gtbelib

gtbelibv1

Generated by Doxygen 1.8.7

Mon Jun 9 2014 23:20:29

Contents

Chapter 1

gtbelib

GTBE driver library for TI Tiva C MCUs

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Module Documentation

4.1 ADC

Drivers for the TI ads1278 ADC Interfaces the TI tiva C Launchpad with the TI ads1278 analog to digital converter. Files: adc_ads1278.c adc_ads1278.h.

Functions

- void ADC_initADC (uint32_t SysClkFreq)
- void ADC_initControls (void)
- void ADC_initDRDYint (void)
- void ADC_initDRDY (void)
- void ADC_initNSYNC (void)
- void ADC_initSSI (uint32_t SysClkFreq)
- void ADC_setTestMode (uint32_t testModeValue)
- void ADC_setCLKfreq (uint32_t clkFreqValue)
- void ADC_setMode (uint32_t modeValue)
- void ADC_setCLKdiv (uint32_t CLKdivValue)
- void ADC_initClkPWM (uint32_t sysClkFreqDiv)
- void ADC_setSerialFormat (uint32_t serialFormatValue)
- void ADC_initUDMAssiRX (void)
- void ADC_SSIRXuDMA_ISR (void)

4.1.1 Detailed Description

Drivers for the TI ads1278 ADC Interfaces the TI tiva C Launchpad with the TI ads1278 analog to digital converter. Files: adc_ads1278.c adc_ads1278.h.

Author

Curtis Mayberry

4.1.2 Function Documentation

4.1.2.1 void ADC_initADC (uint32_t *SysClkFreq*)

Initializes the ADC

Gives clk freq = 25 MHz assuming a system clk freq = 50MHz

/param modeValue Sets the mode of the ADC values: ADC_MODE_HIGH_SPEED, ADC_MODE_HIGH_RESOL, ADC_MODE_LOW_POWER, ADC_MODE_LOW_SPEED

Definition at line 80 of file adc_ads1278.c.

4.1.2.2 void ADC_initClkPWM (uint32_t *sysClkFreqDiv*)

Initializes the M1PWM5 to generate the clock for the ADC Utilizes PWM module 1 output 5 (PF1)

Clock Setup Options ∗25MHz - 50 MHz system clock, sysClkFreqDiv = ADC_PWM_DIV2 ∗26.67MHz - 80 MHz system clock, sysClkFreqDiv = ADC_PWM_DIV3 ∗33.33MHz - 66.67 MHz system clock, sysClkFreqDiv = ADC_PWM_DIV2 (Need 2.0v $<$ DVDD $<$ 2.2v) /param sysClkFreqDiv - frequency divider of the PWM values: ADC_PWM_DIV2, ADC_PWM_DIV3

Definition at line 258 of file adc_ads1278.c.

4.1.2.3 void ADC_initControls (void)

Initializes the control signals of the ADC

Definition at line 93 of file adc_ads1278.c.

4.1.2.4 void ADC_initDRDY (void)

Initialize ∼DRDY pin as an input

Definition at line 126 of file adc_ads1278.c.

4.1.2.5 void ADC_initDRDYint (void)

Initialize ∼DRDY pin as an input interrupt to trigger the read.

Note

need to add the interrupt prototype to the NVIC

Definition at line 109 of file adc_ads1278.c.

4.1.2.6 void ADC_initNSYNC (void)

Initializes ∼SYNC

Definition at line 135 of file adc_ads1278.c.

4.1.2.7 void ADC_initSSI (uint32_t *SysClkFreq*)

initializes SSI for use with the ADC

Definition at line 144 of file adc_ads1278.c.

4.1.2.8 void ADC_initUDMAssiRX (void)

Initializes the SSI RX uDMA to transfer 3 bytes from the fifo to the data buffer

Definition at line 309 of file adc_ads1278.c.

4.1.2.9 void ADC_setCLKdiv (uint32_t *CLKdivValue*)

Sets the clock divider setting of the ADC

/param clkDivValue Sets the clock divider setting values: ADC_CLKDIV_HIGHF, ADC_CLKDIV_LOWF

Definition at line 242 of file adc_ads1278.c.

4.1.2.10 void ADC_setCLKfreq (uint32_t *clkFreqValue*)

Sets the clock divider setting of the ADC

/param clkDivValue Sets the clock divider setting values: ADC_CLKDIV_HIGHF, ADC_CLKDIV_LOWF

Definition at line 186 of file adc_ads1278.c.

4.1.2.11 void ADC_setMode (uint32_t *modeValue*)

Sets the mode of the ADC

```
Mode            | MAX Fdata |
```

ADC_MODE_HIGH_SPEED | 144,531 | ADC_MODE_HIGH_RESOL | 52,734 | ADC_MODE_LOW_POWER | 52,734 | ADC_MODE_LOW_SPEED | 10,547 |

/param modeValue Sets the mode of the ADC values: ADC_MODE_HIGH_SPEED, ADC_MODE_HIGH_RESOL, ADC_MODE_LOW_POWER, ADC_MODE_LOW_SPEED

Definition at line 232 of file adc_ads1278.c.

