make sure to look at the errata. many errors.   the PCM clock control and div are missing!

6.3 General Purpose GPIO Clocks

The General Purpose clocks can be output to GPIO pins. They run from the peripherals clock sources and use clock generators with noise-shaping MASH dividers. These allow the GPIO clocks to be used to drive audio devices.

The fractional divider operates by periodically dropping source clock pulses, therefore the output frequency will periodically switch between:

$$\frac{source\_frequency}{DIVI} \quad \& \quad \frac{source\_frequency}{DIVI+1}$$

Jitter is therefore reduced by increasing the source clock frequency. In applications where jitter is a concern, the fastest available clock source should be used.

The General Purpose clocks have MASH noise-shaping dividers which push this fractional divider jitter out of the audio band.

MASH noise-shaping is incorporated to push the fractional divider jitter out of the audio band if required. The MASH can be programmed for 1, 2 or 3-stage filtering. MASH filter, the frequency is spread around the requested frequency and the user must ensure that the module is not exposed to frequencies higher than 25MHz. Also, the MASH filter imposes a low limit on the range of DIVI.

when MASH=0 the remainder DIVF is ignored. easy mistake!

MASH	min DIVI	min output freq	average output freq	max output freq
0 (int divide)	1	source / (DIVI)	source / (DIVI)	source / (DIVI)
1	2	source / (DIVI)	source / (DIVI + DIVF / 1024)	source / (DIVI + 1)
2	3	source / (DIVI - 1)	source / (DIVI + DIVF / 1024)	source / (DIVI + 2)
3	5	source / (DIVI - 3)	source / (DIVI + DIVF / 1024)	source / (DIVI + 4)

is 4096 (the size of the field)

Table 6-32 Effect of MASH Filter on Frequency

The following example illustrates the spreading of output clock frequency resulting from the use of the MASH filter. Note that the spread is greater for lower divisors.

PLL freq (MHz)	target freq (MHz)	MASH	divisor	DIVI	DIVF	min freq (MHz)	ave freq (MHz)	max freq (MHz)	error
650	18.32	0	35.480	35	492	18.57	18.57	18.57	ok
650	18.32	1	35.480	35	492	18.06	18.32	18.57	ok
650	18.32	2	35.480	35	492	17.57	18.32	19.12	ok
650	18.32	3	35.480	35	492	16.67	18.32	20.31	ok
400	18.32	0	21.834	21	854	19.05	19.05	19.05	ok
400	18.32	1	21.834	21	854	18.18	18.32	19.05	ok
400	18.32	2	21.834	21	854	17.39	18.32	20.00	ok
400	18.32	3	21.834	21	854	16.00	18.32	22.22	ok
200	18.32	0	10.917	10	939	20.00	20.00	20.00	ok
200	18.32	1	10.917	10	939	18.18	18.32	20.00	ok
200	18.32	2	10.917	10	939	16.67	18.32	22.22	ok
200	18.32	3	10.917	10	939	14.29	18.32	28.57	error

Table 6-33 Example of Frequency Spread when using MASH Filtering

It is beyond the scope of this specification to describe the operation of a MASH filter or to determine under what conditions the available levels of filtering are beneficial.

Operating Frequency

The maximum operating frequency of the General Purpose clocks is ~125MHz at 1.2V but this will be reduced if the GPIO pins are heavily loaded or have a capacitive load.

Clock Manager General Purpose Clocks Control (CM_GP0CTL, GP1CTL & GP2CTL)

Address 0x 7e10 1070 CM_GP0CTL
0x 7e10 1078 CM_GP1CTL
0x 7e10 1080 CM_GP2CTL

from the errata, the PCM clock control is right after these at: 0x7e101098.

writes that do not have 0x5a as the upper byte get ignored!

set to non-zero or will ignore DIVF (next page)

when disable, wait until BUSY=0 before making changes.

for i2s we use the fastest clock, the 19.2MHz oscilator (1)

Bit Number	Field Name	Description	Read/ Write	Reset
31-24	PASSWD	Clock Manager password "5a"	W	0
23-11	-	Unused	R	0
10-9	MASH	MASH control 0 = integer division 1 = 1-stage MASH (equivalent to non-MASH dividers) 2 = 2-stage MASH 3 = 3-stage MASH To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB.	R/W	0
8	FLIP	Invert the clock generator output This is intended for use in test/debug only. Switching this control will generate an edge on the clock generator output. To avoid output glitches do not switch this control while BUSY=1.	R/W	0
7	BUSY	Clock generator is running Indicates the clock generator is running. To avoid glitches and lock-ups, clock sources and setups must not be changed while this flag is set.	R	0
6	-	Unused	R	0
5	KILL	Kill the clock generator 0 = no action 1 = stop and reset the clock generator This is intended for test/debug only. Using this control may cause a glitch on the clock generator output.	R/W	0
4	ENAB	Enable the clock generator This requests the clock to start or stop without glitches. The output clock will not stop immediately because the cycle must be allowed to complete to avoid glitches. The BUSY flag will go low when the final cycle is completed.	R/W	0
3-0	SRC	Clock source 0 = GND 1 = oscillator 2 = testdebug0 3 = testdebug1 4 = PLLA per 5 = PLLC per 6 = PLLD per 7 = HDMI auxiliary 8-15 = GND To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB.	R/W	0

