 50x10x10mm

- ▷ will feature support for GPS-enabled games such as a projected re-release or update of Hot Shot Golf, as well in Metal Gear Solid: Portable Ops.
- ▷ The GPS is set to be priced around ě6,000, appx. $54 USD.

2.4.8 Camera



- ▷ add-on will support a new video and VoIP chat service, as well as photo taking.
- ▷ The camera was released in Japan in early November 2006 for around ě5,000, appx. $44 USD

2.5 Development Hardware (DEM-100)

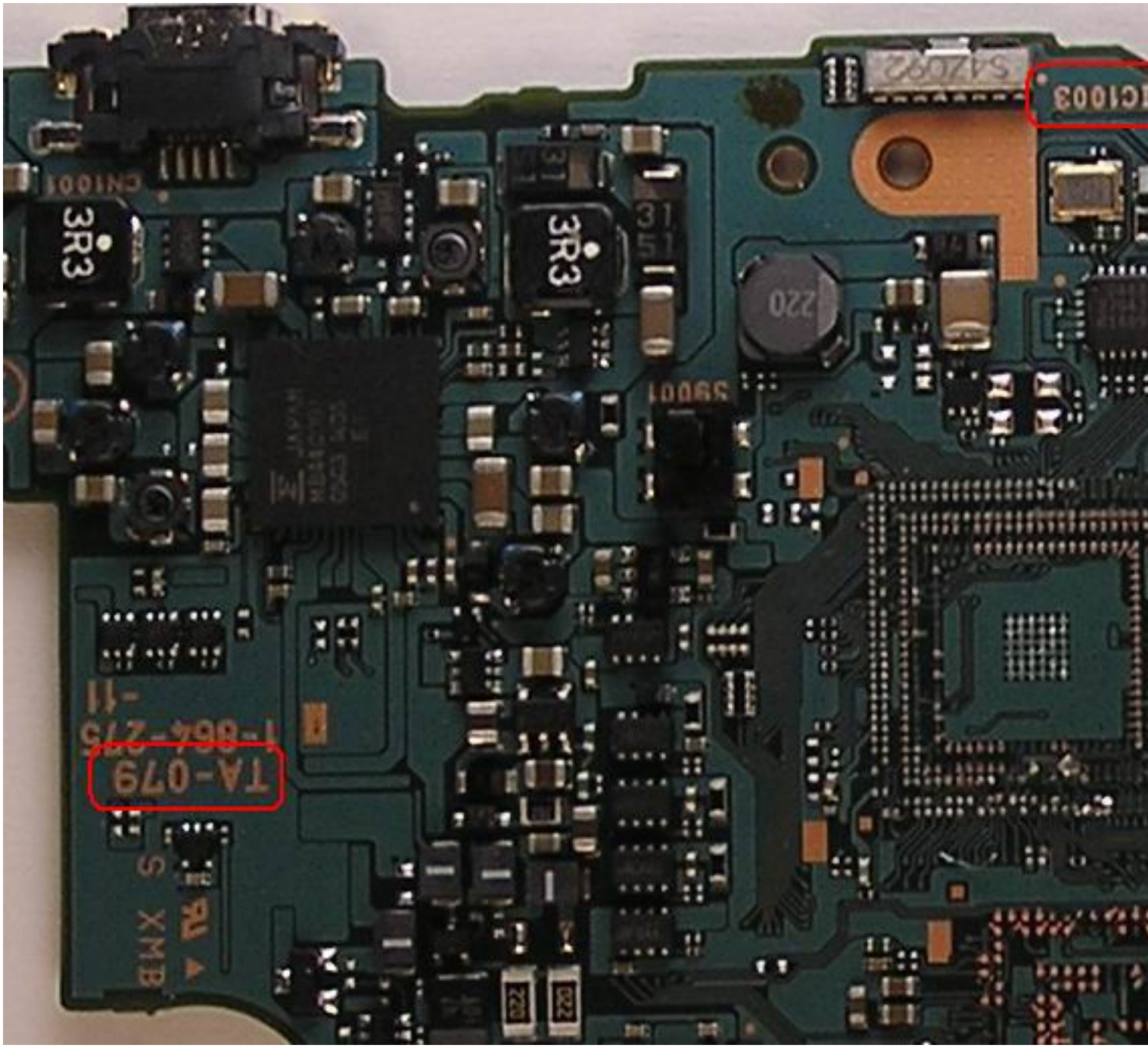
- ▷ 64MB Main Memory instead of 32MB

3 Hardware Overview

3.1 Mainboard

3.1.1 Revisions

3.1.1.1 TA-079 Flash/SDRAM: K5E5658HCM-D060 (3.0V/2.5V)

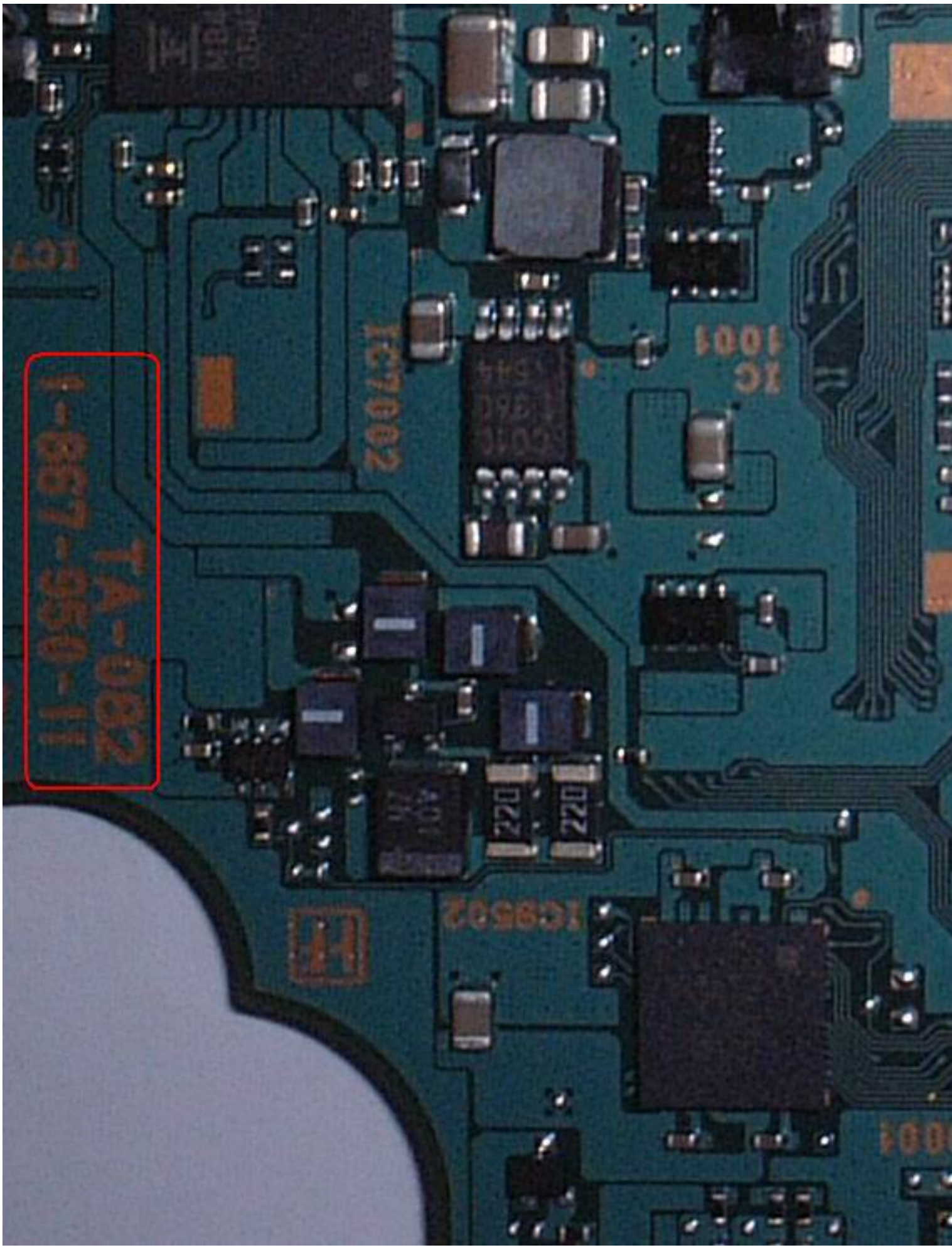


3.1.1.2 TA-080

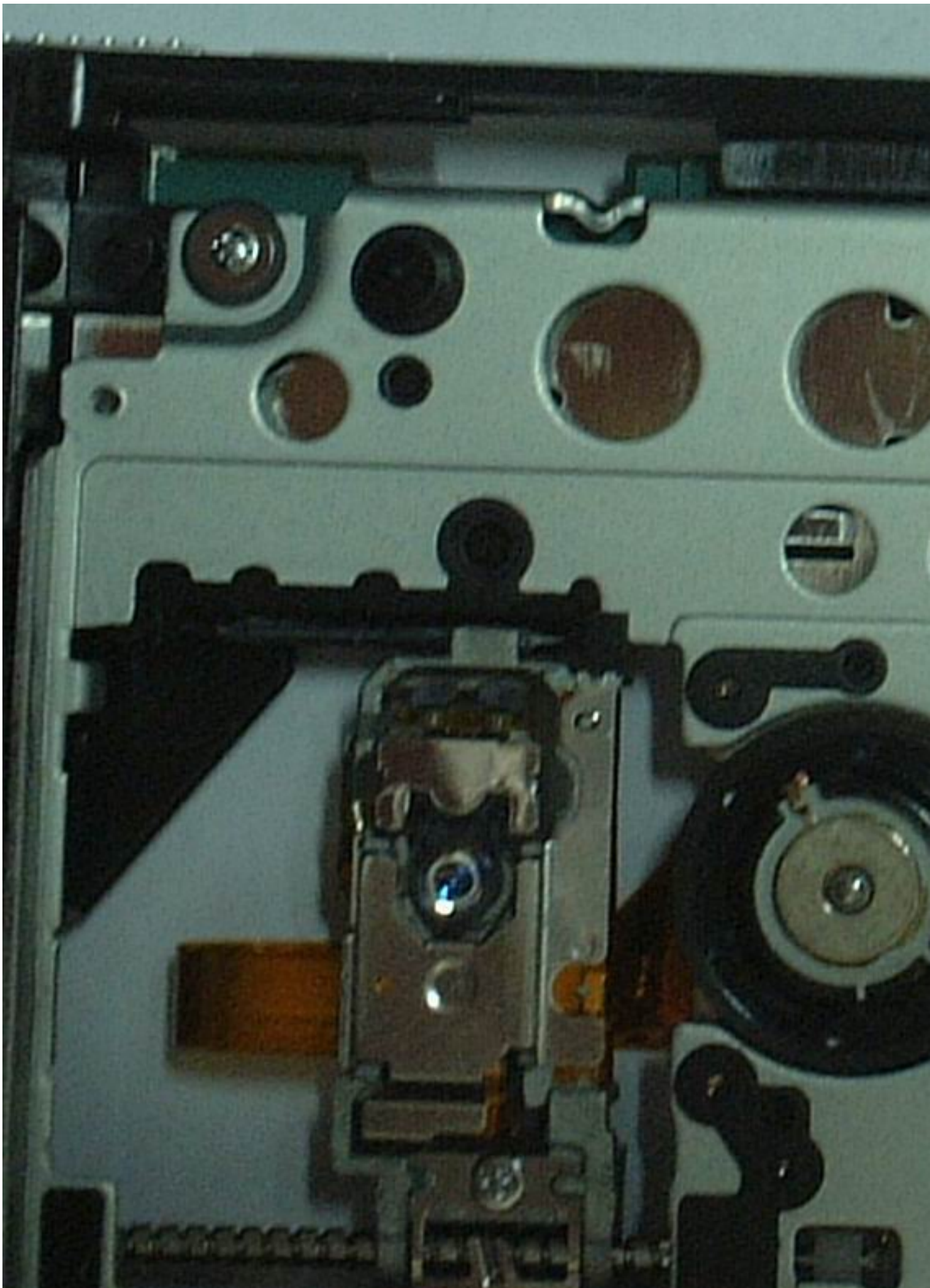
3.1.1.3 TA-081

3.1.1.4 TA-082

CPU Core : CXD2967GG
Media Engine : CXD5026-203GG
Flash/SDRAM: K5E5658ACM-D060 (1.8V/1.8V)


220
220

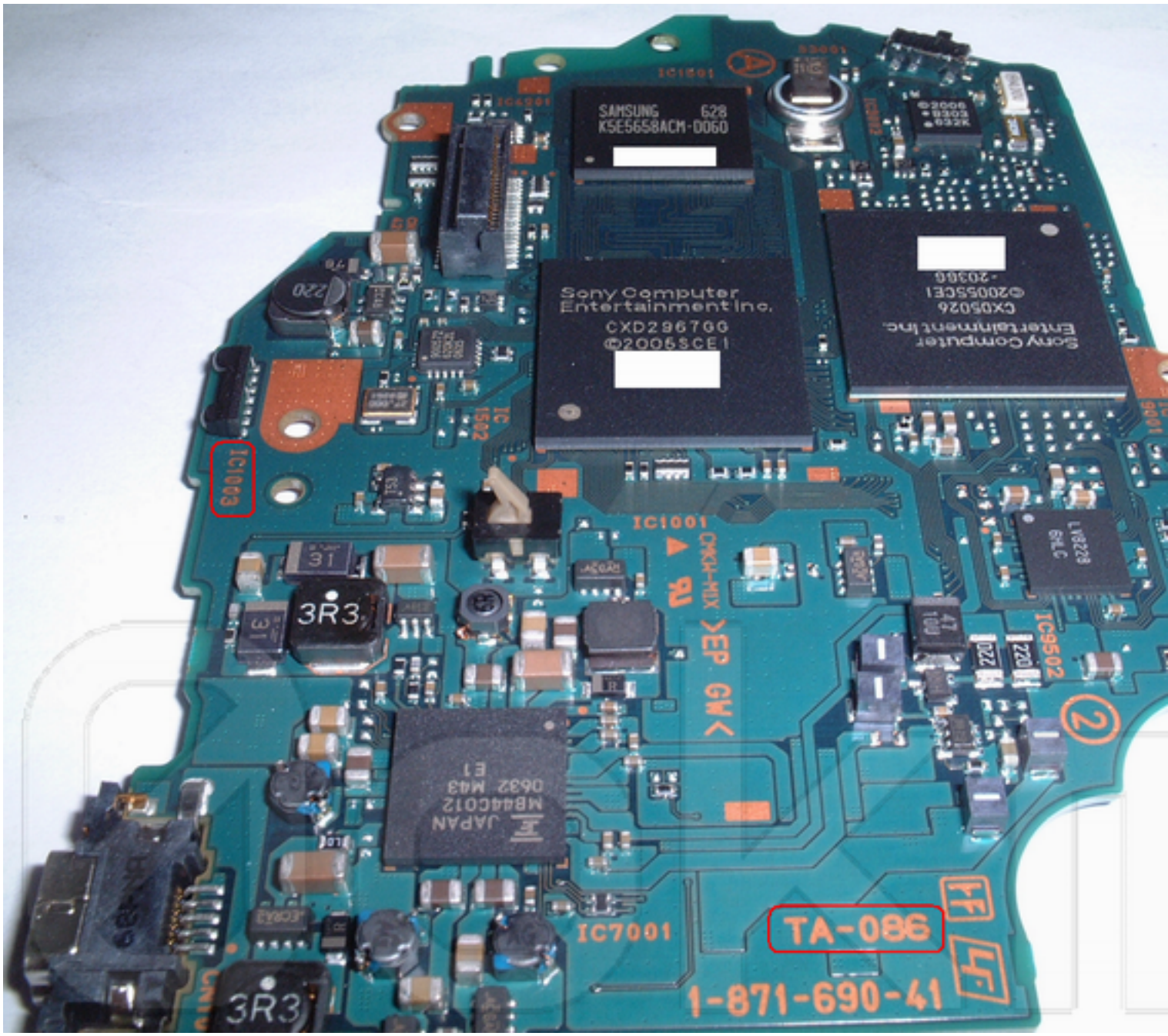
You can identify this Motherboard by opening the UMD door and looking for the IC1003 label:



3.1.1.5 TA-086 CPU Core : CXD2967GG

Media Engine : CXD5026-203GG

MCP : K5E5658ACM-D060 1.8V/1.8V



3.1.2 Semiconductors

▷ ?

SONY
A2707GL
504C28H

Manufacturer: Sony

Part Number: A2703GL

▷ ?

National Semiconductors
JM49SW
L00053B

▷ ?

SN10
5257
TI 52W
Z422

▷ ?

Fairchild Semiconductors
MB44C001
0507 M20
E1

Manufacturer: Fujitsu
Part Number: MB44C001

▷ ?

Freescale semiconductors
SC901583EP
MXAJ0450

or

Freescale semiconductors
SC901583EP
MXAA0445

Manufacturer: Motorola

Part Number: SC901583EP

▷ Graphics Processor Chip (MIPS CPU, 2MB embedded RAM)

Sony Computer Entertainment Inc.
CXD2962GG
(C)2004SCEI
509E90E
644031

or

Sony Computer Entertainment Inc.
CXD2962GG
(C)2004SCEI
445801E
629571

Manufacturer: Sony

Part Number: CXD2962GG

▷ 32MB NAND Flash + 32MB 333MHz DDR SDRAM

Samsung 501
K5E5658HCM-0060
BPL227AEE

or

Samsung 437
K5E5658HCM-D060
BPG036P2

Manufacturer: Samsung

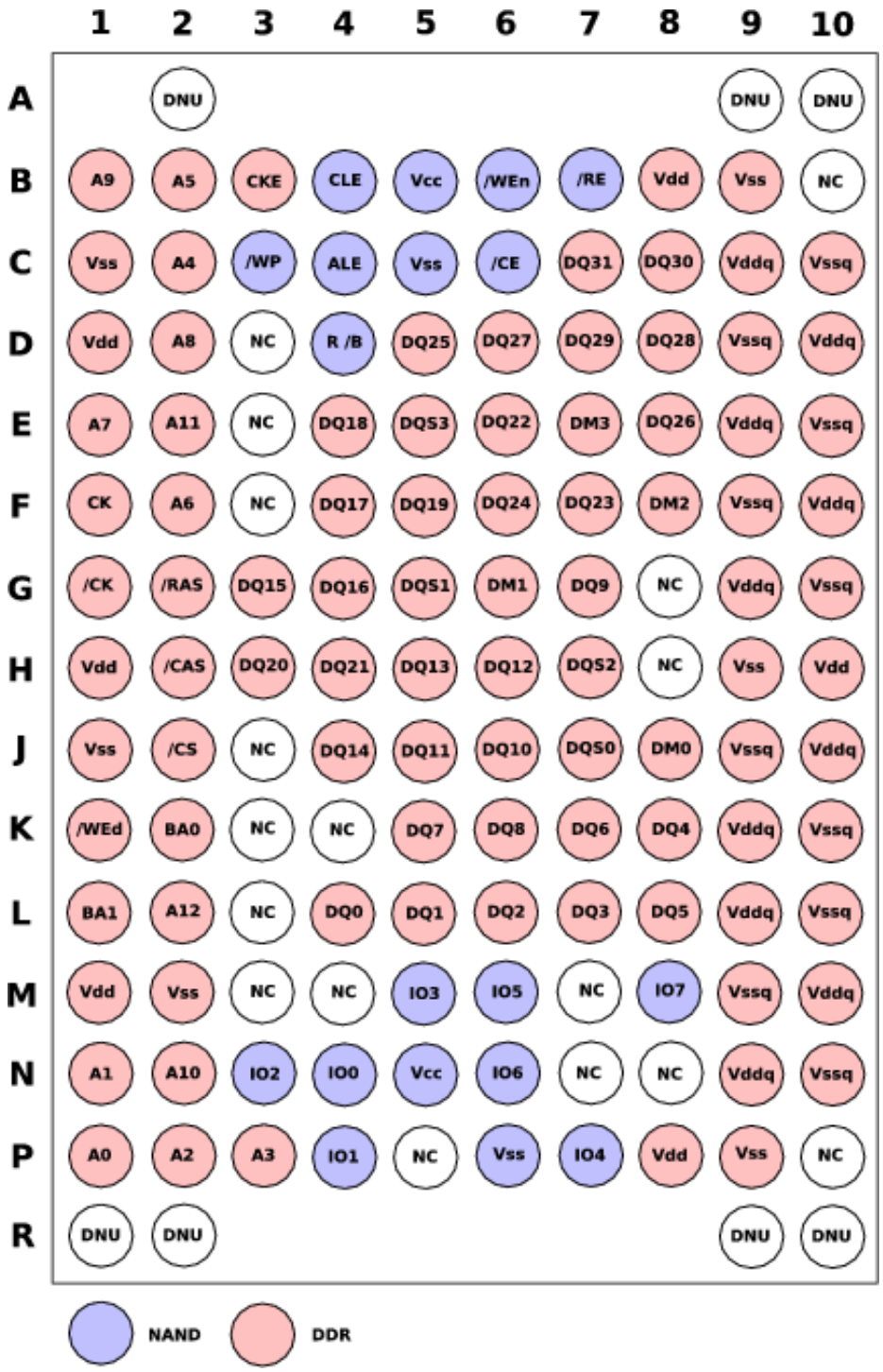
Part Number: K5E5658HCM-D060000

Package: FBGA(FL), 137 balls

Size: 10.5 x 13 x 1.4 mm

Description: Samsung 1st generation MCP 3.0V/2.5V 32MB 8 bit Uniform Block NAND Flash + 32MB 32 bit 6ns CL3 DDR SDRAM in a 137 ball FBGA(LF) package.

This is the pad layout on the PCB, in the PSP's natural orientation, with the main processor off to the left:


1 2 3 4 5 6 7 8 9 10
A DNU DNU DNU
B A9 A5 CKE CLE Vcc /WEn /RE Vdd Vss NC
C Vss A4 /WP ALE Vss /CE DQ31 DQ30 Vddq Vssq
D Vdd A8 NC R /B DQ25 DQ27 DQ29 DQ28 Vssq Vddq
E A7 A11 NC DQ18 DQS3 DQ22 DM3 DQ26 Vddq Vssq
F CK A6 NC DQ17 DQ19 DQ24 DQ23 DM2 Vssq Vddq
G /CK /RAS DQ15 DQ16 DQS1 DM1 DQ9 NC Vddq Vssq
H Vdd /CAS DQ20 DQ21 DQ13 DQ12 DQS2 NC Vss Vdd
J Vss /CS NC DQ14 DQ11 DQ10 DQS0 DM0 Vssq Vddq
K /WEd BA0 NC NC DQ7 DQ8 DQ6 DQ4 Vddq Vssq
L BA1 A12 NC DQ0 DQ1 DQ2 DQ3 DQ5 Vddq Vssq
M Vdd Vss NC NC IO3 IO5 NC IO7 Vssq Vddq
N A1 A10 IO2 IO0 Vcc IO6 NC NC Vddq Vssq
P A0 A2 A3 IO1 NC Vss IO4 Vdd Vss NC
R DNU DNU DNU DNU
NAND
DDR

PIN		description
CK, /CK	DDR	Differential System Clock
CKE		Clock enable
/CS		Chip Select (active low)
/RAS		Row Address Strobe (active low)
/CAS		Clolumn Address Strobe (active low)
/WEd		Write enable (active low)
A0 ... A12		Address Input
BA0 ... BA1		Bank Address Input
DM0 ... DM3		Input Data Mask
DQS0 ... DQS3		Data Strobe
DQ0 ... DQ31		Data Input/Output
Vdd		Power Supply
Vddq		Data out Power
Vss		Ground
Vssq		DQ Ground
/CE	NAND	Chip enable (active low)
/RE		Read enable (active low)
/WP		Write protection (active low)
/WEn		Write enable (active low)
ALE		Address Latch enable
CLE		Command Latch enable (command provided via IO0...IO7 and latched on rising edge of /WE)
R /B		Ready/Busy output (chip busy writing when low, can be read when high)
IO0 ... IO7		Data input/output
Vcc		+3.3V Power Supply
Vss		Ground
NC	-	not connected
DNU		do not use

Access protocol for flash chip is basically same as SAMSUNG's ordinal chip like K9F5608U0C but there exist difference.

Block address should be specified as 3byte length. After writing 1byte command with CLE=H, you must write 4byte address with ALE=H, 3byte block number with 1byte offset within the block. Also you should better to do this sequence not so slowly, or ignored

▷ Media Engine (MIPS CPU, 2MB embedded RAM)

Sony Computer
Entertainment Inc.
CXD1876
(C)2004SCEI
-102GG
508C10E
280221

Manufacturer: Sony

Part Number: CXD1876

▷ RTC, ...

(C)2004
BAR14
07KF

▷ ?

(C)2004
BAR12
46KC

▷ clock stuff

0450 27043 62592	or	0440 27043 62587

converts 27 MHz into:

- ▷ 36.83 MHz ?
- ▷ 22.58 MHz ?
- ▷ 27.00 MHz ?
- ▷ 48.00 MHz USB
- ▷ ? MHz ?

▷ Audio CODEC

Wolfson Microelectronics WM8973G HAAGCRY

Manufacturer: Wolfson Microelectronics

Part Number: WM8973G

3.1.3 other

▷ UMD laser flatcable

18 of 22 used. other 4 have pins allocated on the chip (unknown function)

▷ Crystal oscillator 27 MHz

2700L E52QA

▷ Crystal 4 MHz

[M] 4.00B

▷ Crystal 32.768 KHz

A507Y

3.2 WIFI Daughterboard

The WIFI module is mounted on the underside of the SIRCS / Memory Stick daughterboard. It appears to be a complete self-contained module built on its own PC board. It is completely covered by an aluminum shield which is embossed with the MAC address and several other numerical codes, including the apparent part number: SWU-BXJ154N. It also says "Sony Corporation, Made In China."

3.2.1 Semiconductors

▷ RF Transceiver

88W8010 NNB1

Manufacturer: Marvell Libertas

Part Number: 88W8010

▷ WEP and AES (802.11i) hardware security engine. (ARM9 Processor, 802.11b(g), QoS (802.11e))

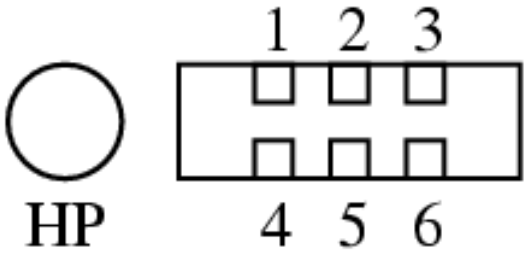
88W8380 BDK1

Manufacturer: Marvell Libertas

Part Number: 88W8380

3.3 Headphones/Remote Control

The headphone jack is a standard 3.5mm stereo, but there is also a small 6 pin connector next to it for the "remote control" that is included in the Value Pack. If we assume the following pin numbering (socket in the PSP as viewed from the outside):



Then the pinout is as follows (tip/ring/sleeve refers to the three parts of the stereo jack)

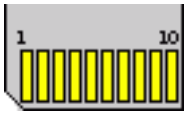
Pin	Wire color	Function
1	Brown	? Shield ? (GND) - (unused by standard Remote/Headphones)
2	Blue	Digital ground (GND)
3	Orange	TXD
4	Green	Sense? (+2.5V, seems to be controlled by PSP) - (unused by standard Remote/Headphones)
5	Yellow	+2.5V (0V when Plug isnt inserted) *1
6	Grey	RXD
Tip	Pink	Left audio (plus 600mv DC BIAS)
Ring	Red	Right audio
Sleeve	Black	Audio ground (GND)

*1) If a jack is plugged in and the PSP is on standby, the 2.5V output is always active, regardless of whether the external device replies to potential PSP queries or not (see below). In other words, when the PSP is on standby, external power is applied indefinitely to any remote device. This is done so the PSP may be woken up using a PLAY command(`0x0001`) over the serial bus.

If a jack is plugged in and the PSP is turned on, things become interesting:

- ▷ As soon as the PSP is turned on, voltage on pin 5 drops from +2.5V to 0V for about 0.5 seconds => this provides any external device plugged onto the remote port with a cold reset, as was previously identified
- ▷ After this reset phase, +2.5V is turned back on but it is only maintained if the remote device replies to a specific query within 5 secs.
- ▷ If no proper reply came from the external device within 5 secs, external voltage is turned off, until the PSP itself is powered off in

3.4 Memory Stick

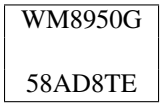


Pin	Signal	Description
1	VSS	
2	BS	IN, Serial protocol bus state signal
3	VCC	IN
4	DIO	IN/OUT, Serial protocol data signal
5		unused/reserved
6	INS	Stick insertion/extraction detect
7		unused/reserved
8	SCLK	IN, Serial protocol clock signal
9	VCC	
10	VSS	

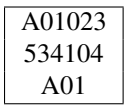
3.5 Talkman Microphone

The circuit board contains three ICs and several smaller 4 or 6-terminal devices:

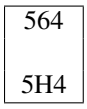
- ▷ A/D Converter

- ▷ USB Controller?

- ▷ ?

- ▷ It appears that the extra pins are power supply lines for the microphone circuit board.
- ▷ All five pins on the USB conector are used. Only four of these are defined for standard USB; the fifth should be NC.

4 CPU Overview

4.1 Registers

32 32bit General Purpose Integer Registers (R0-R31)

0	zero	wired zero
1	at	assembler temp
2	v0	return value
3	v1	
4	a0	argument registers
5	a1	
6	a2	
7	a3	
8	t0	caller saved (o32 old style names: default)
9	t1	
10	t2	
11	t3	
12	t4	caller saved
13	t5	
14	t6	
15	t7	
16	s0	callee saved
17	s1	
18	s2	
19	s3	
20	s4	
21	s5	
22	s6	
23	s7	
24	t8	caller saved
25	t9	
26	k0	kernel temporary
27	k1	
28	gp	global pointer
29	sp	stack pointer
30	fp/s8	frame pointer
31	ra	return address

4.2 Debug Registers

0	DRCNTL	Debug Register Control register
1	DEPC	Debug Exception PC register
2	DDATA0	Debug Data Monitor 0 and Monitor Data register
3	DDATA1	Debug Data Monitor 1 register
4	IBC	Instruction Breakpoint Control/Status register
5	DBC	Data Breakpoint Control/Status register
6	DR6	Reserved
7	DR7	Reserved
8	IBA	Instruction Breakpoint Address register
9	IBAM	Instruction Breakpoint Address Mask register
10	DR10	Reserved
11	DR11	Reserved
12	DBA	Data Breakpoint Address register
13	DBAM	Data Breakpoint Address Mask register
14	DBD	Data Breakpoint Data register
15	DBDM	Data Breakpoint Data Mask register
16	DR16	Undefined
17	DR17	Undefined
18	DR18	Undefined
19	DR19	Undefined
20	DR20	Undefined
21	DR21	Undefined
22	DR22	Undefined
23	DR23	Undefined
24	DR24	Undefined
25	DR25	Undefined
26	DR26	Undefined
27	DR27	Undefined
28	DR28	Undefined
29	DR29	Undefined
30	DR30	Undefined
31	DR31	Undefined

4.3 COP0 (System Control)

4.3.1 Status Registers (mfc/mtc)

0	-		not available (TLB)	
1	-		not available (TLB)	
2	-		not available (TLB)	
3	-		not available (TLB)	
4	-		not available (TLB context)	
5	-		not available (TLB)	
6	-		not available (TLB)	
7			?	
8	r	BadVaddr	virtual address of last error/exception	sysmem
9	r/w	Count	system counter	interruptman,sysmem
10	-		not available (TLB)	
11	r/w	Compare	counter comparison value	interruptman,sysmem
12	r/w	Status	system status	threadman, reboot, mewrapper,mebooterumdvideo,mebooter,loadcore,inte loadexec, exceptionman,sysmem
13	r/w	Cause	exception cause	threadman, mewrapper,mebooterumdvideo,mebooter,interruptman,except
14	r/w	EPC	exception program counter	loadcore,interruptman,exceptionman,sysmem
15	r	PRId	processor revision id	interruptman,sysmem
16	r/w	Config	configuration	utils,reboot,mewrapper,mebooterumdvideo,mebooter,loadcore,sysmem
17			?	
18			? Watch LO	
19			? Watch HI	
20	-		not available (TLB XContext)	
21	r	SCCode	Ssyscall-code<<2	interruptman
22	r	CPUId	CPU ID (0=Main, 1=ME)	threadman, sysreg, reboot,loadcore,interruptman,exceptionman,sysmem
23			?	
24			?	
25	r/w	EBase	virtual address of exception vector	threadman, exceptionman,sysmem
26			? Cache ECC	
27			? Cache Error	
28	r/w	TagLo	cache instruction register	utils,reboot,mewrapper,mebooterumdvideo,mebooter,sysmem
29	r/w	TagHi	cache instruction register	utils,reboot,mewrapper,mebooterumdvideo,mebooter,sysmem
30	r/w	ErrorEPC	error exception program counter	exceptionman,sysmem
31			?	

4.3.2 Control Registers (cfc/ctc)

num					used by	
0	COP0.EPC		context		EBase Handler, general exception handler, error handler,syscall handler	sysmem,interruptm ceptionman
1	COP0.EPC.err	0xbfc00000	context		error (HW,SW,NMI) exception handler, error handler	sysmem,exception
2	COP0.Status		context		EBase Handler, general exception handler,syscall handler	sysmem,interruptm ceptionman
3	COP0.Cause		context		EBase Handler, general exception handler,syscall handler	sysmem,interruptm ceptionman
4	GPR.v0		context	saved v0	general exception handler ,syscall handler	sysmem,interruptm ceptionman
5	GPR.v1		context	saved v1	general exception handler	sysmem,interruptm ceptionman
6	GPR.v0.err	0xbfc00000	context	saved v0	error (HW,SW,NMI) exception handler, EBase Handler	sysmem,exception
7	GPR.v1.err	0xbfc00000	context	saved v1	error (HW,SW,NMI) exception handler, EBase Handler	sysmem,exception
8	EXC_TABLE		vector table	Exception vector table addr	general exception handler	sysmem,exception
9	EXC_31_ERROR	0xbfc00000	vector	Error handler addr	error (HW,SW,NMI) exception handler	sysmem,exception
10	EXC_27_DEBUG	0xbfc01000	vector	Debug handler addr	debug exception handler	sysmem,exception
11	EXC_8_SYSCALL		vector	Syscall handler addr	EBase Handler, register/release exception handler functions	sysmem,exception
12	SC_TABLE		vector table	(1st) syscalls table addr	syscall handler	sysmem,interruptm
13	SC_MAX		int	(1st) max syscall code	syscall handler	sysmem,interruptm
14	GPR.sp.Kernel		context	Stackpointer Kernel		sysmem,threadma interruptman,
15	GPR.sp.User		context		syscall handler	sysmem,threadma interruptman,
16	CurrentTCB		context		syscall handler	sysmem,threadma interruptman,
17	?				?	sysmem
18	NMI_TABLE	0xbfc00000	vector table	NMI vector table addr	error handler	sysmem,exception
19	COP0.Status.err	0xbfc00000	context		EBase Handler, error (HW,SW,NMI) exception handler	sysmem,exception
20	COP0.Cause.err	0xbfc00000	context		error (HW,SW,NMI) exception handler	sysmem,exception
21	?				?	sysmem
22	?				?	sysmem
23	? GPR.v0		? context		?	sysmem
24	? GPR.v1		? context		?	sysmem
25	PROFILER_BASE		vector	profiler hw base addr	general exception handler	sysmem,threadma ruptman, exceptio
26	GPR.v0.dbg	0xbfc01000	context		debug exception handler	sysmem,exception
27	GPR.v1.dbg	0xbfc01000	context		debug exception handler	sysmem,exception
28	DBGENV	0xbfc01000	vector	debug handler env addr	debug exception handler	sysmem,exception
29	?				?	sysmem
30	?				?	sysmem
31	?				?	sysmem

4.4 COP1 (FPU)

32 32bit General Purpose Floatingpoint Registers (FPR0-FPR31)

4.4.1 Status Registers (mfc/mtc)

0			vshmain,video_plugin,update_plugin,sysreg,semawm,savedata_plugin,photo_plugin, paf,pafmini,osk_plugin,opening_plugin,netplay_client_plugin,music_plugin,msvideo_plugin,lcdc,impose_plugin,auth_plugin,c
1			vshmain,video_plugin,update_plugin,sysreg,sysclib,savedata_utility,savedata_plugin,power,photo_plugin, paf,pafmini,osk_plugin,opening_plugin,netplay_server_utility,netconf_plugin,music_plugin,msvideo_plugin,lcdc,impose_plugi
2			video_plugin,sysreg,photo_plugin, paf,pafmini,osk_plugin,music_plugin,msvideo_plugin,lcdc,
3			video_plugin,sysreg,photo_plugin, paf,pafmini,music_plugin,
4			vshmain,video_plugin,paf,pafmini,dialogmain,
5			video_plugin,sysreg,photo_plugin, paf,pafmini,
6			paf,pafmini,
7			
8			video_plugin,paf,pafmini,
9			paf,pafmini,
10			
11			
12			vshmain,video_plugin,update_plugin,sysconf_plugin,sysclib,savedata_utility,savedata_plugin,savedata_auto_dialog,photo_plug paf,pafmini,opening_plugin,netplay_client_plugin,netconf_plugin,music_plugin,msvideo_plugin,auth_plugin,common_gui,dia
13			vshmain,update_plugin,sysconf_plugin,savedata_utility,savedata_plugin,photo_plugin, paf,pafmini,osk_plugin,netplay_client_plugin,netconf_plugin,music_plugin,msvideo_plugin,game_plugin,common_gui,
14			vshmain,video_plugin,sysconf_plugin,savedata_utility,savedata_plugin,photo_plugin, paf,pafmini,music_plugin,msvideo_plugin,game_plugin,
15			syscon
16			paf,pafmini,
17			
18			
19			
20			vshmain,video_plugin,sysconf_plugin,savedata_plugin,photo_plugin, paf,pafmini,osk_plugin,music_plugin,msvideo_plugin,im
21			video_plugin,photo_plugin, paf,pafmini,osk_plugin,music_plugin,msvideo_plugin,game_plugin,common_gui,
22			sysconf_plugin,photo_plugin, paf,pafmini,music_plugin,msvideo_plugin,game_plugin,common_gui,
23			photo_plugin, paf,pafmini,
24			paf,pafmini,
25			paf,pafmini,
26			
27			
28			
29			
30			
31			

4.4.2 Control Registers (cfc/ctc)

0	FIR	Floating Point Implementation Register	sysmem
1	FCR1		
2	FCR2		
3	FCR3		
4	FCR4		
5	FCR5		
6	FCR6		interrupt handler
7	FCR7		
8	FCR8		
9	FCR9		
10	FCR10		
11	FCR11		
12	FCR12		
13	FCR13		
14	FCR14		
15	FCR15		
16	FCR16		
17	FCR17		
18	FCR18		
19	FCR19		
20	FCR20		
21	FCR21		
22	FCR22		
23	FCR23		
24	FCR24		
25	FCCR	Floating Point Condition Codes Register	
26	FEXR	Floating Point Exceptions Register	
27	FCR27		
28	FENR	Floating Point Enables Register	
29	FCR29		
30	FCR30		
31	FCSR	Floating Point Control and Status Register	sysmem, interruptman, paf, pafmini

4.5 COP2 (VFPU)

The psp's VFPU (Vector Floating Point Unit) is a coprocessor that can perform quite a few useful operations. The main purpose of it is vector and matrix processing, but it also supports trigonemtric functions and other mathematical operations, conversions, and mathematical constants.

4.5.1 Registers

The VFPU has 128 single precision floating point (IEEE 754) registers (VFR0-VFR127), but they are arranged and accessed in various ways that make it very flexible. Many of the instructions for the VFPU support operations on:

- ▷ a single register
- ▷ a pair of registers
- ▷ three registers
- ▷ four regiters
- ▷ 2x2 matrix
- ▷ 3x3 matrix
- ▷ 4x4 matrix

And if that weren't enough, it can work with matrices in normal or transposed orders.

The registers are grouped into 8 blocks of 16 registers each. This gives you enough room to work with 8 4x4 matrices, 8 3x3 matrices, 32 2x2 matrices. Or you can store up to 32 quad vectors, 40 triple vectors, 64 paired vectors, or 128 single values.

The register names you use on the VFPU depends highly on the instruction being performed, and can quickly become a nightmare when trying to figure out how to access or modify certain registers. Register names are numbered with 3 digits: Matrix, Column and Row. The tables below show how single, pair, triple, quad and matrix registers are mapped within a single 16 register block

single Register

S000	S010	S020	S030
S001	S011	S021	S031
S002	S012	S022	S032
S003	S013	S023	S033

Quad Columns

C000	C010	C020	C030
....	....	....	....
....	....	....	....
....	....	....	....

Quad Rows

R000	....	....	....
R001	....	....	....
R002	....	....	....
R003	....	....	....

4*4 Matrix

M000	....	....	....
....	....	....	....
....	....	....	....
....	....	....	....

4*4 Transpose Matrix

E000	....	....	....
....	....	....	....
....	....	....	....
....	....	....	....

Triple Columns (1)

C000	C010	C020	C030
....	....	....	....
....	....	....	....

Triple Columns (2)

C001	C011	C021	C031
....	....	....	....
....	....	....	....

Triple Rows (1)

R000	....	....	
R001	....	....	
R002	....	....	
R003	....	....	

Triple Rows (2)

	R010	....	....
	R011	....	....
	R012	....	....
	R013	....	....

3*3 Matrix (1)

M000	....	....	
....	....	....	
....	....	....	

3*3 Matrix (2)

M001	....	....	
....	....	....	
....	....	....	

3*3 Matrix (3)

	M10	....	....
	....	....	....
	....	....	....

3*3 Matrix (4)

	M011	....	....
	....	....	....
	....	....	....

3*3 Transpose Matrix (1)

E000	....	....	
....	....	....	
....	....	....	

3*3 Transpose Matrix (2)

E001	....	....	
....	....	....	
....	....	....	

3*3 Transpose Matrix (3)

	E10	....	....
	....	....	....
	....	....	....

3*3 Transpose Matrix (4)

	E011	....	....
	....	....	....
	....	....	....

Pair Columns

C000	C010	C020	C030
....	....	....	....
C002	C012	C022	C032
....	....	....	....

Pair Rows

R000	....	R020	....
R001	....	R021	....
R002	....	R022	....
R003	....	R023	....

2*2 Matrix **2*2 Transpose Matrix**

M000	M020 ....
....	....
M002	M022 ....
....	....

E000	E020 ....
....	....
E002	E022 ....
....	....