4.1.2.12 void ADC_setSerialFormat (uint32_t *serialFormatValue*)

Sets the serial format for the ADC output Options: SPI or frame sync format Dynamic or fixed output positions Parallel or serial (TDM) output

Note

Controls FORMAT2, FORMAT1, FORMAT0

Definition at line 302 of file adc_ads1278.c.

4.1.2.13 void ADC_setTestMode (uint32_t *testModeValue*)

Sets the test mode of the ADC that tests the digital I/O

/param modeValue Sets the test mode of the ADC values: ADC_TEST_MODE_NORMAL, ADC_TEST_MODE_TEST

Definition at line 176 of file adc_ads1278.c.

4.1.2.14 void ADC_SSIRXuDMA_ISR (void)

uDMA interrupt service routine for receiving data from the ADC using the SSI uDMA RX channel

Definition at line 340 of file adc_ads1278.c.

4.2 ADC

Drivers for the ADI ad5754 DAC. Interfaces the TI tiva C Launchpad with the ADI dac5754 digital to analog converter. Files: dac_ad5754.c dac_ad5754.h.

Functions

- void DAC_initDAC (uint32_t rangeValue, uint32_t pwrSettingsValue, uint32_t SysClkFreq)
- void DAC_initSSI (uint32_t SysClkFreq)
- void DAC_initSSIint (void)
- void DAC_SSIIntEnableEOT (uint32_t ui32Base)
- void DAC_intHandlerSSI (void)
- void DAC_intHandlerSSItimer (void)
- uint32_t DAC_getSSInSettingsCR1 (uint32_t ui32Base)
- void DAC_initCtlLDAC (void)
- void DAC_initCtlCLR (void)
- void DAC_setRange (uint32_t dacAddress, uint32_t rangeValue)
- void DAC_setPowerCtl (uint32_t pwrSettingsValue)
- void DAC_setSettingsCtl (uint32_t ctlSettingsValue)
- void DAC_clearDACs (void)
- void DAC_clearDACsPin (void)
- void DAC_loadDACs (void)
- void DAC_loadDACsPin (void)
- void DAC_loadDACsPinTimer (void)
- void DAC_initTimersLDAC (bool enForcer, bool enQuad, uint32_t pulseWidth)
- void DAC_writeNop (void)
- void DAC_updateDataVolt (uint32_t dacAddress, uint32_t rangeValue, _Bool bin, float voltage)
- void DAC_updateDataDig (uint32_t dacAddress, uint32_t data)
- void DACd_initDAC (uint32_t rangeValue, uint32_t pwrSettingsValue, uint32_t SysClkFreq)
- void DACd_setSettingsCtl (uint32_t ctlSettingsValue)
- void DACd_setRange (uint32_t dacAddress, uint32_t rangeValue)
- void DACd_setPowerCtl (uint32_t pwrSettingsValue)
- void DACd_clearDACs_AH (void)
- void DACd_clearDACs_EH (void)
- void DACd_loadDACs_EH (void)
- void DACd_loadDACs_AH (void)
- void DACd_loadDACsPin_EH (void)
- void DACd_updateDataVolt (uint32_t dacAddress, uint32_t rangeValue, _Bool bin_AD, _Bool bin_EH, float voltage_AD, float voltage_EH)
- void DACd_updateDataDig (uint32_t dacAddress, uint32_t data_AD, uint32_t data_EH)

4.2.1 Detailed Description

Drivers for the ADI ad5754 DAC. Interfaces the TI tiva C Launchpad with the ADI dac5754 digital to analog converter. Files: dac_ad5754.c dac_ad5754.h.

Author

Curtis Mayberry

4.2.2 Function Documentation

4.2.2.1 void DAC_clearDACs (void)

Clears the output of DACs A-D by writing to the control register.

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Definition at line 312 of file dac_ad5754.c.

4.2.2.2 void DAC_clearDACsPin (void)

Clears the output of all 8 DACs by pulling ∼CLR pins low.

Definition at line 321 of file dac_ad5754.c.

4.2.2.3 uint32_t DAC_getSSInSettingsCR1 (uint32_t *ui32Base*)

Reads the SSI control register 1

Parameters

ui32Base	- The base address of the SSIn peripheral

Returns

SSI CR1 register value

Definition at line 226 of file dac_ad5754.c.

4.2.2.4 void DAC_initCtlCLR (void)

Initialize clear Control initializes: ∼CLR - control to clear the DAC output

Note

sets ∼CLR

Definition at line 246 of file dac_ad5754.c.

4.2.2.5 void DAC_initCtlLDAC (void)

Initialize LDAC Controls initializes: ∼LDAC_FORCER, ∼LDAC_Quad

Note

sets ∼LDAC_FORCER and ∼LDAC_Quad

Definition at line 235 of file dac_ad5754.c.

4.2.2.6 void DAC_initDAC (uint32_t *rangeValue,* uint32_t *pwrSettingsValue,* uint32_t *SysClkFreq*)

Initializes the DAC Sets the output voltage range, turns on the selected DACs and sets the control settings. NOTE: Make sure DAC_WRITE_DELAY is defined properly

Sets all DAC voltage output ranges to -5v to +5v turns on DAC output A and B Turns on thermal shutdown, output current clamp, and turns off SDO

Definition at line 119 of file dac_ad5754.c.

