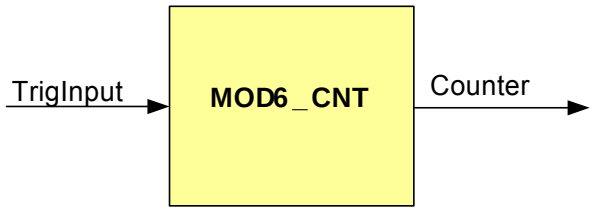
MOD6_CNT	*Modulo 6 Counter Module*

Description

This module implements a modulo 6 counter. It counts from state 0 through 5, then resets to 0 and repeats the process. The state of the output variable *Counter* changes to the next state every time it receives a trigger input through the input variable *TrigInput.*

.



Availability

This IQ module is available in one interface format:

1) The C interface version

Module Properties

Type: Target Independent, Application Independent

Target Devices: x281x or x280x

C Version File Names: mod6_cnt.c, mod6_cnt.h

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

Item	C version	Comments
Code Size□ (x281x/x280x)	12/12 words	
Data RAM	0 words•	
xDAIS ready	No	
XDAIS component	No	IALG layer not implemented
Multiple instances	Yes	
Reentrancy	Yes	

• Each pre-initialized "_iq" MOD6CNT structure consumes 6 words in the data memory

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

C Interface

Object Definition

The structure of MOD6CNT object is defined by following structure definition

```
typedef struct { Uint32  TrigInput;      // Input: Modulo 6 counter trigger 0x0000 or 0x7FFF)
                 Uint32  Counter;        // Output: Modulo 6 counter output  (0,1,2,3,4,5)
                 void  (*calc)();        // Pointer to calculation function
               } MOD6CNT;

typedef MOD6CNT *MOD6CNT_handle;
```

Item	Name	Description	Format*	Range(Hex)
Input	TrigInput	Modulo 6 counter trigger	Q0	0 or 7FFF
Outputs	Counter	Modulo 6 counter output	Q0	0,1,2,3,4,5

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

Special Constants and Data types

MOD6CNT
The module definition is created as a data type. This makes it convenient to instance an interface to the modulo 6 counter. To create multiple instances of the module simply declare variables of type MOD6CNT.

MOD6CNT_handle
User defined Data type of pointer to MOD6CNT module

MOD6CNT_DEFAULTS
Structure symbolic constant to initialize MOD6CNT module. This provides the initial values to the terminal variables as well as method pointers.

Methods

void mod6cnt_calc(MOD6CNT_handle);

This definition implements one method viz., the modulo 6 counter computation function. The input argument to this function is the module handle.

Module Usage

Instantiation
The following example instances two MOD6CNT objects
MOD6CNT mod1, mod2;

Initialization
To Instance pre-initialized objects
MOD6CNT mod1 = MOD6CNT_DEFAULTS;
MOD6CNT mod2 = MOD6CNT_DEFAULTS;

Invoking the computation function

```
mod1.calc(&mod1);
mod2.calc(&mod2);
```

Example

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

```
main()
{

}

void interrupt periodic_interrupt_isr()
{
        mod1.TrigInput = input1;            // Pass inputs to mod1
        mod2.TrigInput = input2;            // Pass inputs to mod2

        mod1.calc(&mod1);                   // Call compute function for mod1
        mod2.calc(&mod2);                   // Call compute function for mod2

        out1 = mod1.Counter;                // Access the outputs of mod1
        out2 = mod2.Counter;                // Access the outputs of mod2

}
```

Technical Background

Counter = 0, when 1st trigger pulse occur (*TrigInput* is set to 0x7FFF for the 1st time)
= 1, when 2nd trigger pulse occur (*TrigInput* is set to 0x7FFF for the 2nd time)
= 2, when 3rd trigger pulse occur (*TrigInput* is set to 0x7FFF for the 3rd time)
= 3, when 4th trigger pulse occur (*TrigInput* is set to 0x7FFF for the 4th time)
= 4, when 5th trigger pulse occur (*TrigInput* is set to 0x7FFF for the 5th time)
= 5, when 6th trigger pulse occur (*TrigInput* is set to 0x7FFF for the 6th time)
and repeats the output states for the subsequent pulses.