



Technical specification

Studer Public Protocol for Xcom-CAN

Date : 23.06.2020
Version : V1.5.0

CONTENTS

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | Conventions used in this document..... | 3 |
| 1.2 | List of acronyms..... | 3 |
| 2 | CAN Interface..... | 4 |
| 3 | CAN Frame format..... | 4 |
| 3.1 | Addressing Studer devices | 5 |
| 3.2 | Multicast accesses..... | 6 |
| 3.3 | Unicast Accesses | 6 |
| 3.4 | Response delay..... | 6 |
| 4 | User infos | 8 |
| 4.1 | Read user info service | 8 |
| 4.1.1 | Example | 9 |
| 5 | Parameters..... | 10 |
| 5.1 | Read parameter service..... | 10 |
| 5.1.1 | Example | 11 |
| 5.2 | Write parameter service | 12 |
| 5.2.1 | Example (write in flash and RAM) | 13 |
| 5.2.2 | Example (write in RAM only: unsaved value) | 14 |
| 5.3 | Change value of parameters on the Xtender inverter..... | 15 |
| 5.4 | Cyclic write of parameters | 15 |
| 5.5 | Data encoding | 15 |
| 5.5.1 | "ENUM" format encoded as a float | 16 |
| 5.5.2 | "BOOL" format encoded as a float | 16 |
| 5.5.3 | "SIGNAL" format encoded as a float | 16 |
| 5.5.4 | "HOUR" format encoded as a float | 16 |
| 5.5.5 | "DAYS of WEEK" format encoded as a float | 16 |
| 5.6 | Message notifications | 17 |
| 5.7 | Message notification service..... | 17 |
| 5.7.1 | Example | 17 |
| 6 | Error codes..... | 18 |

1 INTRODUCTION

This technical specification describes the "Studer Public Protocol" for the Xcom-CAN device (Refer to Xcom-CAN Technical Specification for more information). This protocol enables the control of a Studer system from a third party device (PLC, SCADA, etc.) using a CAN 2.0B based communication interface. Using this protocol, it is possible to:

- Read user info
- Read parameters
- Write parameters
- Receive message notifications

1.1 CONVENTIONS USED IN THIS DOCUMENT

Numbers starting with a "0x" prefix are hexadecimal numbers, otherwise, there are decimal numbers.

1.2 LIST OF ACRONYMS

RCC The Studer Innotec remote control used to configure the Xtender system.

DTE Data Terminal Equipment, the PC or controller system that will communicate with the Xcom-CAN.

2 CAN INTERFACE

The Xcom-CAN Studer Public Protocol uses CAN 2.0B frames (29 bits identifiers).

The speed of the bus can be selected using the DIP switches inside the module. Also, the PIN configuration of the connector can be selected using the jumper configuration. Please refer to the Xcom-CAN Technical Specification for more information.

All Frames are in big endian, so MSB (Most Significant Byte) is sent first on the CAN interface.

3 CAN FRAME FORMAT

The Xcom-CAN and the DTE exchange frames consist of a header of 29 bits followed by a variable number of data bytes depending on frame type.

| CAN Identifier (29 bits) | | | | CAN Data (up to 8 bytes) |
|--------------------------|----------|---------|--------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | Specific to service |
| 10 bits | 10 bits | 3 bits | 6 bits | |

DST ADDR

The destination address, e.g. 101 for the first Xtender, see chapter 3.1 for more details.

SRC ADDR

The source address is the address of the client device, the response will return to this address.

SERVICE

The service that the frame is intended for. The value could be:

- 0x00** : User info read service
- 0x01** : Parameter read service
- 0x02** : Parameter write service
- 0x03** : Message notification service
- 0x04 – 0x07** : Reserved

FLAGS

The **BIT0** flag is the Least Significant Bit (LSB) of the **FLAGS** field. The flags are described hereafter:

- BIT0** : Error → indicates an error in a response
- BIT1** : Reserved
- BIT2** : Reserved
- BIT3** : Reserved
- BIT4** : Reserved
- BIT5** : Reserved

