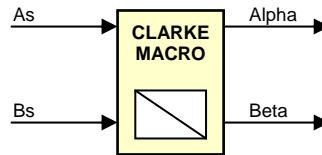


Description

Converts balanced three phase quantities into balanced two phase quadrature quantities.

**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: clarke.h

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

C Interface

Object Definition

The structure of CLARKE object is defined by following structure definition

```
typedef struct { _iq As;           // Input: phase-a stator variable
                _iq Bs;           // Input: phase-b stator variable
                _iq Alpha;        // Output: stationary d-axis stator variable
                _iq Beta;         // Output: stationary q-axis stator variable
        } CLARKE;
```

```
typedef CLARKE *CLARKE_handle;
```

Item	Name	Description	Format	Range(Hex)
Inputs	As	Phase 'a' component of the balanced three phase quantities	GLOBAL_Q	80000000-7FFFFFFF
	Bs	Phase 'b' component of the balanced three phase quantities	GLOBAL_Q	80000000-7FFFFFFF
Outputs	Alpha	Direct axis(d) component of the transformed signal	GLOBAL_Q	80000000-7FFFFFFF
	Beta	Quadrature axis(q) component of the transformed signal	GLOBAL_Q	80000000-7FFFFFFF

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

Special Constants and Data types

CLARKE

The module definition is created as a data type. This makes it convenient to instance an interface to the Clarke variable transformation. To create multiple instances of the module simply declare variables of type CLARKE.

CLARKE_handle

User defined Data type of pointer to CLARKE module

CLARKE_DEFAULTS

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

Methods

CLARKE_MACRO(CLARKE_handle);

This definition implements one method viz., the Clarke variable transformation computation macro. The input argument to this macro is the module handle.

Module Usage

Instantiation

The following example instances two CLARKE objects
CLARKE clarke1, clarke2;

Initialization

To Instance pre-initialized objects

```
CLARKE clarke1 = CLARKE_DEFAULTS;
```

```
CLARKE clarke2 = CLARKE_DEFAULTS;
```

Invoking the computation macro

```
CLARKE_MACRO (clarke1);
```

```
CLARKE_MACRO (clarke2);
```

Example

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

```
main()
{
}

void interrupt periodic_interrupt_isr()
{
    clarke1.As = as1;           // Pass inputs to clarke1
    clarke1.Bs = bs1;           // Pass inputs to clarke1

    clarke2.As = as2;           // Pass inputs to clarke2
    clarke2.Bs = bs2;           // Pass inputs to clarke2

    CLARKE_MACRO (clarke1);     // Call compute macro for clarke1
    CLARKE_MACRO (clarke2);     // Call compute macro for clarke2

    ds1 = clarke1.Alpha;        // Access the outputs of clarke1
    qs1 = clarke1.Beta;         // Access the outputs of clarke1

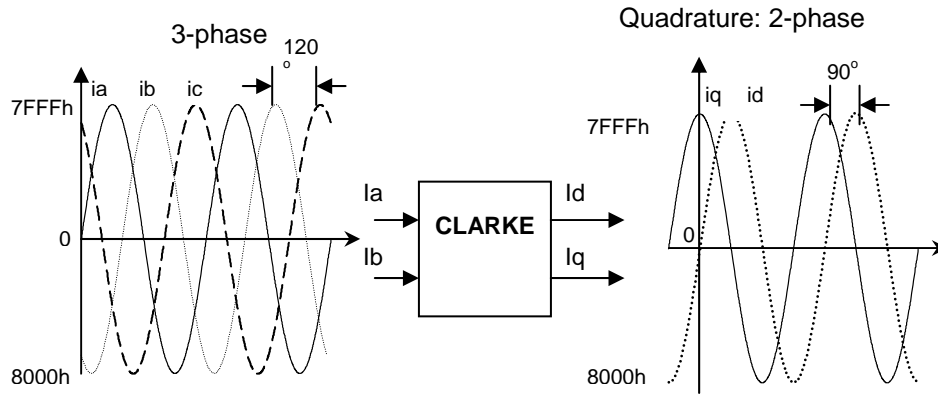
    ds2 = clarke2.Alpha;        // Access the outputs of clarke2
    qs2 = clarke2.Beta;         // Access the outputs of clarke2
}
```

Technical Background

Implements the following equations:

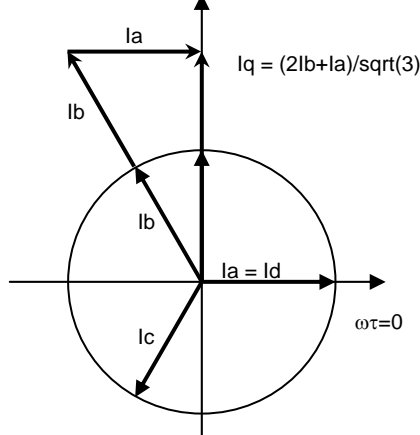
$$\begin{cases} Id = Ia \\ Iq = (2Ib + Ia) / \sqrt{3} \end{cases}$$

This transformation converts balanced three phase quantities into balanced two phase quadrature quantities as shown in figure below:



The instantaneous input and the output quantities are defined by the following equations:

$$\begin{aligned} ia &= I \times \sin(\omega t) \\ ib &= I \times \sin(\omega t + 2\pi / 3) \\ ic &= I \times \sin(\omega t - 2\pi / 3) \\ \begin{cases} id &= I \times \sin(\omega t) \\ iq &= I \times \sin(\omega t + \pi / 2) \end{cases} \end{aligned}$$



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

	Equation Variables	Program Variables
Inputs	ia	As
	ib	Bs
Outputs	id	Alpha
	iq	Beta

Table 1: Correspondence of notations