STA310 Getting Started

TPA Meylan Center - Digital Audio Group		
Version:	1.3	Aug 07 2001
Author :	Olivier GERMAIN	
File :	sta310_an.fm	

STA310 Getting Started

Updates follow-up :

Version	Date	Author	Comments
1.0	May 31 2001	O. GERMAIN	Draft
1.1	30 July 2001	O. GERMAIN	Add Software programmation for decoders
1.2	07 Aug 2001	O. GERMAIN	Add Hardware description of host parallel interface and I2C slave protocol
1.3	10 Aug 2001	O. GERMAIN	Add Description for Main I2s input

1

1) Introduction

1.1) Notation used in this document.

-If not especially mentioned any reference to a register (address and content) in this document is in decimal value.

- All the registers desribed in this document can be both accessed with the host parallel bus interface or with I2C commands.

Some C notations may be used to easier the understanding of this document.

- The function *void write_host_reg(BYTE addr, BYTE data)* will be used to described a write into a specific STA310 register.

The function *BYTE read_read_reg(BYTE)* will be used to describe a read from a specific STA310 register.

1.2) STA310 Initialisation Diagram After H/W reset

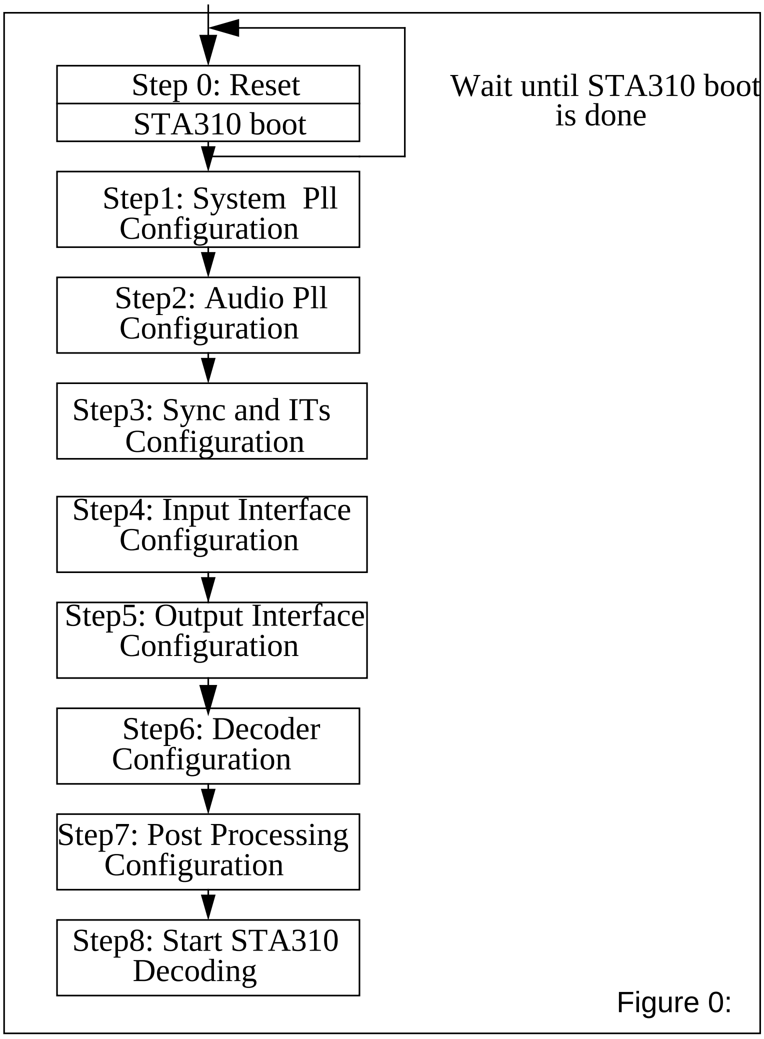

Step 0: Reset
STA310 boot
Wait until STA310 boot is done
Step1: System Pll Configuration
Step2: Audio Pll Configuration
Step3: Sync and ITs Configuration
Step4: Input Interface Configuration
Step5: Output Interface Configuration
Step6: Decoder Configuration
Step7: Post Processing Configuration
Step8: Start STA310 Decoding

Figure 0:

Configuration example0:
The following sequence configure the STA310 after a H/W reset with the following options:

- Ac3 Decoder
- Main I2s input
- Audio Pll generate 256xFS oversampling clock
- 24 bit DAC
- Spdiff output in L/R mode
- No post processing
- No bass managemenent
- Volume Attenuation 10 dB

(Step 0)
Decoder Soft Reset

HOST: Write (01) at @(16)
HOST: Write (08) at @(43)
HOST: Write (00) at @(58)
HOST: Read (01) at @(255)
HOST: Read (01) at @(255)
HOST: Write (01) at @(181)

(Step1)
System Pll Configuration

HOST: Write (03) at @(29)
HOST: Write (209) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (04) at @(29)
HOST: Write (94) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (06) at @(29)
HOST: Write (01) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (07) at @(29)
HOST: Write (00) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (08) at @(29)
HOST: Write (09) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)

```
HOST: Write (09) at @(29)
HOST: Write (01) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (04) at @(24)
HOST: Write (00) at @(24)
HOST: Write (09) at @(29)
HOST: Write (00) at @(17)
HOST: Write (02) at @(24)
HOST: Write (00) at @(24)
HOST: Write (04) at @(24)
HOST: Write (00) at @(24)
```

(Step2)
Audio Pll Configuration

```
HOST: Write (52) at @(182)
HOST: Write (236) at @(183)
HOST: Write (02) at @(184)
HOST: Write (09) at @(185)
HOST: Write (01) at @(186)
HOST: Write (03) at @(187)
HOST: Write (09) at @(188)
HOST: Write (02) at @(189)
HOST: Write (09) at @(190)
HOST: Write (01) at @(191)
HOST: Read (25) at @(18)
HOST: Write (26) at @(18)
```

(Step 3)
Configure ITs

```
HOST: Write (01) at @(08)
HOST: Write (12) at @(07)
```

Configure Sync

```
HOST: Write (00) at @(79)
HOST: Write (00) at @(83)
HOST: Write (00) at @(80)
HOST: Write (00) at @(81)
HOST: Write (00) at @(82)
```

(Step4)

I2s Input Configuration

HOST: Write (17) at @(12)

(Step5)
Pcmout and DAC Configuration

HOST: Write (11) at @(85)
HOST: Read (11) at @(85)
HOST: Write (01) at @(84)
HOST: Read (33) at @(96)
HOST: Write (33) at @(96)

IEC958 Ouput Configuration

HOST: Read (33) at @(96)
HOST: Write (33) at @(96)
HOST: Write (02) at @(94)
HOST: Read (00) at @(114)
HOST: Write (00) at @(95)
HOST: Write (18) at @(97)

(Step6)
Configure Decoder for Ac3

HOST: Write (00) at @(77)
HOST: Write (04) at @(117)
HOST: Write (03) at @(76)

Configure AC3 Decoder Parameters
HOST: Write (01) at @(104)
HOST: Write (02) at @(105)
HOST: Write (255) at @(106)
HOST: Write (255) at @(107)
HOST: Write (00) at @(108)
HOST: Write (07) at @(111)
HOST: Write (03) at @(109)
HOST: Write (00) at @(110)

(Step7)
Configure Effect Type

HOST: Read (00) at @(101)
HOST: Read (00) at @(87)
HOST: Read (00) at @(88)

```
HOST: Read (00) at @(89)
HOST: Read (00) at @(90)
HOST: Read (00) at @(91)
HOST: Read (00) at @(92)
HOST: Read (00) at @(175)
HOST: Read (00) at @(176)
HOST: Write (00) at @(177)
HOST: Write (00) at @(98)
```

Reset Effect Parameter

```
HOST: Read (04) at @(100)
HOST: Write (04) at @(100)
HOST: Write (00) at @(101)
HOST: Write (00) at @(171)
HOST: Write (00) at @(178)
HOST: Write (00) at @(179)
```

Configure Bass Management

```
HOST: Write (00) at @(102)
```

Configure Volume

```
HOST: Write (00) at @(102)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (00) at @(103)
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (01) at @(103)
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (02) at @(103)
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
```

HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Read (00) at @(115)
HOST: Write (00) at @(115)

Configure Channel Cross

HOST: Write (00) at @(86)
HOST: Read (128) at @(127)
HOST: Write (128) at @(127)

(Step 8)
Run and Start Playing music

HOST: Write (01) at @(19)
HOST: Write (00) at @(20)
HOST: Write (01) at @(114)

1.3) Initialization diagram after a software reset

Software reset of the STA310 must be done after each change of alogrythm by the external MCU.The denomination "software reset" do not involve only the SOFTRESET register (@0x10) but a whole sequence descibed in the "step0" sequence. This sequence is part of a overall "warm reset" sequence that reconfigure the STA310 to decode the new algorithm. The "step0" sequence, performs the following action in the STA310:

The following configurations are really reset:

- Volume Control
- Delay Channel
- Karaoke
- Synchro commands (skip, mute, pause, ...)
- external interrupts

The following configurations keep their previous values:

- Decoders
- Post Processing
- VCR mix
- Bass managment
- spdiff mode

When the spdiff input is used in compressed mode the STA310 is able to autodetect a change of algorithm with DTS, MPEG and Dolby Ac3. In this case none of the previous configurations are reset. Only PCM buffer and decoder internal variables are cleared.

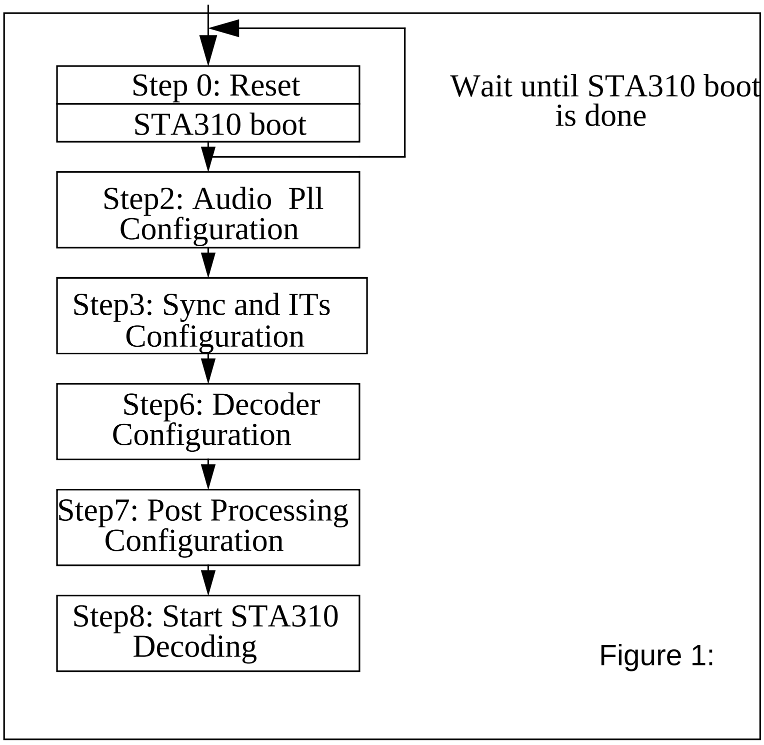

Step 0: Reset
STA310 boot
Wait until STA310 boot is done
Step2: Audio Pll Configuration
Step3: Sync and ITs Configuration
Step6: Decoder Configuration
Step7: Post Processing Configuration
Step8: Start STA310 Decoding

