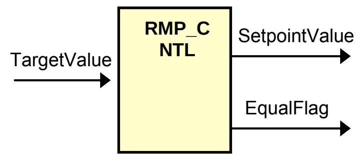
RMP_CNTL	***Ramp Control***

Description

This module implements a ramp up and ramp down function. The output flag variable EqualFlag is set to 7FFFFFFFh when the output variable SetpointValue equals the input variable TargetValue.



Availability

This IQ module is available in one interface format:

1) The C interface version

Module Properties

Type: Target Independent, Application Independent

Target Devices: 28x Fixed Point or Piccolo

C Version File Names: rmp_cntl.c, rmp_cntl.h

IQmath library files for C: IQmathLib.h, IQmath.lib

Item	C version	Comments
Code Size□	55/55 words	
Data RAM	0 words•	
xDAIS ready	No	
XDAIS component	No	IALG layer not implemented
Multiple instances	Yes	
Reentrancy	Yes	

• Each pre-initialized "_iq" RMPCNTL structure consumes 16 words in the data memory

□ Code size mentioned here is the size of the ***calc()*** function

C Interface

Object Definition

The structure of RMPCNTL object is defined by following structure definition

```
typedef struct { _iq  TargetValue;              // Input: Target input
                 Uint32  RampDelayMax;         // Parameter: Maximum delay rate (Q0)
                  _iq  RampLowLimit;           // Parameter: Minimum limit
                  _iq  RampHighLimit;          // Parameter: Maximum limit
                 Uint32 RampDelayCount;        // Variable: Incremental delay (Q0)
                  _iq  SetpointValue;          // Output: Target output
                 Uint32  EqualFlag;            // Output: Flag output (Q0)
                 void  (*calc)();              // Pointer to calculation function
               } RMPCNTL;

typedef RMPCNTL *RMPCNTL_handle;
```

Item	Name	Description	Format*	Range(Hex)
Inputs	TargetValue	Target input	GLOBAL_Q	80000000-7FFFFFFF
Outputs	SetpointValue	Target output	GLOBAL_Q	80000000-7FFFFFFF
	EqualFlag	Flag output	Q0	80000000-7FFFFFFF
RMP_CNTL parameter	RampDelayMax	Maximum delay rate	Q0	80000000-7FFFFFFF
	RampLowLimit	Minimum limit	GLOBAL_Q	80000000-7FFFFFFF
	RampHighLimit	Maximum limit	GLOBAL_Q	80000000-7FFFFFFF
Internal	RampDelayCount	Incremental delay	Q0	80000000-7FFFFFFF

*GLOBAL_Q valued between 1 and 30 is defined in the IQmathLib.h header file.

Special Constants and Data types

RMPCNTL

The module definition is created as a data type. This makes it convenient to instance an interface to ramp control. To create multiple instances of the module simply declare variables of type RAMPGEN.

RMPCNTL_handle

User defined Data type of pointer to RMPCNTL module

RMPCNTL_DEFAULTS

Structure symbolic constant to initialize RMPCNTL module. This provides the initial values to the terminal variables as well as method pointers.

Methods

void rmp_cntl_calc(RMPCNTL_handle);

This definition implements one method viz., the ramp control computation function. The input argument to this function is the module handle.

Module Usage

Instantiation
The following example instances two RMPCNTL objects
RMPCNTL rc1, rc2;

Initialization
To Instance pre-initialized objects
RMPCNTL rc1 = RMPCNTL_DEFAULTS;
RMPCNTL rc2 = RMPCNTL_DEFAULTS;

Invoking the computation function
rc1.calc(&rc1);
rc2.calc(&rc2);

Example

The following pseudo code provides the information about the module usage.

```
main()
{

}

void interrupt periodic_interrupt_isr()
{
    rc1.TargetValue = target1;          // Pass inputs to rc1
    rc2.TargetValue = target2;          // Pass inputs to rc2

    rc1.calc(&rc1);                     // Call compute function for rc1
    rc2.calc(&rc2);                     // Call compute function for rc2

    out1 = rc1.SetpointValue;           // Access the outputs of rc1
    out2 = rc2.SetpointValue;           // Access the outputs of rc2
}
```

Technical Background

This software module implements the following equations:

<u>Case 1:</u> When *TargetValue* > *SetpointValue*

SetpointValue = *SetpointValue*+ _IQ(0.0000305), for t = n . Td, n = 1, 2, 3…
and (*SetpointValue* + _IQ(0.0000305))< *RampHighLimit*
= *RampHighLimit* , for (*SetpointValue* + _IQ(0.0000305))> *RampHighLimit*

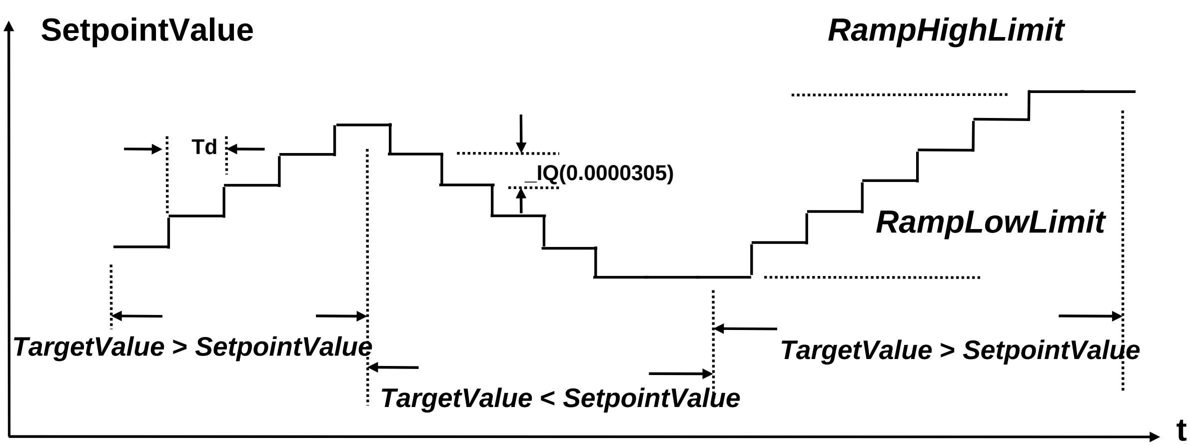
where, Td = *RampDelayMax* . Ts
Ts = Sampling time period

<u>Case 2:</u> When *TargetValue* < *SetpointValue*

SetpointValue = *SetpointValue* - _IQ(0.0000305), for t = n . Td, n = 1, 2, 3…..
and (*SetpointValue* - _IQ(0.0000305))> *RampLowLimit*
= *RampLowLimit* , for (*SetpointValue* - _IQ(0.0000305))<*RampLowLimit*

where, Td = *RampDelayMax* . Ts
Ts = Sampling time period

<u>Note that</u> *TargetValue* and *SetpointValue* variables are in _iq format.



<u>Example</u>:
SetpointValue=0(initial value), TargetValue=1000(user specified),
RampDelayMax=500(user specified), sampling loop time period Ts=0.000025 Sec.
This means that the time delay for each ramp step is Td=500x0.000025=0.0125 Sec.
Therefore, the total ramp time will be Tramp=1000x0.0125 Sec=12.5 Sec