Table 6-34 General Purpose Clocks Control

Clock Manager General Purpose Clock Divisors (CM_GP0DIV, CM_GP1DIV & CM_GP2DIV)

Address 0x 7e10 1074 CM_GP0DIV
0x 7e10 107c CM_GP1DIV
0x 7e10 1084 CM_GP2DIV

```
PCM_DIV is missing!  from the errata it's right
after these at 0x7e10109c.
```

```
as with control
all writes need
to have the upper
byte = 0x5a or
they get ignored
```

Bit Number	Field Name	Description	Read/ Write	Reset
31-24	PASSWD	Clock Manager password "5a"	W	0
23-12	DIVI	Integer part of divisor This value has a minimum limit determined by the MASH setting. See text for details. To avoid lock-ups and glitches do not change this control while BUSY=1.	R/W	0
11-0	DIVF	Fractional part of divisor To avoid lock-ups and glitches do not change this control while BUSY=1.	R/W	0

Table 6-35 General Purpose Clock Divisors

```
note this is calculated weirdly!   let's say you want to down scale the oscillator by
1.2345678 ----- you would set DIVI=1 but you wouldn't set DIVF to 2345, instead you
*multiply* it by 4096 (the size of this field), round the result, and set DIVF to the integer.
so: round(.2345678 * 4096) = round(960.7897087...) = 961.
This gives: DIVI = 1, DIVF = 961.

Very easy mistake to make if you're used to software!   Ask me....
```

8 PCM / I2S Audio

The PCM audio interface is an APB peripheral providing input and output of telephony or high quality serial audio streams. It supports many classic PCM formats including I2S.

The PCM audio interface has 4 interface signals;

PCM_CLK	- bit clock.
PCM_FS	- frame sync signal.
PCM_DIN	- serial data input.
PCM_DOUT	- serial data output.

PCM is a serial format with a single bit data_in and single bit data_out. Data is always serialised MS-bit first.

The frame sync signal (PCM_FS) is used to delimit the serial data into individual frames. The length of the frame and the size and position of the frame sync are fully programmable.

Frames can contain 1 or 2 audio/data channels in each direction. Each channel can be between 8 and 32 bits wide and can be positioned anywhere within the frame as long as the two channels don't overlap. The channel format is separately programmable for transmit and receive directions.

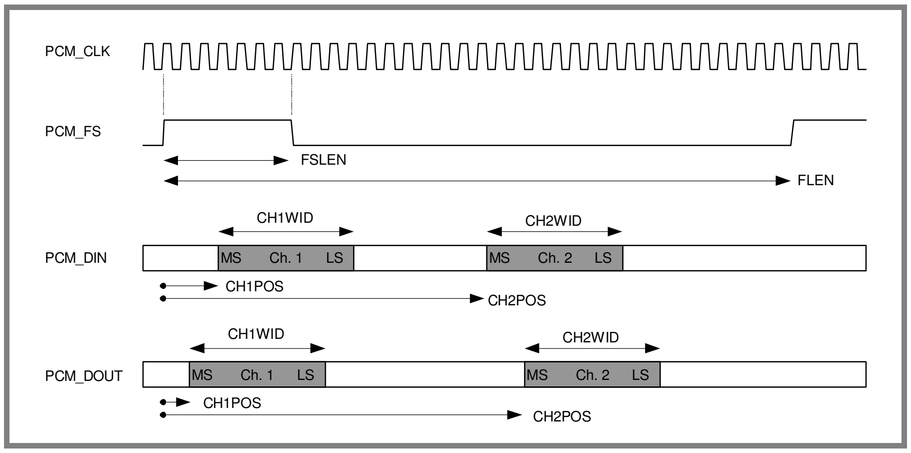

Figure 8-1 PCM Audio Interface Typical Timing

The PCM_CLK can be asynchronous to the bus APB clock and can be logically inverted if required.

The direction of the PCM_CLK and PCM_FS signals can be individually selected, allowing the interface to act as a master or slave device.

The input interface is also capable of supporting up to 2 PDM microphones, as an alternative to the classic PCM input format, in conjunction with a PCM output.

