



Document Number : ES446

VUB300 Interface Control Document

for internal product design
Copyright © 2009-2010 Elan Digital Systems Ltd

Prepared by : Bart, Ana, Piotr

Elan Digital Systems Ltd
 Fareham, Hampshire, UK
 Tel 44 1489 579799 – Fax 44 1489 577 516
www.elandigitalsystems.com

Issue Record :

Issue	Date	Author	Details
1	24/11/09	Ana	Initial release
2-8		Ana	Various updates
9	20/1/2010	Bart	Added Block count and size to CMD header to avoid integer division in the firmware
10	03/FEB/2010	Tony	Changed Error Codes to match those of the VUB200
11	12/FEB/2010	Tony	Add R7 for CMD8
12	16/02/10	Ana	Error codes 22-26 added; general update
13	04/03/10	Ana	Added Error code for overrun and PIO timeout
14	16/03/10	Ana	Little endian format for command structure changed
15	14/04/10	Ana	Added Vendor Specific Request for ROM wait states, 4 byte alignment, power control
16	19/04/2010	Bart	Added CCCR5 to interrupt packet
17	21/07/10	Ana	Offload Vendor Specific Requests added
18	10/08/10	Ana	Interrupt polling command response 0x03 0x08 0x00 when no interrupt detected
19	19/08/10	Tony	Added timeout to IRQ poll command
20	25/08/10	Tony	Added structure of PseudoCode execution results
22	30/09/10	Ana	Removed old interface reference, general update.
23	17/12/10	Tony	Release Version – removed project management details



Table of Contents

1	OVERVIEW.....	3
2	USB CONFIGURATION.....	3
	2.12.1Detecting the SD SDIO device.....	3
	2.22.2The control pipe.....	4
	2.2.1USB Device Configuration.....	4
	2.2.2Application Specific Setup Commands.....	4
	2.32.3Endpoint1 - EP1.....	10
	2.3.1VUB300 SD Command Header.....	10
	2.3.2VUB300 SD Response Header.....	13
	2.3.3Command Response Header	13
	2.3.4Interrupt Header.....	14
	2.3.5Error Header.....	14
	2.3.6Status Header.....	16
	2.3.7PseudoCode Execution Results.....	16
	2.42.4Endpoint 2 - EP2.....	16



1 Overview

VUB300 is USB to SDIO Host interface chip based on existing hardware, supplied to us by SMSC. Proposed ASIC is USB2240. There are hardware limitations that VUB300 inherits from USB2240, such as availability of just 3 USB channels. Control endpoint, interrupt (configurable to bulk) and one bulk in/out pair. VUB300 also inherits support for USB 2.0 high speed, via the Synopsis IP DesignWare core included in the chip.

SMSC have granted us the access to their firmware source code for USB2250, chip bigger in size (dimensions wise) than the proposed USB2240, but effectively with the same ASIC hardware design. During the conference call with SMSC Austin they have confirmed that endpoint 1 on USB2250 is exactly the same as endpoint 1 on USB2240. All the firmware development is done using SMSC development board (SMSC SVB-USB2250 Rev. B). Debugging is done using Proteus debug kit. It has been decided that it is good idea to keep most of the SMSC firmware code, including their kernel module structure.

There has been discussion about VUB300 chip supporting two or more SD/SDIO ports. The interface is designed in such a way that multiple SD/SDIO ports can be addressed, but please note VUB300 can only support one SD/SDIO port. We need extra bulk in/out pair per additional ports that we wish to support. These additional endpoints pairs are not available with USB2240. Possible part option for this project would be SMSC 2611, dual SD port chip.

SMSC are releasing USB2410 chip, with DMA engine supporting SD block sizes that are multiple of 4 (up to 2048). All other transfers are PIOed. This is the first ARM core processor from SMSC.

2 USB Configuration

VUB300 Elan driver will connect to the system SDIO driver and will link to USB host controller. VUB300 does not fit in any of the standard USB device classes and will be identified as a custom USB device supporting a single custom interface.

Vendor ID for VUB300 is VID = 2201 and Product ID is PID = 012C (which is 300 decimal).