Figure 1:

Configuration example0:
The following sequence configures the STA310 after a soft reset. The algorithm has been changed from Ac3 to Mp3 keeping the previous configuration (from H/W reset):

STA310 Warm Reset
(Step 0)

Soft Mute

HOST: Read (00) at @(115)
HOST: Write (01) at @(115)

Decoder Soft Reset

HOST: Write (01) at @(16)
HOST: Write (08) at @(43)
HOST: Write (00) at @(58)
HOST: Read (01) at @(255)

```
HOST: Read (01) at @(255)
HOST: Write (01) at @(181)
```

(Step2)
Audio Pll Configuration

```
HOST: Write (52) at @(182)
HOST: Write (236) at @(183)
HOST: Write (02) at @(184)
HOST: Write (09) at @(185)
HOST: Write (01) at @(186)
HOST: Write (03) at @(187)
HOST: Write (09) at @(188)
HOST: Write (02) at @(189)
HOST: Write (09) at @(190)
HOST: Write (01) at @(191)
HOST: Read (25) at @(18)
HOST: Write (26) at @(18)
```

(Step 3)
Configure ITs

```
HOST: Write (01) at @(08)
HOST: Write (12) at @(07)
```

Configure Sync

```
HOST: Write (00) at @(79)
HOST: Write (00) at @(83)
HOST: Write (00) at @(80)
HOST: Write (00) at @(81)
HOST: Write (00) at @(82)
```

(Step6)
Configure Decoder for Mp3

```
HOST: Write (03) at @(76)
HOST: Write (09) at @(77)
```

(Step7)
Configure Volume

```
HOST: Write (00) at @(102)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (00) at @(103)
```

```
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (01) at @(103)
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Write (04) at @(78)
HOST: Write (04) at @(99)
HOST: Write (02) at @(103)
HOST: Read (00) at @(114)
HOST: Read (00) at @(20)
HOST: Write (00) at @(20)
HOST: Read (04) at @(103)
HOST: Read (04) at @(103)
HOST: Read (00) at @(115)
HOST: Write (00) at @(115)
```

(Step 8)
Run and Start Playing music

```
HOST: Write (01) at @(19)
HOST: Write (00) at @(20)
HOST: Write (01) at @(114)
```

1.4) What can be changed on the fly?

The following paramters can be change on the fly, while the STA310 is decoding:

- Volume Control
- Bass management
- Post processing
- Algorithm options
- Channels delays
- Channels crossing
- Synchro Commands (PACKET_LOCK, SYNC_LOCK)
- PlayStop commands (skip, mute, pause, ...
- Status registers can be read and are updated on a frame basis

2) Step 0: How to reset the STA310

Reseting the STA310 is done with the following set of commands :

```
// Soft Reset
- write_host_reg(16, 1)
// Disable use of memory from the emulation unit
- write_host_reg(43, 8)
// Start the clock
- write_host_reg(58, 0)
// wait until boot from the STA310 is done
while (read_host_reg(255) != 1);

// Enable use of fractionnal Audio Pll
- write_host_reg(181, 1)
```

You can then check the version and identity register (hardware coded) to see if you access correctly to the STA310:

```
==> read_host_reg(0) = 16
==> read_host_reg(1) = 49
```

3) Audio and System PLLs Configuration

3.1) Block Diagram

The Audio and System PLLs are two fractional PLLs. They are controlled, via the host registers, as explained in the following sections

The following is the block diagram of the PLL used for both the Audio and System clock generation.

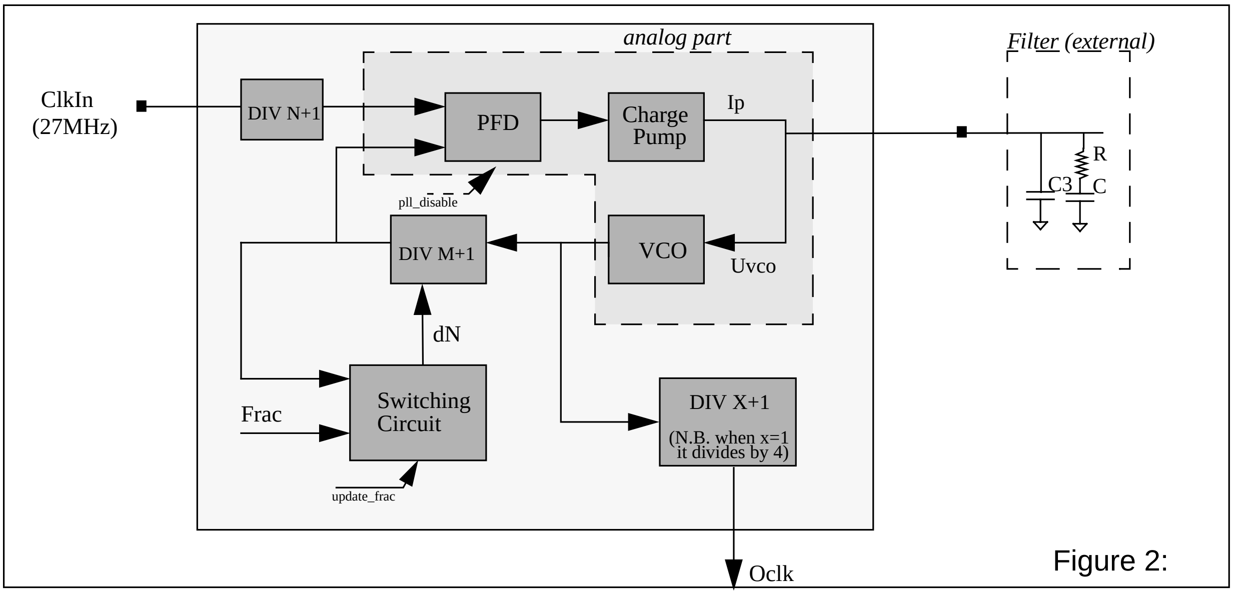


Figure 2:

The equation implemented by the PLL is the following[1]:

$$Oclk = \frac{InputFreq}{N+1} \times \left(M + 1 + \frac{Frac}{65536}\right) \times \frac{1}{X+1}$$

Figure 3:

The N, M, X and Frac coefficients are passed to the PLLs registers through an indirect mechanism, via the Host Registers. .

The Phase Detector input must be in the range of 4Mhz to 14 Mhz. This affect the choice to the N divider; i.e. with the input clock at 27Mhz, we have to set N=1.

The Voltage Controlled Oscillator can generates on output signal in the range of 100Mhz to 200Mhz. This affect the choice of M and Frac coefficients.

The PLL coefficients must be chosen in such a way to respect the PD and VCO ranges.

[1] ***There is an exception when X=1; it divides by 4 (not by 2!!)***

3.2) Step1: System Pll Configuration

The system clock sent to the DSP core and the peripherals can be derived from 4 sources and the selection is performed through the bit 4 and 3 of ***HOST_PllCtrl*** register (@18):

- 00 ***sys_clk*** = external clk
- 01 ***sys_clk*** = external clk divided by 2
- 10 ***sys_clk*** = pll clock
- 11 ***sys_clk*** = pll_clk / 2

At reset the configuration is 11, i.e. ***sys_clk*** is the from System PLL output divided by 2.

In the **STA310** chip, the duty cycle of the internal system clock must be as close as possible to 50%.

It’s recommended to use the divider by 2 in order to ensure a 50% duty cycle, in both the cases, i.e. when the ***sys_clk*** is derived from the external clock source (**CLK**), or from the internal PLL.

The ***CLKOUT*** pad is driven by the ***sys_clk*** divided by a programmable division factor ranging from 1 to 16.
Clkout = Sysclk / (2 * (pad_div + 1))

At reset, assuming an external clock of 27 Mhz, the system PLL output ***Oclk*** is at 47.25Mhz. (N=1, M=13, X=1, Frac=0), which is further divided by 2 to obtain the final sys_clk of 23.625Mhz.

The clockout pad (after the programmable divider which is 3 at reset) will be 2.95 Mhz!

If the input clock is different than 27Mhz, or if you want to run at a higher frequency, you have to configure the system PLL in order to obtain the required frequency.

You can obtain the new coefficients (N,M,X, Frac), applying the formula specified in figure3. See section 3.4 for updating the PLL coefficients via the indirect mechanism. Here some examples of coefficients.

Table 1: Examples of PLL Coeff. for 27Mhz Input clock

N	M	Frac	X	Oclk (Mhz)
1	11	0	1	40.5
1	13	0	1	47.25
1	7	0	0	108.0
1	9	0	0	135.0

3.3) Step2: Audio Pll Configuration

The audio clock can be derived from 3 different sources and the selection is performed through the bit 2,1, 0 of ***HOST_PllCtrl*** register (@18):

- -01 pcm_clk = from external pll, PCMLCK pad is in input mode
- 010 pcm_clk = from audio pll, PCMLCK pad is in output mode.
- 110 pcm_clk = from SPDIF input, PCMCLK pad is in output mode

Note:

At reset the configuration is 001, i.e. pcm_clk from PCMCLK PAD.

If the pcm_clk is derived from the SPDIFIN, the only supported oversampling factor is 256 Fs.

Programmation of the Audio Pll is done with the **N, M, X** and **Frac** coefficients. They are loaded into the PLL registers through a indirect mechanism automatically by the STA310 software, using two sets of pre-defined coefficients; one set for the 44.1Khz frequencies family (11.025, 22.05, 44.1, 88.2, 176.4) and one set for the 48Khz (12, 24, 48, 96, 192; 8, 16, 32, 64, 128). The STA310 software will automatically switch from one set of coefficients to another to generate the correct PCMCLK, LRCLK, and SCLK according to the bitstream frequency.

Unlike the system PLL, the Audio PLL can be programmed to the desired PCMLK value without having to write directly to the PLL register (using the indiretion mechanism). 10 dedicated ***Reference Registers*** are used to store the coefficients for the PLL.

Table2 gives the default coefficents values loaded after boot by the STA310 software. These coefficients are valid under the assumption that the input clock is @ 27Mhz and the external DACs accept an oversampling factor of 384.

.

Table 2: Reference Registers for the Audio PLL

	Add. (dec)	Name	Value at reset
Coefficients for: 384 * 176.4 Khz	187	***HOST_fracl_ref441k***	3
	188	***HOST_frach_ref441k***	9
	189	***HOST_xdiv_ref441k***	1
	190	***HOST_mdiv_ref441k***	9
	191	***HOST_ndiv_ref441k***	1

Table 2: Reference Registers for the Audio PLL

	Add. (dec)	Name	Value at reset
Coefficients for: 384 * 192 Khz	182	***HOST_fracl_ref48k***	52
	183	***HOST_frach_ref48k***	236
	184	***HOST_xdiv_ref48k***	1
	185	***HOST_mdiv_ref48k***	9
	186	***HOST_ndiv_ref48k***	1

At *power-on*, the 48Khz family coefficients are automatically loaded by the software into the audio PLL, modifying appropriately the ***X divider*** in order to generate the ***pcm_clk*** at 384* 48 Khz. (N = 1, M = 9, X = 7, Frac = 60468).

