abc to dq0

SD029 | Posted on June 23, 2021 | Updated on May 27, 2025

Julien ORSINGER
Power Applications Specialist
imperix • in

Table of Contents

The “abc to dq0” block computes the coordinates of a three-phase (*abc*) signal in a rotating reference frame (*dq0*). The angle of the rotating reference frame is given by the second input $\theta = \omega t$.

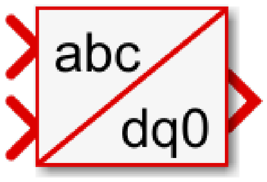
The transformation is performed by applying successively an *abc to alpha-beta-0*, and an *alpha-beta-0 to dq0* transformation:

$$\begin{bmatrix} V_d \\ V_q \\ V_0 \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$

Simulink block

Signal specification

- The first input is a vector of dimension 3, containing the *abc* components of the three-phase signal.
- The second input is the angle θ of the rotating reference frame, in radians.
- The output is a vector of dimension 3, containing the *dq0* components of the three-phase signal in the rotating reference frame.



Parameters

None.

PLECS block

None. The PLECS block *Transformation 3ph->RRF* can be used instead.

C++ functions

The user template located in the installation folder of CPP SDK contains an API folder with implementations of the coordinate transformation functions. The *abc to dq0* function is the following:

```cpp
void abc2DQ0(SpaceVector* rotating, const TimeDomain* physical, const float theta);
```
Code language: C++ (cpp)

Parameters

- `rotating`: pointer on the *dq0* space vector that will be updated. The `SpaceVector` structure is defined below.
- `physical`: pointer on the time domain *abc* data that will be transformed. The `TimeDomain` structure is defined below.
- theta: the angle of the rotating reference frame, in radians.

```cpp
typedef struct{
        float real;       // d-axis component
        float imaginary;  // q-axis component
        float offset;     // homopolar component
} SpaceVector;

typedef struct{
        float A;
        float B;
        float C;
} TimeDomain;
```
Code language: C++ (cpp)