Ids are documented in L:\CommonData\USB PIDs\PIDs for Elan products using Elan VID-0x2201.txt.

USB pipes

- Control pipe (Endpoint 0)
- Bulk In/Out (Endpoint 1)
- Bulk In/Out (Endpoint 2)

2.1 Detecting the SD SDIO device

The Windows operating system will not use SDIO stack when SD memory card is inserted into the SDIO slot. It could, but it appears to be inefficient in supporting SD memory mode.



VUB300 Driver will support both SD and SDIO devices.

2.2 The control pipe

The traffic on the control endpoint will perform twofold function: configuration and application specific.

2.2.1 USB Device Configuration

Control data on the control pipe will be used in the normal USB System Software way for configuration the device when first attached.

2.2.2 Application Specific Setup Commands

Also, control data on the control pipe will be used in application specific way.

N.B., there is possibility that Windows OS will limit bandwidth of the control pipe, to increase the system performance. In the USB 2.0 spec it says (Section 5.5.4) if the control transfers that are attempted consume less than 10% of the frame time for full/low speed endpoint or less than 20% of a microframe for high speed endpoints, the remaining time can be used to support bulk transfers. If there are too many pending control transfers for the available frame time, control transfers are selected to be moved over the bus as appropriate. Etc.

For this reason the idea that command index is sent on control pipe and response received on setup in transaction is rejected.

Application specific implementation is to use control pipe to read interface status. Data bytes are returned as Data In Transaction.

bmRequest Type	bRequest	bRequest value	wValue [binary]	wIndex	wLength	Data
11000000	GET_SYSTEM_PORT_STATUS	0	00000000	Port number	0x0F	0x0F
11000000	GET_HC_INFO	1	xxxxxxxx	-	4	4
01000000	RESET	2	xxxxxxxx	-	0	
01000000	ENTER_DFU	3	00001111	-	0	
01000000	SET_SD_POWER	4	00000000-off 00000001-on	Port number	0	
01000000	SET_SD_MODE	5	00000000-SDIO 00000001-SD	Port number	0	
01000000	SET_SD_DATA_MODE	6	00000000-1 bit 00000001-4 bit	Port number	0	



Interface Control Document for VUB300

Document Number ES446

01000000	SET_HOST_BUSY	7	00000000-!busy 11111111-busy	Port number	0	
01000000	SET_HOST_BLOCK_SIZE	8	wValue	Port number	0	
01000000	SET_FUNCTION_BLOCK_SIZE	9	wValue	Port number	0	
01000000	SET_FNO_BLOCK_SIZE	10	wValue	Port number	0	
01000000	SET_CLK_SPEED	11	xxxxxxx	Port number	8	Send value in KHz
01000000	SET_FET_MODE	12	0x00-100mA 0x01-200mA	Port number	0	
01000000	ENABLE_INTERRUPT	13	00000000-off 00000001-on	Port number	0	
01000000	ENABLE_READ_WAIT	14	00000000-off 00000001-on	Port number	0	
01000000	CLR_ERR	15	xxxxxxx	Port number	0	
01000000	ROM_WAIT_STATES	16	wValue	Port number	0	
01000000	4_BYTE_ALIGNMENT	17	wValue	Port number	0	
01000000	DRIVER_POWER_CTRL	18	wValue	Port number	0	
01000000	unused					
01000000	OFFLOAD_P_SUDO_IRQ	20	0	Port number	N	Pseudo Code
01000000	OFFLOAD_P_SUDO_CMD 53	21	0	Port number	N	Pseudo Code
01000000	IRQ_POLLING	22	00000000-on 00000001-off	Port number	0	0
01000000	SET_SD_TIMEOUT	23	Timeout	Port number	0	
01000000	SET_WAIT_TIMER	24	Timeout	Port number	0	0

Table 1. Setup command structure defined



Interface Control Document for VUB300

Document Number ES446

All of the vendor specific requests above are valid, but the use during data stage is not recommended due to know hardware bug in SMSC chip. When data concurrently appears on USB bus both on IN and on OUT, sie appears not to be able to cope, and IN packet is never placed on the USB bus. Setup is considered as OUT packet.

