ANX9021 HDMI Receiver Firmware Guide

ANX9021 HDMI Receiver Firmware Guide

Basic System Connections

This document describes the application of the ANX9021 in a digital display system, such as a digital television. Figure 1 illustrates a representative system configuration in which the ANX9021 is connected to a host microcontroller via its I²C interface, interrupt output (INTR) and rest input (RESETN).

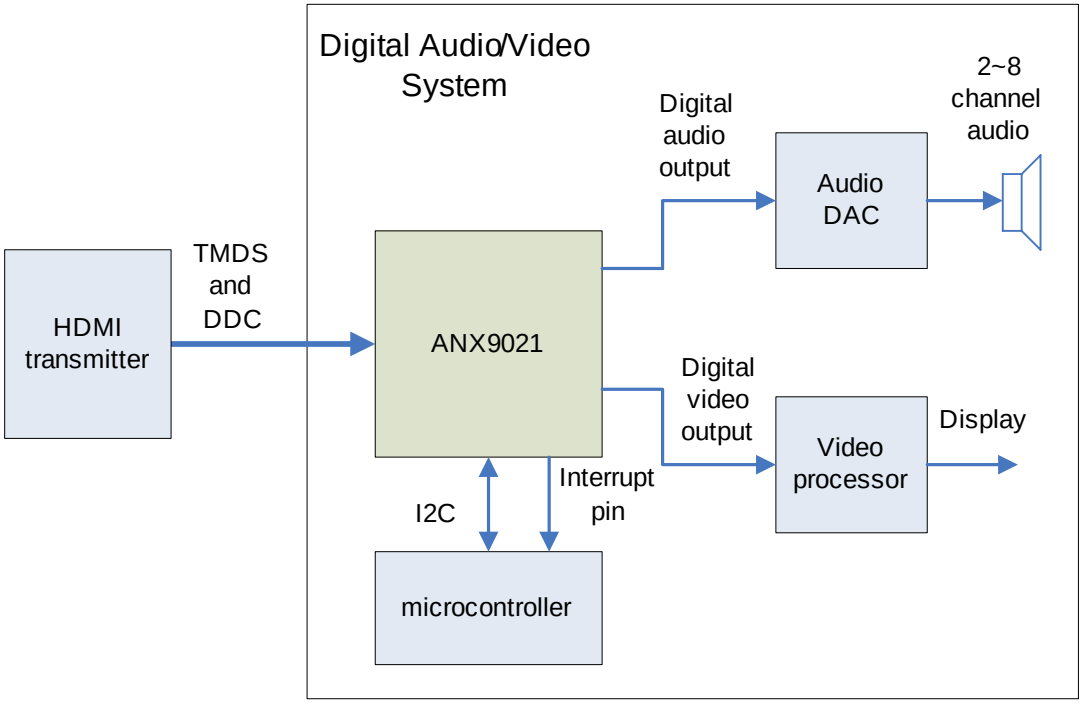

Figure 1 Connections to the HDMI receiver is a digital multimedia system

Initialization and Reset

System Initialization

Initialization and configuration of the ANX9021 must follow the initialization of the display system firmware. The recommended structure of the system firmware as related to the HDMI interface is covered in a later section of this document.

Hardware Reset

The RESETN input to the ANX9021 is commonly connected to a general purpose input/output (GPIO) pin of the host microcontroller. Firmware is responsible for generating the hardware reset signal of the ANX9021 by puling down RESETN for a minimum of 100 ms. The logic state of DEVAD_CLK48B during the rising edge of RESETN also determines

the I²C base address to which the ANX9021 will respond. For simplicity the remainder of this document assumes the base addresses 0x60 and 0x68 are chosen.

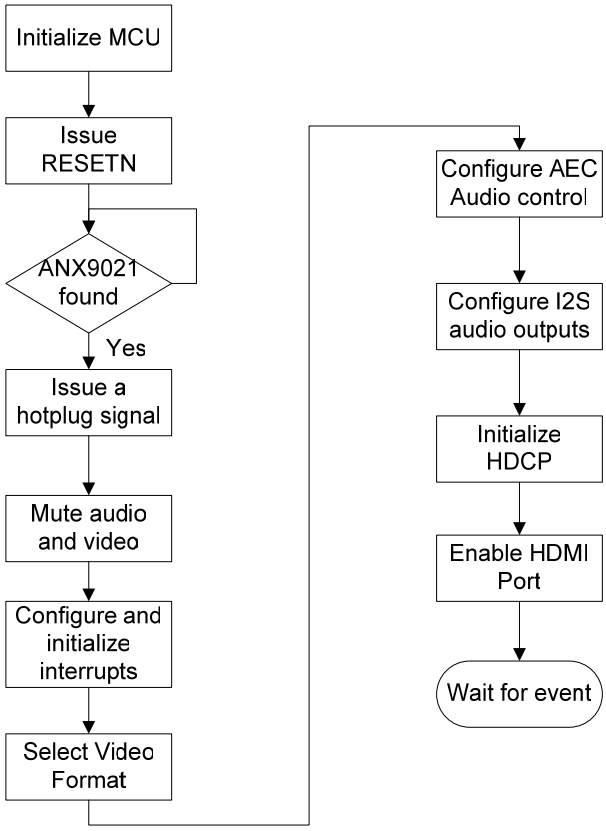

Figure 2 Initialization process for the ANX9021

Locating the ANX9021

Figure 2 summarizes the initialization process. After asserting RESETN, firmware should assure it has established communications with the ANX9021 by confirming that it can read its vendor and device ID registers (see Figure 3).

```
c1 = 0;
while (1)
{
       HDMIRX_reset_pin = 0;
       delay_ms(10);
       HDMIRX_reset_pin = 1;
       delay_ms_(10);
       c = i2c_read_P0_reg(0x02, &c1);
       if ((c == 0) && (c1 == 0x21)) {
              c = i2c_read_P0_reg(0x03, &c1);
              if (c == 0) && (c1 == 0x90))
                     break;
       }
}
```

Figure 3 Sample code confirming communications with the ANX9021

Initialize the ANX9021

After power-on reset, most of the ANX9021 device functions are powered down except for pixel clock detection (CKDT), and HDMI port termination. Firmware needs to configure the other registers appropriately for the system design. Table 1 lists the ANX9021 register functions involved in initialization.

