VS1063a HARDWARE GUIDE MP3/OGG/AAC/WMA/FLAC/ G.711/G.722 AUDIO CODEC CIRCUIT

Key Features

- Encoders:
  MP3; Ogg Vorbis; PCM; IMA ADPCM; G.711 (μ-law, A-law); G.722 ADPCM
- Decoders:
  MP3 (MPEG 1 & 2 audio layer III (CBR +VBR +ABR));
  MP2 (layer II) (optional);
  MPEG4 / 2 AAC-LC(+PNS),
  HE-AAC v2 (Level 3) (SBR + PS);
  Ogg Vorbis; FLAC;
  WMA 4.0/4.1/7/8/9 all profiles (5-384 kbps);
  WAV (PCM, IMA ADPCM, G.711 μ-law/A-law, G.722 ADPCM)
- Full Duplex Codecs:
  PCM; G.711 (μ-law, A-law); G.722 ADPCM; IMA ADPCM
- Streaming support
- Upto 96 KiB RAM for user code and data
- Unique ID for user code protection
- Quiet power-on and power-off
- I2S output interface for external DAC
- Serial control and data interfaces
- Can be used either as a slave co-processor or as a standalone processor
- UART for debugging purposes
- New functions may be added with software and upto 12 GPIO pins

Description

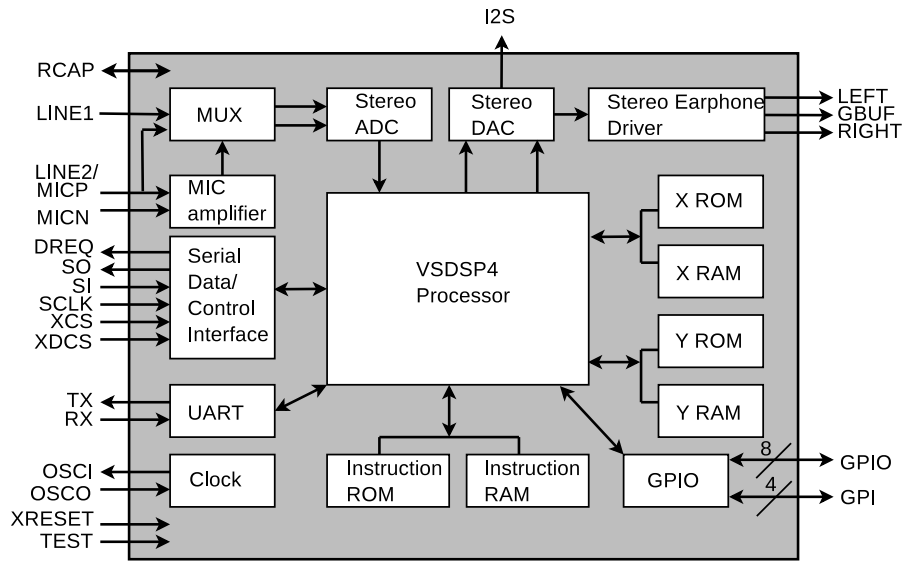
VS1063a is an easy-to-use, versatile encoder, decoder and codec for a multitude of audio formats.

VS1063a contains a high-performance, proprietary low-power DSP core VS_DSP^4, ROM memories, 16 KiB instruction RAM and upto 80 KiB data RAM for user applications running simultaneously with any built-in decoder, serial control and input data interfaces, upto 12 general purpose I/O pins, a UART, as well as a high-quality variable-sample-rate stereo DAC and a stereo ADC, followed by an earphone amplifier and a common voltage buffer.

VS1063a can act both as an “MP3 decoder IC” or “MP3 encoder IC” slave in a system with a microcontroller, or as a stand-alone circuit that boots from external SPI memory.

Applications

- MP3-recording audio player
- Streaming server and client
- Wireless audio transfer
- Standalone player and recorder
- Internet phones



Additional Features

- EarSpeaker Spatial Processing
- Bass & treble controls
- Alternatively a 5-channel equalizer
- AD Mixer allows monitoring A/D converter input while listening to stream
- PCM Mixer allows inserting a sidestream while listening to main stream
- Adjustable Speed Shifter
- Operates with a single 12..13 MHz or 24..26 MHz clock
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Zero-cross detection for smooth volume change
- Stereo earphone driver capable of driving a 30 Ω load
- Separate voltages for analog, digital, I/O
- Lead-free RoHS-compliant package

Further Description

VS1063a is a pin-compatible alternative for VLSI Solution's VS1053. It has all the functionality of VS1053 (except MP1 and MIDI decoding) and many new features, particularly MP3 and Ogg Vorbis recording.

Also full-duplex codec functions for phone applications have been added to VS1063a.

A factory-programmable unique chip ID provides a basis for digital rights management or unit identification features.

Operating Modes

VS1063a operates in one of two host modes: as a slave co-processor or as a standalone processor.

When used as a slave co-processor VS1063a can operate in three different operation modes: *decoder*, *encoder* or *codec* mode. In *decoder mode* VS1063a receives its input bitstream through a serial input bus. The input stream is decoded and passed through an 18-bit digital volume control to an oversampling sigma-delta DAC. Decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory, or even to load user applications. In *encoder mode* VS1063a can reads audio from its analog inputs, optionally compresses the data, which can then be read by the host processor. In *codec mode* VS1063a offers a full-duplex audio interface.

When used as a standalone processor the VS1063a can boot either from SPI EEPROM or FLASH memory. Alternatively code and data can be provided by a host controller.

User Code

Users can write their own user interface or signal processing code for the VS1063a using VSIDE (VLSI Solution's Integrated Development Environment).

As a default, there are 16 KiB of free code RAM and about 4 KiB of free data RAM for user applications. Depending on the application, the data RAM can be expanded to the full 80 KiB that is available in VS1063a.