8.1 Block Diagram

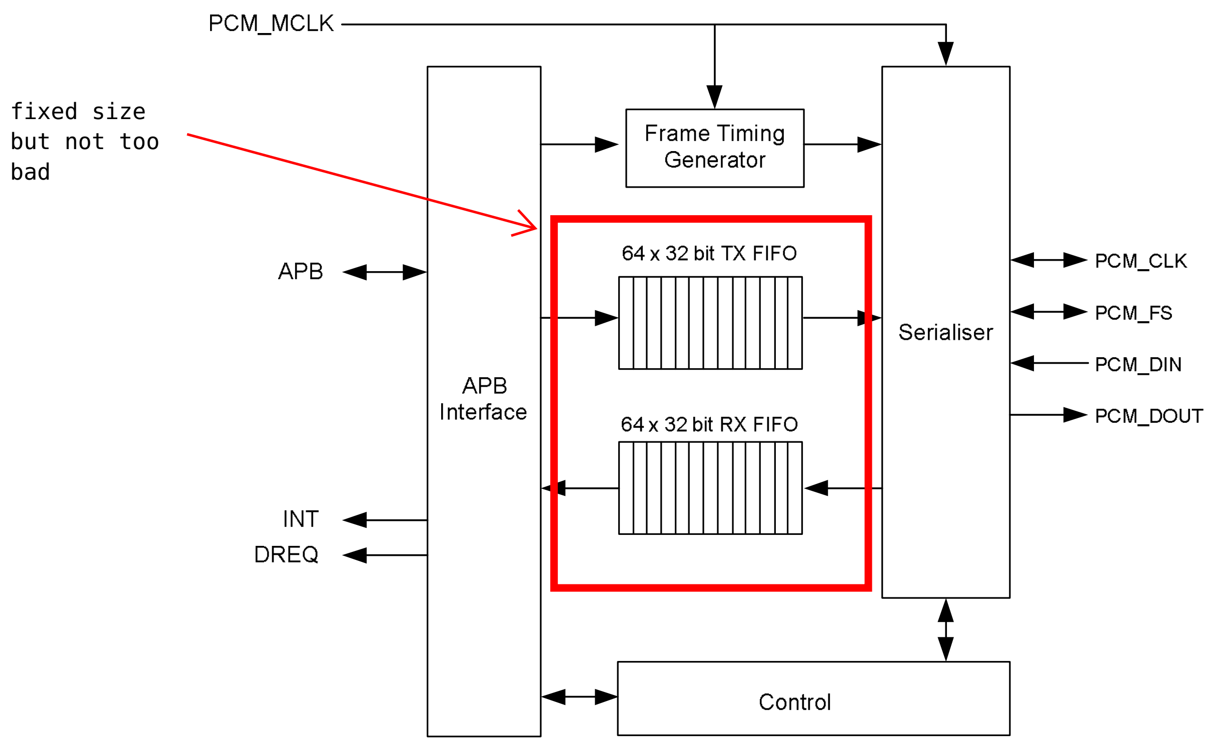

Figure 8-2 PCM Audio Interface Block Diagram

The PCM audio interface contains separate transmit and receive FIFOs. Note that if the frame contains two data channels, they must share the same FIFO and so the channel data will be interleaved. The block can be driven using simple polling, an interrupt based method or direct DMA control.

8.2 Typical Timing

Figure 8-1 shows typical interface timing and indicates the flexibility that the peripheral offers.

Normally PCM output signals change on the rising edge of PCM_CLK and input signals are sampled on its falling edge. The frame sync is considered as a data signal and sampled in the same way.

The front end of the PCM audio interface is run off the PCM_CLK and the PCM signals are timed against this clock. However, the polarity of the PCM_CLK can be physically inverted, in which case the edges are reversed.

In clock master mode (CLKM=0), the PCM_CLK is an output and is driven from the PCM_MCLK clock input.

In clock slave mode (CLKM=1), the PCM_CLK is an input, supplied by some external clock source.

In frame sync master mode (FSM=0), the PCM_FS is internally generated and is treated as a data output that changes on the positive edge of the clock. The length and polarity of the frame sync is fully programmable and it can be used as a standard frame sync signal, or as an L-R signal for I2S.

In frame sync slave mode (FSM=1), the PCM_FS is treated as a data input and is sampled on the negative edge of PCM_CLK. The first clock of a frame is taken as the first clock period where PCM_FS is sampled as a 1 following a period or periods where it was previously a 0. The PCM audio interface locks onto the incoming frame sync and uses this to indicate where the data channels are positioned. The precise timing at the start of frame is shown in Figure 8-3.

Note that in frame sync slave mode there are two synchronising methods. The legacy method is used when the frame length = 0. In this case the internal frame logic has to detect the incoming PCM_FS signal and reset the internal frame counter at the start of every frame. The logic relies on the PCM_FS to indicate the length of the frame and so can cope with adjacent frames of different lengths. However, this creates a short timing path that will corrupt the PCM_DOUT for one specific frame/channel setting.

The preferred method is to set the frame length to the expected length. Here the incoming PCM_FS is used to resynchronise the internal frame counter and this eliminates the short timing path.

8.3 Operation

```
so need to enable the interface, set it up, and then
at the last minute enable RX or TX (with EN=1)
```

The PCM interface runs asynchronously at the PCM_CLK rate and automatically transfers transmit and receive data across to the internal APB clock domain. The control registers are NOT synchronised and should be programmed before the device is enabled and should NOT be changed whilst the interface is running.

Only the EN, RXON and TXON bits of the PCMCS register are synchronised across the PCM - APB clock domain and are allowed to be changed whilst the interface is running.

The EN bit is a global power-saving enable. The TXON and RXON bits enable transmit and receive, and the interface is running whenever either TXON or RXON is enabled.