4.2.2.7 void DAC_initSSI (uint32_t *SysClkFreq*)

initializes SSI0 for use with the DAC

Definition at line 136 of file dac_ad5754.c.

4.2.2.8 void DAC_initSSIint (void)

Initializes the SSI interrupt to load the DAC Uses the SSI TX end of transmission (EOT) interrupt to signal when to trigger the loading of the DAC.

Definition at line 167 of file dac_ad5754.c.

4.2.2.9 void DAC_initTimersLDAC (bool *enForcer,* bool *enQuad,* uint32_t *pulseWidth*)

Initializes Timers for one shot operation of the LDAC pulse

Parameters

enForcer	- enables the LDAC_FORCER timer
enQuad	- enables the LDAC_QUAD timer
pulseWidth	- Number of clock cycles to pulse the LDAC signal low.

Note

Not available on the GTBE-TM4C123GXL

Definition at line 369 of file dac_ad5754.c.

4.2.2.10 void DAC_intHandlerSSI (void)

Interrupt handler for loading the DACs after updating them

Note

Need to add the ISR to the NVIC table in the row labeled "SSIX Rx and Tx"

Definition at line 196 of file dac_ad5754.c.

4.2.2.11 void DAC_intHandlerSSItimer (void)

Interrupt handler for loading the DACs with the timer after updating them

Note

Need to add the ISR to the NVIC table in the row labeled "SSIX Rx and Tx"

Definition at line 210 of file dac_ad5754.c.

4.2.2.12 void DAC_loadDACs (void)

Loads data into the DACs from the data registers of all 4 DACs /note A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ~sync goes high between writes.

Definition at line 332 of file dac_ad5754.c.

4.2.2.13 void DAC_loadDACsPin (void)

Loads all 4 dacs with the value currently in their data register by pulling the ~LDAC pin low

Definition at line 343 of file dac_ad5754.c.

4.2.2.14 void DAC_loadDACsPinTimer (void)

Loads all 4 dacs with the value currently in their data register by pulling the ~LDAC_FORCER pin low

Note

Uses the timer in one shot mode to pulse ~LDAC_FORCER low

Definition at line 356 of file dac_ad5754.c.

4.2.2.15 void DAC_setPowerCtl (uint32_t *pwrSettingsValue*)

Sets the power control settings to power up or down each DAC

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ~sync goes high between writes.

Parameters

pwrSettingsValue	powers up the selected DAC(s) Must be the bitwise or of one or more of the following values: DAC_PUA, DAC_PUB, DAC_PUC, DAC_PUD Those DACs not included will be cleared.

Definition at line 280 of file dac_ad5754.c.

4.2.2.16 void DAC_setRange (uint32_t *dacAddress,* uint32_t *rangeValue*)

Sets the voltage output range value of the given dac

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ~sync goes high between writes.

Inputs:

Parameters

dacAddress	(uint32_t): The dac selected to have its range set values: DAC_ADD_A, DAC_ADD_B, DAC_↩ ADD_C, DAC_ADD_ALL
rangeValue	(uint32_t): the output voltage range value to select

Definition at line 262 of file dac_ad5754.c.

4.2.2.17 void DAC_setSettingsCtl (uint32_t *ctlSettingsValue*)

Sets the DAC control register settings

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Parameters

ctlSettingsValue	sets the selected settings. Must be the bitwise or of one or more of the following values↩ : DAC_CTL_TSD, DAC_CTL_CLAMP, DAC_CTL_CLR_SEL Those settings not included will be cleared.

Definition at line 299 of file dac_ad5754.c.

4.2.2.18 void DAC_SSIIntEnableEOT (uint32_t *ui32Base*)

Enables the TX end of transmission (EOT) feature that allows the TXFF interrupt to trigger immediately when a transmission completes The EOT interrupt triggers the TXFF interrupt This enable is in the CR1 register rather than the interrupt mask (IM) register

Note

Used for EK-TM4C123GXL

Definition at line 183 of file dac_ad5754.c.

4.2.2.19 void DAC_updateDataDig (uint32_t *dacAddress,* uint32_t *data*)

Updates the data in the DAC's data register The data must be loaded before it will be output

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes

Parameters

dacAddress	Selected DAC(s)

data	Digital output to be placed in the data register of the selected DAC(s)

Definition at line 451 of file dac_ad5754.c.

4.2.2.20 void DAC_updateDataVolt (uint32_t *dacAddress*, uint32_t *rangeValue*, _Bool *bin*, float *voltage*)

Updates the data in the DAC's data register to the specified voltage

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes Assumes a Vref = Vrefp-Vrefn = 2.5v The data must be loaded before it will be output

Parameters

dacAddress	Selected DAC(s)
rangeValue	the output voltage range value for the selected DACs
bin	Digital format selection. Selected using pin BIN/∼2sCOMP in hardware Values: OFFSET_BI↩ NARY or TWOS_COMPLEMENT
voltage	Voltage to be output on the chosen DAC(s)

Definition at line 412 of file dac_ad5754.c.

4.2.2.21 void DAC_writeNop (void)

Writes a nop command to the DAC This function can be used to skip writing to a DAC in a daisy chain configuration.

Definition at line 393 of file dac_ad5754.c.