Contents

VS1063 Hardware Guide Front Page 1

Table of Contents 3

List of Figures 5

1 Introduction 6

2 Disclaimer 6

3 Definitions 6

4 The VSDSP Processor Core 7

5 VS1063a Memory Map 8

6 VS1063a Hardware Audio Paths 9
6.1 VS1063a Hardware DAC Audio Paths . . . 9
6.2 VS1063a Hardware ADC Audio Paths . . . 10

7 Hardware Registers 11
7.1 Serial Control Interface (SCI) . . . 11
7.1.1 SCI Hardware Registers . . . 11
7.1.2 Mode SCI_MODE (RW) . . . 11
7.1.3 Status Information SCI_STATUS (RW) . . . 12
7.1.4 SCI Change SCI_CHANGE (R) . . . 13
7.1.5 SCI Interrupts . . . 13
7.2 Serial Data Interface (SDI) . . . 13
7.2.1 SDI Interrupts . . . 13
7.3 Digital-to-Analog Converter (DAC) . . . 14
7.3.1 DAC Registers . . . 14
7.3.2 Volume Control DAC_VOL . . . 14
7.3.3 DAC Interrupts . . . 14
7.4 PLL Controller . . . 15
7.4.1 PLL Control DAC_FCTLH[13:4] . . . 15
7.5 General-Purpose Input/Output (GPIO) . . . 17
7.5.1 GPIO Registers . . . 17
7.6 Interrupt Control . . . 18
7.6.1 Interrupt Control Registers . . . 18
7.7 Universal Asynchronous Receiver/Transmitter (UART) . . . 19
7.7.1 UART Registers . . . 19
7.7.2 Status UART_STATUS . . . 19
7.7.3 Data UART_DATA . . . 20
7.7.4 Data High UART_DATAH . . . 20
7.7.5 Divider UART_DIV . . . 20
7.7.6 UART Interrupts and Operation . . . 21
7.8 Timers . . . 22
7.8.1 Timer Registers . . . 22
7.8.2 Configuration TIMER_CONFIG . . . 22
7.8.3 Configuration TIMER_ENABLE . . . 23

7.8.4 Timer X Startvalue TIMER_Tx[L/H] . . . 23
7.8.5 Timer X Counter TIMER_TxCNT[L/H] . . . 23
7.8.6 Timer Interrupts . . . 23
7.9 I2S DAC Interface . . . 24
7.9.1 I2S Registers . . . 24
7.9.2 Configuration I2S_CONFIG . . . 24
7.10 Analog-to-Digital Converter (ADC) . . . 25
7.10.1 ADC Registers . . . 25
7.10.2 Control ADC_CONTROL . . . 25
7.10.3 ADC Interrupts . . . 25
7.11 Resampler SampleRate Converter (SRC) . . . 26
7.11.1 SRC Registers . . . 26
7.11.2 Control SRC_CONTROL . . . 26
7.11.3 SRC Interrupts . . . 26
7.12 Sidestream Sigma-Delta Modulator (SDM) . . . 27
7.12.1 SDM Registers . . . 27
7.12.2 Control SDM_CONTROL . . . 27
7.12.3 SDM Interrupts . . . 27

8 Latest Document Version Changes 28

9 Contact Information 29

List of Figures

1 VS1063a DAC data paths with some data registers 9
2 VS1063a ADC data paths with some data registers 10
3 RS232 serial interface protocol 19
4 I2S interface, 192 kHz. 24

1 Introduction

This is a hardware guide for the VS1063a. Its intent is to provide the reader with sufficient amounts of information to write programs that take over VS1063a in part or in whole. It is also intended to give users a better understanding of the inner workings of VS1063a.

Because VS1063a shares the hardware and register interfaces with the VS1053 and VS8053 series audio processors, everything written in this document, unless specifically marked otherwise, is applicable to all of them.

2 Disclaimer

This is a *preliminary* guide. It represents VLSI Solution's best attempt at documenting the hardware of VS1063a. However, VLSI Solution is not responsible for any errors, omissions, or misleading statements in this document.

This guide is the second part of a three-part entity, consisting of *VS1063a Datasheet*, *VS1063a Hardware Guide*, and *VS1063a Programmer's Guide*. To avoid repetition, it is assumed that the reader of this guide has already familiarized himself with the VS1063a Datasheet.

3 Definitions

ABR Average BitRate. Bitrate of stream may vary locally, but will stay close to a given number when averaged over a longer time.

B Byte, 8 bits.

b Bit.

CBR Constant BitRate. Bitrate of stream will be the same for each compression block.

Ki "Kibi" = 2^{10} = 1024 (IEC 60027-2).

Mi "Mebi" = 2^{20} = 1048576 (IEC 60027-2).

VBR Variable BitRate. Bitrate will vary depending on the complexity of the source material.

VSIDE VLSI Solution's Integrated Development Environment.

VS_DSP VLSI Solution's DSP core.

W Word. In VS_DSP, instruction words are 32-bit and data words are 16-bit wide.

4 The VSDSP Processor Core

VS_DSP is a 16/32-bit DSP processor core that also had extensive all-purpose processor features.

VLSI Solution's free Integrated Development Environment VSIDE contains all the tools and documentation needed to write, simulate and debug Extended ANSI C or Assembly Language programs for VS1063a.