VUB300 SDIO Port Status Block – returned as response to GET_SYSTEM_PORT_STATUS vendor specific commands, see below:

Response packet will take on header type = 0x04 for all offload commands apart from polling one, as there is no piggybacked commands there.

	Bit 7	6	5	4	3	2	1	0
0	Status Block size in bytes							
1	Header Type							
2	Port number							
3		BUSY	FET_M ODE	WRITE _PROT ECT	4-BIT	SD_PO WER		SD_CA RD_INS ERTED
4							SDIO_ WAIT	SDIO_I NTR_E NABLE
5	SDIO CLOCK SETTING IN KHz							
6								
7								
8								
9	HOST BLOCK SIZE							
10								
11	FUNCTION BLOCK SIZE							
12								
13	FN0 BLOCK SIZE							
14								

Header Type

VUB300 SDIO Port Status Block Header Type is always 0x03 as this is Status Info.

Header Type byte values:

- 0 – Command Response Header
- 1 – Interrupt Header
- 2 – Error Header
- 3 – Status Header
- 4 – Response with offload

**Interface Control Document for VUB300**
Document Number ES446

- 5 – Offload on interrupt disabled
- 6- Offload on interrupt enabled
- 7 – data on CMD53 psudo code
- 8 – interrupt polling command response when no interrupt detected (0x03 0x08 0x00)
- 9 – CMD53 interrupt offload execution IRQ DISABLED
- A - CMD53 interrupt offload execution IRQ ENABLED

SD_CARD_INSERTED

USB2250 supports 4 different memory cards and USB2240 supports 3. When SD card inserted this bit is set to 1.

Port number

Start enumeration from 0. As USB2240 SMSC chip only can support one SD/SDIO port, this will be set to 0 across all of the project structures.

SD_POWER

Set when sd power enabled. SMSC code turns the power to the socket as soon as SD card inserted. Host is given control over power to the SD slot by the means of SET_SD_POWER vendor specific command.

When this vendor command receive firmware will flush USB endpoint 1 and endpoint 2, as well as 2 fmdu firmware endpoints. On power disable all queued work request will be lost.

SD Data Mode

SMSC firmware allows three data modes: 1, 4 or 8 bit mode. VUB300 allows just the first 2 modes: 0 -1 bit mode and 4-BIT = 1 for 4-bit data mode. Default is 1 bit data mode.

WRITE_PROTECT

1 when SD-card write protected. SMSC have assign GIO6 for hardware write protect, but also checks for register setting for write protect (see SET_SD_WRITE_PROTECT).

SDIO_INTR_ENABLE

Set this bit 1 to enable the interrupt. This is bit4 in register SDC_MODE_CTL in the firmware. On interrupt detection firmware disables the interrupt detection till next vendor specific request for interrupt enables received.

ENABLE_READ_WAIT

This is SDIO function. This bit is SDIO_WAIT , bit 7 in register SDC_CTL in the firmware. 0-disabled. NOT USED.

SET_FET_MODE

SMSC has 4 internal FETs giving control to 4 sockets. Available current is 100mA, but by writing 1 to a register bit, this value is increased to 200mA. SET_FET_MODE affects only one SD control FET. In firmware this is FET_CTL3, relevant bits are bit 4 and 5.

Default value for VUB300 is 1 = 200mA.

N.B. SMSC USB2250 can only deliver 200mA which means that the High-Power SDIO are not supported (want up to 500mA).

BUSY

Not used.

FET_MODE

Reporting if with SMSC chip FET for this port is set to supply 100mA or 200mA.



