Table of contents

1 Overview...............................................................................................................................................2
2 Module Properties ................................................................................................................................2
3 Module data definitions ........................................................................................................................3
  3.1 Module inputs ...................................................................................................................................3
  3.2 Module output definitions .................................................................................................................3
  3.3 Module coefficient configuration ......................................................................................................4
4 Module API Description........................................................................................................................5
  4.1 CNTL_PI_INIT..................................................................................................................................5
  4.2 CNTL_PI ..........................................................................................................................................6
5 Usage Example: ...................................................................................................................................7

Table of Figures

Figure 1. PI controller module.......................................................................................................................2

Index of Tables

Table 1. CNTL_PI module dependencies .....................................................................................................2
Table 2. CNTL_PI module components .......................................................................................................2
Table 3. CNTL_PI module miscellaneous properties ...................................................................................3
Table 4. CNTL_PI module component files ................................................................................................3
Table 5. CNTL_PI module inputs ................................................................................................................3
Table 6. CNTL_PI module outputs ............................................................................................................3
Table 7. CNTL_PI module coefficient formats ............................................................................................4
1 Overview

This module implements a PI controller, by implementing the following difference equations.

\[ u(n) = u_p(n) + u_i(n) \] .................................(1)

where the proportional output is \( u_p(n) = K_p(\text{Ref}(n) - \text{Fdbk}(n)) \) ..............................................(2)

the integral output is \( u_i(n) = u_i(n-1) + K_i u_p \) ...................................................(3)

2 Module Properties

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU dependency</td>
<td>C28x</td>
</tr>
<tr>
<td>Device dependency</td>
<td>None (as long as the CPU dependency is satisfied)</td>
</tr>
<tr>
<td>Target application</td>
<td>Closed loop control.</td>
</tr>
<tr>
<td>Math format (precision)</td>
<td>32 bit fixed Q</td>
</tr>
</tbody>
</table>

Table 1. CNTL_PI module dependencies

The CNTL_PI module has the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-based initialization</td>
<td>No</td>
</tr>
<tr>
<td>ASM interrupt initialization</td>
<td>Yes</td>
</tr>
<tr>
<td>ASM runtime macro</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2. CNTL_PI module components
The CNTL_PI module has the following miscellaneous properties:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple instance support</td>
<td>Yes</td>
</tr>
<tr>
<td>Reentrant</td>
<td>No</td>
</tr>
<tr>
<td>Accessible from ‘C’ environment</td>
<td>Yes</td>
</tr>
<tr>
<td>Full configuration from ‘C’ environment</td>
<td>Yes</td>
</tr>
<tr>
<td>Input / Output connection</td>
<td>Pointer to signal net.</td>
</tr>
</tbody>
</table>

*Table 3. CNTL_PI module miscellaneous properties*

**Component files**

C:\tidcs\DPS_C280x\Vxyz\lib\dpslib280x\include\ControlLawMacro.h†

*Table 4. CNTL_PI module component files*

3 Module data definitions

3.1 Module inputs

<table>
<thead>
<tr>
<th>Input name</th>
<th>Description</th>
<th>Data Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref</td>
<td>Reference Set point</td>
<td>Pointer to 16-bit fixed point input data</td>
<td>Q15: [-1, 1] or [-32768, 32767]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fdbk</td>
<td>Feedback, used to calculate the error term</td>
<td>Pointer to 16-bit fixed point input data</td>
<td>Q15: [-1, 1] or [-32768, 32767]</td>
</tr>
</tbody>
</table>

*Table 5. CNTL_PI module inputs*

3.2 Module output definitions

<table>
<thead>
<tr>
<th>Output name</th>
<th>Description</th>
<th>Data Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out</td>
<td>Controller output</td>
<td>Pointer to 16-bit fixed point output data</td>
<td>Q15: [-1, 1] or [-32768, 32767]</td>
</tr>
</tbody>
</table>

*Table 6. CNTL_PI 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.
3.3 Module coefficient configuration

The controller coefficients $K_p$ and $K_i$ in equations (2) and (3) 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.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Description</th>
<th>Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>Controller coefficient</td>
<td>32-bit fixed point</td>
<td>$Q_{26}$: $[-32, 31.99999]$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Controller coefficient</td>
<td>32-bit fixed point</td>
<td>$Q_{26}$: $[-32, 31.99999]$</td>
</tr>
<tr>
<td>MAX</td>
<td>Control output upper</td>
<td>32-bit fixed point</td>
<td>$Q_{24}$: $[-128, 127.99999]$</td>
</tr>
<tr>
<td></td>
<td>saturation bound</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Control output lower</td>
<td>32-bit fixed point</td>
<td>$Q_{24}$: $[-128, 127.99999]$</td>
</tr>
<tr>
<td></td>
<td>saturation bound</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. CNTL_PI module coefficient formats

The coefficients must be declared in (for convenience) the same file which instantiates the macro CNTL_PI_INIT. The order of the coefficients is significant, since the CNTL_PI_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"
.DC DCDC_VLOOP_PI_COEFF1:
   .long 0x03000000 ; Kp
   .long 0x06100000 ; Ki
   .long 0x00FFFFFF ; MAX
   .long 0xFF000000 ; 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_PI_INIT

<table>
<thead>
<tr>
<th>Function Name:</th>
<th>CNTL_PI_INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype:</td>
<td>CNTL_PI_INIT nInstance, address</td>
</tr>
<tr>
<td>Return value:</td>
<td>None</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>None</td>
</tr>
</tbody>
</table>

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_PI_INIT to initialize the PI controller module.

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

CNTL_PI_INIT 1, DCDC_VLOOP_PI_COEFF1
........ ; Other init routines go in here

LRETR
```
4.2 CNTL_PI

Function Name: CNTL_PI

Prototype: CNTL_PI nInstance

Return value: None.

Preconditions: The following preconditions must be satisfied:

The ISR initialization macro CNTL_PI_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 PI controller. This computes the control output depending on the

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

Example: Call the CNTL_PI in an assembler ISR

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

  CNTL_PI  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:

Step 1. Instantiate the INIT macro in assembly (this is one-time pass through code)

```assembly
DCDC_VLOOP_PI_COEFF1:
  .long 0x03000000 ; Kp (Q26)
  .long 0x06100000 ; Ki (Q26)
  .long 0x00FFFFFF ; MAX (Q24)
  .long 0xFF000000 ; MIN (Q24)

; "call" the 1st instantiation of the init macro
CNTL_PI_INIT 1, DCDC_VLOOP_PI_COEFF1
```

Step 2. Instantiate the run time macro in assembly (this is usually looped or ISR code)

; "call" the main macro

```assembly
CNTL_PI 1
```

Step 3. (optional) Declare “Signal Nets” to “connect” the module to in “C”

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

Step 4. Declare the module “Terminal pointers” in “C”

```c
// CNTL_PI terminal external references for 1st instantiation
extern int *CNTL_PI_Ref1, *CNTL_PI_Fdbk1, *CNTL_PI_Out1;
```

Step 5. “Connect” the module terminals to the Signal Nets in “C”.

```c
// CNTL_PI(1) connections
CNTL_PI_Ref1 = &Net1;
CNTL_PI_Out1 = &Net2;
CNTL_PI_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.
```