In operation, the PCM format is programmed by setting the appropriate frame length, frame sync, channel position values, and signal polarity controls. The transmit FIFO should be preloaded with data and the interface can then be enabled and started, and will run continuously until stopped. If the transmit FIFO becomes empty or the receive FIFO becomes full, the RXERR or TXERR error flags will be set, but the interface will just continue. If the RX FIFO overflows, new samples are discarded and if the TX FIFO underflows, zeros are transmitted.

Normally channel data is read or written into the appropriate FIFO as a single word. If the channel is less than 32 bits, the data is right justified and should be padded with zeros. If the RXSEX bit is set then the received data is sign extended up to the full 32 bits. When a frame is programmed to have two data channels, then each channel is written/read as a separate word in the FIFO, producing an interleaved data stream. When initialising the interface, the first word read out of the TX FIFO will be used for the first channel, and the data from the first channel on the first frame to be received will be the first word written into the RX FIFO.

If a FIFO error occurs in a two channel frame, then channel synchronisation may be lost which may result in a left right audio channel swap. RXSYNC and TXSYNC status bits are provided to help determine if channel slip has occurred. They indicate if the number of words in the FIFO is a multiple of a full frame (taking into account where we are in the current frame being transferred). This assumes that an integer number of frames data has been sent/read from the FIFOs.

If a frame is programmed to have two data channels and the packed mode bits are set (FRXP FTXP) then the FIFOs are configured so that each word contains the data for both channels (2x 16 bit samples). In this mode each word written to the TX FIFO contains 2 16 bit samples, and the Least Significant sample is transmitted first. Each word read from the RX FIFO will contain the data received from 2 channels, the first channel received will be in the Least Significant half of the word. If the channels size is less than 16 bits, the TX data will be truncated and RX data will be padded to 16 bits with zeros.

Note that data is always serialised MS-bit first. This is well-established behaviour in both PCM and I2S.

If the PDM input mode is enabled then channel 1 is sampled on the negative edge of PCM_CLK whilst channel 2 is sampled on the positive edge of PCM_CLK.

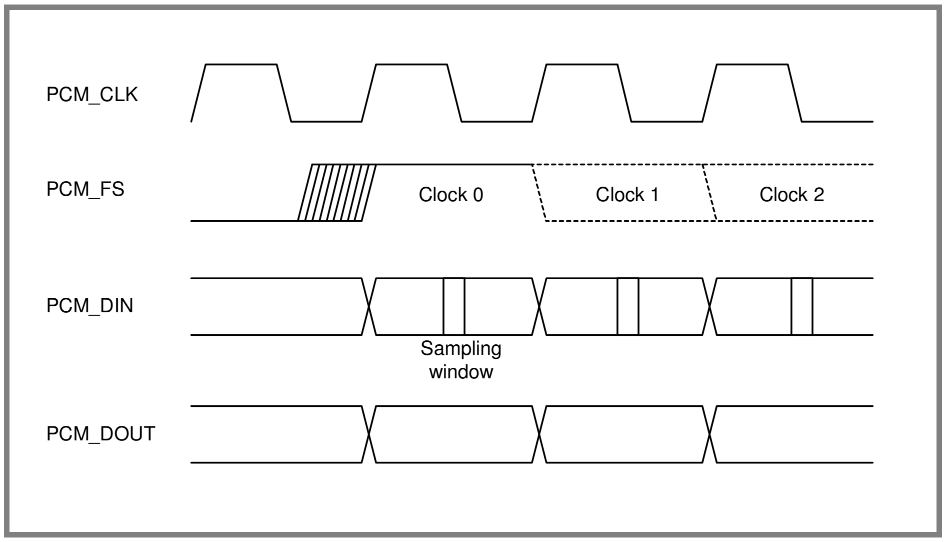

Figure 8-3 Timing at Start of Frame

Note that the precise timing of FS (when it is an input) is not clearly defined and it may change state before or after the positive edge of the clock. Here the first clock of the frame is defined as the clock period where the PCM_FS is sampled (negative edge) as a 1 where it was previously sampled as a 0.

this is how we configure (continues to next page)

8.4 Software Operation

8.4.1 Operating in Polled mode

Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.

If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation. Poll TXW writing sample words to PCMFIFO and RXR reading sample words from PCMFIFO until all data is transferred.

8.4.2 Operating in Interrupt mode

a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.

b) Set INTR and/or INTT to enable interrupts.

c) If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation.

d) When an interrupt occurs, check RXR. If this is set then one or more sample words are available in PCMFIFO. If TXW is set then one or more sample words can be sent to PCMFIFO.

8.4.3 DMA

a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed.

b) Set DMAEN to enable DMA DREQ generation and set RXREQ/TXREQ to determine the FIFO thresholds for the DREQs. If required, set TXPANIC and RXPANIC to determine the level at which the DMA should increase its AXI priority,

c) In the DMA controllers set the correct DREQ channels, one for RX and one for TX. Start the DMA which should fill the TX FIFO.

d) Set TXON and/or RXON to begin operation.

8.5 Error Handling.

In all software operational modes, the possibility of FIFO over or under run exists. Should this happen when using 2 channels per frame, there is a risk of losing sync with the channel data stored in the FIFO. If this happens and is not detected and corrected, then the data channels may become swapped.