At *run-time*, the software of the STA310 will set the audio PLL at the right frequency(according to the input data stream) loading these predefined registers and changing appropriately only the ***X divider***.

If either the input clock or the oversampling factor are different in your application, the 2 sets of ***Reference Registers*** (table2) must be updated and loaded at power-on and after a soft reset.

These ***Reference Registers*** must be configured in such a a way that, given the input clock frequency and oversampling factor of your application, the following equations are respected (for 44.1Khz and 48Khz respectively):

.

$$\frac{NewInputFreq}{N+1} \times \left(M + 1 + \frac{Frac}{65536}\right) \times \frac{1}{X+1} = 176.4 Khz \times NewOversampling$$

Figure 4:

$$\frac{NewInputFreq}{N+1} \times \left(M + 1 + \frac{Frac}{65536}\right) \times \frac{1}{X+1} = 192 Khz \times NewOversampling$$

Figure 5:

Write directly in the 10 ***Reference Registers*** (table2) the appropriate values, via the usual host register mechanism.
The software will exploit them appropriately to configure the audio PLL as explained above.

If the Sampling Frequency is greater than 64Khz, the software will automatically set the oversampling fcator to 128 (regardless of its previous value).

Configuration example 1:

- Use STA310 Audio pll to generate the PCMLCK
- ***PCMCLK*** at 256xFS
- External Clock reference of 27 Mhz

```
// Configure FRACL coefficient for 256x192 khz
write_host_reg(182, 52);
// Configure FRACH coefficient for 256x192 khz
write_host_reg(183, 236)
// Configure X coefficient for 256x192 khz
write_host_reg(184, 2);
// Configure M coefficient for 256x192 khz
write_host_reg(185, 9);
// Configure N coefficient for 256x192 khz
write_host_reg(186, 1);

// Configure FRACL coefficient for 256x176.4 khz
write_host_reg(187, 3);
// Configure FRACH coefficient for 256x176.4 khz
write_host_reg(188, 9)
// Configure X coefficient for 256x176.4 khz
write_host_reg(189, 2);
// Configure M coefficient for 256x176.4 khz
write_host_reg(190, 9);
// Configure N coefficient for 256x176.4 khz
write_host_reg(191, 1);

// Configure PCMCLK pad in output, use audio pll
// SysClk from System PLL divided by 2
write_host_reg(18, 26);
```

Configuration example 2: The PCMLCK is provided by an external PLL

```
write_host_reg(18, 1);
```

3.4) Indirect mechanism

The PLL registers can be configured through an indirection mechanism using the ***HOST_pll_add***, ***HOST_pll_data*** and ***HOST_pll_cmd*** registers. This way must be avoid as much as possible and reserved only to program the system PLL. Use the dedicated ***Reference Registers (table 2)*** for the Audio Pll.

Table 3: PLLs Indirect Registers Address

@ (dec)	Name	Size	Mode	Soft Reset	Hard Reset	Comment
17	HOST_pll_data	8	R/W	NC	0	Data register to configure the different PLL configuration registers
24	HOST_pll_cmd	8	R/W	NC	0	command register for the configuration of the PLLs
29	HOST_pll_add	8	R/W	NC	0	address register to access the different PLL configuration registers

The ***HOST_pll_cmd*** register will allow to update the coefficients of the System PLL [1].

There are two levels of registers (Level 1 & Level 2).

The first level of registers (Level 1) is configured through the indirection mechanism (see an example below)

The second level of registers (Level 2) is a copy of the previous level in order to update, for a given PLL, all the configuration bits at the same time.

This mechanism avoids to have, during the configuration phase, intermediate configurations that are not in line with the final desired configuration.

The mapping of the ***HOST_pll_cmd*** register is the following

:

Table 4: PLLs Indirect Registers command

HOST_pll_cmd	field	Comment
	[1:0]	**00**: no action is performed on the configuration registers of the Level 1. **01**: Read action of the configuration registers. During this phase, the contains of a selected (by HOST_pll_add) configuration register of the Level 1 is copied into the HOST_pll_data register. **10**: Write action of the configuration registers. During this phase, the contains of the HOST_pll_data register is copied into a selected (by HOST_pll_add) configuration register of the Level 1. **11**: do not use.

[1] You can theoretically update in the same way the Audio PLL coefficients, but they will be overwritten by the software when running a decoder

Table 4: PLLs Indirect Registers command

HOST_pll_cmd	field	Comment
	[2]	This bit controls the transfer of the data between the Level 1 and the Level 2 for the System PLL. When this bit is set, all the registers of the Level 1 (*sys_ndiv, sys_mdiv, sys_sdiv, sys_pe_l, sys_pe_h, sys_disable*) are copied into the registers of the Level 2 at the same time. When this bit is cleared, all the Level 2 registers have a stable state independently of the Level 1 registers.
	[3]	This bit controls the transfer of the data between the Level 1 and the Level 2 for the Audio PLL. When this bit is set, all the registers of the Level 1 (*aud_ndiv, aud_mdiv, aud_pe_l, aud_pe_h, aud_sdiv, aud_disable*) are copied into the registers of the Level 2 at the same time. When this bit is cleared, all the Level 2 registers have a stable state independently of the Level 1 registers.

The Level 2 registers addresses to pass to ***HOST_pll_add*** are the following:

Table 5: PLLs Indirect Registers sub Addres

Address	Name	Address	Name
2	Disable System PLL	10	Disable System PLL
3	System PLL Frac Low	11	Audio PLL Frac Low
4	System PLL Frac High	12	Audio PLL Frac High
6	System PLL N divider	14	Audio PLL N divider
7	System PLL X divider	15	Audio PLL X divider
8	System PLL M divider	16	Audio PLL M divider
9	Update System Pll	17	Update AudioPll

Configuration example1: Let's suppose that we want to set the ***sys_clk*** to run @ 70 Mhz;

Since, we will use the PLL clock divided by 2, we have to configure the PLL in order to generate a 140 Mhz signal.

- ***System clock at 140 Mhz***
- External Clock reference of 27 Mhz:
- system clock = pll clock / 2

We Can define 2 other fucntions that perform write and read to/from a PLL register:

```
/* ------------------------------------------------------------------
 * function :       write_syspll_reg
 * parameters:      PLL address register, PLL data register
 * returned value:  none
 * Write into the System PLL register through indirection mechanism
 * ---------------------------------------------------------------*/

 void write_syspll_reg (UBYTE add, UBYTE data)
{
      write_host_reg (29, add);
      write_host_reg (17, data);
      write_host_reg (24, 2);
      write_host_reg (24, 0);
}

/* ----------------------------------------------------------------
 * function :       read_syspll_reg
 * parameters:      PLL address register, PLL data register
 * returned value:  none
 * Read From the System PLL register through indirection mechanism
 * ---------------------------------------------------------------*/

 void read_syspll_reg (U8 add, U8 *data)
{
      write_host_reg (29, add);
      write_host_reg (24, 5);
      *data = read_host_reg (17);
      write_host_reg (24, 0);
}
```

The Configuration for 70Mhz is then:

```
// Update System Pll FRACL coefficient in level 1
write_syspll_reg(3, 209)
// Update System Pll FRACH coefficient in Level 1
write_syspll_reg(4, 94)
// Update System Pll N coefficient in Level 1
write_syspll_reg(6, 1)
// Update System Pll X coefficient in Level 1
write_syspll_reg(7, 0)
// Update System Pll M coefficient in Level 1
write_syspll_reg(8, 9)
// Update System Pll update in Level 1
write_syspll_reg(9, 1)

// Transfert from Level 1 to Level 2
write_host_reg(24, 4)
write_host_reg(24, 0)

// Update System Pll update in Level 1
write_syspll_reg(9, 0)

// Transfert from Level 1 to Level 2
write_host_reg(24, 4)
write_host_reg(24, 0)
```

Note: The update of the PLL is done through the register 9 “update Pll”

```
// SysClk from System PLL divided by 2
write_host_reg(18, 26);
```

3.5) *HOST_PllCtrl* (@18) Configuration Summary

Table 6: Register @18

4	3	2	1	0	Name	PCMCLK PAD
		-	0	1	***pcm_clk*** from PAD	Input
		0	1	1	***pcm_clk*** from Audio PLL	
		1	1	1	***pcm_clk*** from SPDIFIN	
		-	0	0	***Forbidden***	Output
		0	1	0	***pcm_clk*** from Audio PLL	
		1	1	0	***pcm_clk*** from SPDIFIN	
0	0				***sys_clk*** from PAD	
0	1				***sys_clk*** from PAD divided by 2	
1	0				***sys_clk*** from System PLL	
1	1				***sys_clk*** from System PLL divided by 2	

4) Step3: How to configure the Audio Input Interface

The STA310 can get receive an input bitstream either from the I2s input or from the Spdif input, the selection and the configuration is done through 2 registers SIN_SETUP @12 and CAN_SETUP @13

Table 7: Input I2S Configuration

@ hexa	@ dec	Name	Size	Mode	Reset	Comment
0x0C	12	sin_setup	4	R/W	0	input data set up __bits 0-1__ : **0** Paralle input (Dstrb + Data[7:0] + Req) **1** Serial input (Dstrb + Sin + Req) **2** Not used **3** A/D input (Dstrb + Lrclkin + Req + Sin) __bit 2__ : Polarity of the Req signal When high the "REQ" pin meaning is inverted, and the data must be input when "REQ" is "0". When low , the data must be input when "REQ" is "1". __bit 3__: data from SPDIFIN **1**, or from pads **0**
0x0D	13	can_setup	4	R/W	0	A/D converter setup used when sin_setup[1:0]=3 **0** When set the Lrclkin is delay of one cycle (Padding mode) **1** When set the first channel (Left) is with Lrclkin= 1 **2** When set , data are sampled on falling edge of the bit-clock (dstrb) **3** When clear slot count is 32 but only the 16th are extracted

Configuration example 1: How to use the spdiff input of the STA310 ?

If you want to use the spdifin input the register can_setup musy be forced to 0 and sin_setup must be forced to the A/D mode with the spdif bit enable.

```
=> write_host_reg(12, 11);
=> write_host_reg(13, 0);
// The PCMLCK must be provided by the internal spdif pll and the pad is configured in output
=> write_host_reg(18, 30);
```

Configuration example 2: STA310 connected to a STi55xx evaluation board

```
// Serial Input, data input when req is low
=> write_host_reg(12, 5);
```

Configuration example 3: STA310 connected to an external I2s source (STA120 for example)
This configuration can be used when connecting a STA120 or a ADC in I2s mode to the STA310:

=> write_host_reg(12, 3);
=> write_host_reg(13, 1);

5) Step4: Synchronization features and Interrupts Configuration

5.1) Interrupt Status registers configuration

The STA310 has the possibility to send a interrupt to an external MCU when a registred event occurs during decoding. Two status registers (@0x08 and @ 0x07) and 2 interrupt enable registers (0x09, 0x0A) are used to respectively signal and enable events notification. Cf Datasheet for a description of the registers.

