ETH in - Ethernet input mailbox

SD018 | Posted on April 2, 2021 | Updated on July 24, 2025

Stéphane LOVEJOY
Senior Software Developer
imperix • in

Table of Contents

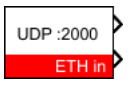
The Ethernet input mailbox block allows receiving up to 1024 bytes of data via Ethernet using the UDP protocol (in SDK versions prior to 2025.2, the size limit was 32-bit). To send up to 1024 bytes of data via Ethernet using the UDP/IP protocol, the Ethernet output mailbox should be used.

The ETH input block reads *n* bytes of data received via UDP/IP on the specified port and applies it to the output port of the block. The value of *n* is defined by the `Signal decoding format` parameter (in bytes) multiplied by the `Number of signals` parameter. The data can be interpreted as one of the following types: int8, int16, int32, uint8, uint16, uint32, float32, or float64. It features a second output that indicates that new data has been received. The value on the output data port will remain unchanged until new data are received.

Simulink block

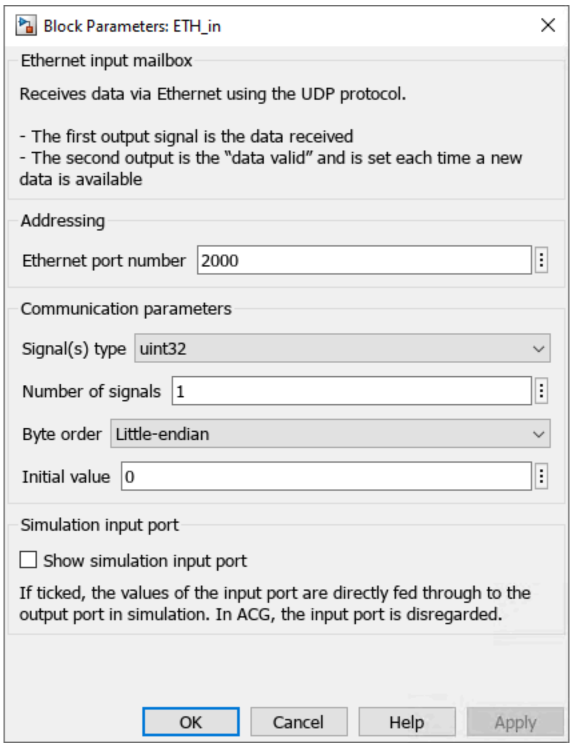
Signal specification

- The data output signal returns *n* bytes of data received via UDP, formatted as a vector. The value of *n* is defined by the `Signal decoding format` parameter (in bytes) multiplied by the `Number of signals` parameter. The output data type is configured by the `Signal decoding format` parameter, and the vector length is defined by the `Number of signals` parameter. (e.g. with `Signal coding format` set to uint32 and `Number of signals` set to 100, the data size read will be 4 x 100 = 400 bytes. The output vector will have a length of 100, with each element occupying 4 bytes)
- The second signal is the data valid output. It is set to 1 each time new data are available.



Parameters

- `Ethernet port number`: sets the port number on which data will be received.
- `Signal decoding format`: defines the type of the data output (int8, int16, int32, uint8, uint16, uint32, float32, or float64).
- `Number of signals`: specifies the vector size of the data output signal.
- `Byte order`: defines the byte order in which the data will be read. (little-endian or big-endian)
- `Initial value`: sets the initial value of the data output before any data are received. The value is interpreted as a uint64.


Block Parameters: ETH_in

Ethernet input mailbox

Receives data via Ethernet using the UDP protocol.

- The first output signal is the data received
- The second output is the "data valid" and is set each time a new data is available

Addressing

Ethernet port number: 2000

Communication parameters

Signal(s) type: uint32

Number of signals: 1

Byte order: Little-endian

Initial value: 0

Simulation input port

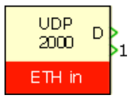
☐ Show simulation input port

If ticked, the values of the input port are directly fed through to the output port in simulation. In ACG, the input port is disregarded.