Register: Offset	Register Name	Initialized Vale	Purpose
0x68: 0x37	HDMI Mute	0x03	Mute audio and video until decoded HDMI data stream is stable to avoid output noises.
0x68: 0xB3	Chip Control	0xE5	Use digital method to generate CKDT (HDMI link clock detection).
0x60: 0x79	Interrupt Control	0x02, System dependent	Initialized value is system dependent. For the Evaluation Board, interrupt signal is set to push-pull output and high polarity.
0x60: 0x75	INTR Mask 1	0xFC	Disable HCDP interrupts to conform to multimedia sources that do not use HDCP.
0x60: 0x76	INTR Mask 2	0xBF	Enable video detection related interrupts.
0x60: 0x77	INTR Mask 3	0x3F	Enable AVI, SPD, audio and MPEG packet interrupts
0x60: 0x78	INTR Mask 4	0xEF	Enable HDCP failure and audio error interrupts
0x60: 0x7D	INTR Mask 5	0xEF	Enable format change related interrupts
0x60: 0x7E	INTR Mask 6	0xFF	Enable new ACP and cable unplug interrupts
0x68: 0x16	ACR Control 3	0x07	Change CTS change threshold to 0x07.
0x60: 0x5F	Auto Video	0x00, System	Set the automatic video output format based on the requirement of the system. The Eval

	Format	dependent	Board uses 0x00 (component video).
0x68: 0x27	I2S Control 2	0xF9, System dependent	Configure I²S outputs. The Eval Board supports up to 8 channels and uses PCM data only.
0x68; 0x29	Audio Control	0x05, System dependent	Enables I²S and SPDIF outputs.
0x60: 0x05	Software Reset	0x09 then 0x80	By writing 0x09 followed by 0x80 to the software rest register, the firmware performs a soft reset, HDCP reset, and enables automatic HDCP reset.
0x60: 0xB5	AEC Control	0x05	Enable auto audio and video control
0x60: 0x09	Port Select	0x11	Enables HDMI Port 0 as the default input

Table 1 Register settings for ANX9021 initialization

Firmware Overview

After the ANX9021 has been properly initialized and when an HDMI source plugs into the active input port, the receiver will detect the TMDS clock over the HDMI cable. A clock detect interrupt will occur, software acknowledges the interrupt, powers up all receiver functions and come to normal operation mode.

In normal operation mode, all interrupt events are available. If the HDMI transmitter sends a valid video stream, the SYNC signal (HSYNC and VSYNC) will be valid, thus allowing the firmware to measure the video format and timing through the corresponding registers. If the video stream is stable and the format is supported, firmware may enable the digital video outputs.

Similarly firmware should wait until wait the audio stream is stable to enable the I²S and/or SPDIF interfaces. Figure 4 shows the high level state transitions of the firmware.

Monitor CKDT is a powered down state. When the HDMI receiver was first initialized, or when the HDMI cable is unplugged, firmware powers down most of the chip functions and checks for TMDS clock detection.

Wait SCDT follows TMDS clock detection. Firmware is waiting for the TMDS link to become stably established by waiting for SCDT to become active.

Wait Video follows sync detection. Firmware is waiting for a stable video stream and confirming that it is in a format that is supported by the system before enabling video output.

Wait Audio is performed after firmware has established that video stream is stable and the input includes audio data stream. Firmware is waiting for a stable and supported audio stream prior to enabling audio output.

Playback is the normal audio and video output state.

Format Error is the state in which firmware has concluded that the system does not support the incoming format and waits for a new video stream.

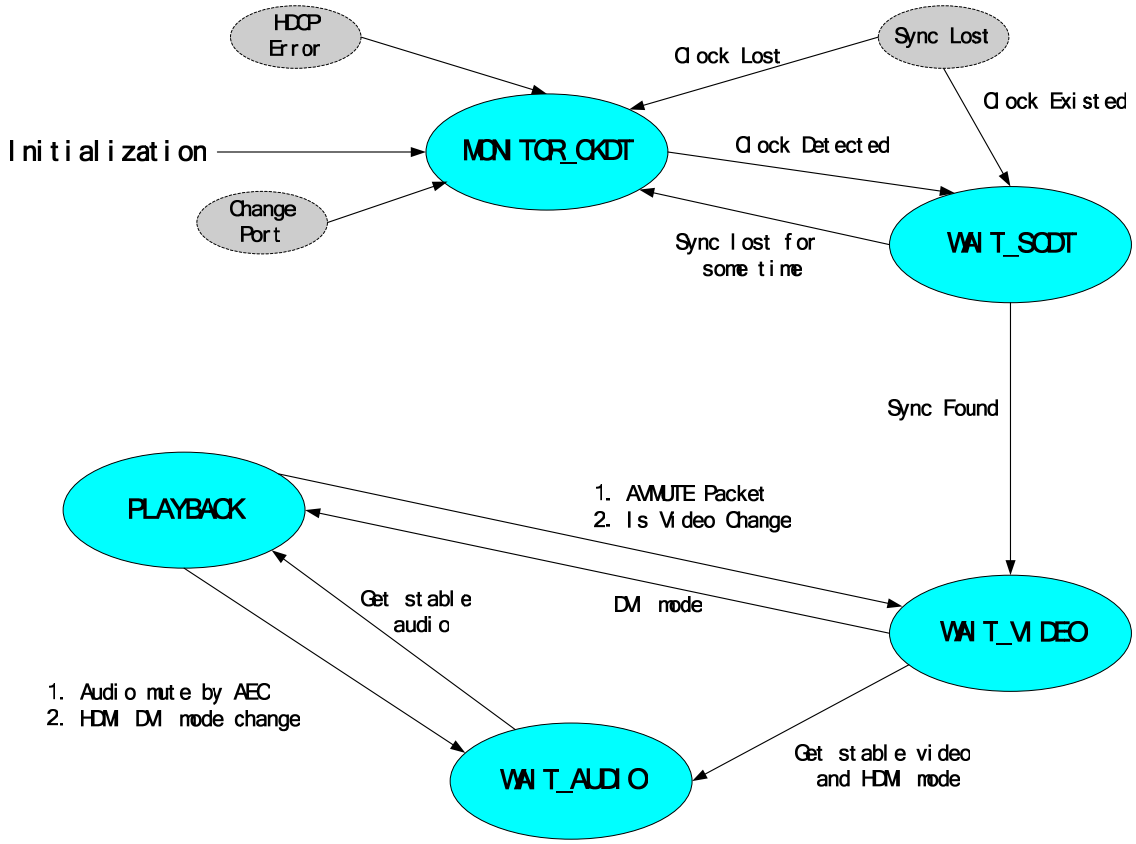

Figure 4 Firmware state transition diagram

Main Loop

The main function is as following:

```
void main (void) {
        init_mcu();
        init_9021_system();

        while (1) {
                . . .
               int_process ();
                timer_process ();
               // misc_process ();
         }
}
```

On above program, init_mcu() is to initial the microcontroller relate things, such as timer; init_9021_system() is to reset the ANX9021 chip and initialize the ANX9021 registers, int_process() is the interrupt event handler, timer_process() is to process the timer event.

