

Table of contents

1	Overview.....	2
2	Module Properties.....	2
3	Module Input and Output Definitions.....	3
3.1	Module inputs	3
3.2	Module outputs	3
4	Module API Description.....	4
4.1	ADC1CONTConf	4
4.2	ADC1CH_DRV_INIT	5
4.3	ADC1CH_DRV	5
5	Usage Example:.....	7
6	Detailed description.....	Error! Bookmark not defined.

Table of Figures

Figure 1.	Continuous conversion, one channel ADC Driver	2
Figure 2.	Connecting the one channel ADC driver.....	7

Index of Tables

Table 1.	ADC1CONT_DRV module dependencies.....	2
Table 2.	ADC1CONT_DRV module components	2
Table 3.	ADC1CONT_DRV module miscellaneous properties.....	3
Table 4.	ADC1CONT_DRV module component files.....	3
Table 5.	ADC1CONT_DRV module inputs.....	3
Table 6.	ADC1CONT_DRV module outputs.....	3
Table 7.	ADC1CONT_DRV channel selection.....	4

1 Overview

This software module controls the ADC on the F/C280x devices. It sets up the ADC in continuous conversion mode, to convert one channel, and to make it available in a variable in the application. The conversion result is unipolar.

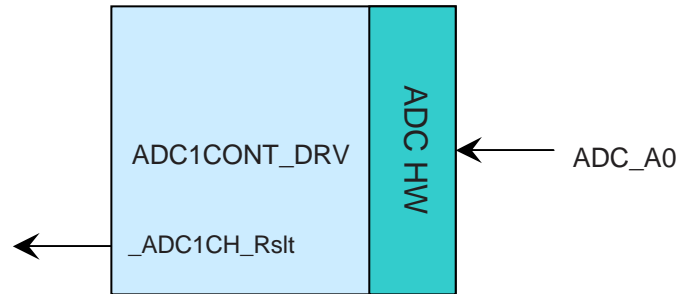


Figure 1. Continuous conversion, one channel ADC Driver

2 Module Properties

This section describes module properties, such as compatible devices, components, invocation etc. The ADC1CONT_DRV module has the following dependencies:

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

Table 1. ADC1CONT_DRV module dependencies

The ADC1CONT_DRV module has the following components:

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

Table 2. ADC1CONT_DRV module components

The ADC1CONT_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. ADC1CONT_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. ADC1CONT_DRV module component files

3 Module Input and Output Definitions

3.1 Module inputs

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

Table 5. ADC1CONT_DRV module inputs

3.2 Module outputs

Output name	Description	Data Format	Output data Range
ADC1CH_Rslt	Driver output pointer	Pointer to 16-bit fixed point output data	Q15: [-1, 1] or [-32768, 32767]

Table 6. ADC1CONT_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 ADC1CONTConf

Function Name: ADC1CONTConf

Prototype: void ADC1CONTConf(int nChannel, int AcqWidth);

Return value: None.

Preconditions: The following preconditions must be satisfied:

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

The ADC1CONTConf function is called from the C environment, and performs driver configuration, including configuration of ADC registers.

❑ **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. ADC1CONT_CONF channel selection

Example: Call the ADC1CONTConf function to initialize EPWM1 module.

```
//-----  
// Set up conversion of channel 0, with acq prescaler at 2 clocks  
//-----  
ADC1CONTConf(0, 2);
```

4.2 ADC1CH_DRV_INIT

Function Name: ADC1CH_DRV_INIT

Prototype: ADC1CH_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 ADC1CH_DRV_INIT to initialize ADC1CH_DRV module.

```
;-----  
; ISR Initialisation  
;-----  
_ISR_Init:  ADC1CH_DRV_INIT  
            LRETR
```

4.3 ADC1CH_DRV

Function Name: ADC1CH_DRV

Prototype: ADC1CH_DRV

Return value: None.

Preconditions: The following preconditions must be satisfied:

- The ADC module clock must be enabled in the PCLKCR0 register.
- C language init routine must be called.
- The ISR initialization macro ADC1CH_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 hardware output. The output is written to the ADC1CH_Rslt variable.

Example: Call the ADC1CH_DRV in an assembler ISR

```
;-----  
; Runtime interrupt service routine  
;-----  
_ISR_Run:    CONTEXT_SAVE                ;call macro  
            ADC1CH_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:

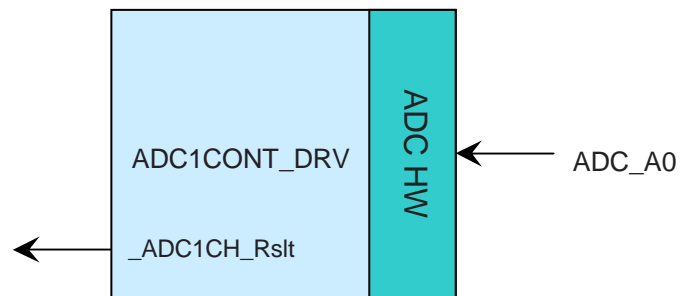


Figure 2. Connecting the one channel ADC driver

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

```
ADC1CONTConf(CHANNEL_NUMBER, ACQ_PRESCALER_SETTING);
```

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

```
; Instantiate the init macro
```

```
ADC1CH_DRV_INIT
```

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

```
; "call" the main macro
```

```
ADC1CH_DRV
```

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

```
int16      AdcValue;
```

```
; Note AdcValue can be simply a global integer variable
```

Step5. Declare the module "Terminal pointers" in "C"

```
// ADC1CH_DRV terminal pointers, external references
```

```
extern int16      * ADC1CH_Rslt;
```

Step6. "Connect" the module terminals to the Signal Nets in "C".

```
// ADC1CH_DRV connections
```

```
ADC1CH_Rslt = &AdcValue;
```

```
// 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.
```