4.2.2.22 void DACd_clearDACs_AH (void)

Clears the output of all 8 DACs by writing to the control registers.

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Definition at line 696 of file dac_ad5754.c.

4.2.2.23 void DACd_clearDACs_EH (void)

Clears the output of all 8 DACs by writing to the control registers.

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Definition at line 706 of file dac_ad5754.c.

4.2.2.24 void DACd_initDAC (uint32_t *rangeValue,* uint32_t *pwrSettingsValue,* uint32_t *SysClkFreq*)

Initializes the DAC in Daisy Chain Operation Sets the output voltage range, turns on the selected DACs and sets the control settings. /note Make sure DAC_WRITE_DELAY is defined properly

/param Output voltage range of all DACs Sets all DAC voltage output ranges to -5v to +5v turns on DAC output A and B Turns on thermal shutdown, output current clamp, and turns off SDO

Definition at line 596 of file dac_ad5754.c.

4.2.2.25 void DACd_loadDACs_AH (void)

Loads data into the DACs from the data registers of all 8 DACs

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Definition at line 724 of file dac_ad5754.c.

4.2.2.26 void DACd_loadDACs_EH (void)

Loads data into the DACs from the data registers of DACs E-H

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Definition at line 715 of file dac_ad5754.c.

4.2.2.27 void DACd_loadDACsPin_EH (void)

Loads all dacs E-H with the value currently in their data register by pulling the ∼LDAC pin low

Definition at line 733 of file dac_ad5754.c.

4.2.2.28 void DACd_setPowerCtl (uint32_t *pwrSettingsValue*)

Sets the power control settings to power up or down each DAC when using Daisy chain operation.

Note

A delay must be placed between consecutive DAC write commands. This must be long enough to ensure that ∼sync goes high between writes.

Parameters

pwrSettingsValue	powers up the selected DAC(s) Must be the bitwise or of a stand alone operation parameter and a daisy chain operation parameter

Those DACs not included will be cleared.

Definition at line 676 of file dac_ad5754.c.

4.2.2.29 void DACd_setRange (uint32_t *dacAddress,* uint32_t *rangeValue*)

Sets the voltage output range value of the given daisy chained DACs

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Inputs:

Parameters

dacAddress	(uint32_t): The dac selected to have its range set
rangeValue	(uint32_t): the output voltage range value to select

Definition at line 649 of file dac_ad5754.c.

4.2.2.30 void DACd_setSettingsCtl (uint32_t *ctlSettingsValue*)

Sets the DAC control register settings in daisy chain operation

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes.

Parameters

ctlSettingsValue	sets the selected settings for each DAC. It is the logical or of a value of a normal and a daisy chain operation parameter Must be the bitwise or of one or more of the following values: DAC↩ _CTL_TSD, DAC_CTL_CLAMP, DAC_CTL_CLR_SEL, DAC_CTL_SKIP_AD with the birwise or of one or more of the following values: DAC_CTL_TSD_EH, DAC_CTL_CLAMP_EH, DAC_C↩ TL_CLR_SEL_EH, DAC_CTL_SKIP_EH

Those settings not included will be cleared.

Definition at line 625 of file dac_ad5754.c.

4.2.2.31 void DACd_updateDataDig (uint32_t *dacAddress,* uint32_t *data_AD,* uint32_t *data_EH*)

Updates the data in the DAC's data register The data must be loaded before it will be output

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes

Parameters

dacAddress	Selected DAC(s) Must be the bitwise or of a stand alone operation parameter and a daisy chain operation parameter
data_AD	Digital output to be placed in the data register of the DACs A through D
data_EH	Digital output to be placed in the data register of the DACs E through H (daisy-chained device)

Definition at line 783 of file dac_ad5754.c.

4.2.2.32 void DACd_updateDataVolt (uint32_t *dacAddress*, uint32_t *rangeValue*, _Bool *bin_AD*, _Bool *bin_EH*, float *voltage_AD*, float *voltage_EH*)

Updates the data in the DAC's data register to the specified voltage

Note

A delay must be placed between consecutive dac write commands. This must be long enough to ensure that ∼sync goes high between writes Assumes a Vref = Vrefp-Vrefn = 2.5v The data must be loaded before it will be output

Parameters

dacAddress	Selected DAC(s) Must be the bitwise or of a stand alone operation parameter and a daisy chain operation parameter
rangeValue	the output voltage range value for the selected DACs Must be the bitwise or of a stand alone operation parameter and a daisy chain operation parameter
bin_AD	Digital format selection for DACs A-D.
bin_EH	Digital format selection for DACs E-H. Selected using pin BIN/∼2sCOMP in hardware Values: OFFSET_BINARY or TWOS_COMPLEMENT
voltage_AD	Voltage to be output on the chosen A-D DAC(s)
voltage_EH	Voltage to be output on the chosen E-H DAC(s)

Definition at line 759 of file dac_ad5754.c.