SPECIFIC TO SERVICE

Specific to used service, see description under each service

3.1 ADDRESSING STUDER DEVICES

| Address in decimal | Devices | Remarks |
|--------------------|---|--|
| 100 | Virtual address to access all XTH, XTM and XTS | See section "multicast accesses" Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter |
| 101 to 109 | A single XTH, XTM or XTS inverter/charger | Ordered by the index displayed on the RCC Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter • Message as source address |
| 191 to 193 | Virtual address to access all XTH, XTM and XTS present on the same phase: 191 for L1 192 for L2 193 for L3 | See section "multicast accesses" Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter |
| 300 | Virtual address to access all VarioTrack | See section "multicast accesses" Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter |
| 301 to 315 | A single VarioTrack | Ordered by the index displayed on the RCC Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter • Message as source address |
| 500 | Virtual address for the RCC group | Used in: <ul style="list-style-type: none"> • Message as destination address |
| 501 to 505 | A single RCC device (RCC, Xcom-232i, Xcom-Can in Studer Public) | Used in: <ul style="list-style-type: none"> • Message as source address |
| 600 | Virtual address to access all BSP (only one BSP per installation) | See section "multicast accesses" Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter |
| 601 | A single BSP | Only one BSP per installation Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter • Message as source address |
| 700 | Virtual address to access all VarioString | See section "multicast accesses" Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter |

| | | |
|------------|----------------------|--|
| 701 to 715 | A single VarioString | Ordered by the index displayed on the RCC Used in: <ul style="list-style-type: none"> • Read User Info • Read Parameter • Write Parameter • Message as source address |
|------------|----------------------|--|

3.2 MULTICAST ACCESSES

Multicast accesses enable the possibility to access a group of devices of the same kind. So it is possible to access all Xtenders (XTH, XTM, XTS or a mix), all VarioTracks or all VarioStrings. In order to do it, you need to use Multicast address as described in the previous table. For reasons of uniformity, the BSP can also be accessed with a multicast address.

A write operation to a multicast address will have the effect to change the parameter value on all devices of the same kind. A read operation will return the value of the first device displayed in the RCC list in the group you are targetted.

As an exemple, if you write a parameter at address "100", all Xtenders will have the corresponding value changed according to your write operation. If you read a parameter or a user information at address "100", the system will return the value of the Xtender from address "101", which is the first Xtender.

Note that the same behavior is guaranteed with phase group accesses, a read parameter/user info operation will return the value of the first Xtender on this phase and a write parameter operation acts on all the Xtenders of the same phase (as defined physically by the jumper on the PCB inside the Xtender).

As an example, if you write a parameter at address "191", all Xtenders present on phase L1 will have the corresponding value changed according to your write operation. If you read a parameter or a user information at address "191", the system will return the value of the first Xtender present on phase L1.

3.3 UNICAST ACCESSES

Unicast accesses enable the possibility to access a single device in the installation. So any single Xtender, VarioTrack or VarioString can be access using Unicast address.

As an exemple, assume that you have 3 Xtenders in your system. Imagine that you want to access the second one displayed in the "System Info" of your RCC, you will need to use Unicast address "102".

3.4 RESPONSE DELAY

The response delay of the Xcom-CAN could be up to 1 second. This is a good value for a timeout in the client implementation.

The response delay depends on the internal Studer bus load (number of devices, number of RCC, values displayed on the RCC, etc.). The use of the data logger on other RCCs could cause a periodic increase of the response delay every 60 seconds as it needs to do more transactions on the Studer CAN bus.

It is strongly recommended not to spam the Xcom-CAN with multiple requests. Even if the Xcom-CAN has a frame buffer, the response will not be faster because of internal Studer bus load. The correct way to communicate with the Xcom-CAN is to send a request and to wait for the response before sending the next request. If no response comes from Xcom-CAN after a delay of 1 second, we can consider that the timeout is over, and another request can be send.

4 USER INFOS

The available user information is the same as the values that can be chosen to be displayed on the RCC. This user information gives the current state of the system. The user information cannot be modified, and their values change during the operation of the system.