Table 8: Example of interrupt configuration(1)

Name	Address	value	
aud_intel	0x07	0x07	Notification of new header, error detected and change in synchronisation
aud_inteh	0x08	0x00	No Norification

Once an interrupt has been received on the IRQB pin, the MCU must check which status bit has been set in the status registers @0x09 and @0x0A and read the associated bench of registers to both acknowledge the interrupt and retrieve the information notified. The table below lists the associated registers corresponding to each of the event available:

Table 9: Example of interrupt configuration(2)

Interrupt register Bitfiled (@0x09, @0x0A)	Description	List of available registers
SYN	Change in synchronization status register	Register @0x40. Reading this register clear the interrupt bit SYN (bit0) in status register 0x09

Table 9: Example of interrupt configuration(2)

Interrupt register Bitfiled (@0x09, @0x0A)	Description	List of available registers
HDR	valid header registred	Registers @0x42, @0x43, @0x44, @0x45. Reading register @0x42 clear the interrupt bit HDR (bit1) in status register 0x09
ERR	Error Detected	Register @0x0F. Reading this register clear the interrupt bit ERR (bit2) in status register 0x09
SFR	Sampling frequency changed	Register @0x05. Reading this register clear the interrupt bit ERR (bit3) in status register 0x09
DEM	Deemphasis changed	Register @0xB5. Reading this register clear the interrupt bit ERR (bit4) in status register 0x09
BOF		
PTS	New PTS data registred	Registers @0x46,->@0x4A. Reading register @0x46 clear the interrupt bit HDR (bit6) in status register 0x09
ANC	Ancillary data registered	Register @0xB5. Reading this register clear the interrupt bit ERR (bit4) in status register 0x09

5.2) Synchronization registers:

AUD_ID_EN, AUD_ID, AUD_ID_EXT(@0x50, @0x51, @0x52) can be used when decoding packets to specify an identifier for a selected program.

Note:
When Elementary stream are decoded it is unusefull.

AUD_PACKET_LOCK and AUD_SYNC_LOCK are used to increase the robustness to the synchronisation. The following defaults value can be used:

- write_host_reg(0x4F, 0);
- write_host_reg(0x50, 0);
- write_host_reg(0x51, 0);
- write_host_reg(0x52, 0);
- write_host_reg(0x53, 0);

6) Step5: How to configure the audio output interface

6.1) How to configure the PCM output interface

Table 10: Output I2S Configuration

@ hexa	@ dec	Name	Size	Mode	Reset	Comment
0x54	84	PcmDivider	8	R/W	No	Divider fro PCm clock *Sclk= PCM_CLK / (2 x (DIVIDER+1))* *if DIVIDER =0 => SCLK= PCMCLK*
0x55	85	PcmConf	8	R/W	No	Pcm Configuration **bits 7:2** : **7** Not-Used **6** Pcm_Order **5** Pcm_Diff : 0 => rigth padded **4** Invert_LRCLK **3** Format 0=>I2c / 1 =>Sony **2** Invert_SCLK **bits 1:0** : Pcm_prec **0** : 16 bit mode (16 Slots) **1** : 18 bit mode (32 Slots) **2** : 20 bit mode (32 Slots) **3**: 24 bit mode (32 Slots)

The PCM block can be configurated to match almost any DAC on the market.

Configuration example 1: STA310 connected to a 24-bit DAC with I2s format (PCM1604) with an over-sampling factor of 384. The spdif input is not used

- 24-bit mode
- I2s data sampled on rising edge of SCLK
- Left channel corresponds to high level of the LRCLK and Right channel to the low level of the LRCLK
- I2s mode (synchro one bit before first data bit)
- Data are right padded (in the first SCLK cycle of LRCLK)
- MSB first

==> write_host_reg(85, 35)

- PCMCLK = 384xFS
- 32 slots/LRCLK
- SCLK = PCMCLK /(2 * (pcmdiv + 1)) = 64 x FS

==> write_host_reg(84, 2)

Configuration example 2: STA310 + 24-bit DAC with I2s format (PCM1604) + Spdif input (ofact is forced to 256xFS).

- 24-bit mode
- I2s data sampled on rising edge of SCLK
- Left channel corresponds to high level of the LRCLK and Right channel to the low level of the LRCLK
- I2s mode (synchro one bit before first data bit)
- Data are right padded (in the first SCLK cycle of LRCLK)
- MSB first

==> write_host_reg(85, 35)

- PCMCLK = 256xFS
- 32 slots/LRCLK
- SCLK = PCMCLK /(2 * (pcmdiv + 1)) = 64 x FS

==> write_host_reg(84, 1)

6.2) How to configure the SPDIF out interface

6.2.1) SPDIF configuration in IEC958 mode (PCM)

Configuration0: DAC: 256xFS and 24-bit , 44.1 khz file, L/R on spdiff:

The following group of registers must be written to configure the spdif in non compressed mode (IEC958). The Left/Right pair of channel is then sent on the SPDIF.

Table 11: IEC958 Configuration for Left/Right on the SPDIF

Name	Address	value	
spdif_conf	0x60	0x21	DAC: 256xFS and 24-bit , 16-bit rounding
spdiff_cat	0x5F	0	Category code General
spdif_status	0x61	0x02	Non compressed mode, copy allowed, no pre-emphasis, SF=44.1 khz
spdif_cmd	0x5E	2	Trasnmit L/R channels on the SPDIF
play	0x13	1	Update spdiff configuration changes

Configuration1: DAC: 256xFS and 24-bit , 44.1 khz file, VCR on spdiff

The following group of registers must be written to configure the spdif in non compressed mode (IEC958). The VCR pair of channel is then sent on the SPDIF.

Table 12: IEC958 Configuration for VCR on the SPDIF

Name	Address	value	
spdif_conf	0x60	0x21	DAC: 256xFS and 24-bit , 16-bit rounding
spdiff_cat	0x5F	0	Category code General
spdif_status	0x61	0x02	Non compressed mode, copy allowed, no pre-emphasis, SF=44.1 khz
spdif_cmd	0x5E	0x82	Trasnmit VCR channels on the SPDIF
play	0x13	1	Update spdiff configuration changes

Only the pairs L/R and VCR can be transmitted on the SPDIFF output.

6.2.2) SPDIF configuration in IEC61937 mode (compressed)

Configuration0: DAC: 256xFS and 24-bit ,
The following group of registers must be written to configure the spdif in non compressed mode (IEC958)

Table 13: IEC61937 Configuration

Name	Address	value	
spdif_conf	0x60	0x21	DAC: 256xFS and 24-bit ,autolatency
spdiff_cat	0x5F	0	Category code General
spdif_status	0x61	0x02	compressed mode, copy allowed,
spdif_cmd	0x5E	3	Bitstream is transmitted in IEC61937 standard
play	0x13	1	Update spdiff configuration changes

Note:
MPEG, Dolby Ac3 and DTS are the only format available for the spdif output when used to transmit compressed data.

6.3) VCR Output

The VCR output can be used to transmit a additional pair of channels in the following case:

- 8-channel stream (DVD-audio, DVD-video) reproduction
- VCR output for external recording

As noted in the datasheet, the only possibilty to have 3D sound on both the VCR output and the Left/Right channel is to configure the VCR to output a copy of the Left/Right channel (h/w copy) and to enable SRS post processing.
When 3D sounds is applied on the VCR output, the effect is disabled on the other 6 channels

One of the following configuration can be used

Table 14: VCR Available Output Configuration

Name	Address	value	
vcr_mix	0xAE 0x62 0x64	0x01 0x40 0x14	Enable SRS 3D sounds on the VCR. Select SRS post processing Select SRS 3D sound
vcr_mix	0xAE 0x62 0x64	0x01 0x80 0x14	Enable VMAX on the VCR. Select VMAX post processing Select VMAX stereo enhanced
vcr_mix	0xAE	0x02	Copy of Left/Right channels to VCR channels
vcr_mix	0xAE	0x08	Prologic Downmix on the VCR channels
vcr_mix	0xAE	0x10	2/0 Downmix on the VCR channels

The VCR channels can be redirectred to the SPDIF F output (cf paragraph 6.2)

7) Step6: How to select and configure a decoder

7.1) Algorithm selection

The STA310 is a 6+2-Channel multistandard audio decoder with the following available main algorithms:

- MPEG1 (stereo)
- Mp3 (stereo)
- MPEG2 (Multichannels 5.1)
- Dolby Digital (Multichannels 5.1)
- DTS (Multichannels 5.1)
- MLP (Stereo or Multichannels 5.1)
- LPCM video (8 channels) & PCM (2 channels)
- LPCM Audio (6 channels)
- CDDA (stereo)

In addition, 2 test signals generator are embedded:

- Beep Tone generator
- Pink Noise generator

Selection of the algorithm to decode is performed through a set of 2 registers (AUD_DECODESEL @0x4D) and (AUD_STREAM_SEL @0x4C).
The STA310 can auto-detect the type of stream transmitted only if the spdif input is used and only in compressed mode. In other words, the STA310 is able only to auto dectect the following streams: DTS, MPEG1-2 (Layer1 or layer2) and DOLBY AC-3. When a new stream type is detected on the spdif input the STA310 automatically performs a soft mute of the PCM output and reset the decoder.
The STA310 is not able to detect that PCM data are transmitted through the SPDIF (IEC958) and auto-detection is not possible with the I2s inputs.

The STA310 can only report the type of stream currently decoded when the spdif input in compressed mode is used. This information is provided in the register 0x43 and is the DTYPE field. An interrupt can be triggered every time a new frame header has been registred allowing you to read the DTYPE value and check the current decoded algorithm.

The table below lists the most commons algortyhms configurations with both Elementay Stream (ES) and IEC61937 stream type since they are supposed to be the most widely used:

Table 15: Common algorithms configurations for STA310

	I2S input (elementary stream)	Spdif input IEC61937/IEC958
Dolby digital (Ac3)	@0x4C = 3, @0x4D = 0	@0x4C = 5, @0x4D = 0
MPEG1	@0x4C = 3, @0x4D = 1	@0x4C = 5, @0x4D = 1
MPEG2	@0x4C = 3, @0x4D = 2	@0x4C = 5, @0x4D = 2

Table 15: Common algorithms configurations for STA310

	I2S input (elementary stream)	Spdif input IEC61937/IEC958
PCM	@0x4C = 3, @0x4D = 3	@0x4C = 3, @0x4D = 3
Pink Noise	@0x4C = 3, @0x4D = 4	
DTS	@0x4C = 3, @0x4D = 6	@0x4C = 3, @0x4D = 6
Beep Tone	@0x4C = 3, @0x4D = 7	
MLP	@0x4C = 3, @0x4D = 8	
MP3	@0x4C = 3, @0x4D = 9	

Note:
All the decoder configuration parameters can be changed “on the fly” while decoding.

7.2) Decoder Configuration

7.2.1) Dolby Digital Decoder Default Configuration

The following table is a example of a default configuration that can be applied when AC3 decoding is selected.