Interrupt Control

The basic structure of the firmware is a loop, which monitors all ANX9021 interrupt events and calls the corresponding service routines for processing. The interrupt processing flow chart is shown as

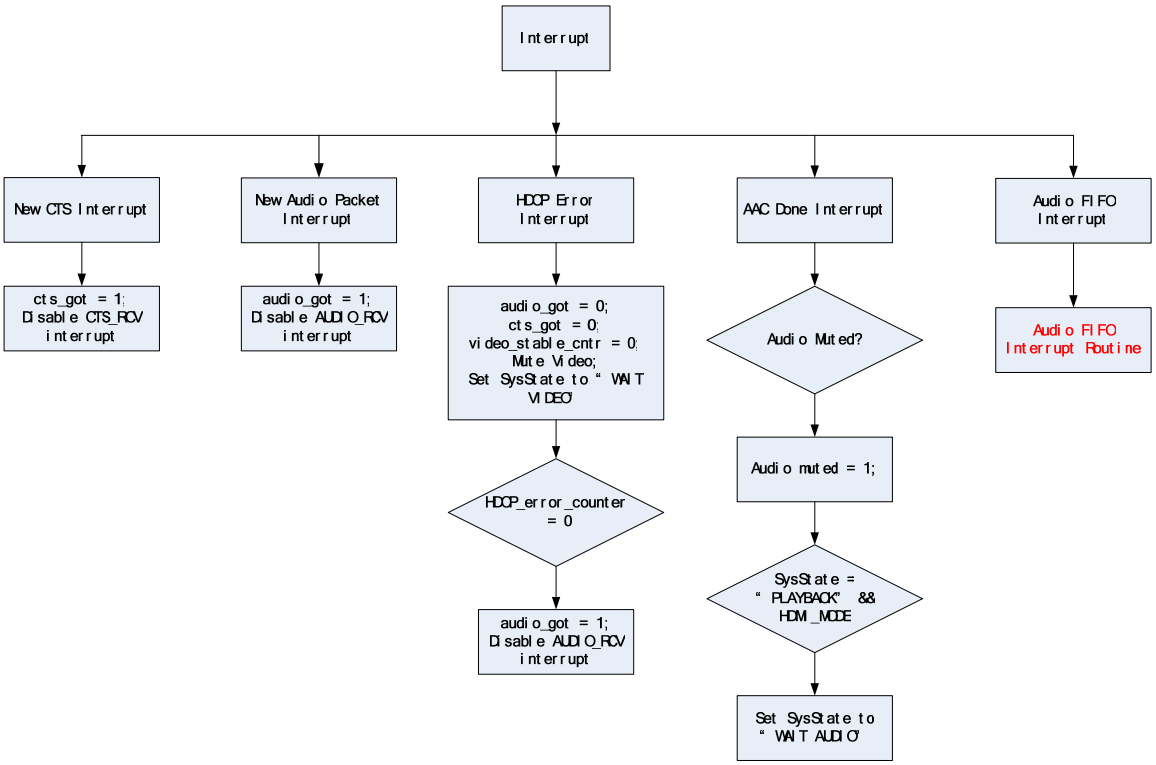

Figure 5 and Figure 6.

When the ANX9021 is in powered down mode, the only available interrupt event is TMDS clock detect (CKDT); so the firmware simply powers up the device (0x60: 0x08, set bit 0 to 1). When the device is powered up, firmware needs to query the interrupt status registers (0x60, offsets 0x71 – 0x74 and 0x7B, 0x7C) to determine the appropriate service routine.

Some processes, such as waiting for incoming video stream to be stable, takes a long time and so waiting for a process to complete before servicing another would introduce unacceptable latencies into the system. Therefore the firmware time multiplexes its activities into defined time slots.

The firmware includes a scheduler that divides pending tasks into four time slots (Slot0, Slot1, Slot2, Slot3), each slot lasts about 8 ms. Slot0 is reserved for video related processes, Slot1 is reserved for audio related processes, and Slot2 is used for error handling. Slot3 is reserved.

