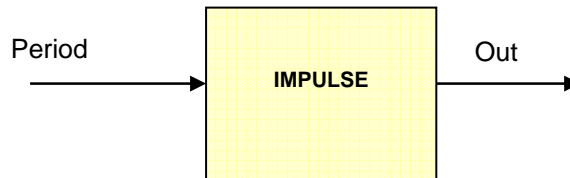


Description

This module implements a periodic impulse macro. The output variable *Out* is set to 0x00007FFF for 1 sampling period. The period of the output signal *Out* is specified by the input *Period*.

**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: impulse.h

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

C Interface

Object Definition

The structure of IMPULSE object is defined by following structure definition

```
typedef struct { Uint32 Period;    // Input: Period of output in # of sampling cycles (Q0)
                Uint32 Out;      // Output: Impulse output (0x00000000 or 0x00007FFF)
                Uint32 Counter;  // Variable: Impulse generator counter (Q0)
            } IMPULSE;
```

```
typedef IMPULSE *IMPULSE_handle;
```

Item	Name	Description	Format	Range(Hex)
Input	Period	Period of output in # of sampling period	Q0	00000000-7FFFFFFF
Output	Out	Impulse output	Q0	0 or 00007FFF
Internal	Counter	Impulse generator counter	Q0	00000000-7FFFFFFF

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

Special Constants and Data types

IMPULSE

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

IMPULSE_handle

User defined Data type of pointer to IMPULSE module

IMPULSE_DEFAULTS

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

Methods

IMPULSE_MACRO (IMPULSE_handle);

This definition implements one method viz., the impulse generator computation macro. The input argument to this macro is the module handle.

Module Usage

Instantiation

The following example instances two IMPULSE objects
IMPULSE ig1, ig2;

Initialization

To Instance pre-initialized objects
IMPULSE ig1 = IMPULSE_DEFAULTS;
IMPULSE ig2 = IMPULSE_DEFAULTS;

Invoking the computation macro

```
IMPULSE_MACRO (ig1);  
IMPULSE_MACRO (ig2);
```

Example

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

```
main()  
{  
  
}  
  
void interrupt periodic_interrupt_isr()  
{  
    ig1.Period = input1;           // Pass inputs to ig1  
    ig2.Period = input2;           // Pass inputs to ig2  
  
    IMPULSE_MACRO (ig1);           // Call compute macro for ig1  
    IMPULSE_MACRO (ig2);           // Call compute macro for ig2  
  
    out1 = ig1.Out;                // Access the outputs of ig1  
    out2 = ig2.Out;                // Access the outputs of ig2  
  
}
```

Technical Background

Implements the following equation:

$$\begin{aligned} Out &= 0x00007FFF, \text{ for } t = n \cdot Tout, n = 1, 2, 3, \dots \\ &= 0, \text{ otherwise} \end{aligned}$$

where,

$Tout$ = Time period of output pulses = $Period \times Ts$

Ts = Sampling time period

