IT6604 HDMI 1.4 3D Receiver Programming Guide

Ver 0.95

Tseng, Jau-Chih

ITE Tech.

Last Update Date: 2010/08/11

History

2010/08/10 – Created by Tseng, Jau-chih
2010/08/11 – Added Audio Part.

Index

Chap 1 Introduction

The IT6604 is a single-port HDMI 1.4 receiver. The IT6604 with its Deep Color capability (up to 36-bit) ensures robust reception of high-quality uncompressed video content, along with state-of-the-art uncompressed and compressed digital audio content such as DTS-HD and Dolby TrueHD in digital televisions and projectors.

Aside from the various video output formats supported, the IT6604 also receives and provides up to 8 channels of I^2S digital audio outputs, with sampling rate up to 192kHz and sample size up to 24 bits, facilitating direct connection to industry-standard low-cost audio DACs. Also, an S/PDIF output is provided to support up to compressed audio of 192kHz frame rate. Super Audio Compact Disc (SACD) is supported at up to 8 channels and 88.2kHz through DSD (Direct Stream Digital ports) ports.

Each IT6604 comes preprogrammed with an unique HDCP key, in compliance with the HDCP 1.2 standard so as to provide secure transmission of high-definition content. Users of the IT6604 need not purchase any HDCP keys or ROMs.

To program IT6604 need using I^2C access the PCSDA (pin24) and PCSCL (pin25) with the frequency under 100KHz. The I^2C address for accessing internal registers are 0x90 or 0x92 depends on the PCADR (pin93) value.

To access the IT6604 internal registers should by the following protocol:

Read:
<I^2C start>-<0x90|w >-<register index>-<I^2C repeater start>-<0x90|r>-<data>(-…-<data>)-<I^2C Stop>

Write:
<I^2C start>-<0x90|w >-<register index>-<data>(-…-<data>)-<I^2C Stop>

In the following document, the register with index will present as Reg<idx>.

Eg: Reg05 means the register with index 0x05.

Chap 2 Initial Progress

The first step of initial IT6604 is to reset the chip.

Activate SYSRSTN (pin89) with low voltage or write reg05[4] = '1', will reset the chip.

When SYSRSTN is high voltage and reg05[4] = '0', IT6604 is under normal operating mode.

Initial Progress

1. Set HPD (HDMI Connection Pin19) to low (if possible).
2. Reg06 = 0x00 to power on all modules.
3. Reg07[3:2] = '11' to turn off the termination.
4. Reg05 = 0x81
5. Reg16 = 0x0F
6. Reg17 = 0x07
7. Reg18 = 0x07
8. Reg8C = 0x20 (5~8 is for initial interrupt mask setting)
9. Load the default value.
10. Reset the HDCP ROM.
11. Configure the HDCP repeater setting
    i) Receiver mode, reg73[7:4] = '0000'
    ii) Repeater mode, reg73[7:4] = '1000'
12. Delay about 500ms to make sure the HPD off enough.
13. Reg07[3:2] = '00'
14. Set HPD to high (if possible).

Initial Value List

reg08 = 0xAE
reg1D = 0x20
reg3B=0x40
reg68=0x03
reg6B=0x11
Reg6C=0x00
Reg93=0x43
Reg94=0x4F
Reg95=0x87
Reg96=0x33

Reg1A/Reg1B/Reg1C/Reg3D[7:6] for the output video format, refer to Video Output.
Reg75/Reg76/Reg78 for the output audio format, refer to Audio Output.

HDCP ROM Reset

To reset HDCP ROM read, by the following step.

1. counter = 0.
2. reg9B[7:6] = '10'
3. reg9B[7:6] = '11'
4. counter = counter + 1 ;
5. if counter < 16, go to step 1.
6. reg9B[7:6] = '00'.

```
void
RDROM_Reset()
{
    BYTE i ;
    BYTE uc ;

    uc = HDMIRX_ReadI2C_Byte(0x9B) & ~(0xC0) ;
    for(i=0 ;i < 16 ; i++ )
```

```
    {
       HDMIRX_WriteI2C_Byte(0x9B, 0x80|uc) ;
       HDMIRX_WriteI2C_Byte(0x9B, 0xC0|uc) ;
    }
    HDMIRX_WriteI2C_Byte(0x9B,uc) ;
}
```

Chap 3 Video Output

IT6604 receive the HDMI or DVI video input, and output up to 30 bit TTL with numerous format, this chapter describe how to configure the video path and video output.

Video Output Flow

IT6604 output video when it got a valid video input.

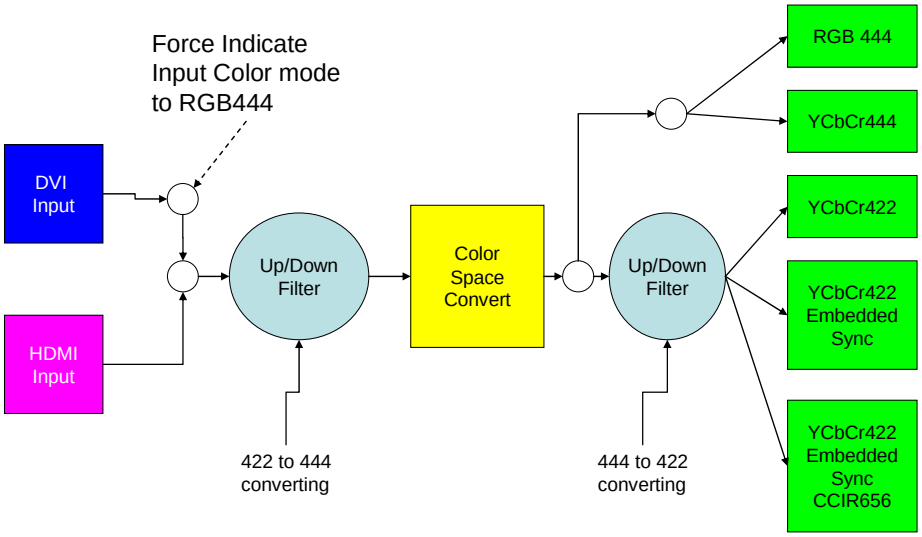
10	Sys_state	RO	7	RXPLL_LOCK	HDMI PLL locked
			6	RXCK_Speed	'1' – Lower than 80MHz '0' – Higher than 80MHz
			5	RXCK_VALID	'1' – RX CLK Valid
			4	HDMI_MODE	'1' – HDMI Mode '0' – DVI Mode Only reliable when SCDT on.
			3	P1_PWR5V_DET	(For CAT6023/IT6605) '1' for HDMI input port 1 with 5V presented
			2	SCDT	'1' – Sync Detected. '0' – Otherwise.
			1	VCLK_DET	'1' – VCLK Detect '0' – Otherwise
			0	PWR5V_DET	'1' - HDMI input port 0 with 5V present.

The following steps are for getting the valid video output:

1. IT6604 should detect 5V in corresponding HDMI port.
2. IT6604 gets valid VCLK (with VCLK_Valid status bit present).
   i) When the first valid VCLK detected, the CDR_Reset should be called to reset the clock and data recovery FIFO.
      → reg97[5] = '1'
      → reg5[7][1] = '1' '1'
      → reg73[3] = '1'
      → reg97[5] = '0'
      → reg5[7][1] = '0' '0'
      → reg73[3] = '0'
3. IT6604 gets valid SCDT (with SCDT status bit present and no SCDT off interrupt present).
4. Configure the video path of IT6604.
5. Reset video FIFO (reg1C[1] = '1' → '0').
6. Turn off the video I/O and video data tri-state.

Video Path

The video path of IT6604 are defined with video input, color space converting, and video output signal format, as the figure:



HDMI input contain up to 36 bits (which IT6604 supported) RGB444, YCbCr444, or YCbCr422 TMDS input with AVI infoframe indicated. DVI input supports only RGB444 video input. The input color are indicated by reg20 or AVI infoframe, and convert to output color space by color space converting matrix (CSC Matrix), then output by color mapping registers and output format controls registers

1B	Video_map	W/R	6 5 4 3 2 1 0	chSyncpol Swap_O16b Swap_Ch422 Swap_OutRB Swap_ML Swap_Pol Swap_RB	0000000	Referring to the map table of Emily.
1C	Video_Ctrl1	W/R	7 6 5 4 3 2 1 0	DNFreeGo SyncEmb EN_Dither EnUdFilt OutDDR 2x656CLK 656FFRst EnAVMuteRst	0x00	Default: dithering and up/dn filter is **enabled** **2x656CLK:** **1:an 2x 656CLK is generated by PLL** **0:no 2x656 CLK**

The detail step are describe in following section.

Video Input Selection

When SCDT and RXCK_Valid are both present, the video input is reliable. If the HDMI_MODE bit is '0', the input is an DVI input mode and default treated as 24bit RGB444 video input, otherwise the input mode should be explained as an HDMI mode with AVI infoframe supported. If the AVI infoframe is not presented, some testing case ask HDMI Rx to receive the input video as an RGB444 mode.

20	CSC_CTRL	W/R	7 6 5 4 3:2 1:0	VDGatting VDIOLDisable VIOSel ForceColMod ColMod_Set CSCSel	0x00	**ForcecolmodI:** 0:color mode auto adjusted according to AVI info 1: color mode is forced by register

For DVI mode, the register setting should be reg20[4] = '1' to ignore the AVI Infoframe bit value, and set reg20[3:2] = '00' as RGB444 mode.

For HDMI mode, to refer the AVI infoframe color mode, reg20[4] should be '0' that IT6604 will refer the AVI infoframe PB[1][6:5] as input color mode.

The received AVI infoframe of IT6604 is stored in regAD ~ regBA with 13 bytes.