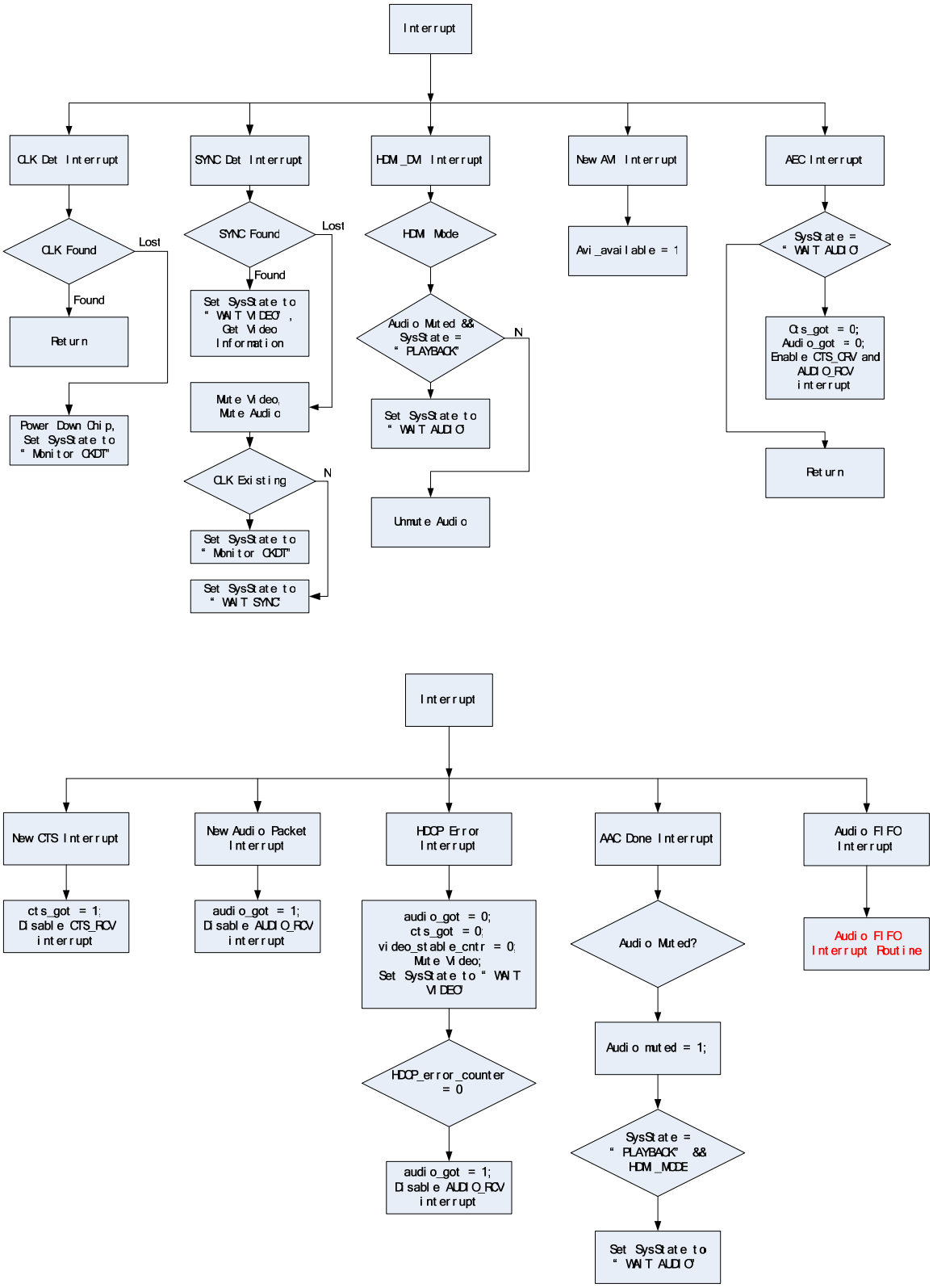

Figure 5 Interrupt processing flowchart

```
void ANX9021_Int_Process(void)
{
      BYTE rc, c, c1, c2, s1, s2, s3, s4, s5, s6, s7, tmp;

      ANX9021_ReadI2C_RX0(STATE_REG, &s7);
      if(s7 & 0x01)
      {
            ANX9021_Sync_Det_Int();
      }

      ANX9021_ReadI2C_RX0(STATE_REG, &c);

      if( c&0x02 )
      {

          if (sysState == MONITOR_CKDT)
          {

                  gm_Printf("ANX9021_Int_Process");

                  gm_Delay10ms(1);
                  ANX9021_WriteI2C_RX0(TMDS_PLL_RNG_CTRL_REG, 0x00);

                  // TMDS reset register(hidden register)
                  ANX9021_WriteI2C_RX0(0x95, 0x08);
                  gm_Delay100ms(4);
                  ANX9021_ReadI2C_RX0(TMDS_PLL_RNG_STATUS_REG, &c);

                  c1 = c & 0x61;
                  if(c&0x02)  c1 = c1 | 0x04;
                  if(c&0x04)  c1 = c1 | 0x10;
                  if(c&0x08)  c1 = c1 | 0x02;
                  if(c&0x10)  c1 = c1 | 0x08;
                  c1 = c1 | 0x80;

                  ANX9021_WriteI2C_RX0(TMDS_PLL_RNG_CTRL_REG, c1);

                  // TMDS reset register(hidden register)
                  ANX9021_WriteI2C_RX0(0x95, 0x08);
                  ANX9021_ReadI2C_RX0(SYS_CTRL1_REG, &c);
                  ANX9021_WriteI2C_RX0(SYS_CTRL1_REG, c & 0xfe);
                  gm_Delay1ms(5);
                  ANX9021_WriteI2C_RX0(SYS_CTRL1_REG, c | 0x01);

                  gm_Delay10ms(5);

                  ANX9021_Set_Sys_State(WAIT_SCDT);
                  ANX9021_ReadI2C_RX0(SYS_CTRL1_REG, &c);
                  ANX9021_WriteI2C_RX0 (SYS_CTRL1_REG, c | 0x01);
                  // power up all
            }
            else
```

```
{
      ANX9021_ReadI2C_RX0(INTR1_REG, &s1);
      ANX9021_WriteI2C_RX0(INTR1_REG, s1);
      ANX9021_ReadI2C_RX0(INTR1_MASK_REG, &c1);

      ANX9021_ReadI2C_RX0(INTR2_REG, &s2);
      ANX9021_WriteI2C_RX0(INTR2_REG, s2);
      ANX9021_ReadI2C_RX0(INTR2_MASK_REG, &c1);

      ANX9021_ReadI2C_RX0(INTR3_REG, &s3);
      ANX9021_WriteI2C_RX0(INTR3_REG, s3);
      ANX9021_ReadI2C_RX0(INTR3_MASK_REG, &c1);

      ANX9021_ReadI2C_RX0(INTR4_REG, &s4);
      ANX9021_WriteI2C_RX0(INTR4_REG, s4);
      ANX9021_ReadI2C_RX0(INTR4_MASK_REG, &c1);

      ANX9021_ReadI2C_RX0(INTR5_REG, &s5);
      ANX9021_WriteI2C_RX0(INTR5_REG, s5);
      ANX9021_ReadI2C_RX0(INTR5_MASK_REG, &c1);

      ANX9021_ReadI2C_RX0(INTR6_REG, &s6);
      ANX9021_WriteI2C_RX0(INTR6_REG, s6);
      ANX9021_ReadI2C_RX0(INTR6_MASK_REG, &c1);

      if (s2 & ANX9021_SCDT_CHANGE)
      {
            // SYNC detect interrupt
            gm_Printf("SYNC detect interrupt.");
            ANX9021_Sync_Det_Int();
      }

      if (s2 & ANX9021_HDMI_DVI_MODE_CHANGE)
      {
            // HDMI_DVI detect interrupt
            gm_Printf("HDMI-DVI mode change interrupt.");
            ANX9021_HDMI_DVI_Int();
      }

      if (s3 & ANX9021_NEW_AVI_DECT)
      {
            // New AVI interrupt
            gm_Printf("New avi interrupt.");
            avi_available =1;
            ANX9021_ReadI2C_RX0(IP_CTRL_REG, &c);
            ANX9021_WriteI2C_RX0(IP_CTRL_REG, c & 0xfe);
      }

      if (s1 & ANX9021_CTS_ACR_CHANGE)
      // some audio exeption
      {
            gm_Printf("Restart_audio_chk.");
```

```
                    ANX9021_Restart_Audio_Chk();
                }

                if (s2 & ANX9021_CTS_RECV)  // detect cts
                {
                    gm_Printf("CTS_rcv_int");
                    ANX9021_Cts_Rcv_Int();
                }

                if (s2 & ANX9021_AUDIO_RECV)  // detect audio
                {
                    gm_Printf("Audio packet received.");
                    ANX9021_Audio_Rcv_Int();
                }

                if (s4 & ANX9021_HDCP_ERROR)  // HDCP error
                {
                    gm_Printf("HDCP error int.");
                    ANX9021_HDCP_Error_Int();
                }

                if ((s4 & 0x01)  && (sysState == PLAYBACK))
                {
                    anx9021_fifo_e_cnt1 ++;
                }
                else{
                    anx9021_fifo_e_cnt1 = 0;
                }

                if ((s4 & 0x02)  && (sysState == PLAYBACK))
                {
                    anx9021_fifo_e_cnt2 ++;
                }
                else{
                    anx9021_fifo_e_cnt2 = 0;
                }

                if (s5 & ANX9021_AAC_MUTE)  // AAC done
                {
                    gm_Printf("Audio Auto Configure done int.");
                    ANX9021_Aac_Done_Int();
                }

            }
        }
}
```