Interface Control Document for VUB300

Document Number ES446

Defined in firmware as:

```
typedef struct S_VUB300_SDIO_PORT_STATUS_STRUCTURE
{
    unit8 mySize;
    unit8 HeaderType;
    uint8 PortID;
    uint8 CrdInserted    : 1;
    uint8 NonSDCrđ       : 1;
    uint8 SDPwrOn        : 1;
    uint8 SDDataMode     : 1;
    uint8 SDWriteProtect : 1;
    uint8 FetMode        : 1;
    uint8 Busy           : 1;
    uint8 mRsrv_1        : 1;
    uint8 IntrEnable     : 1;
    uint8 WaitEnable     : 1;
    uint8 mRsrv_2        : 6;
    uint32 SDClkSetting;
    uint16 HostBlockSize;
    uint16 FunctionBlockSize;
    uint16 FN0BlockSize;
} VUB300_SDIO_PORT_STATUS_STRUCTURE, *PVUB300_SDIO_PORT_STATUS_STRUCTURE;
extern xdata S_VUB300_SDIO_PORT_STATUS_STRUCTURE vub300_sdio_port_status;
```

GET_SYSTEM_PORT_STATUS

VUB300 based on SMSC chip supports only one SD/SDIO port (Port number=0). Status of the port is returned to the host and is of structure VUB300 SDIO Port Status Block, see above.

GET_HC_INFO

Get host controller information. Returns Firmware revision and also number of ports available in the hardware.

Host must call this vendor specific command first as this is where the firmware posts first work request to receive an out from the host.

N.B. Command on EP1 on USB will not be ACKed if this vendor specific request is not called, in turn no data on EP2 will be acked unless commands is processed. You must send this vendor specific request at the start, and please do it just once to avoid overloading the sie work queue.

```
typedef struct S_SDIO_HC_INFO_STRUCTURE {
    UCHAR mySize
    USHORT FirmwareVer;
    UCHAR NumberOfPorts;
} SDIO_HC_INFO_STRUCTURE, *PSDIO_HC_INFO_STRUCTURE;
```

SET_SD_POWER

On card insert, firmware will initialize SD controller and turn the power on to the SD/SDIO card automatically. On car eject firmware switches the power off. Any other time with the card present driver is given power control over SD bus. Driver must poll for status for info on card insertion (this is due to SMSC chip hardware bug).

SET_SD_MODE

NOT USED.

SET_SD_DATA_MODE

Firmware will operate in SD 1-bit data mode by default. Firmware will be snooping commands to detect when to change to the 4-bit data mode. At the same time firmware will act on vendor specific request from the host too.

**ENTER_DFU**

Inherited from SMSC USB2240. "DFU-MARK" is placed on the AMBA bus, and when detected device enters DFU mode. Remember to use force parameter when running SMSC application for DFU updates.

RESET

NOT USED.

SET_HOST_BUSY

NOT USED.

SD spec allows the host to signal the SD/SDIO device that the host is busy, so the card delays further transactions.

Firmware is allowed to implement host busy on DAT2 line without ever notifying the SDIO host.

SET_HOST_BLOCK_SIZE

Default value for the SD block size is 512 bytes. SDIO block size is more flexible than SD. Use this command to set block size to any value between 0 and 2048 bytes. Send new block size as wLength of the vendor specific USB command.

SET_CLK_SPEED

As the spec states until the end of Card Identification Mode the host shall remain at f₀ frequency because some cards may have operational frequency restrictions. Default firmware speed is 200KHz on the SD CLK signal. USB2240 and USB2250 are restricted to just 7 different clock frequency values. Ideally the range of frequencies would be much wider. As part of this vendor specific command send clock frequency in KHz that you want SD clock to operate. In this case it is up to host to make sure the host only sends one of the values as defined below.

```
#define K_SD_SPEED_200KHZ
#define K_SD_SPEED_20MHZ
#define K_SD_SPEED_24MHZ
#define K_SD_SPEED_48MHZ
#define K_SD_SPEED_15MHZ
#define K_SD_SPEED_STREAM_FREQ
#define K_SD_SPEED_0MHZ
```

These values are fixed inside the USB2240 SMSC processors and cannot change.

N.B. Stream frequency value defined above is I think to do with MMC card.

The host controller can change the SD clock frequency by sending this vendor specific command.

ENABLE_INTERRUPT

SMSC USB2250 chip has got dedicated bit in the register SDC_MODE_CTL. Set this bit to '1' to enable SDIO interrupt detection. Firmware disables interrupt detection upon receipt of an interrupt, until the next enable interrupt vendor specific request received.