5 VS1063a Memory Map

X-memory		Y-memory		I-memory	
Address	**Description**	**Address**	**Description**	**Address**	**Description**
0x0000..0x17ff	System RAM	0x0000..0x17ff	System RAM	0x0000..0x0fff	RAM
0x1800..0x187f	User RAM	0x1800..0x187f	User RAM	0x0050..0x0fff	User RAM
0x1880..0x197f	Stack	0x1880..0x197f	Stack	0x1000..0x1fff	-
0x1980..0x3fff	System RAM	0x1980..0x3fff	System RAM	0x2000..0xffff	ROM 56k
0x4000..0xbfff	ROM 32k	0x4000..0xdfff	ROM 40k		and banked
0xc000..0xc0ff	Peripherals	0xe000..0xffff	System RAM	0xc000..0xffff	ROM4 16k
0xc100..0xffff	ROM 15.75k				

Note: The VS1063a Programmer's Guide has a more accurate software memory map.

6 VS1063a Hardware Audio Paths

6.1 VS1063a Hardware DAC Audio Paths

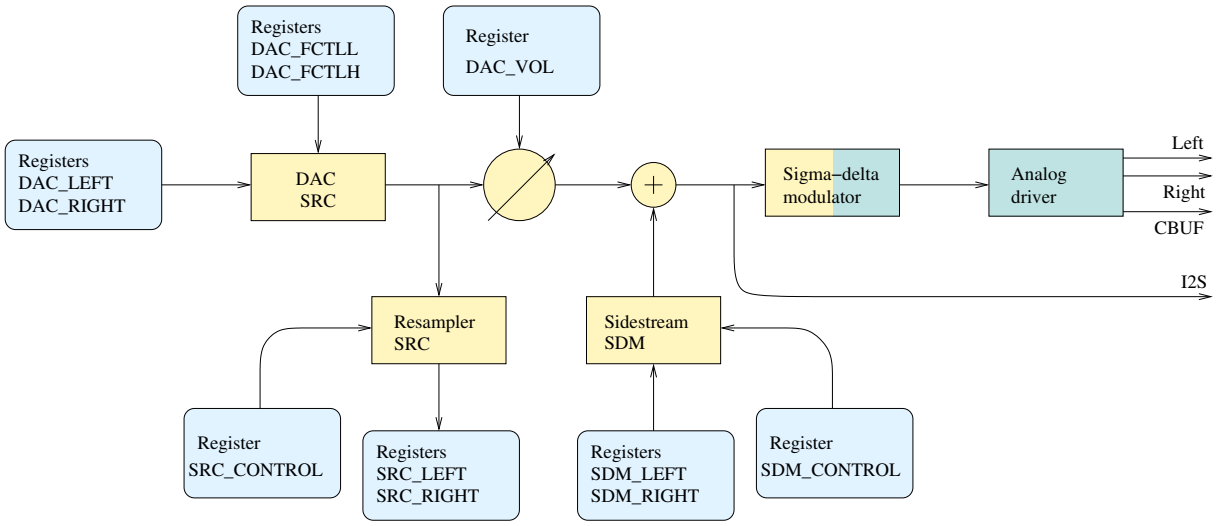


Figure 1: VS1063a DAC data paths with some data registers

Figure 1 presents the VS1063a hardware DAC audio paths.

The main audio path starts from the DAC register (Chapter 7.3) to the high-fidelity, fully digital DAC SRC (Digital-to-Analog Converter SampleRate Converter), which low-pass filters and interpolates the data to the high samplerate of XTALI/2 (nominally 6.144 MHz). This 18-bit data is then fed to the volume control. It then passes through the sigma-delta modulator to the analog driver and analog Left and Right signals.

The user may resample and record the data with the Resampler SampleRate Converter (Chapter 7.11). Because there is no automatic low-pass filtering, it is the user's responsibility to avoid aliasing distortion.

The user may add a PCM sidestream with the Sidestream Sigma-Delta Modulator input (Chapter 7.12). As is the case with the Resampler SampleRate Converter, hardware doesn't offer low-pass filtering, so sufficient aliasing image rejection is the responsibility of the user.

6.2 VS1063a Hardware ADC Audio Paths

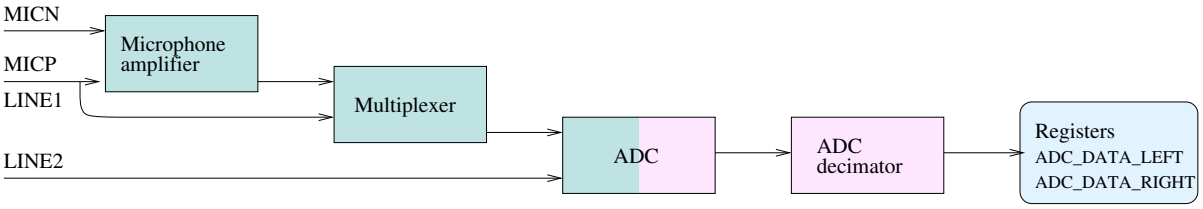


Figure 2: VS1063a ADC data paths with some data registers

Figure 1 presents the VS1063a hardware ADC audio paths.

Analog audio may be fed upto two channels: one as a differential signal to MICN/MICP or as a one-sided signal to Line1, and the other as a one-sided signal to Line2.

If microphone input for the left channel has been selected, audio is fed through a microphone amplifier and that signal is selected by a multiplexer.

Audio is then downsampled to one of four allowed samplerates: XTALI/64, XTALI/128, XTALI/256 or XTALI/512. With the nominal 12.288 MHz crystal, these correspond to 192, 96, 48 or 24 kHz samplerates, respectively (Chapter 7.12).

If the “3 MHz” option bit SS_AD_CLOCK in register SCI_STATUS has been set to 1, then samplerates are divided by two, so the nominal samplerates become 96, 48, 24 and 12 kHz.

7 Hardware Registers

VS1063a registers are memory mapped to X data memory, starting from address 0xC000.

7.1 Serial Control Interface (SCI)

SCI registers are mapped to X memory addresses between 0xC000..0xC00F. In addition to these registers, there is one in address 0xC010, called SCI_CHANGE.

