

Распределение адресного пространства приставки 3DO:

0x0000 0000 ... 0x000F FFFF	DRAM1 Область динамической памяти используемой процессором и DMA контроллером Размер 1 Мбайт
0x0010 0000 ... 0x001F FFFF	DRAM2 Область динамической памяти используемой процессором и DMA контроллером Размер 1 Мбайт
0x0020 0000 ... 0x002F FFFF	VRAM Процессор видит эту память как обычную 32-х битную память, в которой может также располагаться и код программ. Видео контроллер и сопроцессоры рассматривают эту память как две независимые 16-ти битные области, причем деление производится по шине данных - младшие 16 бит относятся к одной области, старшие к другой. Такое необычное деление связано с методикой увеличения разрешения - экстраполяцией реального разрешения 320x240 в телевизионное 640x480.
0x0030 0000 ... 0x02FF FFFF	RESERVED В обычных приставках это не занятая область, но это пространство предусматривалось для возможности расширения как DRAM, так VRAM памяти. Судя по тестам, общий объём памяти мог бы достигать 16 Мбайт
0x0300 0000 ... 0x030F FFFF	ROM Здесь располагается стандартное ПЗУ приставки размером в 1 Мбайт. В нём располагается стартовый код, программы тестирования, ядро ОС (которое, впрочем, есть на каждом 3DO CD-ROM'е), а также гот-диск на котором расположены остальные файлы операционной системы, а также программы, работу которых видит каждый пользователь приставки при включении (анимационная заставка, аудио и Kodak CD проигрыватель, менеджер энергонезависимой памяти). В это же пространство памяти вместо ROM может быть спроецировано содержимое ROM2, которое обычно содержит единственный файл шрифтов - иероглифы (эти ПЗУ присутствуют только в японских и азиатских приставках) Переключение банков ПЗУ осуществляется с помощью регистра CLIO 0x3400084 (более подробно в функции sub_302438C, получающая параметр в R0 - номер банка 0 или 1)
0x0310 0000 ... 0x0313 FFFF	RESERVED?
0x0314 0000 ... 0x0315 FFFF	NVRAM Энергонезависимое ОЗУ объёмом 32Кбайт, однако за счет размещения каждого байта по границе слова (связанно с тем, что ARM60 не имеет логики преобразования размерности шин) занимаемый объём в адресном пространстве процессора составляет 256Кбайт. Для хранения данных в NVRAM используется несколько упрощенная файловая система ОС Опера
0x0316 0000 ... 0x0317 FFFF	NVRAM копия 0x0314 0000 ... 0x0315 FFFF
0x0318 0000 ... 0x031F FFFF???	Возможно порт использует для отладки (diagport?) По его содержимому при старте решается необходимо запускать тесты или нет (запускаются если после сброса четыре последовательных чтения в младшем бите дают последовательность 1 0 1 0
0x0320 0000 ... 0x032F FFFF	FLASH VRAM или SPORT Через эти адреса осуществляется ускоренное (раз в 100) заполнение, либо копирование областей VRAM, Используются особенности используемых микросхем VRAM.
0x0330 0000 ... 0x033F FFFF	MADAM чип, реально регистров всего около 2Кбайт
0x0340 0000 ... 0x034F FFFF	CLIO чип, реально регистров всего около 64Кбайтов. Сам CLIO+DSP занимает пространство 0x0000...0x3FFF, от 0xC000 до 0xFFFF располагается окно для FMV (предварительная информация)
0x0350 0000 ... 0x036F FFFF	RESERVED?
0x0370 0000 ... 0x0373 FFFF	DEBUG ROM 256K? ROM disk с адреса 0x0370 1000???
0x0374 0000 ... 0x0377 FFFF	DEBUG RAM 256K?

VRAM:

VRAM используемая в 3DO двухпортовая - первый порт используется для обменов с процессором и видео процессорами, второй используется для отображения кадрового буфера на ТВ или приёма информации с внешнего декодера MPEG-1 (FMV). Это довольно упрощенная картина того, как реализовано в действительности.

Одна из самых интересных особенностей VRAM в 3DO - это способность копирования и заливки определённым цветом страниц VRAM (каждая размером 512 32-х битных слов) со скоростью нескольких обращений процессора к специальным участкам памяти. Заливка страницы NumOfPage осуществляется при помощи предварительной загрузки внутреннего регистра цвета VRAM по адресу 0x3202000, а затем записи маски по адресу [0x3204000 | NumOfPage] (т.е. NumOfPage=AnyAddressOfVramPage>>9), где маска указывает установленными в "1" бит какие биты необходимо установить в состояние регистра цвета. Таким образом, с помощью двух команд записи процессора могут быть одновременно заполнены 2048 байт VRAM. Другое название этого режима - FLASH write.

Если мгновенное заполнение страниц VRAM может выполняться в любой момент времени, то копирование страниц VRAM может осуществляться только тогда, когда второй порт VRAM не используется для вывода кадрового буфера на экран ТВ или ввода изображения с FMV модуля - для этого предусмотрен специальный интервал времени в течении вертикального кадрового импульса, где содержимое VCNT регистра находится между 10 и 13 (MINSPORTTVCOUNT и MAXSPORTTVCOUNT). Считывание слова процессором по адресу (0x3200000 | NumOfPage1) приводит к копированию выбранной страницы NumOfPage1 в во внутренний буфер VRAM (SAM буфер), за запись маски по адресу (0x3200000 | NumOfPage2) заполняет выбранную страницу NumOfPage2 VRAM содержимым SAM буфера с учетом маски.

В testbin встречается (0x3021814) чтение 0x3206100, 0x3206900. Наверное это конфигурирование VRAM...

Детали можно выяснить по datasheet чипа Toshiba TC528267, патенту WO 95/12876, тестам в BIOS 3DO, по драйверу SPORT расположенном в файле operator? операционной системы (BLOCK2 BIOS).