4.3 Software Defined PLL

drivers for a software defined phase locked loop Used with the Georgia Tech Back End (GTBE) GTBE-EK-TM4C123↩ GXL GTBE-EK-TM4C1294XL Files: dac_ad5754.c dac_ad5754.h

Functions

- void initFIR (firInst *firFilt, float *filtCoeff)
- float FIRfilter (firInst *filt, float input)

4.3.1 Detailed Description

drivers for a software defined phase locked loop Used with the Georgia Tech Back End (GTBE) GTBE-EK-TM4C123↩ GXL GTBE-EK-TM4C1294XL Files: dac_ad5754.c dac_ad5754.h

Author

Curtis Mayberry

4.3.2 Function Documentation

4.3.2.1 float FIRfilter (firInst * *filt*, float *input*)

Applies an input to a running FIR filter, filt. The input is added to a circular buffer

Parameters

filt	- an initialized FIR filter
input	- A new data point to add to the filter's data buffer

Definition at line 77 of file swpll.c.

4.3.2.2 void initFIR (firInst * *firFilt*, float * *filtCoeff*)

Initializes an FIR filter

Definition at line 57 of file swpll.c.

4.4 Frequency Synthesizer

Driver for the Silicon Labs si5351 Frequency synthesizer Interfaces the TI tiva C Launchpad with the Silicon Labs si5351 frequency synthesizer Files: synth_si4351.c synth_si4351.h.

Functions

- void synth_init (bool synthEN, uint8_t xtal_cl, uint8_t *regConfig, uint8_t outputEn)
- void synth_initConfig (uint8_t *regConfig, uint8_t outputEn)
- void synth_initADCreclkENL (bool bEnable)
- void synth_outputDisableAll (void)
- void synth_outputEnable (uint8_t outputEn)
- void synth_powerDownOutputDrivers (void)
- void synth_setInterruptMasks (uint8_t interruptMasks)
- uint32_t synth_readINTstatus (void)
- void synth_clearINT (uint8_t interruptCLR)
- void synth_writeRegConfig (uint8_t *regConfig)
- void synth_PLLreset (uint8_t PLLrst)
- uint32_t synth_readDeviceStatus (void)
- void synth_setXTALcapacitance (uint8_t xtal_cl)
- void synth_fanoutEnable (uint8_t fanoutEN)
- void synth_initI2C (bool fast)
- void synth_writeRegI2C (uint8_t regNum, uint8_t regValue)
- void synth_writeConsecutiveRegsI2C (uint8_t regNumStart, uint8_t regCNT, uint8_t *regValues)
- uint32_t synth_readRegI2C (uint8_t regNum)

4.4.1 Detailed Description

Driver for the Silicon Labs si5351 Frequency synthesizer Interfaces the TI tiva C Launchpad with the Silicon Labs si5351 frequency synthesizer Files: synth_si4351.c synth_si4351.h.

Author

Curtis Mayberry

4.4.2 Function Documentation

4.4.2.1 void synth_clearINT (uint8_t *interruptCLR*)

Clears the selected interrupts in the interrupt status register

Parameters

interruptCLR	- interrupts to be cleared Logical or of one or more of the following values: SYNTH_INT_STATUS_SYS_INIT_STKY SYNTH_INT_STATUS_LOL_B_STKY SYNTH_INT_STATUS_LOL_A_STKY SYNTH_INT_STATUS_LOS_STKY

Note

Each interrupt is sticky and will remain high until it is cleared

Definition at line 216 of file synth_si5351.c.

4.4.2.2 void synth_fanoutEnable (uint8_t *fanoutEN*)

Selects which fanouts to enable

Parameters

fanoutEN	selects which fanouts are enabled Logical or of one or more of the following values: SYNTH_↩ FANOUT_EN_CLKIN SYNTH_FANOUT_EN_XO SYNTH_FANOUT_EN_MS

Definition at line 279 of file synth_si5351.c.

4.4.2.3 void synth_init (bool *synthEN*, uint8_t *xtal_cl*, uint8_t * *regConfig*, uint8_t *outputEn*)

Initializes the synthesizer

Parameters

synthEN	enables or disables the synthesizer outputs
xtal_cl	Crystal input load capacitance selection
regConfig	register configuration generated by ClockBuilder Desktop
outputEn	Outputs to be enabled

Definition at line 77 of file synth_si5351.c.

4.4.2.4 void synth_initADCreclkENL (bool *bEnable*)

Initializes ~ADC_RECLK_ENL to enable (false) or disable (true) the reclking of the ADC output to the synthesizer clock.

Parameters

bEnable	Active low enable signal for the D flip-flop that reclocks the ADC input when using the synthesizer to clock the ADC.

Definition at line 127 of file synth_si5351.c.

4.4.2.5 void synth_initConfig (uint8_t * *regConfig*, uint8_t *outputEn*)

Loads a configuration to the synth

Parameters

regConfig	Predefined register configuration array Choose a config array from synth_si5351_configs.h
outputEn	Enables the synthesizer output

Note

The configuration is generated by ClockBuilder Desktop
Configuration

Definition at line 103 of file synth_si5351.c.

4.4.2.6 void synth_initI2C (bool *fast*)

Initializes the synthesizer I2C serial interface in normal mode

Parameters

fast	- true sets the I2C interface to fast (400kbps) mode false sets the I2C interface to normal (100kbps) mode

Definition at line 293 of file synth_si5351.c.

4.4.2.7 void synth_outputDisableAll (void)

Disables all clock outputs

Definition at line 145 of file synth_si5351.c.

