# Table of contents



# Table of Figures



# Index of Tables

# 1 Overview

This software module controls the ADC on the F/C280x devices. It sets up the ADC in continuous conversion mode, to convert two channels, and to make it available in a variable in the application.

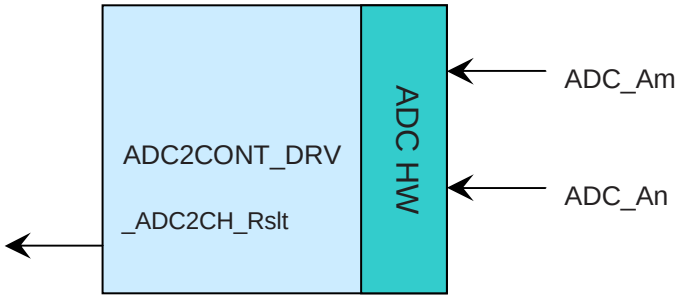


*Figure 1. Continuous conversion, two channel ADC Driver*

# 2 Module Properties

This section describes module properties, such as compatible devices, components, invocation etc. The ADC2CONT_DRV module has the following dependencies:

| Module | Dependency |
|---|---|
| CPU dependency | C28x |
| Device dependency | x2801 / x2806 / x2808 members only |

*Table 1. ADC2CONT_DRV module dependencies*

The ADC2CONT_DRV module has the following components:

| Component | Present |
|---|---|
| C-based initialization | Yes |
| ASM interrupt initialization | Yes |
| ASM runtime macro | Yes |

*Table 2. ADC2CONT_DRV module components*

The ADC2CONT_DRV module has the following miscellaneous properties:

| Property name | Property value |
|---|---|
| Multiple instance support | No |
| Reentrant | No |
| Accessible from 'C' environment | Yes |
| Full configuration from 'C' environment | Yes |
| Input / Output connection | Pointer to signal net. |

*Table 3. ADC2CONT_DRV module miscellaneous properties*

| Component | Files |
|---|---|
| Macro source: | C:\tidcs\DPS_C280x\vXYZ[†]\dplib280x\dplib280x.inc |
| C Interface: | C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.h |
| C source | C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.src |
| Object file archive | C:\tidcs\DPS_C280x\vXYZ\dplib280x\dplib280x.lib |

*Table 4. ADC2CONT_DRV module component files*

# 3 Module Input and Output Definitions

## 3.1 Module inputs

| Input name | Description | Data Format | Range |
|---|---|---|---|
| ADCINAm | F280x/C280x ADC input pin | Analog sense value | See device datasheet for electrical specifications. |
| ADCINAn | F280x/C280x ADC input pin | Analog sense value | See device datasheet for electrical specifications. |

*Table 5. ADC2CONT_DRV module inputs*

## 3.2 Module outputs

| Output name | Description | Data Format | Output data Range |
|---|---|---|---|
| ADC2CH_Rslt | Driver output pointer | Pointer to 16-bit fixed point output data | Q15: [-1, 1]<br>or<br>[-32768, 32767] |

*Table 6. ADC2CONT_DRV module outputs*

[†] The xyz represents the version number directory level. For instance, a 1.00 release would have v100 in its directory path, and v210 would indicate a release 2.10.

# 4 Module API Description

This module has three executable code components, as described in Table 2. Each of these components is described in this section.

## 4.1 ADC2CONTCnf

**Function Name:** ADC2CONTCnf

**Prototype:** void ADC2CONTCnf(int nChannel1, int nChannel2, int AcqWidth);

**Return value**: None.

**Preconditions**: The following preconditions must be satisfied:

The ADC module clock must be enabled in the PCLKCR0 register.

The ADC2CONTCnf function is called from the C environment, and performs driver configuration, including configuration of ADC registers.

- **nChannel1:** Channel to be converted to output 0.

  **Valid Range:** 0-15, corresponding to ADC channels A0-A7 and B0-B7. The channel selection is shown in Table 7 below.

- **nChannel2:** Channel to be converted to output 1.

