LINZ
CENTER OF
MECHATRONICS

# X2C Scope Configuration

Documentation

**Last modified:** 2023-02-10

## Table of Contents

# 1 Document History

| Version | Changes |
|---|---|
| 1 | Created |
| 2 | Added channel source location description in 6.4.2 for all possible channel source types<br>Added supported trigger data source sizes in 6.5.1<br>Fixed the data size to be in SDA memory units instead of bytes for the channel source data type size in 6.4.3. |

## 2 Versions

This documentation is for Scope- and LNet version included with X2Cscope plugin.
LNet version 5. Scope version 2 from X2C revision 1165.

## 3 LNet Services

The LNet protocol does have 2 services to download and upload X2C Block parameter values.

In case of the Scope Block, a third LNet service, “Get RAM Block”, is used to upload the Scope sample data from the Scope buffer when the Scope has finished its sampling.

For further details please refer to the LNet documentation.

## 4 Load- and Save Functions

X2C Blocks do have a Save- and a Load function. The Save function is used to download new parameter values to the Block. The Load function is used to upload the current parameter values from the Block.

The LNet services “Save Parameter” and “Load Parameter” are being used to call the “Save function” or the “Load function” of the appropriate Block.

All data in LNet is transferred LSB (least significant byte) first.

## 5 Scope Block

The X2C Scope is implemented as a “special” X2C Block. Therefore, it does have a Save- and a Load function that can be used to transfer data from or to the Scope Block.

The Save Function sets the Scope state to IDLE before changing any parameter values. This avoids possible undefined behavior if the Scope Save function is interrupted by an X2C Update cycle.

If an invalid value is supplied to the Save function, it will return an error to the LNet “Save Parameter” function. This will cause the LNet “Save Parameter” service to return a “Format error”.

### *5.1 Typical Operation*

1. Get the data memory width of the used target platform
   => neither already known nor can be fetched by getting the target platform type by the LNet “Get Device Info” service
   => required to calculate the Data Set Size (DSS) for pre-trigger mode
2. Upload Scope Parameter by using the LNet “Load Parameter” service
   => fetches the SDA size (if a percentual trigger window is desired)
   => fetch SDA array address (usually does not change if it was created statically) which is required to upload the SDA data when the data sampling procedure has finished
3. Download new Scope Parameters by using the LNet “Save Parameter” service
   => this configures and starts the Scope
4. Periodically upload the Scope Parameters (“Load Parameter”) to poll the Scope state to become IDLE
   => when the Scope state becomes idle, the data sampling procedure has finished
5. Get the “SDA Used Length” neither from the last uploaded parameter set nor by calculating it (=> SDA size - (SDA size modulo DSS))
6. In case of pre-trigger mode, get the “Trigger Event Position” from the last uploaded parameter set
   => required to get the correct SDA organization (see 8.1)
7. Upload the SDA data by using the LNet “Get RAM Block” service
   => use the parameters “SDA address” and “SDA Used Length”
8. Extract the data according to the SDA organization (see 8.1)
   => please pay attention to the special handling for pre-trigger mode

# 6 Scope Save Parameters

The Save Parameters are used to configure the Scope and start the sampling procedure.

### *6.1 Scope state*

Type: uint8

The Scope state starts or stops the sampling procedure. The Scope state also defines if the Scope runs in AUTO (no-trigger) or NORMAL (trigger) mode.

The following values can be used for the Scope state value:

- Stop Scope: 0x00

- Start Scope in AUTO mode: 0x02
- Start Scope in NORMAL mode: 0x01

In AUTO mode, the currently stored trigger settings are not used.

## *6.2 Number of Scope Channels*

Type: uint8

The number of scope channels being used. This value also defines the number of channel configs that must be included within the current Save Parameter set.

## *6.3 Sample time factor*

Type: uint16

This parameter defines a pre-scaler when the Scope is in the sampling mode. This parameter can be used to extend the total sampling time at the cost of sampling resolution. For example, setting this value to 0 means to sample data at every Update function call. Value 1 means to sample every 2nd step, value 2 to sample every 3rd step etc.

