LINZ CENTER OF MECHATRONICS GMBH

LNet Documentation

March 3, 2017

Contents

1 Version History **2**

2 Introduction **3**

3 The LNet frame **3**
- 3.1 SYN . . . 3
- 3.2 SIZE . . . 3
- 3.3 Slave NODE ID . . . 3
- 3.4 DATA . . . 4
- 3.5 CRC . . . 4
- 3.6 In-Frame SYN Detection . . . 4

4 Services **5**
- 4.1 Get Device Info . . . 6
- 4.2 Get Target State . . . 7
- 4.3 Set Target State . . . 7
- 4.4 Erase Flash . . . 8
- 4.5 Get Block Data . . . 9
- 4.6 Put Block Data . . . 9
- 4.7 Get RAM Block . . . 10
- 4.8 Put RAM Block . . . 10
- 4.9 Get Flash Block . . . 11
- 4.10 Put Flash Block . . . 11
- 4.11 Load Parameter . . . 11
- 4.12 Save Parameter . . . 12
- 4.13 Reboot . . . 12

5 Variable Types **13**

6 DSP states **13**

7 Errors **13**

8 Examples **15**
- 8.1 Get Device Info . . . 15
- 8.2 Erase flash . . . 17
- 8.3 Get Block Data . . . 18
- 8.4 Put Block Data . . . 19
- 8.5 Get RAM Block . . . 20
- 8.6 Put RAM Block . . . 21
- 8.7 Reboot . . . 22

1 Version History

LNet history

Version	Changes
4	Updated data structure in getDeviceInfo service
5	Added TableStruct address to getDeviceInfo service Added services getTargetState and setTargetState Added 2 error identifiers

Document history

Version	Changes
1	Created
2	Fixed getDeviceInfo service (see 4.1) description
3	Updated document to LNet version 5
4	Fixed LSB & MSB order in getDeviceInfo example Fixed size and added missing TableStruct address in getDeviceInfo example Update In-Frame SYN detection for better understanding Added DSP states description
5	Fixed response service data in 4.2

2 Introduction

LNet is a master-slave, multi-node protocol. This means the master sends requests to one or more slave nodes in the network and receives response frames from each node.

LNet uses different commands (e.g. read from memory, write to memeory, reboot etc.). Each of these commands uses an unique, so-called 'Service Identifier' (short: Service-ID). The master sends the Service-ID and the Service data (if required) to the slave. The slave processes the service and sends back the same Service-ID, an Error-ID to notify the master about the successful or failed service and (if required) the Service Data (e.g. the contents of a 'read RAM' service).
All values are sent LSB first.

3 The LNet frame

The basic structure of an LNet frame consists of 5 parts:

SYN	SIZE	NODE	DATA	CRC

3.1 SYN

Size: 1 byte
Indicates the start of a frame. This byte is always 0x55.
The value 0x02 is also reserved for future purposes. These 2 reserved values must be specially treated if they occur in any other frame area than in SYN. (see 3.6)

3.2 SIZE

Size: 1 byte
The number of data bytes.
Optional fill-bytes (see 3.6) will not be added to SIZE.

3.3 Slave NODE ID

Size: 1 byte
Identifies the slave to which the master wants to send the frame.
The master sets this byte to the slave ID it wants to communicate with and the slave sets this byte to its own ID when responding to the master.

3.4 DATA

Size: up to 255 bytes
Contains the data. The data area is also divided into several parts. Master and slave use different data structures.

Master data structure (request frame)

Data byte	Name	Description
0	Service ID	Identifies which service will be used
1 ... n	Service data	(optional) service data

Slave data structure (response frame)

Data byte	Name	Description
0	Service ID	Returns the service ID, which was received from master
1	Error-ID	Returns error identifier
2 ... n	Service data	(optional) service data

For more details regarding the services see 4.

3.5 CRC

Size: 1 byte
Contains the checksum.
The checksum is calculated by adding all frame bytes (except fill-bytes, see 3.6) modulo 256.

3.6 In-Frame SYN Detection

LNet has 2 reserved key values: 0x55 and 0x02.
To avoid misinterpretation within SIZE, NODE, DATA or CRC area, these values must be differently handled.
If any of these key values occur within SIZE, NODE or DATA area, a 0x00 'fill-byte' will be added which will not be counted to data size and not be used in checksum calculation.
The checksum will be inverted if it equals one of these key values, so:

- Checksum 0x55 $\Rightarrow$ 0xAA
- Checksum 0x02 $\Rightarrow$ 0xFD

An example if the keywords appear in DATA section:

SYN	SIZE	NODE	DATA	CRC
0x55	0x01	0x01	0x55	0xAC

turns into:

SYN	SIZE	NODE	DATA	FILL	CRC
0x55	0x01	0x01	0x55	0x00	0xAC

4 Services

LNet uses services to process different tasks as: read from memory, write to memory, reboot etc. Every service uses an unique, 1 byte wide, Service-ID.
The master sends the Service-ID and up to 254 bytes service data (maximum frame SIZE minus 1 byte for Service-ID). There are services, which don't required any service data (e.g. getDeviceInfo).
The slave responds with the same Service-ID sent by the master and adds a 1 byte wide Error-ID. This will tell the master a successful or failed service procedure. The slave also adds service data but only up to 253 bytes. (maximum frame SIZe - 2 bytes for Service-ID and Error-ID)
Following the list and description of all LNet services. Please note that some services are not available on all processor types. In this case a 'Service not available' error will be returned (for a list of errors see 7).

List with service identifiers and -names:

Service-ID	Service name
0x00	Get Device Info
0x01	Get Target State
0x02	Set Target State
0x04	Erase Flash
0x07	Get Block Data
0x08	Put Block Data
0x09	Get RAM Block
0x0A	Put RAM Block
0x0B	Get Flash Block
0x0C	Put Flash Block
0x11	Load Parameter
0x12	Save Parameter
0x19	Reboot

4.1 Get Device Info

Service-ID: 0x00

Returns information from target system.

Request service data:
No service data.

Response service data:

Data byte	Description
0 - 1	Monitor version
2 - 3	Application version
4	Maximum target frame size
5 - 6	Processor identifier
7 - 15	Monitor date as ASCII string
16 - 19	Monitor time as ASCII string
20 - 28	Application date as ASCII string
29 - 32	Application time as ASCII string
33	DSP state (see DSP states)
34 - 35	Event type
36 - 39	Event identifier
40 - 43	TableStruct address as 32 bit value

4.2 Get Target State

Service-ID: 0x01

Returns current target state which contains the following parameters:

- DSP state (see DSP states)

Request service data:
No service data.

Response service data:

Data byte	Description
0	DSP state

4.3 Set Target State

Service-ID: 0x02

Sets following target parameters:

- DSP state (see DSP states)

Request service data:

Data byte	Description
0	DSP state

Response service data:
No response service data.

4.4 Erase Flash

Service-ID: 0x04

Erases flash memory sectors.

Request service data:

Data byte	Description
0 - n	Erase sector mask (size depends on flash organization)

Response service data:
No service data.

Each bit in the mask represents a flash sector, for example:

- byte #0, bit 3 = flash sector 3 (or C)
- byte #1, bit 2 = flash sector 5 (or E)

Each set bit represents a flash sector which will be erased.
A minimum of 2 bytes must be sent even if the device has less than 9 sectors.

4.5 Get Block Data

Service-ID: 0x07

Reads block data.

Request service data:

Data byte	Description
0 ... n	block address (size depends on target memory width)

Response service data:

Data byte	Description
0 ... n	block data (depends on block type)

4.6 Put Block Data

Service-ID: 0x08

Writes block data.

Request service data:

Data byte	Description
0 ... n	Block address (size depends on target memory width)
n+1 ... m	block data (length & content depends on block type)

Response service data:
No service data.

4.7 Get RAM Block

Service-ID: 0x09

Reads values from target memory adddress.

Request service data:

Data byte	Description
0 ... n	Memory address (size depends on target memory width)
n+1	Number of bytes to read
n+2	Value data type (see 5 for details)

Response service data:

Data byte	Description
0 ... n	Values

4.8 Put RAM Block

Service-ID: 0x0A

Writes values to target memory address.

Request service data:

Data byte	Description
0 ... n	Memory address (size depends on target memory width)
n+1	Value data type (see 5 for details)
n+2 ... m	Bytes to write to target

Response service data:
No service data.

4.9 Get Flash Block

Service-ID: 0x0B

Reads values from target flash memory address.

The service uses the same data frame structure as 'Get RAM Block'.
The only difference is the usage of Service-ID 0x0B instead of 0x09.
Please refer to 4.7 for more details regarding to data frame structure

4.10 Put Flash Block

Service-ID: 0x0C

Writes values to target flash memory address.

The service uses the same data frame structure as 'Put RAM Block'.
The only difference is the usage of Service-ID 0x0C instead of 0x0A.
Please refer to 4.8 for more details regarding to data frame structure

4.11 Load Parameter

Service-ID: 0x11

Reads block data by using an unique parameter ID.
It uses the same functionality as service 'Get block data' (see 4.5).
The difference is to use a 16 bit unique parameter ID instead of the block address.
This unique parameter ID is linked with a block in the current frame program (application) and must be especially implemented.