AB	AVI_leng	RO	InfoFrame Type Code	InfoFrame Type = 02_{16}						
AC	AVI_VER	RO	InfoFrame Version Number	Version = 01_{16}						
AD	AVI_DB0	RO	Length of AVI InfoFrame	Length of AVI InfoFrame (13)						
AE	AVI_DB1	RO	Data Byte 1	Rsvd=0	Y1	Y0	A0	B1	B0	S1	S0
AF	AVI_DB2	RO	Data Byte 2	C1	C0	M1	M0	R3	R2	R1	R0
B0	AVI_DB3	RO	Data Byte 3	Reserved for Future (shall be 0)						SC1	SC0
B1	AVI_DB4	RO	Data Byte 4	Reserved for Future (shall be 0)							
B2	AVI_DB5	RO	Data Byte 5	Reserved for Future (shall be 0)							
B3	AVI_DB6	RO	Data Byte 6	Line Number of End of Top Bar (lower 8 bits)							
B4	AVI_DB7	RO	Data Byte 7	Line Number of End of Top Bar (upper 8 bits)							
B5	AVI_DB8	RO	Data Byte 8	Line Number of Start of Bottom Bar (lower 8 bits)							
B6	AVI_DB9	RO	Data Byte 9	Line Number of Start of Bottom Bar (upper 8 bits)							
B7	AVI_DB10	RO	Data Byte 10	Pixel Number of End of Left Bar (lower 8 bits)							
B8	AVI_DB11	RO	Data Byte 11	Pixel Number of End of Left Bar (upper 8 bits)							
B9	AVI_DB12	RO	Data Byte 12	Pixel Number of Start of Right Bar (lower 8 bits)							
BA	AVI_DB13	RO	Data Byte 13	Pixel Number of Start of Right Bar (upper 8 bits)							

Where Y1/Y0 are defined in CEA861/B spec, as following figure:

EIA/CEA-861-B

Table 8. AVI InfoFrame Data Byte 1

F7	Future Use, all Zeros
0	

Y1	Y0	RGB or YCbCr
0	0	RGB (default)
0	1	YCbCr 4:2:2
1	0	YCbCr 4:4:4
1	1	Future

A0	Active Format Information Present
0	No Data
1	Active Format Information valid

B1	B0	Bar Info
0	0	Bar Data not valid
0	1	Vert. Bar Info valid
1	0	Horiz. Bar Info Valid
1	1	Vert. and Horiz. Bar Info valid

S1	S0	Scan Information
0	0	No Data
0	1	Overscanned (television)
1	0	Underscanned (Computer)
1	1	Future

Data Byte 2 (Table 9) contains bits that describe colorimetry, picture aspect ratio, and the active format information.

IT6604 will refer the input color space by regAE[6:5] or reg20[3:2] by reg20[4] selection, to decide the decoding of input colors.

Video Output Configuration

The video output format are controlled by reg1B and reg1C, and the output selection are controlled as

following table:

1B	Video_map	W/R	6 5 4 3 2 1 0	chSyncpol Swap_O16b Swap_Ch422 Swap_OutRB Swap_ML Swap_Pol Swap_RB	0000000	Referring to the map table of Emily.
1C	Video_Ctrl1	W/R	7 6 5 4 3 2 1 0	DNFreeGo SyncEmb EN_Dither EnUdFilt OutDDR 2x656CLK 656FFRst EnAVMuteRst	0x00	Default: dithering and up/dn filter is **enabled** **2x656CLK:** **1:an 2x 656CLK is generated by PLL** **0:no 2x656 CLK**

	Reg1B Video_map								Reg1C Video_Ctrl1							
		6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
		chSyncpol	Swap_O16b	Swap_Ch422	Swap_OutRB	Swap_ML	Swap_Pol	Swap_RB	DNFreeGo	SyncEmb	EN_Dither	EnUdFilt	OutDDR	2x656CLK	656FFRst	EnAVMuteRst
Sync separated RGB444					X	X		X								
Sync separated YCbCr444					X	X		X								
Sync separated 24 bit YCbCr422				X				X								
Sync separated 16 bit YCbCr422			√	X				X	√		√	√				
Sync Embedded 16 bit YCbCr 422			√	X				X	√	√	√	√				
Sync Embedded 8 bit YCbCr 422 (CCIR656)			√	X				X	√	√	√	√		√		

√ : Must set

X : Depends on the output mapping.

The above setting only decide the output signal format, the output color space are defined in the reg3D, for determining the output:

Reg	Name	Type	Bit	Name	Default value	Description.
3D	PG_CTRL2	W/R	7:6	OutColMod	10	'00' – RGB444 '01' – YCbCr422 '10' – YCbCr444

To set RGB444, YCbCr422, or YCbCr444 also need to set reg3D[7:6] value.

Usually the output configuration is fixed in one type, thus the setting could be the initial value.

Color Space Matrix

The color space register are defined in reg21~reg35, as following:

20	CSC_CTRL	W/R	7 6 5 4 3:2 1:0	VDGatting VDIOLDisable VIOSel ForceColMod ColMod_Set CSCSel	0x00	**Forcecolmod:** 0:color mode auto adjusted according to AVI info 1: color mode is forced by register
21	CSC_YOFF	W/R	7:0		0x10	
22	CSC_COFF	W/R	7:0		0x80	
23	CSC_RGBOFF	W/R	7:0		0x00	
24	CSC_MTX11_L	W/R	7:0		0xb2	
25	CSC_MTX11_H	W/R	5:0		000100	
26	CSC_MTX12_L	W/R	7:0		0x64	
27	CSC_MTX12_H	W/R	5:0		000010	
28	CSC_MTX13_L	W/R	7:0		0xE9	
29	CSC_MTX13_H	W/R	5:0		000000	
2A	CSC_MTX21_L	W/R	7:0		0x93	
2B	CSC_MTX21_H	W/R	5:0		011100	
2C	CSC_MTX22_L	W/R	7:0		0x16	
2D	CSC_MTX22_H	W/R	5:0		000100	
2E	CSC_MTX23_L	W/R	7:0		0x56	
2F	CSC_MTX23_H	W/R	5:0		011111	
30	CSC_MTX31_L	W/R	7:0		0x49	
31	CSC_MTX31_H	W/R	5:0		011101	
32	CSC_MTX32_L	W/R	7:0		0x9f	
33	CSC_MTX32_H	W/R	5:0		011110	
34	CSC_MTX33_L	W/R	7:0		0x16	
35	CSC_MTX33_H	W/R	5:0		000100	

And the value of color space converting setting are as following table:

Color space converting table		RGB to YUV				YUV to RGB			
		RGB to YUV 601		RGB to YUV 709		YUV to RGB 601		YUV to RGB 709	
	reg	16~ 235	0 ~ 255	16~ 235	0 ~ 255	16~ 235	0 ~ 255	16~ 235	0 ~ 255
Reg_CSCSel[1:0]	20[1:0]	10	10	10	10	11	11	11	11
Reg_YoffSet[7:0]	reg21	0x00	0x10	0x00	0x10	0x00	0x10	0x00	0x10
Reg_CoffSet[7:0]	reg22	0x80	0x80	0x80	0x80	0x80	0x80	0x80	0x80
Reg_RGBOffSet[7:0]	reg23	0x00	0x10	0x00	0x10	0x00	0x10	0x00	0x10
Reg_Matrix11V[13:0]	reg24	0xB2	0x09	0xB8	0xE5	0x00	0x4F	0x00	0x4F
	reg25	0x04	0x04	0x05	0x04	0x08	0x09	0x08	0x09
Reg_Matrix12V[13:0]	reg26	0x64	0x0E	0xB4	0x78	0x6A	0x81	0x53	0xBA
	reg27	0x02	0x02	0x01	0x01	0x3A	0x39	0x3C	0x3B
Reg_Matrix13V[13:0]	reg28	0xE9	0xC8	0x93	0x81	0x4F	0xDF	0x89	0x4B
	reg29	0x00	0x00	0x00	0x00	0x3D	0x3C	0x3E	0x3E
Reg_Matrix21V[13:0]	reg2A	0x93	0x0E	0x49	0xCE	0x00	0x4F	0x00	0x4F
	reg2B	0x3C	0x3D	0x3C	0x3C	0x08	0x09	0x08	0x09
Reg_Matrix22V[13:0]	reg2C	0x18	0x84	0x18	0x84	0xF7	0xC2	0x51	0x56
	reg2D	0x04	0x03	0x04	0x03	0x0A	0x0C	0x0C	0x0E
Reg_Matrix23V[13:0]	reg2E	0x56	0x6E	0x9F	0xAE	0x00	0x00	0x00	0x00
	reg2F	0x3F	0x3F	0x3F	0x3F	0x00	0x00	0x00	0x00
Reg_Matrix31V[13:0]	reg30	0x49	0xAC	0xD9	0x49	0x00	0x4F	0x00	0x4F
	reg31	0x3D	0x3D	0x3C	0x3D	0x08	0x09	0x08	0x09
Reg_Matrix32V[13:0]	reg32	0x9F	0xD0	0x10	0x33	0x00	0x00	0x00	0x00
	reg33	0x3E	0x3E	0x3F	0x3F	0x00	0x00	0x00	0x00
Reg_Matrix33V[13:0]	reg34	0x18	0x84	0x18	0x84	0xDB	0x1E	0x87	0xE7

	reg35	0x04	0x03	0x04	0x03	0x0D	0x10	0x0E	0x10

Video I/O and Video Data I/O Tristate

The video output I/O and video data I/O have tristate control to disable the video output. Whenever the AV mute detected, video output should be disabled.