## *6.4 Channel configuration(s)*

The number of channel configurations that must be provided is defined by the "Number of Scope channels". The minimum number of channels is 1, the maximum is 8. Each channel configuration consists of the following 3 elements:

### 6.4.1 Channel Source type

Type: uint8

The source type defines, on how the "Source location" is being interpreted. There are currently four source types available:

- Type 0x00: Address
- Type 0x01: Control Block
- Type 0x02: Inport
- Type 0x03: Outport

**Currently the X2Cscope plugin only uses the source type "Address".**

### 6.4.2 Channel Source location

Type: uint32

The source location is a 32-bit (4 byte) value that defines the data source. Depended on the “Source type”, this value is interpreted in different ways.

In case of Address source type, the 4 bytes form a 32-bit memory address:

| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|
| Address byte 0 | Address byte 1 | Address byte 2 | Address byte 3 |

*Table 1 Address source location*

In case of Control Block type, the Block Parameter ID (16-bit) and the element number (16-bit) is used:

| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|
| Param ID (LSB) | Param ID (MSB) | Element ID (LSB) | Element ID (MSB) |

*Table 2 Control Block souce location*

In case of Inport- or Outport type, the Block Parameter ID (16-bit) is used. The remaining bytes are currently unused:

| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|
| unused | unused | Param ID (LSB) | Param ID (MSB) |

*Table 3 Inport/Outport source location*

**Currently the X2Cscope plugin uses the “Address” type only.**

#### 6.4.3 Channel Source data type size

Type: uint8

This defines the size of the data type to be sampled in SDA memory units. The SDA memory unit depends on the target platform. See the physical memory organization in 8.2.

## 6.5 Trigger Configuration

The trigger configuration contains all parameters if the trigger mode is selected. A trigger configuration must be provided even if the trigger mode is not used. The trigger configuration consists of the following 6 elements:

### 6.5.1 Trigger Data Type

Type: uint8

The data type defines the number of bytes, the format type (integer or float) and the sign type (unsigned or signed) of the trigger source. The 8-bit value does have the following structure:

- Bit 0 – 3: Trigger source data size <u>in bytes</u>
  - Supported values: 1, 2, 4, 8
- Bit 4: reserved
- Bit 5: Sign type
  - 1 for signed
  - 0 for unsigned
- Bit 6: Format type
  - 1 for float
  - 0 for integer
- Bit 7: Old version format identifier
  - 1 for Scope versions > 1
  - 0 for Scope version 1

In case of float format type, the sign type bit is ignored.

**The X2Cscope plugin uses Scope version 2, so the “Old version format identifier” bit must always be set to 1.**

### 6.5.2 Trigger Source type

This works in the same way than the channel source type. Please refer to 6.4.1.

### 6.5.3 Trigger Source location

This works in the same way than the channel source location. Please refer to 6.4.2.

### 6.5.4 Trigger Level

Type: Depends on the data size defined for Trigger Data Type, see 6.5.1.

The value contains the Scope trigger value. The size (and the interpretation) depends on the parameter Trigger Data Type, see 6.5.1.

### 6.5.5 Trigger delay

Type: int32

Depended on the sign of this value, it configures a pre- or a post-trigger window or disables the trigger delay feature at all. The absolute value defines the number of samples for the pre- or post-trigger window multiplied by the current DSS (Data Set Size, see 8.3).

For example, to get a post-trigger window of 200 samples with a DSS of 3, the value must be set to 200 * 3 * (-1) = -600. To get a pre-trigger window with the same configuration to value must be set to 200 * 3 * (+1) = 600.

| | |
|---|---|
| Value = 0 | No trigger delay |
| Value > 0 | Pre-trigger |
| Value < 0 | Post-trigger |

*Table 4 Trigger delay values*

### 6.5.6 Trigger edge

Type: uint8

This value selects the trigger edge and currently supports 2 modes:

| | |
|---|---|
| Value = 0 | Falling Edge |
| Value = 1 | Rising Edge |

*Table 5 Trigger edge values*

### 6.5.7 Trigger mode

Type: uint8

This value selects the trigger mode and currently supports 2 modes. This flag is intended to replace the trigger information from the Scope state for future releases.