The FIFO's will automatically detect an error condition caused by a FIFO over or under-run and this will set the appropriate latching error bit in the control/status register. Writing a '1' back to this error bit will clear the latched flag.

In a system using a polled operation, the error bits can be checked manually. For an interrupt or DMA based system, setting the INTE bit will cause the PCM interface to generate an interrupt when an error is detected.

If a FIFO error occurs during operation in which 2 data channels are being used then the synchronisation of the data may be lost. This can be recovered by either of these two methods:

a) Disable transmit and receive (TXON and RXON =0). Clear the FIFO's (RXCLR and TXCLR =1). Note that it may take up to 2 PCM clocks for the FIFOs to be physically cleared after initiating a clear. Then preload the transmit FIFO and restart transmission. This of course loses the data in the FIFO and further interrupts the data flow to the external device.

b) Examine the TXSYNC and RXSYNC flags. These flags indicate if the amount of data in the FIFO is a whole number of frames, automatically taking into account where we are in the current frame being transmitted or received. Thus, providing an even number of samples was read or written to the FIFOs, then if the flags are set then this indicates that a single word needs to be written or read to adjust the data. Normal exchange of data can then proceed (where the first word in a data pair is for channel 1). This method should cause less disruption to the data stream.

8.6 PDM Input Mode Operation

The PDM input mode is capable of interfacing with two digital half-cycle PDM microphones and implements a 4th order CIC decimation filter with a selectable decimation factor. The clock input of the microphones is shared with the PCM output codec and it should be configured to provide the correct clock rate for the microphones. As a result it may be necessary to add a number of padding bits into the PCM output and configure the output codec to allow for this.

When using the PDM input mode the bit width and the rate of the data received will depend on the decimation factor used. Once the data has been read from the peripheral a further decimation and filtering stage will be required and can be implemented in software. The software filter should also correct the droop introduced by the CIC filter stage. Similarly a DC correction stage should also be employed.

PDMN	PCM_CLK (MHz)	Peripheral Output Format	OSR	Fs
0 (N=16)	3.072	16 bits unsigned	4	48kHz
1 (N=32)	3.072	20 bits unsigned	2	48kHz

Table 8-1 PDM Input Mode Configuration

8.7 GRAY Code Input Mode Operation

GRAY mode is used for an incoming data stream only. GRAY mode is selected by setting the enable bit (EN) in the PCM_GRAY register.

In this mode data is received on the PCM_DIN (data) and the PCM_FS (strobe) pins. The data is expected to be in data/strobe format. In this mode data is detected when either the data or the strobe change state. As each bit is received it is written into the RX buffer and when 32 bits are received they are written out to the RXFIFO as a 32 bit word. In order for this mode to work the user must program a PCM clock rate which is 4 times faster then the gray data rate. Also the gray coded data input signals should be clean.

annotations are for what is needed for the i2s mic we use
(so: rx only, no interrupts, and 32-bit values)

The normal RXREQ and RXTHR FIFO levels will apply as for normal PCM received data.

If a message is received that is not a multiple of 32 bits, any data in the RX Buffer can be flushed out by setting the flush bit (FLUSH). Once set, this bit will read back as zero until the flush operation has completed. This may take several cycles as the APB clock may be many times faster than the PCM clock. Once the flush has occurred, the bits are packed up to 32 bits with zeros and written out to the RXFIFO. The flushed field (FLUSHED) will indicate how many of bits of this word are valid.

Note that to get an accurate indication of the number of bits currently in the rx shift register (RXLEVEL) the APB clock must be at least 2x the PCM_CLK.

i think this means the oscillator clock?

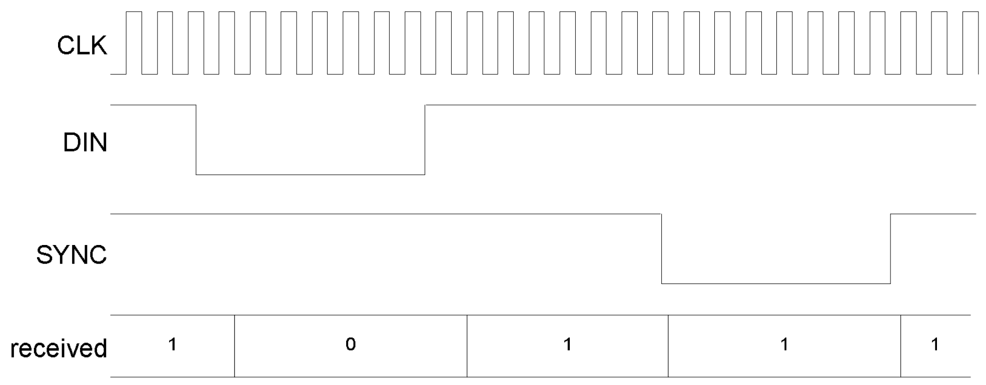

Figure 8-4 Gray mode input format

8.8 PCM Register Map

~~There is only~~ PCM module in the BCM2835. The PCM base address for the registers is 0x7E203000.

