

Variable frequency operation with imperix controllers

PN121 | Posted on March 29, 2021 | Updated on November 20, 2025



Benoît STEINMANN

Software Team Leader

imperix • 

Table of Contents

- [Working principle of variable frequency operation](#)
 - [Remarks](#)
- [Variable frequency with ACG SDK](#)
- [Variable frequency with C++ SDK](#)

This note covers the configuration and implementation of variable frequency operation with imperix controllers (B-Box RCP and B-BoardPRO).

Changing the modulation frequency during the control execution may be useful in applications such as resonant converters or in case of dynamic reconfiguration (start-up of drives for instance). This note shows how the B-Board PRO and B-Box RCP support the real-time tuning of the modulators' switching frequency.

Working principle of variable frequency operation

As a CB-PWM block can be freely mapped to any of the four available CLK blocks, variable-frequency modulation is designed to be implemented by:

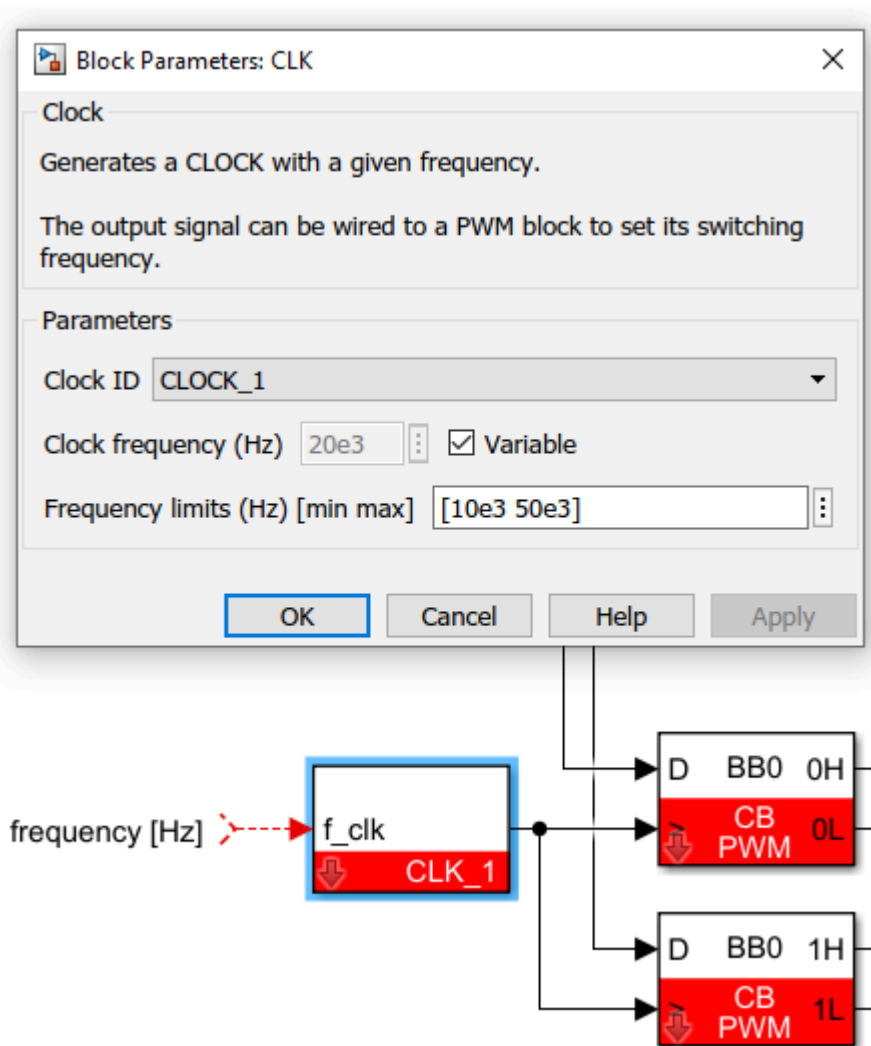
- Mapping the CB-PWM block that must have a variable frequency to a specific CLK.
- Changing the frequency of the corresponding CLK during the execution.