IT6604 implement the automatic mute mechanism, as the following registers:

20	CSC_CTRL	W/R	7 6 5 4 3:2 1:0	VDGatting VDIOLDisable VIOSel ForceColMod ColMod_Set CSCSel	0x00	**Forcecolmod:** 0:color mode auto adjusted according to AVI info 1: color mode is forced by register
89	TriState_Ctrl	W/R	7 6 5 4 3:0	DisVAutomute TriVdIO Tri_vdo Tri_SPDIF Tri_I2S	0x80	**Would be removed if not necessary**

If reg89[6] = '1', all video clock, H/V sync and data are tri-stated. When reg89[7] = '1', AVMute will not affect the video output, otherwise when AVMute = '1', the video output is tri-stated automatically.

If reg89[7] = '1', and AVMute (reg65[2] represent the AVMute status transmit by HDMI general packet) switched from '1' to '0', the video enable should following the procedure:

1. reg1B[1] = '1'
2. reg1B[1] = '0'
3. reg89[6] = '1'
4. reg89[6] = '0'
5. reg20[7] = '1'
6. reg20[7] = '0'

Then the video output will not be tri-stated and the output is available.

Event of Video Process

There are several flags about video status, defined in reg15(interrupt status 0) and reg8B(interrupt status 3), as following:

Reg Offset	Reg_Name	W/R	Bits	Status	Description
0x13	Interrupt0	RO	5 4 3 2 1 0	VidMode_Chg HDMIMode_Chg SCDTOFF SCDTON Pwr5VOff Pwr5Von	Video mode change HDMI/DVI mode swap change Video stable is off Video stable is on Selected port 5V is off Selected port 5V is on
0x14	Interrupt Mask 0	R/W	5 4 3 2 1 0	VidMode_Chg HDMIMode_Chg SCDTOFF SCDTON Pwr5VOff Pwr5Von	1 : enable the int signal by the event.
0x8B	Interrupt3	RO	7 6 5 4 3 2 1 0	CLKCHG_DET rxckon_Det HDCPoff_det Symerr_det CD_det Genpkt_det ISRC2_Det ISRC1_Det	Rx clock change detect Int Rx clock on detect Int
0x8C	Interrupt mask 3	R/W	7 6	CLKCHG_DET HDCPoff_det_mask	1 : enable the int signal by the event.

			5	rxckon_Det_mask	
			4	Symerr_det	
			3	CD_det	
			2	Genpkt_det	
			1	ISRC2_Det	
			0	ISRC1_Det	

When the bit in reg13 or reg8B[7:6] raised, set reg19[0] = '1' → '0' will clear it include the INT signal.

Above event can be referred as video input status change status, the status in reg10 should be referred together when handle the video mode interrupt.

Video input mode resolution

IT6604 provided the input video status registers including the video mode information, the information are located in reg58~reg65, as the following table:

0x58	Vid_mode	RO	3	PxVideoStable	Indicate if video signal is stable
			2	vidfield	Video field number in interlaced mode
			1	vidinterlacemode	Indicate video is in interlaced mode
			0	VidModeChg	Indicate if a video mode change occurs
0x59	Vid_HTotal_L	RO	7:0		The total pixel count of a line [7:0]
0x5A	Vid_HTotal_H	RO	7:4	Vid_HAct[11:8]	The active pixel count of a line[11:8]
			3:0	Vid_Htotal[11:8]	The total pixel count of a line [11:8]
0x5B	Vid_HAct_L	RO	7:4	Vid_HAct[7:0]	The active pixel count of a line[7:0]
0x5C	Vid_Hsync_Wid_L	RO	7:4	Vid_Hsync_Wid[7:0]	The width of Hsync [7:0]
0x5D	Vid_HSync_Wid_H	RO	7:4	Vid_H_Ft_Porch[11:8]	The width of Hsync front porch [11:8]
			3:0	Vid_Hsync_Wid[11:8]	The width of Hsync [11:8]
0x5E	Vid_H_Ft_Porch_L	RO	7:0	Vid_H_Ft_Porch[7:0]	The width of Hsync front porch [7:0]
0x5F	Vid_VTotal_L	RO	7:0	Vid_VTotal[7:0]	The total line count of a field [7:0]
0x60	Vid_VTotal_H	RO	7:4	Vid_Vact[10:8]	The active line count of a field [10:8]
			3:0	Vid_Vtotal[10:8]	The total line count of a field [10:8]
0x61	Vid_Vact_L	RO	7:0	Vid_Vact[7:0]	The active line count of a field [7:0]
0x62	Vid_Vsync2DE	RO	7:0	Video sync to DE[7:0]	The width of vsync back porch
0x63	Vid_V_Ft_Porch	RO	7:0	Vid_V_Ft_Porch[7:0]	The width of vsync front porch
0x64	Vid_pixel_CNT	RO	7:0		Count of crystal clock on each 128 pixels
0x65	Vid_input_st	RO			Video input status
			7:4	Pix_rep	0000 : no repetition
					0001: pixel sent 2 times
					0011: pixel sent 4 times
			2	Avmute	HDMI is in Avmute state
			1	Vsync_in_po	Vsync input polarity
			0	Hsync_in_po	Hsync input polarity

When the video input is valid, the firmware can read for the video mode information.

Chap 4 Audio Output

HDMI audio exists in the packages of data island, appended in the blank of video data. Therefore, audio could only exist with video playback. If the video is not ready, there is no audio.

IT6604 support LCPM audio from 32KHz to 192KHz, 2 channels to 8 channels, compress audio with data rate up to 32bits * 192KHz, High bit rate audio such as TrueHD or DTS-HD audio, and one-bit audio (DSD).

Audio Control Registers

Following table lists the control registers of audio used on IT6604.

reg	Name	type	bit	bit Name	Description	Default
0x75	I2S_Ctrl	W/R	7	Ws_sel	1:invert channel A and channel B select	0
			6:2	I2S_Width	I2S word length, only effective in right justified mode; Maximum value is 24 means of 24 bits, value is the length of word length	11000
			1:0	**I2S_Mode**	**I2S output mode:** **00:32 bit I2S left justified mode, 1T delay** **01:32 bit I2S right justified mode** **10:32 bit I2S left justified mode, 0T delay** **11:undefined**	00
0x76	I2S_Map	W/R	7:6	I2S_Ch3Sel	I2S channel 6 and channel 7 map: 00: map to HDMI channel 0 and channel 1 01: map to HDMI channel 2 and channel 3 10: map to HDMI channel 4 and channel 5 11: map to HDMI channel 6 and channel 7	11
			5:4	I2S_Ch2Sel	I2S channel 4 and channel 5 map: 00: map to HDMI channel 0 and channel 1 01: map to HDMI channel 2 and channel 3 10: map to HDMI channel 4 and channel 5 11: map to HDMI channel 6 and channel 7	10
			3:2	I2S_Ch1Sel	I2S channel 2 and channel 3 map: 00: map to HDMI channel 0 and channel 1 01: map to HDMI channel 2 and channel 3 10: map to HDMI channel 4 and channel 5 11: map to HDMI channel 6 and channel 7	01
			1:0	I2S_Ch0Sel	I2S channel 0 and channel 1 map: 00: map to HDMI channel 0 and channel 1 01: map to HDMI channel 2 and channel 3 10: map to HDMI channel 4 and channel 5 11: map to HDMI channel 6 and channel 7	00
0x77	Audio_Ctrl	W/R	7	Audck_BBen	Enable NLPCM output from SPDIF	0x20
			6	Force_FS	Force Audio FS mode	
			5	Reserved		
			4	Aud_info_force	Force Audio setting from Aud info frame	
			3	Avmute_value	Avmute value when software Avmute is enabledSoftware forced-Avmute mode	
			2	Force_avmute	Software forced-Avmute mode	
			1			
			0			
0x78	MCLK_Ctrl	W/R	7	OSC_En	1:use crystal clock input as audio PLL reference 0:use pixel clock as audio PLL reference	1
			6	**OSCSel**	**1:selected when crystal is of 27MHz**	1
			4	Force_CTS	Software force to set CTS value	0
			3	Force_CTSMODE	use CTS for audio FIFO adjustment	0
			2:0	Mclksel	Mclk output clock multiple number 000:128FS ;001:256FS; 010:384FS ;011:512FS; 100:640FS ;101:768FS; 110:894FS ;111:1024FS;	001
0x7E	FS_SET	W/R	3:0	FS_SET	Software set sampling frequency 0000:44.1KHz; 0010:48KHz; 0011:32 KHz; 1000:88.2KHz; 1010:96 KHz; 1100:176.4KHz; 1110:192KHz	0010

0x7F	N_Rcv1	RO	7:0	N_DEC[7:0]	Audio N parameter decoder value [7:0]	
0x80	N_Rcv2	RO	7:0	N_DEC[15:8]	Audio N parameter decoder value [15:8]	
0x81	N_Rcv3	RO	7:4 3:0	CTS_DEC[3:0] N_DEC[19:16]	Audio CTS parameter decoder value [3:0] Audio N parameter decoder value [19:6]	
0x82	CTS_Rcv2	RO	7:0	CTS_DEC[11:4]	Audio CTS parameter decoder value [11:4]	
0x83	CTS_Rcv3	RO	7:0	CTS_DEC[19:12]	Audio CTS parameter decoder value [19:12]	
0x84	CD	RO	7:4	GCP_CD	Color depth decoder value in GCP 0: default (24bits) 4: 24bits 5: 30bits 6:36bits 7:48bits	
	FS		3:0	F_DEC	Audio FS(sample Freq.) decoder value 0000:44.1KHz; 0010:48KHz; 0011:32 KHz; 1000:88.2KHz; 1010:96 KHz; 1100:176.4KHz; 1110:192KHz	
0x87	HWMute_Ctrl	W/R	6 5 4 3	HWForceMute HWAudFFMuteClr HWMuteClr HWMuteEn	HWForceMute: 0: mute control by channel status 1: mute anyway FIFO mute clear Clear H/W mute H/W Mute enable	0001000
0x89	TriState_Ctrl	W/R	7 6 5 4 3 2 1 0	DisVAutomute Tri_VDIO Tri_VIO Tri_SPDIF Tri_I2S3 Tri_I2S2 Tri_I2S1 Tri_I2S0	Video output Auto Mute Disable tristate video Data output buffer tristate video Control output buffer tristate Audio SPDIF output buffer tristate Audio I2S3 output buffer tristate Audio I2S2 output buffer tristate Audio I2S1 output buffer tristate Audio I2S0 output buffer	0x80
0x8A	Audio_chanSt	RO	 7 6 5 4 3 2 1 0	 Audio_on HBRAudio DSDAudio Audio_layout Audio_src3_valid Audio_src2_valid Audio_src1_valid Audio_src0_valid	Audio received status Audio is ON Audio type is High Bit rate Audio type is DSD Audio layout is 1 or 0 HDMI Audio source 3 valid Flag HDMI Audio source 2 valid Flag HDMI Audio source 1 valid Flag HDMI Audio source 0 valid Flag	
0x8E	Pkt_type_H Ori_sampF	RO	3:0 7:4		Received packet type high bits Original sampling frequency	

