dsPIC33F/PIC24H
ADC Peripheral Module Library Help

Table of Contents

1 Library Features

This peripheral library module provides:

- Support for all dsPIC33F and PIC24H devices with ADC modules.
- Module initialization functions.
- Simple functions to start conversions.
- Function to read conversion results and configure interrupts.
- Simple interface macros to enable/disable interrupts.

2 Using the Library Module in a Project

Library routine parameters can be constructed using an AND based mask. For more information on this masks, see 16-bit Peripheral Libraries.

3 Functions

3.1 BusyADCx

Function Prototype

```
char BusyADC1(void);
char BusyADC2(void);
```

Include	`adc.h`
Description	This function returns the ADC conversion status.
Arguments	None
Return Value	If the value of DONE is '0', then '1' is returned, indicating that the ADC is busy in conversion. If the value of DONE is '1', then '0' is returned, indicating that the ADC has completed conversion.
Remarks:	This function returns the complement of the ADCONx <DONE> bit status which indicates whether the ADC is busy in conversion.
Source File:	`BusyADC1.c` `BusyADC2.c`
Code Example	`while(BusyADC2());`

3.2 CloseADCx

Function Prototype	`void CloseADC1(void);` `void CloseADC2(void);`
Include	`adc.h`
Description	This function turns off the ADC module and disables the ADC interrupts.
Arguments	None
Return Value	None
Remarks:	This function first disables the ADC interrupt and then turns off the ADC module.The Interrupt Flag bit (ADIF) is also cleared.
Source File:	`CloseADC1.c` `CloseADC2.c`
Code Example	`CloseADC2();`

3.3 ConfigIntADCx

Function Prototype	`void ConfigIntADC1(unsigned int config);` `void ConfigIntADC2(unsigned int config);`
Include	`adc.h`
Description	This function configures the ADC interrupt.

Arguments	*config* - ADC interrupt priority and enable/disable information as defined below:
	ADC Interrupt enable/disable
	`ADC_INT_ENABLE`
	`ADC_INT_DISABLE`
	ADC Interrupt priority
	`ADC_INT_PRI_0`
	`ADC_INT_PRI_1`
	`ADC_INT_PRI_2`
	`ADC_INT_PRI_3`
	`ADC_INT_PRI_4`
	`ADC_INT_PRI_5`
	`ADC_INT_PRI_6`
	`ADC_INT_PRI_7`

Return Value	None.
Remarks:	This function clears the Interrupt Flag (ADIF) bit and then sets the interrupt priority and enables/disables the interrupt.
Source File:	`ConfigIntADC1.c` `ConfigIntADC2.c`
Code Example	`ConfigIntADC2(ADC_INT_PRI_3 & ADC_INT_DISABLE);`

3.4 ConvertADCx

Function Prototype	`void ConvertADC1(void);` `void ConvertADC2(void);`
Include	`adc.h`
Description	This function starts the A/D conversion.

Arguments	None
Return Value	None
Remarks:	This function clears the ADxCON1<SAMP> bit and thus stops sampling and starts conversion. This happens only when trigger source for the A/D conversion is selected as Manual, by clearing the ADxCON1 <SSRC> bits.
Source File:	`ConvertADC1.c` `ConvertADC2.c`
Code Example	`ConvertADC2();`

3.5 StopSampADCx

Description	This function is identical to `ConvertADCx`.
Source File:	`#define` to `ConvertADCx` in `adc.h`.

3.6 OpenADCx

Function Prototype

```
void OpenADC1(unsigned int config1,
   unsigned int config2,
   unsigned int config3,
   unsigned int config4,
   unsigned int configport_l,
   unsigned int configport_h,
   unsigned int configscan_h,
   unsigned int configscan_l)
void OpenADC2(unsigned int config1,
   unsigned int config2,
   unsigned int config3,
   unsigned int config4,
   unsigned int configport_l,
   unsigned int configport_h,
   unsigned int configscan_h,
   unsigned int configscan_l)
```

Include `adc.h`

Description This function configures the ADC.

Arguments *config1* - This contains the parameters to be configured in the ADxCON1 register as defined below:

Module On/Off
- `ADC_MODULE_ON`
- `ADC_MODULE_OFF`

Idle mode operation
- `ADC_IDLE_CONTINUE`
- `ADC_IDLE_STOP`

DMA Buffers Write Mode
- `ADC_ADDMABM_ORDER`
- `ADC_ADDMABM_SCATTR`

ADC Module Operation Mode
- `ADC_AD12B_12BIT`
- `ADC_AD12B_10BIT`

Result output format
- `ADC_FORMAT_SIGN_FRACT`
- `ADC_FORMAT_FRACT`
- `ADC_FORMAT_SIGN_INT`
- `ADC_FORMAT_INTG`

Conversion trigger source
- `ADC_CLK_AUTO`
- `ADC_CLK_MPWM`
- `ADC_CLK_TMR`
- `ADC_CLK_INT0`
- `ADC_CLK_MANUAL`