Repeat all of the above with the other 7 blocks of registers. Just change the first digit of the register names to work on a different set

4.5.2 Extra Registers

128	VFPU_PFXS	Source prefix stack
129	VFPU_PFXT	Target prefix stack
130	VFPU_PFXD	Destination prefix stack
131	VFPU_CC	Condition information
132	VFPU_INF4	VFPU internal information 4
133	VFPU_RSV5	Not used (reserved)
134	VFPU_RSV6	Not used (reserved)
135	VFPU_REV	VFPU revision information
136	VFPU_RCX0	Pseudorandom number generator information 0
137	VFPU_RCX1	Pseudorandom number generator information 1
138	VFPU_RCX2	Pseudorandom number generator information 2
139	VFPU_RCX3	Pseudorandom number generator information 3
140	VFPU_RCX4	Pseudorandom number generator information 4
141	VFPU_RCX5	Pseudorandom number generator information 5
142	VFPU_RCX6	Pseudorandom number generator information 6
143	VFPU_RCX7	Pseudorandom number generator information 7

4.6 Instruction Format

Every CPU instruction consists of a single word (32 bits) aligned on a word boundary and the major instruction formats are shown here:

▷ I-Type (Immediate)

op	rs	rt	immediate
oooooo	sssss	ttttt	iiiiiiiiiiiiiiii
31 26	25 21	20 16	15 0

▷ J-Type (Jump)

op	target
oooooo	tttttttttttttttttttttttttt
31 26	25 0

▷ R-Type (Register)

op	rs	rt	rd	shamt	func
oooooo	sssss	ttttt	ddddd	aaaaa	ffffff
31 26	25 21	20 16	15 11	10 6	5 0

where:

op	6-bit operation code
rs	5-bit source register specifier
rt	5-bit target (source/destination) register or branch condition
immediate	16-bit immediate, branch displacement or address displacement
target	26-bit jump target address
rd	5-bit destination register specifier
shamt	5-bit shift amount
func	6-bit function field

4.7 MIPS Instructions

Mnemonic	Opcode	op rs rt offset	Description
lw rt, offset(rs)	0x8c000000	100011 sssss ttttt oooooooooooooooo	LoadWord Relative to Address in General Purpose R
sw rt, offset(rs)	0xac000000	101011 sssss ttttt oooooooooooooooo	StoreWord Relative to Address in General Purpose R

Mnemonic	Opcode	op	rs	rt	immediate	Description
addiu rt,rs,immediate	0x24000000	001001	sssss	ttttt	iiiiiiiiiiiiiiii	Add Immediate Unsigned Word

4.7.1 lw

lw **LoadWord Relative to Address in General Purpose Register**

```
%rt <- word_at_address (offset + %base)
```

lw %rt, offset(%base)	

%rt	GPR Target Register (0...31)
%base	GPR, specifies Source Address Base
offset	signed Offset added to Source Address Base

4.7.2 sw

sw **StoreWord Relative to Address in General Purpose Register**

```
word_at_address (offset + %base) <- %rt
```

sw %rt, offset(%base)	

%rt	GPR Target Register (0...31)
%base	GPR, specifies Source Address Base
offset	signed Offset added to Source Address Base

4.7.3 addiu

addiu **Add Immediate Unsigned Word**

```
%rt <- %rs + sign_extended(immediate)
```

addiu %rt, %rs, immediate	

%rt	GPR Target Register (0...31)
%rs	GPR Source Register (0...31)
immediate	value added to Source Register

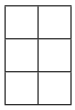
4.8 Allegrex Instructions

Mnemonic	Opcode	op	rs	rt	rd	shamt	func	Description
halt	0x70000000	011100	00000	00000	00000	00000	000000	halt execution until next interrupt
mfic rt,rd	0x70000024	011100	00000	ttttt	ddddd	00000	100100	move from IC (Interrupt) register
mtic rt,rd	0x70000026	011100	00000	ttttt	ddddd	00000	100110	move to IC (Interrupt) register

4.8.1 halt

halt halt execution until next interrupt

halt	

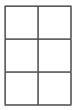


▷ this instruction is used in the idle-thread of the kernel, probably to initiate power saving

4.8.2 mfic / mtic

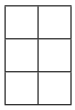
mfic move from IC (Interrupt) register

mfic rt,rd	



mtic move to IC (Interrupt) register

mtic rt,rd	



▷ `mfic $v0, zero`
to save the interrupt state in v0

▷ `mtic zero, zero`
to disable them

▷ `mtic $a0, zero`
to renable based on the original mask in a0

4.9 VFPU Instructions

Mnemonic	Opcode	op	rs	rt	offset	c	Description
lv.q rt, offset(rs)	0xd8000000	110110	sssss	ttttt	oooooooooooooo	0	t LoadVector.Quadword Relative to Addres
sv.q rt, offset(rs), wb	0xf8000000	111110	sssss	ttttt	oooooooooooooo	w	t StoreVector.Quadword Relative to Addres

Mnemonic	Opcode	op	rt	rs	rd	Description
vadd.s rd,rs,rt	0x60000000	011000	000 ttttttt	0 sssssss	0 ddddddd	
vadd.p rd,rs,rt	0x60000080	011000	000 ttttttt	0 sssssss	1 ddddddd	
vadd.t rd,rs,rt	0x60008000	011000	000 ttttttt	1 sssssss	0 ddddddd	
vadd.q rd,rs,rt	0x60008080	011000	000 ttttttt	1 sssssss	1 ddddddd	
vsub.s rd,rs,rt	0x60800000	011010	000 ttttttt	0 sssssss	0 ddddddd	
vsub.p rd,rs,rt	0x60800080	011010	000 ttttttt	0 sssssss	1 ddddddd	
vsub.t rd,rs,rt	0x60808000	011010	000 ttttttt	1 sssssss	0 ddddddd	
vsub.q rd,rs,rt	0x60808080	011010	000 ttttttt	1 sssssss	1 ddddddd	
vdiv.s rd,rs,rt	0x63800000	011000	111 ttttttt	0 sssssss	0 ddddddd	
vdiv.p rd,rs,rt	0x63800080	011000	111 ttttttt	0 sssssss	1 ddddddd	
vdiv.t rd,rs,rt	0x63808000	011000	111 ttttttt	1 sssssss	0 ddddddd	
vdiv.q rd,rs,rt	0x63808080	011000	111 ttttttt	1 sssssss	1 ddddddd	
vmul.s rd,rs,rt	0x64000000	011001	000 ttttttt	0 sssssss	0 ddddddd	
vmul.p rd,rs,rt	0x64000080	011001	000 ttttttt	0 sssssss	1 ddddddd	
vmul.t rd,rs,rt	0x64008000	011001	000 ttttttt	1 sssssss	0 ddddddd	
vmul.q rd,rs,rt	0x64008080	011001	000 ttttttt	1 sssssss	1 ddddddd	
vdot.p rd,rs,rt	0x64800080	011001	001 ttttttt	0 sssssss	1 ddddddd	
vdot.t rd,rs,rt	0x64808000	011001	001 ttttttt	1 sssssss	0 ddddddd	
vdot.q rd,rs,rt	0x64808080	011001	001 ttttttt	1 sssssss	1 ddddddd	
vhdp.p rd,rs,rt	0x66000080	011001	100 ttttttt	0 sssssss	1 ddddddd	
vhdp.t rd,rs,rt	0x66008000	011001	100 ttttttt	1 sssssss	0 ddddddd	
vhdp.q rd,rs,rt	0x66008080	011001	100 ttttttt	1 sssssss	1 ddddddd	
vmin.s rd,rs,rt	0x6D000000	011011	010 ttttttt	0 sssssss	0 ddddddd	
vmin.p rd,rs,rt	0x6D000080	011011	010 ttttttt	0 sssssss	1 ddddddd	
vmin.t rd,rs,rt	0x6D008000	011011	010 ttttttt	1 sssssss	0 ddddddd	
vmin.q rd,rs,rt	0x6D008080	011011	010 ttttttt	1 sssssss	1 ddddddd	
vmax.s rd,rs,rt	0x6D800000	011011	011 ttttttt	0 sssssss	0 ddddddd	
vmax.p rd,rs,rt	0x6D800080	011011	011 ttttttt	0 sssssss	1 ddddddd	
vmax.t rd,rs,rt	0x6D808000	011011	011 ttttttt	1 sssssss	0 ddddddd	
vmax.q rd,rs,rt	0x6D808080	011011	011 ttttttt	1 sssssss	1 ddddddd	
vabs.s rd,rs	0xd0010000	110100	000 0000001	0 sssssss	0 ddddddd	
vabs.p rd,rs	0xd0010080	110100	000 0000001	0 sssssss	1 ddddddd	
vabs.t rd,rs	0xd0018000	110100	000 0000001	1 sssssss	0 ddddddd	
vabs.q rd,rs	0xd0018080	110100	000 0000001	1 sssssss	1 ddddddd	
vneg.s rd,rs	0xd0020000	110100	000 0000010	0 sssssss	0 ddddddd	
vneg.p rd,rs	0xd0020080	110100	000 0000010	0 sssssss	1 ddddddd	
vneg.t rd,rs	0xd0028000	110100	000 0000010	1 sssssss	0 ddddddd	
vneg.q rd,rs	0xd0028080	110100	000 0000010	1 sssssss	1 ddddddd	
vidt.p rd	0xd0030080	110100	000 0000011	0 0000000	1 ddddddd	
vidt.t rd	0xd0038000	110100	000 0000011	1 0000000	0 ddddddd	
vidt.q rd	0xd0038080	110100	000 0000011	1 0000000	1 ddddddd	
vzero.s rd	0xd0060000	110100	000 0000110	0 0000000	0 ddddddd	SetVectorZero.Single
vzero.p rd	0xd0060080	110100	000 0000110	0 0000000	1 ddddddd	SetVectorZero.Pair
vzero.t rd	0xd0068000	110100	000 0000110	1 0000000	0 ddddddd	SetVectorZero.Triple
vzero.q rd	0xd0068080	110100	000 0000110	1 0000000	1 ddddddd	SetVectorZero.Quad
vone.s rd	0xd0070000	110100	000 0000111	0 0000000	0 ddddddd	SetVectorOne.Single
vone.p rd	0xd0070080	110100	000 0000111	0 0000000	1 ddddddd	SetVectorOne.Pair
vone.t rd	0xd0078000	110100	000 0000111	1 0000000	0 ddddddd	SetVectorOne.Triple
vone.q rd	0xd0078080	110100	000 0000111	1 0000000	1 ddddddd	SetVectorOne.Quad
vrcp.s rs,rd	0xd0100000	110100	000 0010000	0 sssssss	0 ddddddd	
vrcp.p rs,rd	0xd0100080	110100	000 0010000	0 sssssss	1 ddddddd	
vrcp.t rs,rd	0xd0108000	110100	000 0010000	1 sssssss	0 ddddddd	
vrcp.q rs,rd	0xd0108080	110100	000 0010000	1 sssssss	1 ddddddd	
vrsq.s rs,rd	0xd0110000	110100	000 0010001	0 sssssss	0 ddddddd	
vrsq.p rs,rd	0xd0110080	110100	000 0010001	0 sssssss	1 ddddddd	
vrsq.t rs,rd	0xd0118000	110100	000 0010001	1 sssssss	0 ddddddd	
vrsq.q rs,rd	0xd0118080	110100	000 0010001	1 sssssss	1 ddddddd	

vsin.s rs,rd	0xd0120000	110100 000 0010010 0 sssssss 0 ddddddd	
vsin.p rs,rd	0xd0120080	110100 000 0010010 0 sssssss 1 ddddddd	
vsin.t rs,rd	0xd0128000	110100 000 0010010 1 sssssss 0 ddddddd	
vsin.q rs,rd	0xd0128080	110100 000 0010010 1 sssssss 1 ddddddd	
vcos.s rs,rd	0xd0130000	110100 000 0010011 0 sssssss 0 ddddddd	
vcos.p rs,rd	0xd0130080	110100 000 0010011 0 sssssss 1 ddddddd	
vcos.t rs,rd	0xd0138000	110100 000 0010011 1 sssssss 0 ddddddd	
vcos.q rs,rd	0xd0138080	110100 000 0010011 1 sssssss 1 ddddddd	
vexp2.s rs,rd	0xd0140000	110100 000 0010100 0 sssssss 0 ddddddd	
vexp2.p rs,rd	0xd0140080	110100 000 0010100 0 sssssss 1 ddddddd	
vexp2.t rs,rd	0xd0148000	110100 000 0010100 1 sssssss 0 ddddddd	
vexp2.q rs,rd	0xd0148080	110100 000 0010100 1 sssssss 1 ddddddd	
vlog2.s rs,rd	0xd0150000	110100 000 0010101 0 sssssss 0 ddddddd	
vlog2.p rs,rd	0xd0150080	110100 000 0010101 0 sssssss 1 ddddddd	
vlog2.t rs,rd	0xd0158000	110100 000 0010101 1 sssssss 0 ddddddd	
vlog2.q rs,rd	0xd0158080	110100 000 0010101 1 sssssss 1 ddddddd	
vsqrt.s rs,rd	0xd0160000	110100 000 0010110 0 sssssss 0 ddddddd	
vsqrt.p rs,rd	0xd0160080	110100 000 0010110 0 sssssss 1 ddddddd	
vsqrt.t rs,rd	0xd0168000	110100 000 0010110 1 sssssss 0 ddddddd	
vsqrt.q rs,rd	0xd0168080	110100 000 0010110 1 sssssss 1 ddddddd	
vasin.s rs,rd	0xd0170000	110100 000 0010111 0 sssssss 0 ddddddd	
vasin.p rs,rd	0xd0170080	110100 000 0010111 0 sssssss 1 ddddddd	
vasin.t rs,rd	0xd0178000	110100 000 0010111 1 sssssss 0 ddddddd	
vasin.q rs,rd	0xd0178080	110100 000 0010111 1 sssssss 1 ddddddd	
vnrcp.s rs,rd	0xd0180000	110100 000 0011000 0 sssssss 0 ddddddd	
vnrcp.p rs,rd	0xd0180080	110100 000 0011000 0 sssssss 1 ddddddd	
vnrcp.t rs,rd	0xd0188000	110100 000 0011000 1 sssssss 0 ddddddd	
vnrcp.q rs,rd	0xd0188080	110100 000 0011000 1 sssssss 1 ddddddd	
vnsin.s rs,rd	0xd01a0000	110100 000 0011010 0 sssssss 0 ddddddd	
vnsin.p rs,rd	0xd01a0080	110100 000 0011010 0 sssssss 1 ddddddd	
vnsin.t rs,rd	0xd01a8000	110100 000 0011010 1 sssssss 0 ddddddd	
vnsin.q rs,rd	0xd01a8080	110100 000 0011010 1 sssssss 1 ddddddd	
vrexp2.s rs,rd	0xd01c0000	110100 000 0011100 0 sssssss 0 ddddddd	
vrexp2.p rs,rd	0xd01c0080	110100 000 0011100 0 sssssss 1 ddddddd	
vrexp2.t rs,rd	0xd01c8000	110100 000 0011100 1 sssssss 0 ddddddd	
vrexp2.q rs,rd	0xd01c8080	110100 000 0011100 1 sssssss 1 ddddddd	
vi2uc.q rd,rs	0xd03c8080	110100 000 0111100 1 sssssss 1 ddddddd	int to unsigned char
vi2s.p rd,rs	0xd03f0080	110100 000 0111111 0 sssssss 1 ddddddd	int to short
vi2s.q rd,rs	0xd03f8080	110100 000 0111111 1 sssssss 1 ddddddd	int to short
vsgn.s rd,rs	0xd04a0000	110100 000 1001010 0 sssssss 0 ddddddd	
vsgn.p rd,rs	0xd04a0080	110100 000 1001010 0 sssssss 1 ddddddd	
vsgn.t rd,rs	0xd04a8000	110100 000 1001010 1 sssssss 0 ddddddd	
vsgn.q rd,rs	0xd04a8080	110100 000 1001010 1 sssssss 1 ddddddd	
vcst.s rd, a	0xd0600000	110100 000 11aaaaa 0 0000000 0 ddddddd	
vcst.p rd, a	0xd0600080	110100 000 11aaaaa 0 0000000 1 ddddddd	
vcst.t rd, a	0xd0608000	110100 000 11aaaaa 1 0000000 0 ddddddd	
vcst.q rd, a	0xd0608080	110100 000 11aaaaa 1 0000000 1 ddddddd	
vf2in.s rd,rs,scale	0xd2000000	110100 100 SSSSSSS 0 sssssss 0 ddddddd	float to int round to near
vf2in.p rd,rs,scale	0xd2000080	110100 100 SSSSSSS 0 sssssss 1 ddddddd	
vf2in.t rd,rs,scale	0xd2008000	110100 100 SSSSSSS 1 sssssss 0 ddddddd	
vf2in.q rd,rs,scale	0xd2008080	110100 100 SSSSSSS 1 sssssss 1 ddddddd	
vi2f.s rd,rs,scale	0xd2800000	110100 101 SSSSSSS 0 sssssss 0 ddddddd	int to float
vi2f.p rd,rs,scale	0xd2800080	110100 101 SSSSSSS 0 sssssss 1 ddddddd	
vi2f.t rd,rs,scale	0xd2808000	110100 101 SSSSSSS 1 sssssss 0 ddddddd	
vi2f.q rd,rs,scale	0xd2808080	110100 101 SSSSSSS 1 sssssss 1 ddddddd	
vmmul.p rd,rs,rt	0xf0000080	111100 000 ttttttt 0 sSssssss 1 ddddddd	(*1)
vmmul.t rd,rs,rt	0xf0008000	111100 000 ttttttt 1 sSssssss 0 ddddddd	(*1)
vmmul.q rd,rs,rt	0xf0008080	111100 000 ttttttt 1 sSssssss 1 ddddddd	(*1)

vhtfm2.p rd,rs,rt	0xf0800000	111100 001 ttttttt 0 sssssss 0 ddddddd	
vtfm2.p rd,rs,rt	0xf0800080	111100 001 ttttttt 0 sssssss 1 ddddddd	
vhtfm3.t rd,rs,rt	0xf1000080	111100 010 ttttttt 0 sssssss 1 ddddddd	
vtfm3.t rd,rs,rt	0xf1008000	111100 010 ttttttt 1 sssssss 0 ddddddd	
vhtfm4.q rd,rs,rt	0xf1808000	111100 011 ttttttt 1 sssssss 0 ddddddd	
vtfm4.q rd,rs,rt	0xf1808080	111100 011 ttttttt 1 sssssss 1 ddddddd	
vmidt.p rd	0xf3830080	111100 111 0000011 0 0000000 1 ddddddd	SetMatrixIdentity.Pair
vmidt.t rd	0xf3838000	111100 111 0000011 1 0000000 0 ddddddd	SetMatrixIdentity.Triple
vmidt.q rd	0xf3838080	111100 111 0000011 1 0000000 1 ddddddd	SetMatrixIdentity.Quad
vmzero.p rd	0xf3860080	111100 111 0000110 0 0000000 1 ddddddd	SetMatrixZero.Pair
vmzero.t rd	0xf3868000	111100 111 0000110 1 0000000 0 ddddddd	SetMatrixZero.Triple
vmzero.q rd	0xf3868080	111100 111 0000110 1 0000000 1 ddddddd	SetMatrixZero.Quad

*1) bit 5 of rs is inverted

VFPU load/store instructions seem to support only 16-byte-aligned accesses (similiar to Altivec and SSE).

4.9.1 lv

lv	**LoadVector Quadword Relative to Address in General Purpose Register**
	`fpu_vtr <- vector_at_address (offset + %gpr)`

`lv.q %vfpu_rt, offset(%base)`	

`%fpu_rt`	VFPU Vector Target Register (column0-31/row32-63)
`%base`	GPR, specifies Source Address Base
`offset`	signed Offset added to Source Address Base

Final Address needs to be 64-byte aligned.

4.9.2 sv

sv	**StoreVector Quadword Relative to Address in General Purpose Register**
	`vector_at_address (offset + %gpr) <- fpu_vtr`

`sv.q %vfpu_rt, offset(%base), cache_policy`	

`%fpu_rt`	VFPU Vector Target Register (column0-31/row32-63)
`%base`	specifies Source Address Base
`offset`	signed Offset added to Source Address Base
`cache_policy`	0 = write-through, 1 = write-back

Final Address needs to be 64-byte aligned.

4.9.3 vzero

vzero	**SetVectorZero (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rt] <- 0.0f`

`vzero.s %vfpu_rt`	Set 1 Vector Component to 0.0f
`vzero.p %vfpu_rt`	Set 2 Vector Components to 0.0f
`vzero.t %vfpu_rt`	Set 3 Vector Components to 0.0f
`vzero.q %vfpu_rt`	Set 4 Vector Components to 0.0f

`%vfpu_rt`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)

4.9.4 vone

vone	**SetVectorOne (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rt] <- 0.0f`

`vone.s %vfpu_rt`	Set 1 Vector Component to 1.0f
`vone.p %vfpu_rt`	Set 2 Vector Components to 1.0f
`vone.t %vfpu_rt`	Set 3 Vector Components to 1.0f
`vone.q %vfpu_rt`	Set 4 Vector Components to 1.0f

`%vfpu_rt`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)

4.9.5 vmzero

vmzero	**SetMatrixZero (Pair/Triple/Quad)**
	`vfpu_mtx[%vfpu_rt] <- 0.0f`

`vmzero.p %vfpu_rt`	Set 2x2 Submatrix to 0.0f
`vmzero.t %vfpu_rt`	Set 3x3 Submatrix to 0.0f
`vmzero.q %vfpu_rt`	Set 4x4 Matrix to 0.0f

`%vfpu_rt`	VFPU Matrix Target Register ([s|p|t|q]reg 0..127)

4.9.6 vmidt

vmidt	**SetMatrixIdentity (Pair/Triple/Quad)**
	`vfpu_mtx[%vfpu_rt] <- identity matrix`

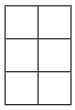
`vmidt.p %vfpu_rt`	Set 2x2 Submatrix to Identity
`vmidt.t %vfpu_rt`	Set 3x3 Submatrix to Identity
`vmidt.q %vfpu_rt`	Set 4x4 Matrix to Identity

`%vfpu_rt`	VFPU Matrix Target Register ([s|p|t|q]reg 0..127)

4.9.7 vmmul

vmmul

`vmmul.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	multiply 2 2x2 Submatrices
`vmmul.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	multiply 2 3x3 Submatrices
`vmmul.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	multiply 2 4x4 Matrices



4.9.8 vrcp

vrcp	**Reciprocal (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- 1.0 / vfpu_regs[%vfpu_rs]`

`vrcp.s %vfpu_rd, %vfpu_rs`	calculate reciprocal (1/z) on single
`vrcp.p %vfpu_rd, %vfpu_rs`	calculate reciprocal (1/z) on pair
`vrcp.t %vfpu_rd, %vfpu_rs`	calculate reciprocal (1/z) on triple
`vrcp.q %vfpu_rd, %vfpu_rs`	calculate reciprocal (1/z) on quad

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.9 vexp2

vexp2	**Exp2 (Single/Pair/Triple/Quad) (calculate 2 raised to the specified real number)**
	`vfpu_regs[%vfpu_rd] <- 2^(vfpu_regs[%vfpu_rs])`

`vexp2.s %vfpu_rd, %vfpu_rs`	calculate 2 ** y
`vexp2.p %vfpu_rd, %vfpu_rs`	calculate 2 ** y
`vexp2.t %vfpu_rd, %vfpu_rs`	calculate 2 ** y
`vexp2.q %vfpu_rd, %vfpu_rs`	calculate 2 ** y

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.10 vlog2

vlog2	**Log2 (Single/Pair/Triple/Quad) (calculate logarithm base 2 of the specified real number)**
	`vfpu_regs[%vfpu_rd] <- log2(vfpu_regs[%vfpu_rs])`

`vlog2.s %vfpu_rd, %vfpu_rs`	
`vlog2.p %vfpu_rd, %vfpu_rs`	
`vlog2.t %vfpu_rd, %vfpu_rs`	
`vlog2.q %vfpu_rd, %vfpu_rs`	

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.11 vsqrt

vsqrt	**SquareRoot (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- sqrt(vfpu_regs[%vfpu_rs])`

`vsqrt.s %vfpu_rd, %vfpu_rs`	calculate square root
`vsqrt.p %vfpu_rd, %vfpu_rs`	calculate square root
`vsqrt.t %vfpu_rd, %vfpu_rs`	calculate square root
`vsqrt.q %vfpu_rd, %vfpu_rs`	calculate square root

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.12 vrsq

vrsq	**ReciprocalSquareRoot (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- 1.0 / sqrt(vfpu_regs[%vfpu_rs])`

`vrsq.s %vfpu_rd, %vfpu_rs`	calculate reciprocal sqrt (1/sqrt(x)) on single
`vrsq.p %vfpu_rd, %vfpu_rs`	calculate reciprocal sqrt (1/sqrt(x)) on pair
`vrsq.t %vfpu_rd, %vfpu_rs`	calculate reciprocal sqrt (1/sqrt(x)) on triple
`vrsq.q %vfpu_rd, %vfpu_rs`	calculate reciprocal sqrt (1/sqrt(x)) on quad

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.13 vsin

vsin	**Sinus (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- sin(vfpu_regs[%vfpu_rs])`

`vsin.s %vfpu_rd, %vfpu_rs`	calculate sin on single
`vsin.p %vfpu_rd, %vfpu_rs`	calculate sin on pair
`vsin.t %vfpu_rd, %vfpu_rs`	calculate sin on triple
`vsin.q %vfpu_rd, %vfpu_rs`	calculate sin on quad

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

note: trig functions on the vfpu expect input values like vsin(degrees/90) or vsin(2/PI * radians)

4.9.14 vcos

vcos	**Cosine (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- cos(vfpu_regs[%vfpu_rs])`

`vcos.s %vfpu_rd, %vfpu_rs`	calculate cos on single
`vcos.p %vfpu_rd, %vfpu_rs`	calculate cos on pair
`vcos.t %vfpu_rd, %vfpu_rs`	calculate cos on triple
`vcos.q %vfpu_rd, %vfpu_rs`	calculate cos on quad

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

Note by John Kelley: trig functions on the vfpu expect input values like vsin(degrees/90) or vsin(2/PI * radians)

4.9.15 vasin

vasin	**ArcSin (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- arcsin(vfpu_regs[%vfpu_rs])`

`vasin.s %vfpu_rd, %vfpu_rs`	calculate arcsin
`vasin.p %vfpu_rd, %vfpu_rs`	calculate arcsin
`vasin.t %vfpu_rd, %vfpu_rs`	calculate arcsin
`vasin.q %vfpu_rd, %vfpu_rs`	calculate arcsin

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.16 vnrcp

vnrcp	**NegativeReciprocal (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- -1/vfpu_regs[%vfpu_rs]`

`vnrcp.s %vfpu_rd, %vfpu_rs`	calculate negative reciprocal
`vnrcp.p %vfpu_rd, %vfpu_rs`	calculate negative reciprocal
`vnrcp.t %vfpu_rd, %vfpu_rs`	calculate negative reciprocal
`vnrcp.q %vfpu_rd, %vfpu_rs`	calculate negative reciprocal

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.17 vnsin

vnsin	**NegativeSin (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- -sin(vfpu_regs[%vfpu_rs])`

`vnsin.s %vfpu_rd, %vfpu_rs`	calculate negative sin
`vnsin.p %vfpu_rd, %vfpu_rs`	calculate negative sin
`vnsin.t %vfpu_rd, %vfpu_rs`	calculate negative sin
`vnsin.q %vfpu_rd, %vfpu_rs`	calculate negative sin

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.18 vrexp2

vrexp2	**ReciprocalExp2 (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- 1/exp2(vfpu_regs[%vfpu_rs])`

`vrexp2.s %vfpu_rd, %vfpu_rs`	calculate 1/(2^y)
`vrexp2.p %vfpu_rd, %vfpu_rs`	calculate 1/(2^y)
`vrexp2.t %vfpu_rd, %vfpu_rs`	calculate 1/(2^y)
`vrexp2.q %vfpu_rd, %vfpu_rs`	calculate 1/(2^y)

`%vfpu_rd`	VFPU Vector Target Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)

4.9.19 vi2uc

vi2uc	**int to unsigned char**

`vi2uc.q %vfpu_rd, %vfpu_rs`	

4.9.20 vi2s

vi2s	**int to short**

`vi2s.p %vfpu_rd, %vfpu_rs`	
`vi2s.q %vfpu_rd, %vfpu_rs`	

4.9.21 vcst

vcst	**StoreConstant (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- constants[%a]`

`vcst.s %vfpu_rd, %a`	store constant into single
`vcst.p %vfpu_rd, %a`	store constant into pair
`vcst.t %vfpu_rd, %a`	store constant into triple
`vcst.q %vfpu_rd, %a`	store constant into quad

`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)
`%a`	VFPU Constant

ID	Constant	Value
0	n/a	0
1	HUGE	340282346638528859811704183484516925440.0
2	SQRT(2)	1.41421
3	1/SQRT(2)	0.70711
4	2/SQRT(PI)	1.12838
5	2/PI	0.63662
6	1/PI	0.31831
7	PI/4	0.78540
8	PI/2	1.57080
9	PI	3.14159
10	E	2,71828
11	LOG2E	1.44270
12	LOG10E	0.43429
13	LN2	0.69315
14	LN10	2.30259
15	2*PI	6.28319
16	PI/6	0.52360
17	LOG10TWO	0.30103
18	LOG2TEN	3.32193
19	SQRT(3)/2	0.86603
20-31	n/a	0

4.9.22 vf2in

vf2in float to int round to near

`vf2in.s %vfpu_rd, %vfpu_rs, scale`	
`vf2in.p %vfpu_rd, %vfpu_rs, scale`	
`vf2in.t %vfpu_rd, %vfpu_rs, scale`	
`vf2in.q %vfpu_rd, %vfpu_rs, scale`	

4.9.23 vi2f

vi2f int to float

`vi2f.s %vfpu_rd, %vfpu_rs, scale`	
`vi2f.p %vfpu_rd, %vfpu_rs, scale`	
`vi2f.t %vfpu_rd, %vfpu_rs, scale`	
`vi2f.q %vfpu_rd, %vfpu_rs, scale`	

4.9.24 vadd

vadd	**VectorAdd (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- vfpu_regs[%vfpu_rs] + vfpu_regs[%vfpu_rt]`

`vadd.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	Add Single
`vadd.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Add Pair
`vadd.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Add Triple
`vadd.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Add Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.25 vsub

vsub	**VectorSub (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- vfpu_regs[%vfpu_rs] - vfpu_regs[%vfpu_rt]`

`vsub.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	Sub Single
`vsub.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Sub Pair
`vsub.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Sub Triple
`vsub.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Sub Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.26 vdiv

vdiv	**VectorDiv (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- vfpu_regs[%vfpu_rs] / vfpu_regs[%vfpu_rt]`

`vdiv.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	div Single
`vdiv.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	div Pair
`vdiv.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	div Triple
`vdiv.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	div Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.27 vmul

vmul	**VectorMul (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- vfpu_regs[%vfpu_rs] * vfpu_regs[%vfpu_rt]`

`vmul.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	mul Single
`vmul.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	mul Pair
`vmul.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	mul Triple
`vmul.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	mul Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.28 vdot

vdot	**VectorDotProduct (Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- dotproduct(vfpu_regs[%vfpu_rs], vfpu_regs[%vfpu_rt])`

`vdot.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Pair
`vdot.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Triple
`vdot.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.29 vhdp

vhdp	**VectorHomogenousDotProduct (Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- homogenousdotproduct(vfpu_regs[%vfpu_rs], vfpu_regs[%vfpu_rt])`

`vhdp.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Pair
`vhdp.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Triple
`vhdp.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Dot Product Quad

`%vfpu_rt`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([s|p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.30 vidt

vidt	**VectorLoadIdentity (Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- identity vector`

`vidt.p %vfpu_rd`	Set 2x1 Vector to Identity
`vidt.t %vfpu_rd`	Set 3x1 Vector to Identity
`vidt.q %vfpu_rd`	Set 4x1 Vector to Identity

`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.31 vabs

vabs	**AbsoluteValue (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- abs(vfpu_regs[%vfpu_rs])`

`vabs.s %vfpu_rd, %vfpu_rs`	Absolute Value Single
`vabs.p %vfpu_rd, %vfpu_rs`	Absolute Value Pair
`vabs.t %vfpu_rd, %vfpu_rs`	Absolute Value Triple
`vabs.q %vfpu_rd, %vfpu_rs`	Absolute Value Quad

`%vfpu_rd`	VFPU Vector Destination Register (m[p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register (m[p|t|q]reg 0..127)

4.9.32 vneg

vneg	**Negate (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- -vfpu_regs[%vfpu_rs]`

`vneg.s %vfpu_rd, %vfpu_rs`	Negate Single
`vneg.p %vfpu_rd, %vfpu_rs`	Negate Pair
`vneg.t %vfpu_rd, %vfpu_rs`	Negate Triple
`vneg.q %vfpu_rd, %vfpu_rs`	Negate Quad

`%vfpu_rd`	VFPU Vector Destination Register (m[p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register (m[p|t|q]reg 0..127)

4.9.33 vsgn

vsgn	**Sign.(Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- sign(vfpu_regs[%vfpu_rs])`

`vsgn.s %vfpu_rd, %vfpu_rs`	Get Sign Single
`vsgn.p %vfpu_rd, %vfpu_rs`	Get Sign Pair
`vsgn.t %vfpu_rd, %vfpu_rs`	Get Sign Triple
`vsgn.q %vfpu_rd, %vfpu_rs`	Get Sign Quad

`%vfpu_rd`	VFPU Vector Destination Register (m[p|t|q]reg 0..127)
`%vfpu_rs`	VFPU Vector Source Register (m[p|t|q]reg 0..127)

Sets rd values to 1 or -1, depending on sign of input values

4.9.34 vmin

vmin	**VectorMin (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- min(vfpu_regs[%vfpu_rs], vfpu_reg[%vfpu_rt])`

`vmin.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Minimum Value Single
`vmin.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Minimum Value Pair
`vmin.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Minimum Value Triple
`vmin.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Minimum Value Quad

`%vfpu_rt`	VFPU Vector Source Register (sreg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.35 vmax

vmax	**VectorMax (Single/Pair/Triple/Quad)**
	`vfpu_regs[%vfpu_rd] <- max(vfpu_regs[%vfpu_rs], vfpu_reg[%vfpu_rt])`

`vmax.s %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Maximum Value Single
`vmax.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Maximum Value Pair
`vmax.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Maximum Value Triple
`vmax.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Get Maximum Value Quad

`%vfpu_rt`	VFPU Vector Source Register (sreg 0..127)
`%vfpu_rs`	VFPU Vector Source Register ([p|t|q]reg 0..127)
`%vfpu_rd`	VFPU Vector Destination Register ([s|p|t|q]reg 0..127)

4.9.36 vtfm

vtfm	VectorTransform (Pair/Triple/Quad)
	`vfpu_regs[%vfpu_rd] <- transform(vfpu_matrix[%vfpu_rs], vfpu_vector[%vfpu_rt])`

`vtfm2.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Transform pair vector by pair matrix
`vtfm3.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Transform triple vector by triple matrix
`vtfm4.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Transform quad vector by quad matrix

`%vfpu_rt`	VFPU Vector Source Register (qreg 0..127)
`%vfpu_rs`	VFPU Matrix Source Register (qmatrix 0..127)
`%vfpu_rd`	VFPU Vector Destination Register (qreg 0..127)

4.9.37 vhtfm

vhtfm	VectorHomogeneousTransform (Pair/Triple/Quad)
	`vfpu_regs[%vfpu_rd] <- homeogenoustransform(vfpu_matrix[%vfpu_rs], vfpu_vector[%vfpu_rt])`

`vhtfm2.p %vfpu_rd, %vfpu_rs, %vfpu_rt`	Homogeneous transform quad vector by pair matrix
`vhtfm3.t %vfpu_rd, %vfpu_rs, %vfpu_rt`	Homogeneous transform quad vector by triple matrix
`vhtfm4.q %vfpu_rd, %vfpu_rs, %vfpu_rt`	Homogeneous transform quad vector by quad matrix

`%vfpu_rt`	VFPU Vector Source Register (qreg 0..127)
`%vfpu_rs`	VFPU Matrix Source Register (qmatrix 0..127)
`%vfpu_rd`	VFPU Vector Destination Register (qreg 0..127)

4.10 Caches

There are two caches: the data cache and the instruction cache. The data cache is used when your program does a load or store to memory, and the instruction cache is used to actually execute all the instructions your program. In general you can ignore the instruction cache unless you're using dynamic code generation, though the discussion of cache locality also applies to the instruction cache.

The PSP's cache structure is pretty simple compared to other CPUs. There's only a 32k L1 cache; there's no L2 cache to worry about.

4.10.1 Cache structure and operation

The 32k of cache is divided up into 64-byte chunks, called cache lines. The cache is managed in terms of cache lines, so even if you only use 1 byte of a line, all 64 bytes are allocated.

When the CPU goes to read a piece of memory, it first looks to see if there's a copy of the memory in cache. If there is, this is called a cache hit, and it can fetch the data in a few cycles. If not, this is a cache miss, and it will take a long time (possibly dozens of cycles) to fetch from main memory. However, on a cache miss, it will find a new cache line for the data, and read from main memory into the cache line; the next time you touch this 64-byte area of memory, it will probably get a cache hit.

Writes are similar. When your program writes to memory, it will just write into the cache, allocating a cache line if necessary. Subsequent writes and reads to that cache line will be cache hits.

A cache line can be in one of three states: invalid, clean or dirty. Invalid means that the cache line has no useful data, and no memory operation will hit it. Clean means that the cache line contains an up-to-date copy of a piece of main memory. Dirty means that the cache line has been written to, and main memory is out of date.

So, what does "allocate a cache line" mean? Because the cache is small relative to main memory, whenever you need a new cache line, you probably need to throw something else out. If the cache line you're replacing is invalid, then you can just start using it. If the line is clean, you can also just drop the old line and start using it. If it is dirty, however, you need to write the old contents back to memory before reusing the line; if you don't then previously written data will effectively disappear.

Note that this means that there's an indefinite, non-deterministic amount of time before a write actually hits main memory. The only thing which normally pushes a dirty cache line into memory is being replaced. If it is never replaced, then it will never be written.

4.10.2 Cache Coherency

All this happens transparently from a software perspective. Apart from the performance effects of all this going on, there's really no way to know its happening, and you can safely ignore it. Or can you?

The tricky part about all this is that the CPU ends up with its own copy of pieces of main memory. If the CPU were the only user of memory in the system, then this would be fine, but the PSP has several other functional units which all use memory, and communicate with the main CPU via memory. In order for this to work, you need to make sure that every user of memory has a consistent and coherent view of memory.

In the Intel world, the CPU performs something called "cache snooping". This means that a dedicated piece of hardware looks at all memory operations to main memory, and checks to see if the CPU's cache has a more up-to-date version of the memory. It also looks at memory writes, and makes sure that the CPU's cache has the most up to date version of the data.

The PSP's MIPS isn't like that. It has no snooping or hardware coherency support, which leads to a problem: if you simply write out a set of commands for the GE into memory, and then tell the GE to run them, there's no guarentee that your commands have actually been written to memory by the time GE tries to run them; they could just be still sitting there in dirty cache lines. You'll see some vertices looking fine, but others are way off in space. You'll see most of your texture, but chunks of it are missing or junk.

4.10.3 The Uncached Address Space

The MIPS offers one solution to this problem: the uncached address space. If you bit-wise OR your pointer with `0x40000000` you end up with a corresponding pointer in the uncached address space, which is generally known as an uncached pointer. These two pointers are aliases: they're two different pointers which refer to the same piece of physical memory.

When you use the uncached pointer, the memory access completely bypasses all the machinery described above: reads will come straight from memory, and writes will go straight to memory.

This leads to a potiential problem. If you use memory through the cached pointer, and then start using the uncached pointer, then you will be in a world of pain. It won't explode, crash or do anything obvious. It may seem to work perfectly well 99% of the time. But then you'll get bitten by strange, non-deterministic, elusive bugs which will move around and disappear every time you try to debug the problem.

When you use uncached memory, it completely ignores the cache, and the cache completely ignores the uncached access. If you write to cached memory, then read via uncached, you won't necessarily see the previously written value because its still in cache. If you write via the uncached pointer, your write may get undone at some later arbitrary point when the dirty cache line eventually gets written.

The solution? You need to:

- ▷ Always use cache-line aligned allocations; this means memalign rather than malloc (and always make sure your allocation is a cache-line size multiple too).
- ▷ Write-invalidate memory before using an uncached pointer alias to the memory.

Note that even if you freshly allocate memory and never touch it with a cached pointer, you still need to write-invalidate the memory range, because it may still be partially cached from when it was previously allocated (this is quite likely, because efficient allocators will try to return still-cached memory for good cache use).

4.10.4 Cache Management Functions

The PSP Kernel provides a set of functions for manipulating the cache:

- ▷ `sceKernelDcacheWritebackAll(void)`
  Writes back all dirty cache-lines in memory. All cache lines which were previously valid will remain valid, but all dirty cache lines will become clean. This is useful for when you write some data to be read by another memory-using device.
- ▷ `sceKernelDcacheWritebackInvalidateAll(void)`
  This writes back all dirty cache-lines, and invalidates the whole cache. This is useful when you want to read some data written by another device. If another device writes memory, but the CPU has clean valid cache lines for that memory, it will read stale data unless you invalidate the cache first. This function is safe because it also writes dirty cache lines, so there's no risk of data loss.
- ▷ `sceKernelDcacheWritebackRange(const void *p, unsigned int size)`
  This writes back a range of memory, making the cache lines in that range clean. p and size should be aligned to the cache-line size. This will probably be more efficient than writing back the whole cache if size is relatively small, but if size is more than around 16k, its probably better to just writeback the whole thing.

▷ `sceKernelDcacheWritebackInvalidateRange(const void *p, unsigned int size)`
This writes back a range of memory and invalidates the cache for that range. p and size should be aligned to the cache-line size. This is like sceKernelDcacheWritebackInvalidateAll, but it only affects the specified memory range. This is likely to be more efficient, because it doesn't completely destroy the cache's working-set. You should always use this on a range of memory before accessing it via an uncached pointer.

▷ `sceKernelDcacheInvalidateRange(const void *p, unsigned int size)`
This function should be used with extreme caution. It will invalidate a range of cache lines; if they were previously dirty, then the dirty data will be discarded. This should be used when you want to force data to be fetched from main memory, and you're certain that there are no dirty cache lines in that range of memory. It is very important that p and size are cache-aligned. Because this function affects whole cache lines, if you pass an unaligned pointer or size, then you may end up affecting unintended data.

5 Media Engine

5.1 Overview

- ▷ Video RAM appears to be inaccessable, at least at the usual address. (there is something mapped at 0x04000000 ?, appears to be mmio and not ram)
- ▷ I/O seems to be accessable (unconfirmed)
- ▷ looks like the exception handler location is set by loading cop0 register 25 (usually perfcnt) with the address of your handler
- ▷ INT 31 catches the ME irq on the main core

5.2 Memory Map

5.2.1 physical Memory

start	end	size	description
0x00000000	0x001fffff	2mb	ME internal RAM
0x08000000	0x09ffffff	32mb	Main Memory
0x1fc00000	0x1fcfffff	1mb	Hardware Exception Vectors (RAM)

5.2.2 Ram Usage

start	end	size	description
0x80000000	0x801fffff	2mb	ME internal RAM
0x88000000	0x89ffffff	32mb	Main Memory
0xbfc00000	0xbfcfffff	1mb	Hardware Exception Vectors (RAM)

5.3 COP0

5.3.1 Status registers (mfc/mtc)

0			
1			
2			
3			
4			
5			
6			
7			
8	r	badvaddr	virtual address of last error/exception
9	r/w	count	system counter
10			
11	r/w	compare	counter comparison value
12	r/w	status	system status
13	r/w	cause	exception cause
14	r/w	EPC	exception program counter
15	r	prid	processor revision id
16	r	config	configuration
17			
18			
19			
20			
21		SC-code	SC-code << 2
22			CPU ID (0=Main, 1=ME)
23			
24	?	?	?
25	r/w	Ebase	virtual address of exception vector
26			
27			
28	r/w	TagLo	cache instruction register
29	r/w	TagHi	cache instruction register
30	r/w	ErrorEPC	error exception program counter
31			

5.3.2 Control Registers (cfc/ctc)

5.4 COP1 (FPU)

5.4.1 Status Registers (mfc/mtc)

5.4.2 Control Registers (cfc/ctc)

6 VME

The VME (Virtual Mobile Engine) is a reconfigurable processor to decode audio/video. in 2002, Sony developed the Virtual Mobile Engine? as a method for achieving significant power reductions and miniaturization in LSIs for audio/visual products. This circuit technology, which can reduce power consumption by approximately 1/4 over conventional general-purpose digital signal processors (DSP), was adopted for use in the CXR704060 LSI used in the Network Walkman "NW-MS70D".

There are minimal system APIs for the VME (disable/enable reset). It appears the VME software is tied into the ME (Media Engine).

6.1 Overview

- ▷ Reconfigurable DSPs
- ▷ 128bit Bus
- ▷ 166MHz @1.2V
- ▷ 5 Giga Operations /sec
- ▷ CODEC Capability
- ▷ 3D Sound, Multi-Channel
- ▷ Synthesizer, Effecter, etc

7 Memory Map

7.1 Segments

virtual address	msb	physical address	size	type	comment	mode(s)
0x0.......	000	0x0.......	1024 MB	KU0	cached	user/supervisor/kernel
0x4.......	010	0x0.......	1024 MB	KU1	uncached	user/supervisor/kernel
0x8.......	100	0x0.......	512 MB	K0	cached	kernel
0xA.......	101	0x0.......	512 MB	K1	uncached	kernel
0xC.......	110	0x0.......	512 MB	K2/KS	cached	supervisor/kernel
0xE.......	111	0x0.......	512 MB	K3	cached	kernel

note: K2 and K3 segments seem to be unused

7.2 physical Memory

start	end	size	description
0x00010000	0x00013fff	16kb	scratchpad
0x04000000	0x041fffff	2mb	Video Memory / Frame Buffer
0x08000000	0x09ffffff	32mb	Main Memory
0x1c000000	0x1fbfffff		Hardware i/o
0x1fc00000	0x1fcfffff	1mb	Hardware Exception Vectors (RAM)
0x1fd00000	0x1fffffff		Hardware i/o

7.3 Ram usage

start	end	size	segment	description
0x04000000	0x041fffff	2mb	KU0	Video Memory / Frame Buffer
0x88000000	0x887fffff	8mb	K0	Kernel Memory
0x08800000	0x09ffffff	24mb	KU0	Userspace Memory
0xbfc00000	0xbfcfffff	1mb	K1	Hardware Exception Vectors (RAM)

7.3.1 Kernel

7.3.1.1 K0

start	end	size	description
0x88000000	0x8837ffff	3.5mb	kernel modules are loaded here
0x88380000			ME Resetcode
0x883d6000	0x883fffff	168k	seems to be unused
0x88400000	0x887fffff	4mb	Module/Threadmanager Memory (v1.5 FW only ?)
0x88C00000			Loadexec Stage 2

7.3.1.2 K1

start	end	size	description
0xbfc00000			Reset Vector? (cop0.9:EXC31_ErrVec)
0xbfc00040	0xbfc000ff		ME Handler
0xbfc00160			(mebooter, mebooter_umdvideo)
0xbfc00400			(sysreg)
0xbfc00600			ME RPC-Call struct (s1, s2, s3, s4, s5, s6, s7, fp, arg0) (me_wrapper)
0xbfc00700			Exception struct (flag, COP0.EPC, COP0.EPC.err, COP0.Status, COP0.Ca COP0.BadVAddr) (mebooter, mebooter_umdvideo, me_wrapper)
0xbfc00ffc			(sysreg)
0xbfc01000	0xbfc01fff	16*0x0100	Exception Vectors? (cop0.10:?)
0xbfc02000	0xbfcfffff	254*0x1000	Exception Vectors? (cop0.9:EXC31_ErrVec)

7.3.2 Userspace

7.3.2.1 KU0

start	end	size	description
0x08800000			
0x08900000			user main program start address

7.3.2.2 KU1 all Memory that can be acessed from KU0 segment, which is cached, can also be acessed from the KU1 segment, which is uncached.