4.4.2.8 void synth_outputEnable (uint8_t *outputEn*)

Enables the selected synthesizer outputs

Parameters

outputEn	- clock outputs to be enabled. Those not included will be disabled Logical or of one or more of the following values: SYNTH_OUT_EN_CTL_CLK7_OEB SYNTH_OUT_EN_CTL_CLK6_OEB SYNTH_OUT_EN_CTL_CLK5_OEB SYNTH_OUT_EN_CTL_CLK4_OEB SYNTH_OUT_EN_↩ CTL_CLK3_OEB SYNTH_OUT_EN_CTL_CLK2_OEB SYNTH_OUT_EN_CTL_CLK1_OEB S↩ YNTH_OUT_EN_CTL_CLK0_OEB

Note

Si5351A (10-Pin MSOP) only has CLK 0-2

Definition at line 165 of file synth_si5351.c.

4.4.2.9 void synth_PLLreset (uint8_t *PLLrst*)

Resets the selected PLL

Parameters

PLLrst	- The PLL(s) to be reset Logical or of one or more of the following values: SYNTH_RST_PLL_↩ PLLB_RST SYNTH_RST_PLL_PLLA_RST

Definition at line 243 of file synth_si5351.c.

4.4.2.10 void synth_powerDownOutputDrivers (void)

Powers down all output drivers

Definition at line 172 of file synth_si5351.c.

4.4.2.11 uint32_t synth_readDeviceStatus (void)

Reads the device status register

Returns

device status (reg 0)

Definition at line 252 of file synth_si5351.c.

4.4.2.12 uint32_t synth_readINTstatus (void)

Reads the interrupt status register

Returns

interrupt status

Note

Each interrupt is sticky and will remain high until it is cleared

Definition at line 200 of file synth_si5351.c.

4.4.2.13 uint32_t synth_readRegI2C (uint8_t *regNum*)

Reads from a single register

Parameters

regNum	- Register number to be read.

Returns

Register value read

Definition at line 367 of file synth_si5351.c.

4.4.2.14 void synth_setInterruptMasks (uint8_t *interruptMasks*)

Sets interrupt masks

Parameters

interruptMasks	- Indicates the interrupts that are to be masked Logical or of one or more of the following values: SYNTH_INT_STATUS_MASK_SYS_INIT_MASK SYNTH_INT_STATUS_MASK_LOL_B_MA↩ SK SYNTH_INT_STATUS_MASK_LOL_A_MASK SYNTH_INT_STATUS_MASK_LOS_MASK

Note

When an interrupt is masked it prevents the associated pin from going low when the interrupt is asserted

Definition at line 189 of file synth_si5351.c.

4.4.2.15 void synth_setXTALcapacitance (uint8_t *xtal_cl*)

Sets the internal load capacitance of the crystal

Parameters

xtal_cl	- internal crystal load capacitance Adds the selected capacitance in parallel with both XTAL inputs Select one of the following values: SYNTH_XTAL_CL_6PF SYNTH_XTAL_CL_8PF SYNTH_XTAL_CL_10PF

Definition at line 266 of file synth_si5351.c.

4.4.2.16 void synth_writeConsecutiveRegsI2C (uint8_t *regNumStart,* uint8_t *regCNT,* uint8_t * *regValues*)

Writes to consecutive registers e.g. writes to register number 1,2,3,...

Parameters

regNumStart	- Register number of the first register to be written.
regCNT	- The number of registers to write. Must have regCNT $>$ 1
regValues	- The register values that are to be written

Definition at line 335 of file synth_si5351.c.

4.4.2.17 void synth_writeRegConfig (uint8_t * *regConfig*)

Writes the device configuration from the register map generated by ClockBuilder Desktop

Parameters

regConfig	- configuration array of each register value

Definition at line 226 of file synth_si5351.c.

4.4.2.18 void synth_writeRegI2C (uint8_t *regNum,* uint8_t *regValue*)

Writes to a single register

Parameters

regNum	- Register number to be written.
regValue	- The register value that is to be written

Definition at line 314 of file synth_si5351.c.

4.5 Temperature Sensor

Drivers for the ADI ADT7310 SPI digital temperature sensor Files: temp_ADT7310.c temp_ADT7310.h.

Functions

- void temp_configSensorContinuousRead (uint32_t *statusreg)
- void temp_resetSPI (void)
- float temp_degC13bitReading (uint32_t reading)
- float temp_degF13bitReading (uint32_t reading)
- void temp_uDMAErrorHandler (void)
- void temp_initSSI0uDMA (void)
- void temp_initTimer2A (void)
- void temp_Timer2IntHandler (void)

4.5.1 Detailed Description

Drivers for the ADI ADT7310 SPI digital temperature sensor Files: temp_ADT7310.c temp_ADT7310.h.

Author

Curtis Mayberry

4.5.2 Function Documentation

4.5.2.1 void temp_configSensorContinuousRead (uint32_t * *statusreg*)

Configures the temperature sensor for continuous read mode and reads the status register Utilizes SSI0

Parameters

statusreg	- A pointer to store the status information that is read

Definition at line 87 of file temp_ADT7310.c.

