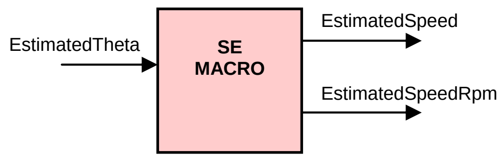
ESTIMATED SPEED	***Speed Calculator Based on Rotor Angle Without Direction Information***

Description This module calculates the motor speed based on the estimated rotor position when the rotational direction information is not available.



Availability This IQ module is available in one interface format:

1) The C interface version

Module Properties **Type:** Target Independent, Application Dependent

Target Devices: 28x Fixed Point or Piccolo

C Version File Names: speed_est.h

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

C Interface

Object Definition

The structure of SPEED_ESTIMATION object is defined by following structure definition

```
typedef struct { _iq EstimatedTheta;            // Input: Electrical angle
                 _iq OldEstimatedTheta;         // History: Electrical angle at previous step
                 _iq EstimatedSpeed;            // Output: Estimated speed in per-unit
                 Uint32 BaseRpm;                // Parameter: Base speed in rpm (Q0)
                 _iq21 K1;                      // Parameter: Constant for differentiator (Q21)
                 _iq K2;                        // Parameter: Constant for low-pass filter
                 _iq K3;                        // Parameter: Constant for low-pass filter
                 int32 EstimatedSpeedRpm;       // Output : Estimated speed in rpm  (Q0)
               } SPEED_ESTIMATION;              // Data type created


typedef SPEED_ESTIMATION *SPEED_ESTIMATION_handle;
```

Item	Name	Description	Format*	Range(Hex)
Inputs	EstimatedTheta	Electrical angle	GLOBAL_Q	00000000-7FFFFFFF (0 – 360 degree)
Outputs	EstimatedSpeed	Computed speed in per-unit	GLOBAL_Q	80000000-7FFFFFFF
	EstimatedSpeedRpm	Speed in rpm	Q0	80000000-7FFFFFFF
ESTIMATED SPEED parameter	K1	K1 = 1/(fb*T)	Q21	80000000-7FFFFFFF
	K2	K2 = 1/(1+T*2*pi*fc)	GLOBAL_Q	80000000-7FFFFFFF
	K3	K3 = T*2*pi*fc/(1+T*2*pi*fc)	GLOBAL_Q	80000000-7FFFFFFF
	BaseRpm	BaseRpm = 120fb/p	Q0	80000000-7FFFFFFF
Internal	OldEstimatedTheta	Electrical angle in previous step	GLOBAL_Q	00000000-7FFFFFFF (0 – 360 degree)

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

Special Constants and Data types

SPEED_ESTIMATION

The module definition is created as a data type. This makes it convenient to instance an interface to speed calculation based on measured rotor angle. To create multiple instances of the module simply declare variables of type SPEED_ESTIMATION.

SPEED_ESTIMATION_handle

User defined Data type of pointer to SPEED_ESTIMATION module

SPEED_ESTIMATION_DEFAULTS

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

Methods

SE_MACRO (SPEED_ESTIMATION_handle);

This definition implements one method viz., the speed calculation based on measured rotor angle computation macro. The input argument to this macro is the module handle.

Module Usage

Instantiation
The following example instances two SPEED_ESTIMATION objects
SPEED_ESTIMATION speed1, speed2;

Initialization
To Instance pre-initialized objects
SPEED_ESTIMATION speed1 = SPEED_ESTIMATION_DEFAULTS;
SPEED_ESTIMATION speed2 = SPEED_ESTIMATION_DEFAULTS;

Invoking the computation macro
SE_MACRO(speed1);
SE_MACRO(speed2);

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

```
main()
{

}


void interrupt periodic_interrupt_isr()
{
        speed1.EstimatedTheta = theta1;                  // Pass inputs to speed1
        speed2.EstimatedTheta = theta2;                  // Pass inputs to speed2

        SE_MACRO (speed1);                               // Call compute macro for speed1
        SE_MACRO (speed2);                               // Call compute macro for speed2

        measured_spd1 = speed1.EstimatedSpeed;           // Access the outputs of speed1
        measured_spd2 = speed2.EstimatedSpeed;           // Access the outputs of speed2
}
```

Technical Background

The typical waveforms of the electrical rotor position angle, θ_e, in both directions can be seen in Figure 1. Assuming the direction of rotation is not available. To take care the discontinuity of angle from 360^o to 0^o (CCW) or from 0^o to 360^o (CW), the differentiator is simply operated only within the differentiable range as seen in this Figure. This differentiable range does not significantly lose the information to compute the estimated speed.