7.4 Hardware

start	end	description
0xbc0000xx		memory interface ?? (mpeg_vsh, sysmem, sysreg, threadman, usb)
0xbc1000xx		System Control (IPL, dmacman, emc_ddr, memlmd, mscm, syscon, sysmem, sysreg, exceptionman, ata, mebooter, mebooter_umdvideo , me_wrapper, reboot, uart4)
0xbc20000x		irq?? (sysreg)
0xbc3000xx		irq?? (interruptman)
0xbc400000		Hardware Profiler (threadman, utils)
0xbc500000		irq, Timer? (systimer)
0xbc6000xx		(threadman)
0xbc8000xx		DMA control (dmacplus)
0xbc9000xx		DMA control (dmacman)
0xbca00000		DMA control (dmacman)
0xbcc00000		ME Control (mebooter, mebooter_umdvideo, me_wrapper)
0xbd0000xx		systemcontrol, watchdog, sram controller ?? (emc_ddr, mpeg_vsh, usb, syscon)
0xbd100000		NAND Flash (ems_sm, mpeg_vsh, reboot)
0xbd1010xx		NAND Flash (ems_sm)
0xbd101200		NAND Flash (ems_sm)
0xbd101300		NAND Flash (ems_sm)
0xbd200000		memstick? (mscm, mpeg_vsh)
0xbd300000		WLAN (wlan)
0xbd40000x		Graphics engine (ge)
0xbd4001xx		(ge)
0xbd400200		(ge)
0xbd4003xx		(ge)
0xbd400400		(ge)
0xbd4008xx		(ge)
0xbd400900		(ge)
0xbd400acx		(ge)
0xbd400b10		(ge)
0xbd5000x0		(ge)
0xbd6000xx		atapi? (ata, umdman)
0xbd70000x		ATA (ata, umdman)
0xbd800000		USB regs (usb, mpeg_vsh)
0xbd800214		USB regs (usb, mpeg_vsh)
0xbd8004xx		USB regs (usb, mpeg_vsh)
0xbde000xx		Crypt Engine (IPL, memlmd, reboot)
0xbdf000xx		umd stuff (umdman)
0xbe0000xx		audio stuff (audio, mpeg_vsh)
0xbe100000		(mgr)
0xbe1400xx		LCDC (display?) (lcdc)
0xbe2000xx		IIC stuff, (which component uses i2c at all -> clock generator and the WM8750 audio codec) (i2c)
0xbe2400xx		general purpose IO (gpio, syscon)
0xbe300000		power management (pwm)
0xbe3400xx		IRDA (sircs)
0xbe4c00xx		UART4 Uart4/kernel debug(?) UART (IPL, uart4, reboot)

0xbe5000xx		UART3(?) headphone remote SIO (hpremote)
0xbe5400xx		UART2(?) IRDA ? (sircs)
0xbe5800xx		UART1(?) Serial EPROM(?) system control ? (syscon)
0xbe7400xx		display controler (display)
0xbf000000		(mpeg_vsh, pspnet_inet)
0xbfa00000		(power)

start	end	description
0xbfe00000	0xbfffffff	? all accessable, but all 0 and can not be written to?
0xbff00000		Nand DMA User Data Buf (rw), 512 bytes buffer to hold DMA data for a user page (emc_sm, reboot)
0xbff00800		Nand User ECC Reg (rw), 32bit Hardware calculated ECC for a user page (emc_sm)
0xbff00900		Nand DMA Spare Data Buf start (rw), 16 bytes buffer to hold DMA data for a spare page (emc_sm)
0xbfff0000		(power, pspnet, sysmem, threadman)
0xbfffffff		(threadman, power, sysmem)

8 Hardware Registers

8.1 ? (threadman)

Registerblock Base	Size of Registerblock	common access size
0xbc000000		32 bit

0xbc000000	4	r/w	Memory Protection 0x08000000 -> 0x081FFFFFF

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
31		0x081c0000 -> 0x081FFFFFF Kernel Write Enable
30		0x081c0000 -> 0x081FFFFFF Kernel Read Enable
29		0x081c0000 -> 0x081FFFFFF User Write Enable
28		0x081c0000 -> 0x081FFFFFF User Read Enable
27		0x08180000 -> 0x081BFFFFF Kernel Write Enable
26		0x08180000 -> 0x081BFFFFF Kernel Read Enable
25		0x08180000 -> 0x081BFFFFF User Write Enable
24		0x08180000 -> 0x081BFFFFF User Read Enable
23		0x08140000 -> 0x0817FFFFF Kernel Write Enable
22		0x08140000 -> 0x0817FFFFF Kernel Read Enable
21		0x08140000 -> 0x0817FFFFF User Write Enable
20		0x08140000 -> 0x0817FFFFF User Read Enable
19		0x08100000 -> 0x0813FFFFF Kernel Write Enable
18		0x08100000 -> 0x0813FFFFF Kernel Read Enable
17		0x08100000 -> 0x0813FFFFF User Write Enable
16		0x08100000 -> 0x0813FFFFF User Read Enable
15		0x080c0000 -> 0x080FFFFFF Kernel Write Enable
14		0x080c0000 -> 0x080FFFFFF Kernel Read Enable
13		0x080c0000 -> 0x080FFFFFF User Write Enable
12		0x080c0000 -> 0x080FFFFFF User Read Enable
11		0x08080000 -> 0x080BFFFFF Kernel Write Enable
10		0x08080000 -> 0x080BFFFFF Kernel Read Enable
9		0x08080000 -> 0x080BFFFFF User Write Enable
8		0x08080000 -> 0x080BFFFFF User Read Enable
7		0x08040000 -> 0x0807FFFFF Kernel Write Enable
6		0x08040000 -> 0x0807FFFFF Kernel Read Enable
5		0x08040000 -> 0x0807FFFFF User Write Enable
4		0x08040000 -> 0x0807FFFFF User Read Enable
3		0x08000000 -> 0x08003FFFF Kernel Write Enable
2		0x08000000 -> 0x08003FFFF Kernel Read Enable
1		0x08000000 -> 0x08003FFFF User Write Enable
0		0x08000000 -> 0x08003FFFF User Read Enable

0xbc000004	4	r/w	Memory Protection 0x08200000 -> 0x083FFFFFF

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
31		0x083c0000 -> 0x083FFFFFF Kernel Write Enable
30		0x083c0000 -> 0x083FFFFFF Kernel Read Enable
29		0x083c0000 -> 0x083FFFFFF User Write Enable
28		0x083c0000 -> 0x083FFFFFF User Read Enable
27		0x08380000 -> 0x083BFFFFF Kernel Write Enable
26		0x08380000 -> 0x083BFFFFF Kernel Read Enable
25		0x08380000 -> 0x083BFFFFF User Write Enable
24		0x08380000 -> 0x083BFFFFF User Read Enable
23		0x08340000 -> 0x0837FFFFF Kernel Write Enable
22		0x08340000 -> 0x0837FFFFF Kernel Read Enable
21		0x08340000 -> 0x0837FFFFF User Write Enable
20		0x08340000 -> 0x0837FFFFF User Read Enable
19		0x08300000 -> 0x0833FFFFF Kernel Write Enable
18		0x08300000 -> 0x0833FFFFF Kernel Read Enable
17		0x08300000 -> 0x0833FFFFF User Write Enable
16		0x08300000 -> 0x0833FFFFF User Read Enable
15		0x082c0000 -> 0x082FFFFFF Kernel Write Enable
14		0x082c0000 -> 0x082FFFFFF Kernel Read Enable
13		0x082c0000 -> 0x082FFFFFF User Write Enable
12		0x082c0000 -> 0x082FFFFFF User Read Enable
11		0x08280000 -> 0x082BFFFFF Kernel Write Enable
10		0x08280000 -> 0x082BFFFFF Kernel Read Enable
9		0x08280000 -> 0x082BFFFFF User Write Enable
8		0x08280000 -> 0x082BFFFFF User Read Enable
7		0x08240000 -> 0x0827FFFFF Kernel Write Enable
6		0x08240000 -> 0x0827FFFFF Kernel Read Enable
5		0x08240000 -> 0x0827FFFFF User Write Enable
4		0x08240000 -> 0x0827FFFFF User Read Enable
3		0x08200000 -> 0x08203FFFF Kernel Write Enable
2		0x08200000 -> 0x08203FFFF Kernel Read Enable
1		0x08200000 -> 0x08203FFFF User Write Enable
0		0x08200000 -> 0x08203FFFF User Read Enable

0xbc000008	4	r/w	Memory Protection 0x08400000 -> 0x085FFFFFF

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
31		0x085c0000 -> 0x085FFFFFF Kernel Write Enable
30		0x085c0000 -> 0x085FFFFFF Kernel Read Enable
29		0x085c0000 -> 0x085FFFFFF User Write Enable
28		0x085c0000 -> 0x085FFFFFF User Read Enable
27		0x08580000 -> 0x085BFFFFF Kernel Write Enable
26		0x08580000 -> 0x085BFFFFF Kernel Read Enable
25		0x08580000 -> 0x085BFFFFF User Write Enable
24		0x08580000 -> 0x085BFFFFF User Read Enable
23		0x08540000 -> 0x0857FFFFF Kernel Write Enable
22		0x08540000 -> 0x0857FFFFF Kernel Read Enable
21		0x08540000 -> 0x0857FFFFF User Write Enable
20		0x08540000 -> 0x0857FFFFF User Read Enable
19		0x08500000 -> 0x0853FFFFF Kernel Write Enable
18		0x08500000 -> 0x0853FFFFF Kernel Read Enable
17		0x08500000 -> 0x0853FFFFF User Write Enable
16		0x08500000 -> 0x0853FFFFF User Read Enable
15		0x084c0000 -> 0x084FFFFFF Kernel Write Enable
14		0x084c0000 -> 0x084FFFFFF Kernel Read Enable
13		0x084c0000 -> 0x084FFFFFF User Write Enable
12		0x084c0000 -> 0x084FFFFFF User Read Enable
11		0x08480000 -> 0x084BFFFFF Kernel Write Enable
10		0x08480000 -> 0x084BFFFFF Kernel Read Enable
9		0x08480000 -> 0x084BFFFFF User Write Enable
8		0x08480000 -> 0x084BFFFFF User Read Enable
7		0x08440000 -> 0x0847FFFFF Kernel Write Enable
6		0x08440000 -> 0x0847FFFFF Kernel Read Enable
5		0x08440000 -> 0x0847FFFFF User Write Enable
4		0x08440000 -> 0x0847FFFFF User Read Enable
3		0x08400000 -> 0x08403FFFF Kernel Write Enable
2		0x08400000 -> 0x08403FFFF Kernel Read Enable
1		0x08400000 -> 0x08403FFFF User Write Enable
0		0x08400000 -> 0x08403FFFF User Read Enable

0xbc00000c	4	r/w	Memory Protection 0x08600000 -> 0x087FFFFF

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
31		0x087c0000 -> 0x087FFFFF Kernel Write Enable
30		0x087c0000 -> 0x087FFFFF Kernel Read Enable
29		0x087c0000 -> 0x087FFFFF User Write Enable
28		0x087c0000 -> 0x087FFFFF User Read Enable
27		0x08780000 -> 0x087BFFFF Kernel Write Enable
26		0x08780000 -> 0x087BFFFF Kernel Read Enable
25		0x08780000 -> 0x087BFFFF User Write Enable
24		0x08780000 -> 0x087BFFFF User Read Enable
23		0x08740000 -> 0x0877FFFF Kernel Write Enable
22		0x08740000 -> 0x0877FFFF Kernel Read Enable
21		0x08740000 -> 0x0877FFFF User Write Enable
20		0x08740000 -> 0x0877FFFF User Read Enable
19		0x08700000 -> 0x0873FFFF Kernel Write Enable
18		0x08700000 -> 0x0873FFFF Kernel Read Enable
17		0x08700000 -> 0x0873FFFF User Write Enable
16		0x08700000 -> 0x0873FFFF User Read Enable
15		0x086c0000 -> 0x086FFFFF Kernel Write Enable
14		0x086c0000 -> 0x086FFFFF Kernel Read Enable
13		0x086c0000 -> 0x086FFFFF User Write Enable
12		0x086c0000 -> 0x086FFFFF User Read Enable
11		0x08680000 -> 0x086BFFFF Kernel Write Enable
10		0x08680000 -> 0x086BFFFF Kernel Read Enable
9		0x08680000 -> 0x086BFFFF User Write Enable
8		0x08680000 -> 0x086BFFFF User Read Enable
7		0x08640000 -> 0x0867FFFF Kernel Write Enable
6		0x08640000 -> 0x0867FFFF Kernel Read Enable
5		0x08640000 -> 0x0867FFFF User Write Enable
4		0x08640000 -> 0x0867FFFF User Read Enable
3		0x08600000 -> 0x08603FFF Kernel Write Enable
2		0x08600000 -> 0x08603FFF Kernel Read Enable
1		0x08600000 -> 0x08603FFF User Write Enable
0		0x08600000 -> 0x08603FFF User Read Enable

0xbc000030	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		1: thread profile mode 3: make profiler accessable in usermode at 0x5c400000 (used in threadman)

0xbc000044	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
9		(used in threadman)

8.2 System Config

Registerblock Base	Size of Registerblock	common access size
0xbc100000		32 bit

0xbc100000	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
25-16		Number of NMI that occured
0-9		

NMI related, looks like enable mask (upper 16bits: kernel lower:user)

0xbc100004	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

NMI related, looks like IRQ latches (written to ACK)

bc100010,..28,..30 might have flags for individual NMI sources

0xbc100040	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-1		RAM size: 0-16M; 1-32M; 2-64M; 3-128M

0xbc100044	4	r/w	SC/ME RPC Interrupt

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		write 1 to post interrupt

The RPC works by posting an interrupt to the other processor using the following code:

asm("sync\n");

_sw(1, 0xBC100044);

asm("sync\n");

If you do that on the SC you interrupt (interrupt 31 ?) the ME, on the ME is does the reverse. On the SC side that is wrapped up in sceSysregInterruptToOther.

0xbc100048	4	r/w	SC/ME Semaphore

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

For semaphores there seems to be a shared lock register at 0xBC100048 which both the ME and the SC can write to and it used as a spin lock.

0xbc10004c	4	r/w	RESET ENABLE

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10		KIRK
8-9		MSIF
7		ATA
6		USB
5		AVC
4		VME
3		AW
2		ME
1		SC
0		Top

0xbc100050	4	r/w	BUS CLOCK ENABLE

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
15-16		Audio
14		UART4 ?
13		EMCSM (nand)
12		?
10-11		MSIF
9		USB
8		ATA
7		KIRK
5-6		DMAC
4		DMACPlus
3		AW ?
2		AW ?
1		AW ?
0		ME

0xbc100078	4	r/w	IO ENABLE

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
19-24		SPI
13-18		UART
12		PWM
11		KEY
10		AUDIO ?
9		SIRCS
8		IIC
6-7		AUDIO
5		LCDC
3-4		MSIF
2		ATA
1		USB
0		EMCSM (nand)

0xbc10007c	4	r/w	GPIO IO ENABLE

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbc100080	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

Access to system memory causes an exception unless 0x00000007 is written into this register.

8.3 ? (interruptman)

Registerblock Base	Size of Registerblock	common access size
0xbc300000		32 bit

0xbc300000	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

upper 2 bits 'enable' ?, upper bits=mask ? (used in irq handler)

0xbc300008	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

upper bits=mask,low 4 bits='ack,enable' ? (used in irq handler)

0xbc300010	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

mask ? (used in irq handler)

0xbc300018	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

mask ? (used in irq handler)

8.4 Profiler

Registerblock Base	Size of Registerblock	common access size
0xbc400000		32 bit

0xbc400000	4	r/w	ENABLE

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description	
0		0	profiling disabled
		1	profiling enabled

first clear all counter registers by writing 0 to them, then enable profiling. counter registers are as follows:

Address	Unit	Description
0xbc400004	cycles	systemck
0xbc400008	cycles	cpu ck
0xbc40000c	cycles	stall (total)
0xbc400010	cycles	stall (internal)
0xbc400014	cycles	stall (memory)
0xbc400018	cycles	stall (COPz)
0xbc40001c	cycles	stall (VFPU)
0xbc400020	cycles	sleep
0xbc400024	cycles	bus access
0xbc400028	times	uncached load
0xbc40002c	times	uncached store
0xbc400030	times	cached load
0xbc400034	times	cached store
0xbc400038	times	I cache miss
0xbc40003c	times	D cache miss
0xbc400040	times	D cache wb
0xbc400044	instructions	COP0 inst
0xbc400048	instructions	FPU inst
0xbc40004c	instructions	VFPU inst
0xbc400050	cycles	local bus

8.5 ME Control

Registerblock Base	Size of Registerblock	common access size
0xbcc00000		32 bit

0xbcc00010	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		reset. set to 1, then wait until 0

0xbcc00030	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

set to 0x00000008 at ME Reset

0xbcc00040	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

set to 0x00000002 at ME Reset

0xbcc00070	4	r/w	

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

set to 0x00000001 at ME Reset

8.6 NAND Flash

Registerblock Base	Size of Registerblock	common access size
0xbd101000	0x100 ?	32 bit

0xbd101000	4	r	NAND Control Register

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
18-31		?
17		Calculate ECC for user page during writing
16		Calculate ECC for user page during reading
0-15		?

0xbd101004	4	r	Status ?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
7		0: NAND is not write-protected, 1: NAND is write-protected
0		0=busy, 1=ready

0xbd101008	4	w	Command

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7		Command (see below)

0xbd10100c	4	w	Address

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-26		Physical page to access

0xbd101014	4	w	Nand Reset Reg

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		Reset NAND controller to default state?

0xbd101020	4	w	Nand DMA Address Reg

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-26		Physical page to access

0xbd101024	4	w	NAND DMA Control

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
19-31		
9		Set to enable DMA transfer? (ECC?) Or set to clear previous status?
8		Set to enable DMA transfer? (USER?) Or set to clear previous status?
2-7		?
1		0 -> Transfer from Nand to Nand Data Buffer 1 -> Transfer from Nand Data Buffer to Nand
0		Set to enable DMA transfer

0xbd101028	4	r	NAND DMA Status

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-31		!=0 means write failed ?

0xbd101038	4	rw	NAND DMA Intr

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

Probably the same bits as bd101024

0xbd101200	4	w	resume (?)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-31		write 0x0b040205 to resume?

0xbd101300	4	rw	NAND serial Data

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
24-31		byte 3
16-23		byte 2
8-15		byte 1
0-7		byte 0

0xbff00000	512	rw	Nand DMA User Data Buf

512 bytes buffer to hold DMA data for a user page.

0xbff00800	4	rw	Nand User ECC Reg

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-31		Hardware calculated ECC for a user page

0xbff00900	16	rw	Nand DMA Spare Data Buf

16 bytes buffer to hold DMA data for a spare page.

8.6.1 Command Set

Function	1st Cycle	2nd Cycle	Acceptable when Busy
Read 1	0x00/0x01		no
Read 2	0x50		no
Read ID	0x90		no
Reset	0xff		yes
Page Program	0x80	0x10	no
Copy-Back Program	0x00	0x8a	no
Block Erase	0x60	0xd0	no
Read Status	0x70		yes

8.6.2 Read ID

- write 0x90 to the Command Register
- write 0x00 to address input
- two sequential read cycles return
  - manufacture code
  - device code

8.6.3 read from NAND

- ▷ Write appropriate flags to Nand Control reg (bd1010000)
- ▷ Write page number to Nand DMA Addr reg (bd101020)
- ▷ Clear appropriate flags in the Nand DMA Intr reg (bd101038)
- ▷ Start DMA transfer by writing the appropriate flags to the Nand DMA control reg (bd101024)
- ▷ Wait for interrupt
- ▷ Copy user data from Nand User Data buf (bff00000 - bff00200) (careful with cache!)
- ▷ Check User ECC status (?) (bff00800)
- ▷ Copy ECC value from from Nand User ECC buf (bff00800)
- ▷ Copy spare data from Nand Spare Data buf (bff00900)
- ▷ Check Spare ECC manually

8.6.4 write to NAND

- ▷ Copy user data to Nand User Data buf (bff00000 - bff00200) (careful with cache!)
- ▷ Write ECC value to Nand User ECC buf (bff00800) (Alternatively, the hw might be able to generate it)
- ▷ Write appropriate flags to Nand Control reg (bd1010000)
- ▷ Write spare data to Nand Spare Data buf (bff00900 - bff00910)
- ▷ Write page number to Nand DMA Addr reg (bd101020)
- ▷ Clear appropriate flags in the Nand DMA Intr reg (bd101038)
- ▷ Start DMA transfer by writing the appropriate flags to Nand DMA control reg (bd101024)
- ▷ Wait for interrupt and process accordingly

(Maybe it's possible to write data using the serial data register too)

8.7 KIRK - Decryption Engine

Registerblock Base	Size of Registerblock	common access size
0xbde00000		32 bit

0xbde00000	4	r/w	Signature

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		'K' 'I' 'R' 'K'

0xbde00004	4	r/w	Version

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		version: '0' '0' '1' '0'

0xbde00008	4	r/w	Error

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		set to 1 on incomplete processing

0xbde0000c	4	r/w	StartProcessing

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		set this to 1 or 2 to start phase 1/2 of the processing

0xbde00010	4	r/w	command

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description							
		command	dest		source		extra	description	
		0x01	buf	size	buf+0x40	size+0x40		decrypt	memlmd, mes
		0x02							
		0x03							
		0x04	buf	size+0x14	buf	size+0x14	0x04,code	block cypher	chnnlsv, mem
		0x05	buf	size+0x14	buf	size+0x14	0x04,0x0100	block cypher	chnnlsv
		0x06							
		0x07	buf	size+0x14	buf	size+0x14	0x05,code	block cypher, scramble	memlmd, mes
		0x08	buf	size+0x14	buf	size+0x14	0x05,0x0100	block cypher	chnnlsv
0-4		0x09							
		0x0a							
		0x0b	buf	size	buf	size		SHA1 (size>=0x14)	memlmd, mes
		0x0c	buf	0x3c	0	0		? some read	memab
		0x0d	buf	0x3c	buf	0x3c		?	
		0x0e	buf	0x14	0	0		dbgsvrgetdata	mesg_led,chn
		0x0f							
		0x10	buf	0x34	buf	0x34			memab
		0x11	0	0	buf	0x64		? some check	memab
		0x12	0	0	buf	0xb8		? some check	openpsid, mer

0xbde00014	4	r/w	result

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		result of semaphore_XXXXXXXX functions (exported)

0xbde00018	4	r/w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbde0001c	4	r/w	pattern

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		pattern to check status of processing

0xbde00020	4	r/w	asyncPattern

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		pattern set before starting an async processing

0xbde00024	4	r/w	asyncPattern_end

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		value of asyncPattern after processing

0xbde00028	4	r/w	pattern_end

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		value of pattern after processing

0xbde0002c	4	r/w	source_addr

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		physical address of source buffer

0xbde00030	4	r/w	dest_addr

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		physical address of destination buffer

0xbde0004c	4	r/w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbde00050	4	r/w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

8.7.1 Keys

```
0x6A, 0x19, 0x71, 0xF3, 0x18, 0xDE, 0xD3, 0xA2, 0x6D,
0x3B, 0xDE, 0xC7, 0xBE, 0x98, 0xE2, 0x4C, 0xE3, 0xDC,
0xDF, 0x42, 0x7B, 0x5B, 0x12, 0x28, 0x7D, 0xC0, 0x7A,
0x59, 0x86, 0xF0, 0xF5, 0xB5, 0x58, 0xD8, 0x64, 0x18,
0x84, 0x24, 0x7F, 0xE9, 0x57, 0xAB, 0x4F, 0xC6, 0x92,
0x6D, 0x70, 0x29, 0xD3, 0x61, 0x87, 0x87, 0xD0, 0xAE,
0x2C, 0xE7, 0x37, 0x77, 0xC7, 0x3C, 0x96, 0x7E, 0x21,
0x1F, 0x65, 0x95, 0xC0, 0x61, 0x57, 0xAC, 0x64, 0xD8,
0x5A, 0x6D, 0x14, 0xD2, 0x9C, 0x54, 0xC6, 0x68, 0x5D,
0xF5, 0xC3, 0xF0, 0x50, 0xDA, 0xEA, 0x19, 0x43, 0xA7,
0xAD, 0xC3, 0x2A, 0x14, 0xCA, 0xC8, 0x4C, 0x83, 0x86,
0x18, 0xAE, 0x86, 0x49, 0xFB, 0x4F, 0x45, 0x75, 0xD2,
0xC3, 0xD6, 0xE1, 0x13, 0x69, 0x37, 0xC6, 0x90, 0xCF,
0xF9, 0x79, 0xA1, 0x77, 0x3A, 0x3E, 0xBB, 0xBB, 0xD5,
0x3B, 0x84, 0x1B, 0x9A, 0xB8, 0x79, 0xF0, 0xD3, 0x5F,
0x6F, 0x4C, 0xC0, 0x28, 0x87, 0xBC, 0xAE, 0xDA, 0x00,
```

```
0x50, 0xCC, 0x03, 0xAC, 0x3F, 0x53, 0x1A, 0xFA, 0x0A,
0xA4, 0x34, 0x23, 0x86, 0x61, 0x7F, 0x97, 0x84, 0x1C,
0x1A, 0x1D, 0x08, 0xD4, 0x50, 0xB6, 0xD9, 0x73, 0x27,
0x80, 0xD1, 0xDE, 0xEE, 0xCA, 0x49, 0x8B, 0x84, 0x37,
0xDB, 0xF0, 0x70, 0xA2, 0xA6, 0x2B, 0x09, 0x4D, 0x3B,
0x29, 0xDE, 0x0B, 0xE1, 0x6F, 0x04, 0x7A, 0xC4, 0x18,
0x7A, 0x69, 0x73, 0xBF, 0x02, 0xD8, 0xA1, 0xD0, 0x58,
0x7E, 0x69, 0xCE, 0xAC, 0x5E, 0x1B, 0x0A, 0xF8, 0x19,
0xE6, 0x9A, 0xC0, 0xDE, 0xA0, 0xB2, 0xCE, 0x04, 0x43,
0xC0, 0x9D, 0x50, 0x5D, 0x0A, 0xD7, 0xFD, 0xC6, 0x53,
0xAA, 0x13, 0xDD, 0x2C, 0x3B, 0x2B, 0xBF, 0xAB, 0x7C,
0xF5, 0xA0, 0x4A, 0x79, 0xE3, 0xF1, 0x7B, 0x2E, 0xB2,
0xA3, 0xAC, 0x8E, 0x0A, 0x38, 0x9B, 0x9E, 0xAA, 0xEC,
0x2B, 0xA3, 0x75, 0x13, 0x75, 0x77, 0x98, 0x6A, 0x66,
0x92, 0x65, 0xBC, 0x97, 0x80, 0x0E, 0x32, 0x88, 0x9F,
0x64, 0xBA, 0x99, 0x8A, 0x72, 0x96, 0x9F, 0xE1, 0xE0,
```

8.8 GPIO

Registerblock Base	Size of Registerblock	common access size
0xbe240000		32 bit

0xbe240004	4	w	Port Read

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbe240008	4	w	Port Write

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbe24000C	4	w	Port Clear

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

8.9 UART4

Registerblock Base	Size of Registerblock	common access size
0xbe4c0000		32 bit

0xbe4c0000	4	r/w	FIFO

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7	r	read byte from recieve buffer
	w	write byte to transmit buffer

0xbe4c0018	4	r/w	STATUS

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
5	TXFULL	1 if transmit buffer full
4	RXEMPTY	1 if recieve buffer empty

0xbe4c0024	4	w	DIV1 - upper bits of Baudrate Divisor

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		(96000000 / baudrate) >> 6

0xbe4c0028	4	w	DIV2 - lower 6 bits of Baudrate Divisor

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-5		(96000000 / baudrate) & 0x3f

0xbe4c002c	4	w	CONTROL

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
6		? (set to 1 if you want to set baudrate)
5		? (set to 1 if you want to set baudrate)

0xbe4c0030	4	w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbe4c0034	4	w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

0xbe4c0044	4	w	?

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

8.10 UART3 Headphone/Remote SIO

Registerblock Base	Size of Registerblock	common access size
0xbe500000		32 bit

0xbe500000	4	r/w	FIFO

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7	r	read byte from recieve buffer
	w	write byte to transmit buffer

0xbe500018	4	r/w	STATUS

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
5	TXFULL	1 if transmit buffer full
4	RXEMPTY	1 if recieve buffer empty

0xbe500024	4	w	DIV1 - upper bits of Baudrate Divisor

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
		(96000000 / baudrate) >> 6

0xbe500028	4	w	DIV2 - lower 6 bits of Baudrate Divisor

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-5		(96000000 / baudrate) & 0x3f

0xbe50002c	4	w	CONTROL

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
6		? (set to 1 if you want to set baudrate)
5		? (set to 1 if you want to set baudrate)

9 Exception Processing

9.1 Exception Cause

The cause of the exception that was raised can be determined by the value of the cause register (causereg >> 2 to be specific) which has the following meaning:

0	INT	Interrupt	Hardware or Software Interrupt.
1	MOD	(n/a) TLB modification	The memory address translation mapped to a TLB entry, but that entr
2	TLBL	(n/a) TLB load/inst fetch	TLB exception caused by a data load (i.e., a load word or similar inst
3	TLBS	(n/a) TLB store	TLB exception caused by a data store (i.e., a store word or similar ins
4	ADEL	Address load/inst fetch	The PC was not word-aligned, or the address the load instruction war
5	ADES	Address store	The address the store instruction wanted to store to was not aligned to
6	IBE	Bus error (instr)	The PC does not correspond to any real area of memory
7	DBE	Bus error (data)	The target address of the load or store instruction does not correspond
8	SYS	Syscall	Some code was trying to call the operating system, using a SYSCALL
9	BP	Breakpoint	Some process executed a BREAK instruction. This is the processors
10	RI	Reserved instruction	Some code executed something which wasn't a valid MIPS-1 instruct
11	CPU	Coprocessor unusable	Some code executed an instruction which tried to reference a coproce
12	OV	Arithmetic overflow	Some code executed an instruction whose arithmetic answer was too
13	TR	Trap	
14	VCEI	Virtual Coherency Exception (instruction).	
15	FPE	FPU Exception	
16		(reserved)	
17		(reserved)	
18		(reserved)	
19		(reserved)	
20		(reserved)	
21		(reserved)	
22		(reserved)	
23	WATCH	Reference to WatchHi/WatchLo address detected.	
24	DEBUG	Debug Exception	
25		(reserved)	
26		(reserved)	
27		(reserved)	
28		(reserved)	
29		(reserved)	
30		(reserved)	
31	VCED	Virtual Coherency Exception (data)	called 'Error' on the PSP

9.2 Reset Vector (HW,SW,NMI)

```
bfc00000(/* v0 */) /* (exceptionman, mebooter, mebooter_umdvideo, me_wrapper, power, sysreg) */
{
    COP0CTRL.6=v0 /* save v0 in cc0.6 (GPR.v0) */
    if(COP0STAT.22!=0) /* get c0.22 (CPU ID?) (if!=0 then ME) */
    {
        goto ME_Reset_Handler; /* jump directly to ME Reset Handler */
    } else {
        call (COP0CTRL.9); /* jump (indirect over vector in cc0.9) to Error Handler (EXC_31_ERROR
handler) */
    }
}
```

9.2.1 ME Reset Handler

```
ME_Reset_Handler() /* bfc00040 (mebooter, mebooter_umdvideo, me_wrapper) */
```

```
{
    *(0xbc100050)=0x00000007; /* bus clock enable AW?/AW?/ME */
    *(0xbc100004)=0xffffffff; /* acknowledge/clear all interrupts */
    *(0xbc100040)=0x00000001; /* set ram size (32mb) */

    k0=COP0STAT.16; /* get c0.16 (Config) */
    COP0STAT.28=0; /* set c0.28 (TagLo) = 0 */
    COP0STAT.29=0; /* set c0.29 (TagHi) = 0 */

    /* invalidate caches */

    k1=0x0800<<((k0>>6)&0x00000007);
    do()
    {
        k1-=0x40;
        asm('cache 0x01, 0($k1)'); /* Index Invalidate (primary Data Cache) */
    } while(k1!=0);
    k1=0x0800<<((k0>>(3))&0x00000007);
    do()
    {
        k1-=0x40;
        asm('cache 0x11, 0($k1)'); /* Hit Invalidate (primary Data Cache) */
    } while(k1!=0);

    COP0STAT.13=0; /* set c0.13 (Cause) = 0 */
    COP0STAT.12=0x20000000; /* set set c0.12 (Status) = 0x20000000 */

    *(0xbcc00010)=0x0001;
    while(*(0xbcc00010)==1){/* wait */};
    *(0xbcc00070)=0x00000001;
    *(0xbcc00030)=0x00000008;
    *(0xbcc00040)=0x00000002;
    sync();

    /* k0=0x88380000 t0=0xbfc00000 sp=0x80200000 */
    88380000(0,0x88300000,0x00080000); /* call handler at 0x88380000 */
}

88380000()
{
...
}
```

9.3 EBASE Vector (IRQ,Syscall)

```
EBase( /* v0, v1 */) /* 8801cd38 */
{
    COP0CTRL.6=v0; /* save v1 in cc0.6 (GPR.v0) */
    COP0CTRL.7=v1; /* save v1 in cc0.7 (GPR.v1) */
    COP0CTRL.0=COP0STAT.30; /* save (EPC) in cc0.0 Exception Program Counter */
    COP0CTRL.2=COP0STAT.12; /* save v1 (Status) in cc0.2 Status register */
```

```
    u32 cause=COP0STAT.13;
    COP0CTRL.3=cause; /* save (Cause) in cc0.3 */;
    cause&=0x7c;
    if(cause!=(8<<2)) /* not syscall?  */
    {
        exception_handler(cause); /* v0=offset in table */
    } else {
        call (COP0CTRL.11); /* jump (indirect over vector in cc0.11) to Syscall Handler (EXC_8_Sysca
handler) */
    }
}
```

9.4 Error Handler

```
EXC_31_ERROR_handler(/* v1 */) /* (exceptionman:0x06c8) */
{
    COP0CTRL.7=v1; /* save v1 in cc0.7 (GPR.v1) */
    COP0CTRL.20=COP0STAT.13; /* save (Cause) in cc0.20 */;
    COP0CTRL.1=COP0STAT.30; /* save (ErrorEPC) in cc0.1 Error Exception Program Counter */
    COP0CTRL.19=COP0STAT.12; /* save v1 (Status) in cc0.19 Status register */
    exception_handler(31<<2); /* v0=0x007c default offset in table */
}
```

9.5 Exception Handler

▷ return from exception using eret

```
exception_handler(u32 offset /* v0 */) /* 8801cd70 (exceptionman:0x0670) */
{
    if (COP0CTRL.25!=NULL) /* Profiler HW Base */
    {
        ; profiler stuff
        *(PROFILER+0x0c)=offset; /* save v0 to PROFILER+0x0c (stall total) */
        v1=*(PROFILER+0x00);
        v0=*(v1+0);
        *(v1+0)=0;
        sync();
        if(*(PROFILER+0x08)==0)
        {
            *(PROFILER+0x04)=v0;
        }
        ; count cpu ticks
        *(PROFILER+0x08)++; /* cpu ck */
        offset=*(PROFILER+0x0c); /* get v0 from PROFILER+0x0c (stall total) */
    }
    /* jump to exception handler from table */
    u8 *Exception_Vector_Table;
    Exception_Vector_Table=COP0CTRL.8; /* Exception Vector Table */
    call((u32)Exception_Vector_Table[offset]);
}

void *ExceptionVectorTable[32] /* 8801ea00 (exceptionman) Exception Vector Table (32 Entries) */
```

```
{
/*  0 */ 88020F74 (interruptman:0x2274) /* IRQ (=default_irq_handler) */
/*  1 */ 8801D130 (hang)while(1);
/*  2 */ 8801D130 (hang)while(1);
/*  3 */ 8801D130 (hang)while(1);
/*  4 */ 8801D130 (hang)while(1);
/*  5 */ 8801D130 (hang)while(1);
/*  6 */ 8801D130 (hang)while(1);
/*  7 */ 8801D130 (hang)while(1);
/*  8 */ 88021E74 (interruptman:0x3174) /* syscall (=EXC_8_Syscall handler) */
/*  9 */ 8801D130 (hang)while(1);
/* 10 */ 8801D130 (hang)while(1);
/* 11 */ 8801D130 (hang)while(1);
/* 12 */ 8801D130 (hang)while(1);
/* 13 */ 8801D130 (hang)while(1);
/* 14 */ 8801D130 (hang)while(1);
/* 15 */ 8801D130 (hang)while(1);
/* 16 */ 8801D130 (hang)while(1);
/* 17 */ 8801D130 (hang)while(1);
/* 18 */ 8801D130 (hang)while(1);
/* 19 */ 8801D130 (hang)while(1);
/* 20 */ 8801D130 (hang)while(1);
/* 21 */ 8801D130 (hang)while(1);
/* 22 */ 8801D130 (hang)while(1);
/* 23 */ 8801D130 (hang)while(1);
/* 24 */ 8801D130 (hang)while(1); /* debug exception */
/* 25 */ 8801D130 (hang)while(1);
/* 26 */ 8801D130 (hang)while(1);
/* 27 */ 8801D130 (hang)while(1);
/* 28 */ 8801D130 (hang)while(1);
/* 29 */ 8801D130 (hang)while(1);
/* 30 */ 8801D130 (hang)while(1);
/* 31 */ 8801D370 (exceptionman:0x0c70) /* error, default (=default_error_handler) */
}
```

note: the PSP Kernel provides a function called `sceKernelRegisterPriorityExceptionHandler` to register a handler in the above table.

9.5.1 error

```
typedef struct
{
    /* 0x00 */ unsigned long unk;
    /* 0x04 */ unsigned long at;
    /* 0x08 */ unsigned long a0;
    /* 0x0c */ unsigned long a1;
    /* 0x10 */ unsigned long a2;
    /* 0x14 */ unsigned long a3;
    /* 0x18 */ unsigned long t0;
    /* 0x1c */ unsigned long t1;
```

```
/* 0x20 */ unsigned long t2;
/* 0x24 */ unsigned long t3;
/* 0x28 */ unsigned long t4;
/* 0x2c */ unsigned long t5;
/* 0x30 */ unsigned long t6;
/* 0x34 */ unsigned long t7;
/* 0x38 */ unsigned long s0;
/* 0x3c */ unsigned long s1;
/* 0x40 */ unsigned long s2;
/* 0x44 */ unsigned long s3;
/* 0x48 */ unsigned long s4;
/* 0x4c */ unsigned long s5;
/* 0x50 */ unsigned long s6;
/* 0x54 */ unsigned long s7;
/* 0x58 */ unsigned long t8;
/* 0x5c */ unsigned long t9;
/* 0x60 */ unsigned long k0;
/* 0x64 */ unsigned long k1;
/* 0x68 */ unsigned long gp;
/* 0x6c */ unsigned long sp;
/* 0x70 */ unsigned long fp;
/* 0x74 */ unsigned long ra;

/* 0x78 */ unsigned long hi;
/* 0x7c */ unsigned long lo;

/* 0x80 */ unsigned long f0;
/* 0x84 */ unsigned long f1;
/* 0x88 */ unsigned long f2;
/* 0x8c */ unsigned long f3;
/* 0x90 */ unsigned long f4;
/* 0x94 */ unsigned long f5;
/* 0x98 */ unsigned long f6;
/* 0x9c */ unsigned long f7;
/* 0xa0 */ unsigned long f8;
/* 0xa4 */ unsigned long f9;
/* 0xa8 */ unsigned long f10;
/* 0xac */ unsigned long f11;
/* 0xb0 */ unsigned long f12;
/* 0xb4 */ unsigned long f13;
/* 0xb8 */ unsigned long f14;
/* 0xbc */ unsigned long f15;
/* 0xc0 */ unsigned long f16;
/* 0xc4 */ unsigned long f17;
/* 0xc8 */ unsigned long f18;
/* 0xcc */ unsigned long f19;
/* 0xd0 */ unsigned long f20;
/* 0xd4 */ unsigned long f21;
/* 0xd8 */ unsigned long f22;
/* 0xdc */ unsigned long f23;
```

```
    /* 0xe0 */ unsigned long f24;
    /* 0xe4 */ unsigned long f25;
    /* 0xe8 */ unsigned long f26;
    /* 0xec */ unsigned long f27;
    /* 0xf0 */ unsigned long f28;
    /* 0xf4 */ unsigned long f29;
    /* 0xf8 */ unsigned long f30;
    /* 0xfc */ unsigned long f31;
} ERRFRAME; /* 0x8801e8c0 */

void *user_error_handler; /* 0x8801d368 */
void *curr_nmi_handler; /* 0x8801d884 */
int flag; /* 0x8801d880 */

default_error_handler(void) /* 8801D370-8801d770 (exceptionman:0x0c70) */
{
    if(flag) goto l8801d76c; // break

    flag++;
    curr_nmi_handler=NULL; /* clear nmi handler addr */
    v0=sp;
    sp=0x8801e8c0;

    /* save at-ra in frame (not shown */
    (ERRFRAME*)sp->hi=mfhi();
    (ERRFRAME*)sp->lo=mflo();
    /* save f0-f31 (not shown) */

    s0=*(0xbc100000);
    if((s0&0x03ff03ff)==0) goto l8801d768; // break

    v1=bitrev(s0); // reverse bit order
    s1=clz(v1); // count left zeros

    if((s0&0x000003ff)==0)
    {
        if((s0&0x03ff0000)==0) goto l8801d768; // break
        a0=1;
        s2=s0>>0x10; // nmi nr
    }
    else
    {
        a0=0;
        s2=s1; // nmi nr
    }


    if(s2==0x00000008)
    {
        v0=0xbc100010;
    }
```

```
else if(s2==0x00000009)
{
    v0=0xbc100028;
}
else
{
    v0=0xbc100034-(s2<<2);
}


v0=*(v0);

if((v0>>0x1f)!=0)
{
    a1=1;
}
else
{
    a1=0;
}

k0=v0&0x80000000;
a3=COP0CTRL.0;
t0=COP0CTRL.1;
v0=COP0CTRL.18; /* NMI vector table addr */
curr_nmi_handler=*(v0+(s2<<2)); /* get addr of handler */

if(curr_nmi_handler)
{
    *(0xbc100004)=s0;
    call(curr_nmi_handler); /* a0=0/1 a1=0/1 k0=0xbc100004; sp=0x8801e8c0; */
}

/* restore f0-f1 (not shown) */
mthi((ERRFRAME*)sp->hi);
mtlo((ERRFRAME*)sp->lo);
/* restore at-ra (not shown) */

flag=0;
COP0STAT.12=COP0CTRL.19&0xffbfffff; /* status */

if(curr_nmi_handler!=NULL)
{
    /* restore remaining regs and return from exception */
    COP0STAT.12=COP0STAT.12&0xffefffff; /* status */
    COP0STAT.13=COP0CTRL.20; /* cause */
    COP0STAT.30=COP0CTRL.1; /* Error EPC */
    v0=COP0CTRL.6;
    v1=COP0CTRL.7;
    eret();
}
```

```
    else
    {
        call(user_error_handler);
    }

l8801d768:
    brk(0x20000);
l8801d76c:
    brk(0x20000);
}
```

9.5.2 interrupt

Number	Subs	Name	Description
0		UART_ALL	
1		SPI_ALL	
2		TIM_PERI_ALL	
3		USB_ALL	
4	32	GPIO	GPIO
5		ATA	ATA/ATAPI
6	16	SPOCK	UMD MAN
7		SMS1	Memstick (MSCM0)
8		SMS2	WLAN
9		MG	
10		AUDIO1	
11		AUDIO2	
12		IIC	I2C
13		KEY	
14		SIRCS	IrDA
15		TIM0_SYS	Systimer 0
16		TIM1_SYS	Systimer 1
17		TIM2_SYS	Systimer 2
18		TIM3_SYS	Systimer 3
19		COUNT	Thread0
20		EMC_SM	NAND
21	10	DMAC128	DMACPLUS
22		DMAC_SC1	DMA0
23		DMAC_SC2	DMA1
24		KIRK	MEMLMD
25	32	AW	GE
26		USB_MAIN	
27			
28			
29			
30	32	VSYNC	Display VBlank
31		SYS_REG	ME Codec
32		UART1	
33		UART2	
34		UART3	
35		UART4	
36		UART5	HP Remote
37		UART6	
38			
39			
40		SPI1	
41		SPI2	

42		SPI3	
43		SPI4	
44		SPI5	
45		SPI6	
46			
47			
48		TIM1_PERI	
49		TIM2_PERI	
50		TIM3_PERI	
51		TIM4_PERI	
52			
53			
54			
55			
56		USB_TS	USB Resume
57		USBCON_TS	USB Ready
58		USBDIS_TS	USB Connect
59		USBREADY_TS	USB Disconnect
60		SMS1_CON	Memstick Insertion (MSCM1)
61		SMS1_DISCON	Memstick Removal (MSCM2)
62		SMS2_CON	WLAN
63		SMS2_DISCON	WLAN
64		SOFT1	
65		SOFT2	Thread1
66		CPUTIMER	Interrupt