  **Valid Range:** 0-15, corresponding to ADC channels A0-A7 and B0-B7. The channel selection is shown in Table 7 below.

- **AcqWidth**: Specifies the acquisition prescaler in clock cycles.

| Channel setting | Channel converted |
|---|---|
| 0 | ADCINA0 |
| 1 | ADCINA1 |
| 2 | ADCINA2 |
| 3 | ADCINA3 |
| 4 | ADCINA4 |
| 5 | ADCINA5 |
| 6 | ADCINA6 |
| 7 | ADCINA7 |
| 8 | ADCINB0 |
| 9 | ADCINB1 |
| 10 | ADCINB2 |
| 11 | ADCINB3 |
| 12 | ADCINB4 |
| 13 | ADCINB5 |
| 14 | ADCINB6 |
| 15 | ADCINB7 |

*Table 7. ADC2CONT_DRV channel selection*

**Example:** Call the ADC2CONTCnf function to initalize EPWM1 module.

```
//---------------------------------------------------------------------
// Set up conversion of channel 0, with acq prescaler at 2 clocks
//---------------------------------------------------------------------
      ADC2CONTCnf(0, 2);
```

## 4.2 ADC2CH_DRV_INIT

**Function Name:** ADC2CH_DRV_INIT

**Prototype:** ADC2CH_DRV_INIT

**Return value**: None.

**Preconditions**: The following preconditions must be satisfied:

The ADC module clock must be enabled in the PCLKCR0 register, and the C language init routine must be called.

This function is the assembler initialization macro, and must be called in addition to the C language initialization routine, for proper operation of the runtime macro routine. This initialization routine must be executed as part of an assembler initialization routine. This macro routine declares variables, initializes variables to known values, and sets up constants for the runtime macro routines.

**Example**: **Call the ADC2CH_DRV_INIT to initialize the ADC2CH_DRV module**.

```
;-------------------------------------------------------
; ISR Initialisation
;-------------------------------------------------------
_ISR_Init:  ADC2CH_DRV_INIT
            LRETR
```

## 4.3 ADC2CH_DRV

**Function Name:** ADC2CH_DRV

**Prototype:** ADC2CH_DRV

**Return value**: None.

**Preconditions**: The following preconditions must be satisfied:

- The ADC module clock must be enabled in the PCLKCR0register.
- C language init routine must be called.
- The ISR initialization macro ADC2CH_DRV_INIT must be instanced and executed in an assembler initialization routine.

This function is the assembler run time macro, and this creates code that forms a bridge between software and the ADC register hardwar output. The output is written to the ADC2CH_Rslt variable.

**Example: Call the ADC2CH_DRV in an assembler ISR**

```
;---------------------------------------------------------
; Runtime interrupt service routine
;---------------------------------------------------------
_ISR_Run:   CONTEXT_SAVE            ;call macro
            ADC2CH_DRV
            . . . .                 ; other code
;---------------------------------------------------------
EXIT_ISR: ;Interrupt management before exit
;---------------------------------------------------------
            MOVW  DP,#ETCLR1>>6
            MOV   @ETCLR1,#0x01     ; Clear EPWM1 Int flag

;---------------------------------------------------------
; Restore context & return
;---------------------------------------------------------
            CONTEXT_REST
            IRET
```

# 5 Usage Example:

Usage Example:

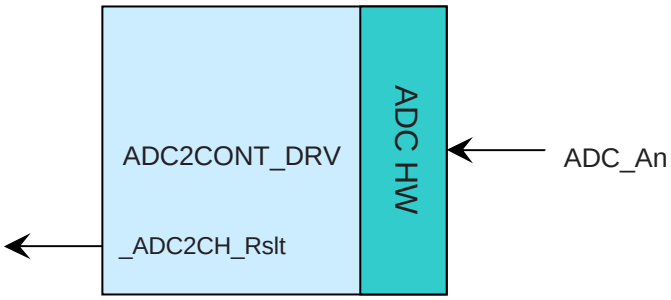


*Figure 2. Connecting the one channel ADC driver*

Step1. Call the driver configuration function in C (this is one-time pass through code)

```
ADC2CONTCnf(CHANNEL_NUMBER, ACQ_PRESCALER_SETTING);
```

Step2. Instantiate the INIT macro in assembly (this is one-time pass through code)

```
; Instantiate the init macro

     ADC2CH_DRV_INIT
```

Step3. Instantiate the run time macro in assembly (this is usually looped or ISR code)

```
; “call” the main macro

     ADC2CH_DRV
```

Step4. (optional) Declare “Signal Nets” to “connect” the module to in “C”

```
int16     AdcValue[2];

; Note AdcValue can be simply an global integer array variable
```

Step5. Declare the module “Terminal pointers” in “C”

```
// ADC2CH_DRV terminal pointers, external references

extern int16     *ADC2CH_Rslt;
```

Step6. “Connect” the module terminals to the Signal Nets in “C”.

```
// ADC2CH_DRV connections
     ADC2CH_Rslt = &AdcValue[0];

// Note this can be done once during init, or dynamically during
// run time operation, i.e. module connections can be
// re-configured to other Nets as required by the application.
```