HDMI Audio Input Status

IT6604 shows the input audio status in reg8A. When reg8A[7] = '1', it means IT6604 received audio sample packages, HBR audio packages, or DSD audio packages. reg8A[6] present if IT6604 got a HBR audio input. reg8A[5] represent if IT6604 got a DSD audio input.

	Audio Present reg8A[7]	HBR reg8A[6]	DSD reg8A[5]	NLPCM reg9C[1]
HD audio (HBR)	1	1	0	1
One-bit audio (DSD)	1	0	1	x
Compressed audio (NLPCM < 192KHz)	1	0	0	1
LPCM audio	1	0	0	0

The input audio channel status are responding in the reg84,reg8E,reg9C~reg9F, as following table:

reg	Name	type	bit	bit Name	Description	Default
0x84	FS		3:0	F_DEC	Audio FS(sample Freq.) decoder value 0000:44.1KHz; 0010:48KHz; 0011:32 KHz; 1000:88.2KHz; 1010:96 KHz; 1100:176.4KHz; 1110:192KHz	
0x8E	Ori_sampF	RO	7:4		Audio Channel Status[39:36] Original sampling frequency 1111:44.0KHz; 1101:48KHz; 1100:32 KHz; 0111:88.2KHz; 0101:96 KHz; 0011:176.4KHz; 0001:192KHz	
9C	Audio_Ch_status0	RO	7:6 5:3 2 1 0	Audio Channel status decoder value [7:0]	'00' – mode 0 for following byte. Other – reserved. D[1] = 0 '000' – 2 channel without pre-emphasis '001' – 2 channel with 50μs/15μs pre-emphasis. '010' – reserved '011' – reserved D[1] = 1 '000' – Default state for other application than linear PCM. 0 - Software for copyright asserted. 1 - Software for no copyright is asserted. 0 – Audio word world represents LPCM samples. 1 – Audio word used for other purpose reserved.	
9D	Audio_Ch_status1	RO		Category code.	Audio Channel status decoder value [15:8]	0x00
9E	Audio_Ch_status2	RO	7:4 3:0	Channel Number Source Number	Audio Channel status decoder value [23:16] "0000" – Do not take into account. "0001" – 1 '0000' – do not take into account '0001' – left channel for stereo format '0010 – right channel for stereo format.	
9F	Audio_Ch_status3	RO	7:4 3:2 1:0	Sample Word Length reserved Clock accurance	Audio Channel status decoder value [35:32] Audio Channel status decoder value [31:30] Audio Channel status decoder value [28:29]	

Software can judge the audio information by the register value.

Audio Output Configure

Default Setting

Reg75 – depends on the DSP configuration.

Reg76 – 0xE4.

Reg77[7] – '1' for accepting the NLPCM on I2S.

Reg78 = 0xC1

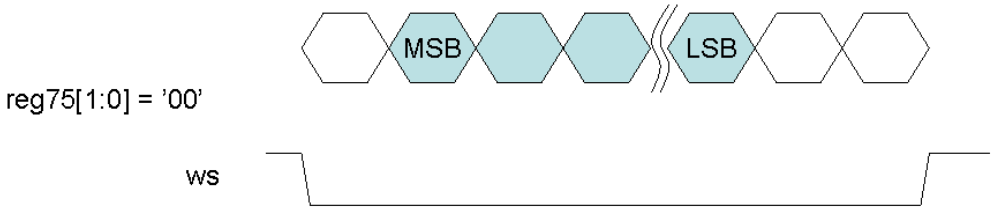
Reg7E[6:4] = '111'.

Reg88[4] = '0' for output HBR audio on I2S0~I2S3, '1' for output to SPDIF.

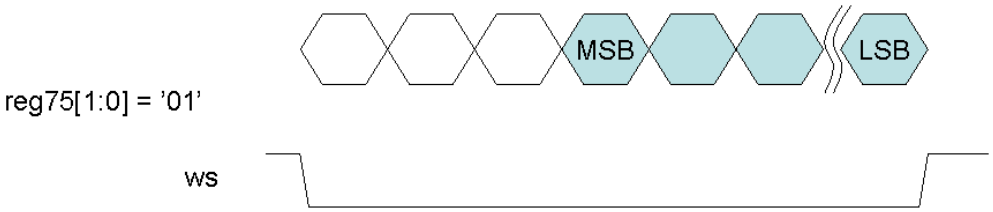
I2S mode and word length

If chose the I2S as output audio, IT6604 provide three types of I2S output format:

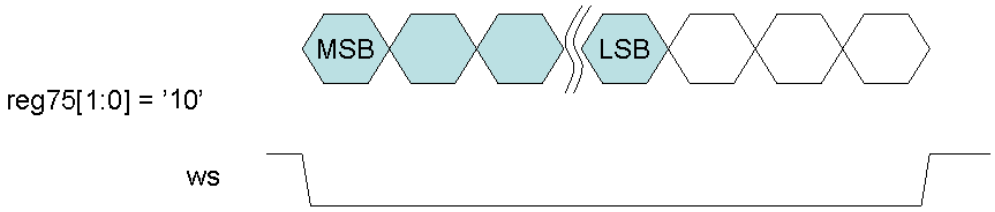
1. 32bit I2S output with 1T delay, MSB fist, Left Justify.



2. 32bit I2S output with MSB fist, Right Justify.



3. 32bit I2S output with 0T delay, MSB fist, Left Justify.



The word length of each sample are controlled by reg75[6:2] with actual value from 16 to 24.

And reg75[7] is control when WS is '0' which channel is carried.

Output LPCM Audio on I2S Channel

When detected the input audio is an LPCM audio, and wish output it on I2S channel, use the procedure:

1. Reg05[2] = '1' → '0'
2. Reg87[4][3] = '1' '1' → '0' '1'
3. reg89[3:0] = '0000' (depends on your I2S source number)

Output LPCM Audio on SPDIF Channel

1. Reg05[2] = '1' → '0'
2. Reg87[4][3] = '1' '1' → '0' '1'
3. reg89[4] = '0'

Output NLPCM Audio on I2S Channel

1. Reg77[7] = '1'
2. Reg05[2] = '1' → '0'

3. Reg87[4][3] = ‘1’ ‘1’ → ‘0’ ‘1’
4. reg89[0] = ’0’ (depends on your I2S source number)

Output NLPCM Audio on SPDIF Channel

1. Reg05[2] = ’1’ → ’0’
2. Reg87[4][3] = ‘1’ ‘1’ → ‘0’ ‘1’
3. reg89[4] = ’0’ (depends on your I2S source number)

Output High Bit Rate on I2S Channel

1. Reg88[4] = ’0’
2. Reg78[2:0] = ’000’
3. Reg05[2] = ’1’ → ’0’
4. Reg87[4][3] = ‘1’ ‘1’ → ‘0’ ‘1’
5. reg89[3:0] = ’0000’

Output High Bit Rate on SPDIF Channel

1. Reg88[4] = ’1’
2. Reg78[2:0] = ’000’
3. Reg05[2] = ’1’ → ’0’
4. Reg87[4][3] = ‘1’ ‘1’ → ‘0’ ‘1’
5. reg89[3:0] = ’0000’ (depends on your I2S source number)

Output LPCM/NLPCM Audio with Force Fs setting

For some audio input contains wrong information thus the IT6604 could not detect the correct sample frequency, you could use “force Fs Setting” to fix the audio problem.

1. Reg77[6] = ’1’
2. Reg87[4][3] = ‘1’ ‘1’ → ‘0’ ‘1’
3. Reg05[2] = ’1’ → ’0’
4. Reg7E[2:0] ← Given fixed frequency. (write at first time)
5. Reg7E[2:0] ← Given fixed frequency. (write at 2nd time)
6. Reg7E[2:0] ← Given fixed frequency. (write at 3rd time)
7. Reg7E[2:0] ← Given fixed frequency. (write at 4th time)
8. Clear the audio error interrupt status.

Error Handling

IT6604 provide two interrupt flag to indicate the audio error:

Reg	Reg_Name	W/R	Bits	Status	Description
0x15	Interrupt2	RO	4 3	AutoAudMute AudFIFOErr	Audio Auto Mute Flag Audio FIFO error Flag

When detect the interrupt by these two flag while the video on state, reconfigure the audio output procedure described above, then it could be clear.