9.5.2.1 Interrupt Cause

9.5.2.2 Interrupt Handler `typedef struct`

```
{
    /* 0x00 */ unsigned long unk000; /* some kind of flag */
    /* 0x04 */ unsigned long at;
    /* 0x08 */ unsigned long gprv0;
    /* 0x0c */ unsigned long gprv1;

    /* 0x10 */ unsigned long a0;
    /* 0x14 */ unsigned long a1;
    /* 0x18 */ unsigned long a2;
    /* 0x1c */ unsigned long a3;

    /* 0x20 */ unsigned long t0;
    /* 0x24 */ unsigned long t1;
    /* 0x28 */ unsigned long t2;
    /* 0x2c */ unsigned long t3;
    /* 0x30 */ unsigned long t4;
    /* 0x34 */ unsigned long t5;
    /* 0x38 */ unsigned long t6;
    /* 0x3c */ unsigned long t7;
    /* 0x40 */ unsigned long s0;
    /* 0x44 */ unsigned long s1;
    /* 0x48 */ unsigned long s2;
    /* 0x4c */ unsigned long s3;
    /* 0x50 */ unsigned long s4;
```

```
/* 0x54 */ unsigned long s5;
/* 0x58 */ unsigned long s6;
/* 0x5c */ unsigned long s7;
/* 0x60 */ unsigned long t8;
/* 0x64 */ unsigned long t9;

/* 0x68 */ unsigned long k0;
/* 0x6c */ unsigned long k1;
/* 0x70 */ unsigned long gp;
/* 0x74 */ unsigned long sp;
/* 0x78 */ unsigned long fp;
/* 0x7c */ unsigned long ra;

/* 0x80 */ unsigned long f0;
/* 0x84 */ unsigned long f1;
/* 0x88 */ unsigned long f2;
/* 0x8c */ unsigned long f3;
/* 0x90 */ unsigned long f4;
/* 0x94 */ unsigned long f5;
/* 0x98 */ unsigned long f6;
/* 0x9c */ unsigned long f7;
/* 0xa0 */ unsigned long f8;
/* 0xa4 */ unsigned long f9;
/* 0xa8 */ unsigned long f10;
/* 0xac */ unsigned long f11;
/* 0xb0 */ unsigned long f12;
/* 0xb4 */ unsigned long f13;
/* 0xb8 */ unsigned long f14;
/* 0xbc */ unsigned long f15;
/* 0xc0 */ unsigned long f16;
/* 0xc4 */ unsigned long f17;
/* 0xc8 */ unsigned long f18;
/* 0xcc */ unsigned long f19;
/* 0xd0 */ unsigned long f20;
/* 0xd4 */ unsigned long f21;
/* 0xd8 */ unsigned long f22;
/* 0xdc */ unsigned long f23;
/* 0xe0 */ unsigned long f24;
/* 0xe4 */ unsigned long f25;
/* 0xe8 */ unsigned long f26;
/* 0xec */ unsigned long f27;
/* 0xf0 */ unsigned long f28;
/* 0xf4 */ unsigned long f29;
/* 0xf8 */ unsigned long f30;
/* 0xfc */ unsigned long f31;

/* 0x100 */ unsigned long unk100; /* COP1CTRL.6 */
/* 0x104 */ unsigned long hi;
/* 0x108 */ unsigned long lo;
/* 0x10c */ unsigned long cop0status;
```

```
    /* 0x110 */ unsigned long cop0epc;
    /* 0x114 */ unsigned long cop0cause;
} IRQFRAME;

void * /* [r] 0x88020f6c 'null' handler address */

// these 3 structs are probably the same
typedef struct
{
/* 0x00 */ u32 unk00;
/* 0x04 */ u32 unk04;
} struct88022610 [4]; /* [r/w] 0x88022610 ?  (2628) */

typedef struct
{
/* 0x00 */ u32 unk00;
/* 0x04 */ u32 unk04;
} struct88022630 [4]; /* [r/w] 0x88022630 ?  */

typedef struct
{
/* 0x00 */ u32 unk00;
/* 0x04 */ u32 unk04;
} struct88022650 [4]; /* [r/w] 0x88022650 ?  */

typedef struct
{
/* 0x00 */ void *entry;
/* 0x04 */ void *gp;
/* 0x08 */ u32
/* 0x0c */ u32 calls;
/* 0x10 */ u32 min_clock_lo;
/* 0x14 */ u32 min_clock_hi;
/* 0x18 */ u32 max_clock_lo;
/* 0x1c */ u32 max_clock_hi;
/* 0x20 */ u32 total_clock_lo;
/* 0x24 */ u32 total_clock_hi;
/* 0x28 */ void *
/* 0x2c */ void *
/* 0x30 */ u32
/* 0x34 */ u32
} IntrHandlerOptionParam *IntrHandlerOption[67]; /* [r/w] 88022770 */

/* [r/w] 0x8802277c ?  some flag */
unsigned long long88022780[4]; /* [w] 0x88022780 stackpointer before calling handler */
/* [w] 0x88022790 ?  cop0stat.9 count */
/* [w] 0x88022794 ?  cop0stat.9 count */
/* [w] 0x88022798 ?  */
/* [r/w] 0x8802279c ?  */
/* [r/w] 0x880227a0 ?  counter */
```

```
typedef struct
{
/* 0x00 */ u32 unk00;
/* 0x04 */ u32 unk04;
} struct880227a4; /* [r] 0x880227a4 ?  */

void * /* [r] 880227ac ?  handler address (8803be58 threadman:?)  */
void * /* [r] 880227b0 ?  handler address (8802d724 threadman:?)  */
/* [r] 880227b4 ?  stack stuff */
void * /* [r/w] 880227d0 ?  handler address */

default_irq_handler(void) /* 88020F74 (interruptman:0x2274) */
{
    /*
        some preparations, set up the stack
        (beware of gotos :])
    */

    v1=sp; // original stackpointer
    if(*(0x8802277c)==0) goto l88020fa4;
    if((COP0CTRL.2&0x18)!=0) goto l88020fb8; /* cop0.status */
    goto l88020f94;
l88020fa4:
    *(0x88022790)=COP0STAT.9; /* count */
l88020fb8:
    if(COP0CTRL.14!=0) goto l88020fc4; /* GPR.sp.Kernel */
l88020f94:
    /* allocate and align stackframe */
    sp=(sp+0xffffffe0)&0xffffffc0;
l88020fc4:

    /*
        save environment on the stack
    */

    (IRQFRAME*)sp->at=at;
    (IRQFRAME*)sp->sp=v1; // original stackpointer
    (IRQFRAME*)sp->gprv0=COP0CTRL.4;
    (IRQFRAME*)sp->gprv1=COP0CTRL.5;
    (IRQFRAME*)sp->a0=a0;
    (IRQFRAME*)sp->a1=a1;
    (IRQFRAME*)sp->a2=a2;
    (IRQFRAME*)sp->a3=a3;
    (IRQFRAME*)sp->k0=k0;
    (IRQFRAME*)sp->k1=k1;
    (IRQFRAME*)sp->gp=gp;
    (IRQFRAME*)sp->fp=fp;
    (IRQFRAME*)sp->ra=ra;
    (IRQFRAME*)sp->hi=mfhi();
    (IRQFRAME*)sp->lo=mflo();
```

```
(IRQFRAME*)sp->cop0status=COP0CTRL.2;
(IRQFRAME*)sp->cop0cause=COP0CTRL.3;
(IRQFRAME*)sp->cop0epc=COP0CTRL.0;
(IRQFRAME*)sp->unk100=COP1CTRL.2;

COP1CTRL.6=0;
COP1CTRL.6=0x00000e00;

/*
    alloc space on stack for local variables
*/

(IRQFRAME*)sp->unk=0;

if(*(0x8802277c))
{
    a1=COP0CTRL.15; /* GPR.sp.User */
    k0=sp;
    if(COP0CTRL.2&0x18) sp=a1; /* cop0status */
    a0=*(0x880227b4) + 0x0240;
    at=(sp<a0);
    if(at!=0)
    {
        COP0STAT.12=(IRQFRAME*)sp->cop0status & 0x2fffffe0; /* status */
        while(1)
        {
            brk(0xfff);
        }
    }
    sp+=0xffffffe0; /* alloc 0x20 bytes on stack */
    *(sp+0x001c)=k0; // save pointer to IRQFRAME
}
else
{
    k0=sp;
    sp=0x880257a0;
    *(sp+0x1c)=k0; // save pointer to IRQFRAME

    for(i=0;i<4;i++)
    {
        struct88022630[i]=struct88022610[i];
    }
}

k1=*(0x8802277c);
*(0x8802277c)++;
v0=88022208(); /* also returns v1 */
struct88022650[k1].unk00=v0;
struct88022650[k1].unk04=v1;
long88022780[k1]=sp;
```

```
    /*
        find number of irq (and put it in a0)
    */

    v0=(IRQFRAME*)k0->cop0status & 0x2fffffe0;
    COP0STAT.12=v0; // status

    if(( (v0 & (IRQFRAME*)k0->cop0cause) &0x8300)!=0)
    {
        v1=((v1<<5)|v1)<<0x10;
        v1=clz(v1); // count left zeros
        k1=66-v1; /* 66=highest irq number */
        v0=88022218(struct88022630[0].unk00,struct88022630[0].unk04);
        a0=k1;
    }
    else
    {
        v0=880221d8(); /* also returns v1 */
        for(k0=0;k0<3 /* ?  */;k0++)
        {
            a0=struct88022630[k0+1].unk00; // 0x88022638
            a1=struct88022630[k0+1].unk04; // 0x8802263c
            a2=a0&v0;
            a3=a1&v1;

            if((a2|a3)!=0)
            {
                v0=88022218(struct88022630[k0].unk00,struct88022630[k0].unk04);
k0=a2;
                k1=a3;
                if(k0)
                {
                    a1=bitrev(k0); // reverse bit order
                    a0=clz(a1); // count left zeros
                }
                else
                {
                    if(k1==0)
                    {
                        /* set handler address to 'null' handler and end irq handling */
                        *(0x880227d0)=*(0x88020f6c); // 88021c98
                        goto l880219d4; // call handler *(0x880227d0)
                    }
                    a1=bitrev(k1); // reverse bit order
                    a0=clz(a1); // count left zeros
                    a0+=0x0020;
                }

                goto l880211e4; // call registered handler in a0
            }
```

```
        }

        /* set handler address to 'null' handler and end irq handling */
        *(0x880227d0)=*(0x88020f6c); // 88021c98
        goto l880219d4; // call handler *(0x880227d0)
    }

l880211e4:

    /*
        call registered handler for individual interrupt (in a0)
    */

    k0=IntrHandlerOption[a0]->entry;
    k1=IntrHandlerOption[a0]->entry;


    if((k1==3)||(k1==0)) // no handler registered
    {
        /* set handler address to 'null' handler and end irq handling */
        *(0x880227d0)=*(0x88020f6c); // 88021c98
        goto l880219d4; // call handler *(0x880227d0)
    }
    *(sp+0x0014)=a0;
    *(0x88022798)=a0;

    if(a0!=0x88022798)
    {
        if((a0+0xffffffc0)<0)
        {
            v0=88022234(a0); /* also returns v1 */
        }
        else
        {
            v0=~((v0+1)<<8);
            v1=COP0STAT.13 & v0; /* cause */
            COP0STAT.13=v1; /* cause */
        }
    }

    while(1)
    {
        *(sp+0x0018)=k0; /* k0:  pointer to IntrHandlerOptionParam */
        a1=*(k0+0x0008);

        a2=*(sp+0x001c); // get pointer to IRQFRAME
        a2=(IRQFRAME*)a2->cop0epc;

        gp=*(k0+0x0004);
        v0=k1&0x0003;
        at=0x0003;
```

```
if(v0==at)
{
    *(0x880227d0)=k1&0xfffffffc; /* handler address */
    goto l880219d4; // call handler *(0x880227d0)
}
else if(v0!=0)
{
    v0=*(sp+0x001c); // get pointer to IRQFRAME
    if((IRQFRAME*)v0->unk000!=4)
    {
        /* save t0...t9 in *(v0+0x20...0x64)(not shown) */
        /* save f0...t31 in *(v0+0x80...0xfc)(not shown) */
        (IRQFRAME*)v0->unk000=4;
    }
}


if((*(k0+0x0030) & 0x0100)==0)
{
    ra=COP0STAT.9; /* count */
    v1=struct880227a4.unk04+(ra<struct880227a4.unk00);
    *(sp+0x000c)=ra;
    *(sp+0x0010)=v1;
}

v0=k1&0xfffffffc;
ra=(a0<0x40);
mtic(ra);
k1=0;

call(v0); /* call handler (jal) */

mtic(0);
k0=*(sp+0x0018);
a0=*(k0+0x0030) & 0x0100;

if(a0==0)
{
    *(sp)=v0;
    a3=0x880227a4;
    v1=struct880227a4.unk04;
    a2=((COP0STAT.9)<(struct880227a4.unk00)); /* count */
    v1+=a2;
    a0=*(sp+0x000c);
    a1=*(sp+0x0010);
    v1-=a1;
    a1=(v0<a0);
    v0-=a0;
    v1-=a1;
    a0=*(k0+0x0010);
```

```
    a1=*(k0+0x0014);

    if((a1<v1)==0)
    {
        if((a1!=v1)|((a0<v0)==0))
        {
            *(k0+0x0010)=v0;
            *(k0+0x0014)=v1;
        }
    }

    a0=*(k0+0x0018);
    a1=*(k0+0x001c);

    if((v1<a1)==0)
    {
        if((v1!=a1)|((v0<a0)==0))
        {
            *(k0+0x0018)=v0;
            *(k0+0x001c)=v1;
        }
    }

    a0=*(k0+0x0020) + v0;
    a1=*(k0+0x0024) + v1 + (a0<v0);
    *(k0+0x0020)+=a0;
    *(k0+0x0024)+=a1;
    v0=*(sp);
}

*(0x8802279c)++;
*(k0+0x000c)++;
a0=*(k0+0x0030) & 0x1000;

if(a0!=0) break;

v0++;
if(v0==0) break;

ra=v0+1;
if(ra==0)
{
    a0=*(sp+0x0014);
    v1=66; // 66=number of highest irq
    if(a0==v1) break;
    v0=a0+0xffffffc0;

    if(v0>=0)
    {
        COP0STAT.12=COP0STAT.12&(((v0+1)<<8)^0xffffffff); /* status */
```

```
                break;
}

            /* make bitmask */
            v0=0;v1=0;
            a1=a0+0xffffffe0;

            if(a1<0)
            {
                v0=1<<a0;
            }
            else
            {
                v1=1<<a1;
            }

            v0^=0xffffffff; v1^=0xffffffff;

            /* AND array with mask (0x60 bytes) */
            a2=0x88022610; // start
            a3=a2+0x0060; // end

            do
            {
                *(a2+0)&=v0;
                *(a2+4)&=v1;
                a2+=8;
            } while(a2<a3);

            break;

        }

        v0=(v0-1)<<2;
        ra=*(sp+0x0018);
        ra=*(ra+0x0028);
        a0=*(sp+0x0014);
        k0=*(ra);
        k1=*(k0);
    } // while(1)

    v0=*(0x8802277c) - 1;
    *(0x8802277c)=v0;

    if(v0==0)
    {
        ra=0x88022628;
    }
    else
    {
        k1=v0<<3;
```

```
        ra=0x88022650+k1;
    }


    88022218(*(ra),*(ra+0x0004));

    /* ********** ********** ********** ********** ********** **********
        thread management
    */

    a0=*(sp+0x001c); // pointer to IRQFRAME

    if(*(0x8802277c)==0)
    {
        v0=0x880227a0;
        *(0x880227a0)++;
        call(*(0x880227b0)); /* call handler (jal) (8802d724 threadman:?)  (note:  accesses memory at the second
4mb!)  */
        a0=*(sp+0x001c); // pointer to IRQFRAME

        if(v0!=0)
        {
            if((IRQFRAME*)a0->unk000!=4)
            {
                /* save t0...t9 in *(a0+0x20...0x64)(not shown) */
                /* save f0...t31 in *(a0+0x80...0xfc)(not shown) */
                (IRQFRAME*)a0->unk000=4;
            }

            a0=call(*(0x880227ac)); /* call handler (jal) (8803be58 threadman:?)  returns pointer to IRQFRAME
*/
        }
    }

    /* ********** ********** ********** ********** ********** **********
        restore environment and return from exception
    */


    sp=a0; // get pointer to IRQFRAME
    a0=(IRQFRAME*)sp->unk000; // flag ?

    if(a0==1)
    {
        /* restore f20...f31 from *(sp+0xd0...0xfc) (not shown) */
        COP1CTRL.6=0;
        COP1CTRL.6=(IRQFRAME*)sp->unk100;
        /* restore s0...s7,gp,fp from *(sp+0x40...0x5c,0x70,0x78) (not shown) */
        ra=(IRQFRAME*)sp->ra; /* handler address */
        v0=0x0008ff00;
        COP0STAT.12=((IRQFRAME*)sp->cop0status & (~v0))|(COP0STAT.12 & v0); /* Status */
```

```
    *(0x88022794)=COP0STAT.9; /* Count */
    /* restore k0,k1,sp from *(sp+0x68,0x6c,0x74) (not shown) */
    v0=1;
    call(ra); /* call handler (j) */
    /* never reaches here ********** ********** ********** */
}
else if(a0!=0)
{
    /* restore f0...f31 from *(sp+0x80...0xfc) (not shown) */
    mthi((IRQFRAME*)sp->hi);
    mtlo((IRQFRAME*)sp->lo);
    /* restore at...fp from *(sp+0x04...0x78) (not shown) */
}
else
{
    mthi((IRQFRAME*)sp->hi);
    mtlo((IRQFRAME*)sp->lo);
    /* restore at...a3,gp,fp from *(sp+0x04...0x1c,0x70,0x78) (not shown) */
}

ra=0x0008ff00;
COP0STAT.12=(((IRQFRAME*)sp->cop0status) & (~ra)) | (COP0STAT.12 & ra); /* status */
COP0STAT.14=(IRQFRAME*)sp->cop0epc; /* epc */
COP1CTRL.6=0;
COP1CTRL.6=(IRQFRAME*)sp->unk100;
*(0x88022794)=COP0STAT.9; /* count */;

/*
    Profiler Stuff
*/

if(COP0CTRL.25!=0) /* PROFILER_BASE */
{
    k0=*(PROFILER_BASE+0x0008);
    if(k0!=0)
    {
        k0--;
        *(PROFILER_BASE+0x0008)=k0;
        if(k0==0)
        {
            k0=*(PROFILER_BASE+0x0004);
            k1=*(PROFILER_BASE);
            *(k1)=k0;
            sync();
        }
    }
}

/* restore k0,k1,ra,sp from *(sp+0x68,0x6c,0x7c,0x74) (not shown) */
eret();
```

```
    /* ********** ********** ********** ********** ********** **********
        restore environment and call handler *(0x880227d0)
    */


l880219d4:
    v0=*(0x8802277c)-1;
    *(0x8802277c)=v0;
    88022218(struct88022650[v0].unk00,struct88022650[v0].unk04);

    sp=*(sp+0x001c); /* get pointer to IRQFRAME */

    if((IRQFRAME*)sp->unk000==4)
    {
        /* restore t0...t9 from *(sp+0x20...0x64) (not shown) */
        /* restore f0...f31 from *(sp+0x80...0xfc) (not shown) */
    }

    mtic(0);
    COP0CTRL.3=(IRQFRAME*)sp->cop0cause; /* cop0.cause */
    COP0CTRL.0=(IRQFRAME*)sp->cop0epc; /* cop0.epc */
    COP0CTRL.4=(IRQFRAME*)sp->gprv0; /* gpr.v0 */
    COP0CTRL.5=(IRQFRAME*)sp->gprv1; /* gpr.v1 */
    v0=0xfff700ff;
    COP0CTRL.2=((IRQFRAME*)sp->cop0status & v0) |((~v0) & COP0STAT.12); /* cop0.status, Status */
    /* restore at,a0...a3,k0..gp,fp,ra from *(sp+...)  (not shown) */
    mthi((IRQFRAME*)sp->hi);
    mtlo((IRQFRAME*)sp->lo);
    COP0STAT.12=(IRQFRAME*)sp->cop0status; /* Status */
    COP1CTRL.6=0;
    COP1CTRL.6=(IRQFRAME*)sp->unk100;
    COP0CTRL.4=(IRQFRAME*)sp->gprv0; /* gpr.v0 */
    COP0CTRL.5=(IRQFRAME*)sp->gprv1; /* gpr.v1 */
    sp=(IRQFRAME*)sp->sp;
    call(*(0x880227d0)) /* call handler (j) */

    /* will never reach here ********** ********** ********** */
}

/* 'null' handler */
void 88021c98(void)
{
    COP0STAT.14=COP0CTRL.0;
    COP0STAT.12=COP0CTRL.2;
    v0=COP0CTRL.4;
    v1=COP0CTRL.5;
    eret();
}
```

```
unsigned long 880221d8(void)
{
    v0=*(0xbc300000) & 0xfffffff0;
    v1=*(0xbc300010);
    return v0; /* also v1 */
}

unsigned long 88022208(void)
{
    v0=*(0xbc300008);
    v1=*(0xbc300018);
    return v0; /* also v1 */
}

unsigned long 88022218(unsigned long a0, unsigned long a1)
{
    *(0xbc300008)=a0|0x0000000f;
    *(0xbc300018)=a1;
    sync();
    return 0xbc300000;
}

unsigned long 88022234(unsigned long a0)
{
    if((0x1f>=a0)&(a0>=0x1e))
    {
        v1=1<<a0;
        v0=0xbc300000;
        *(0xbc300000)=v1;
        sync();
    }
    return v0; /* also v1 */
}
```

9.5.2.3 Thread Management

```
// note: this is the first of two routines called by the interrupt handler
unsigned long 8802d724(void) /* 8802d724 - threadman:  ?  */
{
    a1=0x88040000;
    a0=0x88042a08;
    a2=*(a0+0x0418); // 0x88042e20
    v0=0;
    if(a2==0)
    {
        v1=*(a1+0x2a08); // 0x88042a08
        a2=*(a0+0x0004); // 0x88042a0c
        a1=a2^v1;
        v0=(0<a1);
        if(v0!=0)
        {
```

```
            *(v1+0x00e4)=v1+0x00e8;
        }
    }
    return v0;
}
```

// note: this is the second of two routines called by the interrupt handler

```
8803be58( /* a0 */ ) /* 8803be58 - threadman:?  */
{
    /*
        create stackframe (0x10 bytes) and save s0,s1,s2,ra (not shown)
    */

    s2=0x88040000;
    v1=s2+0x2a08;
    a1=*(v1+0x418); // 0x88042e20
    v0=a0;

    if(a1==0)
    {
        s0=*(s2+0x2a08);
        v0=*(s0+0x000c);

        if(*(s0+0x0108)!=0)
        {
            a0=0xbc400000; /* PROFILER+0x00 */
            a2=v0+0x0010;
            a3=0xbc400000; /* PROFILER+0x00 */
            t0=0xbc400050; /* PROFILER+0x50 */
/* copy profiler regs to *(a2) (0x50 bytes) */
            do
            {
                t3=*(a3+0x00);
                t2=*(a3+0x04);
                a1=*(a3+0x08);
                t1=*(a3+0x0c);
                *(a2)=t3;
                a3+=0x10;
                a2+=0x10;
                *(a2+0xfffffff4)=t2;
                *(a2+0xfffffff8)=a1;
                *(a2+0xfffffffc)=t1;
            } while(a3!=t0);
            v0=*(a3);
            *(a2)=v0;
            v0=*(s0+0x000c);
        }


        a2=+0x0020;
```

```
        s1=s2+0x2a08;
        if(v0==a2) goto 0x8803bf48;
        a2=*(s0+0x0070);
        t0=*(s0+0x00f4);
        a1=*(s0+0x0008);
        a3=*(a2);
        s1=*(t0+0x0074);

        if(a3!=a1)
        {
            a3=*(s0+0x0074);
            t8=0x88040000;
            a0=0x88042634;
            t0=s1;
            880412b8(0x88042634,0x00000000,0x00000000,0x00000000)
            t7=s2+0x2a08;
            t6=*(t7+0x0640);
            t5=*(u8*)(t6+0x15);
            t4=t5<<2;
            a0=s0-t4;
            880405a0(0x00000000,0x00000000,0x00000000,0x00000000)
        }

l8803bf28:
        t9=*(s0+0x00d0);
        a2=*(s0+0x007c);
        if(t9<0) goto l8803c150;
        a2=*(s0+0x0070);

l8803bf38:
        t4=(s1<a2);
        if(t4!=0) goto 0x8803c11c;
        a3=*(s0+0x0074);
        s1=s2+0x2a08;
l8803bf48:
        v0=*(s1+0x0004);

        if(v0==0)
        {
            88040310(a0,a1,a2,a3);
            v0=*(s1+0x0004);
        }

        if(s0==v0)
        {
            v1=t6+0x0001;
            *(s1+0x0680)=v1;
        }
        else
        {
```

```
    t6=*(s1+0x0680);
    88038090(0x00000000,0x00000000,0x00000000,0x00000000);

    t7=s1+0x0428;
    t9=*(t7+0x0004);
    t1=*(s1+0x0428);
    t6=s0+0x0064;
    a1=*(s0+0x0064);
    t2=*(t6+0x0004);
    a0=0;
    t5=t9<<0;
    a3=0;
    t9=a0+t1;
    t8=0;
    t3=(t9<t1);
    t4=t2<<0;
    a2=t5+a3;
    t2=t8+a1;
    t5=a2+t3;
    a0=t4+a3;
    t3=(t2<a1);
    a1=a0+t3;
    t8=(v0<t9);
    t3=v1-t5;
    a0=v0-t9;
    t5=t2+a0;
    t4=t3-t8;
    t9=(t5<a0);
    t8=a1+t4;
    t0=t8+t9;
    t3=v1>>0;
    t2=t0>>0;
    *(t7+0x0004)=t3;
    *(s1+0x0428)=v0;
    *(t6+0x0004)=t2;
    *(s0+0x0064)=t5;
    a0=*(s1+0x067c);
    t9=*(s0+0x00e4);
    a1=a0+0x0001;
    *(s1+0x067c)=a1;
    t1=*(t9);
    v1=t1+0x0001;
    *(t9)=v1;
}


s1=s2+0x2a08;
v1=*(s1+0x0738);
s2=*(s1+0x0004);
```

```
if(v1!=0)
{
    t0=*(s0+0x0010);
    a3=*(s0+0x0008);
    t2=*(s2+0x0010);
    t1=*(s2+0x0008);
    a0=+0x0001;
    a1=0;
    a2=0x00000004;
    call(v1);
    s0=*(s1+0x0004);
}
else
{
    s0=*(s1+0x0004);
}


s1=0x88040000;
v1=*(s0+0x0108);
*(s1+0x2a08)=s0; //0x88042a08

if(v1==0)
{
    COP0CTRL.25=0x00000000; /* PROFILER_BASE */
    a0=*(s0+0x00f4);
}
else
{
    t0=v1+0x0060;
    a3=0xbc400000; /* PROFILER+0x00 */
    a2=v1+0x0010;

    do
    {
        t8=*(a2);
        a1=*(a2+0x0004);
        t4=*(a2+0x0008);
        t7=*(a2+0x000c);
        *(a3)=t8;
        a2+=0x10;
        a3+=0x10;
        *(a3-0x0c)=a1;
        *(a3-0x08)=t4;
        *(a3-0x04)=t7;
    } while(a2!=0);


    v0=*(a2);
    *(a3)=v0;
```

```
            sync();
            t0=*(s0+0x0108);
            COP0CTRL.25=t0; /* PROFILER_BASE */
            a0=*(s0+0x00f4);
        }


        v0=0x40000000;
        t2=*(a0+0x010c);
        a3=t2&v0;

        if(a3!=0)
        {
            a0=s0;
            8803c1b4(0x00000000,0x00000000,0x00000000,0x00000000);
            a0=*(s0+0x00f4);
        }


        COP0CTRL.14=a0; /* GPR.sp.KERNEL */
        t3=*(s0+0x0104);
        COP0CTRL.15=t3; /* GPR.sp.USER */
        a2=s0+0x0100;
        COP0CTRL.16=a2; /* CurrentTCB */
        v0=*(s0+0x00f4);
    }

    /*
        restore ra,s2,s1,s0 and destroy stackframe (0x10 bytes) (not shown)
    */

    return v0;
l8803c11c:
    a1=*(s0+0x0008);
    t8=0x88040000;
    a0=t8+0x2634;
    t0=s1;
    880412b8(0x88042634,0x00000000,0x00000000,0x00000000)
    t7=s2+0x2a08;
    t6=*(t7+0x0640);
    t5=*(u8*)(t6+0x15);
    s1=t5<<2;
    a0=s0-s1;
    880405a0(0x00000000,0x00000000,0x00000000,0x00000000)
    s1=s2+0x2a08;
    goto l8803bf48;
l8803c150:
    a1=*(s0+0x0008);
    v1=*(a2);
    t3=0x88040000;
```

```
    if(v1==a1) goto 0x8803c188;
    a3=*(s0+0x0080);
    a0=t3+0x2634;
    t0=s1;
    880412b8(0x88042634,0x00000000,0x00000000,0x00000000);
    t2=s2+0x2a08;
    a0=*(t2+0x0640);
    a1=*(u8*)(a0+0x15);
    t1=a1<<2;
    a0=s0-t1;
    880405a0(0x00000000,0x00000000,0x00000000,0x00000000)
l8803c188:
    v0=*(s0+0x00f4);
    a3=*(v0+0x010c);
    a2=*(s0+0x0070);
    if((a3&0x0018)!=0) goto 0x8803bf38;
    a2=*(s0+0x007c);
    t0=(s1<a2);
    s1=s2+0x2a08;
    if(t0==0) goto 0x8803bf48;
    a3=*(s0+0x0080);
    goto l8803c11c;
}

// called by 8803be58
88038090()
{
    ...
}

// called by 8803be58
8803c1b4(/* a0 */) /* 8803c1b4 - threadman:  ?  */
{
    sp+=0xfffffff0;
    *(sp+0x0008)=s2;
    s2=0x88040000;
    *(sp+0x0004)=s1;
    s1=a0;
    a0=s2+0x2a08;
    *(sp+0x000c)=ra;
    *(sp)=s0;
    v0=*(0x88042e24);
    a1=*(s1+0x00fc);
    ra=*(sp+0x000c);
    if(v0!=a1)
    {
        v1=*(s1+0x00d0);
        a1=*(a0+0x0420);
        s0=v1>>0x1f;
        s0_d=s0;
```

```
        s0=s2+0x2a08;
        if(a1!=s0_d)
        {
            8802d984(0x00000000,0x00000000,0x00000000,0x00000000)
            if(s0==0)
            {
                a3=*(0xbc000044);
                v0=a3|0x0020;
            }
            else
            {
                a0=*(0xbc000044);
                a2=0xffffffdf;
                v0=a0&a2;
            }
            at=0xbc000000;
            *(0xbc000044)=v0;
            sync();
            t0=s2+0x2a08;
            *(t0+0x0420)=s0;
            s0=s2+0x2a08;
        }
        t1=*(s0+0x041c);
        a0=t1;
        if(t1!=0)
        {
            8802d760(0x00000000,0x00000000,0x00000000,0x00000000);
            t4=*(s1+0x00fc);
        }
        else
        {
            t4=*(s1+0x00fc);
        }
        *(s0+0x041c)=t4;
        8802d874(t4,0x00000000,0x00000000,0x00000000);
        t3=*(s0+0x0684);
        t2=t3+0x0001;
        *(s0+0x0684)=t2;
        ra=*(sp+0x000c);
    }
    s2=*(sp+0x0008);
    s1=*(sp+0x0004);
    s0=*(sp);
    sp+=0x0010;
    return v0;
}

8802d760()
{
```

```
    ...
}

8802d874()
{
    ...
}

8802d984()
{
    ...
}

/* following functions are located in the second 'protected' 4mb */

88040310()
{
    ...
}

880405a0()
{
    ...
}

880412b8()
{
    ...
}
```

9.5.3 syscall

```
typedef struct _SCTABHDR
{
    struct _SCTABHDR *next; /* pointer to next table */
    unsigned long offset; /* offset to substract from syscall code */
    unsigned long num; /* number of entries in list*/
    unsigned long unk; /* ?  */
} SCTABHDR;

typedef struct
{
    unsigned long status; /* COP0CTRL.2 */
    unsigned long epc; /* COP0STAT.14 */
    unsigned long sp; /* sp*/
    unsigned long ra; /* ra*/
    unsigned long k1; /* k1*/
    unsigned long unk14; /* COP1CTRL.2*/
    unsigned long unk18; /* COP0CTRL.4*/
    unsigned long tcb; /* *(COP0CTRL.16) */
```

```
} SCFRAME;

EXC_8_Syscall_handler(/* v0, v1 */) /* 88021e74-88022018 (interruptman:0x3174) */
{
    v0=COP0CTRL.0 /* COP0.EPC */
    v1=COP0CTRL.3 /* COP0.Cause */
    t6=COP0CTRL.13 /* max sc */
    t7=COP0STAT.21 /* sc code */
    v0+=4;
    COP0STAT.14=v0; /* EPC */
    t4=COP0CTRL.12; /* sc tab */
    if(t7<=t6) /* if syscall is in range */
    {
        t4+=t7; /* sc tab + sc code */
        t7=*(t4+0x10)
        if(v1>=0)
        {
            call(t7); /* call regular individual syscall handler */
        }
        while(1)
        {
            break #ffe
        }
    }
    /* further handling for syscall that is not in range */
    if(v1>>=0x1f) v0=ra;
    COP0STAT.14=v0; /* EPC */
    do
    {
        t4=*(t4+0); /* 0x88026820 (8802379c 0) */
        t5=*(t4+4); /* (0x00) 0x8000 (0 x) */
        t6=*(t4+8); /* (0xfc) 0xbffc (0 x) */
        if(t5==0)
        {
            COP0STAT.14=COP0CTRL.0;
            v0=*(0x88021e6c); /* ?  reverse further */
            call($v0);
        }
    } while((t7<t5)|(t6<t7)); /* sccode<t5 or scnum<sccode */
    t7-=t5; /* sccode-=offset */
    t4+=t7; /* sctab+=sccode */
    t7=*(t4+0x10); /* get handler address */

/* get stackframe address */
    if(COP0CTRL.2&0x0018==0) t4=sp; /* COP0.Status */
    else t4=COP0CTRL.15; /* GPR.sp.USER */
    t4-=sizeof(SCFRAME);

(SCFRAME*)t4->status=COP0CTRL.2; /* COP0.Status */
    (SCFRAME*)t4->epc=COP0STAT.14; /* EPC */
```

```
    (SCFRAME*)t4->sp=sp;
    (SCFRAME*)t4->ra=ra;
    (SCFRAME*)t4->k1=k1;
    (SCFRAME*)t4->unk18=COP0CTRL.4; /* GPR.v0 */
    (SCFRAME*)t4->unk14=COP1CTRL.2;

COP1CTRL.6=0;
    COP1CTRL.6=0x00000e00;

/* set frame and call handler */
    sp=t4;
    t6=COP0CTRL.16; /* current.TCB */
    if(t6!=0)
    {
        (SCFRAME*)t4->tcb=*(t6);
        *(t6)=sp;
    }
    k1=(COP0CTRL.2&0x00ff)<<16; /* COP0.Status */
    COP0STAT.12=COP0CTRL.2&0x0000ffe5; /* status */
    call(t7);

/* restore original frame and return */
    mtic(0);
    COP0STAT.12=((SCFRAME*)sp->status&0xfff700ff)|(COP0STAT.12&0x0008ff00); /* status */
    t6=COP0CTRL.16; /* current.TCB */
    if(t6!=0)
    {
        *(t6)=(SCFRAME*)sp->tcb;
    }
    COP1CTRL.6=0;
    COP1CTRL.6=(SCFRAME*)sp->unk14;
    k1=(SCFRAME*)sp->k1;
    ra=(SCFRAME*)sp->ra;
    COP0STAT.14=(SCFRAME*)sp->epc; /* EPC */
    sp=(SCFRAME*)sp->sp;
    eret();
}
```

9.6 Debug Exception Vectors

▷ return from exception using `dreg`

```
bfc01000(/* v0, v1 */) /* (exceptionman, power) */
{
    COP0CTRL.26=v0 /* save v0 in cc0.26 (Ex.GPR.v0) */
    call (COP0CTRL.10); /* jump (indirect over vector in cc0.10) */
}
```

following handlers look all like the one above

```
bfc01100(/* v0 */)
bfc01200(/* v0 */)
```

```
bfc01300(/* v0 */)
bfc01400(/* v0 */)
bfc01500(/* v0 */)
bfc01600(/* v0 */)
bfc01700(/* v0 */)
bfc01800(/* v0 */) /* (me_wrapper) */
bfc01900(/* v0 */)
bfc01a00(/* v0 */)
bfc01b00(/* v0 */)
bfc01c00(/* v0 */)
bfc01d00(/* v0 */)
bfc01e00(/* v0 */)
bfc01f00(/* v0 */)
```

9.6.1 Debug Handler

```
typedef struct
{
    unsigned long flags;
    unsigned long unknown; /* probably DRCTRL */
    unsigned long IBC;
    unsigned long DBC;
    unsigned long IBA;
    unsigned long IBAM;
    unsigned long DBA;
    unsigned long DBAM;
    unsigned long DBD;
    unsigned long DBDM;
} DBGENV;
DBGENV dbgenv;

debug_handler(/* v1 */) /* 8801ce30 (exceptionman:0x0730) */
{
DBGENV *env;
    COP0CTRL.27=v1; /* save v1 */
    env=COP0CTRL.28; /* v0=8801ec10 */
    v1=env->flags;
    if(v1&0x0004)
    {
        goto dbg_handler_0005( env /* v0 */ ); /* store debug environment */
    }
    else if(v1&0x0008)
    {
        goto dbg_handler_000a( env /* v0 */ ); /* restore debug environment */
    }
    else if(v1&0x0001)
    {
        goto dbg_handler_0005( env /* v0 */ ); /* store debug environment */
    }
    else if(v1&0x0002)
    {
        goto dbg_handler_000a( env /* v0 */ ); /* restore debug environment */
```

```
    }
    else if(v1&0x0010) /* single step in kernel mode one instuction then continue */
    {
        goto dbg_handler_0010( env /* v0 */ );
    }
    else if(v1&0x0020) /* single step one instruction in user mode and continue */
    {
        goto dbg_handler_0020( env /* v0 */ );
    }
    else if(v1&0x0040) /* single step in kernel mode one instuction then break into debugger */
    {
        goto dbg_handler_0040( env /* v0 */ );
    }
    else if(v1&0x0080) /* single step one instruction in user mode then break into debugger */
    {
        goto dbg_handler_0080( env /* v0 */ );
    }
    else if(v1&0x0100) /* clear step mode */
    {
        goto dbg_handler_0100( env /* v0 */ );
    }

    /* default */

    DRCNTL&=0xffdf;
    *(COP0CTRL.28+4)=DRCNTL;
    COP0CTRL.4 =COP0CTRL.26; /* GPR.v0 = Ex.GPR.v0 */
    COP0CTRL.5 =COP0CTRL.27; /* GPR.v1 = Ex.GPR.v1 */
    COP0CTRL.0 = DEPC /* COP0.EPC=DEPC */
    DEPC=&8801d10c; /* -> below */
    COP0CTRL.3=COP0STAT.13; /* COP0.Cause = Cause */
    COP0CTRL.1=COP0STAT.30; /* COP0.EPC.err = ErrorEPC */
    v0=COP0STAT.12; /* Status */
    COP0CTRL.2=v0; /* COP0.Status = v0 */
    COP0STAT.12=v0|0x00000002;
    dret();
}

8801d10c()
{
    exception_handler(24<<2);
}
```

this code immediatly follows (?)

```
8801d118( /* v0 */ )
{
    COP0CTRL.26=v0; // save v0
    v0=COP0CTRL.10; // debug handler address
    call(v0) // call debug handler
}
```

9.6.1.1 Debug Sub Handler 0005 dbg_handler_0005(DBGENV *env /* v0 */) /* 0x8801cf30 */

```
{
    DEPC+=4;

    env->flags=0;
    env->IBC=IBC;
    env->DBC=DBC;
    env->IBA=IBA;
    env->IBAM=IBAM;
    env->DBA=DBA;
    env->DBAM=DBAM;
    env->DBD=DBD;
    env->DBDM=DBDM;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.2 Debug Sub Handler 000a dbg_handler_000a(DBGENV *env /* v0 */) /* 0x8801cf90 */

```
{
    DEPC+=4;
    env->flags=0;
    IBC=env->IBC;
    DBC=env->DBC;
    IBA=env->IBA;
    IBAM=env->IBAM;
    DBA=env->DBA;
    DBAM=env->DBAM;
    DBD=env->DBD;
    DBDM=env->DBDM;

    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.3 Debug Sub Handler 0010 dbg_handler_0010(DBGENV *env /* v0 */) /* 0x8801cff0 */

```
{
    env->flags=0x0100; /* clear step mode */
    DEPC=COP0STAT.14; /* DEPC=COP0.EPC */
    COP0STAT.12&=0xfff9; /* COP0.Status */
    DRCNTL|=0x0020;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.4 Debug Sub Handler 0020 dbg_handler_0020(DBGENV *env /* v0 */) /* 0x8801d02c */

```
{
    env->flags=0x0100; /* clear step mode */
    DEPC=COP0STAT.14; /* DEPC=EPC */
    COP0STAT.12=(COP0STAT.12&0xfff9)|0x0010; /* COP0.Status */

    DRCNTL|=0x0020;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.5 Debug Sub Handler 0040 dbg_handler_0040(DBGENV *env /* v0 */) /* 0x8801d070 */

```
{
    env->flags=0;
    DEPC=COP0STAT.14; /* DEPC=EPC */
    COP0STAT.12&=0xfff9; /* COP0.Status */
    DRCNTL|=0x0020;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.6 Debug Sub Handler 0080 dbg_handler_0080(DBGENV *env /* v0 */) /* 0x8801d0a8 */

```
{
    env->flags=0;
    DEPC=COP0STAT.14; /* DEPC=EPC */
    COP0STAT.12=(COP0STAT.12&0xfff9)|0x0010; /* COP0.Status */
    DRCNTL|=0x0020;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

9.6.1.7 Debug Sub Handler 0100 dbg_handler_0100(DBGENV *env /* v0 */) /* 0x8801d0e8 */

```
{
    env->flags=0;
    DRCNTL&=0xffdf;
    v0=COP0CTRL.26; /* restore v0 */
    v1=COP0CTRL.27; /* restore v1 */
    dret();
}
```

10 Video Processing

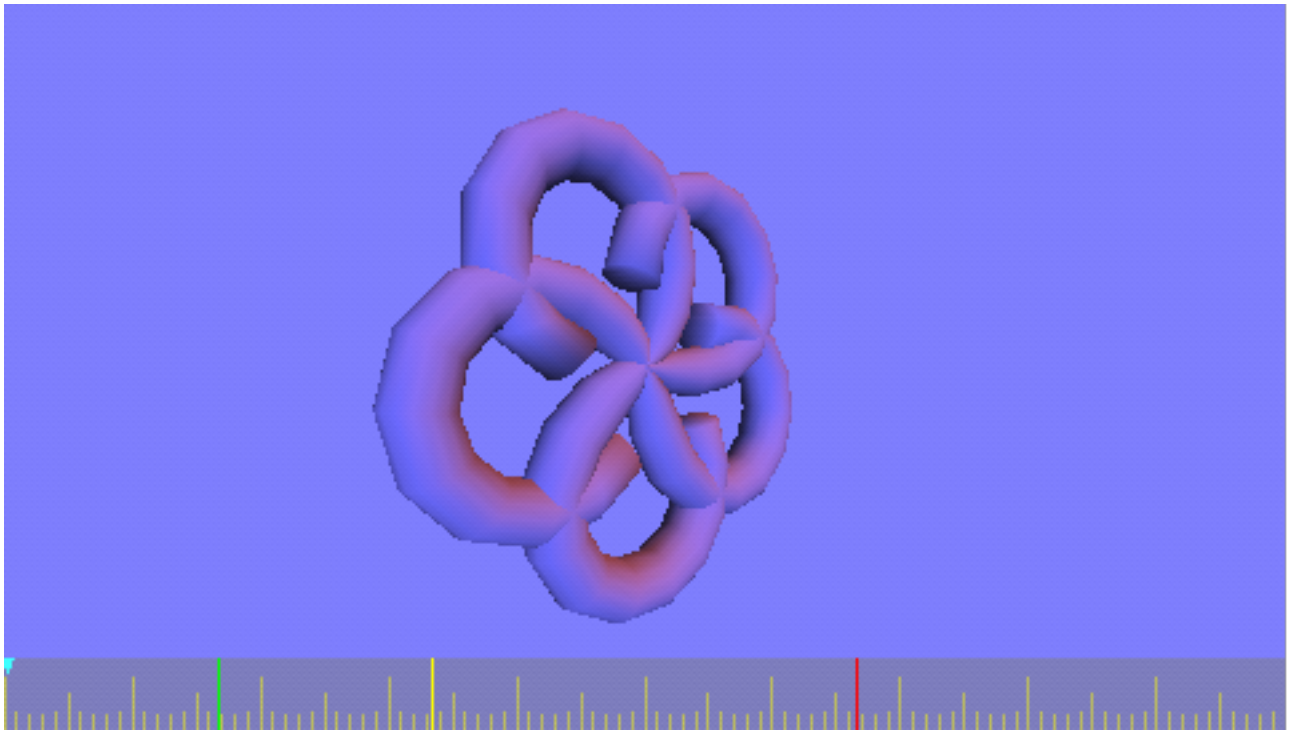
10.1 Overview

- ▷ vram is located at `0x04000000`
- ▷ Pixel format is 16 bit BGR (ABBBBBGGGGGRRRRR.) or 32 bit
- ▷ visible Screen is 480*272 pixel
- ▷ virtual Screensize is 512*272 pixel

10.2 VRAM Mirrors

Writing to the VRAM Mirrors seem to have no effect; setting the drawbuffer pointer to one of these VRAM aliases just works as normal. So these Mirrors only have effects for reads, but work for all readers. (GE, Framebuffer scandout...)