Figure 6 Interrupt processing loop

Video Services

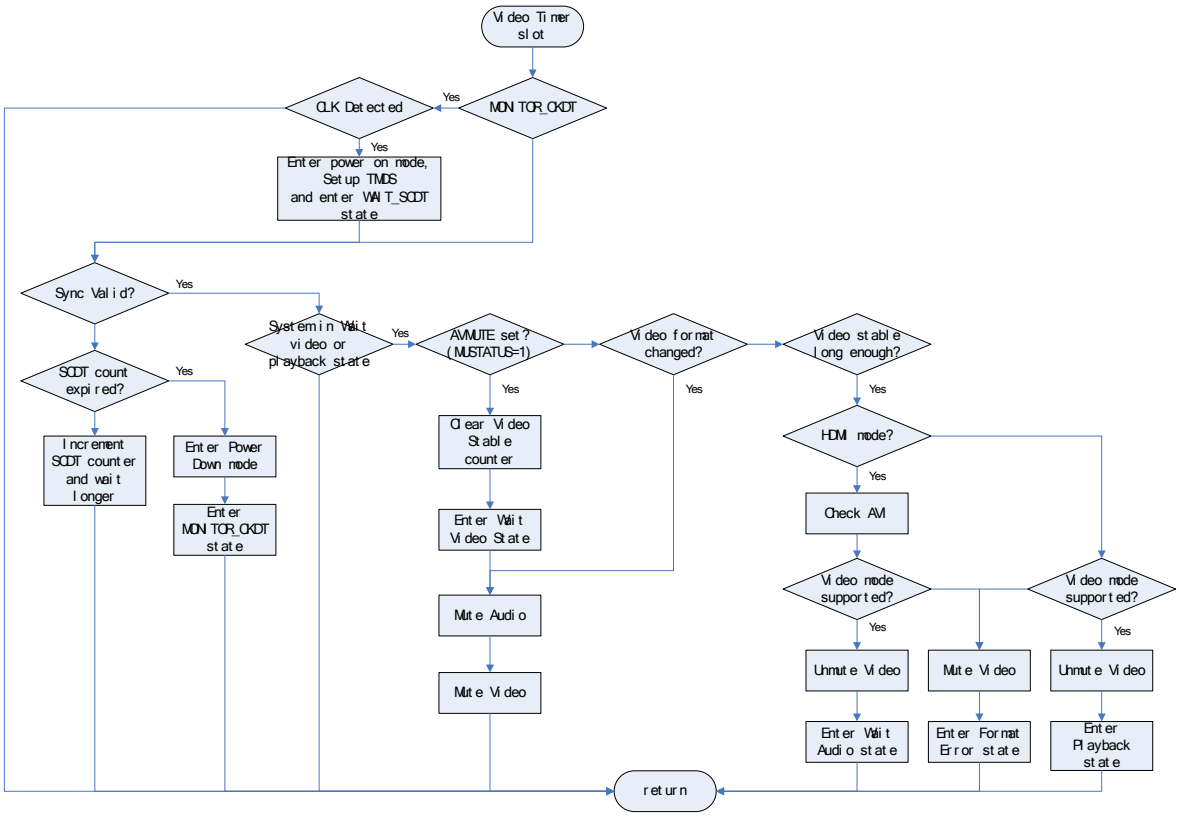

Figure 7 **Video time slot services**

The ANX9021 supports configurable video output formats controlled by the Video Output Format register (0x60, offset 0x5F). The appropriate output format depends on the formats support by the downstream video processor and is therefore specific to a system design.

HDMI supports a large variety of input video formats. Video format information is provided in an AVI InfoFrame to allow the firmware to quickly determine the input format. The ANX9021 supports automatic video configuration which is enabled by setting bit 5 of the AUTO register (0x60, offset 0xB5). It is recommended for the firmware to take advantage of this feature.

When the system is in the Wait Video state, it must determine that decoded video in stable before unmuting the video output. If the reported video timing is changing, the decoded video stream is unstable.

After the video is stable, the system enters the Wait Audio state if the input is HDMI or enters the Playback state if the input is DVI.

Auto Video Configuration

By analysis the arriving AVI InfoFrames and measuring the video streams, the ANX9021 can get most information of coming video, such as video format (RGB 4:4:4, YCbCr 4:4:4, YCbCr 4:2:2), H-resolution, V-resolution, etc. In AVC mode, once the microcontroller assigns the

desired video output format, ANX9021 will adapt its video data path to convert the incoming format to the output format automatically.
By setting the **Auto Control Register (0x60:0xB5), bit 2 to "1"**, the ANX9021 will work in AVC mode. Then user should set the **Auto Video Output Format Control Register (0x60:0x5F)** to the correct value.
For example, set the **Auto Video Output Format Control Register** to **0xC0**, the output format will be YCbCr 4:2:2 (16 Bit), with 16 bit data width, separate sync but without CLK4B, MUX_SYNC, and analog output. If the input video format is RGB 4:4:4, the registers showed in table 2 will be set by the AVC logic automatically.

Register Controlled By AVC					
Register Name	Address (0x60)	Field Name	Bit	Value	Comment
Video Mode Register 1	0x4A	INS_SYNC	7	0	
		MUX_YC	6	0	
		DITHER	5	1	
		RANGE_R2Y	4	1	
		CSPACE_R2Y	3	1	
		UP_SAMPLE	2	0	
		DOWN_SAMPLE	1	0	
		INS_CSYNC	0	0	
Video Mode Register 2	0x49	RANGE_Y2R	3	0	
		CSPACE_Y2R	2	0	
Video Control Register	0x48	INV_VSYNC	7	0	
		INV_HSYNC	6	0	
		CSYNC_VSYNC	5	0	
		CSYNC_HSYNC	4	0	
		CSPACE_Y2R	2	0	
		CSPACE_R2Y	0	1	

Table 2 Registers controlled by AVC Logic in AVC mode

About the detailed description of these three video control registers, please refer to the "ANX9021DS_r0-5.pdf", table 41~43, page 36~37.

Handling AVI InfoFrame

The detailed AVI InfoFrame is described in the HDMI 1.1 Specification (Section 8.2.1) and EIA/CEA-861B Specification (Section 6.1). The content of the AVI InfoFrame is as Table 3.

Packet Byte #	EIA/CEA-861B Byte #	7	6	5	4	3	2	1	0

PB0	N. A.	Checksum							
PB1	Data Byte 1	Rsvd (0)	Y1	Y0	A0	B1	B0	S1	S0
PB2	Data Byte 2	C1	C0	M1	M0	R3	R2	R1	R0
PB3	Data Byte 3	Reserved (0)						SC1	SC0
PB4	Data Byte 4	Rsvd (0)	VIC6	VIC5	VIC4	VIC3	VIC2	VIC1	VIC0
PB5	Data Byte 5	Reserved (0)				PR3	PR2	PR1	PR0
PB6	Data Byte 6	Line Number of End of Top Bar (lower 8 bits)							
PB7	Data Byte 7	Line Number of End of Top Bar (upper 8 bits)							
PB8	Data Byte 8	Line Number of start of Bottom Bar (lower 8 bits)							
PB9	Data Byte 9	Line Number of start of Bottom Bar (upper 8 bits)							
PB10	Data Byte 10	Pixel Number of End of Left Bar (lower 8 bits)							
PB11	Data Byte 11	Pixel Number of End of Left Bar (upper 8 bits)							
PB12	Data Byte 12	Pixel Number of End of Right Bar (lower 8 bits)							
PB13	Data Byte 13	Pixel Number of End of Right Bar (upper 8 bits)							
PB14-PB27	n. a.	Reserved (0)							

Table 3 AVI InfoFrame Packet Contents

The AVI InfoFrame will be carried in the control packet and be transmitted from the source to the sink. The ANX9021 will capture and store the AVI InfoFrame in the registers group which described in Table 4. These registers are read only. The firmware will get some information from these registers if necessary.

Register Address	Register Name	R/W	Default Value	Description
0x68:0x40	AVI_TYPE_CODE	RO	0x82	Defined by HDMI Spec 1.1.
0x68:0x41	AVI_VERSION_NUMBER	RO	0x02	Defined by HDMI Spec 1.1.