**Therefore, it must be in sync with the sent Scope state.**

| | |
|---|---|
| Value = 0 | AUTO mode (no trigger) |
| Value = 1 | NORMAL mode (trigger active) |

*Table 6 Trigger mode values*

## *6.6 Save Function Structure*

This table shows the order in which the data must be provided to the Scope Save Function:

| Element | Name |
|---|---|
| 1 | Scope State |
| 2 | Number of Scope Channels |
| 3 | Sample Time Prescaler |
| 4 | One or multiple Source Configurations<br>=> see Source Configuration elements |

| | |
|---|---|
| 5 | One Trigger Configuration<br>=> see Trigger Configuration elements |

*Table 7 Save Function structure*

Source Configuration structure:

| Element | Name |
|---|---|
| 1 | Source type |
| 2 | Source location |
| 3 | Source data type size |

*Table 8 Source configuration structure*

Trigger Configuration structure:

| Element | Name |
|---|---|
| 1 | Trigger data type |
| 2 | Trigger source type |
| 3 | Trigger location |
| 4 | Trigger level |
| 5 | Trigger delay (window) |
| 6 | Trigger Edge |
| 7 | Trigger Mode |

*Table 9 Trigger configuration structure*

# 7 Scope Load Parameters

The Load Parameters are used to get the Scope Data Array address and maximum size for once. During operation it is used to poll the current Scope state to check, if the Scope operation is currently busy or has finished its operation and to get the currently used size of the Scope Data Array. The currently used size of the Scope Data Array depends on the channel configurations.

## 7.1 Scope state

Type: uint8

The current Scope state. It may be polled to check if the Scope sampling procedure is currently in progress or if it has finished the sample operation.

| | |
|---|---|
| Value = 0 | Scope is idle |
| Value > 0 | Scope is busy |

*Table 10 Load Parameter Scope states*

### 7.2 Number of Scope channels

Type: uint8

The number of active channels. See Save Function element 6.2.

### 7.3 Sample Time Factor

Type: uint16

The current Sample Time Factor. See Save function element 6.3.

### 7.4 Scope Data Array Pointer

Type: uint32

This value is for debug purposes only. It points to the next free location in the Scope Data Array for the next dataset to be stored. This value is an index, not a memory address.

### 7.5 Scope Data Array Address

Type: uint32

This value contains the memory address of the Scope Data Array. This array contains the sampled data.

Also see 8.1 for more information.

### 7.6 Trigger delay

Type: uint32

This is the current trigger delay value. See Save function element 6.5.5.

### 7.7 Trigger Event Position

Type: uint32

This value is only used in case of pre-trigger mode. It is an index for the Scope Data Array and points to the index of the first data set after the trigger event. For all other modes (no-trigger, trigger, post-trigger), the first dataset is always located at the start (index = 0) of the Scope Data Array and therefore this value is not needed for these modes.

Also see 8.1 for more information.

### *7.8 Scope Data Array Used Length*

Type: uint32

This value is the number of the used Scope data array elements. The number of the maximum usable elements depends on the Scope array size and the size of one dataset. For example, a total array size of 1024 elements and a dataset size of 7 elements results in a maximum total usable size of 1024/7 = 146 elements. The last 2 elements are not used in this case.

Also see 8.1 for more information.

### *7.9 Scope Data Array Size*

Type: uint32

This value is the size of the Scope Data Array.

### *7.10 Scope Version*

Type: uint8

This value returns the Scope version. The 8-bit value has the following structure:

- Bit 0 – 7: Scope Version
- Bit 8: Always 1

### *7.11 Load Function Structure*

This table shows the order in which the data must be provided to the Scope Load Function:

| Element | Name |
|---|---|
| 1 | Scope state |
| 2 | Number of Channels |
| 3 | Sample Time Factor |
| 4 | Scope Data Array Pointer |
| 5 | Scope Data Array Address |

| | |
|---|---|
| 7 | Trigger window (delay) |
| 8 | Trigger Event Position |
| 9 | Scope Data Array Used Length |
| 10 | Scope Data Array Size |
| 11 | Scope Version |

*Table 11 Load Function structure*

# 8 Scope Data Array (SDA)

## 8.1 Organization

The SDA contains the data being sampled by the Scope. The datasets are stored in ascending order:

| | | | |
|---|---|---|---|
| Set 1 | Set 2 | … | Set <n> |

Each dataset contains the active channels data in ascending order:

| | | | |
|---|---|---|---|
| Channel 1 | Channel 2 | … | Channel <n> |

The number of datasets that can be stored depends on the SDA size and the dataset channel configuration. Therefore, if the SDA size is not a multiple of the dataset size, some elements may be unused.