10.2.1 VRAM



10.2.1.1 Depth Buffer The raw depth buffer in the normal VRAM space is rearranged in a swizzled-like way. This is the raw dump of the depth buffer converted to an 8bpp greyscale:

10.2.2 VRAM +2Mib

VRAM with "swizzle"

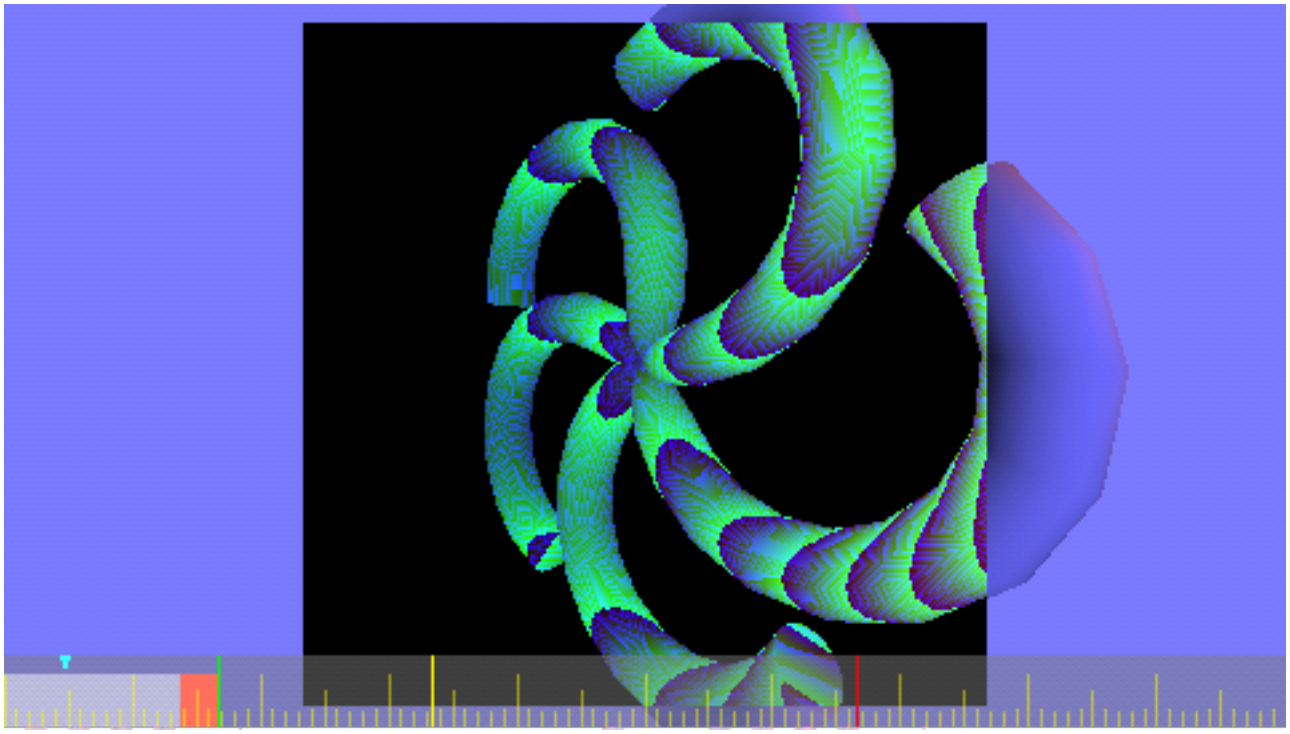
This is clearly a fairly simple structure, with a simple column-wise rearrangement of each 16 pixel (32 byte) strip. When rearranged, it looks as expected:

10.2.3 VRAM +4Mib

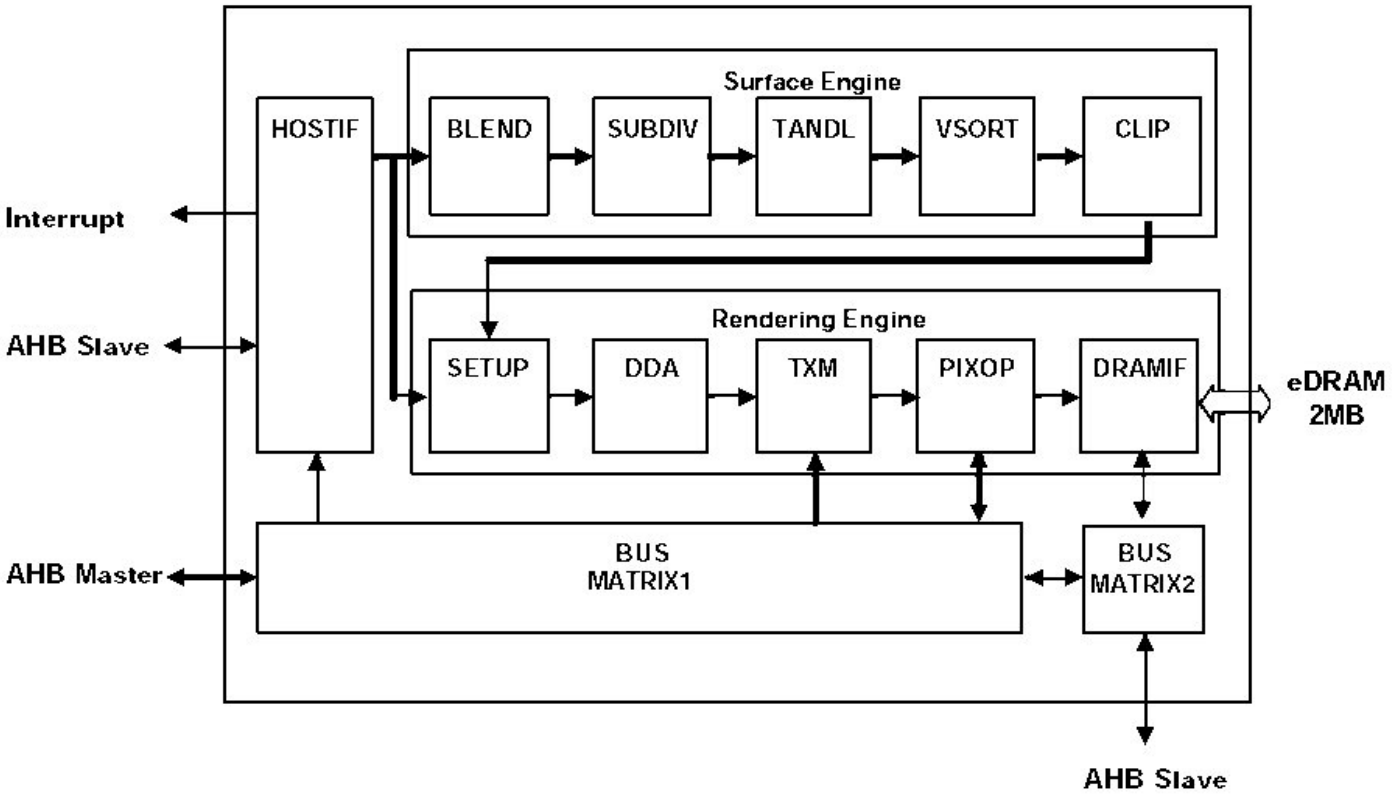
identical to normal VRAM

10.2.4 VRAM +6Mib

VRAM with "swizzle" + 32-byte column interleave. Reading from VRAM+6Mib will give you a proper linearized version of the depth buffer with no effort. The GE sees the same view; a GE copy operation returns the same data (represented as RGB 565):



11 3D Graphics Processing



11.1 GE Command Format

Each command word is divided into two parts, a 8-bit command and a 24-bit argument. The command is in the upper part of the word, and the argument in the lower. The argument can be either integer of a special kind of float that the GE supports (described below).

11.2 GE Floats

Floats processed in the command-stream are 24 bits instead of 32 that are used by the CPU. Conversion from 32 to 24 bits is done by shifting the value down 8 bits, losing the least significant bits of the mantissa.

11.3 Pointers

Some pointers use a shared register when loading addresses called BASE. This register must be written BEFORE you write to the designated register. All these registers are marked with (BASE) after the summary.

Other pointers only use 28 bits of information, and their top bits are referred to as the '4 most significant bits' in pointer, which reflects bits 24-27, not 28-31 which could perhaps be believed from common terminology.

11.4 Enabling Registers

Any command or bit that has 'Enable' in the name implies that setting the first bit (or the bit itself) enables the feature, and no ON/OFF-states are documented.

11.5 GE Command List

num	name	description
0x00	NOP	No Operation
0x01	VADDR	Vertex List (BASE)
0x02	IADDR	Index List (BASE)

0x03		
0x04	PRIM	Primitive Kick
0x05	BEZIER	Bezier Patch Kick
0x06	SPLINE	Spline Surface Kick
0x07	BBOX	Bounding Box
0x08	JUMP	Jump To New Address (BASE)
0x09	BJUMP	Conditional Jump (BASE)
0x0A	CALL	Call Address (BASE)
0x0B	RET	Return From Call
0x0C	END	Stop Execution
0x0D		
0x0E	SIGNAL	Raise Signal Interrupt
0x0F	FINISH	Complete Rendering
0x10	BASE	Base Address Register
0x11		
0x12	VTYPE	Vertex Type
0x13	???	Offset Address (BASE)
0x14	???	Origin Address (BASE)
0x15	REGION1	Draw Region Start
0x16	REGION2	Draw Region End
0x17	LTE	Lighting Enable
0x18	LTE0	Light 0 Enable
0x19	LTE1	Light 1 Enable
0x1A	LTE2	Light 2 Enable
0x1B	LTE3	Light 3 Enable
0x1C	CPE	Clip Plane Enable
0x1D	BCE	Backface Culling Enable
0x1E	TME	Texture Mapping Enable
0x1F	FGE	Fog Enable
0x20	DTE	Dither Enable
0x21	ABE	Alpha Blend Enable
0x22	ATE	Alpha Test Enable
0x23	ZTE	Depth Test Enable
0x24	STE	Stencil Test Enable
0x25	AAE	Anitaliasing Enable
0x26	PCE	Patch Cull Enable
0x27	CTE	Color Test Enable
0x28	LOE	Logical Operation Enable
0x29		
0x2A	BOFS	Bone Matrix Offset
0x2B	BONE	Bone Matrix Upload
0x2C	MW0	Morph Weight 0
0x2D	MW1	Morph Weight 1
0x2E	MW2	Morph Weight 2
0x2F	MW3	Morph Weight 3
0x30	MW4	Morph Weight 4
0x31	MW5	Morph Weight 5
0x32	MW6	Morph Weight 6
0x33	MW7	Morph Weight 7
0x34		
0x35		
0x36	PSUB	Patch Subdivision
0x37	PPRIM	Patch Primitive
0x38	PFACE	Patch Front Face
0x39		
0x3A	WMS	World Matrix Select
0x3B	WORLD	World Matrix Upload
0x3C	VMS	View Matrix Select

0x3D	VIEW	View Matrix upload
0x3E	PMS	Projection matrix Select
0x3F	PROJ	Projection Matrix upload
0x40	TMS	Texture Matrix Select
0x41	TMATRIX	Texture Matrix Upload
0x42	XSCALE	Viewport Width Scale
0x43	YSCALE	Viewport Height Scale
0x44	ZSCALE	Depth Scale
0x45	XPOS	Viewport X Position
0x46	YPOS	Viewport Y Position
0x47	ZPOS	Depth Position
0x48	USCALE	Texture Scale U
0x49	VSCALE	Texture Scale V
0x4A	UOFFSET	Texture Offset U
0x4B	VOFFSET	Texture Offset V
0x4C	OFFSETX	Viewport offset (X)
0x4D	OFFSETY	Viewport offset (Y)
0x4E		
0x4F		
0x50	SHADE	Shade Model
0x51	RNORM	Reverse Face Normals Enable
0x52		
0x53	CMAT	Color Material
0x54	EMC	Emissive Model Color
0x55	AMC	Ambient Model Color
0x56	DMC	Diffuse Model Color
0x57	SMC	Specular Model Color
0x58	AMA	Ambient Model Alpha
0x59		
0x5A		
0x5B	SPOW	Specular Power
0x5C	ALC	Ambient Light Color
0x5D	ALA	Ambient Light Alpha
0x5E	LMODE	Light Model
0x5F	LT0	Light Type 0
0x60	LT1	Light Type 1
0x61	LT2	Light Type 2
0x62	LT3	Light Type 3
0x63	LXP0	Light X Position 0
0x64	LYP0	Light Y Position 0
0x65	LZP0	Light Z Position 0
0x66	LXP1	Light X Position 1
0x67	LYP1	Light Y Position 1
0x68	LZP1	Light Z Position 1
0x69	LXP2	Light X Position 2
0x6A	LYP2	Light Y Position 2
0x6B	LZP2	Light Z Position 2
0x6C	LXP3	Light X Position 3
0x6D	LYP3	Light Y Position 3
0x6E	LZP3	Light Z Position 3
0x6F	LXD0	Light X Direction 0
0x70	LYD0	Light Y Direction 0
0x71	LZD0	Light Z Direction 0
0x72	LXD1	Light X Direction 1
0x73	LYD1	Light Y Direction 1
0x74	LZD1	Light Z Direction 1
0x75	LXD2	Light X Direction 2
0x76	LYD2	Light Y Direction 2

0x77	LZD2	Light Z Direction 2
0x78	LXD3	Light X Direction 3
0x79	LYD3	Light Y Direction 3
0x7A	LZD3	Light Z Direction 3
0x7B	LCA0	Light Constant Attenuation 0
0x7C	LLA0	Light Linear Attenuation 0
0x7D	LQA0	Light Quadratic Attenuation 0
0x7E	LCA1	Light Constant Attenuation 1
0x7F	LLA1	Light Linear Attenuation 1
0x80	LQA1	Light Quadratic Attenuation 1
0x81	LCA2	Light Constant Attenuation 2
0x82	LLA2	Light Linear Attenuation 2
0x83	LQA2	Light Quadratic Attenuation 2
0x84	LCA3	Light Constant Attenuation 3
0x85	LLA3	Light Linear Attenuation 3
0x86	LQA3	Light Quadratic Attenuation 3
0x87	???	Spot light 0 exponent
0x88	???	Spot light 1 exponent
0x89	???	Spot light 2 exponent
0x8A	???	Spot light 3 exponent
0x8B	???	Spot light 0 cutoff
0x8C	???	Spot light 1 cutoff
0x8D	???	Spot light 2 cutoff
0x8E	???	Spot light 3 cutoff
0x8F	ALC0	Ambient Light Color 0
0x90	DLC0	Diffuse Light Color 0
0x91	SLC0	Specular Light Color 0
0x92	ALC1	Ambient Light Color 1
0x93	DLC1	Diffuse Light Color 1
0x94	SLC1	Specular Light Color 1
0x95	ALC2	Ambient Light Color 2
0x96	DLC2	Diffuse Light Color 2
0x97	SLC2	Specular Light Color 2
0x98	ALC3	Ambient Light Color 3
0x99	DLC3	Diffuse Light Color 3
0x9A	SLC3	Specular Light Color 3
0x9B	FFACE	Front Face Culling Order
0x9C	FBP	Frame Buffer Pointer
0x9D	FBW	Frame Buffer Width
0x9E	ZBP	Depth Buffer Pointer
0x9F	ZBW	Depth Buffer Width
0xA0	TBP0	Texture Buffer Pointer 0
0xA1	TBP1	Texture Buffer Pointer 1
0xA2	TBP2	Texture Buffer Pointer 2
0xA3	TBP3	Texture Buffer Pointer 3
0xA4	TBP4	Texture Buffer Pointer 4
0xA5	TBP5	Texture Buffer Pointer 5
0xA6	TBP6	Texture Buffer Pointer 6
0xA7	TBP7	Texture Buffer Pointer 7
0xA8	TBW0	Texture Buffer Width 0
0xA9	TBW1	Texture Buffer Width 1
0xAA	TBW2	Texture Buffer Width 2
0xAB	TBW3	Texture Buffer Width 3
0xAC	TBW4	Texture Buffer Width 4
0xAD	TBW5	Texture Buffer Width 5
0xAE	TBW6	Texture Buffer Width 6
0xAF	TBW7	Texture Buffer Width 7
0xB0	CBP	CLUT Buffer Pointer

0xB1	CBPH	CLUT Buffer Pointer H
0xB2	TRXSBP	Transmission Source Buffer Pointer
0xB3	TRXSBW	Transmission Source Buffer Width
0xB4	TRXDBP	Transmission Destination Buffer Pointer
0xB5	TRXDBW	Transmission Destination Buffer Width
0xB6		
0xB7		
0xB8	TSIZE0	Texture Size Level 0
0xB9	TSIZE1	Texture Size Level 1
0xBA	TSIZE2	Texture Size Level 2
0xBB	TSIZE3	Texture Size Level 3
0xBC	TSIZE4	Texture Size Level 4
0xBD	TSIZE5	Texture Size Level 5
0xBE	TSIZE6	Texture Size Level 6
0xBF	TSIZE7	Texture Size Level 7
0xC0	TMAP	Texture Projection Map Mode + Texture Map Mode
0xC1		Texture Environment Map Matrix
0xC2	TMODE	Texture Mode
0xC3	TPSM	Texture Pixel Storage Mode
0xC4	CLOAD	CLUT Load
0xC5	CMODE	CLUT Mode
0xC6	TFLT	Texture Filter
0xC7	TWRAP	Texture Wrapping
0xC8	TBIAS	Texture Level Bias (???)
0xC9	TFUNC	Texture Function
0xCA	TEC	Texture Environment Color
0xCB	TFLUSH	Texture Flush
0xCC	TSYNC	Texture Sync
0xCD	FFAR	Fog Far (???)
0xCE	FDIST	Fog Range
0xCF	FCOL	Fog Color
0xD0	TSLOPE	Texture Slope
0xD1		
0xD2	PSM	Frame Buffer Pixel Storage Mode
0xD3	CLEAR	Clear Flags
0xD4	SCISSOR1	Scissor Region Start
0xD5	SCISSOR2	Scissor Region End
0xD6	NEARZ	Near Depth Range
0xD7	FARZ	Far Depth Range
0xD8	CTST	Color Test Function
0xD9	CREF	Color Reference
0xDA	CMSK	Color Mask
0xDB	ATST	Alpha Test
0xDC	STST	Stencil Test
0xDD	SOP	Stencil Operations
0xDE	ZTST	Depth Test Function
0xDF	ALPHA	Alpha Blend
0xE0	SFIX	Source Fix Color
0xE1	DFIX	Destination Fix Color
0xE2	DTH0	Dither Matrix Row 0
0xE3	DTH1	Dither Matrix Row 1
0xE4	DTH2	Dither Matrix Row 2
0xE5	DTH3	Dither Matrix Row 3
0xE6	LOP	Logical Operation
0xE7	ZMSK	Depth Mask
0xE8	PMSKC	Pixel Mask Color
0xE9	PMSKA	Pixel Mask Alpha
0xEA	TRXKICK	Transmission Kick

0xEB	TRXSPOS	Transfer Source Position
0xEC	TRXDPOS	Transfer Destination Position
0xED		
0xEE	TRXSIZE	Transfer Size
0xEF		
0xF0		
0xF1		
0xF2		
0xF3		
0xF4		
0xF5		
0xF6		
0xF7		
0xF8		
0xF9		
0xFA		
0xFB		
0xFC		
0xFD		
0xFE		
0xFF		

11.5.1 VADDR

0x01	4	w	VADDR - Vertex List (BASE)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer

11.5.2 IADDR

0x02	4	w	IADDR - Index List (BASE)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer

11.5.3 PRIM

0x04	4	w	PRIM - Primitive Kick

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-18		Primitive type 000 Points 001 Lines 010 Line Strips 011 Triangles 100 Triangle Strips 101 Triangle Fans 110 Sprites (2D Rectangles)
0-15		Number of vertices to kick (0-65535)

11.5.4 BEZIER

0x05	4	w	BEZIER - Bezier Patch Kick

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		V Count
0-7		U Count

11.5.5 SPLINE

0x06	4	w	SPLINE - Spline Surface Kick

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
18-19		V Edges 00 Close/Close 01 Open/Close 10 Close/Open 11 Open/Open
16-17		U Edges 00 Close/Close 01 Open/Close 10 Close/Open 11 Open/Open
8-15		V Count
0-7		U Count

11.5.6 BBOX

0x07	4	w	BBOX - Bounding Box

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-15		Number of vertices to test for conditional rendering (0-65535)

11.5.7 JUMP

0x08	4	w	JUMP - Jump To New Address (BASE)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer

11.5.8 BJUMP

0x09	4	w	BJUMP - Conditional Jump (BASE)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer

11.5.9 CALL

0x0A	4	w	CALL - Call Address (BASE)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer

11.5.10 RET

0x0B	4	w	RET - Return From Call

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

11.5.11 END

0x0C	4	w	END - Stop Execution

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

11.5.12 SIGNAL

0x0E	4	w	SIGNAL - Raise Signal Interrupt

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Signal index to trigger
0-15		Argument to pass to signal handler

11.5.13 FINISH

0x0F	4	w	FINISH - Complete Rendering

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

11.5.14 BASE

0x10	4	w	BASE Base Address Register

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits for address (28 bits total)

11.5.15 VTYPE

0x12	4	w	VTYPE - Vertex Type

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
23		Bypass Transform Pipeline 0 Transformed Coordinates 1 Raw Coordinates
18-20		Number of vertices (Morphing) 000-111: 1-8 vertices
14-16		Number of weights (Skinning) 000-111: 1-8 weights
11-12		Index Format 00 Not using indices 01 8-bit 10 16-bit 11
9-10		Weight Format 00 Not present in vertex 01 8-bit fixed 10 16-bit fixed 11 32-bit floats
7-8		Position Format (3 values XYZ) 00 Not present in vertex 01 8-bit fixed 10 16-bit fixed 11 32-bit floats
5-6		Normal Format (3 values XYZ) 00 Not present in vertex 01 8-bit fixed 10 16-bit fixed 11 32-bit floats
2-4		Color Format (1 value) 000 Not present in vertex 001 010 011 100 16-bit BGR-5650 101 16-bit ABGR-5551 110 16-bit ABGR-4444 111 32-bit ABGR-8888
0-1		Texture Format (2 values ST/UV) 00 Not present in vertex 01 8-bit fixed 10 16-bit fixed 11 32-bit floats

11.5.16 REGION1

0x15	4	w	REGION1 - Draw Region Start

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y Start
0-9		X Start

11.5.17 REGION2

0x16	4	w	REGION2 - Draw Region End

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y End (y + height)-1
0-9		X End (x + width)-1

11.5.18 BOFS

0x2a	4	w	BOFS - Bone Matrix Offset

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Bone Matrix Offset (*)

*) Offset is in values, so each matrix is offset by 3*4 values

11.5.19 BONE

0x2b	4	w	BONE - Bone Matrix Upload

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Matrix Value (GE Float)

Write 3x4 times to upload full bone matrix

11.5.20 MW0

0x2c	4	w	MW0 - Morph Weight 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.21 MW1

0x2d	4	w	MW1 - Morph Weight 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.22 MW2

0x2e	4	w	MW2 - Morph Weight 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.23 MW3

0x2f	4	w	MW3 - Morph Weight 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.24 MW4

0x30	4	w	MW4 - Morph Weight 4

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.25 MW5

0x31	4	w	MW5 - Morph Weight 5

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.26 MW6

0x32	4	w	MW6 - Morph Weight 6

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.27 MW7

0x33	4	w	MW7 - Morph Weight 7

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Morph Value (GE float)

11.5.28 PSUB

0x36	4	w	PSUB - Patch Subdivision

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		T Subdivision
0-7		S Subdivision

11.5.29 PPRIM

0x37	4	w	PPRIM - Patch Primitive

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-1		00 Triangles 01 Lines 10 Points 11

11.5.30 PFACE

0x38	4	w	PFACE - Patch Front Face

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		0 Clockwise 1 Counter-Clockwise

11.5.31 WORLD

0x3b	4	w	WORLD - World Matrix Upload

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Matrix Value (GE Float)

Write 3*4 values for complete matrix

11.5.32 VIEW

0x3d	4	w	VIEW - View Matrix upload

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Matrix Value (GE Float)

Write 3*4 values for complete matrix

11.5.33 PROJ

0x3f	4	w	PROJ - Projection Matrix upload

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Matrix Value (GE Float)

Write 4*4 values for complete matrix

11.5.34 TMA

0x41	4	w	TMATRIX - Texture Matrix Upload

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Matrix Value (GE Float)

Write 3*4 values for complete matrix

11.5.35 XSCALE

0x42	4	w	XSCALE - Viewport Width Scale

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Scale Value (GE Float)

11.5.36 YSCALE

0x43	4	w	YSCALE - Viewport Height Scale

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Scale Value (GE Float)

11.5.37 ZSCALE

0x44	4	w	ZSCALE - Depth Scale

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Scale Value (GE Float)

11.5.38 XPOS

0x45	4	w	XPOS - Viewport X Position

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Offset Value (GE Float)

11.5.39 YPOS

0x46	4	w	YPOS - Viewport Y Position

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Offset Value (GE Float)

11.5.40 ZPOS

0x47	4	w	ZPOS - Depth Position

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Offset Value (GE Float)

11.5.41 USCALE

0x48	4	w	USCALE - Texture Scale U

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Scale Value (GE Float)

11.5.42 VSCALE

0x49	4	w	VSCALE - Texture Scale V

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Scale Value (GE Float)

11.5.43 UOFFSET

0x4a	4	w	UOFFSET - Texture Offset U

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Offset Value (GE Float)

11.5.44 VOFFSET

0x4b	4	w	VOFFSET - Texture Offset V

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Offset Value (GE Float)

11.5.45 OFFSETX

0x4c	4	w	OFFSETX - Viewport offset (X)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		X Offset (12.4 fixed)

11.5.46 OFFSETY

0x4d	4	w	OFFSETY - Viewport offset (Y)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Y Offset (12.4 fixed)

11.5.47 SHADE

0x50	4	w	SHADE - Shade Model

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		Shading type 0 Flat 1 Smooth

11.5.48 CMAT

0x53	4	w	CMAT - Color Material

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-2		Material flags (OR together) 000 001 Ambient 010 Diffuse 011 100 Specular 101 110 111

11.5.49 EMC

0x54	4	w	EMC - Emissive Model Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.50 AMC

0x55	4	w	AMC - Ambient Model Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.51 DMC

0x56	4	w	DMC - Diffuse Model Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.52 SMC

0x57	4	w	SMC - Specular Model Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.53 AMA

0x58	4	w	AMA - Ambient Model Alpha

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7		Alpha Component

11.5.54 SPOW

0x5b	4	w	SPOW - Specular Power

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Power (GE Float)

11.5.55 ALC

0x5c	4	w	ALC - Ambient Light Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.56 ALA

0x5d	4	w	ALA - Ambient Light Alpha

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7		Alpha Component

11.5.57 LMODE

0x5e	4	w	LMODE - Light Model

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		Lighting model 0 Single color 1 Separate specular color

11.5.58 LT0

0x5f	4	w	LT0 Light Type 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		Light Type 00 Directional Light 01 Point Light 10 Spot Light 11
0-1		Light Components 00 Ambient & Diffuse 01 Diffuse & Specular 10 Unknown (diffuse color, affected by specular power) 11

11.5.59 LT1

0x60	4	w	LT1 Light Type 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		Light Type 00 Directional Light 01 Point Light 10 Spot Light 11
0-1		Light Components 00 Ambient & Diffuse 01 Diffuse & Specular 10 Unknown (diffuse color, affected by specular power) 11

11.5.60 LT2

0x61	4	w	LT2 Light Type 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		Light Type 00 Directional Light 01 Point Light 10 Spot Light 11
0-1		Light Components 00 Ambient & Diffuse 01 Diffuse & Specular 10 Unknown (diffuse color, affected by specular power) 11

11.5.61 LT3

0x62	4	w	LT3 Light Type 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		Light Type 00 Directional Light 01 Point Light 10 Spot Light 11
0-1		Light Components 00 Ambient & Diffuse 01 Diffuse & Specular 10 Unknown (diffuse color, affected by specular power) 11

11.5.62 LXP0

0x63	4	w	LXP0 - Light X Position 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.63 LYP0

0x64	4	w	LYP0 - Light Y Position 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.64 LZP0

0x65	4	w	LZP0 - Light Z Position 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.65 LXP1

0x66	4	w	LXP1 - Light X Position 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.66 LYP1

0x67	4	w	LYP1 - Light Y Position 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.67 LZP1

0x68	4	w	LZP1 - Light Z Position 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.68 LXP2

0x69	4	w	LXP2 - Light X Position 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.69 LYP2

0x6a	4	w	LYP2 - Light Y Position 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.70 LZP2

0x6b	4	w	LZP2 - Light Z Position 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.71 LXP3

0x6c	4	w	LXP3 - Light X Position 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.72 LYP3

0x6d	4	w	LYP3 - Light Y Position 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.73 LZP3

0x6e	4	w	LZP3 - Light Z Position 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.74 LXD0

0x6f	4	w	LXD0 - Light X Direction 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.75 LYD0

0x70	4	w	LYD0 - Light Y Direction 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.76 LZD0

0x71	4	w	LZD0 - Light Z Direction 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.77 LXD1

0x72	4	w	LXD1 - Light X Direction 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.78 LYD1

0x73	4	w	LYD1 - Light Y Direction 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.79 LZD1

0x74	4	w	LZD1 - Light Z Direction 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.80 LXD2

0x75	4	w	LXD2 Light X Direction 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.81 LYD2

0x76	4	w	LYD2 - Light Y Direction 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.82 LZD2

0x77	4	w	LZD2 - Light Z Direction 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.83 LXD3

0x78	4	w	LXD3 - Light X Direction 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.84 LYD3

0x79	4	w	LYD3 - Light Y Direction 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.85 LZD3

0x7a	4	w	LZD3 - Light Z Direction 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Vector Component (GE Float)

11.5.86 LCA0

0x7b	4	w	LCA0 - Light Constant Attenuation 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.87 LLA0

0x7c	4	w	LLA0 - Light Linear Attenuation 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.88 LQA0

0x7d	4	w	LQA0 - Light Quadratic Attenuation 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.89 LCA1

0x7e	4	w	LCA1 - Light Constant Attenuation 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.90 LLA1

0x7f	4	w	LLA1 - Light Linear Attenuation 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.91 LQA1

0x80	4	w	LQA1 - Light Quadratic Attenuation 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.92 LCA2

0x81	4	w	LCA2 - Light Constant Attenuation 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.93 LLA2

0x82	4	w	LLA2 - Light Linear Attenuation 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.94 LQA2

0x83	4	w	LQA2 - Light Quadratic Attenuation 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.95 LCA3

0x84	4	w	LCA3 - Light Constant Attenuation 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.96 LLA3

0x85	4	w	LLA3 - Light Linear Attenuation 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.97 LQA3

0x86	4	w	LQA3 - Light Quadratic Attenuation 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Attenuation Factor (GE Float)

11.5.98 ???

0x87	4	w	??? Spot light 0 exponent

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight exponent

11.5.99 ???

0x88	4	w	??? Spot light 1 exponent

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight exponent

11.5.100 ???

0x89	4	w	??? Spot light 2 exponent

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight exponent

11.5.101 ???

0x8a	4	w	??? Spot light 3 exponent

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight exponent

11.5.102 ???

0x8b	4	w	??? Spot light 0 cutoff

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight cutoff angle (cosine of angle)

11.5.103 ???

0x8c	4	w	??? Spot light 1 cutoff

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight cutoff angle (cosine of angle)

11.5.104 ???

0x8d	4	w	??? Spot light 2 cutoff

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight cutoff angle (cosine of angle)

11.5.105 ???

0x8e	4	w	??? Spot light 3 cutoff

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Spotlight cutoff angle (cosine of angle)

11.5.106 ALC0

0x8f	4	w	ALC0 - Ambient Light Color 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.107 DLC0

0x90	4	w	DLC0 - Diffuse Light Color 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.108 SLC0

0x91	4	w	SLC0 - Specular Light Color 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.109 ALC1

0x92	4	w	ALC1 - Ambient Light Color 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.110 DLC1

0x93	4	w	DLC1 - Diffuse Light Color 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.111 SLC1

0x94	4	w	SLC1 - Specular Light Color 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.112 ALC2

0x95	4	w	ALC2 - Ambient Light Color 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.113 DLC2

0x96	4	w	DLC2 - Diffuse Light Color 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.114 SLC2

0x97	4	w	SLC2 - Specular Light Color 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.115 ALC3

0x98	4	w	ALC3 - Ambient Light Color 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.116 DLC3

0x99	4	w	DLC3 - Diffuse Light Color 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.117 SLC3

0x9a	4	w	SLC3 - Specular Light Color 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.118 FFACE

0x9b	4	w	FFACE - Front Face Culling Order

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		Culling Order 0 Clockwise primitives are visible 1 Counter-clockwise primitives are visible

11.5.119 FBP

0x9c	4	w	FBP - Frame Buffer Pointer

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see FBW)

11.5.120 FBW

0x9d	4	w	FBW - Frame Buffer Width

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		8 most significant bits of pointer (see FBP)
0-15		Buffer width in pixels

11.5.121 ZBP

0x9e	4	w	ZBP - Depth Buffer Pointer

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see ZBW)

11.5.122 ZBW

0x9f	4	w	ZBW - Depth Buffer Width

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		8 most significant bits of pointer (see ZBP)
0-15		Buffer width in pixels

11.5.123 TBP0

0xa0	4	w	TBP0 - Texture Buffer Pointer 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW0)

11.5.124 TBP1

0xa1	4	w	TBP1 - Texture Buffer Pointer 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW1)

11.5.125 TBP2

0xa2	4	w	TBP2 - Texture Buffer Pointer 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW2)

11.5.126 TBP3

0xa3	4	w	TBP3 - Texture Buffer Pointer 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW3)

11.5.127 TBP4

0xa4	4	w	TBP4 - Texture Buffer Pointer 4

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW4)

11.5.128 TBP5

0xa5	4	w	TBP5 - Texture Buffer Pointer 5

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW5)

11.5.129 TBP6

0xa6	4	w	TBP6 - Texture Buffer Pointer 6

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW6)

11.5.130 TBP7

0xa7	4	w	TBP7 - Texture Buffer Pointer 7

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TBW7)

11.5.131 TBW0

0xa8	4	w	TBW0 - Texture Buffer Width 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP0)
0-15		Buffer width in pixels

11.5.132 TBW1

0xa9	4	w	TBW1 - Texture Buffer Width 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP1)
0-15		Buffer width in pixels

11.5.133 TBW2

0xaa	4	w	TBW2 - Texture Buffer Width 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP2)
0-15		Buffer width in pixels

11.5.134 TBW3

0xab	4	w	TBW3 - Texture Buffer Width 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP3)
0-15		Buffer width in pixels

11.5.135 TBW4

0xac	4	w	TBW4 - Texture Buffer Width 4

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP4)
0-15		Buffer width in pixels

11.5.136 TBW5

0xad	4	w	TBW5 - Texture Buffer Width 5

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP5)
0-15		Buffer width in pixels

11.5.137 TBW6

0xae	4	w	TBW6 - Texture Buffer Width 6

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP6)
0-15		Buffer width in pixels

11.5.138 TBW7

0xaf	4	w	TBW7 - Texture Buffer Width 7

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see TBP7)
0-15		Buffer width in pixels

11.5.139 CBP

0xb0	4	w	CBP - CLUT Buffer Pointer

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see CBPH)

11.5.140 CBPH

0xb1	4	w	CBPH - CLUT Buffer Pointer H

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		4 most significant bits of pointer (see CBP)

11.5.141 TRXSBP

0xb2	4	w	TRXSBP - Transmission Source Buffer Pointer

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TRXSBW)

11.5.142 TRXSBW

0xb3	4	w	TRXSBW - Transmission Source Buffer Width

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		8 most significant bits of pointer (see TRXSBP)
0-15		Source Buffer Width

11.5.143 TRXDBP

0xb4	4	w	TRXDBP - Transmission Destination Buffer Pointer

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		24 least significant bits of pointer (see TRXDBW)

11.5.144 TRXDBW

0xb5	4	w	TRXDBW - Transmission Destination Buffer Width

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		8 most significant bits of pointer (see TRXDBP)
0-15		Destination Buffer Width

11.5.145 TSIZE0

0xb8	4	w	TSIZE0 - Texture Size Level 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.146 TSIZE1

0xb9	4	w	TSIZE1 - Texture Size Level1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.147 TSIZE2

0xba	4	w	TSIZE2 - Texture Size Level 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.148 TSIZE3

0xbb	4	w	TSIZE3 - Texture Size Level 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.149 TSIZE4

0xbc	4	w	TSIZE4 - Texture Size Level 4

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.150 TSIZE5

0xbd	4	w	TSIZE5 - Texture Size Level 5

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.151 TSIZE6

0xbe	4	w	TSIZE6 - Texture Size Level 6

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.152 TSIZE7

0xbf	4	w	TSIZE7 - Texture Size Level 7

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-15		Height = 2^TH
0-7		Width = 2^TW

11.5.153 TMAP

0xc0	4	w	TMAP - Texture Projection Map Mode + Texture Map Mode

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		Texture Projection Map Mode 00 Position 01 Texture Coordinates 10 Normalized Normal 11 Normal
0-1		Texture Map Mode 00 Texture Coordinates (UV) 01 Texture Matrix 10 Environment Map 11

11.5.154 ???

0xc1	4	w	??? Texture Environment Map Matrix

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-9		2nd column for matrix
0-1		1st Column for matrix

11.5.155 TMODE

0xc2	4	w	TMODE - Texture Mode

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-20		Maximum mipmap level
8-15		???
0		Swizzle Enable

11.5.156 TPSM

0xc3	4	w	TPSM - Texture Pixel Storage Mode

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Pixel Storage Mode 0 16-bit BGR 5650 1 16-bit ABGR 5551 2 16-bit ABGR 4444 3 32-bit ABGR 8888 4 4-bit indexed 5 8-bit indexed 6 16-bit indexed 7 32-bit indexed 8 DXT1 9 DXT3 10 DXT5

11.5.157 CLOAD

0xc4	4	w	CLOAD - CLUT Load

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Number of colors divided by 8

11.5.158 CMODE

0xc5	4	w	CMODE - CLUT Mode

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		???
8-15		mask
2-7		???
0-1		CLUT Pixel Format 00 16-bit BGR 5650 01 16-bit ABGR 5551 10 16-bit ABGR 4444 11 32-bit ABGR 8888

11.5.159 TFLT

0xc6	4	w	TFLT - Texture Filter

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-10		Magnifying filter 000 Nearest 001 Linear 010 011 100 Nearest; Mipmap Nearest 101 Linear; Mipmap Nearest 110 Nearest; Mipmap Linear 111 Linear; Mipmap Linear
0-2		Minifying filter

11.5.160 TWRAP

0xc7	4	w	TWRAP - Texture Wrapping

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8		V Wrap Mode 0 Repeat 1 Clamp
0		U Wrap Mode

11.5.161 TBIAS

0xc8	4	w	TBIAS - Texture Level Bias (???)

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Mipmap bias (signed)
0-15		???

11.5.162 TFUNC

0xc9	4	w	TFUNC - Texture Function

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16		Fragment Double Enable 0 Fragment color is untouched 1 Fragment color is doubled
8		Texture Color Component 0 Texture alpha is ignored 1 Texture alpha is read
0-2:		Texture Effect 000 Modulate 001 Decal 010 Blend 011 Replace 100 Add 101 110 111

11.5.163 TEC

0xca	4	w	TEC -.Texture Environment Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.164 TFLUSH

0xcb	4	w	TFLUSH - Texture Flush

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

Invalidate texture cache on texture change

11.5.165 TSYNC

0xcc	4	w	TSYNC - Texture Sync

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description

Sync with texture transfer (see TRXKICK)

11.5.166 FDIST

0xce	4	w	FDIST - Fog Range

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Range (GE Float)

11.5.167 FCOL

0xcf	4	w	FCOL - Fog Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.168 TSLOPE

0xd0	4	w	TSLOPE - Texture Slope

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Slope (GE Float)

11.5.169 PSM

0xd2	4	w	PSM - Frame Buffer Pixel Storage Mode

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-1		Pixel Storage Mode 00 16-bit BGR 5650 01 16-bit ABGR 5551 10 16-bit ABGR 4444 11 32-bit ABGR 8888

11.5.170 CLEAR

0xd3	4	w	CLEAR - Clear Flags

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-11		Clear flags (OR together) 000 001 Clear Color Buffer 010 Clear Stencil/Alpha Buffer 011 100 Clear Depth Buffer 101 110 111
0		Clear enable

11.5.171 SCISSOR1

0xd4	4	w	SCISSOR1 - Scissor Region Start

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y Start
0-9		X Start

11.5.172 SCISSOR2

0xd5	4	w	SCISSOR2 - Scissor Region End

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y End
0-9		X End

11.5.173 NEARZ

0xd6	4	w	NEARZ - Near Depth Range

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-15		Depth Value

11.5.174 FARZ

0xd7	4	w	FARZ - Far Depth Range

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-15		Depth Value

11.5.175 CTST

0xd8	4	w	CTST - Color Test Function

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-1		Color Function

00	Never pass pixel
01	Always pass pixel
10	Pass pixel if color matches
11	Pass pixel if color differs

11.5.176 CREF

0xd9	4	w	CREF - Color Reference

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Color Reference Value

11.5.177 CMSK

0xda	4	w	CMSK - Color Mask

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-23		Color Mask

11.5.178 ATST

0xdb	4	w	ATST - Alpha Test

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Alpha Mask
8-15		Alpha Reference Value
0-2		Alpha Test Function

000	Never pass pixel
001	Always pass pixel
010	Pass pixel if match
011	Pass pixel if difference
100	Pass pixel if less
101	Pass pixel if less or equal
110	Pass pixel if greater
111	Pass pixel if greater or equal

11.5.179 STST

0xdc	4	w	STST - Stencil Test

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Stencil Mask
8-15		Stencil Reference Value
0-2		Stencil Function

000	Never pass stencil test
001	Always pass stencil test
010	Pass test if match
011	Pass test if difference
100	Pass test if less
101	Pass test if less or equal
110	Pass test if greater
111	Pass test if greater or equal

11.5.180 SOP

0xdd	4	w	SOP - Stencil Operations

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-18		Zfail Op

000	Keep stencil value
001	Zero stencil value
010	Replace stencil value
011	Invert stencil value
100	Increment stencil value
101	Decrement stencil value
110	
111	

8-11		Fail Op
0-3		Pass Op

11.5.181 ZTST

0xde	4	w	ZTST - Depth Test Function

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-2		Function

000	Never pass pixel
001	Always pass pixel
010	Pass pixel when depth is equal
011	Pass pixel when depth is not equal
100	Pass pixel when depth is less
101	Pass pixel when depth is less or equal
110	Pass pixel when depth is greater
111	Pass pixel when depth is greater or equal

11.5.182 ALPHA

0xdf	4	w	ALPHA - Alpha Blend

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
8-11		Destination Function

0000	Source Color
0001	One Minus Source Color
0010	Source Alpha
0011	One Minus Source Alpha
0100	Destination Color
0101	One Minus Destination Color
0110	Destination Alpha
0111	One Minus Destination Alpha
1000	Fix
1001	
1010	
1011	
1100	
1101	
1110	
1111	

4-7		Source Function
0-3		Blend Operation

000	Add
001	Subtract
010	Reverse Subtract
011	Minimum Value
100	Maximum Value
101	Absolute Value
110	
111	

11.5.183 SFIX

0xe0	4	w	SFIX - Source Fix Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.184 DFIX

0xe1	4	w	DFIX - Destination Fix Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Component
8-15		Green Component
0-7		Red Component

11.5.185 DTH0

0xe2	4	w	DTH0 - Dither Matrix Row 0

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
12-15		Column 3
8-11		Column 2
4-7		Column 1
0-3		Column 0

11.5.186 DTH1

0xe3	4	w	DTH1 - Dither Matrix Row 1

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
12-15		Column 3
8-11		Column 2
4-7		Column 1
0-3		Column 0

11.5.187 DTH2

0xe4	4	w	DTH2 - Dither Matrix Row 2

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
12-15		Column 3
8-11		Column 2
4-7		Column 1
0-3		Column 0

11.5.188 DTH3

0xe5	4	w	DTH3 - Dither Matrix Row 3

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
12-15		Column 3
8-11		Column 2
4-7		Column 1
0-3		Column 0

11.5.189 LOP

0xe6	4	w	LOP - Logical Operation

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-3		Logic Op

0000	Clear
0001	And
0010	Reverse And
0011	Copy
0100	Inverted And
0101	No Operation
0110	Exclusive Or
0111	Or
1000	Negated Or
1001	Equivalence
1010	Inverted
1011	Reverse Or
1100	Inverted Copy
1101	Inverted Or
1110	Negated And
1111	Set

11.5.190 ZMSK

0xe7	4	w	ZMSK - Depth Mask

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-15		Depth Write Mask

11.5.191 PMSKC

0xe8	4	w	PMSKC - Pixel Mask Color

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
16-23		Blue Write Mask
8-15		Green Write Mask
0-7		Red Write Mask

11.5.192 PMSKA

0xe9	4	w	PMSKA - Pixel Mask Alpha

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0-7		Alpha Write Mask

11.5.193 TRXKICK

0xea	4	w	TRXKICK - Transmission Kick

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
0		0 16-bit texel size 1 32-bit texel size

11.5.194 TRXSPOS

0xeb	4	w	TRXSPOS - Transfer Source Position

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y Position
0-9		X Position

11.5.195 TRXDPOS

0xec	4	w	TRXDPOS - Transfer Destination Position

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Y Position
0-9		X Position

11.5.196 TRXSIZE

0xee	4	w	TRXSIZE - Transfer Size

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
10-19		Height = Transfer Height-1
0-9		Width = Transfer Width-1

11.6 Texture Cache

The texture cache is very important on the PSP (as it was on the PS2). From experiments it seems to be 8kB, so that means it's 64x32 in 32-bit, 64x64 in 16-bit, 128x64 in 8-bit and 128x128 in 4-bit (the sizes are qualified guesses by looking at the PS2). Ordering your draws so that locality in uv-coordinates is maximized will make sure your rendering is optimal.