Remarks

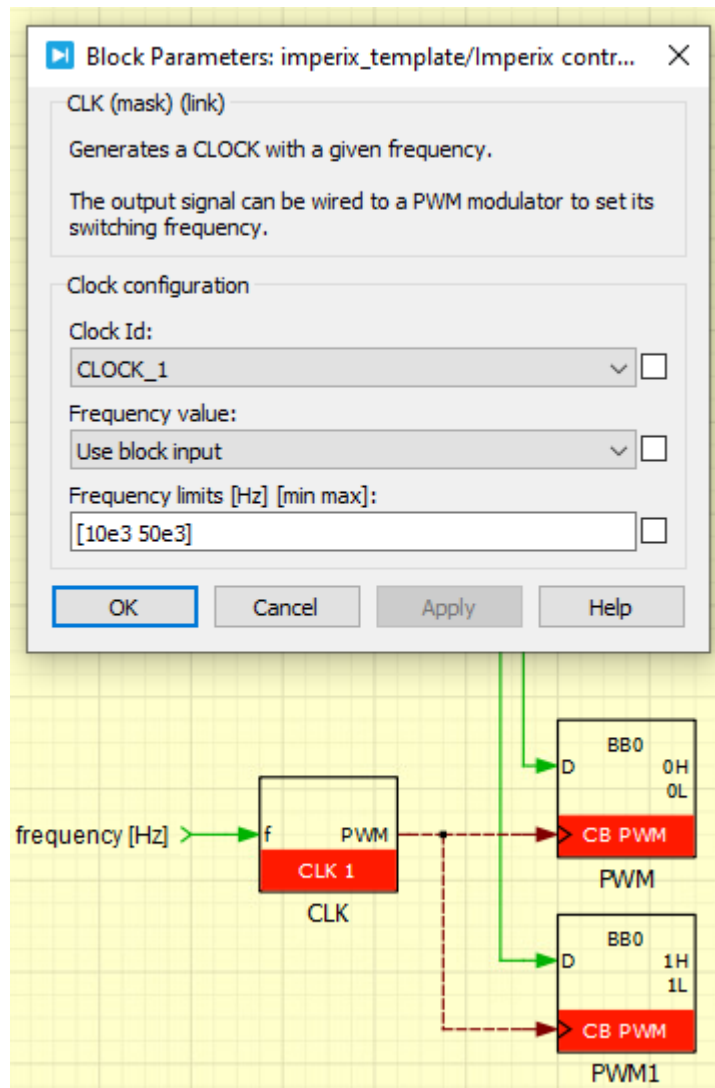
- More than one variable-frequencies can be used simultaneously.
- The interrupt and the sampling processes are linked *by design* to the same CLK (CLK0) and their frequency can not be changed during the execution. Consequently, an additional CLK block, mapped on CLK1, CLK2, or CLK3, must be used for variable frequency operation.
- The CLK blocks are implemented such that frequency changes are glitch-less, i.e. does not generate any unexpected behavior during the transition. Any frequency step can therefore be done at any time during operation.
- A change in the frequency is only applied at reset of the clock source counter. This means that the CLK block completes the running period before changing its frequency.

Variable frequency with ACG SDK

The screenshots below show how to use and configure a [Clock generator \(CLK\)](#) block for variable frequency operation.



Simulink



PLECS

Variable frequency with C++ SDK

As shown in the following code snippet, the frequency change is performed by re-configuring the clock generator `CLOCK_1` during real-time control execution. This example uses the following configuration:

| Clock | Linked to | Configuration |
|----------------------|----------------------------|-------------------------|
| <code>CLOCK_0</code> | Interrupt and sampling | Fixed frequency (50kHz) |
| <code>CLOCK_1</code> | <code>PWM_CHANNEL_0</code> | Variable frequency |

```

tUserSafe UserInit(void)
{
    // The interrupt and sampling use CLOCK_0
    Clock_SetFrequency(CLOCK_0, 50e3);
    ConfigureMainInterrupt(UserInterrupt, CLOCK_0, 0.5);

    // The PWM_CHANNEL_0 uses CLOCK_1 which is set as real-time tunable

```

```

Clock_SetFrequency(CLOCK_1, 10e3);
Clock_ConfigureAsRealTimeTunable(CLOCK_1);

CbPwm_ConfigureClock(PWM_CHANNEL_0, CLOCK_1);
CbPwm_ConfigureOutputMode(PWM_CHANNEL_0, COMPLEMENTARY);
CbPwm_ConfigureCarrier(PWM_CHANNEL_0, TRIANGLE);
CbPwm_ConfigureDeadTime(PWM_CHANNEL_0, 1e-6);
CbPwm_SetDutyCycle(PWM_CHANNEL_0, 0.5);
CbPwm_Activate(PWM_CHANNEL_0);

Adc_ConfigureInput(0, GAIN_I, 0.0);
Adc_ConfigureInput(1, GAIN_V, 0.0);

return SAFE;
}

// Global variables can be observed from Cockpit
float current;
float voltage;
float freq;

tUserSafe UserInterrupt(void)
{
    current = Adc_GetValue(0);
    voltage = Adc_GetValue(1);

    freq = GetOptimizedFrequency(current, voltage);

    Clock_SetFrequency(CLOCK_0, freq);

    return SAFE;
}Code language: C++ (cpp)

```