CLK - Clock generators

SD002 | Posted on April 2, 2021 | Updated on May 27, 2025

Benoît STEINMANN
Software Team Leader
imperix

Table of Contents

The B-Box RCP and B-Board PRO digital controllers hold **4 clock generators,** which provide time-bases to use with various peripherals such as the PWMs and the control task routine. In a multi-device configuration, the clocks are propagated to all the devices and stay synchronized within ±2 ns.

Features:

- **Variable frequency**: the clock generators support glitch-less reconfiguration during execution as explained in Variable frequency operation with the B-Box/B-Board (PN121).
- **Synchronization**: in a multi-device configuration, all clock generators are intrinsically synchronized.
- **Simultaneous reset**: all the clock generators are reset at the same time. It implies that, if the frequency of a clock generator is a multiple of another one, they are guaranteed to stay in phase (e.g. 20kHz and 40 kHz).

Simulink block

A CLK block using *CLOCK_0* is already embedded in the CONFIG block. See CONFIG – Interrupt configuration to learn more.

Signal specification

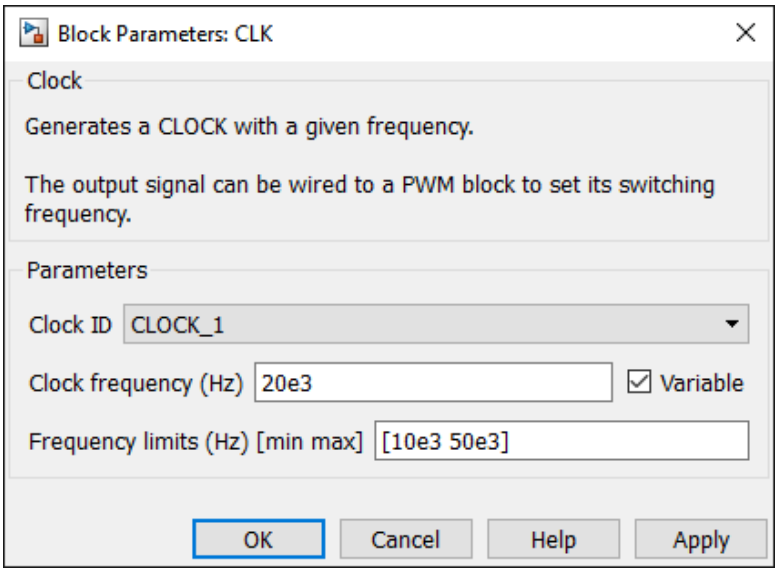
- The output can be wired to a PWM block input signal > to set its switching frequency.
- The input is only visible if the `variable` parameter is selected and it sets the frequency of the clock during real-time execution.

Standard parameters

- `Clock ID` selects which clock generator to configure. (`CLOCK_0` is already used by the CONFIG block and cannot be instantiated in a separate clock block.)
- `Initial clock frequency` configures the frequency of the clock in Hertz (Hz). If the desired frequency is not achievable (because of the peripheral resolution of 4 ns), the clock frequency is replaced by the closest achievable frequency and a warning is generated in BB Control utility software.
  This value is overwritten by the input signal value during run-time if the frequency is configured as `Variable`. In simulation, this value is used as the initial value during the first clock period.

Advanced parameters

- `Frequency value` the option “variable frequency“ enables the reconfiguration of the clock frequency during real-time execution using the input signal.
- `Frequency limits` sets the minimal and maximal frequencies that the clock generator uses as saturation points.



PLECS block

A CLK block using *CLOCK_0* is already embedded in the CONFIG block. See CONFIG – Interrupt configuration to learn more.

Signal specification

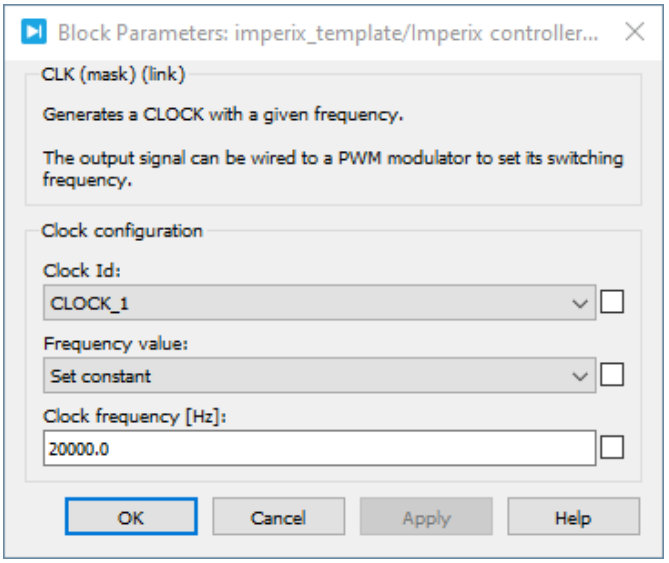
- The output can be wired to a PWM block input signal > to set its switching frequency.
- The input is only visible if the `Frequency value` is set as *variable frequency* and it sets the frequency of the clock during real-time execution.