Взаимодействие пользовательских программ с ОС Орега осуществляется с помощью:

- o Вызова пользовательских функций через таблицу указателей на них (`_KernelBase` и прочие).
Особенность этих функций - выполнение в среде пользовательской задачи и ограничение доступа к ресурсам приставки
- o Программные прерывания, т.е. вызов функции по её номеру заданному в команде SWI процессора ARM60. Эти функции выполняются с правами супервизора и имеют полный доступ к аппаратуре.

```

0x00101  extern void Debug(void);

#define KERNELSWI  0x10000
0x10000  Item CreateSizedItem(int32 ctype,TagArg *p,int32 size);
0x10001  extern int32 WaitSignal(uint32 sigMask);
0x10002  extern Err SendSignal(Item task,uint32 sigMask);
0x10003  Err DeleteItem(Item i);
0x10004  Item FindItem(int32 ctype,TagArg *tp);
0x10005  Item OpenItem(Item foundItem, void *args);
0x10006  Err UnlockItem(Item s);
0x10006  Err UnlockSemaphore(Item s);
0x10007  int32 LockItem(Item s,uint32 flags);
0x10007  int32 LockSemaphore(Item s,uint32 flags);
0x10008  Err CloseItem(Item i);
0x10009  extern void Yield(void);
0x1000a  int32 SetItemPri(Item i,uint8 newpri);
0x1000b  illegal???
0x1000c  illegal???
0x1000d  void *AllocMemBlocks(int32 size, uint32 typebits);
0x1000e  extern void kprintf(char *fmt, ...);
0x1000f  extern Item GetThisMsg(Item msg);
0x10010  extern Err SendMsg(Item mp,Item msg, const void *dataptr, int32 datasize);
0x10010  extern Err SendSmallMsg(Item mp,Item msg, uint32 val1, uint32 val2);
0x10011  uint32 ReadHardwareRandomNumber(void);
0x10012  extern Err ReplyMsg(Item msg, int32 result, const void *dataptr, int32 datasize);
0x10012  extern Err ReplySmallMsg(Item msg, int32 result, uint32 val1, uint32 val2);
0x10013  extern Item GetMsg(Item mp);
0x10014  Err ControlMem(void *p, int32 size, int32 cmd, Item task);
0x10015  extern int32 AllocSignal(uint32 sigMask);
0x10016  extern Err FreeSignal(uint32 sigMask);
0x10017  extern void *SetFunction(Item, int32 vnum, int32 vtype, void *newfunc);
0x10018  Err SendIO(Item ior, const IOInfo *ioiP);
0x10019  Err AbortIO(Item ior);
0x1001a  dbugtrigger???
0x1001b  bcopy???
0x1001c  Err SetItemOwner(Item i,Item newOwner);
0x1001d
0x1001e  extern int MayGetChar(void);
0x1001f
0x10020
0x10021  int32 SystemScavengeMem(void);
0x10022  void CompleteIO(IOReq *ior);
0x10023
0x10024  Item FindAndOpenItem(int32 ctype,TagArg *tp);
0x10025  Err DoIO(Item ior, const IOInfo *ioiP);
0x10026  uint32 SampleSystemTime(void);
0x10027  extern Err SetExitStatus(int32 status);
0x10028  extern Item WaitPort(Item mp,Item msg);
0x10029  Err WaitIO(Item ior);

#define GRAFSWI  0x20000
0x20001  Err SetReadAddress( Item bitmapItem, ubyte *buffer, int32 width );
0x20002  Err ResetReadAddress( Item bitmapItem );
0x20003  Err SetClipOrigin( Item bitmapItem, int32 x, int32 y );
0x20004  grNULROUTINE
0x20005  Err EnableVAVG( Item screenItem );
0x20006  Err DisableVAVG( Item screenItem );
0x20007  Err EnableHAVG( Item screenItem );

```

```

0x20008 Err DisableHVG( Item screenItem );
0x20009 Err SetScreenColor( Item screenItem, uint32 colorEntry );
0x2000A Err ResetScreenColors( Item screenItem );
0x2000B grNULROUTINE
0x2000C grNULROUTINE
0x2000D Err SetScreenColors( Item screenItem, uint32 *entries, int32 count );
0x2000E swiSuperResetCurrentFont( ??? )
0x2000F grNULROUTINE
0x20010 grNULROUTINE
0x20011 Err AddScreenGroup( Item screenGroup, TagArg *targs );
0x20012 Err RemoveScreenGroup( Item screenGroup );
0x20013 Err SetClipWidth( Item bitmapItem, int32 clipWidth );
0x20014 Err SetClipHeight( Item bitmapItem, int32 clipHeight );
0x20015 grNULROUTINE
0x20016 grNULROUTINE
0x20017 Err DrawScreenCels( Item screenItem, CCB *ccb);
0x20018 grNULROUTINE
0x20019 grNULROUTINE
0x2001A grNULROUTINE
0x2001B swiSuperOpenRAMFont( ??? )
0x2001C grNULROUTINE
0x2001D Err DrawText16( GrafCon *gcon, Item bitmapItem, uint16 *text );
0x2001E grNULROUTINE
0x2001F Err DrawChar( GrafCon *gcon, Item bitmapItem, uint32 character );
0x20020 grNULROUTINE
0x20021 Err DrawTo( Item bitmapItem, GrafCon *grafcon, Coord x, Coord y );
0x20022 grNULROUTINE
0x20023 Err FillRect( Item bitmapItem, GrafCon *gc, Rect *r );
0x20024 Err SetCurrentFontCCB( CCB *ccb );
0x20025 Font *GetCurrentFont( void );
0x20026 Err DrawText8( GrafCon *gcon, Item bitmapItem, uint8 *text );
0x20027 Err DrawCels ( Item bitmapItem, CCB *ccb);
0x20028 grNULROUTINE
0x20029 Err SetCEControl( Item bitmapItem, int32 controlWord, int32 controlMask );
0x2002A Err SetCEWatchDog( Item bitmapItem, int32 db_ctr );
0x2002B grNULROUTINE
0x2002C grNULROUTINE
0x2002D Err DisplayScreen( Item screenItem0, Item screenItem1 );
0x2002E grNULROUTINE
0x2002F Err SetVDL( Item screenItem, Item vdlItem );
0x20030 grNULROUTINE
0x20031 grNULROUTINE
0x20032 realCreateScreenGroup( ??? )
0x20033 Err ModifyVDL ( Item vdlItem, TagArg* vdlTags);
0x20034 _SETVBLATTRS_
0x20035 _GETVBLATTRS_

#define FILEFOLIOSWI 0x30000
0x30000 extern Item OpenDiskFile(char *path);
0x30001 extern int32 CloseDiskFile( Item fileItem);
0x30002
0x30003
0x30004 extern Item MountFileSystem( Item deviceItem, int32 unit, uint32 blockOffset);
0x30005 extern Item OpenDiskFileInDir( Item dirlItem, char *path);
0x30006 extern Item MountMacFileSystem(char *path);
0x30007 extern Item ChangeDirectory(char *path);
0x30008 extern Item GetDirectory(char *pathBuf, int pathBufLen);
0x30009 extern Item CreateFile(char *path);