PCM Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	CS_A	PCM Control and Status	32
0x4	FIFO_A	PCM FIFO Data	32
0x8	MODE_A	PCM Mode	32
0xc	RXC_A	PCM Receive Configuration	32
0x10	TXC_A	PCM Transmit Configuration	32
0x14	DREQ_A	PCM DMA Request Level	32

0x18	INTEN_A	PCM Interrupt Enables	32
0x1c	INTSTC_A	PCM Interrupt Status & Clear	32
0x20	GRAY	PCM Gray Mode Control	32

CS_A Register

Synopsis This register contains the main control and status bits for the PCM. The bottom 3 bits of this register can be written to whilst the PCM is running. The remaining bits cannot.

Bit(s)	Field Name	Description	Type	Reset
31:26		***Reserved*** *- Write as 0, read as don't care*		
25	STBY	RAM Standby This bit is used to control the PCM Rams standby mode. By default this bit is 0 causing RAMs to start initially in standby mode. Rams should be released from standby prior to any transmit/receive operation. Allow for at least 4 PCM clock cycles to take effect. This may or may not be implemented, depending upon the RAM libraries being used.	RW	0x0
24	SYNC	PCM Clock sync helper. This bit provides a software synchronisation mechanism to allow the software to detect when 2 PCM clocks have occurred. It takes 2 PCM clocks before the value written to this bit will be echoed back in the read value.	RW	0x0
23	RXSEX	RX Sign Extend 0 = No sign extension. 1 = Sign extend the RX data. When set, the MSB of the received data channel (as set by the CHxWID parameter) is repeated in all the higher data bits up to the full 32 bit data width.	RW	0x0
22	RXF	RX FIFO is Full 0 = RX FIFO can accept more data. 1 = RX FIFO is full and will overflow if more data is received.	RO	0x0

PCM clock
rate is
much
slower
than
2 ARM CPU
cycles.
don't skip
this

21	TXE	TX FIFO is Empty 0 = TX FIFO is not empty. 1 = TX FIFO is empty and underflow will take place if no more data is written.	RO	0x1
20	RXD	Indicates that the RX FIFO contains data 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 sample.	RO	0x0
19	TXD	Indicates that the TX FIFO can accept data 0 = TX FIFO is full and so cannot accept more data. 1 = TX FIFO has space for at least 1 sample.	RO	0x1
18	RXR	Indicates that the RX FIFO needs reading 0 = RX FIFO is less than RXTHR full. 1 = RX FIFO is RXTHR or more full. This is cleared by reading sufficient data from the RX FIFO.	RO	0x0
17	TXW	Indicates that the TX FIFO needs Writing 0 = TX FIFO is at least TXTHR full. 1 = TX FIFO is less then TXTHR full. This is cleared by writing sufficient data to the TX FIFO.	RO	0x1
16	RXERR	RX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	RW	0x0
15	TXERR	TX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	RW	0x0
14	RXSYNC	RX FIFO Sync 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync.	RO	0x0
13	TXSYNC	TX FIFO Sync 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync.	RO	0x0
12:10		***Reserved** - Write as 0, read as don't care*		

for mic
we receive

reset these
for mic i2s
driver

9	DMAEN	DMA DREQ Enable 0 = Don t generate DMA DREQ requests. 1 = Generates a TX DMA DREQ requests whenever the TX FIFO level is lower than TXREQ or generates a RX DMA DREQ when the RX FIFO level is higher than RXREQ.	RW	0x0
8:7	RXTHR	Sets the RX FIFO threshold at which point the RXR flag is set 00 = set when we have a single sample in the RX FIFO 01 = set when the RX FIFO is at least full 10 = set when the RX FIFO is at least 11 = set when the RX FIFO is full	RW	0x0
6:5	TXTHR	Sets the TX FIFO threshold at which point the TXW flag is set 00 = set when the TX FIFO is empty 01 = set when the TX FIFO is less than full 10 = set when the TX FIFO is less than full 11 = set when the TX FIFO is full but for one sample	RW	0x0
4	RXCLR	Clear the RX FIFO . Assert to clear RX FIFO. This bit is self clearing and is always read as clear Note that it will take 2 PCM clocks for the FIFO to be physically cleared.	WO	0x0
3	TXCLR	Clear the TX FIFO Assert to clear TX FIFO. This bit is self clearing and is always read as clear. Note that it will take 2 PCM clocks for the FIFO to be physically cleared.	WO	0x0
2	TXON	Enable transmission 0 = Stop transmission. This will stop immediately if possible or else at the end of the next frame. The TX FIFO can still be written to to preload data. 1 = Start transmission. This will start transmitting at the start of the next frame. Once enabled, the first data read from the TX FIFO will be placed in the first channel of the frame, thus ensuring proper channel synchronisation. The frame counter will be started whenever TXON or RXON are set. This bit can be written whilst the interface is running.	RW	0x0

1	RXON	Enable reception. 0 = Disable reception. This will stop on the next available frame end. RX FIFO data can still be read. 1 = Enable reception. This will be start receiving at the start of the next frame. The first channel to be received will be the first word written to the RX FIFO. This bit can be written whilst the interface is running.	RW	0x0
0	EN	Enable the PCM Audio Interface 0 = The PCM interface is disabled and most logic is gated off to save power. 1 = The PCM Interface is enabled. This bit can be written whilst the interface is running.	RW	0x0