Table 16: Ac3 Default Register Configuration

Name	Address	value	
ac3_decode_lfe	0x68	1	Enable LFE channel by default
ac3_comp_mod	0x69	2	(1) Products with line level or for power amplified outputs
ac3_hdr	0x6A	0	(2) Full dynamic range for lound sounds
ac3_ldr	0x6B	0	Full dynamic range for quiet sounds
ac3_rpc	0x6C	0	Mute blocks when CRC error is detected
ac3_karamode	0x6D	0	(3)karaoke aware implementation
ac3_dualmode	0x6E	0	Output as stereo when 2/0 downmix is selected
ac3_downmix	0x6F	7	Default 3/2 (L, C, R, LS, RS) output channel configuration

(1) For deeper information rf (“Dolby Licensee Manual”)
(2) High level cut compression scaling is only allowed when not downmixing
(3) The Karaoke bitstreams are automatically detected. The STA310 choose then the appropriate karaoke reproduction mode depending on the number of vocal channels in the bitstream and the ouput channel configuration.

7.2.2) MPEG1 Decoder Default Configuration

MPEG-1 (ISO/IEC 11172-3) provides

Single-channel ('mono') and two-channel ('stereo' or 'dual mono') coding of digitized sound waves at 32, 44.1, and 48 kHz sampling rate. The predefined bitrates range from 32 to 448 kbit/s for Layer I, from 32 to 384 kbit/s for Layer II, and from 32 to 320 kbit/s for Layer III.

The following table is a example of a default configuration that can be used when MPEG1 decoding is applied

Table 17: MPEG1 Default Register Configuration

Name	Address	value	
mp_skip_lfe	0x68	0	Enable LFE channel by default
mp_prog_numberr	0x69	0	Select program 0, L0,R0 in front channels
mp_dual_mode	0x6E	0	Output as stereo when L0,R0 or Lt,Rt downmix is selected
mp_drc	0x6A	0	No dynamic range control, use full dynamic range
mp_crc_off	0x6C	1	Do not check CRC
mp_mc_off	0x6D	1	mandatory for MPEG1 bitstream
mp_downmix	0x6F	1	Default 2/0 (L, R) output channel configuration

7.2.3) MPEG2 Decoder Default Configuration

MPEG-2 BC (ISO/IEC 13818-3) provides

A backwards compatible (BC) multichannel extension to MPEG-1; up to 5 main channels plus a 'low frequent enhancement' (LFE) channel can be coded; the bitrate range is extended up to about 1 Mbit/s;

An extension of MPEG-1 towards lower sampling rates 16, 22.05, and 24 kHz for bitrates from 32 to 256 kbit/s (Layer I) and from 8 to 160 kbit/s (Layer II & Layer III).

The following table is a example of a default configuration that can be used when MPEG2 decoding is applied

Table 18: MPEG2 Default Register Configuration

Name	Address	value	
mp_skip_lfe	0x68	0	Enable LFE channel by default

Table 18: MPEG2 Default Register Configuration

Name	Address	value	
mp_prog_numberr	0x69	0	Select program 0, L0,R0 in front channels
mp_dual_mode	0x6E	0	Output as stereo when L0,R0 or Lt,Rt downmix is selected
mp_drc	0x6A	0	No dynamic range control, use full dynamic range
mp_crc_off	0x6C	1	Do not check CRC
mp_mc_off	0x6D	16	Allow decoding of Multichanel, and allow denormalisation
mp_downmix	0x6F	6	Default 3/2 (L, C, R, LS, RS) output channel configuration

7.2.4) PCM (16-bit Stereo) ES Default Decoder Configuration

The following table is a example of a default configuration that can be used when PCM (16-bit stereo) is applied either from the IEC958 input or the I2S input. As mentioned, the STA310 support only 16-bit stereo pcm samples in elementary stream mode.

Table 19: PCM (16-bit stereo) Default Register Configuration

Name	Address	value	
sfreq	0x05	1	(1)44.1khz sampling frequency
lpcmv_downmix	0x6F	0	No Downmix
lpcmv_ch_assign	0xA8	1	DVD Standart specification
lpcmv_multi_chs	0xA9	0	Stereo PCM

- Downmix gain coeffcients (@0x99-@0xA4) and downmix phase coefficients (@0x97-@0x98) will not be set since they are only necessary when the downmix register (“lpcmv_downmix”, @6F) is forced to 2/0.
- The downsampling feature provides by the register (“lpcmv_force_dws”, @0x70) wil not be set since this feature is only available in PES DVD video mode and not in ES mode
- This configuration must be used also when connecting the digital output of a CD player to the STA310.

(1) For other sampling frequency, please consult the datasheet page 43

7.2.5) Pink Noise Generator Default Decoder Configuration

The following table is a example of a default configuration that can be used when Pink Noise Generator is applied

Table 20: Pink Noise Default Register Configuration

Name	Address	value	
pn_downmix	0x6F	0x3F	Enable pink noise on all 6 channels
ocfg	0x66	0	Disable any bass management

- All channels must be set to 10db Attenuation. Check paragraph xxx to know how to apply volume control to the STA310

7.2.6) CDDA Default Decoder Configuration

This configuration will not be detail here since this is a particular mode that can be used only when the STA310 is used as compagnion chip for a STI55xx familly chip.

7.2.7) DTS Default Decoder Configuration

The following table is a example of a default configuration that can be used when DTSdecoding is applied

Table 21: DTS Default Register Configuration

Name	Address	value	
dts_lfe	0x68	1	Enable LFE channel by default
dts_16_14	0x68	0	(1)Default settings to 14-bit mode bitstream
dts_drc	0x6A	0	Disable dynamic range control, use full encoder dynamic
dts_dual	0x6B	0	Channel A is selected on the center channel for dual mode
dts_downmix	0x6F	7	Default 3/2 (L, C, R, LS, RS) output channel configuration

(1)Since a DTS bitstream for CD or LD is encoded with the linear PCM space, when this bitstream passes through the D/A stage without proper decoding , excessive white noise may be outputted from thhe analog stage. To help mitigate this problem, DTS bitstream for CD and LD only occupy the lower 14-bits of the PCM data word. This reduces the noise level by 12dB at the analog output stage.

7.2.8) PCM beep Tone Default Configuration

The following table is a example of a default configuration that can be used when the PCM beep generator is applied

Table 22: PCM Beep Tone Default Register Configuration

Name	Address	value	
pcm_btone	0x68	0	triangle at 24 khz
sfreq	0x05	0	48khz sampling frequency

Note:

- The beep tone is a triangular signal
- It is also available on the spdiff output.

The formula below give the relationship between the register value and the PCM beep tone frequency

$$Beeptonefrequency = \frac{sfreq}{2 \times (pcmbtone + 1)}$$

Figure 6:

7.2.9) MLP Default Decoder Configuration

The following table is a example of a default configuration that can be used when the MLP decoder is applied

Table 23: MLP Default Register Configuration

Name	Address	value	
mlp_crc	0x6C	0	Do not check the 4 CRCs
mlp_downmix	0x6F	6	Default 3/2 (L, C, R, LS, RS) output channel configuration
mlp_drc	0x6A	0	Disable Dynamic range control
mlp_force_dws	0x70	1	Automatic Downsampling from 192/176.4khz to 96/88.2 khz
mlp_fe	0x68	1	Decode LFE

- Automatic Downsampling can be removed if the external DAC is able to handle 192 khz sampling frequency
-

7.2.10) MP3 Default Decoder Configuration

There is no specific configuration for the MP3 decoder.

8) Step7: Post Processing configuration

8.1) Post Processing Selection

The following table lists the different available configurations: .

Table 24: Post Processing configurations

Post Processing Configurations	Address	value	Channels Configuration
Prologic decoder	0x62	0x01	(2 to 5)
Circle Surround	0xB1	0x01	(2 to 5)
SRS	0x62	0x40	(5 to 2) or (2 to 2)
VMAX	0x62	0x80	(5 to 2) or (2 to 2)
Prologic + SRS	0x62	0x41	(2 to 5) ->(5 to 2)
Prologic + VMAX	0x62	0x81	(2 to 5) ->(5 to 2)
Circle Surround + SRS	0x62 0xB1	0x01 0x01	(2 to 5) ->(5 to 2)
Circle Surround + VMAX	0x62 0xB1	0x80 0x01	(2 to 5) ->(5 to 2)
Prologic + Karaoke			
Prologic + Channel Delay			
channel delay + SRS			
channel delay + VMAX			

Note:
All the post processing configuration parameters can be changed “on the fly” while decoding.

8.2) Post Processing configuration

8.2.1) Prologic Decoder Configuration

The Dolby licensee Manual requires that a device including dolby Pro Logic decoding include the following features:

- For all 2-channel sources Pro logic decoding is OFF

- For all 2-channel sources Pro logic decoding is ON

The STA310 includes a extra feature allowing automatic activation of the Pro Logic based on the “Dolby surround mode parameter” set in the bitstream.

Table 25: Pro Logic Default Register Configuration

Name	Address	value	
pl_ab	0x64	3	Enable auto-balance function and wide surround mode
pl_downmix	0x65	0x0F	Default 3/2 (L, C, R, LS, RS, LFE) output channel confg

8.2.2) Circle Surround Decoder Configuration

Two registers are used to configure the “Circle surround” algorithm:

Table 26: Circle Surround Register Configuration

Name	Address	Description
circle_mode	0xB2	Configure circle surround mode
circle_delay	0xB3	(1)Specifies delay used for frequency in surround channels

(1) The delay can be set in range from 0 seconds to 1440/Fs second (where ‘FS’ is the sampling frequency in Hz):

$$Delay = 6 \times \frac{(delay)}{(FS)}$$

Figure 7:

:
The following table describes the bit field of the "cicle_mode" register. Each bit can be individually set or reset.

Table 27: Circle Surround mode register ronfiguration

Bit	value	
0	0 1	Music mode Cinema mode

Table 27: Circle Surround mode register ronfiguration

Bit	value	
1	0 1	Center mode Phantom mode
2	0 1	No filtering on center channel High pass filter is applied on center channel
3	0 1	Normal mode Wide mode
4	0 1	4.2.4 mode 5.2.5 mode

8.2.3) SRS Decoder Configuration

The SRS configuration table depends on the number of input channels.:

- When a 5.1 input stream is applied to the SRS decoder, “truesurround” algorithm must be applied with some variation when only 1 surround channel is present.
- When a 2-channel input stream is applied, “srs3d” algorithm is applied.
- “Srs3d” can nevertheless be aplied to a 5.1 channel input stream if a downmix (L0,R0) is applied before being used.

One of the following configuration can be used:

Table 28: SRS Default Register Configuration

Name	Address	value	
srs_mode_ctl	0x64	0x04	Truesurround (5 to 2)
srs_mode_ctl	0x64	0x14	SRS 3D (2 to 2)
srs_mode_ctl	0x64	0x40	SRS prologic with Mono Surround on Ls (4 to 2)
srs_mode_ctl	0x64	0x80	SRS prologic with Mono Surround on Rs (4 to 2)

8.2.4) VMAX decoder configuration

The VMAX configuration table depends on the number of input channels.:

- When a 5.1 input stream is applied to the VMAX decoder, “virtual theather“ algorithm must be applied with some variation when only 1 surround channel is present.

- When a 2-channel input stream is applied, “stereo enhanced” algorithm is applied.
- “stereo enhanced” can nevertheless be aplied to a 5.1 channel input stream if a downmix (L0,R0) is applied before being used

One of the following configuration can be used

Table 29: VMAX Default Register Configuration

Name	Address	value	
vmax_mode	0x64	0x04	virtual theater(5 to 2)
vmax_mode	0x64	0x08	VMAX prologic with Mono Surround on Ls (4 to 2)
vmax_mode	0x64	0x14	VMAX stereo enhanced (2 to 2)

8.2.5) Bass Management

Bass mangement was not mentioned in the table 14 since it can be applied after any kind of post processing or decoders.The register ocfg (@0x66) allows configuration of the bass management according to the **dolby digital recommendations**.