```

```

0x3000a extern Err DeleteFile(char *path);
0x3000b extern Item CreateAlias(char *aliasPath, char *realPath);
0x3000c extern int32 LoadOverlay(void *mumble);
0x3000d extern Err DismountFileSystem(char *name);

#define AUDIOSWI 0x40000

0x40000 Err TweakKnob(Item KnobItem, int32 Value);
0x40001 Err StartInstrument(Item Instrument, TagArg *TagList);
0x40002 Err ReleaseInstrument(Item Instrument, TagArg *TagList);
0x40003 Err StopInstrument(Item Instrument, TagArg *TagList);
0x40004 Err TuneInsTemplate(Item InsTemplate, Item Tuning);
0x40005 Err TuneInstrument(Item Instrument, Item Tuning);
0x40006 Err ReservedAudioSWI1(void);
0x40007 Err TestHack(TagArg *args);
0x40008 Err ConnectInstruments(Item SrcIns, char *SrcName, Item DstIns, char *DstName);
0x40009 uint32 TraceAudio(int32 Mask);
0x4000a int32 AllocAmplitude(int32 Amplitude);
0x4000b Err FreeAmplitude(int32 Amplitude);
0x4000c Err DisconnectInstruments(Item SrcIns, char *SrcName, Item DstIns, char *DstName);
0x4000d Err SignalAtTime(Item Cue, AudioTime Time);
0x4000f Err SetAudioRate(Item Owner, frac16 Rate);
0x40010 Err SetAudioDuration(Item Owner, uint32 Frames);
0x40011 Err TweakRawKnob(Item KnobItem, int32 Value);
0x40012 Err StartAttachment(Item Attachment, TagArg *tp);
0x40013 Err ReleaseAttachment(Item Attachment, TagArg *tp);
0x40014 Err StopAttachment(Item Attachment, TagArg *tp);
0x40015 Err LinkAttachments(Item At1, Item At2);
0x40016 Err MonitorAttachment(Item Attachment, Item Cue, int32 CueAt);
0x40017 Err SetMasterTuning(frac16 Frequency);
0x40018 Err AbandonInstrument(Item Instrument);
0x40019 Item AdoptInstrument(Item InsTemplate);
0x4001a Item ScavengeInstrument(Item InsTemplate, uint8 Priority, int32 MaxActivity, int32 IfSystemWide);
0x4001b Err SetAudioItemInfo(Item AnyItem, TagArg *tp);
0x4001c Err PauseInstrument(Item Instrument);
0x4001d Err ResumeInstrument(Item Instrument);
0x4001e int32 WhereAttachment(Item Attachment);
0x4001f uint32 IncrementGlobalIndex(void);
0x40020 Err BendInstrumentPitch(Item Instrument, frac16 BendFrac);
0x40021 Err AbortTimerCue(Item Cue);
0x40022 Err EnableAudioInput(int32 OnOrOff, TagArg *Tags);
0x40023
0x40024 Err ReadProbe(Item Probe, int32 *ValuePtr);
0x40025
0x40026 uint16 GetAudioFrameCount(void);
0x40027 int32 GetAudioCyclesUsed(void);

#define MATHSWI 0x50000

0x50000 void MulVec3Mat33_F16(vec3f16 dest, vec3f16 vec, mat33f16 mat);
0x50001 void MulMat33Mat33_F16(mat33f16 dest, mat33f16 src1, mat33f16 src2);
0x50002 void MulManyVec3Mat33_F16(vec3f16 *dest, vec3f16 *src, mat33f16 mat, int32 count);
0x50003 void MulObjectVec3Mat33_F16(void *objectlist[], ObjOffset1 *offsetstruct, int32 count);
0x50004 void MulObjectMat33_F16(void *objectlist[], ObjOffset2 *offsetstruct, mat33f16 mat, int32 count);
0x50005 void MulManyF16(frac16 *dest, frac16 *src1, frac16 *src2, int32 count);
0x50006 void MulScalarF16(frac16 *dest, frac16 *src, frac16 scalar, int32 count);
0x50007 void MulVec4Mat44_F16(vec4f16 dest, vec4f16 vec, mat44f16 mat);
0x50008 void MulMat44Mat44_F16(mat44f16 dest, mat44f16 src1, mat44f16 src2);
0x50009 void MulManyVec4Mat44_F16(vec4f16 *dest, vec4f16 *src, mat44f16 mat, int32 count);
0x5000a void MulObjectVec4Mat44_F16(void *objectlist[], ObjOffset1 *offsetstruct, int32 count);

```

```
0x5000b void MulObjectMat44_F16(void *objectlist[], ObjOffset2 *offsetstruct, mat44f16 mat, int32 count);
0x5000c frac16 Dot3_F16(vec3f16 v1, vec3f16 v2);
0x5000d frac16 Dot4_F16(vec4f16 v1, vec4f16 v2);
0x5000e void Cross3_F16(vec3f16 dest, vec3f16 v1, vec3f16 v2);
0x5000f frac16 AbsVec3_F16(vec3f16 vec);
0x50010 frac16 AbsVec4_F16(vec4f16 vec);
0x50011 void MulVec3Mat33DivZ_F16(vec3f16 dest, vec3f16 vec, mat33f16 mat, frac16 n);
0x50012 void MulManyVec3Mat33DivZ_F16(mmv3m33d *s);
```

На микросхеме MADAM выполнено управление DRAM и VRAM, реализованы многоканальный DMA контроллер и сопроцессор с фиксированной запятой

Использует для своих целей 2Кбайта адресного пространства процессора начиная с адреса 0x3300000

		by read	by write
0x0330 0000	По чтению даёт версию MADAM и указывает реализованные возможности За записи возможно печать во вне символа для протоколирования (для отладки)???	constant	log kprintf

Определяет конфигурацию памяти, младшие два бита определяют размер VRAM (1M/2M), бит номер два всегда 0?, затем два бита размер DRAM1 (0/1/4/16M) и ещё два бита размер DRAM2 (0/1/4/16M). Обычно здесь должно быть 0x29 для всех выпущенных на рынок приставок. В таблице ниже первый байт конфигурация памяти, второй общий объём памяти для этого случая:

```
ROM:03000370 1A 10      byte_3000370 DCB 0x1A, 0x10      ; DATA XREF: sub_30002C4+70
ROM:03000372 19 10      DCB 0x19, 16
ROM:03000374 52 0A      DCB 0x52, 10
ROM:03000376 51 09      DCB 0x51, 9
ROM:03000378 4A 07      DCB 0x4A, 7
ROM:0300037A 32 07      DCB 0x32, 7
ROM:0300037C 49 06      DCB 0x49, 6
ROM:0300037E 31 06      DCB 0x31, 6
ROM:03000380 42 06      DCB 0x42, 6
ROM:03000382 12 06      DCB 0x12, 6
ROM:03000384 41 05      DCB 0x41, 5
ROM:03000386 11 05      DCB 0x11, 5
ROM:03000388 2A 04      DCB 0x2A, 4
ROM:0300038A 22 03      DCB 0x22, 3
ROM:0300038C 0A 03      DCB 0xA, 3
ROM:0300038E 29 03      DCB 0x29, 3
ROM:03000390 21 02      DCB 0x21, 2
ROM:03000392 09 02      DCB 9, 2
ROM:03000394 02 02      DCB 2, 2
ROM:03000396 01 01      DCB 1, 1
ROM:03000398 00 00      DCB 0, 0
```

0x0330 0004		reg	w=0x29
-------------	--	-----	--------

Разрешение каналов DMA, каждый бит разрешает соответствующий канал DMA (точное соответствие пока не ясно до конца). Название регистра MCTL???