Interrupt poll command will enable the SDIO interrupt detection.

CLR_ERR

NOT USED.

ROM_WAIT_STATES

By default this register in SMSC chip is set to maximum on power up.



Interface Control Document for VUB300

Document Number ES446

With VUB300 boards driver should send 0x1C.
With the final ASIC the driver should send 0x04.

Register affected VUB300 is DEV_CFG 0x0803. Writing 0 to internal ROM wait state bits means 0 wait states, 1 = 1 wait states, etc.
Driver sends the Vendor specific Request 16 with the value to write to this register. MSByte is ignored, LSByte is written to this register, overriding both internal and external wait states.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DEV_CFG	Reserved (0)			Internal ROM		External ROM		

```
mcu_register_wr(x_dev_cfg, rqstp->wValueLo);
delay = 5;
while(delay--) {}
```

4_BYTE_ALIGNMENT

By default the firmware will check if PIO transfer is divisible by 4 and will do this transfer as in 4 byte alignment mode. Driver has the way to turn this off, just for debug purpose or firmware flexibility. Send wValue 0 to turn off 4 byte alignment.

DRIVER_PWR_CTRL

Not tested, not used – issues with media manager and SMSC chip power control.

2.3 Endpoint1 - EP1

VUB300 will be implementing 2 endpoint pairs for data transfer: EP1 for commands and responses, EP2 for data bulk transfers.

EP1 pipe is used for command and response transfers between the host and the SD/SDIO card.

2.3.1 VUB300 SD Command Header

	Bit 7	6	5	4	3	2	1	0
0	Structure size in bytes							
1	Header Type							
2	Port number							
3	Rd/Wr	Command Type						
4	COMMAND_INDEX							
5	TransferSize							
6								
7								



Interface Control Document for VUB300

Document Number ES446

8	
9	RESPONSE_TYPE
10	COMMAND_ARGUMENT
11	
12	
13	
14	BlockCount
15	BlockSize
16	
17	
18	BlockBoundary
19	
20	Irqpoll timeout lsb
21	Irqpoll timeout msb
22...63	(NULL Packing bytes to 64 total)

StructureSize

Number of valid bytes in this Command packet header (set to 18)

Header Type

For command header set Header Type to 0x00.

Header Type byte values:

- 0 – Command Response Header
- 1 – Interrupt Header
- 2 – Error Header
- 3 – Status Header
- 4 – Response with offload
- 5 – Offload on interrupt disabled
- 6- Offload on interrupt enabled
- 7 – data on CMD53 psudo code
- 8 – interrupt polling command response when no interrupt detected (0x03 0x08 0x00)
- 9 – CMD53 interrupt offload execution IRQ DISABLED
- A - CMD53 interrupt offload execution IRQ ENABLED

COMMAND_TYPE:

- 0x00 – Standard command
- 0x01 – Dummy command, return status block. Not used. Ana 16.03.2010.
- 0x02 – Polling for an interrupt
- 0x03-0xFF – undefined

Rd/Wr

Set 1 for write operation and 0 for read operation.



Interface Control Document for VUB300

Document Number ES446

This bit only valid when transmit size is greater than zero.

Port number

For VUB300 this is set to 0 as SMSC USB2240 can only support one SD/SDIO port.

COMMAND_INDEX

One byte wide. E.g. for command 53 pass value 53.

RESPONSE_TYPE

Depending on the SMSC firmware it might be easier to instead of length the host sends a response type e.g. 0x00=none, 0x01=R1, etc.

```
typedef enum {
    SDRT_UNSPECIFIED = 0,
    SDRT_NONE,
    SDRT_1,
    SDRT_1B,
    SDRT_2,
    SDRT_3,
    SDRT_4,
    SDRT_5,
    SDRT_5B,
    SDRT_6,
    SDRT_7,
} SD_RESPONSE_TYPE;
```

TransferSize

This is set to the transfer size in bytes. (Big-Endian format)

BlockCount

Number of blocks in this transfer. The BlockCount should be set to 0 if the TransferSize is less than the BlockSize. (Big-Endian format)