Only 3 of these 17 registers contain direct hardware control. Those registers and their hardware registers bits are described in this section. For documentation of the software register bits, see the VS1063a Datasheet.

7.1.1 SCI Hardware Registers

SCI registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC000	rw	0x4000[2]	SCI_MODE	Mode control
0xC001	rw	0x000C[2]	SCI_STATUS	Status of VS1063a
0xC010	r	0x00	SCI_CHANGE[5:0]	Last SCI access address

[1] Firmware changes the value of this register immediately to 0x4800.
[2] Firmware changes the value of this register immediately to 0x48 (analog enabled), and after a short while to 0x40 (analog drivers enabled).

7.1.2 Mode SCI_MODE (RW)

SCI_MODE bits				
Name	**Bit**	**Function**	**Value**	**Description**
SM_DACT	8	DCLK active edge	0 1	rising falling
SM_SDIORD	9	SDI bit order	0 1	MSb first MSb last
SM_SDISHARE	10	Share SPI chip select	0 1	no yes
SM_SDINEW	11	VS1002 native SPI modes	0 1	no yes
SM_LINE1	14	MIC / LINE1 selector	0 1	MICP LINE1
SM_CLK_RANGE	15	Input clock range	0 1	12..13 MHz 24..26 MHz

SM_DACT defines the active edge of data clock for SDI. When ’0’, data is read at the rising edge, when ’1’, data is read at the falling edge.

When SM_SDIORD is clear, bytes on SDI are sent MSb first. By setting SM_SDIORD, the user may reverse the bit order for SDI, i.e. bit 0 is received first and bit 7 last. Bytes are, however, still sent in the default order. This register bit has no effect on the SCI bus.

Setting SM_SDISHARE and SM_SDINEW makes SCI and SDI share the same chip select.

Setting SM_SDINEW will activate VS1002 native serial modes. Note, that this bit is set as a default when VS1063a is started up.

SM_LINE_IN is used to select the left-channel input for ADPCM recording. If ’0’, differential microphone input pins MICP and MICN are used; if ’1’, line-level MICP/LINEIN1 pin is used.

SM_CLK_RANGE activates a clock divider in the XTAL input. When SM_CLK_RANGE is set, the clock is divided by 2 at the input. From the chip’s point of view e.g. 24 MHz becomes 12 MHz. SM_CLK_RANGE should be set as soon as possible after a chip reset.

7.1.3 Status Information SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS1063a. It also controls some low-level things that the user does not usually have to care about.

SCI_STATUS bits		
Name	**Bits**	**Description**
SS_SWING	14:12	Set swing to +0 dB, +0.5 dB, .., or +3.5 dB
SS_VCM_OVERLOAD	11	GBUF overload indicator ’1’ = overload
SS_VCM_DISABLE	10	GBUF overload detection ’1’ = disable
SS_APDOWN2	3	Analog driver powerdown
SS_APDOWN1	2	Analog internal powerdown
SS_AD_CLOCK	1	AD clock select, ’0’ = 6 MHz, ’1’ = 3 MHz
SS_REFERENCE_SEL	0	Reference voltage selection, ’0’ = 1.23 V, ’1’ = 1.65 V

If AVDD is higher at least 3.3 V, SS_REFERENCE_SEL can be set to select 1.65 V reference voltage to increase the analog output swing.

SS_AD_CLOCK can be set to divide the AD modulator frequency by 2 if XTALI/2 is too much.

SS_APDOWN2 controls analog driver powerdown. SS_APDOWN1 controls internal analog powerdown. These bit are meant to be used by the system firmware only.

VS1063a contains GBUF protection circuit which disconnects the GBUF driver when too much current is drawn, indicating a short-circuit to ground. SS_VCM_OVERLOAD is high while the overload is detected. SS_VCM_DISABLE can be set to disable the protection feature.

SS_SWING allows you to go above the 0 dB volume setting. Value 0 is normal mode, 1 gives +0.5 dB, and 2 gives +1.0 dB. Settings from 3 to 7 cause the DAC modulator to be overdriven and should not be used. You can use SS_SWING with I2S to control the amount of headroom.

7.1.4 SCI Change SCI_CHANGE (R)

SCI_CHANGE bits		
Name	**Bits**	**Description**
SCI_CH_WRITE	4	1 if last access was a write cycle
SCI_CH_ADDR	3:0	SCI address of last access

SCI_CHANGE contains the last SCI register that has been accessed through the SCI bus, as well as whether the access was a read or write operation.

7.1.5 SCI Interrupts

Whenever an external SPI read from or write to an SCI register is done, an SCI interrupt is generated. The interrupt routine can then check SCI_CHANGE to see which SCI register was accessed and whether the access was a read or write operation.

7.2 Serial Data Interface (SDI)

Whenever two bytes have been written to the SDI bus, an interrupt is generated and the data can be read as a 16-bit big-endian value from the SDI registers. The user can control the DREQ pin as if it was a general-purpose output through its own register bit.

SDI registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC011	r	x	SDI_DATA	Last received 2 bytes, big-endian
0xC012	w	0	SDI_DREQ[0]	DREQ pin control

7.2.1 SDI Interrupts

Each time a new byte pair has been received, an SDI interrupt is generated.

7.3 Digital-to-Analog Converter (DAC)

7.3.1 DAC Registers

DAC registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC013	rw	0	DAC_FCTLL	DAC samplerate control, 16 LSbs
0xC014	rw	0	DAC_FCTLH	DAC samplerate control 4 MSbs, PLL control
0xC015	rw	0	DAC_LEFT	DAC left channel PCM value
0xC016	rw	0	DAC_RIGHT	DAC right channel PCM value
0xC045	rw	0	DAC_VOL	DAC hardware volume

The internal 20-bit register DAC_FCTL is calculated from DAC_FCTLH and DAC_FCTLL registers as follows: DAC_FCTL = (DAC_FCTLH & 15) $\times$ 65536 + DAC_FCTLL. Highest supported value for DAC_FCTL is 0x80000.