Chap 5 HDMI Infoframe

IT6604 received the HDMI package in the specified registers. There are interrupt flags for the infoframe arrival, defined in reg14 interrupt status 1:

Reg	Reg_Name		Bits	Status	Description
0x14	Interrupt1	RO	7	PktLeftMute	Left Mute Packet is received
			6	NewAudioPkt_Det	New Audio Packet detect
			5	NewACPPkt_Det	New ACP Packet detect
			4	NewSPDPkt_Det	New SPD Packet detect
			3	NewMPEGPkt_Det	New MPEG Packet detect
			2	NewAVIPkt_Det	New AVI Packet detect
			1	NoAVI_Rcv	No AVI Packet is received
			0	PktSetMute	Set Mute Packet is received

When “new” infoframe or HDMI package arrived, the flags defined above will be raise. If there is no AVI infoframe arrived under HDMI mode, the NoAVI_Rcv flag will be raise. The interrupt mask 1 register (reg17) defined the mask with same field.

The getting infoframes are put in the following registers:

Reg Offset	Reg_Name	W/R	Bits	Status	Description	Default
0xA8	Pkt_rec_typeL	W/R	7:0		Decide which kind of packet to be fully recorded on General PKT registers.	0x83
0xA9	SRC_ver	RO			SPD infoFrame Version	
0xAA	SRC_pb25	RO			SPD infoFrame Data Byte 25	
0xAB	AVI_leng	RO			AVI infoFrame Length	
0xAC	AVI_VER	RO			AVI infoFrame Version	
0xAD	AVI_DB0	RO			AVI infoFrame Data Byte 0	
0xAE	AVI_DB1	RO			AVI infoFrame Data Byte 1	
0xAF	AVI_DB2	RO			AVI infoFrame Data Byte 2	
0xB0	AVI_DB3	RO			AVI infoFrame Data Byte 3	
0xB1	AVI_DB4	RO			AVI infoFrame Data Byte 4	
0xB2	AVI_DB5	RO			AVI infoFrame Data Byte 5	
0xB3	AVI_DB6	RO			AVI infoFrame Data Byte 6	
0xB4	AVI_DB7	RO			AVI infoFrame Data Byte 7	
0xB5	AVI_DB8	RO			AVI infoFrame Data Byte 8	
0xB6	AVI_DB9	RO			AVI infoFrame Data Byte 9	
0xB7	AVI_DB10	RO			AVI infoFrame Data Byte 10	
0xB8	AVI_DB11	RO			AVI infoFrame Data Byte 11	
0xB9	AVI_DB12	RO			AVI infoFrame Data Byte 12	
0xBA	AVI_DB13	RO			AVI infoFrame Data Byte 13	
0xBB	AVI_DB14	RO			AVI infoFrame Data Byte 14	
0xBC	AVI_DB15	RO			AVI infoFrame Data Byte 15	
0xDC	Audio_Ver	RO			Audio infoFrame Version	
0xDD	Audio_DB0	RO			Audio infoFrame Data Byte 0	
0xDE	Audio_DB1	RO			Audio infoFrame Data Byte 1	
0xDF	Audio_DB2	RO			Audio infoFrame Data Byte 2	
0xE0	Audio_DB3	RO			Audio infoFrame Data Byte 3	
0xE1	Audio_DB4	RO			Audio infoFrame Data Byte 4	
0xE2	Audio_DB5	RO			Audio infoFrame Data Byte 5	
0xE3	Audio_leng	RO			Audio infoFrame Length	
0xE4	MPEG_Ver	RO			MPEG infoFrame Version	
0xE5	MPEG_leng	RO			MPEG infoFrame Length	
0xE6	MPEG_DB0	RO			MPEG infoFrame Data Byte 0	
0xE7	MPEG_DB1	RO			MPEG infoFrame Data Byte 1	
0xE8	MPEG_DB2	RO			MPEG infoFrame Data Byte 2	
0xE9	MPEG_DB3	RO			MPEG infoFrame Data Byte 3	
0xEA	MPEG_DB4	RO			MPEG infoFrame Data Byte 4	
0xEB	MPEG_DB5	RO			MPEG infoFrame Data Byte 5	
0xEC	ACP_HB0	RO			ACP packet Header Byte 0	
0xED	ACP_HB1	RO			ACP packet Header Byte 1	
0xEE	ACP_HB2	RO			ACP packet Header Byte 2	
0xEF	ACP_DB0	RO			ACP packet Data Byte 0	
0xF0	ACP_DB1	RO			ACP packet Data Byte 1	
0xF1	ACP_DB2	RO			ACP packet Data Byte 2	

0xF2	ACP_DB3	RO			ACP packet Data Byte 3	
0xF3	ACP_DB4	RO			ACP packet Data Byte 4	
0xF4	ACP_DB5	RO			ACP packet Data Byte 5	
0xF5	ACP_DB6	RO			ACP packet Data Byte 6	
0xF6	ACP_DB7	RO			ACP packet Data Byte 7	
0xF7	ACP_DB8	RO			ACP packet Data Byte 8	
0xF8	ACP_DB9	RO			ACP packet Data Byte 9	
0xF9	ACP_DB10	RO			ACP packet Data Byte 10	
0xFA	ACP_DB11	RO			ACP packet Data Byte 11	
0xFB	ACP_DB12	RO			ACP packet Data Byte 12	
0xFC	ACP_DB13	RO			ACP packet Data Byte 13	
0xFD	ACP_DB14	RO			ACP packet Data Byte 14	
0xFE	ACP_DB15	RO			ACP packet Data Byte 15	
0xFF	ACP_rec_type	W/R	7:0		The packet type will be stored to ACP register	0x04

If you want the gotten infoframes, can return the value in above registers.

Chap 6 HDCP Support

IT6604 support HDCP sink with repeater type capability.

When HDCP authentication started with An, AKSV written into IT6604, the authentication start interrupt will be issued, and the authentication done interrupt will be issued when the M0' calculated out and R0' was read back.

Reg	Reg_Name	W/R	Bits	Status	Description
0x15	Interrupt2	RO	2	ECCERR	EDD error Flag
			1	Auth_done	Authentication Done Flag
			0	Auth_start	Authentication Start Flag

HDCP Repeater setting

On initial time

1. reg73[7] = '1' for claiming HDCP repeater mode. (reg73[7] = '1' → BCap[6] = '1')
2. reg73[6] = '0' to clear the HDCP KSV FIFO ready (BCaps[5] = '0') .

0x40	*Bcaps*	1	Rd	Bit 7: HDMI_RESERVED Use of this bit is reserved. HDCP Receivers not capable of supporting HDMI must clear this bit to 0. Bit 6: REPEATER, HDCP Repeater capability. When set to one, this HDCP Receiver supports downstream connections as permitted by the Digital Content Protection LLC license. This bit does not change while the HDCP Receiver is active. Bit 5: READY, KSV FIFO ready. When set to one, this HDCP Repeater has built the list of attached KSVs and computed the verification value *V'*. This value is always zero during the computation of *V'*. Bit 4: FAST. When set to one, this device supports 400 KHz transfers. When zero, 100 KHz is the maximu m transfer rate supported. Note that 400KHz transfers are not permitted to any device unless all devices on the I^2C bus are capable of 400KHz transfer. The transmitter may not be able to determine if the EDID ROM, present on the HDCP Receiver, is capable of 400KHz operation. This bit does not change while the HDCP Receiver is active. Bits 3-2: Reserved (must be zero). Bit 1: 1.1_FEATURES. When set to one, this HDCP Receiver supports Enhanced Encryption Status Signaling (EESS), Advance Cipher, and Enhanced Link Verification options. For the HDMI protocol, Enhanced Encryption Status Signaling (EESS) capability is assumed regardless of this bit setting. This bit does not change while the HDCP Receiver is active. Bit 0: FAST_REAUTHENTICATION. When set to 1, the receiver is capable of receiving (unencrypted) video signal during the session re-authentication. All HDMI-capable receivers shall be capable of performing the fast re-authentication even if this bit is not set. This bit does not change while the HDCP Receiver is active.

table: BCaps

3. Reg73[5:4] = '00'

When executing situation:

4. When the authentication start interrupt detected, clear the KSV FIFO Ready bit (reg73[6] = '0').
5. Software should pass the HDCP authentication start message to next HDCP stage, issue a new HDCP start to its HDCP sink.

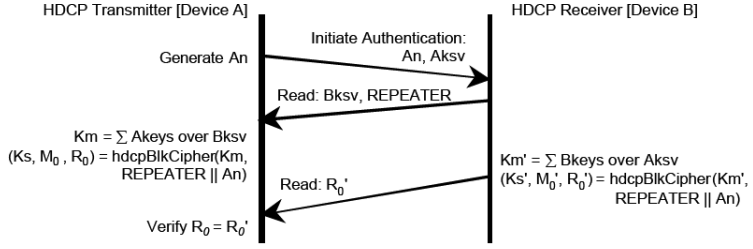


Figure 2-1. First Part of Authentication Protocol

6. Wait for HDCP authentication done interrupt, it means the M0' and R0' is ready, the first part is done.

7. When the authentication to the sinks of this repeater are done, collects the KSV Lists (include the BKSVs of those direct sinks), and update them to the KSV List registers.
IT6604 supports maximum 8 sets of KSV, the KSV set 0 ~ set 3 locate in reg1C1~reg1D4 (under bank1, where reg0F ← 0x01), and the set 4 ~ set 7 locate in reg181~reg194.