- ⇨ Elements used = SDA size / dataset size
- ⇨ Elements unused = SDA size modulo dataset size

Usually, the datasets are stored in ascending order in the SDA. That means the first dataset is located at the first element and the last dataset at the end of the SDA. For example, an SDA that can store 10 datasets:

| E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 |
|---|---|---|---|---|---|---|---|---|---|
| DS1 | DS2 | DS3 | DS4 | DS5 | DS6 | DS7 | DS8 | DS9 | DS10 |

*Table 12 SDA layout example*

An exception to this case is if the pre-trigger mode is used. Because the Scope does have to collect data for the pre-trigger window, the location of the datasets depends on the location of the trigger event and the pre-trigger window. In this case the SDA is used as a circular buffer.

For example, an SDA that can store 10 datasets. The trigger event occurred at element 7. The pre-trigger window is set to 4. The sample procedure continues to sample after element 7 until the end of SDA is reached. The next dataset will be stored at the begin of SDA until the last dataset at element 3 is sampled.

| E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 |
|---|---|---|---|---|---|---|---|---|---|
| DS8 | DS9 | DS10 | DS1 | DS2 | DS3 | DS4 | DS5 | DS6 | DS7 |

*Table 13 Pre-trigger SDA layout example*

### *8.2 Physical element size*

The SDA usually is an C array. To avoid waste of memory, the C array type is always the same than the memory data width in which the SDA is allocated. For example, a target platform with an 8-bit RAM memory width uses an uint8 array type, a target platform with a 16-bit RAM memory width uses an uint16 array type.

This is important to use the correct LNet upload procedure to get the SDA content.

### *8.3 Data Set Size (DSS)*

The Data Set Size is the number of all channel configurations in SDA memory units. As mentioned in 0, the SDA memory unit depends on the target platform. For example, the following channel config has different DSS on different target systems.

Example Channel Config

- Channel #1, uint16 type
- Channel #2, uint32 type

DSS @ 8-bit platform: 2 (uint16) + 4 (uint32) = 6

DSS @ 16-bit platform: 1 (uint16) + 2 (uint32) = 3

# 9 Examples

## 9.1 Save Parameter Examples

Example 1:

Configuration:

- Target: dsPIC or PIC32 (RAM memory width = 8 bit)
- No trigger (AUTO mode)
- Channel 1: Address = 0x11223344, Data type = uint16
- Sample time factor = 4 (sample every 5th step)

Since a trigger configuration must be provided (even if the trigger feature is not used), a “fake” trigger setup may be provided:

- Trigger data type = uint16
- Trigger source type = Address
- Trigger location = 0x0000’0000
- Trigger level = 0x0000 (Trigger data type size is 2 bytes, so trigger level is also 2 bytes wide)
- Trigger delay = 0x0000’0000
- Trigger Edge = RISING
- Trigger Mode = AUTO

Save function data layout:

| Byte | Name | Value |
|---|---|---|
| 0 | Scope state, set for no-trigger (AUTO) mode | 0x02 |
| 1 | Number of channels | 0x01 |
| 2 | Sample time factor LSB | 0x04 |
| 3 | Sample time factor MSB | 0x00 |
| 4 | Channel #1, Source type, Address type | 0x00 |
| 5 | Channel #1, Source location, Address LSB | 0x44 |
| 6 | Channel #1, Source location, Address LSB | 0x33 |
| 7 | Channel #1, Source location, Address MSB | 0x22 |
| 8 | Channel #1, Source location, Address MSB | 0x11 |
| 9 | Channel #1, Data type size<br>=> uint16 @ 8-bit RAM width = 2 | 0x02 |
| 10 | Trigger data type, uint16<br>=> Bit 7 is always set because of “New Scope Version”<br>=> Bit 5 Sign Bit is not set because of unsigned data type<br>=> Bit 4 Type Bit is not set because of integer data type | 0x82 |

| | | |
|---|---|---|
| | => Bit 0-3 is the byte size of uint16 = 2 | |
| 11 | Trigger source type, Address | 0x00 |
| 12 | Trigger source location, Address LSB | 0x00 |
| 13 | Trigger source location, Address LSB | 0x00 |
| 14 | Trigger source location, Address MSB | 0x00 |
| 15 | Trigger source location, Address MSB | 0x00 |
| 16 | Trigger level LSB | 0x00 |
| 17 | Trigger level MSB | 0x00 |
| 18 | Trigger delay LSB | 0x00 |
| 19 | Trigger delay LSB | 0x00 |
| 20 | Trigger delay MSB | 0x00 |
| 21 | Trigger delay MSB | 0x00 |
| 22 | Trigger edge, RISING | 0x01 |
| 23 | Trigger mode, AUTO | 0x00 |