PLECS block

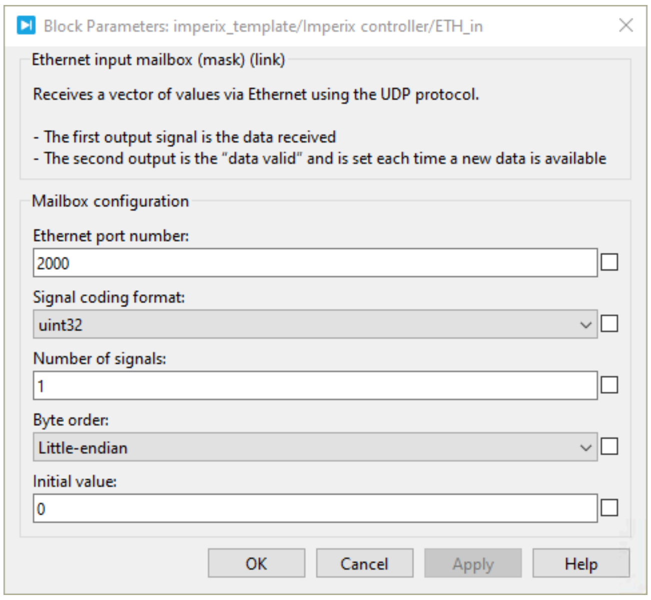
Signal specification

- The data output signal returns 32 bits of data received via UDP. The output data type is configured by `Signal decoding format` parameter.
- The second signal is the data valid output. It is set to 1 each time new data are available.



Parameters

- `Ethernet port number`: sets the port number on which data will be received.
- `Signal decoding format`: defines the type of the data output (int8, int16, int32, uint8, uint16, uint32, float32, or float64).
- `Number of signals`: specifies the vector size of the data output signal.
- `Byte order`: defines the byte order in which the data will be read. (little-endian or big-endian)
- `Initial value`: sets the initial value of the data output before any data are received. The value is interpreted as a uint64.


Block Parameters: imperix_template/Imperix controller/ETH_in
OK
Cancel
Apply
Help

Ethernet input mailbox (mask) (link)

Receives a vector of values via Ethernet using the UDP protocol.

- The first output signal is the data received
- The second output is the "data valid" and is set each time a new data is available

Mailbox configuration

Ethernet port number: 2000

Signal coding format: uint32

Number of signals: 1

Byte order: Little-endian

Initial value: 0

C++ functions

`Eth_ConfigureInputMailbox` **— Configure an Ethernet input mailbox**

`bool Eth_ConfigureInputMailbox(unsigned int mailboxId, unsigned int port, tEndianness endianness = LITTLE_ENDIAN,`

Routine used to configure an Ethernet UDP input Mailbox. It has to be called in UserInit().

`Eth_ConfigureInputMailboxInitialValue` **— Configure an Ethernet input mailbox** initial value

`bool Eth_ConfigureInputMailboxInitialValue(unsigned int mailbox_id, void* data, size_t size);Code language: C++ (cpp)`

`void Eth_ConfigureInputMailboxInitialValue(unsigned int mailboxId, unsigned int initialValue);Code language: C++ (cpp)`

`void Eth_ConfigureInputMailboxInitialValue(unsigned int mailboxId, int initialValue);Code language: C++ (cpp)`

`void Eth_ConfigureInputMailboxInitialValue(unsigned int mailboxId, float initialValue);Code language: C++ (cpp)`

Routines used to set the initial value read for an Input mailbox. These functions can be used to configure the initial value returned by the Eth_Read functions before any UDP message is received. They have to be called in UserInit().

`Eth_Read` **— Read received value**

`int Eth_Read(unsigned int mailboxId, void* data, size_t size);Code language: C++ (cpp)`

`int Eth_Read(unsigned int mailboxId, unsigned int &data);Code language: C++ (cpp)`

`int Eth_Read(unsigned int mailboxId, int &data);Code language: C++ (cpp)`

`int Eth_Read(unsigned int mailboxId, float &data);Code language: C++ (cpp)`

These functions are used to send data on Ethernet using UDP. They have to be called during the control interrupt.