8. Calculated the V' with KSV0~KSVn, BStatus (include depth and down streams number, IT6604 put them in reg1D5 and 1D6), and M0' (reg1EB~reg1F2) with SHA-1 algorithm, then put the V' value to register reg1D7~reg1EA.
The buffer order to calculation of SHA-1 can refer the following sample:

Ksv0	0x35796a172e
Ksv1	0x478e71e20f
Ksv2	0x74e85397a6
Bstatus	0x0203
M0	0x372d3dce38bbe78f
SHA-1 transform input	2e 17 6a 79 35 0f e2 71 8e 47 a6 97 53 e8 74 03 02 8f e7 bb 38 ce 3d 2d 37 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c8
SHA-1 H0b	0x0fcbd586
SHA-1 H1	0xefc107ef
SHA-1 H2	0xccd70a1d
SHA-1 H3	0xb1186dda
SHA-1 H4	0x1fb3ff5e
KSV FIFO (port 43)	2e 17 6a 79 35 0f e2 71 8e 47 a6 97 53 e8 74
Ports 41-42	03 02
Ports 20-23	86 d5 cb 0f
Ports 24-27	ef 07 c1 ef
Ports 28-2b	1d 0a d7 cc
Ports 2c-2f	da 6d 18 b1
Ports 30-33	5e ff b3 1f

9. After the BStatus, KSV FIFO, and V' are written ready, write the KSV ready bit in reg73[6] = '1', then the HDCP source will detect the second part of authentication is done by the repeater, and try to complete the HDCP third part.

Note: To avoid the auto AV mute filtering out the video output from IT6604, when the input video is ready with AVMute status, the reg89[7:5] should be '100' to forward the VCLK and sync to next HDCP stage, then the HDCP could be completed.

HDCP registers for repeater function

Block2 (reg0F[0] = '1')

Reg Offset	Reg_Name	type	Bits	Default	Description
0x180	KSV_FIFO40	W/R		0x00	HDCP KSVFIFO4 [7:0]

0x181	KSV_FIFO41	W/R		0x00	HDCP KSVFIFO4 [15:8]
0x182	KSV_FIFO42	W/R		0x00	HDCP KSVFIFO4 [23:16]
0x183	KSV_FIFO43	W/R		0x00	HDCP KSVFIFO4 [31:24]
0x184	KSV_FIFO44	W/R		0x00	HDCP KSVFIFO4 [39:32]
0x185	KSV_FIFO50	W/R		0x00	HDCP KSVFIFO5 [7:0]
0x186	KSV_FIFO51	W/R		0x00	HDCP KSVFIFO5 [15:8]
0x187	KSV_FIFO52	W/R		0x00	HDCP KSVFIFO5 [23:16]
0x188	KSV_FIFO53	W/R		0x00	HDCP KSVFIFO5 [31:24]
0x189	KSV_FIFO54	W/R		0x00	HDCP KSVFIFO5 [39:32]
0x18A	KSV_FIFO60	W/R		0x00	HDCP KSVFIFO6 [7:0]
0x18B	KSV_FIFO61	W/R		0x00	HDCP KSVFIFO6 [15:8]
0x18C	KSV_FIFO62	W/R		0x00	HDCP KSVFIFO6 [23:16]
0x18D	KSV_FIFO63	W/R		0x00	HDCP KSVFIFO6 [31:24]
0x18E	KSV_FIFO64	W/R		0x00	HDCP KSVFIFO6 [39:32]
0x18F	KSV_FIFO70	W/R		0x00	HDCP KSVFIFO7 [7:0]
0x190	KSV_FIFO71	W/R		0x00	HDCP KSVFIFO7 [15:8]
0x191	KSV_FIFO72	W/R		0x00	HDCP KSVFIFO7 [23:16]
0x192	KSV_FIFO73	W/R		0x00	HDCP KSVFIFO7 [31:24]
0x193	KSV_FIFO74	W/R		0x00	HDCP KSVFIFO7 [39:32]
0x1C1	KSV_FIFO00	W/R		0x00	HDCP KSVFIFO0 [7:0]
0x1C2	KSV_FIFO01	W/R		0x00	HDCP KSVFIFO0 [15:8]
0x1C3	KSV_FIFO02	W/R		0x00	HDCP KSVFIFO0 [23:16]
0x1C4	KSV_FIFO03	W/R		0x00	HDCP KSVFIFO0 [31:24]
0x1C5	KSV_FIFO04	W/R		0x00	HDCP KSVFIFO0 [39:32]
0x1C6	KSV_FIFO10	W/R		0x00	HDCP KSVFIFO1 [7:0]
0x1C7	KSV_FIFO11	W/R		0x00	HDCP KSVFIFO1 [15:8]
0x1C8	KSV_FIFO12	W/R		0x00	HDCP KSVFIFO1 [23:16]
0x1C9	KSV_FIFO13	W/R		0x00	HDCP KSVFIFO1 [31:24]
0x1CA	KSV_FIFO14	W/R		0x00	HDCP KSVFIFO1 [39:32]
0x1CB	KSV_FIFO20	W/R		0x00	HDCP KSVFIFO2 [7:0]
0x1CC	KSV_FIFO21	W/R		0x00	HDCP KSVFIFO2 [15:8]
0x1CD	KSV_FIFO22	W/R		0x00	HDCP KSVFIFO2 [23:16]
0x1CE	KSV_FIFO23	W/R		0x00	HDCP KSVFIFO2 [31:24]
0x1CF	KSV_FIFO24	W/R		0x00	HDCP KSVFIFO2 [39:32]
0x1D0	KSV_FIFO30	W/R		0x00	HDCP KSVFIFO3 [7:0]
0x1D1	KSV_FIFO31	W/R		0x00	HDCP KSVFIFO3 [15:8]
0x1D2	KSV_FIFO32	W/R		0x00	HDCP KSVFIFO3 [23:16]
0x1D3	KSV_FIFO33	W/R		0x00	HDCP KSVFIFO3 [31:24]
0x1D4	KSV_FIFO34	W/R		0x00	HDCP KSVFIFO3 [39:32]
0x1D5	BstatusL	W/R	7:0	0x00	HDCP BStatus[7:0]
0x1D6	BstatusH	W/R	3:0	0x0	HDCP Bstatus[11:8]
0x1D7	SHA1_H00	W/R		0x00	HDCP SHA1_H0 [7:0]
0x1D8	SHA1_H01	W/R		0x00	HDCP SHA1_H0 [15:8]
0x1D9	SHA1_H02	W/R		0x00	HDCP SHA1_H0 [23:16]
0x1DA	SHA1_H03	W/R		0x00	HDCP SHA1_H0 [31:24]
0x1DB	SHA1_H10	W/R		0x00	HDCP SHA1_H1 [7:0]
0x1DC	SHA1_H11	W/R		0x00	HDCP SHA1_H1 [15:8]
0x1DD	SHA1_H12	W/R		0x00	HDCP SHA1_H1 [23:16]
0x1DE	SHA1_H13	W/R		0x00	HDCP SHA1_H1 [31:24]
0x1DF	SHA1_H20	W/R		0x00	HDCP SHA1_H2 [7:0]
0x1E0	SHA1_H21	W/R		0x00	HDCP SHA1_H2 [15:8]
0x1E1	SHA1_H22	W/R		0x00	HDCP SHA1_H2 [23:16]
0x1E2	SHA1_H23	W/R		0x00	HDCP SHA1_H2 [31:24]
0x1E3	SHA1_H30	W/R		0x00	HDCP SHA1_H3 [7:0]
0x1E4	SHA1_H31	W/R		0x00	HDCP SHA1_H3 [15:8]
0x1E5	SHA1_H32	W/R		0x00	HDCP SHA1_H3 [23:16]
0x1E6	SHA1_H33	W/R		0x00	HDCP SHA1_H3 [31:24]
0x1E7	SHA1_H40	W/R		0x00	HDCP SHA1_H4 [7:0]
0x1E8	SHA1_H41	W/R		0x00	HDCP SHA1_H4 [15:8]
0x1E9	SHA1_H42	W/R		0x00	HDCP SHA1_H4 [23:16]
0x1EA	SHA1_H43	W/R		0x00	HDCP SHA1_H4 [31:24]
0x1EB	M0_B0	RO			HDCP Mi value readback [7:0]
0x1EC	M0_B1	RO			HDCP Mi value readback [15:8]
0x1ED	M0_B2	RO			HDCP Mi value readback [23:16]
0x1EE	M0_B3	RO			HDCP Mi value readback [31:24]
0x1EF	M0_B4	RO			HDCP Mi value readback [39:32]
0x1F0	M0_B5	RO			HDCP Mi value readback [47:40]
0x1F1	M0_B6	RO			HDCP Mi value readback [55:48]

0x1F2	M0_B7	RO			HDCP Mi value readback [63:56]

Chap 7 3D Support

HDMI1.4 defined the 3D format and 3D infoframe in Vendor-Specific InfoFrame Packet.

The format are as following:

Table 8-10 HDMI Vendor Specific InfoFrame Packet Header

Byte ¥ Bit #	7	6	5	4	3	2	1	0
HB0	Packet Type = 0x81							
HB1	Version = 0x01							
HB2	0	0	0	Length = Nv				

Table 8-11 HDMI Vendor Specific InfoFrame Packet Contents

Packet Byte #	7	6	5	4	3	2	1	0
PB0	Checksum							
PB1	24bit IEEE Registration Identifier (0x000C03) (least significant byte first)							
PB2								
PB3								
PB4	HDMI_Video_Format			Rsvd (0)	Rsvd (0)	Rsvd (0)	Rsvd (0)	Rsvd (0)
(PB5)	HDMI_VIC							
	3D_Structure				Reserved(0)			
(PB6)	3D_Ext_Data				Reserved(0)			
.. PB(Nv)	Reserved (0)							

- Length [5bits] This 5 bits field defines the length of HDMI vendor specific InfoFrame payload.
- HDMI_Video_Format [3bits] This value defines the structure of extended video formats exclusively defined within this HDMI specification.