DXTn is decompressed into 32-bit when loaded into the cache, so what you gain in shrinking the texture-size, you lose in texture-cache. If you can, use 4- or 8-bit textures, which will allow a much larger area to be kept in the cache.

11.7 Memory Bandwidth

texture reads from user memory (mem range 0x08800000 - 0x01800000) have a bandwidth of 50MB/s

texture reads from GE memory or VRAM (mem range 0x04000000 - 0x00200000) have a bandwidth of 500MB/s

if you have a texture in user memory it is possible to load that texture to VRAM at a bandwidth of 150MB/s

12 Audio Processing

12.1 Overview

▷ 44100 Hz Sample Frequency

13 Infrared Port

The PSP comes with support for IRDA and Sony’s "SIRCS" protocol (useful for Sony devices only)

14 WLAN

15 USB Port

16 UMD

17 Memory Stick

18 Headphone/Remote Control

18.1 Audio Input

18.2 Serial Communications

The PSP communicates with the microcontroller inside the remote control using RS232 serial communication (although the voltages are different of course, 0V and +2.5V) using 8N1 framing at 4800bps. The protocol consists of command packages which can be send by either the PSP or the remote control. A package is exchanged as follows:

Sender	Reciever	description
0xF0		Request to transmit
	0xF8	Clearance to transmit
0xFD		Packet starts
cmd		Command code + phase
params		Zero or more bytes of parameter data
checksum		XOR of the cmd and params bytes
0xFE		Packet ends
	0xFA/0xFB	Packet received correctly

If the packet is not received correctly, or the receiver is too busy to allow the packet to be transmitted, the corresponding 0xFA/0xFB/0xF8 is not sent, in which case the sender should wait a while (60 ms) and then try again from the 0xF0. If no answer is received in a long time (> 1s), a BREAK can be sent to reset the communication channel, after which the state should be the same as if the remote control had been disconnected and reconnected again.

The least significant bit of the cmd byte is the phase indicator, which is used to differentiate a new command from the retransmission of an old one. The first packet sent from a particular device has phase 0 (LSB = 0), and is acknowledged with 0xFA. Then the phase is inverted each time a new packets is sent. Packets with phase 1 are acknowledged with 0xFB. Phase is not shared, so when the PSP sends a packet it does not affect the phase of the remote control, and vice versa.

Note that there seems to be no particular way to know how many parameter bytes are contained in the message, as the parameter bytes or the checksum could contain an 0xFE as well. It is therefore necessary to know how many parameter bytes each command takes.

The command sent by the remote control to inform the PSP of what buttons are pressed is 0x84. It takes two parameter bytes, which if interpreted as a 16-bit integer (little endian) forms a bitfield like so:

bit	value	button
0	`0x0001`	Play/Pause
1	`0x0002`	? (unused)
2	`0x0004`	Fast Forward
3	`0x0008`	Rewind
4	`0x0010`	Vol +
5	`0x0020`	Vol -
6	`0x0040`	? (unused)
7	`0x0080`	Hold

Buttons that are pressed have their corresponding bits set to 1. Buttons that are not pressed or do not exist have their corresponding bits set to 0.

The 0x80 command has some parameter bytes, and I'm guessing these are used to identify the type of device connected. There could also be any number (well, a bit over 100 at least) of commands to request specific kinds of services from the PSP.

19 Flash Memory

19.1 Physical Layout

The PSP MCP uses a 32MB NAND with the following layout:

- ▷ 512+16 bytes per page
- ▷ 32 pages per block (16K+512)
- ▷ 2048 blocks per device (32MB+1MB)

A block is the smallest erasable unit, a page the smallest writable (programmable). A Block holds 32 pages (for the latest small page NAND devices, including the MCP used for the PSP).

19.2 User Area (Main Data)

The IPL doesn't seem to be part of any kind of FS (blocks appear at fixed physical locations). Everything else (above 1MiB phys) is FAT12 with a SmartMedia style Block mapping but with a custom mapping area (i.e. different layout from what is/was mandated for SM).

Only FAT organized area of on-board flash chip, system file volume and configuration file volume, can be accessed via FAT Filesystem. The bootstrap area is unreachable by the flash and lflash drivers. (lflash returns all 0x00)

19.2.1 Physical Layout (unmapped)

start	end	size	description
0x00000000	0x000FFFFF	1MB	bootstrap Area
0x00100000	0x01ffffff	31MB	mapped Area

19.2.2 Logical Layout (mapped)

When the Flashdriver starts up it reads all the extra data sections (usually from the first page of each block). From this data it extracts the logical block number which in turn is used to build up a table (index is LBN, value is PBN). Reading from logical Blocks works by simple address translation (LBN->PBN). Writing is usually done using a write before erase strategy, i.e. an emtpy block is filled with the data (new/replacement), then the LBN entry is remapped to the new PBN and the old physical block is erased (and goes either back to the free pool are becomes a bad block).

start		end		size	description
Offset	Block	Offset	Block		
0x00000000	0x000				Master Boot Record (MBR)
0x00008000	0x002				Partition Boot Record (PBR)
0x0000c000	0x003			24MiB	FAT12 Partition #1 (flash0)
0x01808000	0x602				
0x0180C000	0x603			4MiB	FAT12 Partition #2 (flash1)
0x01C08000	0x702				
0x01C0C000	0x703				FAT12 Partition #3 (empty)
0x01D08000	0x742				
0x01D0C000	0x743				FAT12 Partition #4 (empty)
0x01DF8000	0x77e				
0x01DFC000	0x77f				Last Block

19.2.3 Bootstrap (IPL Area)

The IPL, region and serial number are located within the nand non-fat area (using an ecrypted form)

start	end	size	block	description
0x00000000		64k	1-3	? (all 0xff)
0x00010000	0x00013fff	16k	4	physical block numbers of IPL
0x00014000			5	block numbers of IPL (duplicated)
0x00018000			6	block numbers of IPL (duplicated)
0x0001c000			7	block numbers of IPL (duplicated)
0x00020000			8	block numbers of IPL (duplicated)
0x00024000			9	block numbers of IPL (duplicated)
0x00028000			10	block numbers of IPL (duplicated)
0x0002c000			11	block numbers of IPL (duplicated)
0x00030000	0x0003ffff			all 0xff
0x00040000		16k	16-29	encrypted IPL (encrypted chunks of 0x1000 bytes each)
	0x000bffff			rest 0xff (max 0x20 blocks free for IPL)
0x000c0000				ID Storage Area
0x000d4180	0x000FFFFF			rest 0xff

19.2.3.1 IPL Block Mapping Physical blocks 4-11 hold mapping information. Each block contains the same information, for redundancy presumably. If one of these blocks becomes invalid, the next one is used etc. If all these blocks are bad the PSP might be dead

19.2.4 ID Storage Area

Various subsystems in the PSP make use of the id-storage including usb, wlan, umd, etc. (The firmware provides a driver in idstorage.prx to facilitate manipulations.)

The id-storage area begins at 0xc0000 and appears to be used to store low-level information. The id-storage area is an associative array and information is stored using key/value pairs. The id-storage seems a little coupled to the the physical storage as each key maps to an area of 512-bytes, which is equal to the pagesize of the PSP standard nand-flash, and it seems 512-byte page operations are intended.

key 0x100-0x11F same as key 0x120-0x13F

old ver psp haven't key 0x046, 0x047

old old ver psp haven't key 0x140

19.2.4.1 Index The keys are stored in an index which consists of two nand pages of 512 bytes. The index is identified by byte 6 of the spare area being 0x73. Byte 7 might be the id-storage version number. Byte 8 must be 1 (or possibly 0) and might indicate whether the storage is formatted or not, and a value greater than 1 in byte 9 indicates that the id-storage is read-only.

Keys are 16-bit integers. The location of the data associated with a key is identified by the key's position in the index. For instance, a key appearing at position 97 (byte 194) in the index will find its associated data at location: 0xc0000 + (97 * 512) = 0xcc200.

19.2.4.2 key 0x041 : USB Descriptor

offset	description	
0x0000	idVendor	4C 05
0x0002		00 00
0x0004	bLength	0A
0x0005		03
0x0006	iManufacturer String	'S.o.n.y.'
0x0044	? bNum	05
0x0045		00 00 00
0x0048	idProduct	C8 01
0x004A		00 00
0x004C	bLength	16
0x004D	? bDescriptorType	03
0x004E	iProduct String	'P.S.P. .T.y.p.e. .A.'
0x008C	idProduct	C9 01
0x008E		00 00
0x0090	bLength	16
0x0091	? bDescriptorType	03
0x0092	iProduct String	'P.S.P. .T.y.p.e. .B.'
0x00D0	idProduct	CA 01
0x00D2		00 00
0x00D4	bLength	16
0x00D5	? bDescriptorType	03
0x00D6	iProduct String	'P.S.P. .T.y.p.e. .C.'
0x0114	idProduct	CB 01
0x0116		00 00
0x0118	bLength	16
0x0119	? bDescriptorType	03
0x011A	iProduct String	'P.S.P. .T.y.p.e. .D.'
0x0158	idProduct	CC 01
0x015A		00 00
0x015C	bLength	16
0x015D	? bDescriptorType	03
0x015E	iProduct String	'P.S.P. .T.y.p.e. .E.'

19.2.4.3 key 0x044 : MAC Address

19.2.4.4 key 0x050 : Serial Number

19.2.5 FAT Area

FAT12 with a cluster size of 16K which conveniently matches the erase block size.

19.3 Spare Area (extra Data)

start	end	size	description	
0x00		4	user_ecc	calculated per 512 byte page of user data (byte 3 is always 0x00)
0x04		1	block_fmt	0xff = IPL, 0x00 = FAT
0x05		1	block_stat	0xff = valid block
0x06		2	block_addr	logical block number for FAT, mostly 0xff 0xff for IPL, 0x73 0x01 = ID-Storage Index
0x08		2	?	ID-Storage Index =0x01 0x01 / IPL = 0x38 0x4a or 0x01 0x01 / others 0x00 0x00
0x0a		2	?	ID-Storage Index =0xff 0xff / IPL = 0xc6 0x6d or 0xff 0xff / others 0x00 0x00
0x0c		2	spare_ecc	calculated from bytes 0x04-0x0b of spare area (12 bit, high nybble always 0xf)
0x0e		2	?	always 0xff 0xff

note: If reading a dump from a live PSP, it is important to verify the ECC. Hardware automatically reclaims single-bit errors in the user-area, but for the spare area this must be done manually.

19.4 Tools

▷ `dumpipl` (MrBrown, Tyranid, John Kelley) dump IPL from Flash to Memstick [runs on PSP]

20 Flash Memory Structure (flash0)

```
/DATA
  /CERT
/DIC
/FONT
/KD
  /RESOURCE
/VSH
  /ETC
  /MODULE
  /RESOURCE
```

20.1 DATA Subdirectory

20.1.1 CERT Subdirectory

Contains lots of certificates. They are ordinal base64 encoded certificate, not encrypted.

Filename	Size			
Class1_PCA_G2_v2.cer	1122	SHA1/RSA1024	VeriSign	*1
Class1_PCA_G3v2.cer	1508	SHA1/RSA2048	VeriSign	*1
Class1_PCA_ss_v4.cer	854	MD2 /RSA1024	VeriSign	*1
Class2_PCA_G2_v2.cer	1126	SHA1/RSA1024	VeriSign	*1
Class2_PCA_G3v2.cer	1504	SHA1/RSA2048	VeriSign	*1
Class2_PCA_ss_v4.cer	848	MD2 /RSA1024	VeriSign	*1
Class3_PCA_G2_v2.cer	1122	SHA1/RSA1024	VeriSign	*1
Class3_PCA_G3v2.cer	1508	SHA1/RSA2048	VeriSign	*1
Class3_PCA_ss_v4.cer	848	MD2 /RSA1024	VeriSign	*1
Class4_PCA_G2_v2.cer	1122	SHA1/RSA1024	VeriSign	*1
Class4_PCA_G3v2.cer	1508	SHA1/RSA2048	VeriSign	*1
RSA1024_v1.cer	1066	SHA1/RSA1024	ValiCert	*2
RSA2048_v3.cer	1233	SHA1/RSA2048	RSA Security	*2
RSA_SecureServer.cer	840	MD2 /RSA1024	RSA Data Security	*2
SCE_CA01.cer	1387	SHA1/RSA2048	SCEI	*3
SCE_CA02.cer	1387	SHA1/RSA2048	SCEI	*3
SCE_CA03.cer	1387	SHA1/RSA2048	SCEI	*3
SCE_CA04.cer	1387	SHA1/RSA2048	SCEI	*3
SCE_CA05.cer	1387	SHA1/RSA2048	SCEI	*3
VeriSign_TSA_CA.cer	1402	SHA1/RSA1024	VeriSign, Time Stamping Authority	*4

1) These are relating to 'Primary Certificate Authority' certificates from VeriSign. They have specific groups that monitor and certify Certificate Authorities, providing direct trust to CA certificates. These form the root of the trust network for signed code. Pretty much every Windows machine has these for use in Internet Explorer and the like.

2) These are related to the BSAFE technology RSA Security provides. They are likely used for the wireless communications, as BSAFE has wireless security software packages aimed at systems like ARM for things like SSL over WiFi (sound familiar?). I don't know if they are linked through Verisign's PCAs or form their own root. It would make more sense if they were signed by either Verisign's PCAs or by one of Sony's CAs.

3) A series of certificates in Sony's control, very likely signed by the PCA certificates mentioned above. These are probably used to sign code certificates for developers, and those certificates are included with the games themselves. So code signatures are done by the developer, while encryption is done by Sony. The trust can still be verified by checking the signed game certificate, seeing that it belongs to SCE_CA0x, and then seeing /that/ belongs to Verisign, which is the root trust node.

4)Says exactly what it is on the tin, used to time-stamp things in such a way that it cannot be spoofed. (i.e, Verisign encrypts the time stamp of a signing with their private key, allowing everyone to verify the time stamp, but nobody can make a different time stamp that can be verified correctly without VeriSign's key)

This as a whole is a trust tree, to setup a base list of trusted certificates for the PSP. Anything signed directly by the owners of these certificates, or using a key which has been signed by the owners of these certificates will be trusted. (I.E. can the certificate presented by the game/software to be run be verified as to be connected to these certificates?)

20.2 DIC Subdirectory

Filename	Size
apotp.dic	1346880
atokp.dic	939166
aux0.dic	14886
aux1.dic	9647
aux2.dic	4631
aux3.dic	13172

20.3 FONT Subdirectory

contains various Fonts used by the PSP OS

Filename	Size
jpn0.pgf	1679100
ltn0.pgf	123896
ltn1.pgf	113200
ltn10.pgf	58256
ltn11.pgf	55924
ltn12.pgf	61816
ltn13.pgf	58788
ltn14.pgf	64100
ltn15.pgf	59924
ltn2.pgf	129652
ltn3.pgf	115940
ltn4.pgf	132536
ltn5.pgf	121548
ltn6.pgf	138472
ltn7.pgf	124868
ltn8.pgf	56512
ltn9.pgf	54484

20.4 KD Subdirectory

20.4.1 Kernel Modules

Module Filename	API-Module	Format	v1.0 size	version	v1.5 size	version
ata.prx	sceATA_ATAPI_driver	~PSP	13232			1.2
audio.prx	sceAudio_Driver	~PSP	9040			1.2
audiocodec.prx	sceAudiocodec_Driver	~PSP	3248	1.1		1.1
blkdev.prx	sceBLK_driver	~PSP	3712	1.1		1.1
chkreg.prx	sceChkreg	~PSP	3488			1.2
clockgen.prx	sceClockgen_Driver	~PSP	2416	1.1		1.1
codec.prx	sceWM8750_Driver	~PSP	4096			1.2
ctrl.prx	sceController_Service	~PSP	5600			1.2
display.prx	sceDisplay_Service	~PSP	7248			1.2
dmacman.prx	sceDMAManager	~PSP	6032			1.2
dmacplus.prx	sceDMACPLUS_Driver	~PSP	8768			1.2
emc_ddr.prx	sceDDR_Driver	~PSP	2384	1.1		1.1
emc_sm.prx	sceNAND_Driver	~PSP	8080	1.1		1.1
exceptionman.prx	sceExceptionManager	~PSP	3248			1.2
fatmsmod.prx	sceMSFAT_Driver	~PSP	71760			1.2
ge.prx	sceGE_Manager	~PSP	8720			1.2
gpio.prx	sceGPIO_Driver	~PSP	3184			1.2
hpremote.prx	sceHP_Remote_Driver	~PSP	6800			1.2
i2c.prx	sceI2C_Driver	~PSP	4368	1.1		1.1
idstorage.prx	sceIdStorage_Service	~PSP	7072	1.1		1.1

ifhandle.prx	sceNetIfhandle_Service	~PSP	10848	1.1		1.1
impose.prx	sceImpose_Driver	~PSP	32480			1.2
init.prx	sceInit	~PSP	7056			1.2
interruptman.prx	sceInterruptManager	~PSP	9872			1.2
iofilemgr.prx	sceIOFileManager	~PSP	11520			1.2
isofs.prx	sceIsofs_driver	~PSP	23520			1.2
lcdc.prx	sceLCDC_Driver	~PSP	3328	1.1		1.1
led.prx	sceLED_Service	~PSP	2448	1.1		1.1
lfatfs.prx	sceLFatFs_Driver	~PSP	37472			1.2
lflash_fatfmt.prx	sceLflashFatfmt	~PSP	6192	1.1		1.1
libatrac3plus.prx	sceATRAC3plus_Library	~PSP	10192	1.1		1.1
libhttp.prx	SceHttp_Library	~PSP	36896	1.1		1.1
libparse_http.prx	SceParseHTTPheader_Library	~PSP	3008	1.1		1.1
libparse_uri.prx	SceParseURI_Library	~PSP	8112	1.1		1.1
libupdown.prx	SceUpdateDL_Library	~PSP	10928	1.1		1.1
loadcore.prx	sceLoaderCore	~PSP	41168			1.2
loadexec.prx	sceLoadExec	~PSP	8016			1.2
me_for_vsh.prx	me_for_vsh	~PSP	1040	1.1		1.1
me_wrapper.prx	sceMeCodecWrapper	~PSP	13008	1.1		1.1
mebooter.prx	sceMeBooter	~PSP	285856	1.1		1.1
mebooter_umdvideo.prx	sceMeBooter	~PSP	126448	1.1		1.1
mediaman.prx	sceUmd_driver	~PSP	8240			1.2
mediasync.prx	sceMediaSync	~PSP	2816			1.2
memab.prx	sceMemab	~PSP	15216			1.2
memlmd.prx	sceMemlmd	~PSP	8800			1.2
mesg_led.prx	sceMesgLed	~PSP	14128			1.2
mgr.prx	sceMgr_Driver	~PSP	20720			1.2
modulemgr.prx	sceModuleManager	~PSP	13824			1.2
mpeg_vsh.prx	sceMpeg_library	~PSP	19664			1.2
mpegbase.prx	sceMpegbase_Driver	~PSP	4304			1.2
msaudio.prx	sceMsAudio_Service	~PSP	8112			1.2
mscm.prx	sceMScm_Driver	~PSP	16048			1.2
msstor.prx	sceMSstor_Driver	~PSP	20352			1.2
openpsid.prx	sceOpenPSID_Service	~PSP	3136			1.2
peq.prx	scePEQ_Library_driver	~PSP	1728	1.1		1.1
power.prx	scePower_Service	~PSP	12608			1.2
pspnet.prx	sceNet_Library	~PSP	27472	1.1		1.1
pspnet_adhoc.prx	sceNetAdhoc_Library	~PSP	20080			1.2
pspnet_adhoc_auth.prx	sceNetAdhocAuth_Service	~PSP	10832			1.2
pspnet_adhoc_download.prx	sceNetAdhocDownload_Library	~PSP	7904	1.1		1.1
pspnet_adhoc_matching.prx	sceNetAdhocMatching_Library	~PSP	9088	1.1		1.1
pspnet_adhocctl.prx	sceNetAdhocctl_Library	~PSP	17968			1.2
pspnet_ap_dialog_dummy.prx	sceNetApDialogDummy_Library	~PSP	2608	1.1		1.1
pspnet_apctl.prx	sceNetApctl_Library	~PSP	22784			1.2
pspnet_inet.prx	sceNetInet_Library	~PSP	130944			1.2
pspnet_resolver.prx	sceNetResolver_Library	~PSP	6880	1.1		1.1
pwm.prx	scePWM_Driver	~PSP	1904	1.1		1.1
reboot.prx	sceReboot	~PSP	53136			1.2
registry.prx	sceRegistry_Service	~PSP	16896			1.2
rtc.prx	sceRTC_Service	~PSP	11136			1.2
semawm.prx	sceSemawm	~PSP	34768			1.2
sircs.prx	sceSIRCS_IrDA_Driver	~PSP	6464	1.1		1.1
stdio.prx	sceStdio	~PSP	3744			1.2
sysclib.prx	sceSysclib	~PSP	6032			1.2
syscon.prx	sceSYSCON_Driver	~PSP	9936	1.1		1.1
sysmem.prx	sceSystemMemoryManager	~PSP	72304			1.2
sysmem_uart4.prx	sceSystemMemoryManager	~PSP	27536			1.2
sysreg.prx	sceSYSREG_Driver	~PSP	5808	1.1		1.1

systimer.prx	sceSystimer	~PSP	2736	1.1		1.1
threadman.prx	sceThreadManager	~PSP	44512			1.2
uart4.prx	sceUart4	~PSP	2288			1.2
umd9660.prx	sceUmd9660_driver	~PSP	17504			1.2
umdman.prx	sceUmdMan_driver	~PSP	34864			1.2
usb.prx	sceUSB_Driver	~PSP	29248			1.2
usbstor.prx	sceUSB_Stor_Driver	~PSP	8656	1.1		1.1
usbstorboot.prx	sceUSB_Stor_Boot_Driver	~PSP	13088			1.2
usbstormgr.prx	sceUSB_Stor_Mgr_Driver	~PSP	10720			1.2
usbstorms.prx	sceUSB_Stor_Ms_Driver	~PSP	9328	1.1		1.1
usersystemlib.prx	sceKernelLibrary	~PSP	1168	1.1		1.1
utility.prx	sceUtility_Driver	~PSP	9216			1.2
utils.prx	sceKernelUtils	~PSP	10272			1.2
vaudio.prx	sceVaudio_driver	~PSP	2784	1.1		1.1
vaudio_game.prx	sceVaudio_driver	~PSP	1088	1.1		1.1
videocodec.prx	sceVideocodec_Driver	~PSP	3824	1.1		1.1
vshbridge.prx	sceVshBridge_Driver	~PSP	2704	1.1		1.1
wlan.prx	sceWlan_Driver	~PSP	114480			1.2

[PSP] means ~PSP type encrypted file

20.4.2 Boot Configurations

Filename	Description	Format	v1.0 size	version	v1.5 size	version
pspcnf_tbl.txt	List of Possible Configurations	~PSP	432			
pspbtcnf.txt	VSH Configuration	~PSP	1584			
pspbtcnf_game.txt	Game Configuration	~PSP	1376			
pspbtcnf_updater.txt	Updater Configuration	~PSP	1600			

20.4.2.1 Configuration Table pspcnf_tbl.txt

```
vsh /kd/pspbtcnf.txt
game /kd/pspbtcnf_game.txt
updater /kd/pspbtcnf_updater.txt
```

20.4.2.2 VSH Configuration

20.4.2.3 Game Configuration

20.4.2.4 Updater Configuration

20.5 VSH Subdirectory

20.5.1 ETC Subdirectory

Filename	Size
jis2ucs.bin	131072
jis2ucs.cbin	16182
ucs2jis.bin	131072
ucs2jis.cbin	33672

20.5.1.1 Version Info

Filename	Format	Size
index.dat	~PSP	480
version.txt	plain	135

index.dat is used to store version/built information about the current firmware. version.txt is simply the decrypted (plaintext) version of the same data.

All the firmware revisions from 1.00 to 2.01 can load decrypted index.dat (aka version.txt) and share the very same index.dat decryption keys while 2.50+ cannot load decrypted index.dat and cannot load old index.dat (featuring another encryption) either. That move was done by sony to prevent downgrading by swapping the index.dat (as it has been done on 2.00)

Having a corrupted index.dat in flash0:/vsh/etc/ will result on the psp viewing any eboot/umd (including updaters) as corrupted data and wont load those (this happends on all versions up to 2.50 as far as I could test <Ookm>)

When using the 2.50 index.dat with 2.00 firmware revision it will see it as corrupted, as the 2.00 firmware does not have the required keys to decrypt the new index.dat files as well as the newer firmwares no longer possess the keys required to decrypt older index.dat or the ability to load those decrypted.

The hexadecimal Number in the system: line is exactly the value returned by the `sceKernelDevkitVersion` Syscall.

20.5.1.1.1 1.0

```
release:1.00:
build:106,1:root@psp-vsh
system:16214,0x00100000:
vsh:2004_1104_s16214_p3883_v8335:
```

```
release:1.00:
build:228,0,3,1,0:root@psp-vsh
system:17919@release_103a,0x01000300:
vsh:p4029@special_day1,v9972@special_day1,20041201:
```

20.5.1.1.2 1.5

```
release:1.50:
build:376,0,3,1,0:root@psp-vsh
system:20182@release_150,0x01050001:
vsh:p4201@release_150,v11079@release_150,20050201:
```

20.5.1.1.3 1.51

```
release:1.51:
build:513,0,3,1,0:root@psp-vsh
system:22984@release_151,0x01050100:
vsh:p4388@release_151_sc,v12875@release_151_sc,20050507:
```

20.5.1.1.4 1.52

```
release:1.52:
build:555,0,3,1,0:root@psp-vsh
system:23740@release_152,0x01050200:
vsh:p4421@release_152,v13394@release_152,20050525:
```

20.5.1.1.5 2.0

```
release:2.00:
build:725,0,3,1,0:root@psp-vsh
system:26084@release_200,0x02000010:
vsh:p4705@release_200,v15867@release_200,20050726:
target:1:WorldWide
```

20.5.1.1.6 2.01

```
release:2.01:
build:822,0,3,1,0:root@psp-vsh
system:26084@release_200,0x02000010:
vsh:p4793@release_201,v18444@release_201,20050928:
target:1:WorldWide
```

20.5.1.1.7 2.5

```
release:2.50:
build:863,0,3,1,0:root@vsh-build
system:28611@release_250,0x02050010:
vsh:p4810@release_250,v19039@release_250,20051011:
target:1:WorldWide
```

20.5.1.1.8 2.6 from update eboot:

```
release:2.60:
build:962,0,3,1,0:root@vsh-build
system:29904@release_260,0x02060010:
vsh:p5029@release_260,v20391@release_260,20051125:
target::WorldWide
```

from retail (version I) PSP:

```
release:2.60:
build:985,0,3,1,0:root@vsh-build
system:29904@release_260,0x02060010:
vsh:p5029@release_260,v20603@release_260_2,20051209:
target:1:WorldWide
```

20.5.1.1.9 2.7

```
release:2.70:
build:1238,0,3,1,0:builder@vsh-build2
system:33151@release_270,0x02070010:
vsh:p5186@release_270,v22631@release_270,20060420:
target::WorldWide
```

20.5.1.1.10 2.71

```
release:2.71:
build:1299,0,3,1,0:builder@vsh-build2
system:33696@release_271,0x02070110:
vsh:p5218@release_271,v22873@release_271,20060529:
target::WorldWide
```

20.5.1.1.11 2.8

20.5.1.1.12 2.81

```
build:1450,0,3,1,0:builder@vsh-build2
system:35536@release_281,0x02080110:
vsh:p5291@release_281,v24983@release_281,20060828:
target:1:WorldWide
```

20.5.1.1.13 3.0

20.5.1.1.14 3.01

```
release:3.01:
build:1628,0,3,1,0:builder@vsh-build2
system:36993@release_301,0x03000110:
vsh:p5403@release_301,v27265@release_301,20061122:
target:1:WorldWide
```

20.5.2 MODULE Subdirectory

Module Filename	API-Module		v1.0		v1.5	
			size	version	size	version
auth_plugin.prx	auth_plugin_module	[PSP]	5856	1.1		1.1
chnnlsv.prx	sceChnnlsv	[PSP]	8464			1.2
common_gui.prx	sceVshCommonGui_Module	[PSP]	16944	1.1		1.1
common_util.prx	sceVshCommonUtil_Module	[PSP]	15392	1.1		1.1
dialogmain.prx	sceDialogmain_Module	[PSP]	22784	1.1		1.1
game_plugin.prx	game_plugin_module	[PSP]	33168	1.1		1.1
heaparea1.prx	scePafHeaparea_Module	[PSP]	1952	1.1		1.1
heaparea2.prx	scePafHeaparea_Module	[PSP]	1952	1.1		1.1
impose_plugin.prx	impose_plugin_module	[PSP]	4256	1.1		1.1
msgdialog_plugin.prx	sceVshMSDPlugin_Module	plain	8996	1.1		1.1
msvideo_plugin.prx	msvideo_plugin_module	[PSP]	149184	1.1		1.1
music_plugin.prx	music_plugin_module	[PSP]	204608	1.1		1.1
netconf_plugin.prx	sceVshNetconf_Module	[PSP]	39744	1.1		1.1
netplay_client_plugin.prx	sceVshGSPlugin_Module	[PSP]	16432	1.1		1.1
netplay_server_utility.prx	sceVshGSUtility_Module	[PSP]	10592			1.2
opening_plugin.prx	opening_plugin_module	[PSP]	4960	1.1		1.1
osk_plugin.prx	sceVshOSK_Module	[PSP]	35520	1.1		1.1
paf.prx	scePaf_Module	[PSP]	599072	1.1		1.1
pafmini.prx	scePaf_Module	[PSP]	513184	1.1		1.1
photo_plugin.prx	photo_plugin_module	[PSP]	79056	1.1		1.1
savedata_auto_dialog.prx	sceVshSDAuto_Module	[PSP]	60224	1.1		1.1
savedata_plugin.prx	sceVshSDPlugin_Module	[PSP]	61344	1.1		1.1
savedata_utility.prx	sceVshSDUtility_Module	[PSP]	59344	1.1		1.1
sysconf_plugin.prx	sysconf_plugin_module	[PSP]	42464	1.1		1.1
update_plugin.prx	update_plugin_module	[PSP]	15840	1.1		1.1
video_plugin.prx	video_plugin_module	[PSP]	137936	1.1		1.1
vshmain.prx	vsh_module	[PSP]	67040	1.1		1.1

20.5.3 RESOURCE Subdirectory

20.5.3.1 Background Images The background images of the VSH. (60x34 bitmaps).

Filename	Size
01.bmp	6176
02.bmp	6176
03.bmp	6176
04.bmp	6176
05.bmp	6176
06.bmp	6176
07.bmp	6176
08.bmp	6176
09.bmp	6176
10.bmp	6176
11.bmp	6176
12.bmp	6176

20.5.3.2 Localized Resources

Filename	Size
auth_plugin.rco	4556
game_plugin.rco	57148
gameboot.pmf	200704
impose_plugin.rco	87828
msgdialog_plugin.rco	7028
msvideo_plugin.rco	158124
music_plugin.rco	220976
netconf_dialog.rco	68552
netplay_plugin.rco	12560
opening_plugin.rco	254480
osk_plugin.rco	318548
osk_utility.rco	121384
photo_plugin.rco	182604
savedata_plugin.rco	68328
savedata_utility.rco	64428
sysconf_plugin.rco	151540
system_plugin.rco	98136
system_plugin_bg.rco	10776
system_plugin_fg.rco	45508
topmenu_plugin.rco	216320
update_plugin.rco	14048
video_plugin.rco	26464
video_plugin_videotoolbar.rco	115888

21 Flash Memory Structure (flash1)

/DIC
/REGISTRY
/VSH
  /THEME

21.1 DIC Subdirectory

Filename	Size
atokl0.dat	15360

21.2 REGISTRY Subdirectory

contains the System Registry

Filename	Size
system.ireg	?
system.dreg	?

21.3 VSH Subdirectory

21.3.1 THEME Subdirectory

22 Memory Stick Structure

```
/PSP
  /GAME
    /UPDATE
  /MUSIC
  /PHOTO
  /SAVEDATA
  /SYSTEM
    /BROWSER
/MP_ROOT
  /100MNV01
  /01MAQ100
  /100MAQ10
/HIFI
/CONTROL
  /PACKAGES
    /PKGxxxxx
/DCIM
  /101MSDCF
/MISC
```

22.1 Root Directory

In the root directory there are three entries which are of relevance to the PSP. The first is the file `MEMSTICK.IND` (or `MSTK_PRO.IND`) which just seems to be a indication that the stick is formatted (it is not specific to the PSP). The second is the directory psp which contains subdirectories for the different types of data used by the PSP. These are game, music, photo, and savedata. Not all subdirectories may exist if no data of the corresponding type is stored. The contents of the subdirectories are detailed in the following sections. In addition, there may be a mp_root directory in the root. This directory is for storing video, and should contain only a subdirectory called 100mnv01.

22.1.1 PSP Subdirectory

22.1.1.1 Game Subdirectory The game directory is for PSP software to be run directly from the memory stick. The Files are in PBP format (see Fileformats Section)

22.1.1.1.1 Update Subdirectory official Firmware Updates should be placed here.

22.1.1.2 Music Subdirectory The music directory contains audio tracks for the music player. MPEG layer 3 files can be used as long as their filenames end with ".mp3". ID3 tags are supported and will be displayed by the player. It is possible to create subdirectories to put the tracks in, but only one level of subdirectories is supported.

22.1.1.3 Photo Subdirectory This directory contains picture files that can be viewed in the photo viewer. The files should be in JPEG format, and the filenames should end with ".jpg". Like with the music directory, one level of subdirectories is possible.

22.1.1.4 Savedata Subdirectory This is where the data saved by games goes. Each game creates a subdirectory with the product code of the game (e.g. ILJS00002) to get a private namespace, and then adds the following files to it:

- ▷ `ICON0.PNG`

  A still picture icon in PNG format (24 bits per pixel, 144Œ80 pixels (standard); 300x170 (maximum))

- ▷ `ICON1.PMF`

  An animated version of the same icon, file format currently unknown. (Optional.)

▷ PIC1.PNG

A full-screen background picture for the file manager in PNG format (24 bits per pixel, 480Œ272 pixels) (Optional.)

▷ SND0.AT3

Background music to play in the file manager, ATRAC3plus encoded in a WAV file. (Optional.) must not be larger than 500kb, and not longer than 55 seconds.

▷ PARAM.SFO

Metadata about the game, such as parental rating information. This is a PSF file with a category of MS. In addition to this, the game will of course have its actual save data, typically in a file called data.bin although any name could be used as well as multiple files.

22.1.2 MP_Root Subdirectory

22.1.2.1 100MNV01 Subdirectory Here video clips can be stored for viewing in the video player. According to the manual, the clip should be encoded using MPEG-4 (H.264/AVC MP Level3), but I have not yet found one that works... The maximum allowed bitrate is specified as 768kbps. Filenames must be on the format m4vnnnnn.mp4, where nnnnn is a 5 digit number. Remember that the mp_root directory should be in the root directory and not in the psp subdirectory. A thumbnail file can optionally be included, and will give a visual indication of the video's contents, as well as include any custom title. It must share the filename of the video it belongs to, but ends in a .THM extension instead of .MP4.

22.1.2.2 01maq100 Subdirectory

22.1.2.3 100maq10 Subdirectory used for AVC on Firmware 2.0 and newer

22.1.3 HIFI Subdirectory

used for DRM Protected ATRAC3 files

▷ A3xxxxxx.MSA

ATRAC3 or ATRAC3PLUS song files

▷ GPxxxxx.MSF

ATRAC3 or ATRAC3PLUS group info and names

▷ PBLIST.MSF

▷ GPLIST.MSF

▷ MGCRL.MSF

▷ 0001000A.MSF

22.1.4 CONTROL Subdirectory

used for DRM Protected ATRAC3 files

▷ NAME.MSF

22.1.4.1 PACKAGES Subdirectory

▷ DEVICE.SAL

22.1.4.1.1 PKGxxxxx Subdirectory

▷ package.xml

Song information in XML format similar in function to ID3V2 tags

22.1.5 DCIM Subdirectory

used by the Sony Cybershot Camera for Photos in jpg format

22.1.6 MISC Subdirectory

used by the Sony Cybershot Camera, ignored by the PSP

23 UMD Game Structure

/PSP_GAME
  /SYSDIR
  /USRDIR

23.1 Root Directory

▷ UMD_DATA.BIN

start	end	size	description
0x00		0x0b	Gamecode (terminated by 0x7c)
0x0b		0x11	unique disk id (terminated by 0x7c)
0x1c		0x05	number of disk ? (terminated by 0x7c)
0x21		0x0f	? (terminated by 0x7c)

23.1.1 PSP_GAME Subdirectory

▷ ICON0.PNG
thumbnail icon

▷ ICON1.PNG
thumbnail icon highlighted

▷ ICON1.PMF
movie icon highlighted

▷ PARAM.SFO

▷ SND0.AT3
ambient sound

▷ PIC0.PNG

▷ PIC1.PNG
background image

note: the files in this directory resemble the contents of the PBP fileformat (see fileformats section)

23.1.1.1 Sysdir Subdirectory

▷ EBOOT.BIN
encrypted main executable

▷ BOOT.BIN
main executable

23.1.1.2 Usrdir Subdirectory contains the 'user' game files which can be different for any game.

24 UMD Video Structure

```
/UMD_VIDEO
  /RESOURCE
  /CLIPINF
  /STREAM
```

24.1 Root Directory

24.1.1 UMD_VIDEO Subdirectory

- ▷ PARAM.SFO
- ▷ ICON1.PMF
- ▷ SND0.AT3
- ▷ ICON0.PNG
- ▷ PIC0.PNG
- ▷ PIC1.PNG
- ▷ PLAYLIST.UMD

24.1.1.1 RESOURCE Subdirectory

- ▷ EN100000.RCO

24.1.1.2 CLIPINF Subdirectory

- ▷ xxxxx.CLP (x = 0...9)

24.1.1.3 STREAM Subdirectory

- ▷ xxxxx.MPS (x = 0...9)

25 UMD Audio Structure

```
/UMD_AUDIO
  /RESOURCE
  /CLIPINF
  /STREAM
```

25.1 Root Directory

25.1.1 UMD_VIDEO Subdirectory

- ▷ PARAM.SFO
- ▷ ICON1.PMF
- ▷ SND0.AT3
- ▷ ICON0.PNG
- ▷ PIC0.PNG
- ▷ PIC1.PNG
- ▷ PLAYLIST.UMD

25.1.1.1 RESOURCE Subdirectory

- ▷ EN100000.RCO

25.1.1.2 CLIPINF Subdirectory

- ▷ xxxxx.CLP (x = 0...9)

25.1.1.3 STREAM Subdirectory

- ▷ xxxxx.MPS (x = 0...9)

26 File Formats

Note on the Tools Sections: at the bottom of every Fileformats Section there might be a list of some related Tools.

26.1 ELF (Executable & Linkable Fileformat)

this is an Industry-Standard Fileformat used by many Operating Systems, Compilers etc. (refer to one of the many free Documentations for Details)

26.1.1 Tools

since this is a widely accepted standard, many available (non PSP specific) tools support it, for example

▷ `psp-objdump` (GNU) show contents, structure, disassemble...

26.2 PRX (PSP Relocateble eXecutable)

Sony's PRX (PSP Relocation eXecutable?) format is a relocation executable based on the standard ELF format. It is distinguised from a normal ELF file by having customised Program Headers, Non-standard MIPS relocation sections and a unique ELF type.

26.2.1 Program Headers

A valid PRX must have at least one program header in order to be loadable, due to the way the relocation entries work. In all program headers the Physical address is not used in the way it is described in the ELF documentation. In the first program header in the list the physical address is actually set to the offset of the .rodata.sceModuleInfo in the PRX file. It is not the load address in memory. In any subsequent program headers the physical address is set to 0. Just to slightly complicate matters if the PRX file is a kernel module then the most significant bit must be set in the phsyical address of the first program header.

As a side note the data referenced by the Program Headers must at least be aligned to 16 byte boundaries otherwise the kernel ELF loader will fail (tested on v1.0 and v1.5).

26.2.2 special Sections

26.2.2.1 .sceStub.text (Systemcall Stubs)

```
sceXXX:
        jr $ra
        nop
```

26.2.2.2 .lib.ent.top (Marks Beginning of Entry Section)

contains one 32bit word with the value 0x00000000

26.2.2.3 .lib.ent:

_library_entry:

	description
32bit word	Addr: Name of Export Library (default: 0)
u16	BCD Version
u16	module attributes
u8	size of export entry in dwords
u8	number of variables
u16	number of Functions
32bit word	Addr: __entrytable in .rodata.sceResident

26.2.2.4 .lib.ent.btm (Marks End of Entry Section)

contains one 32bit word with the value 0x00000000

26.2.2.5 .lib.stub.top (Marks Beginning of Stub Section)

contains one 32bit word with the value 0x00000000

26.2.2.6 .lib.stub (Stub Entries) __stub_module_sceXXX:

	description
32bit word	Addr: __stub_modulestr in .rodata.sceResident
u16	Import Flags
u16	Library Version
u16	Number of Stubs to Import
u16	Size of the Stub itself (in 32bit words)
32bit word	Addr: __stub_nidtable in .rodata.sceNid
32bit word	Addr: sceXXX stub in .sceStub.text

26.2.2.7 .lib.stub.btm (Marks End of Stub Section) contains one 32bit word with the value 0x00000000

26.2.2.8 .rodata.sceModuleInfo: module_info:

	description
u16	Module Attributes 0x0000 Module starts in User Mode 0x1000 Module starts in Kernel Mode
u16	Module Version (2 chars)
28 bytes	Module Name (0 terminated)
32bit word	Addr: GP
32bit word	Addr:.lib.ent
32bit word	Addr:.lib.ent.btm
32bit word	Addr:.lib.stub
32bit word	Addr:.lib.stub.btm

26.2.2.9 .rodata.sceResident (magic words and their memory offsets)

1. first comes a list of magic words (__entrytable),a PRX (PSP module) can have

Magic	description
0xd3744be0	module_bootstart
0x2f064fa6	module_reboot_before
0xadf12745	module_reboot_phase
0xd632acdb	module_start
0xcee8593c	module_stop
0xf01d73a7	module_info
0x0f7c276c	

2. now immediatly follows a list of the memory offsets for the magic (referenced in .lib.stub)

26.2.2.10 .rodata.sceNid (Import stubs hashes; referenced in .lib.stub)

26.2.3 Custom Relocation Format

The first customisation is the section type of the PRX relocation entries differ from that used in standard ELFs. In standard ELFs a relocation section is of type 9, in a PRX they are of type 0x700000A0. The second customisation is in the entries themselves. Each entry is 2 32bit words, the first word is the offset field of the relocation, the second is a compound structure consisting of the standard MIPS relcocation type and a custom base selection field.

This is represented in C like this:

```
// Defines for the r_info field
#define ELF32_R_ADDR_BASE(i) (((i)>>16) & 0xFF)
#define ELF32_R_OFS_BASE(i) (((i)>>8) & 0xFF)
#define ELF32_R_TYPE(i) (i&0xFF)

typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
} Elf32_Rel;

// MIPS Reloc Entry Types
#define R_MIPS_NONE 0
#define R_MIPS_16 1
#define R_MIPS_32 2
#define R_MIPS_REL32 3
#define R_MIPS_26 4
#define R_MIPS_HI16 5
#define R_MIPS_LO16 6
#define R_MIPS_GPREL16 7
#define R_MIPS_LITERAL 8
#define R_MIPS_GOT16 9
#define R_MIPS_PC16 10
#define R_MIPS_CALL16 11
#define R_MIPS_GPREL32 12
```

OFS_BASE determines which program header the r_offset field is based from. So if r_offset is 0x100 and OFS_BASE is 0 (which is a PH starting at address 0) then the address to read is at 0x100.

