BiSS-C - Digital encoder input

SD108 | Posted on January 30, 2026 | Updated on January 30, 2026

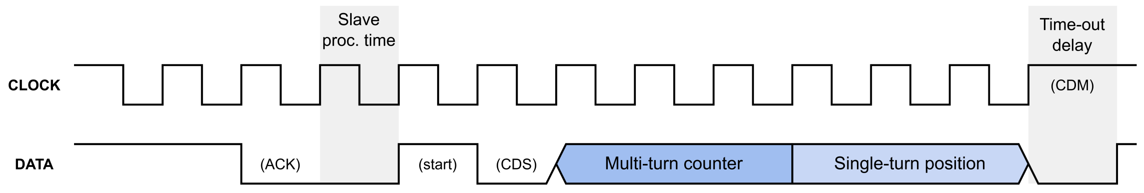
François LEDENT
Development Engineer
imperix · in

Table of Contents

The BiSS-C block instantiates a Bidirectional Synchronous Serial – Communication (BiSS-C) master to communicate with BiSS-C-compatible digital encoders and similar digital sensors, typically in motor drive applications.

This block is only compatible with the B-Box4, which provides two RS-485-based serial ports (SERIAL A and SERIAL B). Both ports can interface with sensors using three widely adopted communication protocols: SSI, BiSS-C (this page) and EnDat2.2.

Along with the single-turn position, this block supports the reception of a multi-turn counter and follows the common assumption that the single-turn and multi-turn values are placed side by side in the frame received from the sensor, as illustrated below.



BiSS-C frame with multi-turn and single-turn data side by side, as assumed by the BiSS-C block.
(ACK) = acknowledgment bit ; (start) = start bit ; (CDM/S) = control data master/slave

As the BiSS-C protocol does not define separate versions, this driver is compatible with any sensor that uses binary-encoded BiSS-C. Gray encoding is currently not supported.

BiSS-C protocol in a nutshell

Created in 2022 by iC Haus GbmH, the BiSS-C protocol is a master-slave serial standard widely used in industrial applications. Often considered as an extension to SSI, the BiSS-C protocol supports a higher clock frequency (up to 10 MHz) and propagation delay compensation.

The interface is composed of two unidirectional signals, CLOCK (driven by the master) and DATA (driven by the slave), regardless of the resolution (i.e., number of bits per revolution) of the sensor. Differential signaling improves the noise immunity and allows for longer cable runs, making it suitable for industrial environments.

As illustrated above, the frame transmission consists of the following steps:

1. The controller (master) initiates the transmission by starting the CLOCK.
2. The sensor (slave) acknowledges by driving DATA signal to LOW on the second rising edge of the CLOCK (see ‘ack’ on the figure). This mechanism allows the master to estimate the total propagation delay by comparing the CLOCK and DATA edges.
3. When needed, the slave can maintain the DATA signal to LOW for several CLOCK cycles, to sample the position or for internal computations for instance (see ‘slave delay’).

4. The slave triggers the payload transmission by sending the start bit (see ‘start’).
5. The first bit of the payload is the slave-to-master control data bit (see ‘CDS’).
6. The slave transmits the payload, bit by bit on DATA, synchronously with the CLOCK.
7. At the end of the transmission, the slave maintains the DATA signal to LOW during the time-out. The master can optionally send a master-to-slave control data bit at this moment.
8. The CLOCK and DATA return to IDLE.

The BiSS-C protocol provides room for a control mechanism to access the slave’s configuration memory. However, since the control bits are distributed across multiple frames—with only two control bits per frame (see “CDM” and “CDS”)—this mechanism significantly increases complexity and is currently not supported in the imperix implementation.

While the BiSS-C standard defines how the clock and data signals are exchanged between the master and the slave, it does *not* specifies the frame structure. The content of the frame may therefore vary from one sensor to another.

The angle within the current rotation is usually called **single-turn position**. Most digital encoders also offer the capability to track the number of rotations, usually referred to as **multi-turn counter**.