0x0330 0008	Разрешение канала 1 S-Port Read Transfers 0x2000->*0x03300008 VDL ??? CLUTXEN???	???	handle
-------------	--	-----	--------

Разрешение канала 2 S-Port Read Transfers 0x4000->*0x03300008 Line Video Address??? VSCTXEN?

Разрешение канала Player Bus Data DMA 0x8000->*0x03300008

0x0330 000C	??? (в testbin пишется 0x178906 - делитель что-ли или управляющее слово)	reg	reg
-------------	--	-----	-----

AbortInfo (AbortBits) -- описание с чего это вдруг MADAM выставил сигнал abort процессору. Под каждый вид такого исключения отводится отдельный бит:

```
DCB "ABORT 0:ROMF" ",0" "Rom Failure"
DCB "ABORT 1:ROMW" ",0" "Rom Write"
DCB "ABORT 2:CLIOT" ",0" "CLIO timeout"
DCB "ABORT 3:HARDU" ",0" "User access to hardware"
DCB "ABORT 4:SYSRAMU",0 "User access to SYSRAM"
DCB "ABORT 5:FENCEV",0 "Fance Violation" (Memory Protect error)
DCB "ABORT 6:VPR" ",0" "Virtual Page Error"
DCB "ABORT 7:R26E" ",0" "Out of 26 bit address range"
DCB "ABORT 8:SPSC" ",0" "SPORT while SC"
DCB "ABORT 9:BITE" ",0" "Byte Access to hardware"
DCB "ABORT 10:BADDEC",0 "**A detect"
DCB "ABORT 11:ARPS" ",0" "ARM to Regis/PIP while SIP"
DCB "ABORT 12:BWACC",0 "Byte address, Word access"
DCB "ABORT **: ",0 "**This abort is most likely from a 'lowmem abort' card."
**The actual offending instruction may be up to two"
**instructions before the indicated PC."
```

0x0330 0020		reg	reg
-------------	--	-----	-----

DMA Priv Violation (PrivBits) - Это регистр описывает причину выставления бита ошибки при прерывании от DMA контроллера

		reg	reg
--	--	-----	-----

бит 0: 0x001 DMA access to SYSRAM

бит 1: 0x002 SPORT access to SYSRAM (здесь SPORT - VDL DMA канал)

0x0330 0024	бит 2: 0x004 REGIS access to SYSRAM
-------------	-------------------------------------

бит 3: 0x008 DMA access over VRAMSIZE

бит 4: 0x010 SPORT access over VRAMSIZE (здесь SPORT - VDL DMA канал)

бит 5: 0x020 REGIS access over VRAMSIZE

бит 6: 0x040 REGIS math failure

0x0330 0028	DrawCels(): while(*STATbits & SPRON) { , где SPRON=0x10 DrawCels(): if(*STATbits & SPRPAU) { , где SPRPAU=0x20	reg	R/O
-------------	---	-----	-----

Управление VDP, смотри патент US5,572,235 TABLE 1.8
TABLE 1.8

Spryte Render Stopping and Starting

The Spryte rendering process is started from the CPU by writing a meaningless value to the 'SPRSTRT' address in the memory address manipulator chip (MAMC). This sets the 'Spryte-ON' flipflop. Writing to SPRSTART while the 'Spryte-ON' flipflop is already on will have no effect at all. BUT DON'T DO IT. The race condition of Spryte just now ending and you just now writing is not preventable.

A start is always a 'cold' start. Previously running Spryte things have been canceled.

It is required that the software has setup the data in memory and the first SCoB address in the DMA stack correctly. The SYSTEM WILL CRASH if the Spryte data or the first SCoB

address are faulty.

The Spryte process is stopped cold when the CPU writes to `SPRSTOP`. All intermediate states of the engines have been RESET. The data pointers are now WRONG. Spryte processing is dead. Except, of course, for the SPRPAUS flipflop.

When the Spryte engine is completely finished with all of its functions including sending the last of the rendered data to memory, it will reset its Spryte-ON flipflop.

Spryte processing is not overlapped in the hardware. One Spryte is allowed to totally complete prior to starting the next Spryte in the list. This total completion includes the processing of trailing transparent pixels and the outputting of the last data values to memory.

The CPU can request that the Spryte rendering engine stop operating at the end of the current Spryte but not reset itself by writing to `SPRPAUS`.

This will set a `SPRPAUS` flipflop that will cause the Spryte rendering engine to set its `PAUSE` flipflop at the end of the current Spryte. When the PAUSE flipflop is set by this mechanism (not by other means), the SPRPAUS flipflop will be cleared.

The SPRPAUS event is only a one shot thing. Obviously this can only be done if the CPU already has access to the system bus. SPRPAUS can be set by the CPU even if the Spryte engine is off.

When an interrupt is present, the PAUSE flipflop is held in a `set` condition.

This will allow the Spryte rendering engine to pause cleanly and allow the CPU to notice the interrupt.

The CPU can reset the PAUSE flipflop by writing to `SPRCNTU`. If the Spryte engine was on but paused it will now continue. If not, it won't. This is also when the CPU might avail itself of the SPRPAUS function.

Once a render process has started, the CPU is effectively asleep.

Nothing has actually been done to the CPU, it just can't get any cycles on the system bus. When the CPU finally does get some cycles, it needs to decide why and service the situation. The reasons it got a cycle are that an interrupt is present, or that the Spryte engine is paused or done, or somehow, the bus has temporarily become available. After servicing the interrupt or CPU requested Spryte pause, it will be up to the CPU to then decide whether or not to continue the render process. The CPU can determine the status of the render process by reading the appropriate status bit(s).

If the Spryte rendering engine is on but not paused, the CPU should just sit in its status bit check loop. This is a temporary situation and will soon change.

If the engine is on and paused, the CPU could just issue the SPRCNTU. If there are no Sprytes left, the Spryte engine will turn itself off and no harm done.

If on and paused and at end of current Spryte, then the CPU can use the Spryte rendering engine for its purpose without fear of damaging a Spryte in process.