8.2.6) Volume Control

The STA310 can apply volume control after any post-processing or decoders. It is the last post-processing of the chain. The resolution of the attenuation is 0.5 dB.
To write the volume, the following procedure must be followed:

- Check that the DSP is ready to accept new volume settings (CHAN_IDX = 4)
- Write the attenuation of the pair of channel.
- Select the pair of channel

Configuration example1: 10 dB attenuation to all channels:

The function below is used to check the state of the decoder

```
char AUDRV_IsRunning(void)
{
    char   value;

    value = read_host_reg(AUDRV_REG_RUN) & MASK_BIT_0;
```

```
     return ( value);
}
```

The function below is used to apply volume to the decoder

```
void AUDRV_SetChannelVolumeConf()
{
.....U8 value;

...../*  Write the volume for left and right channels */

.....write_host_reg (0x4E,20);
.....write_host_reg (0x63,20);
.....write_host_reg (0x67,0);

...../* if chip is in idle mode => the volume update must be forced by access to MUTE or PLAY register */
.....if ( !AUDRV_IsRunning ())
.....{
    . ....value = read_host_reg (0x14);
..... ....write_host_reg (0x14, value);
.....}

...../* wait until the DSP is ready to accept new volume */
.....while ( read_host_reg(0x67) != 4 ) ;

...../* Write the volume for center and subwoofer channels */
.....write_host_reg (0x4E, 20);
.....write_host_reg (0x63, 20);
.....write_host_reg (0x67, 1);

...../* if chip is in idle mode => the volume update must be forced by access to MUTE or PLAY register */
.....if ( !AUDRV_IsRunning ())
.....{
..... ....value = read_host_reg (0x14);
..... ....write_host_reg (0x14, value);
.....}

...../* wait until the DSP is ready to accept new volume */
.....while ( read_host_reg(0x67) != 4 ) ;

     /* write the volume for left surround and right surround channels */
     write_host_reg (0x4E, 20);
     write_host_reg (0x63, 20);
     write_host_reg (0x67, 2);

     /* if chip is in idle mode => the volume update must be forced by access to MUTE or PLAY register */
     if ( !AUDRV_IsRunning ())
     {
         value = read_host_reg (0x14);
         write_host_reg (0x14, value);
     }

     /* wait until the DSP is ready to accept new volume */
     while ( read_host_reg(0x67) != 4 ) ;
}
```

9) Step8: How to start playing music

Once the initialisation has been perform the STA310 can be started:
// Start the PCM block
==> write_host_reg(19, 1)
// Unmute the decoder
==> write_host_reg(20, 0)
// Initialisation is done, start playing music
==> write_host_reg(114, 1)

10) Hardware Interface

10.1) The Parallel Interface Protocol

The STA310 can be controlled by a host device (micro-controller, host bus, ...) using an I2C interface or a general purpose host parallel interface. When the pin SELCI2C is low level, the chip is controlled through the host parallel interface. The pin MAINI2CADR must then be set to high level

The address bus HADR[7:0] is used to select one of the 256 register locations. Some registers are Read/ Write, and some are write or read only. The signal “HRWB” defines whether the register access is a read or a write (high for read, low for write).
A cycle is defined by the assertion of signal “DCSB”. In response to this signal the signal “WAITB” is always asserted. The address, for a read or write, must be setup before the “DCSB” line is activated. If a read cycle is requested the data lines “HDATA” will be driven by the STA310. For a write cycle the STA310 will latch the data placed on the data lines HDATA[] on the rising edge “of DCSB”.

10.1.1) Write Access waveforms in normal mode :

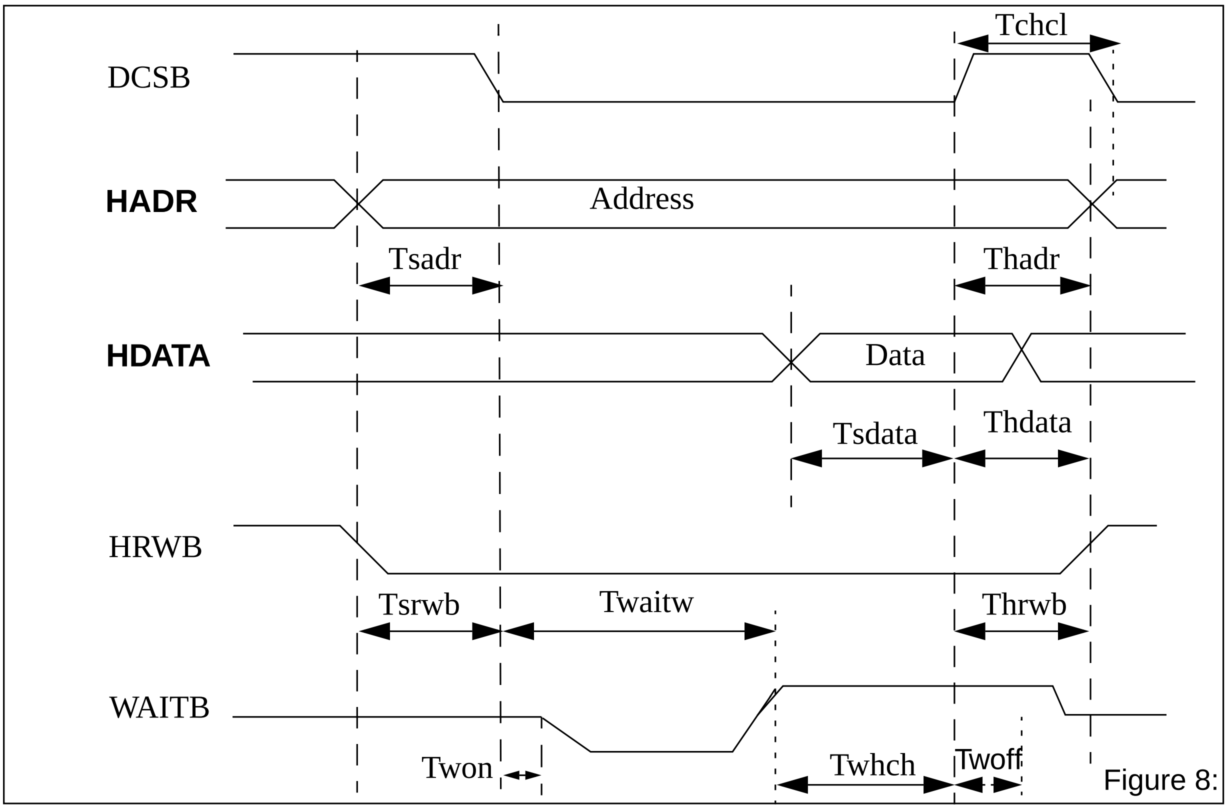

DCSB
HADR
HDATA
HRWB
WAITB
Tchcl
Address
Tsadr
Thadr
Data
Tsdata
Thdata
Tsrwb
Twaitw
Thrwb
Twon
Twhch
Twoff

Figure 8:

10.1.2) Read Access waveforms in Normal mode :

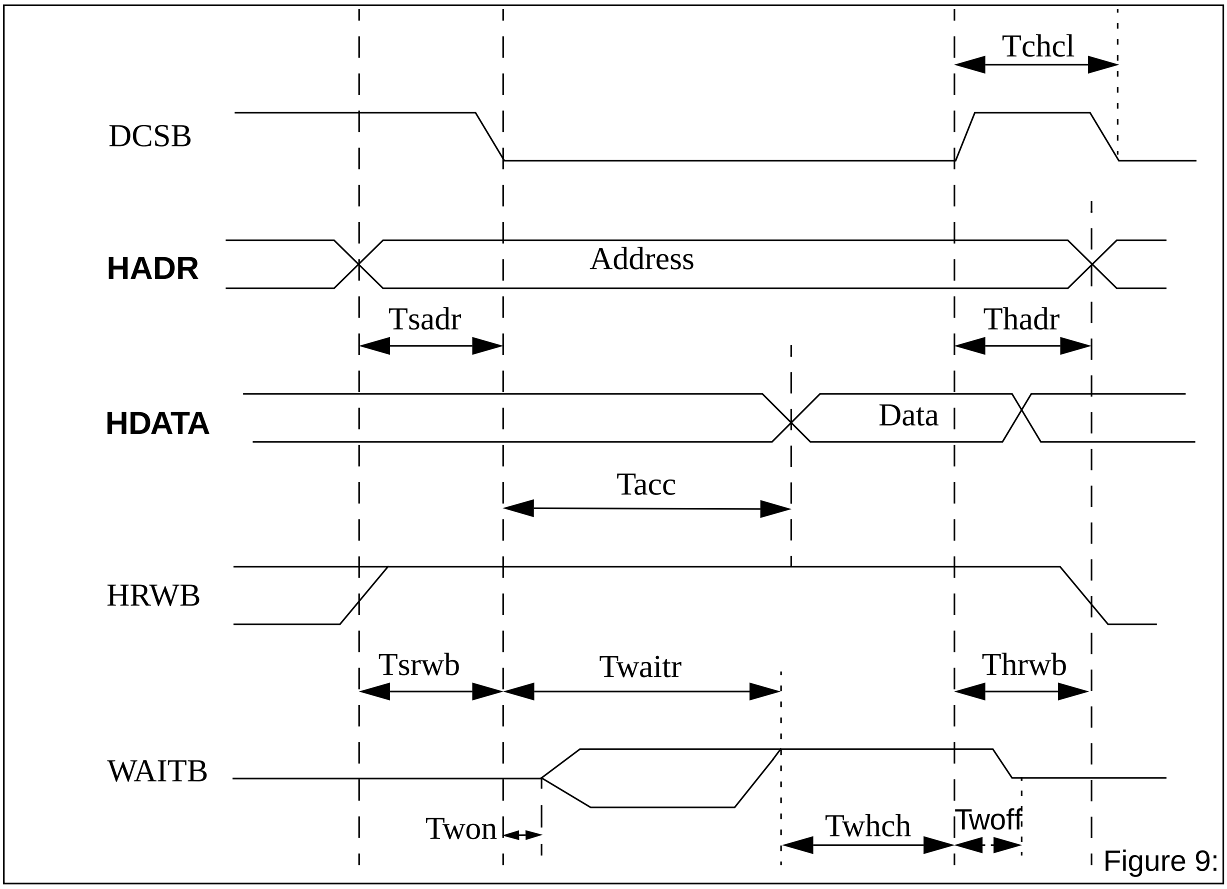


Figure 9:

10.1.3) Timings of this interface

Tsys is the period of the system clock of the STA310.

