ADC04b_DRV	*General Purpose 4 Conversion ADC Driver (bipolar)*

Description

This module allows 4-channel analog-to-digital conversion of bipolar signals with programmable gains and offsets. The conversions are triggered on GP Timer 1 underflow (for 281x) or EPWM1 CNT_zero event (for 280x).

Availability

This 16-bit module is available in one interface format:

1) The C interface version

Module Properties

Type: Target Dependent, Application Independent

Target Devices: x281x or x280x

C Version File Names: f281xadc04b.c, f281xadc04b.h (for x281x)
f280xadc04b.c, f280xadc04b.h (for x280x)

IQmath library files for C: N/A

Item	C version	Comments
Code Size□ (x281x/x280x)	138/150 words	
Data RAM	0 words•	
xDAIS ready	No	
XDAIS component	No	IALG layer not implemented
Multiple instances	Yes	
Reentrancy	Yes	

• Each pre-initialized ADCVALSB structure consumes 17 words in the data memory

□ Code size mentioned here is the size of the ***init()*** and ***read()*** functions

C Interface

Object Definition

The structure of ADCVALSB object is defined by following structure definition

```
typedef struct { int16 Ch1Gain;        // Parameter: Gain for channel 1 (Q13)
                 int16 Ch1Offset;      // Parameter: Offset for channel 1 (Q15)
                 int16 Ch1Out;         // Output: Channel 1 output (Q15)
                 int16 Ch2Gain;        // Parameter: Gain for channel 2 (Q13)
                 int16 Ch2Offset;      // Parameter: Offset for channel 2 (Q15)
                 int16 Ch2Out;         // Output: Channel 2 output (Q15)
                 int16 Ch3Gain;        // Parameter: Gain for channel 3 (Q13)
                 int16 Ch3Offset;      // Parameter: Offset for channel 3 (Q15)
                 int16 Ch3Out;         // Output: Channel 3 output (Q15)
                 int16 Ch4Gain;        // Parameter: Gain for channel 4 (Q13)
                 int16 Ch4Offset;      // Parameter: Offset for channel 4 (Q15)
                 int16 Ch4Out;         // Output: Channel 4 output (Q15)
                 Uint16 ChSelect;      // Parameter: ADC channel selection
                 void (*init)();       // Pointer to the init function
                 void (*read)();       // Pointer to the read function
               } ADCVALSB;

typedef ADCVALSB *ADCVALSB_handle;
```

Item	Name	Description	Format	Range(Hex)
Inputs	ADCINw, ADCINx, ADCINy, ADCINz	ADC pins in 281x device where w,x,y,z correspond to the channel numbers selected by Ch_sel	N/A	0-3 V
Outputs	Ch1Out	w^{th} channel digital representation	Q15	8000-7FFF
	Ch2Out	x^{th} channel digital representation	Q15	8000-7FFF
	Ch3Out	y^{th} channel digital representation	Q15	8000-7FFF
	Ch4Out	z^{th} channel digital representation	Q15	8000-7FFF
ADCVALSB parameter	ChSelect	16-bit ADC channel select format can be seen as: ChSelect = zyxwh	Q0	w,x,y,z are in between 0h -> Fh
	Ch1Gain	Gain for w^{th} channel. Modify this if default gain is not used.	Q13	8000-7FFF
	Ch2Gain	Gain for x^{th} channel. Modify this if default gain is not used.	Q13	8000-7FFF
	Ch3Gain	Gain for y^{th} channel. Modify this if default gain is not used.	Q13	8000-7FFF
	Ch4Gain	Gain for z^{th} channel. Modify this if default gain is not used.	Q13	8000-7FFF
	Ch1Offset	Offset for w^{th} channel. Modify this if default offset is not used.	Q15	8000-7FFF
	Ch2offset	Offset for x^{th} channel. Modify this if default offset is not used.	Q15	8000-7FFF
	Ch3Offset	Offset for y^{th} channel. Modify this if default offset is not used.	Q15	8000-7FFF
	Ch4Offset	Offset for z^{zh} channel. Modify this if default offset is not used.	Q15	8000-7FFF

Special Constants and Data types

ADCVALSB
The module definition is created as a data type. This makes it convenient to instance an interface to the ADCVALSB driver. To create multiple instances of the module simply declare variables of type ADCVALSB.

ADCVALSB_handle
User defined Data type of pointer to ADCVALSB module

ADCVALSB_DEFAULTS
Structure symbolic constant to initialize ADCVALSB module. This provides the initial values to the terminal variables as well as method pointers.