<u>Example 2:</u>

Configuration:

- Target: dsPIC or PIC32 (RAM memory width = 8 bit)
- Trigger Mode
- Sample Time Factor = 0 (sample every step)

Channel configuration:

- Channel 1: Address type, location = 0xDEAD’CAFE, type = uint32
- Channel 2: Address type, location = 0x8899’AABB, type = int16
- Data Set Size (DSS) = Ch 1 + Ch 2 = 4 (uint32) + 2 (uint16) = 6

Trigger configuration:

- Trigger data type = int32
- Trigger source type = ADDRESS
- Trigger location = 0x1234’5678
- Trigger level = 70’000 (Trigger data type size is 4 bytes, so trigger level is also 4 bytes wide)
- Trigger delay = Pre-trigger, 100 samples
  => Delay value = 100 * DSS = 600 (positive because pre-trigger)
- Trigger Edge = FALLING
- Trigger Mode = NORMAL

Save function data layout:

| Byte | Name | Value |
|---|---|---|
| 0 | Scope state, set for trigger (NORMAL) mode | 0x01 |
| 1 | Number of channels | 0x02 |
| 2 | Sample time factor LSB | 0x00 |
| 3 | Sample time factor MSB | 0x00 |
| 4 | Channel #1, Source type, Address type | 0x00 |
| 5 | Channel #1, Source location, Address LSB | 0xFE |
| 6 | Channel #1, Source location, Address LSB | 0xCA |
| 7 | Channel #1, Source location, Address MSB | 0xAD |
| 8 | Channel #1, Source location, Address MSB | 0xDE |
| 9 | Channel #1, Data type size<br>=> uint32 @ 8-bit RAM width = 4 | 0x04 |
| 10 | Channel #2, Source type, Address type | 0x00 |
| 11 | Channel #2, Source location, Address LSB | 0xBB |
| 12 | Channel #2, Source location, Address LSB | 0xAA |
| 13 | Channel #2, Source location, Address MSB | 0x99 |

| | | |
|---|---|---|
| 14 | Channel #2, Source location, Address MSB | 0x88 |
| 15 | Channel #2, Data type size<br>=> uint16 @ 8-bit RAM width = 2 | 0x02 |
| 16 | Trigger data type, int32<br>=> Bit 7 is always set because of “New Scope Version”<br>=> Bit 5 Sign Bit is set because of signed data type<br>=> Bit 4 Type Bit is not set because of integer data type<br>=> Bit 0-3 is the byte size of uint32 = 4 | 0xA4 |
| 17 | Trigger source type, Address | 0x00 |
| 18 | Trigger source location, Address LSB | 0x78 |
| 19 | Trigger source location, Address LSB | 0x56 |
| 20 | Trigger source location, Address MSB | 0x34 |
| 21 | Trigger source location, Address MSB | 0x12 |
| 22 | Trigger level LSB | 0x70 |
| 23 | Trigger level LSB | 0x11 |
| 24 | Trigger level MSB | 0x01 |
| 25 | Trigger level MSB | 0x00 |
| 26 | Trigger delay LSB | 0x58 |
| 27 | Trigger delay LSB | 0x02 |
| 28 | Trigger delay MSB | 0x00 |
| 29 | Trigger delay MSB | 0x00 |
| 30 | Trigger edge, FALLING | 0x00 |
| 31 | Trigger mode, NORMAL | 0x01 |

# List of Tables

## List of Abbreviations

| | |
|---|---|
| SDA | Scope Data Array |
| DSS | Data Set Size |

## List of References

LNet (version 5) documentation, X2C, http://x2c.lcm.at

## Legal Notice

Linz Center of Mechatronics GmbH

Altenberger Straße 69
4040 Linz
Austria

Phone: +43 (0)732 / 2468 – 6002
Fax: +43 (0)732 / 2468 – 6005

Company Data

Technical Research and Product Development
Landes- als Handelsgericht Linz
FN 207547 t

UID-Nr.: ATU52054707