> END OF TABLE 1.8 <

0x0330 0100	DrawCels(): *SPRSTRT = 0; /* GO */	w/o	handle
0x0330 0104	DrawCels(): *SPRSTOP = 0;	w/o	handle
0x0330 0108	DrawCels(): *SPRCNTU = 0;	w/o	handle
0x0330 010C	DrawCels(): *SPRPAUS = 0; /* If timed out, issue pause request */	w/o	handle

0x0330 0110	<p>Подробности в US5,596,693: Method for controlling a spryte rendering processor</p> <p>DrawCels(): *CCBCTL0 = bitmap->bm_CECtrl; // General spryte-rendering engine control word</p>	reg	reg
0x0330 0130	<p>Подробности в US5,596,693: Method for controlling a spryte rendering processor</p> <p>DrawCels(): *REGCTL0 = bitmap->bm_REGCTL0; //Controls the modulo for reading source frame buffer data into the primary and/or secondary input port of the spryte engine and for writing spryte image result data from the spryte engine into a destination frame buffer in VRAM. The modulo effectively indicates the number of pixels per scan line as represented in the respective frame buffer in VRAM.</p>	reg	reg
0x0330 0134	<p>DrawCels(): *REGCTL1 = bitmap->bm_REGCTL1; // X and Y clip values, effectively indicating the number of pixels in the X and Y dimensions which make up the frame buffer. Bits 26:16 indicate the last writeable row (counting from row 0) in the Ydimension and bits 10:0 indicate the last writeable column (counting from col. 0) in the X dimension. All other bits must be zero. As an example, a value of 00EF013F indicates that the frame buffer data is represented in 320x240 format.</p>	reg	reg
0x0330 0138	<p>DrawCels(): *REGCTL2 = bitmap->bm_REGCTL2; // Read base address. Indicates the address in VRAM of the upper left corner pixel of the source frame buffer data.</p>	reg	reg
0x0330 013C	<p>DrawCels(): *REGCTL3 = bitmap->bm_REGCTL3; // Write base address. Indicates the address in VRAM of the upper left corner pixel of the destination frame buffer (CFBD).</p>	reg	reg

FANCE control (механизм защиты памяти)

0x0330 0230	0L	?reg	?reg
0x0330 0234	0R	?reg	?reg
0x0330 0238	1L	?reg	?reg
0x0330 023C	1R	?reg	?reg
0x0330 0270	2L	?reg	?reg
0x0330 0274	2R	?reg	?reg
0x0330 0278	3L	?reg	?reg
0x0330 027C	3R	?reg	?reg

Блок адресов 0x3300400..0x33005FF используется DMA контроллером, по 4 слова на каждый канал (обычно)

Sound CPUram2DSP DMA group

по 4-ре слова [текущий (адрес, длина), следующий (адрес, длина)]

0x0330 0400	CPUram2DSP.0	reg	reg
0x0330 0410	CPUram2DSP.1	reg	reg
0x0330 0420	CPUram2DSP.2	reg	reg
0x0330 0430	CPUram2DSP.3	reg	reg
0x0330 0440	CPUram2DSP.4	reg	reg
0x0330 0450	CPUram2DSP.5	reg	reg
0x0330 0460	CPUram2DSP.6	reg	reg
0x0330 0470	CPUram2DSP.7	reg	reg
0x0330 0480	CPUram2DSP.8	reg	reg
0x0330 0490	CPUram2DSP.9	reg	reg
0x0330 04A0	CPUram2DSP.10	reg	reg
0x0330 04B0	CPUram2DSP.11	reg	reg
0x0330 04C0	CPUram2DSP.12	reg	reg

FMV DMA group1

0x0330 04D0		?reg	?reg
0x0330 04D4		?reg	?reg
0x0330 04E0		?reg	?reg
0x0330 04E4		?reg	?reg

XBUS DMA group

0x0330 0540	XBUS DMA: адрес куда или откуда	reg	reg
0x0330 0544	XBUS DMA: сколько байт скопировать	reg	reg

FMV DMA group2

0x0330 0550		?reg	?reg
0x0330 0554		?reg	?reg
0x0330 0560		?reg	?reg
0x0330 0564		?reg	?reg

Player Bus Data DMA group

0x0330 0570	PlayBus DMA: адрес куда	reg	reg
0x0330 0574	PlayBus DMA: сколько байт копировать (если младшие четыре байта 0xFFFC, т.е. -4, то сбор закончен)	reg	reg
0x0330 0578	PlayBus DMA: адрес откуда	reg	reg

S-Port Read Transfers DMA group

0x0330 0580	Current CLUT address Адрес начала списка VDL, длина не указывается DMA контроллеру!	reg	reg
0x0330 0584	Next CLUT address	reg	reg
0x0330 0588	CLUT Mid-Line Address	reg	reg
0x0330 058C	- - -		
0x0330 0590	Previous Line Video Address	reg	reg
0x0330 0594	Current Line Video Address	reg	reg
0x0330 0598	Previous Line Mid-Line Address	reg	reg
0x0330 059C	Current Line Mid-Line Address	reg	reg

An 8-register DMA grouping for spryte control (look United States Patent 5.596.693)

0x0330 05A0	0 CURRENT SCOB ADDRESS	reg	reg
0x0330 05A4	1 NEXT SCOB ADDRESS DrawCels(): *NEXTPTR = (ulong)ccb;	reg	reg
0x0330 05A8	2 PIP ADDRESS	reg	reg
0x0330 05AC	3 SPRYTE DATA ADDRESS	reg	reg
0x0330 05B0	4 ENGINE A FETCH ADDRESS	reg	reg
0x0330 05B4	5 ENGINE A LENGTH	reg	reg
0x0330 05B8	6 ENGINE B FETCH ADDRESS	reg	reg
0x0330 05BC	7 ENGINE B LENGTH	reg	reg

Блок адресов 0x3300600..0x33007FF используется сопроцессором

Сопроцессор вычисляет результат перемножения двух матриц 4x4 (похоже можно задать, чтобы вычисления производились с матрицами меньшего размера - для скорейшего получения результата)