Strengths and limitations

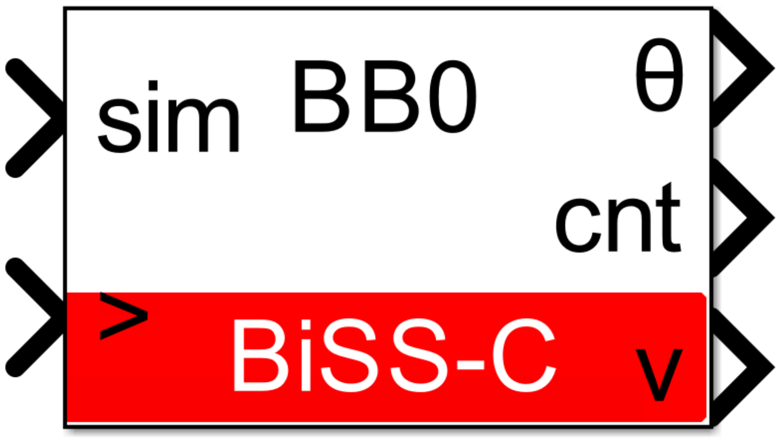
The BiSS-C protocol is valued for its simple and robust design. With only two differential and unidirectional digital lines, it can be easily integrated and performs reliably even in electrically noisy environments.

The synchronous data transfer provides accurate, low-latency position feedback, making it well suited for real-time control. Moreover, it supports higher clock frequencies (up to 10 MHz) and allows the compensation of the propagation delay.

Simulink block

Signal specification

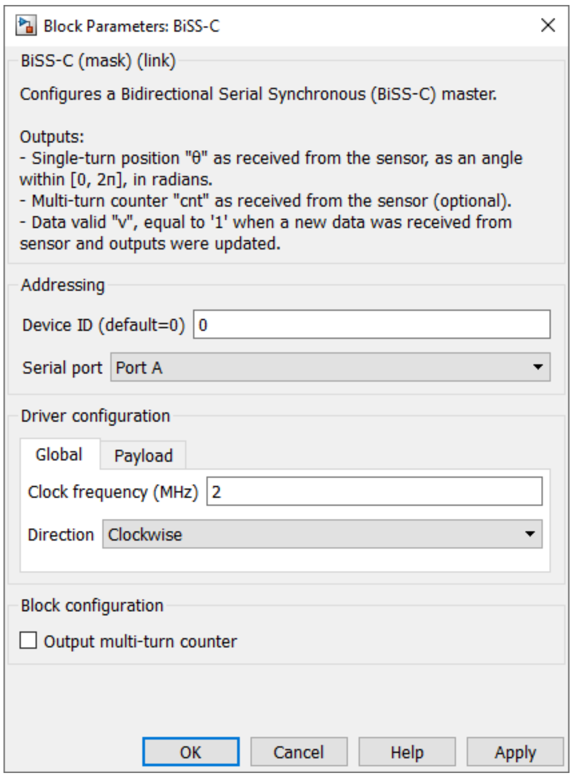
- The output θ is the single-turn position, in radians, in [0, 2π].
- The output `cnt` is the multi-turn counter, as received from the sensor. This port is hidden by default but it can be shown using the `Output multi-turn counter` checkbox.
- The output `v` is the data valid signal, set to 1 each time a new data are available.
- The input `sim` is a 2-dimensional vector used in simulation and represents the angle value (in radians) and multi-turn counter (-) computed by the simulation plant model.
- The `>` input signal needs to be connected to the sampling clock generated by the CONFIG block.



Parameters

- `Device ID` selects which device to address when used in a multi-device configuration.
- `Serial port` selects the serial port.
- `Clock frequency` specifies the transmission clock frequency for the BiSS-C communication, in MHz. The clock frequency cannot exceed 10 MHz, as defined by the BiSS-C standard.
- `Direction` specifies the direction as clockwise or counterclockwise. When counterclockwise is selected, the output angle θ is 2π-θ’, where θ’ is the angle received from the sensor.

- `Single-turn resolution` specifies the resolution (i.e., number of bits) for the single-turn position. The single-turn resolution cannot exceed 32 bits.
- `Multi-turn resolution` specifies the resolution (i.e., number of bits) for the multi-turn counter. The multi-turn resolution cannot exceed 32 bits.
- `Output multi-turn counter` shows or hides the multi-turn counter output `cnt`.



Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Addressing

Device ID (default=0): 0

Serial port: Port A

Driver configuration

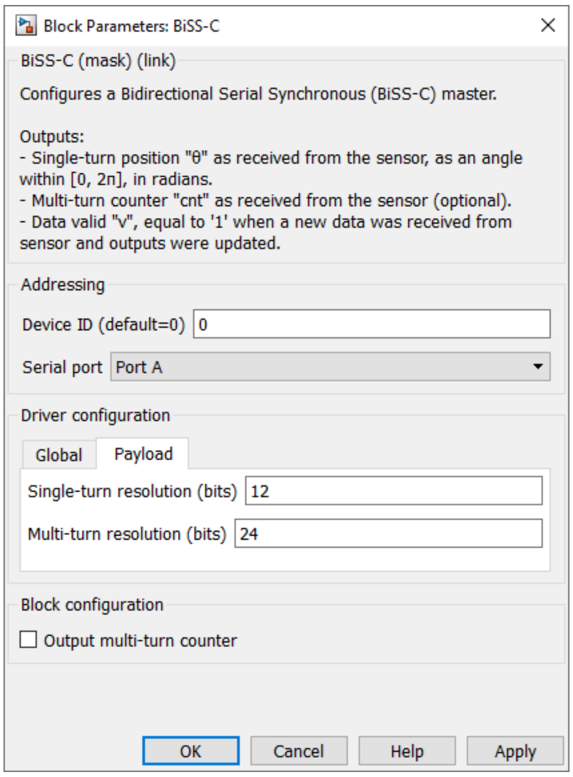
Global | Payload

Clock frequency (MHz): 2

Direction: Clockwise

Block configuration

☐ Output multi-turn counter



Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Addressing

Device ID (default=0): 0

Serial port: Port A

Driver configuration

Global | Payload

Single-turn resolution (bits): 12

Multi-turn resolution (bits): 24

Block configuration

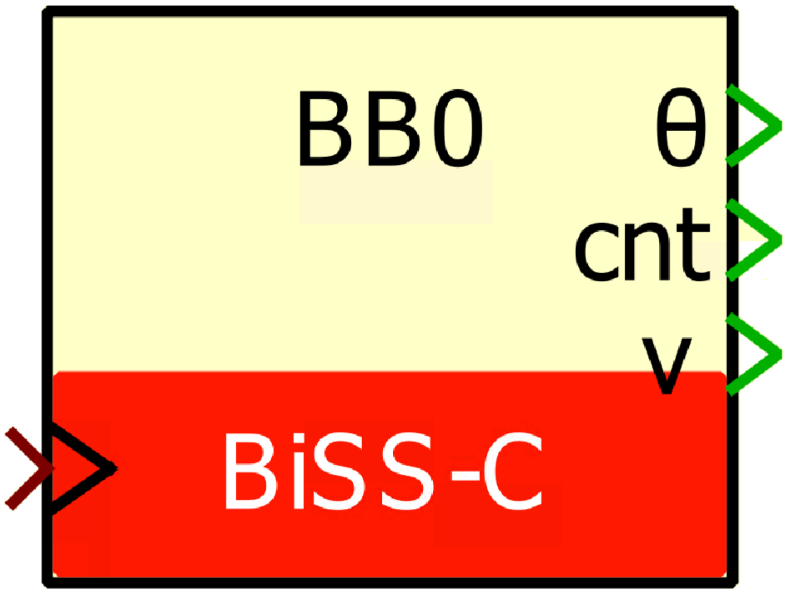
☐ Output multi-turn counter

PLECS block

Signal specification

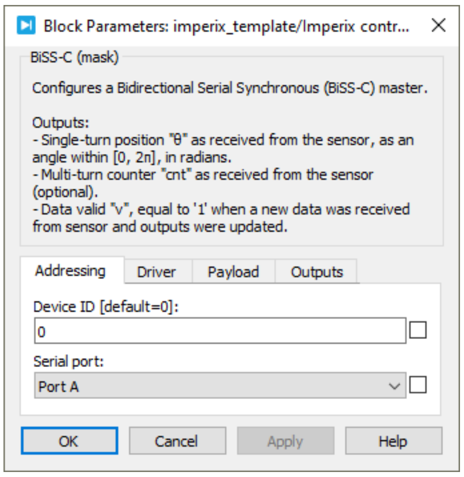
- The output θ is the single-turn position, in radians, in $[0, 2\pi]$.

- The output `cnt` is the multi-turn counter, as received from the sensor. This port is hidden by default but it can be shown using the `Output multi-turn counter` checkbox.
- The output v is the data valid signal, set to 1 each time a new data are available.
- The input `sim` is a 2-dimensional vector used in simulation and represents the angle value (in radians) and multi-turn counter (-) computed by the simulation plant model.
- The > input signal needs to be connected to the sampling clock generated by the CONFIG block.