If we define C = DAC_FCTL and X = XTALI in Hz, then the resulting samplerate f_s of the associated DAC SampleRate Converter is $f_s = C \times X \times 2^{-27}$.

Example:
If C = 0x80000 and X = 12.288 MHz then $f_s = 524288 \times (12.288 \times 10^6) \times 2^{-27} = 48000$ (Hz).

Note: FCTLH bits 13:4 are used for the PLL Controller. See Chapter 7.4 for details.

7.3.2 Volume Control DAC_VOL

DAC_VOL bits		
Name	**Bits**	**Description**
LEFT_FINE	15:12	Left channel gain +0.0 dB. . .+5.5 dB (0 to 11)
LEFT_COARSE	11:8	Left channel attenuation in -6 dB steps
RIGHT_FINE	7:4	Right channel volume +0.0 dB. . .+5.5 dB (0 to 11)
RIGHT_COARSE	3:0	Right channel attenuation in -6 dB steps

Normally DAC_VOL is handled by the firmware. DAC_VOL depends on the software register SCI_VOL and the bass and treble settings in SCI_BASS (and optionally SS_SWING bits in SCI_STATUS).

7.3.3 DAC Interrupts

Whenever a new stereo sample has been sent out and a new value is required, a DAC interrupt is generated.

7.4 PLL Controller

The Phase-Locked Loop (PLL) controller is used to generate clock frequencies that are higher than the incoming (crystal-based) clock frequency. The PLL output is used by the CPU core and some peripherals.

Configurable features include:

- VCO Enable/Disable
- Select VCO or input clock to be output clock
- Route VCO frequency to output pin
- Select PLL clock multiplier

At the core of the PLL controller is the VCO, a high frequency oscillator, whose oscillation frequency is adjusted to be an integer multiple of some input frequency. As the name “Phase-Locked Loop” suggests, this is done by comparing the phase of the input frequency against the phase of a signal which is derived from the VCO output through frequency division.

If the system is stable, e.g. the comparison phase difference remains virtually zero, the PLL is said to be “in lock”. This means that the output frequency of the VCO is stable and reliable.

The PLL is preceded by a division-by-two unit. Thus, with a nominal XTALI = 12.288 MHz, the internal clock frequency CLKI can be adjusted with an accuracy of XTALI/2 = 6.144 MHz.

7.4.1 PLL Control DAC_FCTLH[13:4]

PLL control lies in DAC_FCTL bits 13:4. To see what bits 3:0 do, see Chapter 7.3.

FREQCTLH PLL bits		
Name	**Bits**	**Description**
FCH_PLL_LOCKED	13	0=lock failed since last test (read-only)
FCH_PLL_SET_LOCK	12	1:Sets FCH_PLL_LOCKED to 1 to start lock test
FCH_PLL_VCO_OUT_ENA	11	Route VCO to GPIO pin (VS1000:second cs pin)
FCH_PLL_FORCE_PLL	9	1:System clock is VCO / 0:System clock is inclk
FCH_PLL_DIV_INCLK	8	divide inclk by 2 (for 1.5, 2.5 or 3.5 x clk)
FCH_PLL_RATE	7:4	PLL rate control

The PLL locked status can be checked by generating a high-active pulse (writing first “1” , then “0”) to FCH_PLL_SET_LOCK and reading FCH_PLL_LOCKED. FCH_PLL_LOCKED is set to “1” along with the high level of FCH_PLL_SET_LOCK and to “0” whenever the PLL falls out of lock. So if the “1” remains in FCH_PLL_LOCKED, PLL is in sync.

The PLL controller's operation is optimized for frequencies around 12. . . 13 MHz. If you use an 24. . . 26 MHz input clock, set the extra clock divider bit SM_CLK_RANGE in register SCI_MODE to 1 before activating the PLL.

It's recommended to change the PLL rate in small steps and wait for the PLL to stabilize after each change. For diagnostic purposes, the PLL clock output (VCO) can be routed to an I/O pin so it can be scanned with an oscilloscope.

FCH_PLL_RATE (bits 7:4) control PLL multiplication rate. PLL multiplier is (FCH_PLL_RATE + 1). When FCH_PLL_RATE is 0, the VCO is powered down and output clock is forced to be input clock (same as if FCH_PLL_FORCE_PLL = 0).

7.5 General-Purpose Input/Output (GPIO)

7.5.1 GPIO Registers

GPIO registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC017	rw	0	GPIO_DDR[9:0]	Direction
0xC018	r	0	GPIO_IDATA[11:0]	Values read from the pins
0xC019	rw	0	GPIO_ODATA[8:0]	Values set to the pins

GPIO_DIR is used to set the direction of the GPIO pins. 1 means output. GPIO_ODATA remembers its values even if a GPIO_DIR bit is set to input.

GPIO_IDATA is used to read the pin states. In VS1063 also the SDI and SCI input pins can be read through GPIO_IDATA: SCLK = GPIO_IDATA[8], XCS = GPIO_IDATA[9], SI = GPIO_IDATA[10], and XDCS = GPIO_IDATA[11].

In addition to data direction control for GPIO pins 0 to 7, there are two additional control bits in GPIO_DIR. GPIO_DIR[8] switches SO into software control, so that the value in GPIO_ODATA[8] is shown on SO whenever xCS is low. When GPIO_DIR[9] is set to '1' SCI and SDI are disabled. When using SCLK, xCS, SI and xDCS as general-purpose input, GPIO_DIR[9] prevents transitions in them from getting random data from SDI and/or SCI.

GPIO registers don't generate interrupts.