		by read	by write
0x0340 0000	Старший байт версия кристалла, остальные указывают на то, какие возможности реализованы...	const	r/o
0x0340 0004		?reg	?reg
0x0340 0008		?reg	?reg
0x0340 000C		?reg	?reg
0x0340 0010		?reg	?reg
0x0340 0014		?reg	?reg
0x0340 0018		?reg	?reg
0x0340 001C		?reg	?reg
0x0340 0020	<p>Описывает причину сброса консоли. Запись нуля сбрасывает установленные биты. Опрашиваются в программах биты 0,1,6. Похоже на то, что установка битов 0,1 свидетельствует о нормальной причине сброса (холодной). 6-й же бит похоже указывает на то, что перезагрузка тёплая, при этом startbin перегружает только BLOCK0 - возникает это возможно при смене диска или при срабатывании watchdog - не ясно пока...</p> <p>Также сюда пишут 0x30 вовремя кадрового гасящего импульса чтобы добиться "Sort reset for Uncle", после записи этого значения в коде поставили 15 команд NOP.</p>	?reg	?reg
0x0340 0024		?reg	?reg
0x0340 0028		reg	handle
0x0340 002C		reg	reg
0x0340 0030		reg	reg
0x0340 0034		reg	reg
0x0340 0038		reg	reg
0x0340 003C		reg	reg
0x0340 0040	<p>При чтении считываются состояния бит источников прерываний (флажки), по записи биты равные 1 устанавливают соответствующие бит флажков прерываний в 1. Видимо так можно симитировать программно аппаратное прерывание.</p> <p>Источники прерываний с бОльшими номерами более приоритетные (реализуется программно в обработчике прерываний).</p> <p>VINT0 идет каждый четный полукадр, а VINT1 -- каждый нечетный (это самое начало гасящего импульса).</p> <p>бит 00 - VINT0</p> <p>бит 01 - VINT1 (сюда вешается VSyncTimerFirg, ControlPort, SPORTfirg, GraphicsFirg)</p> <p>бит 02 - EXINT (прерывание от устройств на XBUS, т.е. например от CDROM)</p> <p>бит 03: Timer0.F Прерывания от таймеров, возможны только от нечётных (старших в парах)</p> <p>бит 04: Timer0.D</p> <p>бит 05: Timer0.B</p> <p>бит 06: Timer0.9</p> <p>бит 07: Timer0.7</p> <p>бит 08: Timer0.5</p> <p>бит 09: Timer0.3</p> <p>бит 10: Timer0.1</p> <p>бит 11: AudioTimer</p> <p>бит 12: AudioDMA_DSPtoRAM0</p> <p>бит 13: AudioDMA_DSPtoRAM1</p> <p>бит 14: AudioDMA_DSPtoRAM2</p> <p>бит 15: AudioDMA_DSPtoRAM3</p> <p>бит 16: AudioDMA_DSPfromRAM0</p> <p>бит 17: AudioDMA_DSPfromRAM1</p> <p>бит 18: AudioDMA_DSPfromRAM2</p> <p>бит 19: AudioDMA_DSPfromRAM3</p> <p>бит 20: AudioDMA_DSPfromRAM4</p> <p>бит 21: AudioDMA_DSPfromRAM5</p> <p>бит 22: AudioDMA_DSPfromRAM6</p> <p>бит 23: AudioDMA_DSPfromRAM7</p> <p>бит 24: AudioDMA_DSPfromRAM8</p> <p>бит 25: AudioDMA_DSPfromRAM9</p> <p>бит 26: AudioDMA_DSPfromRAM10</p> <p>бит 27: AudioDMA_DSPfromRAM11</p> <p>бит 28: AudioDMA_DSPfromRAM12</p> <p>бит 29: XBUS DMA transfer complite</p> <p>бит 30: ??? Пустой обработчик - возможно даже watchdog (если это прерывание разрешено и повторно пришло, а предыдущее не было обработано, то сброс, а?)</p> <p>бит 31 - Указывает, что есть ещё прерывания в регистре 0x0340 0060</p>	handle	handle
0x0340 0044	При чтении считываются состояния бит источников прерываний (флажки), по записи биты равные 1 сбрасывают соответствующие бит флажков прерываний в 0.	handle	handle
0x0340 0048	По чтению выдаётся маска прерываний, по записи биты равные 1 устанавливают соответствующие биты маски прерываний в 1.	handle	handle
0x0340 004C	По чтению выдаётся маска прерываний, по записи биты равные 1 сбрасывают соответствующие биты маски прерываний в 0.	handle	handle
0x0340 0050	0x1 0000 << n Принудительный останов DSP DMA канала???	w/o	handle

Этот регистр описывает причину возникновения прерывания от контроллера DMA: биты 0..14 показывают видимо по каким каналам возникла ситуация UnderFlow бит 15 - неизвестно биты 16..19 показывают видимо по каким каналам возникла ситуация OverFlow биты 20 - указывает, что возникли проблемы с правами доступа, подробности расписанные в регистре MADAM 0x330 0024 бит 21 (DMA NFW) говорит о том, что биты 22..27 содержат какое-то значение addr				
0x0340 0058		reg	r/o	
<hr/>				
Продолжение регистра 0x0340 0040: бит 0 - #32: бит 1 - #33: бит 2 - #34: FMVVideo/FMVAudio (он же Woody) Прерывание от внешнего FMV модуля (Здесь также может быть прерывание от некого "ns Readers", видимо тоже подключаемого через FMV разъём расширения, - использует пространство адресов 0x31C xxxx) бит 3 - #35: бит 4 - #36: FMVVideoDMAfrRAM Прерывание от встроенного в 3DO DMA контроллера бит 5 - #37: FMVVideoDMAtoRAM бит 6 - #38: FMVAudioDMAfrRAM бит 7 - #39: FMVAudioDMAtoRAM бит 8 - #40: Прерывание от DMA контроллера				
0x0340 0060		handle	handle	
<hr/>				
0x0340 0064	аналогично 0x0340 0044	handle	handle	
0x0340 0068	аналогично 0x0340 0048	handle	handle	
0x0340 006C	аналогично 0x0340 004C	handle	handle	
<hr/>				
0x0340 0080	???	reg	reg	
В этом регистре чтобы изменить какой-либо из четырех младших бит требуется также устанавливать соответствующий бит разрешающий модификацию бита (это будет бит с номером на 4 больше). Т.е. Чтобы сбросить мл. бит не достаточно записать 0 в этот регистра, требуется записать 0x10, а чтобы установить это бит - 0x11. Для 1-го бита соответственно 0x20/0x22 и т.д.				
0x0340 0084	0-й бит: ??? PAL/NTSC селектор или что-то очень похожее на это... 1-й бит: Разрешением звука от DSP 2-й бит: определяет какой банк ПЗУ выбирается (при этом 6 бит разрешает изменение банка ПЗУ). Смотри функцию по адресу 302438C в ПЗУ. 3-й бит: возможно вход в режим HardReboot	reg	handle	
<hr/>				
<u>Hardware Timers</u> Аппаратура приставки содержит 16-ть 16-ти битных декрементирующих таймеров, из которых могут быть собраны тамера большей разрядности путём указания признака CASCADE для всех таймеров цепочки, кроме первого. В цепочке таймера с меньшими номерами являются младшими, т.е. только при переполнении таймера с меньшим номером производится декремент таймера с большим номером. Каждый нечетный таймер (мл. бит номера которого равен 1) может генерировать прерывания процессору - описания в регистре 0x340 004x). Каждый таймер конфигурируется 4-мя битами: бит 0: DECREMENT (похоже это просто разрешение работы этого таймера) бит 1: RELOAD - после переполения (перехода через ноль) взять новое значение из регистра перезагрузки (иначе загружается 0xFFFF, хотя скорее таймер просто останавливается, т.е. у него сбрасывается бит DECREMENT). бит 2: CASCADE - декремент этого таймера должен производиться при переполнении предыдущего таймера в цепочке (с номером на 1 меньшим). бит 3: FLABLODE - "всеми используется, но как назвать не знают". Я не встречал использования, так, что пока без комментариев. --- Все биты конфигурации хранятся в двух 32-х битных регистрах				
0x0340 0200	Через этот адрес производится установка бит конфигурации для таймеров 0..7 (младшие 4 бита соответствуют таймеру 0 и т.д.)	handle	handle	
0x0340 0204	Через этот адрес производится сброс бит конфигурации для таймеров 0..7	handle	handle	
0x0340 0208	Через этот адрес производится установка бит конфигурации для таймеров 0..7 (младшие 4 бита соответствуют таймеру 8 и т.д.)	handle	handle	
0x0340 020C	Через этот адрес производится сброс бит конфигурации для таймеров 8..F	handle	handle	
0x0340 0220	Возможно через этот регистр задаётся общая тактовая частота для таймеров (slack tick)	reg	reg	
<hr/>				
0x0340 0100	Начиная с этого адреса располагаются пары 16-ти битных регистров Timer0 выровненные по границе слов, по 8 байт на каждый таймер, первым идёт поле ht_cnt, второе очевидно называлось бы вроде как ht_reload. Поле ht_cnt будет уменьшатся до нуля при разрешенном таймера, а затем, в зависимости от режима, это поле может быть перегружено значением из ht_reload.	reg	reg	
...				
0x0340 017F	В патентах есть такой фрагмент "if(((HardTimer*)Timer0)[tid].ht_cnt == 0x0000) {"			