Table 8-12 HDMI_Video_Format

Value [2…0]	description
000	No additional HDMI video format is presented in this packet.
001	Extended resolution format (e.g. used for 4K x 2K video) present. 1 byte of HDMI_VIC parameter value follows.
010	3D format indication present. 3D_Structure, and potentially 3D_Ext_Data, follows.
011 ~ 111	Reserved for future use

- HDMI_VIC [1byte] HDMI proprietary Video Format Identification Code. When transmitting any video format defined in this section 8.2.3.1, an HDMI Source shall set the HDMI_VIC field to the Video Code for that format. The HDMI_VIC value is defined in section 8.2.3.1.
- 3D_Structure [4bits] This 4 bit field defines the transmission format of 3D video data. The value of "0000" means the Frame packing structure described in this section. The value of "1000" means the Side-by-Side (Half) structure described in this section. The value of "0110" means the Top-and-Bottom[1] structure described in this section. For other values, see Appendix H (Table H-2).

Table 8-13 3D_Structure

Value	Meaning
0000	Frame packing
0001 ~ 0101	Reserved for future use
0110	Top-and-Bottom
0111	Reserved for future use.
1000	Side-by-Side (Half)
1001 ~ 1111	Reserved for future use

For Side-by-Side (Half), the original left and right pictures are sub-sampled to half resolution on the horizontal axis. Sub-sampled pictures are arranged in Side-by-Side layout. See Figure 8-5.

For Top-and-Bottom, the original full left and right pictures are sub-sampled to half resolution on the vertical axis. Sub-sampled pictures are arranged in Top-and-Bottom layout. See Figure 8-6.

- 3D_Ext_Data [4bits] The meaning of this field depends on the 3D_Structure value.
  If 3D_Structure is 1000 (Side-by-Side (Half)), the 3D_Ext_Data field is added in the HDMI Vendor Specific InfoFrame and shall be 00XX (i.e. from 0000 to 0011, horizontal sub-sampling). For other values, see Appendix H.
  If 3D_Structure is 1001~1111, the 3D_Ext_Data field is also added and indicates additional information about the 3D format.
  If 3D_Structure is 0000~0111, the 3D_Ext_Data field shall not be present.

In IT6604 use general packet registers to receive the common type of HDMI packet.

To set regA8 as 0x81, the VSDB packet will be received in the following registers:

Address	Reg_Name	W/R	Description
0xBD	GENPKT_HB0	RO	This is a general-purpose register recording one kind of the packets.

			Genaral Packet Header Byte 0
0xBE	GENPKT_HB1	RO	Genaral Packet Header Byte 1
0xBF	GENPKT_HB2	RO	Genaral Packet Header Byte 2
0xC0	GENPKT_DB0	RO	Genaral Packet Data Byte 0
0xC1	GENPKT_DB1	RO	Genaral Packet Data Byte 1
0xC2	GENPKT_DB2	RO	Genaral Packet Data Byte 2
0xC3	GENPKT_DB3	RO	Genaral Packet Data Byte 3
0xC4	GENPKT_DB4	RO	Genaral Packet Data Byte 4
0xC5	GENPKT_DB5	RO	Genaral Packet Data Byte 5
0xC6	GENPKT_DB6	RO	Genaral Packet Data Byte 6
0xC7	GENPKT_DB7	RO	Genaral Packet Data Byte 7
0xC8	GENPKT_DB8	RO	Genaral Packet Data Byte 8
0xC9	GENPKT_DB9	RO	Genaral Packet Data Byte 9
0xCA	GENPKT_DB10	RO	Genaral Packet Data Byte 10
0xCB	GENPKT_DB11	RO	Genaral Packet Data Byte 11
0xCC	GENPKT_DB12	RO	Genaral Packet Data Byte 12
0xCD	GENPKT_DB13	RO	Genaral Packet Data Byte 13
0xCE	GENPKT_DB14	RO	Genaral Packet Data Byte 14
0xCF	GENPKT_DB15	RO	Genaral Packet Data Byte 15
0xD0	GENPKT_DB16	RO	Genaral Packet Data Byte 16
0xD1	GENPKT_DB17	RO	Genaral Packet Data Byte 17
0xD2	GENPKT_DB18	RO	Genaral Packet Data Byte 18
0xD3	GENPKT_DB19	RO	Genaral Packet Data Byte 19
0xD4	GENPKT_DB20	RO	Genaral Packet Data Byte 20
0xD5	GENPKT_DB21	RO	Genaral Packet Data Byte 21
0xD6	GENPKT_DB22	RO	Genaral Packet Data Byte 22
0xD7	GENPKT_DB23	RO	Genaral Packet Data Byte 23
0xD8	GENPKT_DB24	RO	Genaral Packet Data Byte 24
0xD9	GENPKT_DB25	RO	Genaral Packet Data Byte 25
0xDA	GENPKT_DB26	RO	Genaral Packet Data Byte 26
0xDB	GENPKT_DB27	RO	Genaral Packet Data Byte 27

If the regBD value is 0x81, the VSDB is received then check the input 3D type. That is the job of the scalar on the next stage of IT6604.

Part 2 – Software Release Code Reference

Chap 8 Introduce

ITE release a sample package as following file list, with a 8051 project for customer developing. The file list are as following:

File name	Description
src\mainmcu.c	
src\hdmirx.c	major code.
src\hdmirx.h	include file for exporting function.
src\typedef.h	The common type definition file.
src\IO.c	The sample code for providing service as I^2C calling function.
src\IO.h	header file for io.c
src\MCU.H	The definition for demo board pin definition.
src\515XRAM.H	8051 register definition header file.
src\reg_c51.h	8051 regsiter defintioin head file.
src\debug.h	debug macro definition.
src\urt.c	debug procedure.
src\urt.h	
src\VERSION.H	The version string definition code to highlight the using version.

For those system with their own processor and procedure, you can refer the main function in mcumain.c to migrate the code into your system.

Chap 9 Flow

The sample main loop as following block:

```
int main( void )
{
   static BOOL bSignal ;
   BOOL bOldSignal, bChangeMode ;

   InitCAT6023() ;
   while(1) {
      if(ReadRXIntPin())
      {
         Check_HDMInterrupt() ;
      }
      MCUTimerServiceISR();
      if( TimerServF )
      {
         TimerServF = FALSE ;
         bOldSignal = bSignal ;
         bSignal = CheckHDMIRX() ;
         bChangeMode = ( bSignal != bOldSignal ) ;
      }
      if( bChangeMode )
      {
         // if Changed Mode ...
         if( bSignal )
         {
            // if signal is TRUE , then ...
            printf("getCAT6023HorzTotal() = %d\n",getCAT6023HorzTotal()) ;
            printf("getCAT6023HorzActive() = %d\n",getCAT6023HorzActive()) ;
            printf("getCAT6023HorzFrontPorch() = %d\n",getCAT6023HorzFrontPorch()) ;
            printf("getCAT6023HorzSyncWidth() = %d\n",getCAT6023HorzSyncWidth()) ;
            printf("getCAT6023HorzBackPorch() = %d\n",getCAT6023HorzBackPorch()) ;
            printf("getCAT6023VertTotal() = %d\n",getCAT6023VertTotal()) ;
            printf("getCAT6023VertActive() = %d\n",getCAT6023VertActive()) ;
            printf("getCAT6023VertFrontPorch() = %d\n",getCAT6023VertFrontPorch()) ;
            printf("getCAT6023VertSyncToDE() = %d\n",getCAT6023VertSyncToDE()) ;
            printf("getCAT6023VertSyncBackPorch() = %d\n",getCAT6023VertSyncBackPorch()) ;
            printf("getCAT6023VertSyncWidth() = %d\n",getCAT6023VertSyncWidth()) ;
         }
         else
         {
            // if signal is FALSE , then ...
         }
         bChangeMode = FALSE ; // clear bChange Mode action
      }
      else
      {
         // if not change mode, ...
         if( bSignal )
         {
            // if signal is TRUE , then ...
         }
         else
         {
            // if signal is FALSE , then ...
         }
      }
```

```
    }
    return 0;
}
```

The system flow should call the InitCAT6023() in the initial time, and call CheckHDMIRX() in the main loop. As the following figure:

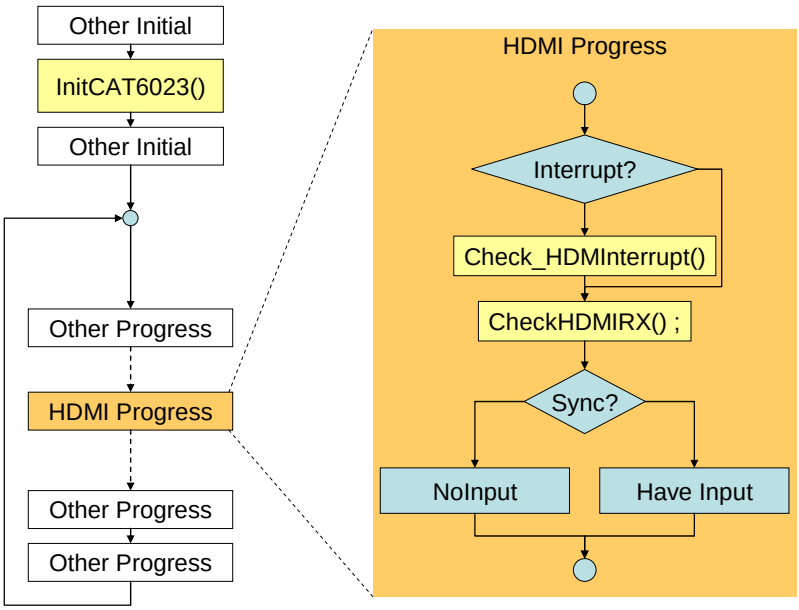


Fig. HDMI System Flow

Chap 10 Data Type

Following describe the data type defined in the source code.

```
#define FALSE 0
#define TRUE 1
#define SUCCESS 0
#define FAIL -1
#define ON 1
#define OFF 0
typedef enum _SYS_STATUS {
    ER_SUCCESS = 0,
    ER_FAIL,
    ER_RESERVED
} SYS_STATUS ;
```

Chap 11 Sample Code Required Interface

This chapter describes those interfaces the porting system should provide. The following function need the system provide to CAT6023.C for referring registers in CAT6023.