0x68:0x42	AVI_INFOFRAME_LENGTH	RO	0x0D	Length of AVI InfoFrame (13)
0x68:0x43	AVI_PB0	RO	0	Checksum
0x68:0x44	AVI_PB1	RO	0	AVI InfoFrame Data Bytes
0x68:0x45	AVI_PB2	RO		
0x68:0x46	AVI_PB3	RO		
0x68:0x47	AVI_PB4	RO		
0x68:0x48	AVI_PB5	RO		
0x68:0x49	AVI_PB6	RO		
0x68:0x4A	AVI_PB7	RO		
0x68:0x4B	AVI_PB8	RO		
0x68:0x4C	AVI_PB9	RO		
0x68:0x4D	AVI_PB10	RO		
0x68:0x4E	AVI_PB11	RO		
0x68:0x4F	AVI_PB12	RO		
0x68:0x50	AVI_PB13	RO		

Table 4 AVI InfoFrame Registers

Checking Video Formats

Incoming video format may change during playback. Firmware is required to detect the change and mute the video output until the new video stream is stable to avoid any viewable artifacts. To detect video format changes, firmware should monitor the following registers

- Video horizontal resolution, low byte (register 0x60, offset 0x3A)
- Video horizontal resolution, high byte (register 0x60, offset 0x3B)
- Video vertical resolution, low byte (register 0x60, offset 0x3C)
- Video vertical resolution, high byte (register 0x60, offset 0x3D)
- Video pixel clock counter (register 0x60, offset 0x6F)

The video format may be changed during the playing back state, and then the firmware should catch the change and do some corresponding process just like muting video and video before the new format video is stable. The firmware can catch the format change from the change of five registers. These registers are listed in table 5.

Register Name	Address	Description
Video Horizontal Resolution Low Byte Register	0x60:0x3A	
Video Horizontal Resolution High Byte Register	0x60:0x3B	
Video Vertical Resolution Low Byte Register	0x60:0x3C	
Video Vertical Resolution High Byte Register	0x60:0x3D	
Video Pixel Clock Counter Register	0x60:0x6F	

Table 5 Video Format Change Checking Related Registers

About the detailed description of these three video control registers, please refer to the "ANX9021DS_r0-5.pdf", table 39~40, page 35~36, and table 57,page 42.

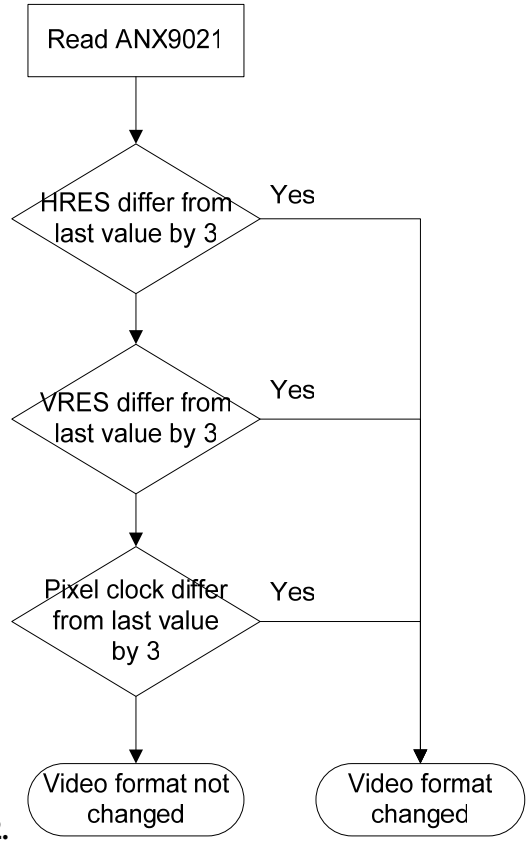

Figure 8 shows the flow chart of the firmware used to determine video format change. In every video time slot, firmware will fetch the values of these five registers and compare them against the values recorded from the previous time slot. If the results differ by more than 3, video format has changed.

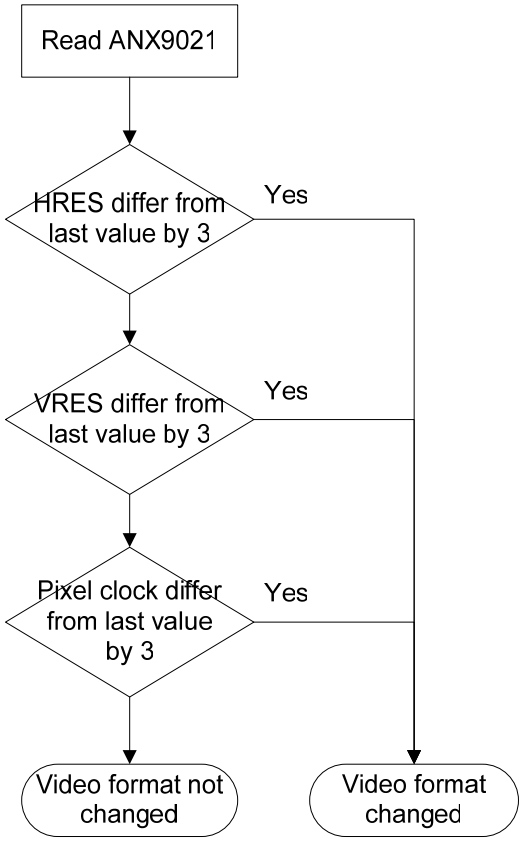

Figure 8 Video format change determination flow chart

Audio Services

In HDMI mode, after video is stable, the firmware will begin audio processing. Basically, video processing related registers are described in Table 6.

Register Name	Address	Bit	Value	Comment
HDCP receiver status shadow	0x60:0x30	4	1	Bit 4 shows whether the work mode is DVI or HDMI.
HDMI mute control register	0x68:0x37	4	0	Bit 4 shows whether the audio is mute or not.
Auto Control Register	0x60:0xB5	0	1	Set bit 0 = 1 to enable auto audio config.
Auto Control Register	0x60:0xB5	5	1	AAC will control I2S and SPDIF output.

Table 6 Basic Audio Processing Related Registers

If the ANX9021 is used to receive an HDMI data stream, it will have to process both video and audio information. If the incoming data is in DVI mode, only video data is involved.

Therefore, when the incoming format is DVI, the system enters the Playback state as soon as video becomes stable. When the incoming format is HDMI, the system enters the Wait Audio state. Only after audio also becomes stable does the system enters Playback.