FIFO_A Register

Synopsis This is the FIFO port of the PCM. Data written here is transmitted, and received data is read from here.

```
GET32(FIFO_A)
to get
next 32-bit
value (RXD=1)
```

Bit(s)	Field Name	Description	Type	Reset
31:0		***Reserved*** *- Write as 0, read as don't care*		

MODE_A Register

Synopsis This register defines the basic PCM Operating Mode. It is used to configure the frame size and format and whether the PCM is in master or slave modes for its frame sync or clock. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31:29		***Reserved*** *- Write as 0, read as don't care*		

28	CLK_DIS	PCM Clock Disable 1 = Disable the PCM Clock. This cleanly disables the PCM clock. This enables glitch free clock switching between an internal and an uncontrollable external clock. The PCM clock can be disabled, and then the clock source switched, and then the clock re-enabled. 0 = Enable the PCM clock.	RW	0x0
27	PDMN	PDM Decimation Factor (N) 0 = Decimation factor 16. 1 = Decimation factor 32. Sets the decimation factor of the CIC decimation filter.	RW	0x0
26	PDME	PDM Input Mode Enable 0 = Disable PDM (classic PCM input). 1 = Enable PDM input filter. Enable CIC filter on input pin for PDM inputs. In order to receive data RXON must also be set.	RW	0x0
25	FRXP	Receive Frame Packed Mode 0 = The data from each channel is written into the RX FIFO. 1 = The data from both RX channels is merged (1st channel is in the LS half) and then written to the RX FIFO as a single 2x16 bit packed mode word. First received channel in the frame goes into the LS half word. If the received data is larger than 16 bits, the upper bits are truncated. The maximum channel size is 16 bits.	RW	0x0
24	FTXP	Transmit Frame Packed Mode 0 = Each TX FIFO word is written into a single channel. 1 = Each TX FIFO word is split into 2 16 bit words and used to fill both data channels in the same frame. The maximum channel size is 16 bits. The LS half of the word is used in the first channel of the frame.	RW	0x0
23	CLKM	PCM Clock Mode 0 = Master mode. The PCM CLK is an output and drives at the MCLK rate. 1 = Slave mode. The PCM CLK is an input.	RW	0x0

we are in master mode; since not transmitting i don't know if we need to disable the PCM clock.   can't hurt.

22	CLKI	Clock Invert this logically inverts the PCM_CLK signal. 0 = Outputs change on rising edge of clock, inputs are sampled on falling edge. 1 = Outputs change on falling edge of clock, inputs are sampled on rising edge.	RW	0x0
21	FSM	Frame Sync Mode 0 = Master mode. The PCM_FS is an output and we generate the frame sync. 1 = Slave mode. The PCM_FS is an input and we lock onto the incoming frame sync signal.	RW	0x0
20	FSI	Frame Sync Invert This logically inverts the frame sync signal. 0 = In master mode, FS is normally low and goes high to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 1 after being a 0. 1 = In master mode, FS is normally high and goes low to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 0 after being a 1.	RW	0x0
19:10	FLEN	Frame Length Sets the frame length to (FLEN+1) clocks. Used only when FSM == 0. 1 = frame length of 2 clocks. 2 = frame length of 3 clocks. etc	RW	0x0
9:0	FSLEN	Frame Sync Length Sets the frame sync length to (FSLEN) clocks. This is only used when FSM == 0. PCM_FS will remain permanently active if FSLEN >= FLEN. 0 = frame sync pulse is off. 1 = frame sync pulse is 1 clock wide. etc	RW	0x0

```
for mic, we need clock 64x the datarate.  so we configure for two 32-bit data channels: FLEN=63
and FSLEN=32 (off for 32 clocks, on for 32 clocks)
```

RXC_A Register

Synopsis Sets the Channel configurations for Receiving. This sets the position and width of the 2 receive channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be from the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset

need to enable the channel 1 and set it to 32-bits. make sure you check that the expected number of low bits are 0! (our mic puts out 18 bits)

31	CH1WEX	Channel 1 Width Extension Bit This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
30	CH1EN	Channel 1 Enable 0 = Channel 1 disabled and no data is received from channel 1 and written to the RX FIFO. 1 = Channel 1 enabled.	RW	0x0
29:20	CH1POS	Channel 1 Position This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
19:16	CH1WID	Channel 1 Width This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of (CH1WEX* 16) + CH1WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
14	CH2EN	Channel 2 Enable 0 = Channel 2 disabled and no data is received from channel 2 and written to the RX FIFO. 1 = Channel 2 enabled.	RW	0x0
13:4	CH2POS	Channel 2 Position This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
3:0	CH2WID	Channel 2 Width This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of (CH2WEX* 16) + CH2WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0