BlockSize

The block size (in bytes) for this transfer. If the BlockCount is 0, then the BlockSize may be ignored and the TransferSize is used for a "byte mode" transfer. (Big-Endian format)

Example pseudo-code for TransferSize etc:

```
if (TransferSize >= BlockSize) {
    TransferSize = BufferSize;
    BlockSize = 512; //valid for SDHC cards!
    BlockCount = BufferSize / BlockSize;
    ASSERT(TransferSize == (BlockSize x BlockCount));
} else {
    TransferSize = BufferSize;
    BlockSize = 0;
    BlockCount = 0;
}
```

Irqpoll timeout lsb/Irqpoll timeout lsb

8051 Timer/Counter loaded with this value. Counter up, so 0xFFFF is the shortest delay.

e.g. 64 bytes (16 01 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AA BB is timeout value of 0xAABB

NULL packing



Due to pre fetch fifo SMSC bug the OUT packets on EP1 are 64 bytes long and there is need for a dummy command on EP1 to flush the pre fetch fifo.

Command header defined in firmware as:

```
typedef struct S_COMMAND_HEADER_STRUCTURE
{
    uint8 mySize;
    uint8 HeaderType;
    uint8 PortID;
    uint8 RdWr : 1;
    uint8 CmdType :7;
    uint8 CmdIndex;
    uint32 TransferSize;
    uint8 ResponseType;
    uint32 CmdArg;
    uint16 BlockCount;
    uint16 BlockSize;
    t_uw16 CardBlockBoundary;
} COMMAND_HEADER_STRUCTURE,*PCOMMAND_HEADER_STRUCTURE;
extern xdata COMMAND_HEADER_STRUCTURE vub300_command_header;
```

Firmware will then strip the header, wrap the command with the start bit, the transmit bit, CRC, the stop bit and send the SD command out. The response received from the card is then sent on EP1 as In Transfer with the header 4 byte block.

2.3.2 VUB300 SD Response Header

The type of the response is read from the first byte from the response block.

- 0 – Command Response Header
- 1 – Interrupt Header
- 2 – Error Header
- 3 – Status Header

2.3.3 Command Response Header

	Bit 7	6	5	4	3	2	1	0
0	Structure size in bytes							
1	Header Type							
2	Port number							
3	Rd/Wr	COMMAND_TYPE						
4	COMMAND_INDEX							
5 on	SDIO COMMAND RESPONSE							

Response Header



Interface Control Document for VUB300

Document Number ES446

Command header on EP1 is responded to with the response header block with header type = 0x0. When the SD/SDIO card response is not expected the firmware will still send a short Command Response Header (5 bytes long).

```
typedef struct S_EP1_COMMAND_RESPONSE_STRUCTURE
{
    uint8 mySize;
    uint8 HeaderType;
    uint8 PortID;
    uint8 RdWr :1;
    uint8 CommandType : 7;
    uint8 CommandIndex;
    uint8 CommandResponse[17];
} EP1_COMMAND_RESPONSE_STRUCTURE,*PEP1_COMMAND_RESPONSE_STRUCTURE;
extern xdata EP1_COMMAND_RESPONSE_STRUCTURE vub300_ep1_command_response;
```

2.3.4 Interrupt Header

	Bit 7	6	5	4	3	2	1	0
0	Block size in bytes							
1	Header Type							
2	Port number							
3...	Interrupt pseudo code execution							

Interrupt Header

On interrupt firmware checks for pseudo code, and executes loading the bytes onto the interrupt structure and sending up to the driver.

2.3.5 Error Header

	Bit 7	6	5	4	3	2	1	0
0	Block size in bytes (0x04)							
1	Header Type (0x02)							
2	Port number							
3	Error Code							

Error Header

When error occurs and host needs to be notified immediately, 4 bytes block will be placed on the EP1 with the Header type = 0x02. This could be very useful if firmware has detected CRC error on data lines; host could be immediately notified.

Host then must send CLR_ERR vendor specific command to signal error acknowledgement.