Request service data:

Data byte	Description
0 ... 1	Unique parameter ID for a block

Response service data:

Data byte	Description
0 ... n	Block data (depends on block type)

4.12 Save Parameter

Service-ID: 0x12

Writes block data by using an unique parameter ID.
It uses the same functionality than service 'Put Block Data' (see 4.6) with the difference to use a 16 bit unique parameter ID instead of the block address.
This unique parameter ID is linked with a block in the current frame program (application) and must be especially implemented.

Request service data:

Data byte	Description
0 ... 1	Unique parameter ID for a block
2 ... n	Block data (length depends on block type)

Response service data:
No service data.

4.13 Reboot

Service-ID: 0x19

Reboots the target.

Request service data:
No service data.

Response service data:
No service data.

In case of success, this service will not send a response frame.

5 Variable Types

Some services require extra information about how to treat the received/sent data. The data type value is defined as the number of bytes required to cover the data type. This data type values are currently implemented:

Value	Data type width
0x01	8 bit
0x02	16 bit
0x04	32 bit

6 DSP states

The DSP state indicates the current state of X2C.
Following states are being supported by X2C:

State name	Value	Description
MONITOR	0x00	Monitor runs on target but no application
APPLICATION LOADED	0x01	Application runs on target ⇒ X2C Update function is being executed
IDLE	0x02	Application is idle ⇒ X2C Update Function is not being executed
INIT	0x03	Application is initializing and usually changes to state 'IDLE' after being finished
APPLICATION RUNNING - POWER OFF	0x04	Application is running with disabled power electronics
APPLICATION RUNNING - POWER ON	0x05	Application is running with enabled power electronics

7 Errors

If the target system detects an error condition either in the protocol header or in the data area, an Error-ID is returned.
The Error-ID is located at data byte #1 ('Error-ID' in slave response frame).
This is a list of all possible Protocol- & service error identifiers:

Error-ID	Description
0x00	No error
0x13	Checksum error
0x14	Format error
0x15	Size too large
0x21	Service not available
0x22	Invalid DSP state
0x30	Flash write error
0x31	Flash write protect error
0x40	Invalid Parameter ID
0x41	Invalid Block ID
0x42	Parameter limit error
0x43	Parameter table not initialized
0x50	Power-on error

8 Examples

Examples with whole LNet frame for each service.
These examples were done using an TMS320F28035 processor (this processor type does have 32 bit memory address width).

8.1 Get Device Info

Read the system's device info.

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x01	Data frame size
2	0x01	Slave node ID
3	0x00	Service-ID for 'get Device Info'
4	0x57	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x2E	Data frame size
2	0x01	Slave node ID
3	0x00	Service-ID for 'get Device Info'
4	0x00	No error
5	0x04	Monitor version (LSB)
6	0x00	Monitor version (MSB)
7	0x00	Application version (LSB)
8	0x00	Application version (MSB)
9	0xFF	Maximum target frame size
10	0x01	Processor identifier (LSB)
11	0x51	Processor identifier (MSB)
12	0x41	Monitor date (ASCII character 'A')
13	0x70	Monitor date (ASCII character 'p')
14	0x72	Monitor date (ASCII character 'r')
15	0x31	Monitor date (ASCII character '1')
16	0x38	Monitor date (ASCII character '8')
17	0x32	Monitor date (ASCII character '2')
18	0x30	Monitor date (ASCII character '0')
19	0x31	Monitor date (ASCII character '1')
20	0x32	Monitor date (ASCII character '2')
21	0x31	Monitor time (ASCII character '1')
22	0x31	Monitor time (ASCII character '1')

23	0x35	Monitor time (ASCII character ’5’)
24	0x35	Monitor time (ASCII character ’5’)
25	0x20	Application date (ASCII character ’ ’)
26	0x20	Application date (ASCII character ’ ’)
27	0x20	Application date (ASCII character ’ ’)
28	0x20	Application date (ASCII character ’ ’)
29	0x20	Application date (ASCII character ’ ’)
30	0x20	Application date (ASCII character ’ ’)
31	0x20	Application date (ASCII character ’ ’)
32	0x20	Application date (ASCII character ’ ’)
33	0x20	Application date (ASCII character ’ ’)
34	0x20	Application time (ASCII character ’ ’)
35	0x20	Application time (ASCII character ’ ’)
36	0x20	Application time (ASCII character ’ ’)
37	0x20	Application time (ASCII character ’ ’)
38	0x00	DSP state
39	0x00	Event type (LSB)
40	0x00	Event type (MSB)
41	0x00	Event identifier(LSB)
42	0x00	Event identifier(LSB)
43	0x00	Event identifier(MSB)
44	0x00	Event identifier(MSB)
45	0x00	TableStruct address (LSB)
46	0x00	TableStruct address (LSB)
47	0x00	TableStruct address (MSB)
48	0x00	TableStruct address (MSB)
49	0x96	Checksum

8.2 Erase flash

Erase flash sectors 1 (B) and (H).