0x0340 0300	<p>Запись 0x0000 0001 запрещает CPUram2DSP.0 DMA пересылку Запись 0x0000 0002 запрещает CPUram2DSP.1 DMA пересылку Запись 0x0000 0004 запрещает CPUram2DSP.2 DMA пересылку Запись 0x0000 0008 запрещает CPUram2DSP.3 DMA пересылку Запись 0x0000 0010 запрещает CPUram2DSP.4 DMA пересылку Запись 0x0000 0020 запрещает CPUram2DSP.5 DMA пересылку Запись 0x0000 0040 запрещает CPUram2DSP.6 DMA пересылку Запись 0x0000 0080 запрещает CPUram2DSP.7 DMA пересылку Запись 0x0000 0100 запрещает CPUram2DSP.8 DMA пересылку Запись 0x0000 0200 запрещает CPUram2DSP.9 DMA пересылку Запись 0x0000 0400 запрещает CPUram2DSP.10 DMA пересылку Запись 0x0000 0800 запрещает CPUram2DSP.11 DMA пересылку Запись 0x0000 1000 запрещает CPUram2DSP.12 DMA пересылку</p> <p>Запись 0x0001 0000 запрещает DSP2CPUram.0 DMA пересылку Запись 0x0002 0000 запрещает DSP2CPUram.1 DMA пересылку Запись 0x0004 0000 запрещает DSP2CPUram.2 DMA пересылку Запись 0x0008 0000 запрещает DSP2CPUram.3 DMA пересылку</p>	handle	handle
0x0340 0304	<p>Запись 0x0000 0001 разрешает CPUram2DSP.0 DMA пересылку Запись 0x0000 0002 разрешает CPUram2DSP.1 DMA пересылку Запись 0x0000 0004 разрешает CPUram2DSP.2 DMA пересылку Запись 0x0000 0008 разрешает CPUram2DSP.3 DMA пересылку Запись 0x0000 0010 разрешает CPUram2DSP.4 DMA пересылку Запись 0x0000 0020 разрешает CPUram2DSP.5 DMA пересылку Запись 0x0000 0040 разрешает CPUram2DSP.6 DMA пересылку Запись 0x0000 0080 разрешает CPUram2DSP.7 DMA пересылку Запись 0x0000 0100 разрешает CPUram2DSP.8 DMA пересылку Запись 0x0000 0200 разрешает CPUram2DSP.9 DMA пересылку Запись 0x0000 0400 разрешает CPUram2DSP.10 DMA пересылку Запись 0x0000 0800 разрешает CPUram2DSP.11 DMA пересылку Запись 0x0000 1000 разрешает CPUram2DSP.12 DMA пересылку</p> <p>Запись 0x0001 0000 разрешает DSP2CPUram.0 DMA пересылку Запись 0x0002 0000 разрешает DSP2CPUram.1 DMA пересылку Запись 0x0004 0000 разрешает DSP2CPUram.2 DMA пересылку Запись 0x0008 0000 разрешает DSP2CPUram.3 DMA пересылку</p> <p>Запись 0x0010 0000 разрешает XBUS DMA пересылку По чтению даёт разрешенные DMA каналы в CLIO</p> <p>В этом и следующем регистре похоже вроде приостановки DMA или разрешение запуска второго набора регистров адреса или длины -- пока не ясно...</p>	handle	handle
0x0340 0308	<p>Старое -> Запись 0x0010 0000 запрещает XBUS DMA пересылку По чтению даёт разрешенные DMA каналы в CLIO</p>	?reg	?reg
0x0340 0400 ... 0x0340 04FF	<p>XBUS DMA управляющие слова... Первые 4 слова...</p> <p>0x0340 0400 - XBUS to RAM: (0x4000, а затем 0x800) RAM to XBUS: (0x4200, а затем 0x800) 0x0340 0404 - XBUS to RAM: 0x280 RAM to XBUS: 0x80 0x0340 0408 - ? 0x0340 040C - размер DMA обмена (дублирует 0x0330 0544, но на 4 больше)</p> <p>clio[0x400] = инициализация DMA 0x4000 = Init 0x200 = Direction (set = XB2RAM, clear = RAM2XB) 0x800 = Clean Ready Bit 0x80 = ReadyBit</p> <p>clio[0x404] = Command to XBUS controller 0x200 == Direction Set 0x80 = Wait for "Go" signal 0x40 = Abort Transfer 0x20 = Flush Cache 0x400 = Send 17*STB 0x100 = Wait for RDY signal from device 0x800 = Flush Status</p>	handle	handle
0x0340 0500 ... 0x0340 053F	XBUS по записи SELECTION по чтению -reserv-	handle	handle
0x0340 0540 ... 0x0340 057F	XBUS по записи WR_POL по чтению RD_POL	handle	handle
0x0340 0580 ... 0x0340 05BF	XBUS по записи WR_COM по чтению RD_STAT	handle	handle

0x0340 05C0 ...	XBUS по записи WR_DATA по чтению RD_DATA	handle	handle
0x0340 05FF			

Пространство DSP: 0x0340 1000 ... 0x340 3FFF

0x0340 1000 ...	Упраление DSP (чтение/запись)	reg	reg
0x0340 17FF			

0x0340 1800 ...	Память програм DSP - NRAM (доступ только по записи). В одном слове ARM - два слова DSP	reg	reg
0x0340 1BFF			
0x0340 1C00 ...	Похоже на резерв или алиас памяти программ 0x0340 1800 ... 0x340 1BFF No name. Made in Taiwan...	reg	reg
0x0340 1FFF			
0x0340 2000 ...	Память програм DSP - NRAM (доступ только по записи). В одном слове ARM - одно слово DSP	reg	reg
0x0340 27FF			
0x0340 2800 ...	Похоже на резерв или алиас памяти программ 0x0340 2000 ... 0x340 27FF	reg	reg
0x0340 2FFF			