4.5.2.2 float temp_degC13bitReading (uint32_t *reading*)

Converts digital temperature reading to degrees Celcius

Parameters

reading	- 13 bit digital temperature reading from the temperature sensor

Returns

Temperature in degrees Celcius

Definition at line 145 of file temp_ADT7310.c.

4.5.2.3 float temp_degF13bitReading (uint32_t *reading*)

Converts digital temperature reading to degrees Fahrenheit

Parameters

reading	- 13 bit digital temperature reading from the temperature sensor

Returns

Temperature in degrees Fahrenheit

Definition at line 163 of file temp_ADT7310.c.

4.5.2.4 void temp_initSSI0uDMA (void)

Initializes the SSI0 to be used with the SSI0 uDMA

Definition at line 208 of file temp_ADT7310.c.

4.5.2.5 void temp_initTimer2A (void)

Initializes Timer 2A to control read sequence timing

Definition at line 241 of file temp_ADT7310.c.

4.5.2.6 void temp_resetSPI (void)

Resets the SPI bus to clear any partial instructions, etc. Utilizes SSI0

Definition at line 123 of file temp_ADT7310.c.

4.5.2.7 void temp_Timer2IntHandler (void)

Timer Interrupt handler Be sure that the interrupt is added to the NVIC by adding it to the "Timer 2 subtimer A location in tm4c123gh6pm_startup_CCS.c

Definition at line 264 of file temp_ADT7310.c.

4.5.2.8 void temp_uDMAErrorHandler (void)

uDMA transfer error handler Need to add to NVIC table in the "uDMA Error" Row

Definition at line 184 of file temp_ADT7310.c.

4.6 Tivaware Extension

Extension for the Tivaware Driver Library for TI Tiva C MCUs Used with the Georgia Tech Back End (GTBE) GTBE-EK-TM4C123GXL GTBE-EK-TM4C1294XL Files: tw_extension.c tw_extension.h.

Functions

- int32_t twe_initFlash (uint32_t codeStart, uint32_t codeReserveLength, uint32_t dataStart, uint32_t dataEraseLength)
- int32_t twe_eraseFlashRange (uint32_t startAddress, uint32_t eraseLength)
- int32_t twe_protectFlashRange (uint32_t startAddress, uint32_t protectLength, tFlashProtection eProtect)
- void twe_FLASH_badWriteISR (void)
- void twe_initFPU (void)
- void twe_initFPUlazy (void)
- void twe_initUART (uint32_t SysClkFreq, uint32_t baudRate)
- void twe_initSystem80MHz (void)
- void twe_initUDMAcontroller (void)
- void twe_uDMAErrorHandler (void)
- void twe_initProcessingIndicator (void)
- void twe_SSIIntEnableEOT (uint32_t ui32Base)
- void twe_I2CMasterVerify (uint32_t ui32base, bool burst, bool receive)

4.6.1 Detailed Description

Extension for the Tivaware Driver Library for TI Tiva C MCUs Used with the Georgia Tech Back End (GTBE) GTBE-EK-TM4C123GXL GTBE-EK-TM4C1294XL Files: tw_extension.c tw_extension.h.

Author

Curtis Mayberry

4.6.2 Function Documentation

4.6.2.1 int32_t twe_eraseFlashRange (uint32_t *startAddress*, uint32_t *eraseLength*)

Erases the indicated flash memory

Parameters

startAddress	Start location of flash memory that is to be erased it must land on a 1KB boundry
eraseLength	Length of data to erase in flash memory it must land on a 1KB boundry, set to 0x0 to skip erasure

Definition at line 139 of file tw_extension.c.

4.6.2.2 void twe_FLASH_badWriteISR (void)

Interrupt handler (ISR) that executes when an improper write to flash is attempted

Definition at line 185 of file tw_extension.c.

4.6.2.3 void twe_I2CMasterVerify (uint32_t *ui32base*, bool *burst*, bool *receive*)

Verifies that a transmission from the master has completed successfully

Parameters

ui32base	Base address of the I2C peripheral
burst	Set to true if burst mode was used for the transmission
receive	Set to true if a receive transmission is to be verified Set to false if a send transmission is to be verified

Note

Waits until a transmission is complete and then checks that no errors have occurred
If an error occurs the I2C transmission is stopped, the error LED is lit and the program enters an infinite loop to hold the state.

Definition at line 647 of file tw_extension.c.

4.6.2.4 int32_t twe_initFlash (uint32_t *codeStart,* uint32_t *codeReserveLength,* uint32_t *dataStart,* uint32_t *dataEraseLength*)

Initializes the Flash Protects the part of the flash memory from address codeStart through codeStart + codeReserve↩ Length by setting to only execute (FlashExecuteOnly) Sets up the flash interrupt in case the program attempts to access the protected flash portions. Make sure the ISR is added to the NVIC table

Note

Make sure the NVIC is not contained in the protected code memory

Parameters

codeStart	- start location of flash memory that is to be reserved it must land on a 2KB boundry
codeReserve↩ Length	- length of code to protect in flash memory it must land on a 2KB boundry, set to 0x0 to skip protection
dataStart	- start location of flash memory that is to be erased it must land on a 1KB boundry
dataEraseLength	- length of data to erase in flash memory it must land on a 1KB boundry, set to 0x0 to skip erasure

Definition at line 119 of file tw_extension.c.