Note that in VS1063a the VSDSP registers can be read and written through the SCI_WRAMADDR and SCI_WRAM registers. You can thus use the GPIO pins quite conveniently.

7.6 Interrupt Control

7.6.1 Interrupt Control Registers

Interrupt registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC01A	rw	0	INT_ENABLE[9:0]	Interrupt enable
0xC01B	w	0	INT_GLOB_DIS[-]	Write to add to interrupt counter
0xC01C	w	0	INT_GLOB_ENA[-]	Write to subtract from interrupt counter
0xC01D	r	0	INT_COUNTER[3:0]	Interrupt counter

INT_ENABLE controls the interrupts. The control bits are as follows:

INT_ENABLE bits		
Name	**Bits**	**Description**
INT_EN_SDM	9	Enable Sigma Delta Modulator interrupt
INT_EN_SRC	8	Enable SampleRate Converter interrupt
INT_EN_TIM1	7	Enable Timer 1 interrupt
INT_EN_TIM0	6	Enable Timer 0 interrupt
INT_EN_RX	5	Enable UART RX interrupt
INT_EN_TX	4	Enable UART TX interrupt
INT_EN_ADC	3	Enable ADC interrupt
INT_EN_SDI	2	Enable SDI interrupt
INT_EN_SCI	1	Enable SCI interrupt
INT_EN_DAC	0	Enable DAC interrupt

Note: It may take upto 6 clock cycles before changing INT_ENABLE has any effect.

Writing any value to INT_GLOB_DIS adds one to the interrupt counter INT_COUNTER and effectively disables all interrupts. It may take upto 6 clock cycles before writing to this register has any effect.

Writing any value to INT_GLOB_ENA subtracts one from the interrupt counter INT_COUNTER, unless it already was 0, in which case nothing happens. If, after the operation INT_COUNTER becomes zero, interrupts selected with INT_ENABLE are restored. An interrupt routine should always write to this register as the last thing it does, because interrupts automatically add one to the interrupt counter, but subtracting it back to its initial value is the responsibility of the user. It may take upto 6 clock cycles before writing this register has any effect.

By reading INT_COUNTER the user may check if the interrupt counter is correct or not. If the register is not 0, interrupts are disabled.

7.7 Universal Asynchronous Receiver/Transmitter (UART)

RS232 UART implements a serial interface using rs232 standard.

Figure 3: RS232 serial interface protocol

When the line is idling, it stays in logic high state. When a byte is transmitted, the transmission begins with a start bit (logic zero) and continues with data bits (LSB first) and ends up with a stop bit (logic high). 10 bits are sent for each 8-bit byte frame.

7.7.1 UART Registers

UART registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC028	r	0	UART_STATUS[4:0]	Status
0xC029	r/w	0	UART_DATA[7:0]	Data
0xC02A	r/w	0	UART_DATAH[15:8]	Data High
0xC02B	r/w	0	UART_DIV	Divider

7.7.2 Status UART_STATUS

A read from the status register returns the transmitter and receiver states.

UART_STATUS bits		
Name	**Bits**	**Description**
UART_ST_FRAMEERR	4	Framing error (stop bit was 0)
UART_ST_RXORUN	3	Receiver overrun
UART_ST_RXFULL	2	Receiver data register full
UART_ST_TXFULL	1	Transmitter data register full
UART_ST_TXRUNNING	0	Transmitter running

UART_ST_FRAMEERR is set if the stop bit of the received byte was 0.

UART_ST_RXORUN is set if a received byte overwrites unread data when it is transferred from the receiver shift register to the data register, otherwise it is cleared.

UART_ST_RXFULL is set if there is unread data in the data register.

UART_ST_TXFULL is set if a write to the data register is not allowed (data register full).

UART_ST_TXRUNNING is set if the transmitter shift register is in operation.

7.7.3 Data UART_DATA

A read from UART_DATA returns the received byte in bits 7:0, bits 15:8 are returned as '0'. If there is no more data to be read, the receiver data register full indicator will be cleared.

A receive interrupt will be generated when a byte is moved from the receiver shift register to the receiver data register.

A write to UART_DATA sets a byte for transmission. The data is taken from bits 7:0, other bits in the written value are ignored. If the transmitter is idle, the byte is immediately moved to the transmitter shift register, a transmit interrupt request is generated, and transmission is started. If the transmitter is busy, the UART_ST_TXFULL will be set and the byte remains in the transmitter data register until the previous byte has been sent and transmission can proceed.

7.7.4 Data High UART_DATAH

The same as UART_DATA, except that bits 15:8 are used.

7.7.5 Divider UART_DIV

UART_DIV bits		
Name	**Bits**	**Description**
UART_DIV_D1	15:8	Divider 1 (0..255)
UART_DIV_D2	7:0	Divider 2 (6..255)

The divider is set to 0x0000 in reset. The ROM boot code must initialize it correctly depending on the master clock frequency to get the correct bit speed. The second divider (D_2) must be from 6 to 255.

The communication speed $f = \frac{f_m}{(D_1+1)\times(D_2)}$, where f_m is the master clock frequency, and f is the TX/RX speed in bps.

Example divider values for common communication speeds at 49.152 MHz master clock:

Example UART speeds, $f_m = 49.152\,MHz$		
Comm. Speed [bps]	UART_DIV_D1	UART_DIV_D2
4800	255	40
9600	255	20
14400	233	15
19200	255	10
28800	243	7
38400	159	8
57600	121	7
115200	60	7

7.7.6 UART Interrupts and Operation

Transmitter operates as follows: After an 8-bit word is written to the transmit data register it will be transmitted instantly if the transmitter is not busy transmitting the previous byte. When the transmission begins a TX_INTR interrupt will be sent. Status bit [1] informs the transmitter data register empty (or full state) and bit [0] informs the transmitter (shift register) empty state. A new word must not be written to transmitter data register if it is not empty (bit [1] = '0'). The transmitter data register will be empty as soon as it is shifted to transmitter and the transmission is started. It is safe to write a new word to transmitter data register every time a transmit interrupt is generated.