Auto sampling select
```
ADC_AUTO_SAMPLING_ON
ADC_AUTO_SAMPLING_OFF
```
Sampling Mode Control
```
ADC_SIMULTANEOUS
ADC_MULTIPLE
```
Sample enable
```
ADC_SAMP_ON
ADC_SAMP_OFF
```

config2 - This contains the parameters to be configured in the ADCON2 register as defined below:
Voltage Reference
```
ADC_VREF_AVDD_AVSS
ADC_VREF_EXT_AVSS
ADC_VREF_AVDD_EXT
ADC_VREF_EXT_EXT
```
Scan selection
```
ADC_SCAN_ON
ADC_SCAN_OFF
```
A/D channels utilized
```
ADC_CONVERT_CH0123
ADC_CONVERT_CH01
ADC_CONVERT_CH0
```
DMA Address Increment Rate
```
ADC_DMA_ADD_INC_1
ADC_DMA_ADD_INC_2
.....
ADC_DMA_ADD_INC_15
ADC_DMA_ADD_INC_16
```
Buffer mode select
```
ADC_ALT_BUF_ON
ADC_ALT_BUF_OFF
```
Alternate Input Sample mode select
```
ADC_ALT_INPUT_ON
ADC_ALT_INPUT_OFF
```

config3 - This contains the parameters to be configured in the ADCON3 register as defined below:
Auto Sample Time bits
```
ADC_SAMPLE_TIME_0
ADC_SAMPLE_TIME_1
.....
ADC_SAMPLE_TIME_30
ADC_SAMPLE_TIME_31
```
Conversion Clock Source select
```
ADC_CONV_CLK_INTERNAL_RC
ADC_CONV_CLK_SYSTEM
```
Conversion clock select
```
ADC_CONV_CLK_Tcy2
ADC_CONV_CLK_Tcy
ADC_CONV_CLK_3Tcy2
ADC_CONV_CLK_2Tcy
.....
ADC_CONV_CLK_32Tcy
```

config4 - This contains the parameters to be configure in ADxCON4 register as defined below:
DMA Buffer Locations per Analog Input
```
ADC_DMA_BUF_LOC_128
ADC_DMA_BUF_LOC_64
```

```
ADC_DMA_BUF_LOC_32
.....
ADC_DMA_BUF_LOC_1
```

configport_h - This contains the pin select to be configured into the ADPCFG register as defined below:

```
ENABLE_ALL_ANA_16_31
ENABLE_ALL_DIG_16_31
ENABLE_AN16_ANA
ENABLE_AN17_ANA
ENABLE_AN18_ANA
.....
ENABLE_AN31_ANA
```

configport_l - This contains the pin select to be configured into the ADPCFG register as defined below:

```
ENABLE_ALL_ANA_0_15
ENABLE_ALL_DIG_0_15
ENABLE_AN0_ANA
ENABLE_AN1_ANA
ENABLE_AN2_ANA
.....
ENABLE_AN15_ANA
```

configscan_l - This contains the scan select parameter to be configured into the ADxCSSL register as defined below:

```
SCAN_NONE
SCAN_ALL
SKIP_SCAN_AN0
SKIP_SCAN_AN1
.....
SKIP_SCAN_AN15
```

configscan_h - This contains the scan select parameter to be configured into the ADxCSSH register as defined below:

```
SCAN_NONE
SCAN_ALL
SKIP_SCAN_AN16
SKIP_SCAN_AN1
.....
SKIP_SCAN_AN31
```

Return Value None

Remarks: This function configures the ADC for the following parameters:
Operating mode, Sleep mode behavior, DMA Buffer Write Mode, Data o/p format, ADC Module Mode, Conversion trigger source, Sampling control, VREF source, Scan selection, DMA address increment rate, Buffer Fill mode, Alternate i/p sample mod, Auto sample time, Conv. clock source, Conv Clock Select bits, Port Config Control bits.

Source File: OpenADC1.c
OpenADC2.c

Code Example

```
unsigned int config1;
unsigned int config2;
unsigned int config3;
unsigned int config4;
unsigned int configport_h;
unsigned int configport_l;
unsigned int configscan_h;
unsigned int configscan_l;

config1 = ADC_MODULE_ON & ADC_IDLE_STOP &
ADC_ADDMABM_ORDER & ADC_AD_12B_12BIT &
ADC_FORMAT_SIGN_INT & ADC_CLK_TMR &
```

```
AUTO_SAMPLING_ON & ADC_MULTIPLE ;

config2 = ADC_VREF_AVDD_AVSS & ADC_SCAN_ON &
ADC_CONVERT_CH0123 & ADC_DMA_ADD_INC_1 ;

config3 =  ADC_CONV_CLK_SYSTEM & ADC_SAMPLE_TIME_3 &
ADC_CONV_CLK_Tcy2;

config4 = ADC_DMA_BUF_LOC_32;

configport_h = ENABLE_ALL_ANA_16_31;

configport_l = ENABLE_ALL_ANA_0_15;

configscan_l = SCAN_ALL
configscan_l = SCAN_ALL;

OpenADC1(config1,config2,config3,config4,configport_l,configport_h,
configscan_h,configscan_l);
```

3.7 ReadADCx

Function Prototype
```
unsigned int ReadADC1(unsigned char bufIndex);
unsigned int ReadADC2(unsigned char bufIndex);
```
Include `adc.h`

Description This function reads the ADC Buffer register which contains the conversion value.