ADDR_BASE determines which program header the current address value in memory should be relocated from. So for example if ADDR_BASE was 1, program header 1 is loaded to 0x1000 and the current address stored in the ELF is 0xF0 then the resulting address is 0x10F0.

26.2.4 Unique ELF type

PRX files report the value 0xFFA0 as their type in the header instead of 0x0002 which is usual for normal MIPS ELF files.

26.2.5 Tools

- ▷ `prxtool` (Tyranid) show content, structure, convert prx to elf, create idc script...
- ▷ `psp-prxgen` (Tyranid) create prx from elf
- ▷ `nidattack` (adresd, djhuevo) bruteforce NID cracker
- ▷ `prxdecrypt` (MrBrown, Tyranid, John Kelley) decrypt [runs on PSP]

26.3 PBP

A PBP file collects the files needed for a game executable from a MemoryStick into a single file, for easier transfer. The files are simply concatenated with a small index at the start. There does not seem to be any alignment requirements.

All the offsets are in bytes from the beginning of the PBP file, and store in unsigned little endian 32 bit format (ul32).

start	end	size	description
0	3	4	0 "PBP" A file type identification cookie. A zero byte is followed by the three uppercase ASCII characters "PBP"
4	7	4	0 0 1 0 This might be some kind of indication of the PBP version. Currently it's always two 0 bytes followed by a 1 byte and then one more 0 byte.
8	11	4	ul32 Offset of param.sfo data
12	15	4	ul32 Offset of icon0.png data (thumbnail icon)
16	19	4	ul32 Offset of icon1.pmf data (movie icon highlighted)
20	23	4	ul32 Offset of PNG image of unknown purpose (thumbnail icon highlighted ?)
24	27	4	ul32 Offset of pic1.png data (background image)
28	31	4	ul32 Offset of snd0.at3 data (ambient sound)
32	35	4	ul32 Offset of PSP data
36	39	4	ul32 Offset of PSAR data

26.3.1 Tools

▷ `unpack-pbp` (Dan Peori aka Oopo) show content, structure, extract ...

▷ `pack-pbp` (Dan Peori aka Oopo) create pbp file

26.4 PSF (SFO)

PSF files are used in various places on the PSP to store metadata about other files. It contains a list of keys, and the values associated with these keys. This can be information such as parental level, and language. Numerical data is stored in little endian format, I will use the notation ul32 for "unsigned little endian 32 bit" etc.

The file starts with a header, giving the number of key/value pairs and the offsets for the main parts of the file:

start	end	size	description
0	3	4	0 "PSF" A file type identification cookie. A zero byte is followed by the three uppercase ASCII characters "PSF".
4	7	4	1 1 0 0 This might be some kind of indication of the PSF version. Currently it's always two 1 bytes followed by two 0 bytes.
8	11	4	ul32 Offset from the start of the file to the start of the key table (in bytes)
12	15	4	ul32 Offset from the start of the file to the start of the value table (in bytes)
16	19	4	ul32 Number of key/value pairs in the index

This header is immediately followed by the index table, which has one entry per key/value pair. This table seems to always be sorted alphabetically on the key string, allowing binary search to be used, although it is unknown if this is actually guaranteed. The entries look like this:

start	end	size	description
0	1	2	ul16 Offset of the key name into the key table (in bytes)
2	2	1	4 Unknown, always 4. Maybe alignment requirement for the data?
3	3	1	ul8 Datatype of the value, see below.
4	7	4	ul32 Size of value data, in bytes
8	11	4	ul32 Size of value data plus padding, in bytes
12	15	4	ul32 Offset of the data value into the value table (in bytes)

Value data is always aligned to a 4 byte boundary, so if the size of the data is not dividable by four, the data is padded with zero bytes. The two size fields in the index entry gives the size with and without this padding, respectively. It is allowed to add arbitrary amounts of extra padding (as long as alignment is ensured), which makes it easier to modify data in place. Some games seem to take advantage of this to update the text descriptions as the player progresses in the game.

After the index table comes the key table, at the offset (from the beginning of the file) indicated in the file header. Each key is a NUL-terminated ASCII string. The keys are referenced from the index table by offset from tge key table start, so the first key will have offset 0.

The last part of the file is the value table, again at an offset indicated in the file header. Since value data is required to be aligned, zero padding may exist between the key table and the value table. The offset in the file header will indicate the true start of the value table though.

The type of data in the value table depends on the type field of the index entry that references that particular value. The known types are:

Code	Type	description
0	BIN	Arbitrary binary data, interpretation depending on key
2	TXT	UTF-8 text string, NUL-terminated. (The NUL is included in the data size.)
4	INT	An sl32 integer

Before listing the various known keys, the key CATEGORY should be mentioned. This key exists in all PSF files, and indicate the type of entity described by the PSF file. It has TXT data, and the currertly known values are:

category	description	
WG	WLAN Game	a game runable via Gamesharing
MS	MemoryStick Save	a savegame
MG	MemoryStick Game	a game runnable from MemoryStick
UG	UMD Game	a game runnable from UMD
UV	UMD Video	
UA	UMD Audio	
UC	UMD Cleaning Disc	

Depending on the category, different keys may be relevant. In the following table of observed keys, an * indicates that the key occurs in that category of PSF.

key	type	WG	MS	MG	UG	description
BOOTABLE	INT			*	*	Setting this to 1 seems to indicate that the game should be auto-launched at bootup.
CATEGORY	TXT		*	*	*	Category of PSF, as per the table above
DISC_ID	TXT			*	*	Product number of the game(?), e.g. "ABCD-00000"
DISC_NUMBER	INT				*	Which disc (out of DISC_TOTAL) is this? (Counts from 1.)
DISC_TOTAL	INT				*	Total number of UMD discs for this game.
DISC_VERSION	TXT			*	*	Version of the game(?), e.g. "1.00"
DRIVER_PATH	TXT			*		Unknown.
LANGUAGE	TXT			*		Language of the game. "JP" indicates Japanese, even though this is not the proper ISO 639 code...
PARENTAL_LEVEL	INT		*	*	*	Minimum parental control level needed to access this file (1-11, 1=general audience, 5=12 years, 7=15 years, 9=18 years)
PSP_SYSTEM_VER	TXT			*	*	Version of PSP system software required to run the game(?), e.g. "1.00"
REGION	INT			*	*	Bitmask of allowed regions. 0x8000 is region 2?
SAVEDATA_DETAIL	TXT		*			Text shown under the "Details" heading in the save game menu. Can contain multiple lines of text by embedding CR LF.
SAVEDATA_DIRECTORY	TXT		*			The name of the subdirectory to savedata where this game stores its savefiles (e.g. UCJS10001)
SAVEDATA_FILE_LIST	BIN		*			A list of filenames the game uses for the actual save data (typically something like "DATA.BIN"). Data format currently unknown
SAVEDATA_PARAMS	BIN		*			Additional parameters of unknown function and data format.
SAVEDATA_TITLE	TXT		*			Text shown under the "Saved Data" heading in the save game menu.
TITLE	TXT		*	*	*	Text shown under the "Game" heading in the save game menu.
TITLE_0	TXT		*	*	*	Localized version of the TITLE attribute: Japanese
TITLE_2	TXT		*	*	*	Localized version of the TITLE attribute: French
TITLE_3	TXT		*	*	*	Localized version of the TITLE attribute: Spanish
TITLE_4	TXT		*	*	*	Localized version of the TITLE attribute: German
TITLE_5	TXT		*	*	*	Localized version of the TITLE attribute: Italian
TITLE_6	TXT		*	*	*	Localized version of the TITLE attribute: Dutch
TITLE_7	TXT		*	*	*	Localized version of the TITLE attribute: Portuguese
TITLE_8	TXT		*	*	*	Localized version of the TITLE attribute: Russian
UPDATER_VER	TXT			*		Used by the firmware updater program to denote the version it upgrades the firmware to.

26.4.1 Tools

▷ `SFOParse` (Chris Barrera a.k.a. Gorim) show contents

▷ `mksfo` (MrBrown) create file

26.5 PSP

start	end	size	description
0x00	3	4	'~PSP'
0x04		2	attribute 1 SCE_MODULE_ATTR_CANT_STOP 2 SCE_MODULE_ATTR_LOAD 4 SCE_MODULE_ATTR_START
0x06		2	comp_attribute 1 FLAG_COMPRESS 2 FLAG_NORELOC (ie. norel=PFX; rel=PRX)
0x08		1	module version lo
0x09		1	module version hi
0x0a		28	name
0x26		1	fileformat version (=1)
0x27		1	nsegments
0x28		4	elf_size (unencrypted)
0x2c		4	psp_size (encrypted)
0x30		4	entry
0x34		4	modinfo_offset (high 8 bits are substracted from low 24 bits)
0x38		4	bss_size
0x3c			alignment (4 16bit values)
0x44			address (4 32bit values)
0x54			size (4 32bit values)
0x64			? (6 32bit values)
0x7c		1	type
0x7d		3	? (3 8bit values)
0x80		0x30	?
0xb0		4	elf_size_comp; (*1) psp_size - 0x150 (== elf_size if uncompressed file)
0xb4		4	? always 0x00000080 ?
0xb8		0x18	? always 0x00 ?
0xd0		4	ID ?
0xd4		0x7c	?

*1) elf_size_comp is the size of the compressed elf; if the file is not compressed, it is equal to elf_size; rounded up to the next align boundary, is equal to psp_size - 0x150

26.5.1 Tools

▷ `psardump` (PspPet) decrypt [runs on PSP]

26.6 PSAR

26.6.1 Structure

▷ 1. Header

▷ 2. type A section

a. Header

b. Data

▷ 3. type A section

a. Header

b. Data

▷ 4. type B section

a. Header

b. Data

... alternating type A and type B sections ...

- ▷ N-1.type A section
  a. Header
  b. Data
- ▷ N. type B section
  a. Header
  b. Data

Type A : 272 bytes (0x110)
Type B : Variable size data

26.6.2 Header

start	end	size	description
0	3	4	'PSAR'
4	7	4	0x01, 0x00, 0x00, 0x00
8	11	4	Size of the archive file (not including the PSAR header)
12	15	4	0x01, 0x00, 0x00, 0x00

26.6.3 Section Header

start	end	size	description
0		0xb0	??
		4	`u32 Size of data (without padding)`
		`0x04`	`[0] always 0x80 ??`
		`0x18`	`[*] Always 0x00 ??`
		`0x04`	`[3] always 0x06 ??`
		`0x0C`	??
		`0x70`	??

26.6.4 Type A Section (Data Block)

Data in Sections is padded to 16 bytes alignment. A "type 1" Section always contains 0x110 bytes of Data, and 0x260 bytes total (including Header).

26.6.5 Type B Section (compressed Data Block)

A "type 2" Section contains variable amount of Data.

26.6.6 Tools

- ▷ `psardump` (PspPet) extract, unpack and decrypt files [runs on PSP]

26.7 Gamesave

26.7.1 Tools

26.8 PMF (PSMF)

PSMF, or PlayStation Movie Format, is a proprietary movie format created by Sony for the PSP. PSMF videos can be as small as 64x64 pixels, and have a framerate of 29.97fps. The video codec used is H.264, also known as MPEG-4 Part 10 AVC. The audio codec is the Sony proprietary ATRAC3plus.

start	end	size	description
0x00	0x03	4	'PSMF'
0x04	0x07	4	'0012' (icon) or '0014' (movie)
0x0c	0x0f	4	the filesize without the header (Filesize of pmf in bytes - 2,048 bytes)
0x5c	0x5f	4	Total time (take the total value and then div it by 60 then 30 then 60)
0x76	0x79	4	Total time (take the total value and then div it by 60 then 30 then 60)
0x8d	0x8e	2	width of the movie (add a zero)
0x8f	0x90	2	height of the movie (add a zero)

The PMF file has a 2048 byte header, the actual MPEG-2 Program Stream starts with a 32-bit "pack code" which is 0x000001BA; this appears 2048 bytes into the file.

26.9 PGF

The PSP font format (.PGF files) is a bitmap based font format. Each letter (as well as its shadow) is a single, 4bpp bitmap, saved in the font file in a RLE compressed form. The bitmaps are encoded using either vertical or horizontal rows, depending on a certain 2-bit field in character metrics.

Every [character, shadow] bitmap pair is preceded by a character metrics record. For Latin fonts the length of this record appears to be 12 bytes (with an optional 7-byte extension), for other families it's different. It's not known at this time what is the determinant of the record length. The metrics record contains the following fields:

- ▷ 14-bit offset of the shadow header record
- ▷ 7-bit width
- ▷ 7-bit height
- ▷ 7-bit signed horizontal adjustment
- ▷ 7-bit ascender
- ▷ 2-bit transposition (1 - horizontal rows, 2 - vertical rows)
- ▷ 1-bit modified record field (adds a 7-byte extension to the 12-byte header for ltn0.pgf)
- ▷ 46 bits of unknown data
- ▷ 5-bit horizontal advance

To find the character metrics one has to read the main pointer table. The table is constructed of N-bit pointers, where N is found in the file header at offset 0x1C. The number of pointers (and characters) can be found in the file header at offset 0x14.

It is not known yet how to locate the main pointer table.

The RLE compression works on 4-bit nibbles (the low nibble of a byte is considered to precede the high nibble in the stream). There are two sequences defined for this RLE:

- ▷ a nibble N<8: take next nibble and replicate N+1 times into the output stream
- ▷ a nibble N>7: take next 16-N nibbles and copy directly into the output stream

26.9.1 Tools

- ▷ `ttf2pgf` (Skylark) convert Truetype to pgf format
- ▷ `mkfontset` (Skylark) create a set of fonts suitable for the PSP Firmware

26.10 THM

THM files, or "thumbnail" files, are nothing more than JPEG images. Specifically, they are 160x120 pixels, and use the .THM file extension.

26.11 MP4

note: this refers to MP4 files as required by the player in the VSH

- ▷ Video Limitation

  Resolution: 320 x 240 (QVGA), Nonstandard resolutions can be used but are still limited to the 76,800 pixel resolution of QVGA.

  Codec: MPEG-4 SP (Simple Profile), which has different headers than the more common MPEG-4 formats.

- ▷ Audio Limitation

  Codec: AAC

  Sampling Rate: 24000hz

  Bitrate Limitation: 1-768kb/s & 1500kb/s. Any combination of video and audio bitrate that is equal to or less than 768kb/s is acceptable (i.e. 640kb/s video + 128kb/s audio = 768kb/s total, or 300kb/s video + 32kb/s audio = 332kb/s total). The PSP also supports a bitrate of 1500kb/s, but no bitrates inbetween 768kb/s and 1500kb/s.

note: ffmpeg can create PSP compatible mpeg4 files using the '3gp' profile

26.12 AT3

26.13 PNG

these are standard PNG image files.

26.14 RCO

.rco files are localized resources.

26.15 IREG

Block Mapping File for the System Registry

26.15.1 Header

IREG starts with a 0x5C-byte header

offset	size	description
0x00	4	?
0x04	4	?
0x08	0x14	full SHA-1 checksum, possibly of the whole file (with checksum bytes cleared before checksumming)
0x1c		?
0x58	4	?

26.15.2 Entries

IREG entries are - for a change - 0x3a-byte and there are 256 of them (after the header). Only a few fields of the IREG entry are known, the most important being:

offset	size	description
0x04	0x02	parent index (16-bit, little endian) - it's the index of the parent entry in the IREG (1.5 and 2.0 firmwares differ about the "no parent" value - 0x0000 or 0xFFFF :)
0x0a	0x02	number of entries in the DREG block described by this IREG entry (16-bit, little endian)
0x0c	0x02	number of DREG sectors used by this IREG entry (16-bit, little endian)
0x10	0x1c	entry name (28 bytes, null-terminated)
0x2c		
0x2c	7*0x0e	7-sector chain description

26.15.2.1 Sector chains Sector chains are described by the 14-byte field, made up of 7 16-bit little endian DREG sector indices. Those indicate the sequence of DREG sectors in a given DREG block.

offset	size	description
0x00	2	DREG sector index 1
0x02	2	DREG sector index 2
0x04	2	DREG sector index 3
0x06	2	DREG sector index 4
0x08	2	DREG sector index 5
0x0a	2	DREG sector index 6
0x0c	2	DREG sector index 7

26.16 DREG

Every 512-byte DREG sector contains a certain number (specified in the IREG and in the DREG header) of 32-byte entries.

offset	size	description
0	16*0x20	DREG Entry

26.16.1 Entry

Type	description
1	Subdirectory
2	Integer
3	String
4	Secret
0x0f	Block Header

26.16.1.1 Block header Only the first sector in a block (as defined in the IREG) contains a block header, and it is always the first entry.

offset	size	description
0	1	=0x0F (Entry Type)
1	1	?
2	2	The short (or byte? not sure) is block size in 512-byte units
4-5	2	allocation unit (size of keys? always 32)
6-7	2	(unsigned 16-bit little-endian) - number of free entries in the block
8	2	Number of tags - 1 (start of free space?)
10	2	Number of tag slots (i.e. deducting strings at the end)
12	2	(Short) number of keys following
14-17	4	reduced SHA-1 checksum for integrity verification (*)
18-?		(MSB of byte 18 - entry 0) - allocation map (1 for an allocated entry)

*) The bytes are computed as follows: calculate SHA1 of a block with checksum bytes zeroed, and then XOR the 20 bytes of the SHA-1 into 4 bytes of checksum. Basically, those bytes are the only protection for data contents (DREG).

26.16.1.2 Subdirectory To enter the directory, a lookup in IREG to retrieve the sector indices is required.

offset	size	description
0	1	=0x01 (Entry Type)
1	31	directory name (null-terminated string)

26.16.1.3 Integer

offset	size	description
0	1	=0x02 (Entry Type)
1	27	name
28	4	(little-endian, signed) value

26.16.1.4 String

offset	size	description
0	1	=0x03 (Entry Type)
1	27	name
28	2	(little endian, unsigned) length value (includes the terminating NUL)
30	1	flag byte of unknown content
31	1	starting DREG entry index

The starting index is the index of the (32-byte) DREG entry in the current block that holds the beginning of the string contents. Remember that string contents can span arbitrarily many entries, and even sectors - they just have to fit in a single block.

offset	size	description
0	32	String Contents

26.16.1.5 Secret

offset	size	description
0	1	=0x04 (Entry Type)
1	27	name
28	2	(little endian, unsigned) length value (includes the terminating NUL)
30	1	flag byte of unknown content
31	1	starting DREG entry index

The starting index is the index of the (32-byte) DREG entry in the current block that holds the beginning of the string contents. Remember that string contents can span arbitrarily many entries, and even sectors - they just have to fit in a single block.

offset	size	description
0	32	String Contents

26.16.2 Tools

- ▷ `parsedreg2` (Skylark, Freeplay)
- ▷ `fixupdreg2` (Skylark, Freeplay) recalculate SHA1 hashes used to ensure data integrity

26.17 CER

ordinal base64 encoded certificate, not encrypted.

26.18 DIC

26.19 flash

raw flash image format used by the ”Undiluted Platinum” Modchip flasher. Contains a linear image of the full Flashrom content (data and spare areas interleaved for each physical page)

26.20 ISO

plain UMD Image. contains a linear image of all sectors of a UMD (unused sectors at the end might be omitted)

26.21 DAX

compressed ISO Image used by ”DAX ISO Loader”

26.22 CSO

compressed ISO Image used by ”Devhook”

26.23 ezip

compressed ISO Image used by ”Epsilon BIOS”

27 Graphic Formats

27.1 1555 ABGR

15 8	7 0
abbb bbgg	gggr rrrr

bit(s)		description
	a	alpha
	b	blue
	g	green
	r	red

27.2 4444 ABGR

15 8	7 0
aaaa bbbb	gggg rrrr

bit(s)		description
	a	alpha
	b	blue
	g	green
	r	red

27.3 565 BGR

15 8	7 0
bbbb bggg	gggr rrrr

bit(s)		description
	b	blue
	g	green
	r	red

27.4 8888 ABGR

31 24	23 16	15 8	7 0
aaaa aaaa	bbbb bbbb	gggg gggg	rrrr rrrr

bit(s)		description
	a	alpha
	b	blue
	g	green
	r	red

27.5 swizzling

Internally, the GE processes textures as 16 bytes by 8 rows blocks (independent of actual pixelformat, so a 32*32 32-bit texture is a 128*32 texture from the swizzlings point of view). When you are not swizzling, this means it will have to do scattered reads from the texture as it moves the block into its texture-cache, which has a big impact on performance. To improve on this, you can re-order your textures into these blocks so that it can fetch one entire block by reading sequentially.

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0G 0H 0I 0J 0K 0L 0M 0N 0O 0P 0Q 0R 0S 0T 0U 0V
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	1G 1H 1I 1J 1K 1L 1M 1N 1O 1P 1Q 1R 1S 1T 1U 1V
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F	2G 2H 2I 2J 2K 2L 2M 2N 2O 2P 2Q 2R 2S 2T 2U 2V
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F	3G 3H 3I 3J 3K 3L 3M 3N 3O 3P 3Q 3R 3S 3T 3U 3V
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F	4G 4H 4I 4J 4K 4L 4M 4N 4O 4P 4Q 4R 4S 4T 4U 4V
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F	5G 5H 5I 5J 5K 5L 5M 5N 5O 5P 5Q 5R 5S 5T 5U 5V
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F	6G 6H 6I 6J 6K 6L 6M 6N 6O 6P 6Q 6R 6S 6T 6U 6V
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F	7G 7H 7I 7J 7K 7L 7M 7N 7O 7P 7Q 7R 7S 7T 7U 7V

The block above is a 32 bytes by 8 lines texture block (so it could be a 8*8 32-bit block, or a 16*8 16-bit block). Each pixel is represented here by a vertical index (first value) of 0-7. The second index is the horizontal index, ranging from 0-U. When reorganizing this for swizzling, we will order the data so that when the GE needs to read something in the first 16Œ8 block, if can just fetch that entire block, instead of offsetting into the texture for each line it has to read. The resulting swizzled portion looks like this:

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
0G 0H 0I 0J 0K 0L 0M 0N 0O 0P 0Q 0R 0S 0T 0U 0V 1G 1H 1I 1J 1K 1L 1M 1N 1O 1P 1Q 1R 1S 1T 1U 1V
2G 2H 2I 2J 2K 2L 2M 2N 2O 2P 2Q 2R 2S 2T 2U 2V 3G 3H 3I 3J 3K 3L 3M 3N 3O 3P 3Q 3R 3S 3T 3U 3V
4G 4H 4I 4J 4K 4L 4M 4N 4O 4P 4Q 4R 4S 4T 4U 4V 5G 5H 5I 5J 5K 5L 5M 5N 5O 5P 5Q 5R 5S 5T 5U 5V
6G 6H 6I 6J 6K 6L 6M 6N 6O 6P 6Q 6R 6S 6T 6U 6V 7G 7H 7I 7J 7K 7L 7M 7N 7O 7P 7Q 7R 7S 7T 7U 7V

Notice how the rectangular 16*8 blocks have ended up as sequential data, ready for direct reading by the GE.

27.6 S3TC Compression

Texture formats 8, 9 and 10 are DXT1, DXT3 and DXT5. The hardwares format is a little different from the standard (as you'd find in a .DDS file, for example).

27.6.1 DXT1

For DXT1, each 4x4 texel block has 2 16-bit 565 colours, and 16 2-bit per-texel fields (8 bytes/block). The PSP hardware expects the per-texel bits to come first, followed by the two colours. Colours are in RGB 565 format.

27.6.2 DXT3

27.6.3 DXT5

For DXT3 and DXT5, each 4x4 block has 8 bytes of alpha data followed by 8 bytes of pixel data. The PSP reverses this, so it wants the pixel data followed by alpha data. Also, the pixel data is normally encoded in the same way as the DXT1 blocks, which is also true for the PSP. The encoding is the same as for DXT1 textures, the colours are in RGB 565 format.

28 Boot Process

28.1 Cold Boot

28.1.1 embedded Bootstrap

does minimal initialization, copies Stage 1 to RAM and executes it.

28.1.2 IPL Stage 1

decrypts and executes Stage 2

28.1.3 IPL Stage 2

initializes the System, boots PRXs in 'VSH Mode' (from `/kd/pspbtcnf.txt`) and finally launches the VSH.

28.2 Load Exec

28.2.1 Stage 1

sceKernelLoadExec

- ▷ do some sanity checks
  return 0x80020064 if called from interrupt handler
  0x800200d3 on *file==NULL or other error
- ▷ call LoadExec

LoadExec

- ▷ start "LoadExecBody" as new thread

LoadExecBody

- ▷ call LoadExecAction

LoadExecAction

- ▷ call sub_FCC

sub_FCC
LoadExecAction

- ▷ gunzip to `0x88C00000`
- ▷ call `0x88C00000`, execution continues here (no return)

28.2.2 Stage 2

initializes the System, boots PRXs in 'Game Mode' (from `/kd/pspbtcnf_game.txt`) ,or 'Updater Mode' (from `/kd/pspbtcnf_updater.txt`
if the Executable is launched from an updater directory, and finally launches the Game or Updater. Similar to IPL Stage 2

28.3 Exit Game

initializes the System, boots PRXs in 'VSH Mode' (from `/kd/pspbtcnf.txt`) and finally launches the VSH.

28.4 reboot.prx

29 Kernel

29.1 Devices

29.1.1 Block Devices

Name	r	w	blocksize	seekable	description
msstor:	*	*	512		Memory Stick (whole; mbr, partition1,...)
msstor0:	*	*			alias for msstor:
msstor0p0:					partition 0
msstor0p1:					partition 1
mscm:	*	*		no	Memory Stick
mscm0:	*	*			
mscmhc:	*	*			
mscmhc0:	*	*			
umd:	*		2048		UMD
umd1:					alias for umd:
umd00:					alias for umd:
umd01:					alias for umd:
lflash:	*	*	512		internal flash
lflash?:					(?=any number) alias for lflash:
lflash0:0,0					internal flash, logical partition 0 (flash0)
lflash0:0,1					internal flash, logical partition 1 (flash1)
rda:	*	*	any	no	infrared Port
irda:	*	*	any	no	alias for rda:
irda?:	*	*	any	no	(?=any number) alias for rda:

29.1.2 Filesystems

Name	r	w	seekable	description
fatms0:	*	*		Memorystick
ms0:	*	*		alias for fatms0:
fatms:	*	*		alias for fatms0:
umd0:	*			UMD
isofs:	*			UMD
isofs0:	*			alias for isofs:
flash0:				internal flash, system file volume
flashfat:				alias for flash0:
flashfat0:				alias for flash0:
flash1:				internal flash, configuration file volume
flashfat1:				alias for flash1:
host0:				devkit (SC) fileserver
host1:				devkit (ME) fileserver

29.2 Return Codes

29.2.1 Structure

31 24	23 16	15 8	7 0
....	....	....	....

bit(s)		description
31		0 OK 1 Error
30		0 normal 1 critical
28-29		reserved/unused
16-27		facility
0-15		type of error

29.2.2 Facilities

code	description
0x00000000	General
0x00010000	Errno
0x00020000	Kernel

29.2.3 General Errors

code	description

29.2.4 Errnos

code	description

29.2.5 Kernel Errors

code	description
0x80020001	ERROR
0x80020002	NOTIMP
0x80020032	ILLEGAL_EXPCODE
0x80020033	EXPHANDLER_NOUSE
0x80020034	EXPHANDLER_USED
0x80020035	SYCALLTABLE_NOUSED
0x80020036	SYCALLTABLE_USED
0x80020037	ILLEGAL_SYSCALLTABLE
0x80020038	ILLEGAL_PRIMARY_SYSCALL_NUMBER
0x80020039	PRIMARY_SYSCALL_NUMBER_INUSE
0x80020064	ILLEGAL_CONTEXT
0x80020065	ILLEGAL_INTRCODE
0x80020066	CPUDI

0x80020067	FOUND_HANDLER
0x80020068	NOTFOUND_HANDLER
0x80020069	ILLEGAL_INTRLEVEL
0x8002006a	ILLEGAL_ADDRESS
0x8002006b	ILLEGAL_INTRPARAM
0x8002006c	ILLEGAL_STACK_ADDRESS
0x8002006d	ALREADY_STACK_SET
0x80020096	NO_TIMER
0x80020097	ILLEGAL_TIMERID
0x80020098	ILLEGAL_SOURCE
0x80020099	ILLEGAL_PRESCALE
0x8002009a	TIMER_BUSY
0x8002009b	TIMER_NOT_SETUP
0x8002009c	TIMER_NOT_INUSE
0x800200a0	UNIT_USED
0x800200a1	UNIT_NOUSE
0x800200a2	NO_ROMDIR
0x800200c8	IDTYPE_EXIST
0x800200c9	IDTYPE_NOT_EXIST
0x800200ca	IDTYPE_NOT_EMPTY
0x800200cb	UNKNOWN_UID
0x800200cc	UNMATCH_UID_TYPE
0x800200cd	ID_NOT_EXIST
0x800200ce	NOT_FOUND_UIDFUNC
0x800200cf	UID_ALREADY_HOLDER
0x800200d0	UID_NOT_HOLDER
0x800200d1	ILLEGAL_PERM
0x800200d2	ILLEGAL_ARGUMENT
0x800200d3	ILLEGAL_ADDR
0x800200d4	OUT_OF_RANGE
0x800200d5	MEM_RANGE_OVERLAP
0x800200d6	ILLEGAL_PARTITION
0x800200d7	PARTITION_INUSE
0x800200d8	ILLEGAL_MEMBLOCKTYPE
0x800200d9	MEMBLOCK_ALLOC_FAILED
0x800200da	MEMBLOCK_RESIZE_LOCKED
0x800200db	MEMBLOCK_RESIZE_FAILED
0x800200dc	HEAPBLOCK_ALLOC_FAILED
0x800200dd	HEAP_ALLOC_FAILED
0x800200de	ILLEGAL_CHUNK_ID
0x800200df	NOCHUNK
0x800200e0	NO_FREECHUNK
0x8002012c	LINKERR
0x8002012d	ILLEGAL_OBJECT
0x8002012e	UNKNOWN_MODULE
0x8002012f	NOFILE
0x80020130	FILEERR
0x80020131	MEMINUSE
0x80020132	PARTITION_MISMATCH
0x80020133	ALREADY_STARTED
0x80020134	NOT_STARTED
0x80020135	ALREADY_STOPPED
0x80020136	CAN_NOT_STOP
0x80020137	NOT_STOPPED

0x80020138	NOT_REMOVABLE
0x80020139	EXCLUSIVE_LOAD
0x8002013a	LIBRARY_NOT_YET_LINKED
0x8002013b	LIBRARY_FOUND
0x8002013c	LIBRARY_NOTFOUND
0x8002013d	ILLEGAL_LIBRARY
0x8002013e	LIBRARY_INUSE
0x8002013f	ALREADY_STOPPING
0x80020140	ILLEGAL_OFFSET
0x80020141	ILLEGAL_POSITION
0x80020142	ILLEGAL_ACCESS
0x80020143	MODULE_MGR_BUSY
0x80020144	ILLEGAL_FLAG
0x80020145	CANNOT_GET_MODULELIST
0x80020146	PROHIBIT_LOADMODULE_DEVICE
0x80020147	PROHIBIT_LOADEXEC_DEVICE
0x80020148	UNSUPPORTED_PRX_TYPE
0x80020149	ILLEGAL_PERM_CALL
0x8002014a	CANNOT_GET_MODULE_INFORMATION
0x8002014b	ILLEGAL_LOADEXEC_BUFFER
0x8002014c	ILLEGAL_LOADEXEC_FILENAME
0x8002014d	NO_EXIT_CALLBACK
0x80020190	NO_MEMORY
0x80020191	ILLEGAL_ATTR
0x80020192	ILLEGAL_ENTRY
0x80020193	ILLEGAL_PRIORITY
0x80020194	ILLEGAL_STACK_SIZE
0x80020195	ILLEGAL_MODE
0x80020196	ILLEGAL_MASK
0x80020197	ILLEGAL_THID
0x80020198	UNKNOWN_THID
0x80020199	UNKNOWN_SEMID
0x8002019a	UNKNOWN_EVFID
0x8002019b	UNKNOWN_MBXID
0x8002019c	UNKNOWN_VPLID
0x8002019d	UNKNOWN_FPLID
0x8002019e	UNKNOWN_MPPID
0x8002019f	UNKNOWN_ALMID
0x800201a0	UNKNOWN_TEID
0x800201a1	UNKNOWN_CBID
0x800201a2	DORMANT
0x800201a3	SUSPEND
0x800201a4	NOT_DORMANT
0x800201a5	NOT_SUSPEND
0x800201a6	NOT_WAIT
0x800201a7	CAN_NOT_WAIT
0x800201a8	WAIT_TIMEOUT
0x800201a9	WAIT_CANCEL
0x800201aa	RELEASE_WAIT
0x800201ab	NOTIFY_CALLBACK
0x800201ac	THREAD_TERMINATED
0x800201ad	SEMA_ZERO
0x800201ae	SEMA_OVF
0x800201af	EVF_COND
0x800201b0	EVF_MULTI
0x800201b1	EVF_ILPAT
0x800201b2	MBOX_NOMSG

0x800201b3	MPP_FULL
0x800201b4	MPP_EMPTY
0x800201b5	WAIT_DELETE
0x800201b6	ILLEGAL_MEMBLOCK
0x800201b7	ILLEGAL_MEMSIZE
0x800201b8	ILLEGAL_SPADADDR
0x800201b9	SPAD_INUSE
0x800201ba	SPAD_NOT_INUSE
0x800201bb	ILLEGAL_TYPE
0x800201bc	ILLEGAL_SIZE
0x800201bd	ILLEGAL_COUNT
0x800201be	UNKNOWN_VTID
0x800201bf	ILLEGAL_VTID
0x800201c0	ILLEGAL_KTLSID
0x800201c1	KTLS_FULL
0x800201c2	KTLS_BUSY
0x80020258	PM_INVALID_PRIORITY
0x80020259	PM_INVALID_DEVNAME
0x8002025a	PM_UNKNOWN_DEVNAME
0x8002025b	PM_PMINFO_REGISTERED
0x8002025c	PM_PMINFO_UNREGISTERED
0x8002025d	PM_INVALID_MAJOR_STATE
0x8002025e	PM_INVALID_REQUEST
0x8002025f	PM_UNKNOWN_REQUEST
0x80020260	PM_INVALID_UNIT
0x80020261	PM_CANNOT_CANCEL
0x80020262	PM_INVALID_PMINFO
0x80020263	PM_INVALID_ARGUMENT
0x80020264	PM_ALREADY_TARGET_PWRSTATE
0x80020265	PM_CHANGE_PWRSTATE_FAILED
0x80020266	PM_CANNOT_CHANGE_DEVPWR_STATE
0x80020267	PM_NO_SUPPORT_DEVPWR_STATE
0x800202bc	DMAC_REQUEST_FAILED
0x800202bd	DMAC_REQUEST_DENIED
0x800202be	DMAC_OP_QUEUED
0x800202bf	DMAC_OP_NOT_QUEUED
0x800202c0	DMAC_OP_RUNNING
0x800202c1	DMAC_OP_NOT_ASSIGNED
0x800202c2	DMAC_OP_TIMEOUT
0x800202c3	DMAC_OP_FREED
0x800202c4	DMAC_OP_USED
0x800202c5	DMAC_OP_EMPTY
0x800202c6	DMAC_OP_ABORTED
0x800202c7	DMAC_OP_ERROR
0x800202c8	DMAC_CHANNEL_RESERVED
0x800202c9	DMAC_CHANNEL_EXCLUDED
0x800202ca	DMAC_PRIVILEGE_ADDRESS
0x800202cb	DMAC_NO_ENOUGHSPACE
0x800202cc	DMAC_CHANNEL_NOT_ASSIGNED
0x800202cd	DMAC_CHILD_OPERATION
0x800202ce	DMAC_TOO_MUCH_SIZE
0x800202cf	DMAC_INVALID_ARGUMENT
0x80020320	MFILE
0x80020321	NODEV
0x80020322	XDEV

0x80020323	BADF
0x80020324	INVAL
0x80020325	UNSUP
0x80020326	ALIAS_USED
0x80020327	CANNOT_MOUNT
0x80020328	DRIVER_DELETED
0x80020329	ASYNC_BUSY
0x8002032a	NOASYNC
0x8002032b	REGDEV
0x8002032c	NOCWD
0x8002032d	NAMETOOLONG
0x800203e8	NXIO
0x800203e9	IO
0x800203ea	NOMEM
0x800203eb	STDIO_NOT_OPENED
0x8002044c	CACHE_ALIGNMENT

29.2.6 Network Errors

code	description

29.2.7 unspecified Errors

code	description
0xffffffd0	?
0xffffffd3	prx tag not found?
0xffffffd5	descramble error?

29.3 Versions

29.3.1 1.0

- ▷ The first batch of PSPs was shipped with this firmware in Japan.
- ▷ 1.0 will run an unsigned binary in a PBP file without worry.

29.3.2 1.5

- ▷ 1.5 will refuse to run an unsigned binary in a PBP file, but will execute a bare elf file if you can provide that file after the PSP has already loaded the PBP.
- ▷ the 1.50-US and the 1.50 JP flash0 are identical

Files added/modified from 1.0:

`flash0:/kd/ata.prx`
`flash0:/kd/audio.prx`

```
flash0:/kd/audiocodec.prx
flash0:/kd/blkdev.prx
flash0:/kd/chkreg.prx
flash0:/kd/clockgen.prx
flash0:/kd/codec.prx
flash0:/kd/ctrl.prx
flash0:/kd/display.prx
flash0:/kd/dmacman.prx
flash0:/kd/dmacplus.prx
flash0:/kd/emc_ddr.prx
flash0:/kd/emc_sm.prx
flash0:/kd/exceptionman.prx
flash0:/kd/fatmsmod.prx
flash0:/kd/ge.prx
flash0:/kd/gpio.prx
flash0:/kd/hpremote.prx
flash0:/kd/i2c.prx
flash0:/kd/idstorage.prx
flash0:/kd/ifhandle.prx
flash0:/kd/impose.prx
flash0:/kd/init.prx
flash0:/kd/interruptman.prx
flash0:/kd/iofilemgr.prx
flash0:/kd/isofs.prx
flash0:/kd/lcdc.prx
flash0:/kd/led.prx
flash0:/kd/lfatfs.prx
flash0:/kd/lflash_fatfmt.prx
flash0:/kd/libatrac3plus.prx
flash0:/kd/libhttp.prx
flash0:/kd/libparse_http.prx
flash0:/kd/libparse_uri.prx
flash0:/kd/libupdown.prx
flash0:/kd/loadcore.prx
flash0:/kd/loadexec.prx
flash0:/kd/me_for_vsh.prx
flash0:/kd/me_wrapper.prx
flash0:/kd/mebooter.prx
flash0:/kd/mebooter_umdvideo.prx
flash0:/kd/mediaman.prx
flash0:/kd/mediasync.prx
flash0:/kd/memab.prx
flash0:/kd/memlmd.prx
flash0:/kd/mesg_led.prx
flash0:/kd/mgr.prx
flash0:/kd/modulemgr.prx
flash0:/kd/mpeg_vsh.prx
flash0:/kd/mpegbase.prx
flash0:/kd/msaudio.prx
```

flash0:/kd/mscm.prx
flash0:/kd/msstor.prx
flash0:/kd/openpsid.prx
flash0:/kd/peq.prx
flash0:/kd/power.prx
flash0:/kd/pspbtcnf.txt
flash0:/kd/pspbtcnf_game.txt
flash0:/kd/pspbtcnf_updater.txt
flash0:/kd/pspcnf_tbl.txt
flash0:/kd/pspnet.prx
flash0:/kd/pspnet_adhoc.prx
flash0:/kd/pspnet_adhoc_auth.prx
flash0:/kd/pspnet_adhoc_download.prx
flash0:/kd/pspnet_adhoc_matching.prx
flash0:/kd/pspnet_adhocctl.prx
flash0:/kd/pspnet_ap_dialog_dummy.prx
flash0:/kd/pspnet_apctl.prx
flash0:/kd/pspnet_inet.prx
flash0:/kd/pspnet_resolver.prx
flash0:/kd/pwm.prx
flash0:/kd/reboot.prx
flash0:/kd/registry.prx
flash0:/kd/rtc.prx
flash0:/kd/semawm.prx
flash0:/kd/sircs.prx
flash0:/kd/stdio.prx
flash0:/kd/sysclib.prx
flash0:/kd/syscon.prx
flash0:/kd/sysmem.prx
flash0:/kd/sysmem_uart4.prx (removed, only in 1.00-JP)
flash0:/kd/sysreg.prx
flash0:/kd/systimer.prx
flash0:/kd/threadman.prx
flash0:/kd/uart4.prx
flash0:/kd/umd9660.prx
flash0:/kd/umdman.prx
flash0:/kd/usb.prx
flash0:/kd/usbstor.prx
flash0:/kd/usbstorboot.prx
flash0:/kd/usbstormgr.prx
flash0:/kd/usbstorms.prx
flash0:/kd/usersystemlib.prx
flash0:/kd/utility.prx
flash0:/kd/utils.prx
flash0:/kd/vaudio.prx
flash0:/kd/vaudio_game.prx
flash0:/kd/videocodec.prx
flash0:/kd/vshbridge.prx
flash0:/kd/wlan.prx

flash0:/kd/resource/impose.rsc (only in 1.50-US)
flash0:/vsh/etc/index.dat
flash0:/vsh/etc/jis2ucs.bin
flash0:/vsh/etc/jis2ucs.cbin
flash0:/vsh/etc/version.txt
flash0:/vsh/module/auth_plugin.prx
flash0:/vsh/module/chnnlsv.prx
flash0:/vsh/module/common_gui.prx
flash0:/vsh/module/common_util.prx
flash0:/vsh/module/dialogmain.prx
flash0:/vsh/module/game_plugin.prx
flash0:/vsh/module/heaparea1.prx
flash0:/vsh/module/heaparea2.prx
flash0:/vsh/module/impose_plugin.prx
flash0:/vsh/module/msgdialog_plugin.prx
flash0:/vsh/module/msvideo_plugin.prx
flash0:/vsh/module/music_plugin.prx
flash0:/vsh/module/netconf_plugin.prx
flash0:/vsh/module/netplay_client_plugin.prx
flash0:/vsh/module/netplay_server_utility.prx
flash0:/vsh/module/opening_plugin.prx
flash0:/vsh/module/osk_plugin.prx
flash0:/vsh/module/paf.prx
flash0:/vsh/module/pafmini.prx
flash0:/vsh/module/photo_plugin.prx
flash0:/vsh/module/savedata_auto_dialog.prx
flash0:/vsh/module/savedata_plugin.prx
flash0:/vsh/module/savedata_utility.prx
flash0:/vsh/module/sysconf_plugin.prx
flash0:/vsh/module/update_plugin.prx
flash0:/vsh/module/video_plugin.prx
flash0:/vsh/module/vshmain.prx
flash0:/vsh/resource/auth_plugin.rco
flash0:/vsh/resource/game_plugin.rco
flash0:/vsh/resource/impose_plugin.rco
flash0:/vsh/resource/msgdialog_plugin.rco
flash0:/vsh/resource/msvideo_plugin.rco
flash0:/vsh/resource/music_plugin.rco
flash0:/vsh/resource/netconf_dialog.rco
flash0:/vsh/resource/netplay_plugin.rco
flash0:/vsh/resource/opening_plugin.rco
flash0:/vsh/resource/osk_plugin.rco
flash0:/vsh/resource/osk_utility.rco
flash0:/vsh/resource/photo_plugin.rco
flash0:/vsh/resource/savedata_plugin.rco
flash0:/vsh/resource/savedata_utility.rco
flash0:/vsh/resource/sysconf_plugin.rco
flash0:/vsh/resource/system_plugin.rco
flash0:/vsh/resource/system_plugin_bg.rco

```
flash0:/vsh/resource/system_plugin_fg.rco
flash0:/vsh/resource/topmenu_plugin.rco
flash0:/vsh/resource/update_plugin.rco
flash0:/vsh/resource/video_plugin.rco
flash0:/vsh/resource/video_plugin_videotoolbar.rco
```

29.3.3 1.51

- ▷ The ability to run unencrypted, unsigned binaries was removed in this Firmware.

29.3.4 1.52

- ▷ The first batch of european PSPs was shipped with this firmware

29.3.5 2.0

29.3.5.1 new Features

- ▷ Network
  - ▷ Internet browser was added. (Doesn't yet support Macromedia Flash, some webpages will not be displayed correctly)
- ▷ Video
  - ▷ Jump function was added (UMD Video and UMD Music).
  - ▷ A-B repeat function was added (UMD Video, UMD Music and Memory Stick Duo)
  - ▷ 4:3 screen mode was added (Memory Stick Duo)
  - ▷ Voice switch function was added (Memory Stick Duo)
  - ▷ MP4 AVC support was added (Memory Stick Duo)
- ▷ Music
  - ▷ SonicStage version 3.2 now supports using ATRAC3plus with the Memory Stick PRO Duo on the PSP.
  - ▷ MP4 AAC and WAV PCM support added (Memory Stick Duo)
- ▷ Photo
  - ▷ Wallpaper function was added.
  - ▷ Sending and receiving of images was added.
  - ▷ TIFF, GIF, PNG and BMP support added.
- ▷ Settings
  - ▷ Korean language was added.
  - ▷ Theme setting was added.
  - ▷ Security setting was added.
  - ▷ WPA-PSK support added.