4.6.2.5 void twe_initFPU (void)

Initializes the FPU

Definition at line 198 of file tw_extension.c.

4.6.2.6 void twe_initFPUlazy (void)

Initializes the FPU with lazy stacking enabled

Definition at line 205 of file tw_extension.c.

4.6.2.7 void twe_initProcessingIndicator (void)

Initializes the processing indicator GPIO output

Definition at line 387 of file tw_extension.c.

4.6.2.8 void twe_initSystem80MHz (void)

Initializes the UART interrupt for commands Receives commands for the device

Definition at line 329 of file tw_extension.c.

4.6.2.9 void twe_initUART (uint32_t *SysClkFreq,* uint32_t *baudRate*)

Initializes the UART

Parameters

SysClkFreq	- clock frequency of the system
baudRate	- baud rate of the UART e.g. 115200 to connect to PC

Note

UART is connected to the stellaris virtual serial port through the USB connection
Configuration: 8 data bits one stop bit no parity

Definition at line 230 of file tw_extension.c.

4.6.2.10 void twe_initUDMAcontroller (void)

Initialize the uDMA controller at the system level. It also enables it to continue to run while the processor is in sleep.

Note

Individual uDMA channels need to be initialized seperately.
Must also initialize the uDMA table immediately following this call. i.e. MAP_uDMAControlBaseSet(uDMAcontrol↩Table);

Definition at line 348 of file tw_extension.c.

4.6.2.11 int32_t twe_protectFlashRange (uint32_t *startAddress,* uint32_t *protectLength,* tFlashProtection *eProtect*)

Protects the indicated flash memory

Parameters

startAddress	Start location of flash memory that is to be reserved it must land on a 2KB boundry
protectLength	Length of code to protect in flash memory it must land on a 2KB boundry, set to 0x0 to skip protection
eProtect	Protection type

Definition at line 163 of file tw_extension.c.

4.6.2.12 void twe_SSIIntEnableEOT (uint32_t *ui32Base*)

Enables the TX end of transmission (EOT) feature that allows the TXFF interrupt to trigger immediately when a transmission completes

Note

The EOT interrupt triggers the TXFF interrupt
The TM4C1294NCPDT also has an independent EOT interrupt
This enable is in the CR1 register rather than the interrupt mask (IM) register

Definition at line 511 of file tw_extension.c.

4.6.2.13 void twe_uDMAErrorHandler (void)

uDMA Error Handler

Note

Need to add the ISR to the NVIC table in the row labeled "uDMA Error"

Definition at line 369 of file tw_extension.c.

Chapter 5

Data Structure Documentation

5.1 firInst Struct Reference

`#include <swpll.h>`

Data Fields

- float * coeff
- float dataBuffer [NUM_TAPS]
- float * pData [NUM_TAPS]
- float * dataHead
- float * dataBufferStart
- float * dataBufferEnd

5.1.1 Detailed Description

fir filter instance for use with CMSIS A structure to define an FIR filter in the CMSIS DSP library

Definition at line 69 of file swpll.h.

5.1.2 Field Documentation

5.1.2.1 float* coeff

FIR filter coefficients

Definition at line 70 of file swpll.h.

5.1.2.2 float dataBuffer[NUM_TAPS]

Input Data buffer

Definition at line 71 of file swpll.h.

5.1.2.3 float* dataBufferEnd

end of the data buffer

Definition at line 75 of file swpll.h.

5.1.2.4 float* dataBufferStart

start of the data buffer

Definition at line 74 of file swpll.h.

5.1.2.5 float* dataHead

head of the data

Definition at line 73 of file swpll.h.

5.1.2.6 float* pData[NUM_TAPS]

Pointer to the data

Definition at line 72 of file swpll.h.

The documentation for this struct was generated from the following file:

- F:/Documents/GitHub/gtbelib/swpll.h

5.2 floatFlash Struct Reference

`#include <tw_extension.h>`

Data Fields

- union {
- uint32_t address

5.2.1 Detailed Description

Data type for saving a float to flash memory Allows a float to be read as if it were a unsigned integer

Definition at line 37 of file tw_extension.h.

5.2.2 Field Documentation

5.2.2.1 uint32_t address

data memory address

Definition at line 42 of file tw_extension.h.

5.2.2.2 union { ... } data

float data to be stored

The documentation for this struct was generated from the following file:

- F:/Documents/GitHub/gtbelib/tw_extension.h

5.3 ints24 Union Reference

`#include <adc_ads1278.h>`

Data Fields

- int32_t intn [1]
- uint32_t uintn [1]

5.3.1 Detailed Description

Data type for saving incoming 24bit data

Definition at line 73 of file adc_ads1278.h.

5.3.2 Field Documentation

5.3.2.1 int32_t intn[1]

24 bit integer stored as a signed int

Definition at line 74 of file adc_ads1278.h.

5.3.2.2 uint32_t uintn[1]

24 bit integer stored as an unsigned int

Definition at line 75 of file adc_ads1278.h.

The documentation for this union was generated from the following file:

- F:/Documents/GitHub/gtbelib/adc_ads1278.h