Receiver operates as follows: It samples the RX signal line and if it detects a high to low transition, a start bit is found. After this it samples each 8 bit at the middle of the bit time (using a constant timer), and fills the receiver (shift register) LSB first. Finally the data in the receiver is moved to the reveive data register, the stop bit state is checked (logic high = ok, logic low = framing error) for status bit[4], the RX_INTR interrupt is sent, status bit[2] (receive data register full) is set, and status bit[2] old state is copied to bit[3] (receive data overrun). After that the receiver returns to idle state to wait for a new start bit. Status bit[2] is zeroed when the receiver data register is read.

RS232 communication speed is set using two clock dividers. The base clock is the processor master clock. Bits 15-8 in these registers are for first divider and bits 7-0 for second divider. RX sample frequency is the clock frequency that is input for the second divider.

7.8 Timers

There are two 32-bit timers that can be initialized and enabled independently of each other. If enabled, a timer initializes to its start value, written by a processor, and starts decrementing every clock cycle. When the value goes past zero, an interrupt is sent, and the timer initializes to the value in its start value register, and continues downcounting. A timer stays in that loop as long as it is enabled.

A timer has a 32-bit timer register for down counting and a 32-bit register for holding the timer start value written by the processor. Timers have also a 2-bit TIMER_ENA register. Each timer is enabled (1) or disabled (0) by a corresponding bit of the enable register.

7.8.1 Timer Registers

Timer registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC030	r/w	0	TIMER_CONFIG[7:0]	Timer configuration
0xC031	r/w	0	TIMER_ENABLE[1:0]	Timer enable
0xC034	r/w	0	TIMER_T0L	Timer0 startvalue - LSBs
0xC035	r/w	0	TIMER_T0H	Timer0 startvalue - MSBs
0xC036	r/w	0	TIMER_T0CNTL	Timer0 counter - LSBs
0xC037	r/w	0	TIMER_T0CNTH	Timer0 counter - MSBs
0xC038	r/w	0	TIMER_T1L	Timer1 startvalue - LSBs
0xC039	r/w	0	TIMER_T1H	Timer1 startvalue - MSBs
0xC03A	r/w	0	TIMER_T1CNTL	Timer1 counter - LSBs
0xC03B	r/w	0	TIMER_T1CNTH	Timer1 counter - MSBs

7.8.2 Configuration TIMER_CONFIG

TIMER_CONFIG bits		
Name	**Bits**	**Description**
TIMER_CF_CLKDIV	7:0	Master clock divider

TIMER_CF_CLKDIV is the master clock divider for all timer clocks. The generated internal clock frequency $f_i = \frac{f_m}{c+1}$, where f_m is the master clock frequency and c is TIMER_CF_CLKDIV. Example: With a 12 MHz master clock, TIMER_CF_DIV=3 divides the master clock by 4, and the output/sampling clock would thus be $f_i = \frac{12MHz}{3+1} = 3MHz$.

7.8.3 Configuration TIMER_ENABLE

TIMER_ENABLE bits		
Name	**Bits**	**Description**
TIMER_EN_T1	1	Enable timer 1
TIMER_EN_T0	0	Enable timer 0

7.8.4 Timer X Startvalue TIMER_Tx[L/H]

The 32-bit start value TIMER_Tx[L/H] sets the initial counter value when the timer is reset. The timer interrupt frequency $f_t = \frac{f_i}{c+1}$ where f_i is the master clock obtained with the clock divider (see Chapter 7.8.2 and c is TIMER_Tx[L/H].

Example: With a 12 MHz master clock and with TIMER_CF_CLKDIV=3, the master clock $f_i = 3MHz$. If TIMER_TH=0, TIMER_TL=99, then the timer interrupt frequency $f_t = \frac{3MHz}{99+1} = 30kHz$.

7.8.5 Timer X Counter TIMER_TxCNT[L/H]

TIMER_TxCNT[L/H] contains the current counter values. By reading this register pair, the user may get knowledge of how long it will take before the next timer interrupt. Also, by writing to this register, a one-shot different length timer interrupt delay may be realized.

7.8.6 Timer Interrupts

Each timer has its own interrupt, which is asserted when the timer counter underflows.

7.9 I2S DAC Interface

The I2S DAC Interface makes it possible to attach an external DAC to the system.

Note: The samplerate of the audio file and the I2S rate are independent. All audio will be automatically converted to 6.144 MHz for VS1063 DAC and to the configured I2S rate using a high-quality sample-rate converter.

Note: In VS1063a the I2S pins share different GPIO pins than in VS1033 to be able to use SPI boot and I2S in the same application.

7.9.1 I2S Registers

I2S registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC040	r/w	0	I2S_CONFIG[3:0]	I2S configuration

7.9.2 Configuration I2S_CONFIG

I2S_CONFIG bits		
Name	**Bits**	**Description**
I2S_CF_MCLK_ENA	3	Enables the MCLK output (12.288 MHz)
I2S_CF_ENA	2	Enables I2S, otherwise pins are GPIO
I2S_CF_SRATE	1:0	I2S rate, "10" = 192, "01" = 96, "00" = 48 kHz

I2S_CF_ENA enables the I2S interface. After reset I2S is disabled and the pins are used for GPIO inputs.

I2S_CF_MCLK_ENA enables the MCLK output. The frequency is either directly the input clock (nominal 12.288 MHz), or half the input clock when mode register bit SM_CLK_RANGE is set to 1 (24-26 MHz input clock).