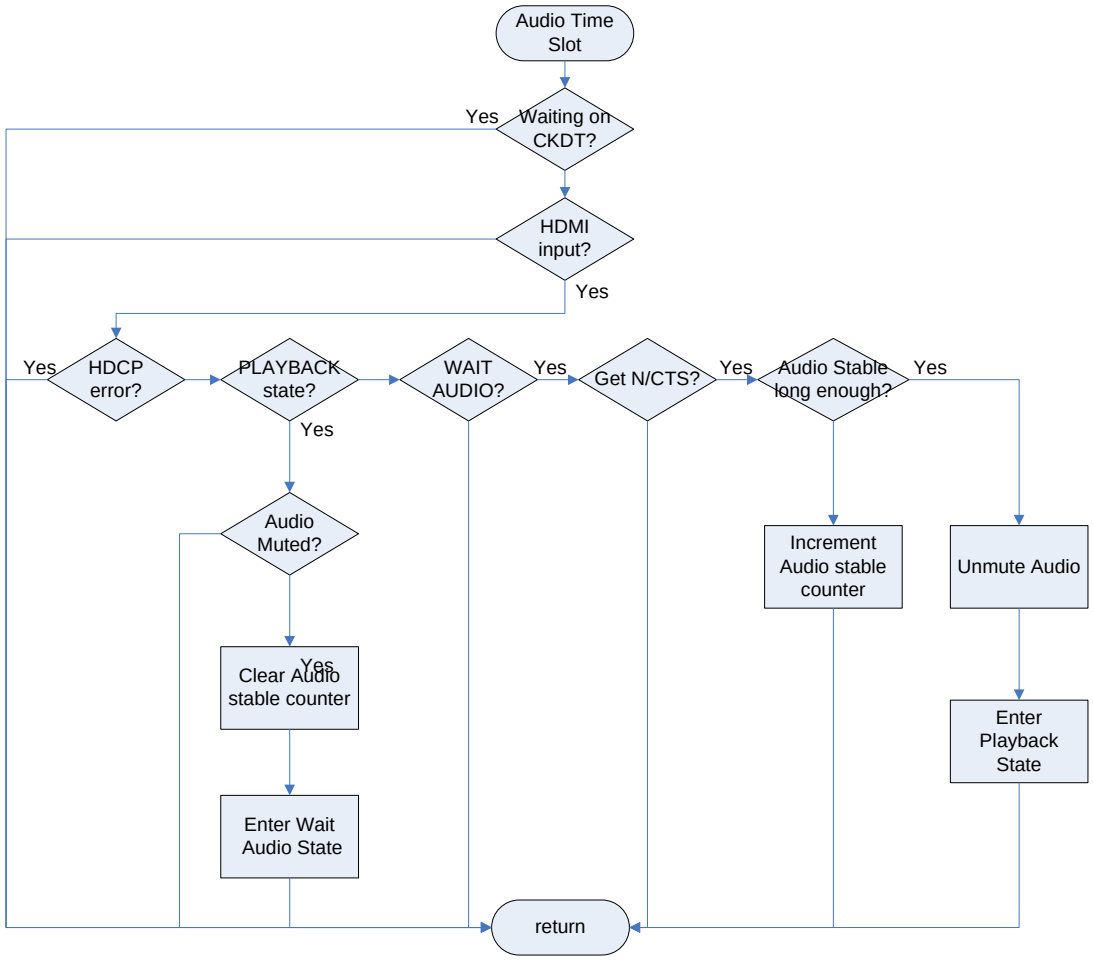

Figure 9 Audio time slot services

Figure 9 shows the audio time slot flow chart. The registers involved in audio processing follows:

- HDCP receiver status shadow (register 0x60, offset 0x30)
- HDMI mute (register 0x68, offset 0x17)
- Auto control (register 0x60, offset 0xB5)

Automatic Audio Control

The ANX9021 supports automatic audio control, which is enabled by setting AAC_OE and AAC_EN in the Auto Control register. When an audio exception occurs, ANX9021 hardware automatically mutes the audio output. Unmute is controlled by the firmware.

The related code can be found in the routine initial.

```
i2c_write_p0_reg (AEC_EN0_REG, 0xe7);
i2c_write_p0_reg (AEC_EN1_REG, 0xd9);
```

```
i2c_write_p0_reg (AEC_EN2_REG, 0x06);
i2c_write_p0_reg(AEC_CTRL_REG, 0x05);
```

Auto audio control is enable by setting AAC_EN described in Table 7. AAC_OE controls whether I2S and SPDIF output is influenced by AAC_EN.

Register Name	Address	Bit	Default Value	Description
AAC_OE	0x60:0xB5	5	0x0	AAC control of I2S and SPDIF output. 1 = enable AAC to control I2S and SPDIF output; 0 = AAC has no control of I2S and SPDIF.
AAC_EN	0x60:0xB5	0	0x0	AAC enable control. 1 = enabled; 0 = disabled.

Table 7 Auto audio control Related Registers

When AAC_EN is set, each bit of the following three registers enables a condition which triggers hardware soft mute. Besides, most of these conditions (except 2) also set corresponding interrupt bit. The details are listed in Table 8.

Register Name	Address	Bit	Default Value	Description
AEC_EN00	0x60:0xB6	0	0x0	Cable unplug exception enable control.
AEC_EN01	0x60:0xB6	1	0x0	PLL unlocked exception enable control.
AEC_EN02	0x60:0xB6	2	0x0	ACR N changed exception enable control.
AEC_EN03	0x60:0xB6	3	0x0	ACR CTS changed exception enable control.
AEC_EN04	0x60:0xB6	4	0x0	Video clock changed exception enable control.
AEC_EN05	0x60:0xB6	5	0x0	InforFrame CP mute set exception enable control.
AEC_EN06	0x60:0xB6	6	0x0	Sync detect exception enable control.
AEC_EN07	0x60:0xB6	7	0x0	Clock switch detect exception enable control.
AEC_EN08	0x60:0xB7	0	0x0	HDMI mode change exception enable control.
AEC_EN09	0x60:0xB7	1	0x0	Audio FIFO underrun exception enable control.
AEC_EN10	0x60:0xB7	2	0x0	Audio FIFO overrun exception enable control.
AEC_EN11	0x60:0xB7	3	0x0	CTS reused exception enable control.
AEC_EN12	0x60:0xB7	4	0x0	Fs changed exception enable control.
AEC_EN13	0x60:0xB7	5	0x0	Interlace changed exception enable control.
AEC_EN14	0x60:0xB7	6	0x0	Sync Polarity changed exception enable control.
AEC_EN15	0x60:0xB7	7	0x0	H resolution changed exception enable control.
AEC_EN16	0x60:0xB8	0	0x0	V resolution changed exception enable control.

AEC_EN17	0x60:0xB8	1	0x0	Link error exception enable control.
AEC_EN18	0x60:0xB8	2	0x0	Fn clock changed exception enable control.

Table 8 Auto Exception Enable Registers

Each audio exception enable bit listed in Table 8 enable one condition to trigger the hardware soft mute. Unmute is triggered by firmware.

Audio Output Configuation

Audio out general control is implemented by setting the registers list in Table 9.

Name	Address	Bit	Default Value	Description
PASS_SPDIF_ERR	0x68: 0x29	4	0x1	Pass SPDIF error control. 0 = do not pass SPDIF type of errors, conceal errors by repeating last good sample; 1 = pas all audio data, regardless of errors.
PASS_AUDIO_ERR	0x68: 0x29	3	0x1	Pass audio error control. 0 = do not pass errors, conceal errors by repeating last sample; 1 = pass all audio data, regardless of errors.
SOFT_MUTE_EN	0x68: 0x29	5	0x0	Soft mute enable. 1 = enable; 0 = disable.
CH3_MU	0x68: 0x32	3	0x0	Channel 3 mute.
CH2_MU	0x68: 0x32	2	0x0	Channel 2 mute.
CH1_MU	0x68: 0x32	1	0x0	Channel 1 mute.
CH0_MU	0x68: 0x32	0	0x0	Channel 0 mute.

Table 9 Audio out general control Registers

Audio out I2S control is implemented by setting the registers list in Table 10.