Table 30: Host interface timings

Timing Name	Min	Max	Description
Tsadr	5 ns		Hadr to dcsb set-up time
Tsdata	5 ns		Hdata to dcsb set-up time
Tsrwb	5 ns		Hrwb to dcsb set-up time
Thadr	0 ns		Hadr to dcsb hold time
Thdata	0 ns		Hdata to dcsb hold time

Table 30: Host interface timings

Timing Name	Min	Max	Description
Thrwb	0 ns		Hrwb to dcsb hold time
Twaitw	0.5 Tsys+10	1.5 Tsys+10	Maximum wait time when writing
Twaitr	1.5 Tsys+10	2.5 Tsys+10	Maximum wait time when reading
Tchcl	2 Tsys + 10		dcsb high to dcsb low
Twoff		5 ns	Dcsb high to waitb off
Twon		5 ns	Dcsb low to waitb on
Twhch	0 ns		Waitb high to Dcsb high
Tacc	Tsys+10	2 Tsys+10	Dcsb low to Hdata ready

10.2) I2C Slave format protocol:

When the pin SELCI2C is high, the chip is controlled through the I^{2}C interface. The I^{2}C unit works at up to 400khz in slave mode with 7-bit addressing.

- The pin MAINI2CADR selects the device address. When MAINI2CADR is high the slave address is 0x5C, when low the device address is equal to the value on the address bus (A0..A6).
- The pin SDAI2C is the serial data line.
- The pin SCLKI2C is the serial clock.

10.2.4) The protocol:

Data changes on the SDA line must only occur when SCLKI2C clock is low excepts for START and STOP conditions. In that case, the transition is done when the clock is High.
A START condition is identified by a High to Low transition of the SDA line while the clock signal is High. A START condition must precede any command for a data transfer.
A STOP condition is identified by a Low to High transition of the SDA line while the clock signal SCLKI2C is High. A STOP condition terminates the communications between the STA310 and the master of the I2C bus.
An Acknowledge bit is used to indicate a successful data transfer. The bus transmitter, either the master or the slave, releases the SDA line after sending 8 bits of data. During the 9th clock pulse, the receiver pulls the SDA line to Low to acknowledge the reception of 8 bits of data.
During the data transfer, the I2C slave interface of the STA310 samples the SDA line on the rising edge of the SCLKI2C clock. The SDA signal has to be stable during the rising edge of the clock and the data can change only when the SCLKI2C clock line is low.

10.2.5) Addressing of the STA310

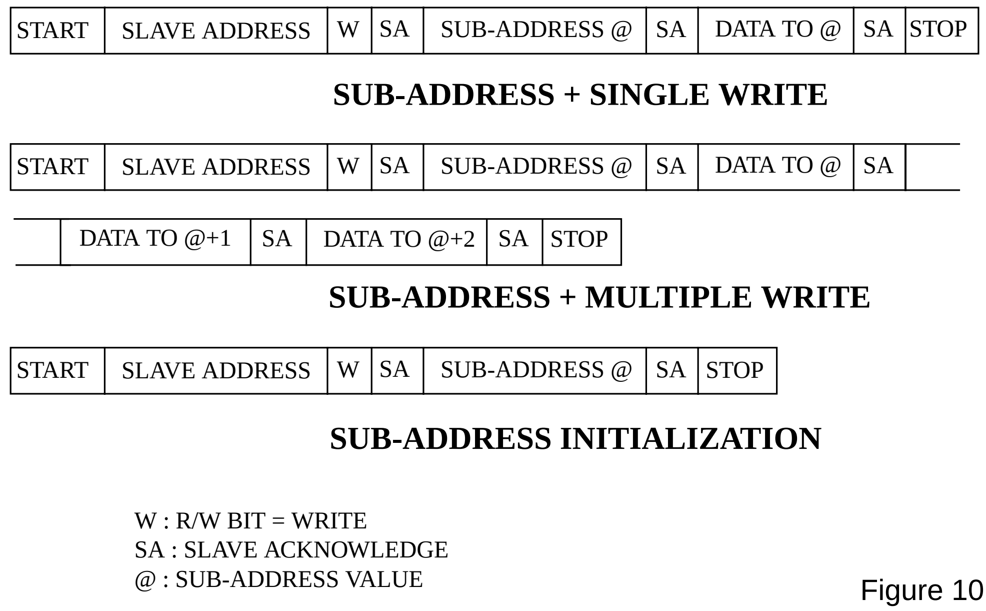
To start the communications between the master and the STA310, the master must initiate the transfer with a START condition. Then, the master has to send on the SDA line 8 bits (MSB first) corresponding to the device I2C address (7 bits) and the mode bit RW (Read or Write).
The 7 most significant bits are the address of the device.The 8th bit (LSB) selects a read (bit set to 1) or write (bit set to 0) operation. After a START condition, the STA310 I2C slave interface identifies on the I2C bus the device address and, if the address matches, acknowledges this match on the SDA line during the 9th bit time frame. The byte following the device identification byte is the address of the Host register to be accessed.

10.2.6) Sub-address initialization

This mode is used for the initialization of the Host address register (sub-address value). The Host address register is the register that points the data register to be accessed (read or write).

10.2.7) “Sub-address + single write” & “Sub-address + multiple write”

The second mode, the multiple write, exploites the autoincrementation of the sub-address pointer to avoid to initialize, for sequential accesses of the Host registers, the sub-address at each write operation. The length of a multiple write is limited to the size of the Host register area (256 locations).



START	SLAVE ADDRESS	W	SA	SUB-ADDRESS @	SA	DATA TO @	SA	STOP

SUB-ADDRESS + SINGLE WRITE

START	SLAVE ADDRESS	W	SA	SUB-ADDRESS @	SA	DATA TO @	SA	

	DATA TO @+1	SA	DATA TO @+2	SA	STOP

SUB-ADDRESS + MULTIPLE WRITE

START	SLAVE ADDRESS	W	SA	SUB-ADDRESS @	SA	STOP

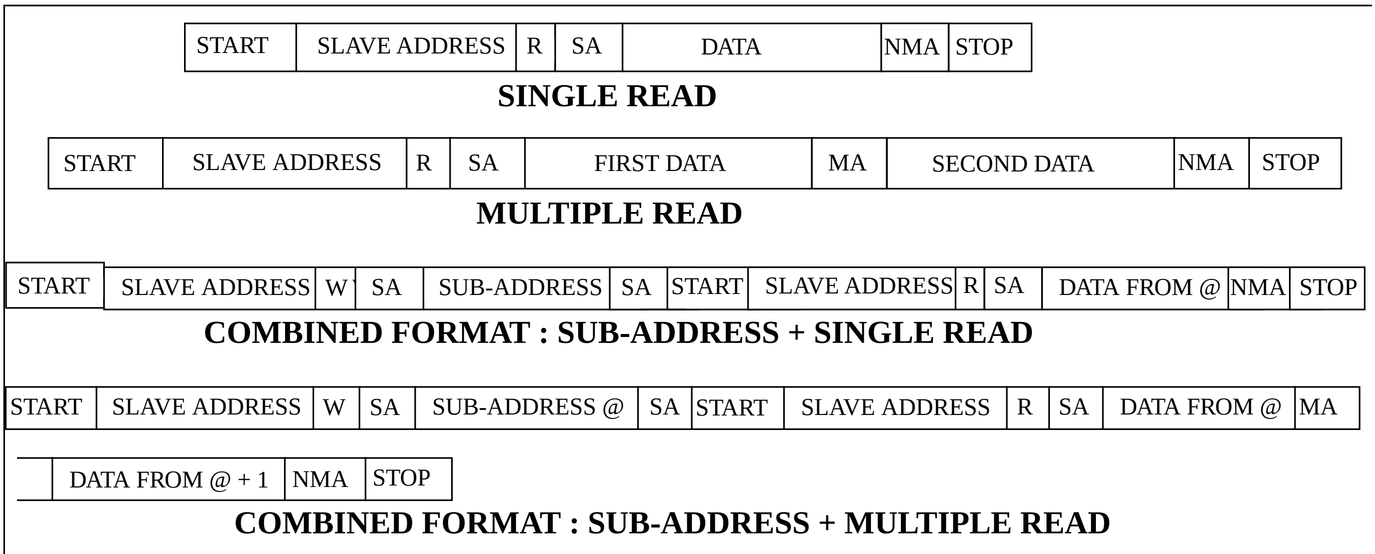
SUB-ADDRESS INITIALIZATION

W : R/W BIT = WRITE
SA : SLAVE ACKNOWLEDGE
@ : SUB-ADDRESS VALUE

Figure 10:

10.2.8) “Single read” & “multiple read” & “sub-address + single read” & “sub-address + multiple read”

The single read operations are performed from the current sub-address value. The sub-address value can be initialized using the “sub_address initialization” sequence presented in the previous transfer chart : see the “combined format - sub-address + single read” diagram in the following chart.
The multiple read operations are performed from the current sub-address value and the sub-address register is automatically incremented at each access. The sub-address value can be initialized using the “sub_address initialization” sequence presented in the previous transfer chart : see the “combined format - sub-address + multiple read” diagram in the following chart. The length of a multiple read is limited to the size of the Host register area (256 locations). A multiple read is terminated by a Non Master Acknowledge followed by a STOP condition.



START	SLAVE ADDRESS	R	SA	DATA	NMA	STOP

SINGLE READ

START	SLAVE ADDRESS	R	SA	FIRST DATA	MA	SECOND DATA	NMA	STOP

MULTIPLE READ

START	SLAVE ADDRESS	W	SA	SUB-ADDRESS	SA	START	SLAVE ADDRESS	R	SA	DATA FROM @	NMA	STOP

COMBINED FORMAT : SUB-ADDRESS + SINGLE READ

START	SLAVE ADDRESS	W	SA	SUB-ADDRESS @	SA	START	SLAVE ADDRESS	R	SA	DATA FROM @	MA

	DATA FROM @ + 1	NMA	STOP

COMBINED FORMAT : SUB-ADDRESS + MULTIPLE READ

R : R/W BIT = READ
SA : SLAVE ACKNOWLEDGE
MA : MASTER ACKNOWLEDGE
NMA : NO MASTER ACKNOWLEDGE (LAST DATA)
@ : SUB-ADDRESS VALUE

Figure 11:

10.3) The Main I2S input

The Main I2S input is compatible with both I2S and Sony protocol. The interface is configured using the Host registers described in the Host register section of this document.
When the serial mode is selected the STA310 uses a four-signal data interface that provides an input data line SIN, an input clock DSTRB, a word clock input LRCLKIN and a handshake output signal REQ.

10.3.9) Modes without LRCLKIN

In this mode the signal LRCLKIN is not used by the STA310. The input data SIN is sampled on the rising edge of DSTRB. When the STA310 input buffer (i.e. input FIFO) is almost full the REQ signal is asserted. The polarity of REQ signal is programmable through the register sinsetup (@ 0x0C). The data must be sent most significant bits first. In this mode the register cansetup (@0x0D) is not taaken into account. This mode is mainly use to transfer compressed data.

.

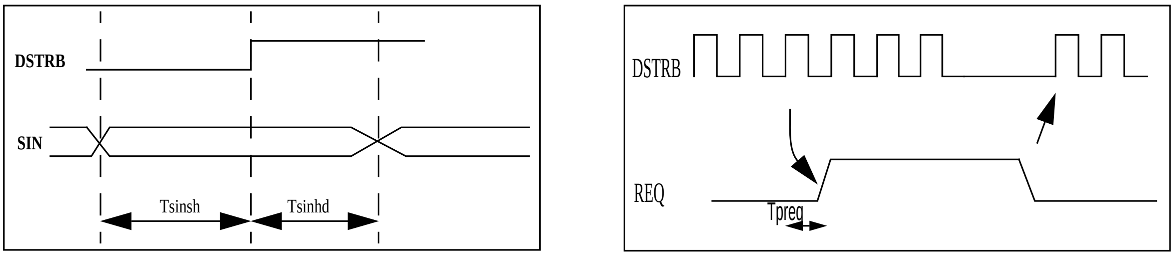


Figure 12:

When the STA310 cannot accept further data the REQ is deasserted, the DSTRB clock must be stopped as soon as possible to avoid data loss. After the REQ is deasserted, the STA310 is still able to accept data for a limited number of clock cycles.