Arguments *`bufIndex`* - This is the ADC buffer number which is to be read.

Return Value None

Remarks: This function returns the contents of the ADC Buffer register. You should provide a value between '0' to '15' to ensure correct read of the ADCxBUF0 to ADCxBUFF.

Source File: `ReadADC1.c` `ReadADC2.c`

Code Example
```
unsigned int result;
result = ReadADC1(3);
```

3.8 SetChanADCx

Function Prototype
```
void SetChanADC1(unsigned int channel123,
   unsigned int channel0);
void SetChanADC2(unsigned int channel123,
   unsigned int channel0);
```
Include `adc10.h`

Description This function sets up ADCx input channels

Arguments *`channel123`* - This contains the S/H 1, 2 and 3 input select parameter to be configured into the ADCHS register as defined below:

A/D Channel 1, 2, 3 Negative input for Sample A
- `ADC_CH123_NEG_SAMPLEA_9_10_11`
- `ADC_CH123_NEG_SAMPLEA_6_7_8`
- `ADC_CH123_NEG_SAMPLEA_VREFN`

A/D Channel 1, 2, 3 Negative input for Sample B
- `ADC_CH123_NEG_SAMPLEB_9_10_11`
- `ADC_CH123_NEG_SAMPLEB_6_7_8`
- `ADC_CH123_NEG_SAMPLEB_VREFN`

A/D Channel 1, 2, 3 Positive input for Sample A
- `ADC_CH123_POS_SAMPLEA_3_4_5`
- `ADC_CH123_POS_SAMPLEA_0_1_2`

A/D Channel 1, 2, 3 Positive input for Sample B

ADC_CH123_POS_SAMPLEB_3_4_5
ADC_CH123_POS_SAMPLEB_0_1_2

channel0 - This contains the S/H 0 input select parameter to be configured into the ADCHS register as defined below:
A/D Channel 0 positive i/p select for Sample A
ADC_CH0_POS_SAMPLEA_AN0
ADC_CH0_POS_SAMPLEA_AN1
.....
ADC_CH0_POS_SAMPLEA_AN31
A/D Channel 0 negative i/p select for Sample A
ADC_CH0_NEG_SAMPLEA_AN1
ADC_CH0_NEG_SAMPLEA_VREFN
A/D Channel 0 positive i/p select for Sample B
ADC_CH0_POS_SAMPLEB_AN0
ADC_CH0_POS_SAMPLEB_AN1
.....
ADC_CH0_POS_SAMPLEB_AN31
A/D Channel 0 negative i/p select for Sample B
ADC_CH0_NEG_SAMPLEB_AN1
ADC_CH0_NEG_SAMPLEB_VREFN

Return Value	None
Remarks:	This function configures the inputs for sample multiplexers A and B by writing to ADxCH123 and ADxCHS0 register.
Source File:	SetChanADC1.c SetChanADC2.c
Code Examples	SetChanADC1(ADC_CH0_POS_SAMPLEA_AN0 & ADC_CH0_NEG_SAMPLEA_VREFN, ADC_CH0_POS_SAMPLEB_AN31 & ADC_CH0_NEG_SAMPLEB_VREFN);

4 Macros

4.1 EnableIntADC1

Macro	EnableIntADC1
Overview	This macro sets ADC Interrupt Enable bit in Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	EnableIntADC1;

4.2 DisableIntADC1

Macro	DisableIntADC1
Overview	This macro clears ADC Interrupt Enable bit in Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	DisableIntADC1;

4.3 SetPriorityIntADC1

Macro	SetPriorityIntADC1
Overview	This macro sets ADC Interrupt Priority bits of Interrupt Priority Control register.
Input	Interrupt Priority Level.
Output	None
Remarks	None
Code Example	SetPriorityIntADC1(7);

4.4 EnableIntADC2

Macro	`EnableIntADC2`
Overview	This macro sets ADC Interrupt Enable bit in Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	`EnableIntADC2;`

4.5 DisableIntADC2

Macro	`DisableIntADC2`
Overview	This macro clears ADC Interrupt Enable bit in Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	`DisableIntADC2;`

4.6 SetPriorityIntADC2

Macro	`SetPriorityIntADC2`
Overview	This macro sets ADC Interrupt Priority bits of Interrupt Priority Control register.
Input	Interrupt Priority Level.
Output	None
Remarks	None
Code Example	`SetPriorityIntADC2(7);`