I2S_CF_SRATE controls the output samplerate. When set to 48 kHz, SCLK is MCLK divided by 8, when 96 kHz SCLK is MCLK divided by 4, and when 192 kHz SCLK is MCLK divided by 2. I2S_CF_SRATE can only be changed when I2S_CF_ENA is 0.

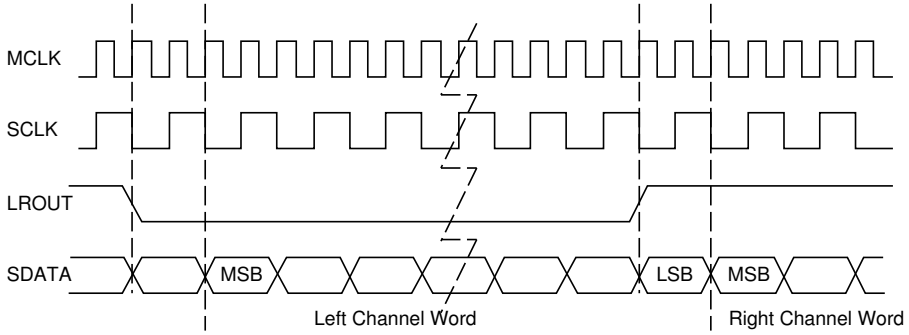


Figure 4: I2S interface, 192 kHz.

To enable I2S first write 0xc017 to SCI_WRAMADDR and 0x00f0 to SCI_WRAM, then write 0xc040 to SCI_WRAMADDR and 0x000c to SCI_WRAM.

7.10 Analog-to-Digital Converter (ADC)

7.10.1 ADC Registers

ADC modulator registers control Analog-to-Digital conversions of VS1063a.

ADC Decimator registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC042	rw	0	ADC_CONTROL[4:0]	ADC control
0xC043	r	0	ADC_DATA_LEFT	ADC left channel data
0xC044	r	0	ADC_DATA_RIGHT	ADC right channel data

7.10.2 Control ADC_CONTROL

ADC_CONTROL controls the ADC and its associated decimator unit.

ADC_CONTROL bits		
Name	**Bits**	**Description**
ADC_MODU2_PD	4	Right channel powerdown
ADC_MODU1_PD	3	Left channel powerdown
ADC_DECIM_FACTOR	2:1	ADC Decimator factor: - 3 = downsample to XTALI/512 (nominal 24 kHz) - 2 = downsample to XTALI/256 (nominal 48 kHz) - 1 = downsample to XTALI/128 (nominal 96 kHz) - 0 = downsample to XTALI/64 (nominal 192 kHz)
ADC_ENABLE	0	Set to activate ADC converter and decimator

Note: Setting bit SS_AD_CLOCK in register SCI_STATUS will halve the operation speed of the A/D unit, and thus halve the resulting samplerate.

7.10.3 ADC Interrupts

Each time a new (stereo) sample has been generated, an ADC interrupt is generated.

7.11 Resampler SampleRate Converter (SRC)

The resampler SRC makes it possible to catch audio from the DAC path.

Note: hardware makes no attempts at low-pass filtering data. If the SRC samplerate is lower than the DAC samplerate, aliasing may and will occur.

7.11.1 SRC Registers

Resampler SRC registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC046	rw	0	SRC_CONTROL[12:0]	SRC control
0xC047	r	0	SRC_DATA_LEFT	SRC left channel data
0xC048	r	0	SRC_DATA_RIGHT	SRC right channel data

7.11.2 Control SRC_CONTROL

SRC_CONTROL Bits		
Name	**Bits**	**Description**
SRC_ENABLE	12	Set to enable SRC
SRC_DIV	11:0	Set samplerate to XTALI/2/(SRC_DIV+1)

7.11.3 SRC Interrupts

Each time a new (stereo) sample has been generated, an SRC interrupt is generated.

7.12 Sidestream Sigma-Delta Modulator (SDM)

The Sidestream Sigma-Delta Modulator makes it possible to insert a digital side stream on top of existing audio.

Note: The SDM provides a direct, low-delay side channel to the Sigma-Delta DACs of VS10xx. It makes no attempts at low-pass filtering data. Thus there will be practically no image rejection. If using low samplerates, this may cause audible aliasing distortion.

7.12.1 SDM Registers

Sidestream SDM registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC049	rw	0	SDM_CONTROL[12:0]	SDM control
0xC04A	rw	0	SDM_DATA_LEFT	SDM left channel data
0xC04B	rw	0	SDM_DATA_RIGHT	SDM right channel data

7.12.2 Control SDM_CONTROL

SDM_CONTROL Bits		
Name	**Bits**	**Description**
SDM_ENABLE	12	Set to enable SDM
SDM_DIV	11:0	Set samplerate to XTALI/2/(SDM_DIV+1)

7.12.3 SDM Interrupts

Each time a new (stereo) sample is needed, an SDM interrupt is generated.

8 Latest Document Version Changes

This chapter describes the latest and most important changes to this document.

Version 1.15, 2014-12-19

- Updated telephone number in Chapter 9, *Contact Information*.

Version 1.14, 2014-11-13

- Added GPIO_DDR[9:8] and GPIO_ODATA[8] explanation to Section 7.5, *General-Purpose Input/Output (GPIO)*.
- Other, minor changes.

Version 1.02, 2012-12-05

- Minor corrections.
- Removed preliminary status.

Version 0.40, 2011-09-02

- Minor corrections.

Version 0.30, 2011-05-05

- First published release of the guide.

9 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

URL: http://www.vlsi.fi/
Phone: +358-50-462-3200
Commercial e-mail: sales@vlsi.fi

For technical support or suggestions regarding this document, please participate at
http://www.vsdsp-forum.com/
For confidential technical discussions, contact
support@vlsi.fi