0x0340 3000 ...	Внешние регистры DSP - EIRAM (доступ только по записи). В одном слове ARM - два слова DSP	reg	reg
0x0340 33FF			
0x0340 3400 ...	Внешние регистры DSP - EIRAM (доступ только по записи). В одном слове ARM - одно слова DSP	reg	reg
0x0340 37FF			

0x0340 3800 ...	Внешние регистры DSP - EORAM (доступ только по чтению). В одном слове ARM - два слова DSP	reg	reg
0x0340 3BFF			
0x0340 3C00 ...	Внешние регистры DSP - EORAM (доступ только по чтению). В одном слове ARM - одно слова DSP	reg	reg
0x0340 3FFF			

Описание некотрых регистров DSP

0x0340 17E8	Сброс DSP, запись нуля (наверное и любого другого значения) приводик к сбросу DSP По чтению 16-ти битный генератор случайных чисел (шума) Управление DSP: 0-Stop, 1-Start	handle	r/o
0x0340 17F0			
0x0340 17FC			

0x0340 1800 ... 0x0340 1BFF	Начиная с этого адреса располагается программа DSP - 256 слов (512 ячеек DSP), одно слово ARM два 16-ти битных слова DSP. 0x8380 - это инструкция NOP ;-) С большой вероятностью доступ write only.
-----------------------------	--

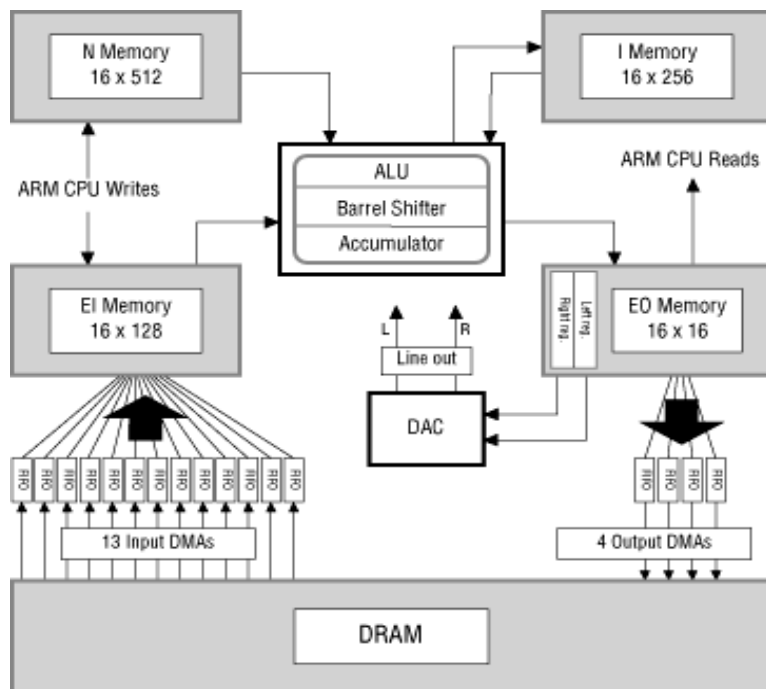
0x0340 3C00	Значение этого регистра возвращает функция swiGetAudioCyclesUsed (для DSP 0x300)
0x0340 3C08	Значение этого регистра возвращает функция swiGetAudioFrameCount (для DSP 0x302)

0x0340 3FB8	соответствует регистру 0x3EE DSP. Это значение передаваемое обработчику прерывания в ARM'e (используется для организации AudioTimer)
-------------	--

FMV регистры

0x0340 C000	gWoody
0x0340 C004	
0x0340 C008	
0x0340 C00C	
0x0340 C010	fmcvntls
0x0340 C014	fmcvntlc
0x0340 C018	
0x0340 C01C	
0x0340 C020	sizereg
0x0340 C028	vid1reg
0x0340 C02C	vid2reg

Краткое описание в /OnLineDoc/DevDocs/ppgflidr/mgsfldr/mpgflidr/02mpg002.html
Расположен в адресном пространстве CLIO



Уникальные коды ошибок выдаваемы операционной системой.

0xD57B9000	0xD556F000	no error
0xD57B9001	0xD556F001	Bad Item
0xD57B9002	0xD556F002	undefined tag
0xD57B9003	0xD556F003	bad value in tagarg list
0xD57B9004	0xD556F004	NotPrivileged
0xD57B9005	0xD556F005	Item with that name not found
0xD57B9006	0xD556F006	insufficient memory to complete
0xD57B9007	0xD556F007	Bad SubType for this Folio/Cmd
0xD57B9008	0xD556F008	System Software error: call NTG
0xD57B9009	0xD556F009	ptr/range is illegal for this task
0xD57B900A	0xD556F00A	operation aborted
0xD57B900B	0xD556F00B	bad unit in IOReq
0xD57B900C	0xD556F00C	bad cmd in IOReq
0xD57B900D	0xD556F00D	bad IOArgs in ioinfo
0xD57B900E	0xD556F00E	bad Name for Item
0xD57B900F	0xD556F00F	IO in progress for this IOReq
0xD57B9010	0xD556F010	Operation not supported at this time
0xD57B9011	0xD556F011	IO incomplete for this IOReq
0xD57B9012	0xD556F012	NotOwner
0xD57B9013	0xD556F013	Device offline
0xD57B9014	0xD556F014	Device error
0xD57B9015	0xD556F015	Medium error
0xD57B9016	0xD556F016	End of medium
0xD57B9017	0xD556F017	Illegal parameter(s)

Kernel Extended Error Table:

0xD57B9100	no error
0xD57B9101	Bad Type Specifier
0xD57B9102	Unknown SubType for this Folio
0xD57B9103	Resource Table OverFlow
0xD57B9104	DriverItem is not valid
0xD57B9105	Item is not opened by you
0xD57B9106	Item Table Full
0xD57B9107	BadMemCmd
0xD57B9108	Out of signals
0xD57B9109	Msg already sent / Bad State
0xD57B910A	NoReplyPort
0xD57B910B	size less than minimum
0xD57B910C	BadPriority
0xD57B910D	illegal operation for thread
0xD57B910E	BadQuanta
0xD57B910F	BadStackSize
0xD57B9110	NoTimer
0xD57B9111	Can't Open
0xD57B9112	RSA failed in CreateTask
0xD57B9113	No devices on XBUS
0xD57B9114	>15 devices on XBUS
0xD57B9115	Replyport required
0xD57B9116	Illegal signal
0xD57B9117	Bad Lock arg
0xD57B9118	Bad AIF header

0	Zorro!
0xD556F101	No such file
0xD556F102	Not a directory
0xD556F103	No such filesystem
0xD556F104	Illegal filename
0xD556F105	Medium I/O error
0xD556F106	Volume offline
0xD556F107	Hardware error
0xD556F108	Bad parameter
0xD556F109	Filesystem full
0xD556F10A	Filesystem damaged
0xD556F10B	File[system] busy

0xD556F10C
0xD556F10D

Duplicate filename
Read-only file[system]