TXC_A Register

Synopsis Sets the Channel configurations for Transmitting. This sets the position and width of the 2 transmit channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be used in the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31	CH1WEX	Channel 1 Width Extension Bit This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
30	CH1EN	Channel 1 Enable 0 = Channel 1 disabled and no data is taken from the TX FIFO and transmitted on channel 1. 1 = Channel 1 enabled.	RW	0x0
29:20	CH1POS	Channel 1 Position This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
19:16	CH1WID	Channel 1 Width This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of (CH1WEX* 16) + CH1WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
14	CH2EN	Channel 2 Enable 0 = Channel 2 disabled and no data is taken from the TX FIFO and transmitted on channel 2. 1 = Channel 2 enabled.	RW	0x0

13:4	CH2POS	Channel 2 Position This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
3:0	CH2WID	Channel 2 Width This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of (CH2WEX* 16) + CH2WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0

DREQ_A Register

Synopsis Set the DMA DREQ and Panic thresholds. The PCM drives 2 DMA controls back to the DMA, one for the TX channel and one for the RX channel. DMA DREQ is used to request the DMA to perform another transfer, and DMA Panic is used to tell the DMA to use its panic level of priority when requesting thins on the AXI bus. This register cannot be changed whilst the PCM is running.

Bit(s)	**Field Name**	**Description**	**Type**	**Reset**
31		***Reserved*** *- Write as 0, read as don't care*		
30:24	TX_PANIC	TX Panic Level This sets the TX FIFO Panic level. When the level is below this the PCM will assert its TX DMA Panic signal.	RW	0x10
23		***Reserved*** *- Write as 0, read as don't care*		
22:16	RX_PANIC	RX Panic Level This sets the RX FIFO Panic level. When the level is above this the PCM will assert its RX DMA Panic signal.	RW	0x30
15		***Reserved*** *- Write as 0, read as don't care*		
14:8	TX	TX Request Level This sets the TX FIFO DREQ level. When the level is below this the PCM will assert its DMA DREQ signal to request more data is written to the TX FIFO.	RW	0x30
7		***Reserved*** *- Write as 0, read as don't care*		

6:0	RX	<u>RX Request Level</u> This sets the RX FIFO DREQ level. When the level is above this the PCM will assert its DMA DREQ signal to request that some more data is read out of the RX FIFO.	RW	0x20

INTEN_A Register

Synopsis Set the reasons for generating an Interrupt. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved** - Write as 0, read as don't care*		
3	RXERR	<u>RX Error Interrupt</u> Setting this bit enables interrupts from PCM block when RX FIFO error occurs.	RW	0x0
2	TXERR	<u>TX Error Interrupt</u> Setting this bit enables interrupts from PCM block when TX FIFO error occurs.	RW	0x0
1	RXR	<u>RX Read Interrupt Enable</u> Setting this bit enables interrupts from PCM block when RX FIFO level is greater than or equal to the specified RXTHR level.	RW	0x0
0	TXW	<u>TX Write Interrupt Enable</u> Setting this bit enables interrupts from PCM block when TX FIFO level is less than the specified TXTHR level.	RW	0x0

INTSTC_A Register

Synopsis This register is used to read and clear the PCM interrupt status. Writing a 1 to the asserted bit clears the bit. Writing a 0 has no effect.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved** - Write as 0, read as don't care*		

3	RXERR	RX Error Interrupt Status / Clear This bit indicates an interrupt occurred on RX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
2	TXERR	TX Error Interrupt Status / Clear This bit indicates an interrupt occurred on TX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
1	RXR	RX Read Interrupt Status / Clear This bit indicates an interrupt occurred on RX Read. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
0	TXW	TX Write Interrupt Status / Clear This bit indicates an interrupt occurred on TX Write. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0

GRAY Register

Synopsis This register is used to control the gray mode generation. This is used to put the PCM into a special data/strobe mode. This mode is under 'best effort ' contract.

Bit(s)	Field Name	Description	Type	Reset
31:22		***Reserved*** *- Write as 0, read as don't care*		
21:16	RXFIFOLEVEL	The Current level of the RXFIFO This indicates how many words are currently in the RXFIFO.	RO	0x0
15:10	FLUSHED	The Number of bits that were flushed into the RXFIFO This indicates how many bits were valid when the flush operation was performed. The valid bits are from bit 0 upwards. Non-valid bits are set to zero.	RO	0x0
9:4	RXLEVEL	The Current fill level of the RX Buffer This indicates how many GRAY coded bits have been received. When 32 bits are received, they are written out into the RXFIFO.	RO	0x0

3		***Reserved** - Write as 0, read as don't care*		
2	FLUSH	Flush the RX Buffer into the RX FIFO This forces the RX Buffer to do an early write. This is necessary if we have reached the end of the message and we have bits left in the RX Buffer. Flushing will write these bits as a single 32 bit word, starting at bit zero. Empty bits will be packed with zeros. The number of bits written will be recorded in the FLUSHED Field. This bit is written as a 1 to initiate a flush. It will read back as a zero until the flush operation has completed (as the PCM Clock may be very slow).	RW	0x0
1	CLR	Clear the GRAY Mode Logic This Bit will reset all the GRAY mode logic, and flush the RX buffer. It is not self clearing.	RW	0x0
0	EN	Enable GRAY Mode Setting this bit will put the PCM into GRAY mode. In gray mode the data is received on the data in and the frame sync pins. The data is expected to be in data/strobe format.	RW	0x0