BYTE HDMIRX_ReadI2C_Byte(BYTE RegAddr);

Parameter:`	Provide a register sub-address of CAT6023 I^2C access.
Return:	A byte read back from CAT6023 I^2C address subaddress RegAddr.
Remark:	This function should be provided by main system to implement a I^2C access of CAT6023 registers.

SYS_STATUS HDMIRX_WriteI2C_Byte(BYTE RegAddr,BYTE d);

Parameter:`	RegAddr – a subaddress of a register to write d – a byte value to write the address RegAddr.
Return:	ER_SUCCESS – write success. ER_Fail – write fail.
Remark:	This function should be provided by main system to implement a I^2C access of CAT6023 registers.

SYS_STATUS HDMIRX_ReadI2C_ByteN(BYTE RegAddr,BYTE *pData,int N);

Parameter:`	RegAddr – a subaddress of a register to read pData – a byte pointer to a buffer for return the value to read. N – the number to read.
Return:	ER_SUCCESS –read success. ER_Fail –read fail.
Remark:	This function should be provide to read N bytes value from sub address RegAddr.

SYS_STATUS HDMIRX_WriteI2C_ByteN(BYTE RegAddr,BYTE *pData,int N);

Parameter:`	RegAddr – a subaddress of a register to write pData – a byte pointer to a buffer for return the value to write. N – the number to write.
Return:	ER_SUCCESS – write success. ER_Fail – write fail.
Remark:	This function should be provide to write N bytes value into sub address RegAddr.

Chap 12 Software Interface

The following function is the software interface provided in HDMIRX.C and simplified description.

void InitHDMIRX()

Parameter: N/A
Return Value: N/A
Remark: Call this function on the first initial time or after any hardware reset behavior of IT6604. After select HDMI port, we suggest to call this function for confirm all the behavior normal.

void Check_HDMIInterrupt()

Parameter: N/A
Return Value: N/A
Remark: The function to check the HDMI interrupt status. If you implement the interrupt pin, when interrupt activated, this function should be called.
IT6604 always response interrupt status whatever the mask set or not, the interrupt mask just activate the interrupt signal. For a polling only system this function was be called in CheckHDMIRX().

BOOL CheckHDMIRX()

Parameter: N/A
Return Value: FALSE: The video signal does not output.
TRUE: The video signal output from HDMIRX, the next stage TTL processor can handle the signal and ready to prepare audio output.
Remark: This function have to put in major loop for monitoring the status and executing the final status machine inner of HDMIRX driver code. The reset behavior are all implemented in the driver code and depends on the running of final state machine executing.

void SelectHDMIPort(BYTE ucPort)

Parameter: ucPort: 0 – Port A/1 – Port B
Return Value: N/A
Remark: This function is for the system to implement multiple port input. It should be called out of the main loop, and decide by manual selection.

BYTE GetCurrentHDMIPort()

Parameter: N/A
Return Value: 0 – Port A/1 – Port B
Remark: This function is used to get current input port.

void PowerDownHDMI()

Parameter: N/A
Return Value: N/A
Remark: If the system switch the video input to other type, call this function to power down IT6604 can saving the power consumption. After call this function, the next HDMI loop should start from InitHDMIRX()./

BOOL IsHDMIRXHDMIMode()

Parameter: N/A
Return Value: 0 – Input is under DVI mode/1 – Input is under HDMI mode
Remark: none.

BOOL IsHDMIRXInterlace()

Parameter: N/A
Return Value: FALSE – the mode is a progress mode
TRUE – the mode is a interlaced mode.

WORD getHDMIRXHorzTotal()

Parameter: N/A
Return Value: Return the horizontal total value in pixel after divided pixel repetition.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXHorzActive()

Parameter: N/A
Return Value: Return the horizontal active value in pixel after divided pixel repetition.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXHorzFrontPorch()

Parameter: N/A
Return Value: Return the horizontal front porch value in pixel after divided pixel repetition.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXHorzSyncWidth()

Parameter: N/A
Return Value: Return the horizontal front porch value in pixel after divided pixel repetition.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXHorzBackPorch()

Parameter: N/A
Return Value: Return the horizontal front porch value in pixel after divided pixel repetition.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXVertTotal()

Parameter: N/A
Return Value: Return the vertical total value in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXVertActive()

Parameter: N/A
Return Value: Return the vertical active value in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXVertFrontPorch()

Parameter: N/A
Return Value: Return the vertical front porch value in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when

the video input is stable.

WORD getHDMIRXVertSyncToDE()

Parameter: N/A
Return Value: Return the value from vertical sync start to data enable start in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXVertSyncBackPorch()

Parameter: N/A
Return Value: Return the vertical back porch value in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

WORD getHDMIRXVertSyncWidth()

Parameter: N/A
Return Value: Return the vertical sync width value in scan line, and only represent the value for one field if under interlaced mode.
Remark: The value is for main system judge the input mode. The value is only valid when the video input is stable.

BYTE getHDMIRXxCnt()

Parameter: N/A
Return Value: The clock count to judge video clock.
Remark: IT6604 implement a counter to measure the input clock. The count return the 27MHz crystal reference clock count number under 128 pixel clock of input video, therefore, if the clock number is N, the PCLK should be 27MHz * 128/N.

BYTE getHDMIRXOutputColorMode()

Parameter: N/A
Return Value: 0 – RGB444 color mode
1 – YCbCr422 color mode
2 – YCbCr444 color mode
Remark: The function is just implemented for reference to get the output color mode. Most case the output format was fixed under project negeotiate.

BYTE getHDMIRXOutputColorDepth()

Parameter: N/A
Return Value: 0 – default (24 bit)
4 – 24bit
5 – 30bit
6 – 36bit
Remark: The function return the color depth defined in HDMI 1.3.

BOOL getHDMIRXAudioInfo(BYTE *pbAudioSampleFreq, BYTE *pbValidCh)

Parameter: pbAudioSampleFreq – the pointer to a byte value for return audio sample frequence.
pbValidCh – the pointer to a byte value for return valid channel number.
Return Value: TRUE – success/FALSE – otherwise.
Remark: AudioSampleFreq – the encoded value as following table

Value	Description
0	44.1KHz
2	48KHz

3	32KHz
8	88.2KHz
9	768KHz (for HBR)
0xA	96KHz
0xC	176.4KHz
0xE	192KHz

ValidCh – four channels indication represented in bit field
bit[0] : 1 – channel 0 valid / 0 – otherwise
bit[1] : 1 – channel 1 valid / 0 – otherwise
bit[2] : 1 – channel 2 valid / 0 – otherwise
bit[3] : 1 – channel 3 valid / 0 – otherwise

BYTE getHDMIRXAudioStatus()

Parameter: N/A
Return Value: return a byte with audio status encoding.
Remark: return byte with the bit field return value:

Bit	Value meaning
7	1: audio on 0: otherwise
6	1: input audio is an HBR audio 0: otherwise
5	1: input audio is an DSD (one-bit) audio 0: otherwise
4	1: input audio is an non-LPCM audio 0: otherwise
3	1: the input audio use layout 1 audio sample packet. 0: otherwise
0	1: the input audio use layout 0 audio sample packet. 0: otherwise

BOOL getHDMIRXAudioChannelStatus(BYTE ucIEC60958ChStat[])

Parameter: A five bytes array to store a IEC60958 audio channel status.
Return Value: TRUE – success/FALSE - other
Remark: Please refer *IEC60958-3* document to check the five bytes definition.

void setHDMIRX_HBROutput(BOOL HBR_SPDIF)

Parameter: TRUE – set HBR audio output from SPDIF channel
FALSE – set HBR audio output from I2S four channel.
Return Value: N/A
Remark: High Bit Rate (HBR) audio can output from the SPDIF or I2S channel. The setting of these two paths are different, need to control by manual.

void setHDMIRX_SPDIFOutput()

Parameter: N/A
Return Value: N/A
Remark: Set common audio output from SPDIF. I2S output will be turned off.

void setHDMIRX_I2SOutput(BYTE src_enable)

Parameter: src_enable[3:0] to decide which channel should be enabled.
Return Value: N/A
Remark: The function configures the output from I2S channel, and SPDIF channel will be tristated.

BOOL GetAVIInfoFrame(BYTE *pData)

Parameter: pData – a byte array to get AVI Infoframe.
Return Value: Return TRUE if success, otherwise FALSE.
Remark: Software can use this function to get the input AVI Infoframe.

BOOL GetAudioInfoFrame(BYTE *pData)

Parameter: pData – a byte array to get Audio Infoframe.
Return Value: Return TRUE if success, otherwise FALSE.

Remark: Software can use this function to get the input Audio InfoframBOOL GetACPPacket(BYTE *pData)

Parameter: pData – a byte array to get ACP Packet.
Return Value: Return TRUE if success, otherwise FALSE.
Remark: Software can use this function to get the input ACP Packet.

BOOL GetVENDORSPECInfoFrame(BYTE *pData)

Parameter: pData – a byte array to get Vendor Specific Infoframe.
Return Value: Return TRUE if success, otherwise FALSE.
Remark: Software can use this function to get the input Vendor Specific Infoframe.

BOOL GetMPEGInfoFrame(BYTE *pData)

Parameter: pData – a byte array to get MPEG Infoframe.
Return Value: Return TRUE if success, otherwise FALSE.
Remark: Software can use this function to get the input MPEG Infoframe.