Name	Address	Bit	Default Value	Description
I2S_MODE	0x68:0x29	2	0x0	I2S output mode control. 0 = All I2S outputs are grounded (SD, SCK, WS); 1 = SCK and WS toggle, SD is on or off depending on the value in I2S_CTRL2.
SCK_EDGE	0x68:0x26	6	0x1	Sample clock edge select. 0 = sample edge is rising; 1 = sample edge is falling.
SIZE_SEL	0x68:0x26	5	0x0	Word size select. 1 = 16 bits; 0 = 32 bits.
MSB_SIGN_EXT	0x68:0x26	4	0x0	MSB sign extension enable control. 0 = enabled; 1 = disabled.
WS_POL	0x68:0x26	3	0x0	Word select left/right polarity select. 0 = left polarity when works select is low; 1 = left polarity when word select is high.
JUST_CTRL	0x68:0x26	2	0x0	SD Justification control. 1 = data is right

				justified; 0 = data is left justified.
DIR_CTRL	0x68:0x26	1	0x0	SD data Indian (MSB or LSB first) control. 0 = MSB first; 1 = LSB first.
SHIFT1	0x68:0x26	0	0x0	WS to SD shift first bit. 0 = fist bit shift (Philips Spec); 1 = no shift.
SD3_EN	0x68: 0x27	7	0x0	I2S Channel 3 output control. 0 = disabled (always output low); 1 = channel enabled.
SD2_EN	0x68: 0x27	6	0x0	I2S Channel 2 output control. 0 = disabled (always output low); 1 = channel enabled.
SD1_EN	0x68: 0x27	5	0x0	I2S Channel 1 output control. 0 = disabled (always output low); 1 = channel enabled.
SD0_EN	0x68: 0x27	4	0x0	I2S Channel 0 output control. 0 = disabled (always output low); 1 = channel enabled.
MCLK_EN	0x68: 0x27	3	0x0	MCLK enable. 0 = tri-state MCLKOUT; 1 = enable MCLKOUT.
VUCP_EN	0x68: 0x27	1	0x0	VUCP bits enable. 0 = send only 24 real data bits via I2S; 1 = send 28 bits of data with VUCP bist.
PCM_DET	0x68: 0x27	0	0x1	I2S data pass select. 0 = pass whatever data is in the S/PDIF packets; 1 = pass only data from S/PDIF packets which are recognized as PCM data. When non-PCM data is detected, send 0.
SD3_MAP	0x68: 0x28	7:6	0x3	I2S Channel 3 data stream select. 0 = stream 0; 1 = stream 1; 2 = stream 2; 3 = stream 3;
SD2_MAP	0x68: 0x28	5:4	0x2	I2S Channel 2 data stream select. 0 = stream 0; 1 = stream 1; 2 = stream 2; 3 = stream 3;
SD1_MAP	0x68: 0x28	3:2	0x1	I2S Channel 1 data stream select. 0 = stream 0; 1 = stream 1; 2 = stream 2; 3 = stream 3;
SD0_MAP	0x68: 0x28	1:0	0x0	I2S Channel 0 data stream select. 0 = stream 0; 1 = stream 1; 2 = stream 2; 3 = stream 3;
SW3	0x68: 0x2E	7	0x0	Swap left/right channel on I2S channel 3. 1 = swap; 0 = no swap.
SW2	0x68: 0x2E	6	0x0	Swap left/right channel on I2S channel 2. 1 = swap; 0 = no swap.
SW1	0x68: 0x2E	5	0x0	Swap left/right channel on I2S channel 1. 1 = swap; 0 = no swap.
SW0	0x68: 0x2E	4	0x0	Swap left/right channel on I2S channel 0. 1 = swap; 0 = no swap.

LENOV	0x68: 0x29	7	0x0	Enable overwrite of the audio sample length for the I2S data. 0 = take from the HDMI packet; 1 = take form AUDO_MUTE bit [7:4].
LEN_OVERRIDE	0x68: 0x32	7:4	0x0	Audio word length override value. When LENOV = 1, this value is used to set the word length for I2S output instead of the length extracted from channel status bits.

Table 10 Audio out I2S control Registers

Audio out S/PDIF control is implemented by setting the registers list in Table 11.

Name	Address	Bit	Default Value	Description
SP_MODE	0x68: 0x29	1	0x0	SPDIF flat line enable. 0 = SPDIF output always produces valid bi-phase mark encoded data, even during flat line; 1 = SPDIF output is zero (grounded) if flat line is detected.
SP_EN	0x68: 0x29	0	0x0	SPDIF output enable. 1 = enabled; 0 = disabled (output flat line or zero according to SP_MODE)

Table 11 Audio out S/PDIF control Registers

Audio in S/PDIF channel status can be read out through the registers listed in Table 12

Name	Address	Bit	Default Value	Description
MODE	0x68: 0x2A	7:6	0x0	00 = PCM Audio
PRE_EMPHASIS	0x68: 0x2A	5:3	0x0	000 = 2 audio channels without pre-emphasis; 001 = 2 audio channels with 50/15 usec pre-emphasis ???
SW_CPRGT	0x68: 0x2A	2	0x0	0 = software for which copyright is asserted; 1 = software for which no copyright is asserted
NON_PCM	0x68: 0x2A	1	0x0	0 = audio sample word represents linear PCM samples; 1 = audio sample word used for other purposes.
PROF_APP	0x68: 0x2A	0	0x0	0 = consumer applications; 1 = professional applications.
CAT_CODE	0x68: 0x2B	7:0	0x0	Category code (corresponding to channel status bits [15:8])

CH_NUM	0x68: 0x2C	7:4	0x0	Channel number (corresponding to channel status bits [23:20])
SOURCE_NUM	0x68: 0x2C	3:0	0x0	Source number (corresponding to channel status bits [19:16])
FS_FREQ	0x68: 0x30	3:0	0x0	Sampling clock frequency (corresponding to channel status bits [27:24]). 0000 = 44.1 KHz; 0010 = 48 KHz; 0011 = 32 KHz; 1000 = 88.2 KHz; 1010 = 96 KHz; 176.4 KHz; 1110 = 192 KHz; others = reserved.
CLK_ACCUR	0x68: 0x30	5:4	0x0	Clock accuracy (corresponding to channels status bits [29:28]). These two bits define the sampling frequency tolerance. The bits are set in the transmitter.
AUD_LENGTH	0x68: 0x31	7:4	0x0	Audio word length (corresponding to channel status bits [35:33]). When AUD_MX = 0, 000 = 16 bits; 010 = 18 bits; 100 = 19 bits; 101 = 20 bits; 110 = 17 bits; when AUD_MX = 1, 001 = 20 bits; 010 = 22 bits; 100 = 23 bits; 101 = 24 bits; 110 = 21 bits.
AUD_MX	0x68: 0x31	3:1	0x0	Audio word length Max (corresponding to channel status bits 32). 0 = maximal word length is 20 bits; 1 = maximal word length is 24 bits.

Table 12 Audio in S/PDIF channel status Registers

Audio status overwrite registers are listed in table 13.

Name	Address	Bit	Default Value	Description
OW_CHEN	0x68: 2E	0	0x0	Channel status overwrite enable. 0 = no overwrite to channel status bits; 1 = channel status bits 2 and 8-15 are overwritten with the values in OW_B2 and OW_CHST5
OW_B2	0x68: 2E	2	0x0	Channel status Bit 2 overwrite data. This bit value is used for CHST1 Bit 2 when OW_CHEN = 1.
OW_CHST5	0x68: 2F	7:0	0x0	Channel status byte 5 overwrite data. This field value is used for overwriting channel status Bits [15:8] when OW_CHEN is 1.

Table 13 Audio status overwrite registers