Methods

void F281X_adc04b_drv_init(ADCVALSB *);
void F281X_adc04b_drv_read(ADCVALSB *);

void F280X_adc04b_drv_init(ADCVALSB *);
void F280X_adc04b_drv_read(ADCVALSB *);

This default definition of the object implements two methods – the initialization and the runtime read function for ADCVALSB measurement. This is implemented by means of a function pointer, and the initializer sets this to F281X_adc04b_drv_init and F281X_adc04b_drv_read functions for x281x or F280X_adc04b_drv_init and F280X_adc04b_drv_read functions for x280x. The argument to this function is the address of the ADCVALSB object.

Module Usage

Instantiation
The following example instances one ADCVALSB object
ADCVALSB adc04b_1;

Initialization
To Instance pre-initialized objects
ADCVALSB adc04b_1 = ADCVALSB_DEFAULTS;

Invoking the computation function
adc04b_1.init(&adc04b_1);
adc04b_1.read(&adc04b_1);

Example
The following pseudo code provides the information about the module usage.

```
main()
{

        adc04b_1.init(&adc04b_1);                    // Call init function for adc04b_1

}
```

```
void interrupt periodic_interrupt_isr()
{

        adc04b_1.read(&adc04b_1);                       // Call read function for adc04b_1

        adc1 = _IQ15toIQ((long)adc04b_1.Ch1Out);            // adc1 is in GLOBAL_Q
        adc2 = _IQ15toIQ((long)adc04b_1.Ch2Out);            // adc2 is in GLOBAL_Q
        adc3 = _IQ15toIQ((long)adc04b_1.Ch3Out);            // adc3 is in GLOBAL_Q
        adc4 = _IQ15toIQ((long)adc04b_1.Ch4Out);            // adc4 is in GLOBAL_Q

}
```

Technical Background

The ADCIN pins accepts the analog input signals in the range of 0-3 volts for x28xx based DSP with ground referenced to 0 volt (VREFLO = 0).

Consequently, before connecting these signals to ADCIN pins, the hardware adjustment by external op-amp circuits (for gain and offset adjustments) for these analog signals such that they represent according to the proper scaling.

Four output variables of the module (Ch1Out, Ch2Out, Ch3Out, and Ch4Out) are computed after four ADC analog input signals are digitized as seen below:

Ch1Out = Ch1Gain*ADCINw_Q15 + Ch1Offset
Ch2Out = Ch2Gain*ADCINx_Q15 + Ch2Offset
Ch3Out = Ch3Gain*ADCINy_Q15 + Ch3Offset
Ch4Out = Ch4Gain*ADCINz_Q15 + Ch4Offset

Note that ADCINn_Q15 (n=w,x,y,z) are already converted to Q15 number.

Basically, the signals can be categorized into two main types: bipolar and unipolar signals. In this module, four ADCIN input signals are supposed to be bipolar. Thus, the Q15-number conversion is necessary for the bipolar signal measurements after the input signals are digitized as seen in Figure 1.

The input signals are typically sensed and re-scaled within the range of 0-3 volts for x28xx based DSP with the appropriate scaling. It is emphasized that the ADC unit is 12-bit resolution with left justified in the 16-bit ADC result register. Thus, the ADC output range is in between 0000h and FFF0h.

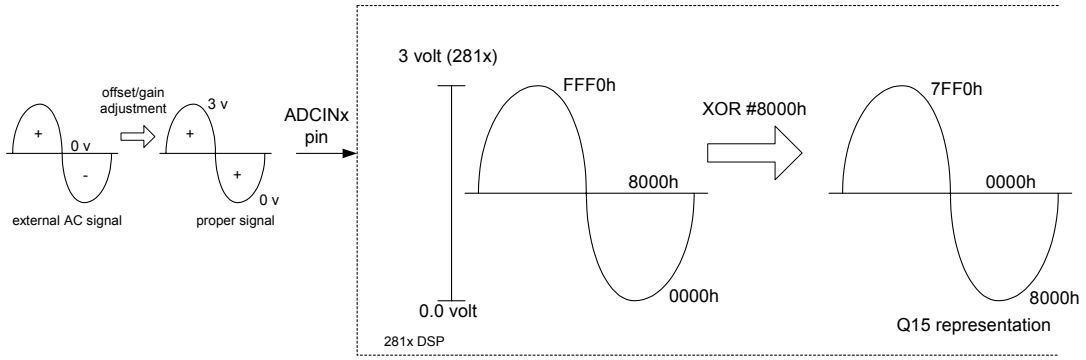

Figure 1: Q15-number conversion for bipolar signal measurements

After converted to Q15-number, the number is distorted a little bit about the maximum value (e.g., 7FF0h for bipolar at the maximum value of 7FFFh).