Standard parameters

- `Clock ID` selects which clock generator to configure. (`CLOCK_0` is already used by the CONFIG block and cannot be instantiated in a separate clock block.)
- `Initial clock frequency` configures the frequency of the clock in Hertz (Hz). If the desired frequency is not achievable (because of the peripheral resolution of 4 ns), the clock frequency is replaced by the closest achievable frequency and a warning is generated in BB Control utility software.
  This value is overwritten by the input signal value during run-time if the frequency is configured as `Variable`. In simulation, this value is used as the initial value during the first clock period.

Advanced parameters

- `Frequency value` the option “variable frequency“ enables the reconfiguration of the clock frequency during real-time execution using the input signal.
- `Frequency limits` sets the minimal and maximal frequencies that the clock generator uses as saturation points.



C++ functions

`Clock_SetFrequency` — Configure the clock frequency

```
void Clock_SetFrequency(tClock clock, float freq);Code language: C++ (cpp)
```

Configures the frequency of the clock in Hertz (Hz).

If the desired frequency is not achievable (because of the peripheral resolution of 4 ns), the clock frequency is replaced by the closest achievable frequency and a warning is generated in BB Control utility software.

This routine has to be called in `UserInit()`. It can also be called in the interrupt if the clock has been set as real-time tunable using `Clock_ConfigureAsRealTimeTunable()`.

Parameters

- `clock`: the clock to configure (*CLOCK_0*, *CLOCK_1*, *CLOCK_2* or *CLOCK_3*)
- `freq`: the clock frequency in Hertz (Hz)

`Clock_SetPeriod` — Configure the clock period

```
void Clock_SetPeriod(tClock clock, unsigned int period);Code language: C++ (cpp)
```

Configures the period of the clock in ticks (1 tick = 4 ns).

It can be used in place of the standard `Clock_SetFrequency()` to configure the frequency of the clock such as: $\text{frequency} = \frac{1}{\text{period} \times \text{prescaler} \times 4\,\text{ns}}$.

It has to be called in `UserInit()`. It can also be called in the control interrupt routine if the clock has been set as real-time tunable using `Clock_ConfigureAsRealTimeTunable()`.

Parameters

- `clock`: the clock to configure (*CLOCK_0*, *CLOCK_1*, *CLOCK_2* or *CLOCK_3*)
- `period`: the period of the clock in ticks (1 tick = 4 ns). The maximal value is 65635.

`Clock_SetPrescaler` — Configure the clock prescaler

```
void Clock_SetPrescaler(tClock clock, unsigned int prescaler);Code language: C++ (cpp)
```

Configures the prescaler of the clock.

It can be used in place of the standard `Clock_SetFrequency()` to configure the frequency of the clock such as: $\text{frequency} = \frac{1}{\text{period} \times \text{prescaler} \times 4\,\text{ns}}$.

It has to be called in `UserInit()`. It can also be called in the control interrupt routine if the clock has been set as real-time tunable using `Clock_ConfigureAsRealTimeTunable()`.

Parameters

- `clock`: the clock to configure (*CLOCK_0*, *CLOCK_1*, *CLOCK_2* or *CLOCK_3*)

- `prescaler`: the prescaler of the clock. The maximal value is 65635.
  Using `prescaler=0` sets the prescaler to 1 and is therefore equivalent to `prescaler=1`.

`Clock_ConfigureAsRealTimeTunable` — Enable variable frequency operation

```
void Clock_ConfigureAsRealTimeTunable(tClock clock);Code language: C++ (cpp)
```

Enables the reconfiguration of the clock frequency during real-time execution. In other words it allows using `Clock_SetFrequency()`, `Clock_SetPeriod()` or `Clock_SetPrescaler()` in the control interrupt routine.

Parameters

- `clock`: the clock to configure (*CLOCK_0*, *CLOCK_1*, *CLOCK_2* or *CLOCK_3*)