Table of contents

Table of Figures

Index of Tables

1 Overview

This module implements a 2 pole – 2 zero controller, by implementing the second order difference equation shown in equation (2) below.

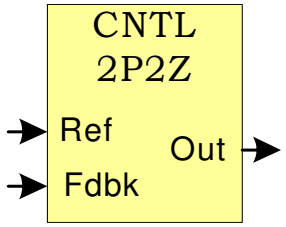


Figure 1. Two pole two zero controller module

$$\frac{U(z)}{E(z)} = \frac{B_2 z^2 + B_1 z + B_0}{-A_2 z^2 - A_1 z + 1} \quad \text{.....(1)}$$

$$U(n) = A_1 \cdot U(n-1) + A_2 \cdot U(n-2) + B_0 \cdot E(n) + B_1 \cdot E(n-1) + B_2 \cdot E(n-2) \quad \text{.....(2)}$$

2 Module Properties

This section describes module properties, such as compatible devices, components, invocation etc. The CNTL_2P2Z module has the following dependencies:

Module	Dependency
CPU dependency	C28x
Device dependency	None (as long as the CPU dependency is satisfied)
Target application	Closed loop control.
Math format (precision)	32 bit fixed Q

Table 1. CNTL_2P2Z module dependencies

The CNTL_2P2Z module has the following components:

Component	Present
C-based initialization	No
ASM interrupt initialization	Yes
ASM runtime macro	Yes

Table 2. CNTL_2P2Z module components

The CNTL_2P2Z module has the following miscellaneous properties:

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

Table 3. CNTL_2P2Z 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. CNTL_2P2Z module component files

3 Module data definitions

3.1 Module inputs

Input name	Description	Data Format	Range
Ref	Reference Set point	Pointer to 16-bit fixed point input data	Q15: [-1, 1] or [-32768, 32767]
Fdbk	Feedback, used to calculate the error term	Pointer to 16-bit fixed point input data	Q15: [-1, 1] or [-32768, 32767]

Table 5. CNTL_2P2Z module inputs

3.2 Module output definitions

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

[†] 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.

Table 6. CNTL_2P2Z module outputs

3.3 Module coefficient configuration

The controller coefficients A1, A2, B0, B1 and B2 in equation (1) above are specified in a record in memory. All coefficients are 32 bits wide, and are in a Q format as described below. The coefficient formats are shown in Table 7 below.

Coefficient	Description	Format	Range
B2	Controller coefficient	32-bit fixed point	Q26: [-32, 31.99999]
B1	Controller coefficient	32-bit fixed point	Q26: [-32, 31.99999]
B0	Controller coefficient	32-bit fixed point	Q26: [-32, 31.99999]
A2	Controller coefficient	32-bit fixed point	Q26: [-32, 31.99999]
A1	Controller coefficient	32-bit fixed point	Q26: [-32, 31.99999]
MAX	Control output upper saturation bound	32-bit fixed point	Q24: [-128, 127.99999]
MIN	Control output lower saturation bound	32-bit fixed point	Q24: [-128, 127.99999]

Table 7. CNTL_2P2Z module coefficient formats

The coefficients must be declared in (for convenience) the same file which instantiates the macro CNTL_2P2Z_INIT. The order of the coefficients is significant, since the CNTL_2P2Z_INIT macro takes the base of this coefficient record as a parameter and all the coefficients are accessed relative to this address. Here is an example of how to set up coefficients. Each coefficient is declared as a '.long' which creates a 32-bit constant.

```
                        .sect   "CNTL_coeff"
DCDC_VLOOP_COEFF1:
                        .long 141465485   ; B2
                        .long -308700774  ; B1
                        .long 167772160   ; B0
                        .long -5368709    ; A2
                        .long 72477573    ; A1
                        .long 0x00FFFFFF  ; MAX
                        .long 0x00000000  ; MIN
```

4 Module API Description

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

4.1 CNTL_2P2Z_INIT

Function Name:	CNTL_2P2Z_INIT
Prototype:	CNTL_2P2Z_INIT nInstance, address
Return value:	None
Preconditions:	None

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.

- nInstance**:** Specifies the instance number of the current instance.
  - **Valid Range:** Limited only by available memory in the application.

Example: Call the CNTL_2P2Z_INIT to initalize the two pole, two zero controller module.

```
;-----------------------------------------------------------------
; ISR Initialisation
;-----------------------------------------------------------------
_ISR_Init:  . . . . . .  ; Other init routines go in here

            CNTL_2P2Z_INIT    1, DCDC_VLOOP_COEFF1

            . . . . . .  ; Other init routines go in here

            LRETR
```

4.2 CNTL_2P2Z

Function Name: CNTL_2P2Z

Prototype: CNTL_2P2Z nInstance

Return value: None.

Preconditions: The following preconditions must be satisfied:

The ISR initialization macro CNTL_2P2Z_INIT must be instanced in an assembler initialization routine, and must run prior to this instance of the controller routine.

This function is the assembler run time macro, and this creates code that runs the run time computation for the two pole – two zero controller. This computes the control ooutput dependeing on the

- nInstance**:** Specifies which controller instance is computed.
  - **Valid Range:** Limited only by available memory in the application.

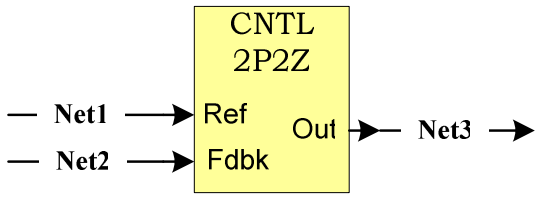
Example: Call the CNTL_2P2Z in an assembler ISR

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


            CNTL_2P2Z   1
;-------------------------------------------------------
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:



Step1. Instantiate the INIT macro in assembly (this is one-time pass through code)

```
PFC_ILOOP_COEFF:
                    .long     -8760341        ; B2 Q26
                    .long     3265008         ; B1 Q26
                    .long     12025350        ; B0 Q26
                    .long     14900244        ; A2 Q26
                    .long     52208619        ; A1 Q26
                    .long     0x00FFFFFF      ; MAX Q24
                    .long     0x00000000      ; MIN Q24

; “call” the 1st instantiation of the init macro

     CNTL_2P2Z_INIT      1, PFC_ILOOP_COEFF
```

Step2. Instantiate the run time macro in assembly (this is usually looped or ISR code)

```
; “call” the main macro

     CNTL_2P2Z      1
```

Step3. (optional) Declare “Signal Nets” to “connect” the module to in “C”

```
int  Net1, Net2, Net3;
```

Step4. Declare the module “Terminal pointers” in “C”

```
// CNTL_2P2Z terminal external references for 1st instantiation

extern int  *CNTL_2P2Z_Ref1, *CNTL_2P2Z_Fdbk1, *CNTL_2P2Z_Out1;
```

Step5. “Connect” the module terminals to the Signal Nets in “C”.

```
// CNTL_2P2Z(1) connections

     CNTL_2P2Z_Ref1 = &Net1;
     CNTL_2P2Z_Out1 = &Net2;
     CNTL_2P2Z_Fdbk1 = &Net3;

// 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.
```