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x03	Data frame size
2	0x01	Slave node ID
3	0x04	Service-ID for ’erase Flash’
4	0x82	Erase sector mask, byte #0
5	0x00	Erase sector mask, byte #1
6	0xDF	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x02	Data frame size
2	0x00	FILL BYTE
3	0x01	Slave node ID
4	0x04	Service-ID for ’erase Flash’
5	0x00	No error
8	0x5C	Checksum

8.3 Get Block Data

Read block data from a Gain block, 16 bit implementation.
The block is located at address 0x9568.
It holds a gain value of 0.75 (Q-value = 0x6000, shift factor = 15).

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x05	Data frame size
2	0x01	Slave node ID
3	0x07	Service-ID for 'Get Block Data'
4	0x68	Block address (byte #0)
5	0x95	Block address (byte #1)
6	0x00	Block address (byte #2)
7	0x00	Block address (byte #3)
8	0x5F	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x05	Data frame size
2	0x01	Slave node ID
3	0x07	Service-ID for 'Get Block Data'
4	0x00	No error
5	0x00	Gain value (byte #0)
6	0x60	Gain value (byte #1)
7	0x0F	Gain shift factor
8	0xD1	Checksum

8.4 Put Block Data

Write block data to a Gain block, 16 bit implementation.
The block is located at address 0x9568.
Write the value 0.25 (Q-value = 0x2000, shift factor = 15).

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x08	Data frame size
2	0x01	Slave node ID
3	0x08	Service-ID for 'Put Block Data'
4	0x68	Block address (byte #0)
5	0x95	Block address (byte #1)
6	0x00	Block address (byte #2)
7	0x00	Block address (byte #3)
8	0x00	Gain value (byte #0)
9	0x20	Gain value (byte #1)
10	0x0F	Gain shift factor
11	0x92	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x02	Data frame size
2	0x00	FILL BYTE
3	0x01	Slave node ID
4	0x08	Service-ID for 'Put Block Data'
5	0x00	No error
8	0x60	Checksum

8.5 Get RAM Block

Read 2x 32 bit values from memory address 0x9602.
The values 0xBA44D1DC and 0x7E208699 are stored at this location.

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x07	Data frame size
2	0x01	Slave node ID
3	0x09	Service-ID for ’get RAM Block’
4	0x02	Memory address (byte #0)
5	0x00	FILL BYTE
6	0x96	Memory address (byte #1)
7	0x00	Memory address (byte #2)
8	0x00	Memory address (byte #3)
9	0x08	Number of bytes to read (2x 32 bit = 8 bytes)
10	0x04	Value data type (see 5)
12	0x0A	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x0A	Data frame size
2	0x01	Slave node ID
3	0x09	Service-ID for ’get RAM Block’
4	0x00	No error
5	0xDC	Value #1, byte #0
6	0xD1	Value #1, byte #1
7	0x44	Value #1, byte #2
8	0xBA	Value #1, byte #3
9	0x99	Value #2, byte #0
10	0x86	Value #2, byte #1
11	0x20	Value #2, byte #2
12	0x7E	Value #2, byte #3
13	0xD1	Checksum

8.6 Put RAM Block

Write 3x 16 bit values to memory address 0x9600.
The values to write are 0xBEEF, 0xCAFE, 0x5502.

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x0C	Data frame size
2	0x01	Slave node ID
3	0x0A	Service-ID for 'put RAM Block'
4	0x00	Memory address (byte #0)
5	0x96	Memory address (byte #1)
6	0x00	Memory address (byte #2)
7	0x00	Memory address (byte #3)
8	0x02	Value data type (see 5)
9	0x00	FILL BYTE
10	0xEF	Value #1, byte #0
11	0xBE	Value #1, byte #1
12	0xFE	Value #2, byte #0
13	0xCA	Value #2, byte #1
14	0x02	Value #3, byte #0
15	0x00	FILL BYTE
16	0x55	Value #3, byte #1
17	0x00	FILL BYTE
18	0xD0	Checksum

Response frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x02	Data frame size
2	0x00	FILL BYTE
3	0x01	Slave node ID
4	0x0A	Service-ID for 'put RAM Block'
5	0x00	No error
6	0x62	Checksum

8.7 Reboot

Reboot device.

Request frame:

Byte	Value	Description
0	0x55	Start of frame
1	0x01	Data frame size
2	0x01	Slave node ID
3	0x19	Service-ID for ’reboot’
4	0x70	Checksum

Response frame:
No response frame if reboot was successful.