Parameters

- `Device ID` selects which device to address when used in a multi-device configuration.
- `Serial port` selects the serial port.
- `Clock frequency` specifies the transmission clock frequency for the BiSS-C communication, in MHz. The clock frequency cannot exceed 10 MHz, as defined by the BiSS-C standard.
- `Direction` specifies the direction as clockwise or counterclockwise. When counterclockwise is selected, the output angle θ is 2π-θ', where θ' is the angle received from the sensor.
- `Single-turn resolution` specifies the resolution (i.e., number of bits) for the single-turn position. The single-turn resolution cannot exceed 32 bits.
- `Multi-turn resolution` specifies the resolution (i.e., number of bits) for the multi-turn counter. The multi-turn resolution cannot exceed 32 bits.
- `Output multi-turn counter` shows or hides the multi-turn counter output `cnt`.


Block Parameters: imperix_template/Imperix contr...

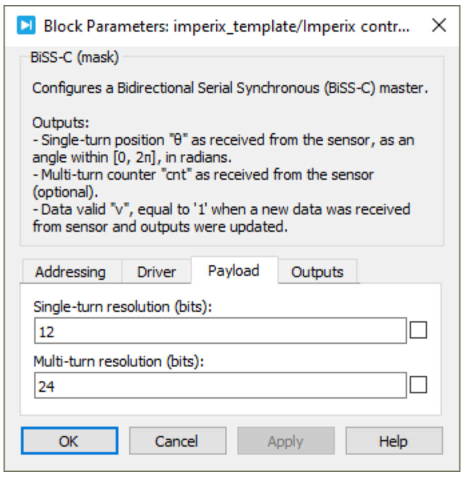
BiSS-C (mask)

Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Device ID [default=0]: 0

Serial port: Port A


Block Parameters: imperix_template/Imperix contr...

BiSS-C (mask)

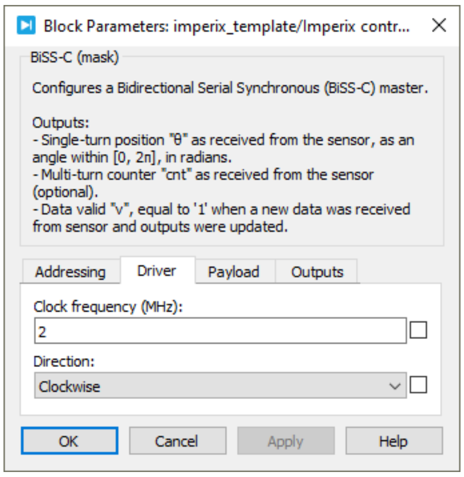
Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Addressing | Driver | Payload | Outputs

Single-turn resolution (bits): 12

Multi-turn resolution (bits): 24


Block Parameters: imperix_template/Imperix contr...

BiSS-C (mask)

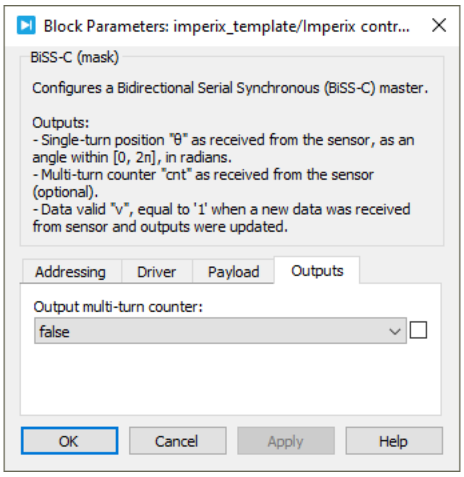
Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Addressing | Driver | Payload | Outputs

Clock frequency (MHz): 2

Direction: Clockwise


Block Parameters: imperix_template/Imperix contr...

BiSS-C (mask)

Configures a Bidirectional Serial Synchronous (BiSS-C) master.

Outputs:
- Single-turn position "θ" as received from the sensor, as an angle within [0, 2π], in radians.
- Multi-turn counter "cnt" as received from the sensor (optional).
- Data valid "v", equal to '1' when a new data was received from sensor and outputs were updated.