Table 31: Main I2s Timing for Modes without LRCLKIN(*)

Symbol	Parameter	Min	Max	Unit
FdstrbSerial	DSTRB Max frequency in serial mode		System Clock	Mhz
T dstrblow	DSTRB low pulse	5		ns
Tdstrbhigh	DSTRB high pulse	5		ns
Tsinsh	SIN setup time to DSTRB rising edge	5		ns

Figure 11:

Table 31: Main I2s Timing for Modes without LRCLKIN(*)

Symbol	Parameter	Min	Max	Unit
Tsinhd	SIN Hold time to DSTRB rising edge	5		ns
Tp req	Propagation time from DSTRB rising to REQ inactive	4 * sys_clk period	6 * sys_clk period	ns

(*) Preliminary values

10.3.10) “Lrclkin” modes

These modes are used mainly for non compressed data (but they can be used also for compressed data). The LRCLKIN signal is used to make the distinction between the left and right channels. In these modes any edge of the LRCLKIN signal indicates a word boundary. The data transfer between the input interface and the fifo are done on a byte basis. After the edge (rising or falling) of the LRCLKIN, a new byte is transferred to the first stage of the STA310 input interface every 8 $\overline{\text{DSTRB}}$ clock cycles. If the number of time slots is not a multiple of 8, the remaining data are lost. The polarity of LRCLKIN and $\overline{\text{DSTRB}}$ are programmable. The LRCLKIN can be delayed by one time slot, in order to support PCM delayed mode.
See the host register chapter for further details.

Example 1 :
Only the first byte is transfered to the MMDSP+ Openplatform because the number of time slot is 12 (8+4). SIN and LRCLKIN are sampled on the falling edge of DSTRB. In this case sinsetup = 3 and cansetup = LeftFirstChannel + FallingStrobe + AllSlot = 2+4+8 =14.

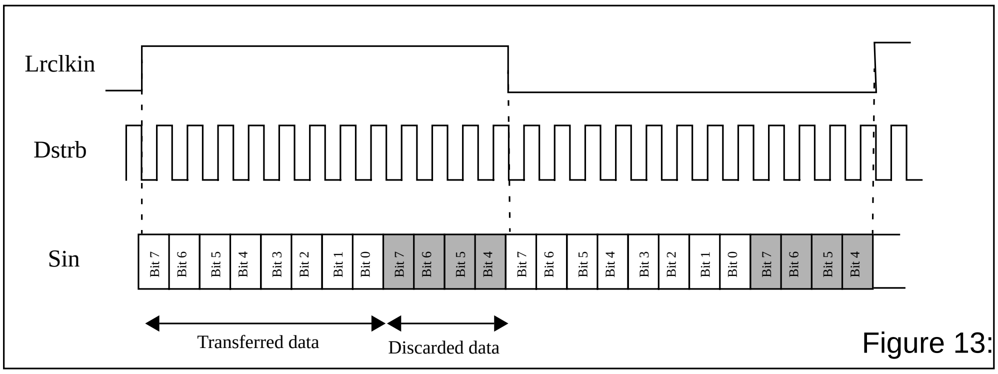


Figure 13:

Example 2 :
Only the first 2 bytes are transfered to the STA310 because the number of slot is 20 (16+4). SIN and LRCLKIN are sampled on the falling edge of $\overline{\text{DSTRB}}$. The data are in delayed mode.
The register configuration is sinsetup = 3 and cansetup = DelayMode + RigthFirstChannel + FallingStrobe + AllSlot = 1+ 2 + 4+ 8 =15.

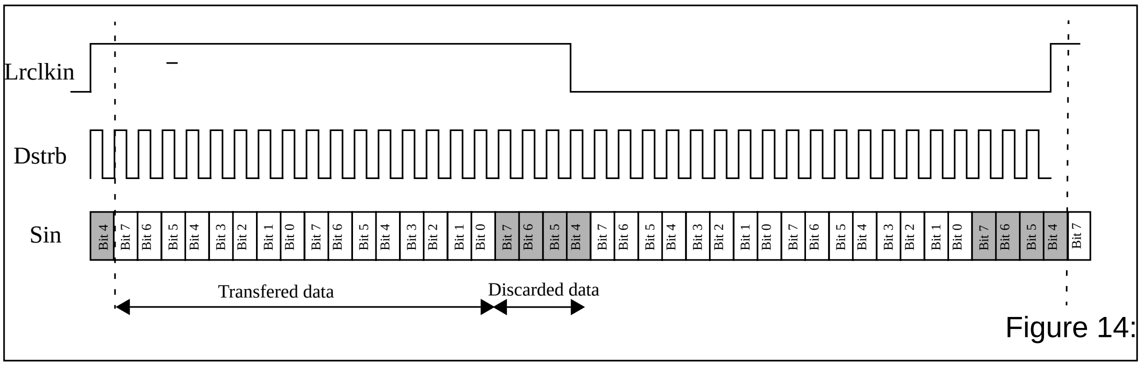


Figure 14:

Example 3:
This mode is a specific mode where only the first 16 data bits are transfered. The remaining bits are discarded.The register configuration is sinsetup = 3 and cansetup = Delay_Mode + FallingStrobe = 1 + 4 = 5

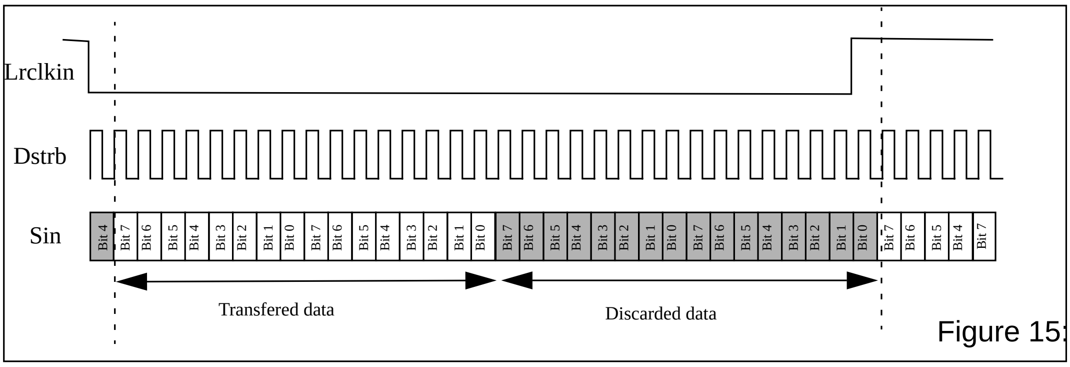


Figure 15:

10.3.11) Data Parallel interface

In this mode the data must be presented on the 8 bit parallel bus HDATA[]. On the rising edge of the $\overline{\text{DSTRB}}$ clock, the data byte is sampled by the STA310. The $\overline{\text{REQ}}$ pin is used to signal when the input fifo is full. When $\overline{\text{REQ}}$ is deasserted the transfer must be stopped to avoid to loose data. The HDATA[] bus is shared between the HOST Parallel interface and the Data Parallel interface, to avoid conflict the $\overline{\text{DSTRB}}$ signal and the DCSB signal must respect certain timing constraints. The register configuration is sinsetup = 0 and cansetup = dont care.

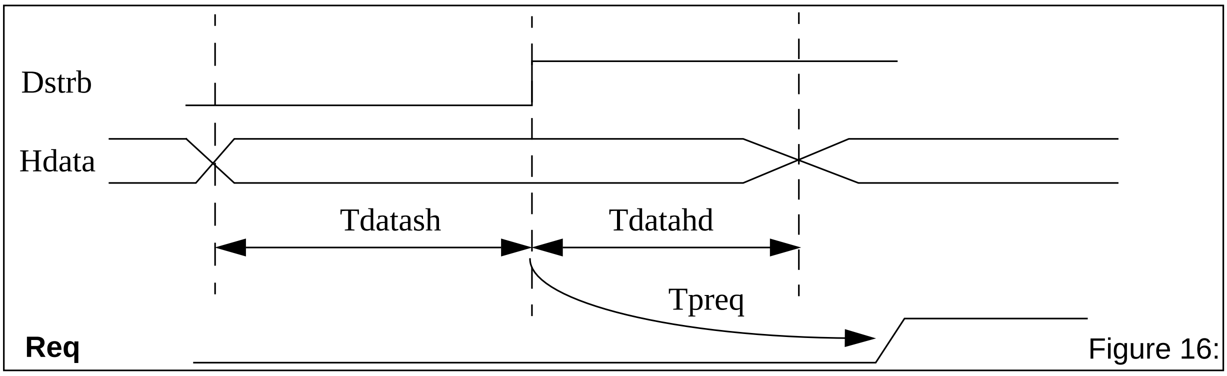


Figure 16:

Table 32: Timing for data parallel interface(*)

Symbol	Parameter	Min	Max	Unit
Fdstrbpara	DSTRB Max frequency in parallel mode		sys_clk frequency/8	Mhz
T dstrblow	DSTRB low pulse	5		ns
Tdstrbhigh	DSTRB high pulse	5		ns
Thdatash	HDATA setup time to DSTRB rising edge	5		ns
Thdatahd	HDATA Hold time to DSTRB rising edge	5		ns
Tp req	Propagation time from DSTRB rising to REQ inactive	4 * sys_clk period	6 * sys_clk period	ns

(*) Preliminary values

10.3.12) The input FIFO

The input FIFO is a 2048 bit FIFO.

The $\overline{\text{REQ}}$ pin of the STA310 is connected to the FIFO ‘almost full’ level indicator. The level of the indicator is set to 2023 bits. It means that if the level of data in the FIFO is higher or egal to 2024 bits, the $\overline{\text{REQ}}$ pin is active otherwise the REQ pin is inactive.
The $\overline{\text{REQ}}$ pin polarity is programmable (i.e. active High or Low) by the Host register sinsetup[2] (@ = 0x0C).
The maximum bit rate at the input of the FIFO (pin SIN and DSTRB of the STA310) is the maximum value between 50 Mbit/s and the sys_clk frequency of the STA310.

The input FIFO can also be filled byte per byte by writting into the Host register datain (@ = 0x0E) using the I2C slave interface or the Host parallel interface. In that case the maximum data rate (in Byte per second) is 1/8 times the sys_clk frequency of the STA310.

The maximum number of data that can be transmitted after $\overline{\text{REQ}}$ has become invalid is:

$$Nbits = \left(23 - \frac{6 \times Fdstrb}{Fdstrbmax}\right)$$

Figure 17:

where
Fdstrb: $\overline{\text{DTSRB}}$ frequency
Fdstrbmax: Max $\overline{\text{DSTRB}}$ frequency = System Clock

10.4) The Second I2s input