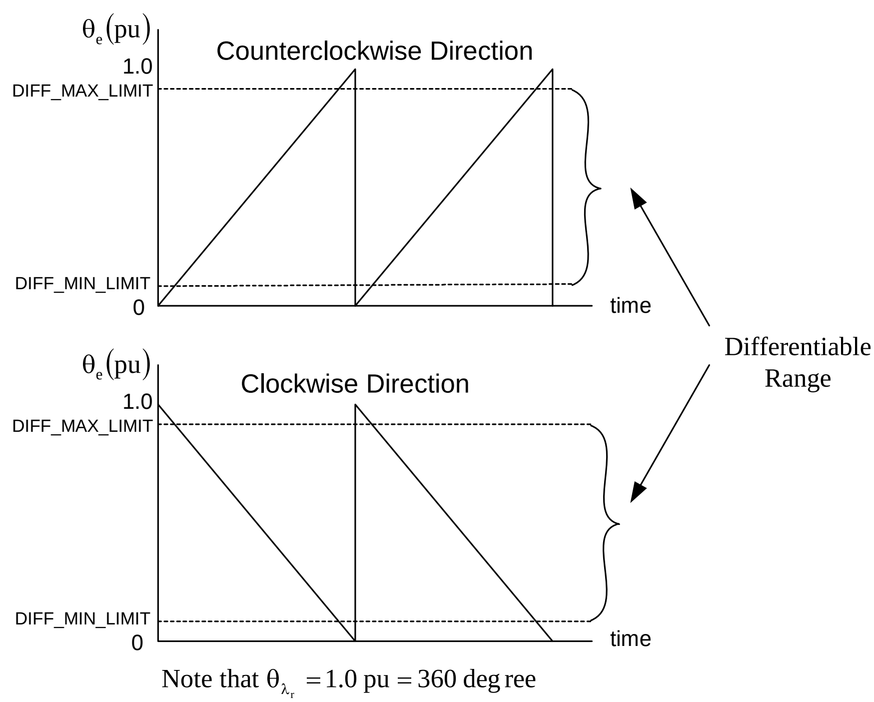


Figure 1: The waveforms of rotor position in both directions

The differentiator equation of rotor position can be expressed as follows.

$$\omega_e(k) = K_1\left(\theta_e(k) - \theta_e(k-1)\right) \tag{1}$$

where $K_1 = \dfrac{1}{f_b T}$, f_b is base frequency (Hz) and T is sampling period (sec).

In addition, the rotor speed is necessary to be filtered out by the low-pass filter in order to reduce the amplifying noise generated by the pure differentiator. The simple 1st-order low-pass filter is used, then the actual rotor speed to be used is the output of the low-pass filter, $\hat{\omega}_e$, seen in following equation. The continuous-time equation of 1st-order low-pass filter is as

$$\frac{d\hat{\omega}_e}{dt} = \frac{1}{\tau_c}\left(\omega_e - \hat{\omega}_e\right) \qquad (2)$$

where $\tau_c = \frac{1}{2\pi f_c}$ is the low-pass filter time constant (sec), and f_c is the cut-off frequency (Hz). Using backward approximation, then (2) finally becomes

$$\hat{\omega}_e(k) = K_2\hat{\omega}_e(k-1) + K_3\omega_e(k) \qquad (3)$$

where $K_2 = \frac{\tau_c}{\tau_c + T}$, and $K_3 = \frac{T}{\tau_c + T}$.

Next, Table 1 shows the correspondence of notations between variables used here and variables used in the program (i.e., speed_est.c, speed_est.h). The software module requires that both input and output variables are in per unit values.

	Equation Variables	**Program Variables**
Input	θ_e	EstimatedTheta
Output	$\hat{\omega}_e$	EstimatedSpeed
Others	K1	K1
	K2	K2
	K3	K3

Table 1: Correspondence of notations