29.3.5.2 updated Files

```
flash0:/data/cert/Equifax_S_CA.cer
flash0:/data/cert/Equifax_S_eBiz_CA-1.cer
flash0:/data/cert/GeoTrust_G_CA.cer
flash0:/font/shadow.pgf
flash0:/kd/cert_loader.prx
flash0:/kd/http_storage.prx
flash0:/kd/libdnas.prx
flash0:/kd/libdnas_core.prx
flash0:/kd/libssl.prx
flash0:/kd/mcctrl.prx
flash0:/kd/pspnet_adhoc_transfer_int.prx
flash0:/kd/resource
flash0:/kd/resource/big5_table.dat
flash0:/kd/resource/cp949_table.dat
flash0:/kd/resource/gbk_table.dat
flash0:/vsh/etc/cp1251ucs.bin
flash0:/vsh/etc/cp1252ucs.bin
flash0:/vsh/etc/ucs2uhc.bin
flash0:/vsh/etc/uhc2ucs.bin
flash0:/vsh/module
flash0:/vsh/module/dnas_plugin.prx
flash0:/vsh/module/htmlviewer_plugin.prx
flash0:/vsh/module/htmlviewer_ui.prx
flash0:/vsh/module/htmlviewer_utility.prx
flash0:/vsh/module/libfont_hv.prx
flash0:/vsh/module/libslim.prx
flash0:/vsh/module/libwww.prx
flash0:/vsh/module/netconf_plugin_auto_bfl.prx
flash0:/vsh/module/netconf_plugin_auto_nec.prx
flash0:/vsh/module/netfront.prx
flash0:/vsh/resource/dnas_plugin.rco
flash0:/vsh/resource/htmlviewer.fbm
flash0:/vsh/resource/htmlviewer.gim
flash0:/vsh/resource/htmlviewer.msg
flash0:/vsh/resource/htmlviewer.res
flash0:/vsh/resource/htmlviewer.snd
flash0:/vsh/resource/htmlviewer_plugin.rco
flash0:/vsh/resource/netfront.rc
flash0:/vsh/resource/netfront.skn
flash0:/vsh/resource/netfront.uhc
flash1:/net/http
ipl:/psp_ipl.bin
```

29.3.5.3

29.3.6 2.01

29.3.6.1 new Features

This was a quick release by Sony to fix the TIFF overflow exploit found in the previous version

29.3.6.2 updated Files

```
paf.prx
index.dat
version.txt
```

29.3.7 2.5

29.3.7.1 new Features

- ▷ Streaming Video Support
- ▷ Unicode support in the Browser with automatic Encoding Detection

- ▷ Save your text size settings in the Browser
- ▷ Save your Browser input history (URLs)
- ▷ Videos with DRM can be played
- ▷ NTP (Network Time Protocol) support
- ▷ WPA and PSK have been added to network setting
- ▷ Korean input keyboard method

29.3.8 2.6

29.3.8.1 new Features

- ▷ Revisions to strengthen security have been added
- ▷ [LocationFree Player] has been added as a feature under [Network]
- ▷ [Auto-Select] and [Unicode (UTF-8)] have been added as options to [Encoding] under [View] in the [Internet Browser] menu bar
- ▷ [Text Size] and [Display Mode] settings of the [Internet Browser] can now be saved
- ▷ The input history of online forms accessed through the [Internet Browser] can now be saved
- ▷ Copyright-protected video can now be played under [Video]. (This applies to video data saved on Memory Stick)
- ▷ [Set via Internet] has been added as an option to [Date & Time Settings] under [Settings]
- ▷ WPA-PSK (AES) has been added as a security method under [Network Settings]
- ▷ Korean input mode has been added to the on-screen keyboard
- ▷ [RSS Channel] has been added as a feature under [Network]
- ▷ [Simplified Chinese (GB18030)] and [Traditional Chinese (Big5)] have been added as options to [Encoding] under [View] in the [Internet Browser] menu bar
- ▷ [Volume Adjustment] has been added as a feature to [LocationFree Player]
- ▷ You can now download video data that supports copyright protection using the [Internet Browser]
- ▷ WMA has been added as a codec that can be played under [Music]. (This applies to music data saved on the Memory Stick)

29.3.9 2.7

- ▷ GTA exploit has been patched ("Load failed. Savegame is corrupted" is message displayed during launch).

29.3.9.1 new Features

- ▷ [Internet Browser] now supports Macromedia Flash contents playback.
  - ▷ You need to enable the Flash contents playback in the [System Settings].
  - ▷ The version of the flash player is Macromedia Flash Player 6 (a part of the functions is not supported).
- ▷ The settings of the [Internet Browser] is added into [Settings] -> [Connection Settings]
- ▷ The audio contents from channels in the [RSS Channel] section now can be saved into your memory stick.
- ▷ [Auto] option added to [Rate Change] in [Location Free Player].
- ▷ Added file extension to playable AAC format.
- ▷ You can simply put a JPEG file in the same folder as the music, creating the art for the playlist.
- ▷ Added [Enable Flash Player] in [System Settings].
  - ▷ To change this option, you need to connect to the Internet

- ▷ "Simplified Chinese" and "Traditional Chinese" added to [System Settings] -> [System Language].
- ▷ Added [RSS Channel Settings].
- ▷ Added [UMD Video L & R Button] into [Video Settings].
- ▷ Fixed some issues when using a memory stick with more than 2GB free space.

29.3.9.2 new modules `amctrl.prx`

```
avcodec.prx
game_install_plugin.prx
iofilemgr_dnas.prx
irda.prx
mm_flash.prx
psheet.prx
usbacc.prx
usbcam.prx
usbgps.prx
usbgps_serial.prx
usbmic.prx
usbpspcm.prx
video_main_plugin.prx
```

29.3.10 2.71

29.3.11 2.8

[Network]

- ▷ In [RSS Channel], the download function for animation contents and image contents is now supported.
- ▷ In [Location Free Player], it is now possible to login via wireless LAN access point.

[Music]

- ▷ AAC files in ".3gp" extension can now be played.

[Misc]

- ▷ Supporting saving to "MUSIC", "PICTURE" and "VIDEO" folders in "Memory Stick Duo".
- ▷ Adding the next downloadable game demo to the "Memory Stick Duo".

29.3.12 2.81

29.3.13 2.82

- ▷ Ability to play Flash content in the Internet Browser (Connection to internet required for license)
- ▷ Connection Settings added under Settings in Internet Browser
- ▷ Ability to save content added in RSS to Memory Stick
- ▷ Automatic has been added under Rate in Location Free Player
- ▷ UMD Video L/R button added under Video Settings in Settings
- ▷ Ability to disable Chapter Skip feature of the L/R Buttons (UMD Video)
- ▷ New playable extension - AAC

- ▷ Simplified Chinese, Tradition Chinese added as new languages
- ▷ RSS channel Settings added in Settings
- ▷ Demos can now be downloaded from Browser and saved on Memory Stick
- ▷ Video output can now be displayed correctly when an external tuner is selected in Location Free Player
- ▷ Ability to download Video and Image content under RSS Channel
- ▷ Ability to register devices via a wireless LAN access point under Location Free Player
- ▷ Ability to play AAC files with .3gp extension under Music
- ▷ Ability to play content saved in MUSIC, PICTURE and VIDEO folders on a Memory Stick
- ▷ Added security strengthening revisions.

29.3.14 3.0

- ▷ Remote Play - Remote play is a new feature in Firmware 3.00 that allows you to remotely control your PlayStation 3 from your PSP. This also includes the display of PS3 content on the PSP. "You can display a PLAYSTATIONő3 system screen on a PSP system and play content that is on the PS3 system. To use this feature, you must adjust the necessary settings on the PSP system and the PS3 system." Using this new mode of playback, one can control the Photo, Music, Video, and Internet Browser features of the PlayStation 3 from a remote location via their Playstation Portable.
- ▷ Video Compatibility - In this updated version of the Playstation Portable firmware, you are also able to play a few new video formats. The Motion JPEG format (M-JPEG), is an "informal name for multimedia formats where each video frame or interlaced field of a digital video sequence is separately compressed as a JPEG image". The PlayStation Portable plays both the Linear PCM and the u-Law Versions of the Motion JPEG video format.
- ▷ In addition, you will now be able to access the Camera (functionality) from the photo option menus, for quicker easier access when taking photos or video.
- ▷ Another nifty function is the ability to finally turn off Auto Play for inserted UMD Discs via UMD Auto Boot.
- ▷ PlayStation Games - Here's the big tip you've been waiting for. Finally, Sony is going to drop their highly anticipated PlayStation One emulator onto the PSP. From the manual however, there seems to be a unavoidable catch. If you don't have a PS3, your not going to be enjoying PlayStation One games emulating on Sony's PlayStation One emulator for PSP anytime soon. From the manual it states that you must connect to the Playstation Online store with your PSP connected to the PlayStation 3 in order to download and play the games. In addition, they mention that you can in fact share the games, but you must activate the other system in the Friends menu as a PS3 Network Account.

29.3.15 3.01

- ▷ security fixes

29.3.16 3.02

29.3.17 3.03

29.4 Exploits

29.4.1 Kxploit (Code Execution)

found and Proof of Concept by: spanish PSPDEV team

29.4.1.1 Overview

29.4.1.2 Details

All kxploit does is create two directories, like this:

`/MYPROG%`
`/MYPROG`

or, to hide the 'broken data' items, like this:

`/MYPROG~1%` (exactly 8 characters including ~1)
`/MYPROG_____1` (exactly 32 characters)

The first contains an 'empty' PBP file (no actual executable) and the second the real unsigned binary. The PSP sees one as corrupt (and shows the corrupt icon) and one as valid. Once you launch the valid one, the PSP incorrectly parses the "%" sign as part of a standard printf-style formatting string, and so removes it, and then finds the elf file and loads it.
Memory stick swap works in the same way - it finds the pbp first on the first memory stick, and then finds the elf on the second after having run the pbp from the menu.

note: the filename hack to hide the broken icons has a subtle problem:

if you copy `MYPROG~1%` first:
`MYPROG~1` is the short name for `MYPROG~1%`
`MYPROG~2` is the short name for `MYPROG_____1`

if you copy `MYPROG_____1` first:
`MYPROG~1` is the short name for `MYPROG_____1`
`MYPROG~2` is the short name for `MYPROG~1%`

The second case works properly. The first does not. Remember why the kxploit trick works at all: the vsh sees a nicely formed file in "`MYPROG~1%`", but then passes "`MYPROG~1`" to the bootstrap, which executes the bare ELF. If "`MYPROG~1`" is the short name for the wrong directory, of course it won't work.

29.4.1.3 __SCE__ variant ("SCEKxploit")

a simelar bug can be exploited, name the two directories like this:

`%__SCE__MYPROG`
`__SCE__MYPROG`

this variation of the Kxploit has the advantage that it hides the corrupted icons without having the above mentioned subtle problem (since the shortened filenames of the two directories can not be confused).

29.4.2 TIFF Exploit (Code Execution)

found and Proof of Concept by: Niacin, Skylark
works in firmware version 2.0.

29.4.2.1 Overview

The exploit involves using a wallpaper and a TIFF image file containing a buffer overflow. Since the data from the wallpaper is in a known location(VRAM) we can use the TIFF overflow to jump to the known VRAM location and execute code.

29.4.2.2 Details

29.4.3 GTA Savegame Exploit (Code Execution)

found and Proof of Concept by: Edison Carter
works in firmware version 2.0 (required to run GTA) up to 2.6 (2.7 fixes the GTA exploit) .The Exploit was patched in a second batch of GTA.

German Version:

- ULES 00182 - Unpatched

Europe (UK/EU) Version:

- ULES-00151# - Unpatched - Contains fw 2.0 Update on UMD
- ULES-00151#2 - Patched - Contains the 2.60 update on UMD

North American (US) Version:

- ULUS 10041 - Patched - Contails UPDL 010050 on the UMD.
- ULUS 10041 - Unpatched - Contains UPDL 0048501A 5, plus IFPI L332 in very small letters.
- ULUS 10041 Unpatched, and Patched UMDs look exactly the same... Only the small codes are different.

Another slight variation that is also on the spine of the UMD case. The18 logo in a red circle is present in the pre 2.6 version, but in the patched 2.6 game the 18 red circle logo isn't present on the spine. Another indication is the copyright Date, if its 2005 then its unpatched, if its 2006 then its patched.

29.4.3.1 Overview

The GTA exploit is a classic stack buffer overflow, in the savedata processing.

29.4.3.2 Details

In essence, the savedata mostly consists of a large structure, with an element indicating the total size. GTA allocates a statically-sized buffer for this to be read into, on the stack - presumably using sizeof(savestruct) or similar. But it copies the number of bytes given by the .size element from the savedata into the stack buffer. By editing the .size element in the saved data, we can therefore force a buffer overflow. The .size element is at offset 0004 in the DATA.BIN file, in the savegame folder.

Note that the DATA.BIN is encrypted, so you need to use something like the savedata sample from the pspsdk in order to modify it.

29.4.4 LoadExec Exploit (gain Kernel access)

found and Proof of Concept by: Hitchhikr

works in firmware version 2.5 and 2.6

29.4.4.1 Overview

29.4.4.2 Details

The exploit is located in a function which can be found in the loadexec.prx file at address `0x88064C94` (game mode) in the firmware 2.6 (the same bug is also present in the firmware 2.5), a module located in the kernel space memory (therefore running in kernel mode).

The purpose of this procedure (used in other functions like "sceLoadExec") is to check that the drive part of a filename is valid & legit. It allocates 48 bytes of stack and the return address to the calling function is stored at the end of it (from 44th to 47th bytes).

It starts by checking the first char of the string to see if it's an empty drive name, if it's not, the routine extracts the part of the filename that contain the drive name and copies it into the allocated stack, it only stops when it encounters a ':' char.

Since it doesn't check any string length during the copy, if the drive name we supply is big enough it'll overwrite the rest of the stack based values, like the return address for example.

That's why a drive name of 48 chars (+ an extra ':' char to let the loop ends) containing an address to an arbitrary position in memory (pointing to a function of ours for example) located from the 44th to 47th chars in the filename will allow us to run any code we want in the context of the executing routine (kernel mode) as when it ends, it reloads the return address from the stack and directly jumps to it.

29.5 Network Update

When you select "Network Update" in the PSP menu, it will fetch a file from the web, this file currently has the following contents:

- Japanese PSP (JP Region)

  fetches from `http://fj01.psp.update.playstation.org/update/jp/psp-updatelist.txt`

```
# JP
Dest=00;ImageVersion=00000000;CDN=http://dj01.psp.update.playstation.org/update/jp/
nodata;CDN_Timeout=30;
```

  or

```
# JP
Dest=00;ImageVersion=000002d5;CDN=http://dj01.psp.update.playstation.org/update/jp/
2005_0824_50c7032754835b588319c1a6c652cdc0/EBOOT.PBP;CDN_Timeout=30;
```

▷ American PSP (US Region)

fetches from `http://fj01.psp.update.playstation.org/update/us/psp-updatelist.txt`

```
# US
Dest=01;ImageVersion=000002d5;CDN=http://du01.psp.update.playstation.org/update/us/
2005_0824_50c7032754835b588319c1a6c652cdc0/EBOOT.PBP;CDN_Timeout=30;
```

▷ European PSP (EU Region)

fetches from `http://fj01.psp.update.playstation.org/update/eu/psp-updatelist.txt`

```
# EU
Dest=02;ImageVersion=000002d5;CDN=http://de01.psp.update.playstation.org/update/eu/
2005_0824_50c7032754835b588319c1a6c652cdc0/EBOOT.PBP;CDN_Timeout=30;
```

If an image with a higher version than what is currently installed is available, the PSP can download it from the URL specified after CDN= and install it. The upgrade image consists of a game file in the PBP format, which should reflash the system software when run.

29.6 Network Test

In order for the PSP to check for updates, you must make sure you have valid Wi-Fi settings. In the "SETTINGS->Network Settings->Infrastructure Mode", if you selection the triangle button while the cursor is on a connection name, you can select the "Test Connection" and the PSP will actually try to reach this URL: http://fj00.psp.update.playstation.org/networktest/trial.txt

```
p
```

29.7 Registry

The PSP stores some non-critical settings (fonts, language, owners name, WEP passwords, user password) in a set of 2 files. Those files, named 'system.dreg' and 'system.ireg' can be called "the registry", not unlike the Windows one. Since the registry is placed on Flash1, it can be accessed by userland code in any version from 1.50 to 2.60.

For some reason (possibly wear leveling the Flash), the PSP registry is pretty awkwardly defined. Namely, the DREG part (data) consists of 512-byte sectors, not unlike hardware sectors on a hard disk. The IREG part (info) contains information on finding those sectors, since some blocks can be longer than 1 sector.

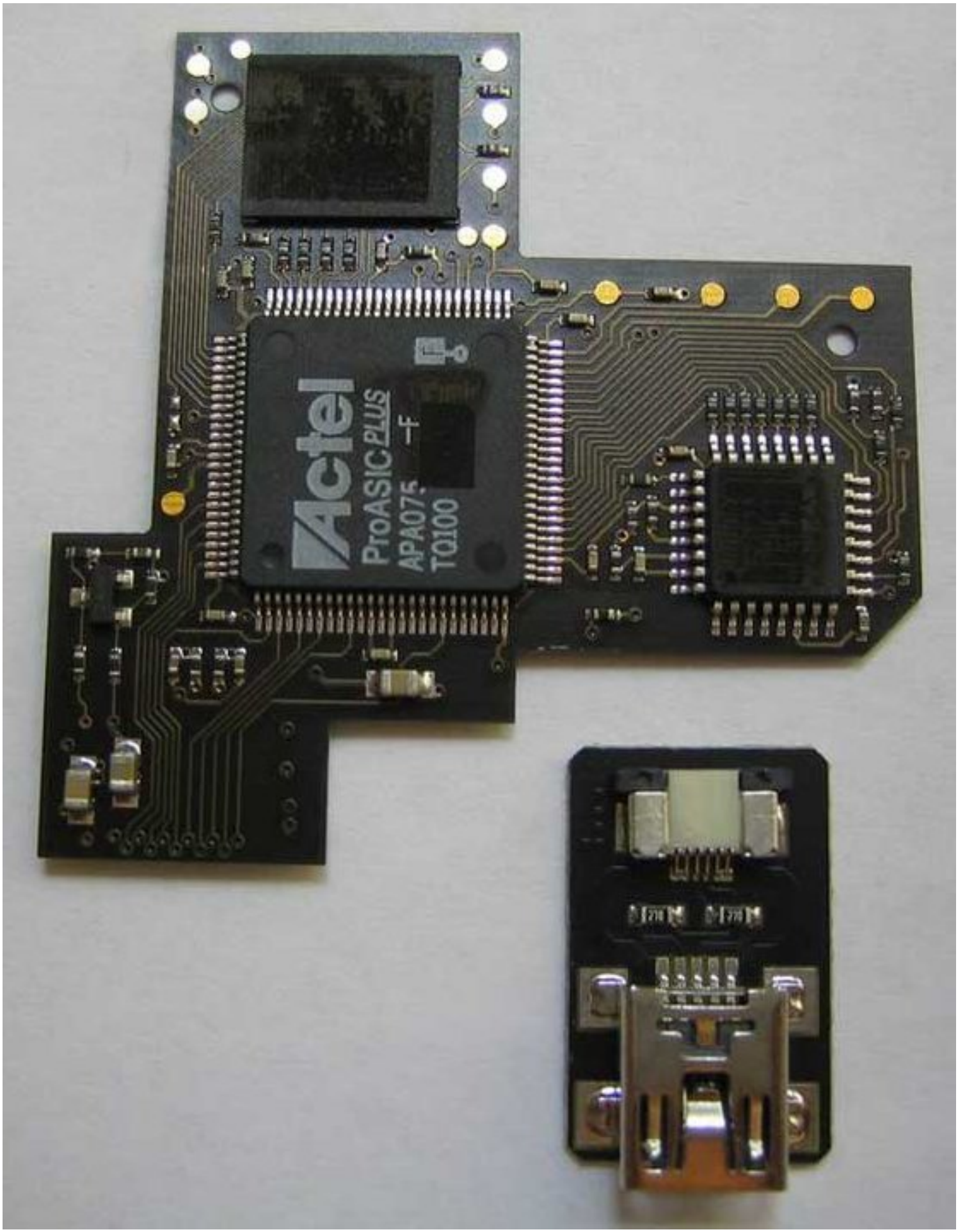
This is very similar to a filesystem - IREG part works as a "FAT" and DREG part works as the data area.

29.8 VSH

29.9 Game Sharing

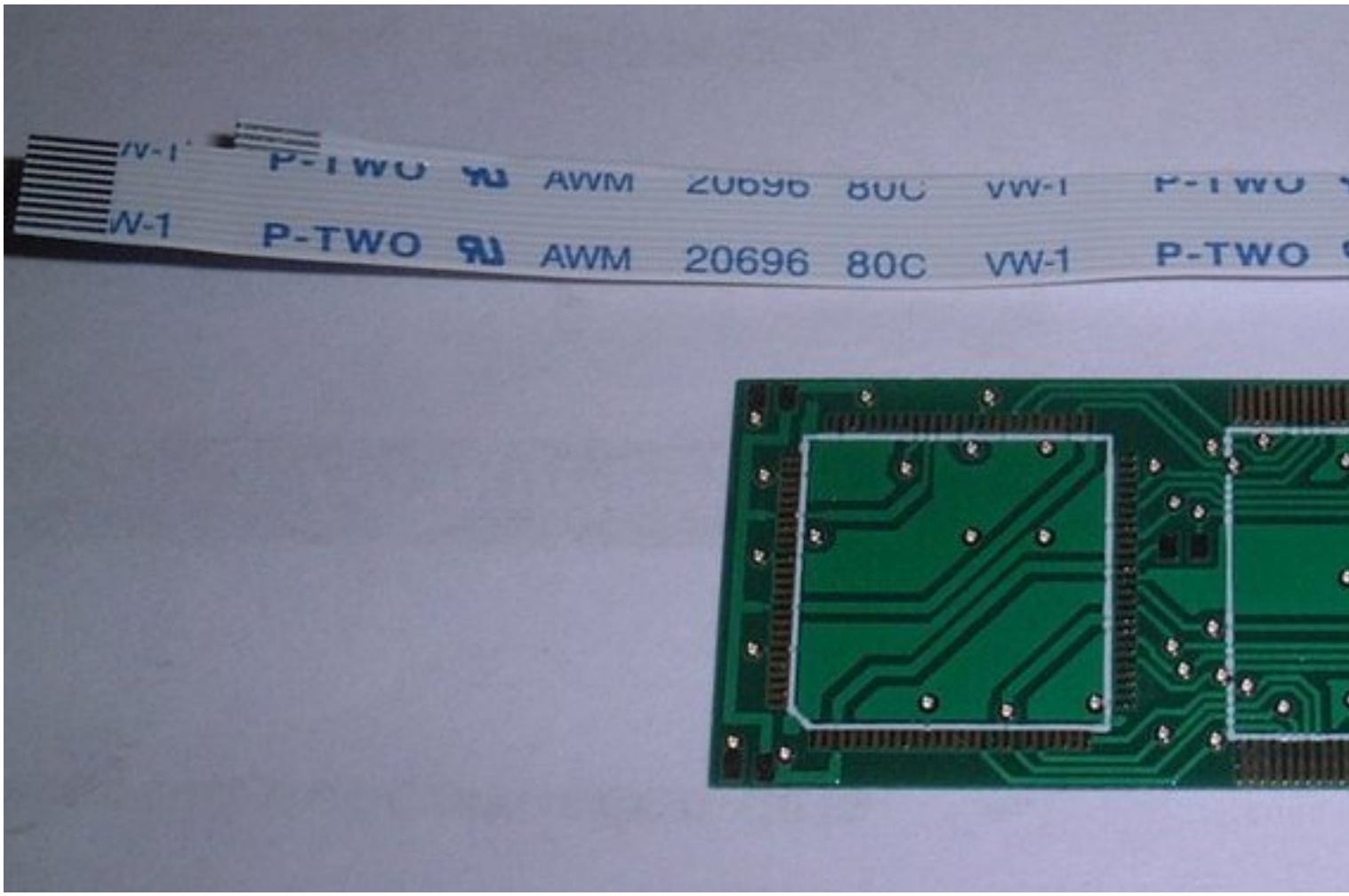
30 Modchips

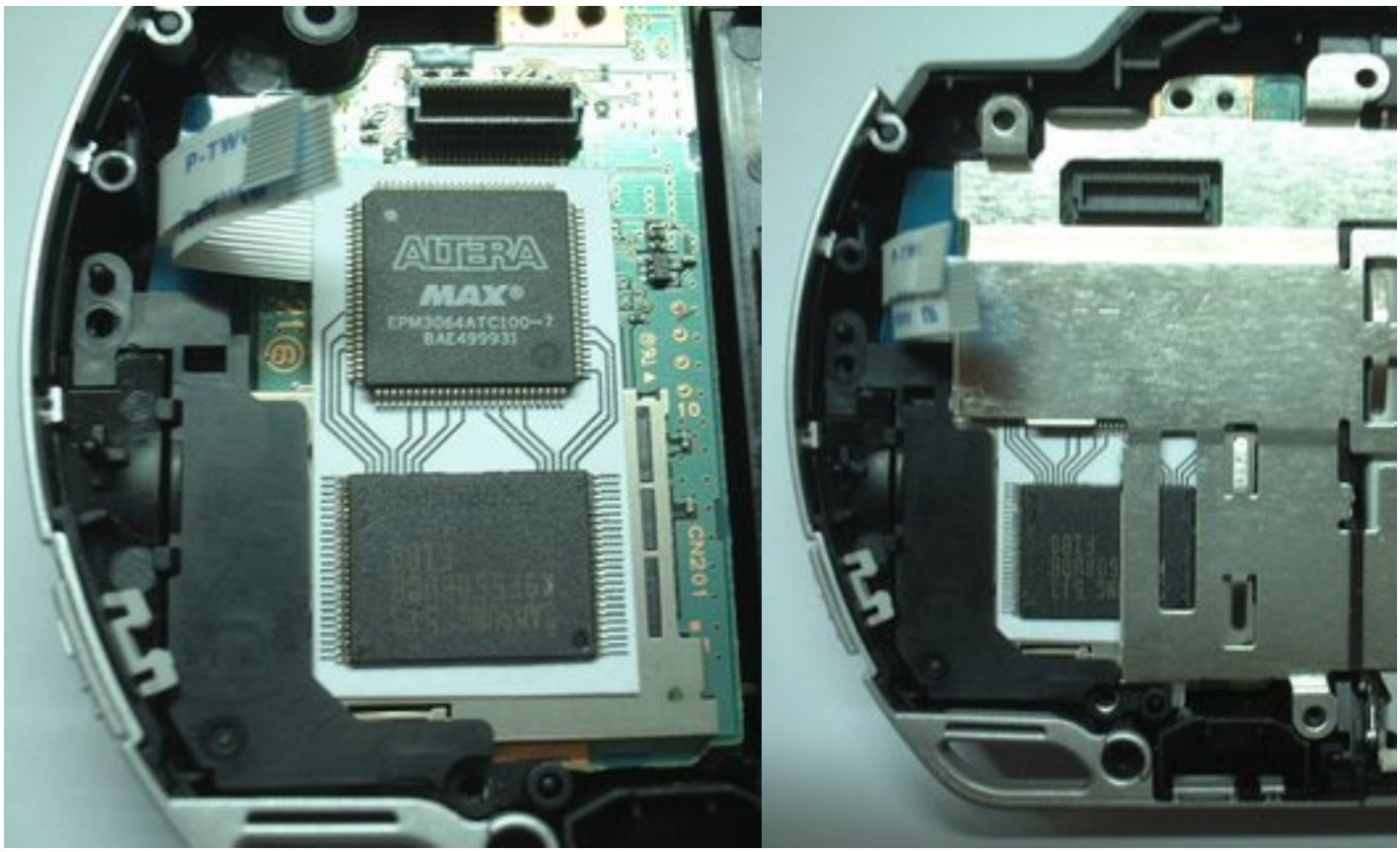
30.1 Undiluted Platinum (UP)



Developer: ???
Price: around 80 Euro
Features (unconfirmed):

30.2 0okm's Multi Firmware Module (MFM)



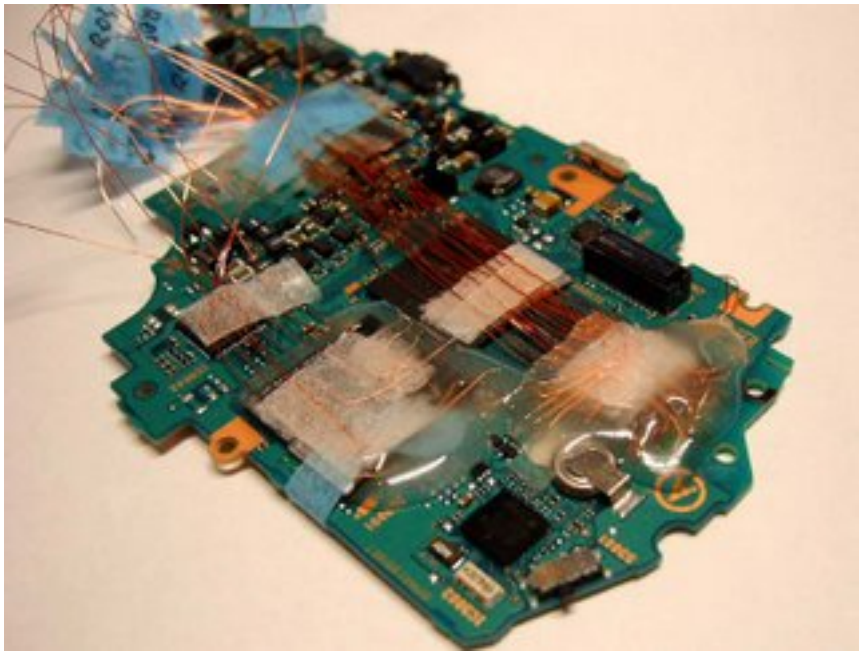


Developer: 0okm
Price: around 50USD (unconfirmed)
Features (unconfirmed):

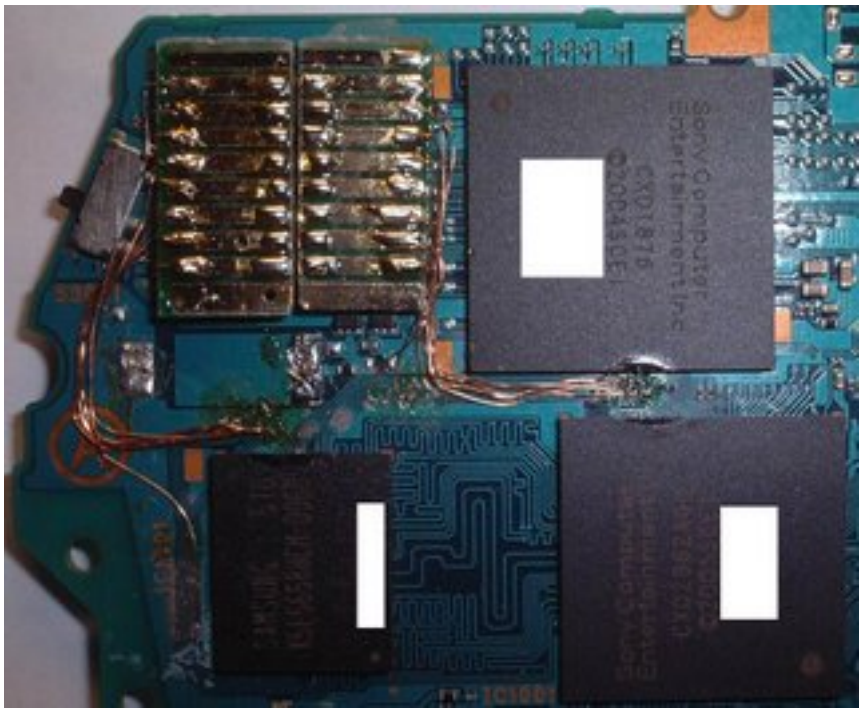
- ALTERA MAX 3000A Cost-Optimized CPLD
- 480 Mbps High Speed USB 2.0(PSP built-in)
- Brand New 32MB Nand Flash onboard (same type as used in PSP)
- Stable and reliable flashing software freely available for download
- PC EPP LPT Interface Adaptor(option)

30.3 Homemade Flash Interfaces

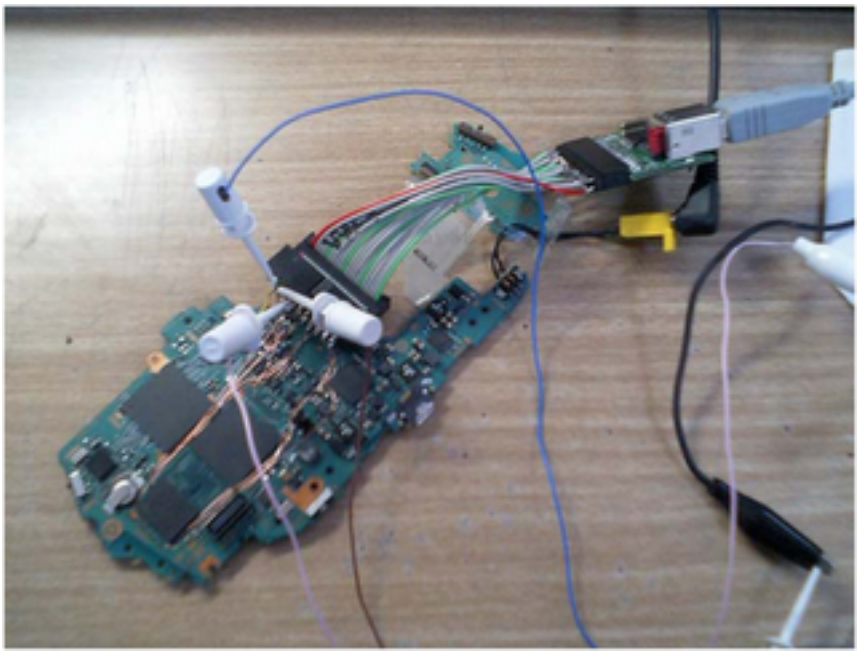
30.3.1 Nem



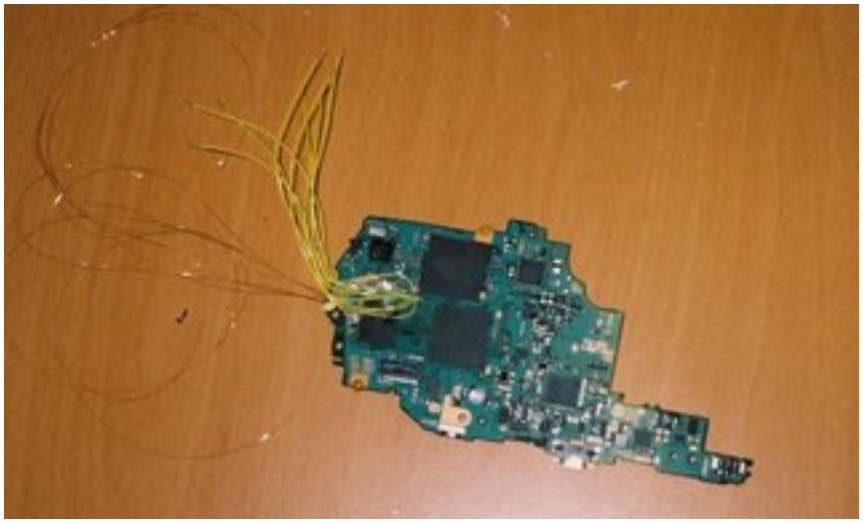
30.3.2 0okm



30.3.3 Booster



30.3.4 ryoko no usagi



31 Appendix

31.1 GCC Quick How To

note: the instructions in this chapter are only for dyhards that want to bootstrap their own GCC from vanilla sources. For everyone else a toolchain containing allegrex specific patches is highly recommended. For short: you dont need this :)

31.1.1 compile ASM to object:

```
<GCCROOT>/bin/???-elf-as -c \
-I <GCCROOT>/???-elf/include -I <additional includes> \
testasm.s -o testasm.o
```

31.1.2 compile C to object:

```
<GCCROOT>/bin/???-elf-gcc -c \
```

```
-I <GCCROOT>/???-elf/include -I <additional includes> \
-nostdlib testc.c -o testc.o
```

31.1.3 compile C++ to object:

```
<GCCROOT>/bin/???-elf-g++ -c \
-I <GCCROOT>/???-elf/include -I <additional includes> \
-nostdlib -fno-exceptions testcpp.cpp -o testcpp.o
```

31.1.4 link objects

```
<GCCROOT>/bin/???-elf-ld -T mips-pspbin.x -o test.elf crt0.o \
<GCCROOT>/lib/gcc-lib/???-elf/3.3/crtbegin.o \
<GCCROOT>/lib/gcc-lib/???-elf/3.3/crtend.o \
testasm.o testc.o testcpp.o -lg -lstdc++ -lm -lc -lnosys
```

you only need to link against crtbegin.o/crtend.o if you are using c++, and you only need -lg,-lstdc++,-lc,-lm if you are actually using these libraries (of course:)). however if you do so, linking against -lnosys as well is essential.

31.1.5 remove unneeded sections (debug info etc) from object

```
<GCCROOT>/bin/???-elf-strip -s test.elf
```

31.1.6 convert object to plain binary

```
<GCCROOT>/bin/???-elf-objcopy -O binary test.elf test.bin
```

31.1.7 convert absolute address into filename/line number/function

compile with "-g" flag, then use

```
<GCCROOT>/bin/???-elf-addr2line -f -e test.elf <address>
```

31.1.8 Building a Crosscompiler

configure options:

```
--target=misel-elf
--with-cpu=r4000
--disable-threads
--enable-languages=c
--disable-shared
--disable-nls
--with-newlib
```

note: a specialised 'allegrex' port is highly recommended. r4000 (or r5900) will work, but is suboptimal

31.1.9 Linker Script

to do

31.1.10 Startup Code

to do

31.2 Games

- ▷ AC Formula Front - FromSoftware
- ▷ Ape Escape - SCEJ
- ▷ Axel Impact - Axis Entertainment Inc.
- ▷ Bomberman - Hudson Soft
- ▷ BBG - SEED9
- ▷ Darkstalkers Chronicle - Capcom
- ▷ Derby - SCEJ
- ▷ Detective Adventure Jinguji - WorkJam
- ▷ Devil May Cry - Capcom
- ▷ DoraSlot - Dorasu Corp.
- ▷ Dokodemo Issho - SCEJ
- ▷ Dynasty Warriors - Koei
- ▷ The Evil Village - Now Production
- ▷ Far East of Eden - Hudson Soft
- ▷ The Gagharv - Bandai
- ▷ Ghost in the Shell Stand Alone Complex - SCEJ
- ▷ Gran Turismo 4 Mobile - SCEJ
- ▷ - Marvelous Interactive
- ▷ Harvest Moon - Marvelous Interactive
- ▷ Hot Shots Golf (AKA Everybody's Golf) - SCEJ
- ▷ Kollon - CyberFront Corp.
- ▷ Legend of River King - Marvelous Interactive
- ▷ License of Intelligence - Now Production
- ▷ - Marvelous Interactive
- ▷ Mah-Jong Fight Club - Konami
- ▷ Mah-jong Mate - Success Corp
- ▷ Mahjong - Koei
- ▷ Makai Wars - Nippon Ichi Software
- ▷ Metal Gear Acid - Konami
- ▷ Mobile Suit Gundam - Bandai
- ▷ Moji-Pittan - Namco
- ▷ Monkey Games - SCEJ
- ▷ Need For Speed Underground - EA
- ▷ New Ridge Racer - Namco
- ▷ Pilot Academy - Marvelous Interactive
- ▷ Popolocrois Story - SCEJ

- ▷ Powerful Proyakyu - Konami
- ▷ Pro-wrestling - Yuke's
- ▷ Project S - Sega
- ▷ Puyo Pop Fever - Sega
- ▷ Puzzle Bobble - Taito
- ▷ RS Revolution - Spike
- ▷ - Hudson Soft
- ▷ Romance of the Three Kingdoms - Koei
- ▷ Shintenmakai - Idea Factory
- ▷ - Marvelous Interactive
- ▷ Shutkou Battle - Genki
- ▷ Super Robot Wars - Banpresto
- ▷ TGM-K - Akira
- ▷ T.O.E. - Namco
- ▷ Talkman - SCEJ
- ▷ Techniccute - Akira
- ▷ Ten No Kagi, Chi No Mon - SCEJ
- ▷ Tiger Woods PGA Tour - EA
- ▷ Viewtiful Joe - Capcom
- ▷ Vulcanus Online - Zepetto Studios
- ▷ Winning Eleven (aka Pro-Evolution Soccer) - Konami
- ▷ Ys VI - The Ark of Napishtim - Konami

31.3 Developers

- ▷ FromSoftware
- ▷ Axis Entertainment Inc.
- ▷ SEED9
- ▷ WorkJam
- ▷ Capcom
- ▷ Dorasu Corp.
- ▷ CyberFront Corp.
- ▷ Now Production
- ▷ Success Corp
- ▷ Nippon Ichi Software
- ▷ Bandai
- ▷ Yuke's
- ▷ Sega
- ▷ Taito

- ▷ Spike
- ▷ Hudson Soft
- ▷ Koei
- ▷ Idea Factory
- ▷ Marvelous Interactive
- ▷ Genki
- ▷ Banpresto
- ▷ Namco
- ▷ Akira
- ▷ SCEJ
- ▷ EA
- ▷ Zepetto Studios
- ▷ Konami
- ▷ UBI Soft

32 References

- ▷ U.S. Pat. 6,817,021 (Disk device and guide member)
- ▷ U.S. Pat. 6,345,747 (Strap Assembly)
- ▷ U.S. Pat. Application 20040266529 (Methods and systems for remote execution of game content and presentation on a wireless portable device) - PS3 to PSP connection
- ▷ U.S. Pat. Design D517,552 (Keyboard)
- ▷ U.S. Pat. Design D516,080 (Keyboard)
- ▷ Debug Information in 'Puzzle Bobble' (Error Codes, Kernel API Names etc...)
- ▷ WM8750 Datasheet
- ▷ Libertas 88W8000G/88W8510 Datasheet
- ▷ MIPS R4000 Microprocessor User Manual
- ▷ NEC Vr5432 Microprocessor User Manual (Debug Registers)
- ▷ Samsung Memory and Storage Product Selection Guide
- ▷ Samsung Multi Chip Package Product Codes
- ▷ ECMA Standard 365 (UMD Specification)
- ▷ K4X56163PE-L(F)G Datasheet (16M x16 Mobile DDR SDRAM)
- ▷ K9F5608U0B Datasheet (32M x 8 Bit NAND Flash Memory)

32.1 Sources

- ▷ http://www.uspto.gov
- ▷ http://www.mips.com
- ▷ http://www.sdmi.org
- ▷ http://www.sony.com
- ▷ http://www.sony.net
- ▷ http://www.lik-sang.com/psp.html
- ▷ http://www.chipworks.com
- ▷ http://www.extremetech.com
- ▷ http://www.rsasecurity.com
- ▷ http://pinouts.ru
- ▷ http://www.edcheung.com/automa/sircs.htm
- ▷ http://www.hifi-remote.com/sony/
- ▷ http://www.ecma-international.org

33 Credits

besides freely available datasheets and patents, this document was created based on information provided by the following people. if you think you are missing in this list, please keep me informed so i can add you immediately.

Marcus Comstedt (http://mc.pp.se/psp/)	Memstick Layout, PBP and PSF Format, Network update, some other misc stuff
Loser, MrBrown (ps2dev forums)	Kernel Devices
Chip, Neovangelist (pspsdk)	GE Register Names / Commands
Jihad (http://www.hitmen-console.org)	Hardware Addresses
Darkfader (darkfader.net)	Hardware Part Numbers
psp-wiki contributors (http://www.pspbrew.com/wiki/)	misc stuff
MrBrown, Tyranid (pspsdk)	Hardware Profiler Info
Tyranid (pspsdk)	SIO Register Info
Skywalker, Xor37h (http://www.hitmen-console.org)	PSPInside Programming, Kernel Hacking
crazyc (ps2dev forums)	ME Info
Chip	texture swizzling
Holger, MrMr, John Kelley (ps2dev forums)	VFPU instruction Info
Tyranid	PRX Format Info
nem	Flash Info
Dr.Vegetable	Flash Info, Hardware Pics
Skylark, FreePlay, TeamOverload	System Registry and Font Info
Florin Sasu	Hardware Register Info
Jeremy Fitzhardinge	Cache HowTo

note: various other info was taken from various other people/posts from ps2dev forum. i don't remember them all, bear with me :) let me know if you feel you should be credited for something specific and i'll add it. Some more credits can also be found in the changelog file.

moreover, many thanks must go to everyone who helped making this document more consistant and error free by proofreading and pointing out mistakes, in particular Skywalker, Jihad, xor37h, Tyranid, bri3d ...