Addressing | Driver | Payload | Outputs

Output multi-turn counter: false

C++ functions

`BissC_Init` — Initialize the BiSS-C master

```cpp
void BissC_Init(unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function initializes the serial port *port* of the device *device* for the BiSS-C communication protocol.

It has to be called in `UserInit()`.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: Serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_ConfigureClkFrequency` — Configure the transmission clock frequency

```cpp
void BissC_ConfigureClkFrequency(float clk_frequency, unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function configures the frequency of the transmission clock for the BiSS-C communication on the serial port *port* of the device *device*, in MHz. The clock frequency cannot exceed 10 MHz, as defined by the BiSS-C standard.

It has to be called in `UserInit()`.

TPI Rev. F: Only serial port A is available.

Parameters

- `clk_frequency`: transmission clock frequency, in MHz
- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_ConfigureDirection` — Configure the rotation direction

```
void BissC_ConfigureDirection(tRs485Direction direction, unsigned int port, unsigned int device);Code language: C++ (
```

This function configures the direction for the BiSS-C communication on the serial port *port* of the device *device*, as *clockwise* or *counterclockwise*. When *counterclockwise* is selected, the output angle θ is 2π-θ’, where θ’ is the angle received from the sensor.

It has to be called in `UserInit()`.

TPI Rev. F: Only serial port A is available.

Parameters

- `direction`: direction (RS485_CW or RS485_CCW)
- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_ConfigurePayload` — Configure the frame payload

```
void BissC_ConfigurePayload(unsigned short n_bit_before, unsigned short n_bit_multiturn, unsigned short n_bit_sing
```

This function configures the expected structure of the frame received from the sensor. The function is based on the common assumption that the single-turn and multi-turn values are placed side by side in the frame received from the sensor.

The single-turn and multi-turn resolutions cannot exceed 32 bits each, and the total payload length cannot exceed 64 bits (i.e., n_bit_before + n_bit_multiturn + n_bit_singleturn + n_bit_after $\leq$ 64).

In general, n_bit_before might be 0 and n_bit_

As an example, the function call for a E36CM-BIS-1311-1224 (Hohner) is *BissC_ConfigurePayload(0, 24, 12, 8, <port>, <device>)*.

It has to be called in `UserInit()`.

TPI Rev. F: Only serial port A is available.

Parameters

- `n_bit_before`: number of bits before the multi-turn value in the frame received from the sensor
- `n_bit_multiturn`: multi-turn resolution, i.e. number of bits composing the multi-turn value in the frame received from the sensor
- `n_bit_singleturn`: single-turn resolution, i.e. number of bits composing the single-turn value in the frame received from the sensor
- `n_bit_after`: number of bits after the single-turn value in the frame received from the sensor
- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_ReadFrame` — Read the current frame

```
void BissC_ReadFrame(unsigned int port, unsigned int device);Code language: C++ (cpp)
```

This function triggers the CPU to read and interpret the last frame received from the sensor.

It must be called in the control interrupt routine.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_GetPosition` — Get the sensor’s position

```cpp
float BissC_GetPosition(unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function returns the single-turn position received from the sensor as an angle, in radians, in [0, 2π].

It must be called in the control interrupt routine, after *BissC_ReadFrame*.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_GetMultiturnCounter` — Get the sensor's multi-turn counter

```cpp
unsigned int BissC_GetMultiturnCounter(unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function returns the multi-turn counter received from the sensor.

It must be called in the control interrupt routine, after *BissC_ReadFrame*.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_GetDataValid` — Get the data valid flag

```cpp
unsigned int BissC_GetDataValid(unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function returns the data valid flag indicating if a new data was received from the sensor. The flag being ‘1’ means that the outputs of *BissC_GetPosition* and *BissC_GetMultiturnCounter* were updated since last interruption. If the flag is ‘0’, the outputs were not updated and have the same value.

It must be called in the control interrupt routine, after *BissC_ReadFrame*.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)

`BissC_ErrorCheck` — Raise warning in case of communication error

```cpp
void BissC_ErrorCheck(unsigned int port, unsigned int device);
```
Code language: C++ (cpp)

This function checks the communication status and raises warnings in case of communication error(s).

It must be called in the control interrupt routine.

TPI Rev. F: Only serial port A is available.

Parameters

- `port`: serial port, 0 (port A) or 1 (port B)
- `device`: imperix device to address in a multi-device configuration (default: 0)