Interface Control Document for VUB300

Document Number ES446

Error Code	Read Aborted	meaning
0		reserved
1		Timeout on a 1-bit operation
2		Timeout on a 4-bit operation
3		Received 1-bit CRC from card was wrong
4		Received 4-bit CRC from card was wrong
5		Device reported 1-bit CRC error
6		Device reported 4-bit CRC error
7		CMD end-bit not seen
8		The 1-bit DATA end-bit not seen
9		The 4-bit DATA end-bit not seen
10		Unexpected Timeout on a 1-bit operation
11		Unexpected Timeout on a 4-bit operation
12		Illegal CMD
13		Device not inserted
14		Data Transfer Length > 64K
15	yes	Timeout on 1-Bit Data Start Bit
16	yes	Timeout on 4-Bit Data Start Bit
17	yes	Invalid CMD53 for current state
18	yes	Unknown CMD53 Error
19	yes	Reserved CMD53 Error
20	yes	Invalid CMD53 Function Number
21	yes	CMD53 Out of Range
22		VUB300_STAT_CMD_ERR
23		VUB300_STAT_DATA_ERR
24		VUB300_STAT_CMD_TIMEOUT
25		SDC_RDY stuck on 0 (0 = not ready)
26		VUB300 unhandled error
27		VUB300 overrun
28		VUB300 PIO Timeout
29		VUB300 Busy
30		CCCR5 reg read command error
31		VUB300 Pseudo code err
32		VUB300_EP1_NOT_EMPTY

Those commands that expect data to be received on Bulk EndPoint 2 will have that data transfer aborted if so marked in the above table.

Errors detected by the firmware are cleared by the host sending CLR_ERR vendor specific command.

These are defines for the USB2250/USB2240 firmware:

```
#define VUB300_CRD_NOT_INSERTED 13
#define VUB300_ABOVE_64K 14
#define VUB300_STAT_CMD_ERR 22
#define VUB300_STAT_DATA_ERR 23
#define VUB300_STAT_CMD_TIMEOUT 24
#define VUB300_SDC_RDY_ERR 25
#define VUB300_UNHANDLED_ERR 26
```

```
typedef struct S_EP1_ERROR_RESPONSE_STRUCTURE
```



```
{
    uint8 mySize;
    uint8 HeaderType;
    uint8 PortID;
    uint8 ErrByte;
} EP1_ERROR_RESPONSE_STRUCTURE,*PEP1_ERROR_RESPONSE_STRUCTURE;
extern xdata EP1_ERROR_RESPONSE_STRUCTURE vub300_error_header;
```

2.3.6 Status Header

Return VUB300 SDIO Port Status Block, defined in section 2.2.2.

2.3.7 PseudoCode Execution Results

The data results of executing either Transfer or Interrupt PseudoCode will be a number of blocks of 8 bytes. Each such 8 byte block will consist of 4 bytes of the SDIO CMD52 and 4 bytes of the SD response.

Each CMD52 consists of 6 bytes: 74 cc cc cc cc ss

Each SD response consists of 6 bytes: 34 rr rr rr rr ss

where cc cc cc cc are the first 4 bytes in each block of responses and rr rr rr rr are the second 4 bytes in each block of responses and where ss are the SD checksums (which are not returned).

Please see firmware project source code for Pseudo codes and their meaning/implementation. Driver sends vendor specific request for interrupt offload and for command offload. Firmware will execute pseudo code routine if the length of the data array send by host is greater than 0.

2.4 Endpoint 2 - EP2

VUB300 will be implementing 2 endpoint pairs for data transfer: EP1 for commands and responses, EP2 for data bulk transfers.

Data on the EP2 will be sent without any header or footers or rappers. It is assumed that all the options are set by the option byte included with the command sent on EP1.

If SD/SDIO card is working in 4-bit mode, firmware will arrange data in bytes and check CRC for the data lines. If CRC error detected ERR3 bit will be set.

Although the interrupt can be signalled during the data transfer on SD DAT lines, this will in no way be shown on the EP2 data transfers.

On EP1 command acknowledgement firmware will analyse the command and post data buffers accordingly.