The fields that are specific to parameter frames are:

INFO ID : The identifier of a user info (see "Xtender serial protocol appendix")
DATA : Value of the user info in 32-bit Floating point format (see IEEE 754)

4.1 READ USER INFO SERVICE

The client side sends a request frame with the following format:

| CAN Identifier | | | | CAN Data |
|----------------|----------------|---------|--------|------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | INFO ID |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x0 | 0x00 | 0x0000 to 0xFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x0 (read user info service)
FLAGS : 0x00
INFO ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")

In case of success, the Xcom-CAN responds with a frame with the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | INFO ID | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x0 | 0x00 | 0x0000 to 0xFFFF | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x0 (read user info)
FLAGS : 0x00 (success)
INFO ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
VALUE : 0x00000000 to 0xFFFFFFFF

In case of error, the Xcom-CAN reponds with a frame with the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | INFO ID | ERROR CODE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x0 | 0x01 | 0x0000 to 0xFFFF | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x0 (read user info service)
FLAGS : 0x01 (error)
INFO ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
ERROR CODE : 0x00000000 to 0xFFFFFFFF (see chapter 6)

4.1.1 Example

Read the battery voltage on the first VarioTrack.

Request frame :

| CAN Identifier | | | | CAN Data | |
|----------------|----------|--------------------|--------|----------|--|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | INFO ID | |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | |
| 301 (VT1) | 1 | 0 (Read user info) | 0 | 11000 | |
| 0x12D | 0x001 | 0x0 | 0x00 | 0x2AF8 | |

Response frame :

| CAN Identifier | | | | CAN Data | |
|----------------|-----------|--------------------|--------|----------|------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | INFO ID | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 1 | 301 (VT1) | 0 (Read user info) | 0 | 11000 | 12.6 |
| 0x001 | 0x12D | 0x0 | 0x00 | 0x2AF8 | 0x4149999A |

The battery voltage of the first VarioTrack is 12.6V.

5 PARAMETERS

All parameters accessible from RCC can also be modified with the protocol. The behaviour is the same as if a physical person changes the value with the remote control buttons.

The fields that are specific to parameter frames are:

- PARAM ID** : The identifier for a parameter (see "Xtender serial protocol appendix")
- PART** : The parameter part
- For Write parameter:
- 0x00** : write parameter value in Flash (and RAM)
 - 0x04** : write parameter value in RAM only
- For Read parameter:
- 0x00** : read parameter value from Flash
 - 0x01** : read parameter minimum possible value
 - 0x02** : read parameter maximum possible value
 - 0x04** : read parameter value from Flash (not possible to read from RAM directly)
- DATA** : The parameter value in case of a read and value to write in case of a write.
The data are always in 32-bit Floating point format (see IEEE 754).

5.1 READ PARAMETER SERVICE

The client side sends a request frame with the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x1 | 0x00 | 0x0000 to 0xFFFF | 0x00 |

- DST ADDR** : 0x000 to 0x3FF
- SRC ADDR** : 0x000 to 0x3FF
- SERVICE** : 0x1 (read parameter service)
- FLAGS** : 0x00
- PARAM ID** : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
- PART** : 0x00

In case of success, the Xcom-CAN responds with a frame with the following format:

| CAN Identifier | | | | CAN Data | | |
|----------------|----------------|---------|--------|------------------|--------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x1 | 0x00 | 0x0000 to 0xFFFF | 0x00 | 0x00000000 to 0xFFFFFFFF |

- DST ADDR** : 0x00 to 0x3FF
- SRC ADDR** : 0x00 to 0x3FF
- SERVICE** : 0x1 (read parameter service)
- FLAGS** : 0x00 (success)
- PARAM ID** : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
- PART** : 0x00
- VALUE** : 0x00000000 to 0xFFFFFFFF

In case of error, the Xcom-CAN responds with a frame with the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | ERROR CODE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x1 | 0x01 | 0x0000 to 0xFFFF | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x1 (read parameter service)
FLAGS : 0x01 (error)
PARAM ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
ERROR CODE : 0x00000000 to 0xFFFFFFFF (see chapter 6)

5.1.1 Example

Read the battery charge current parameter on the first VarioTrack.

Request frame :

| CAN Identifier | | | | CAN Data | |
|----------------|----------|--------------------|--------|----------|--------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits |
| 301 (VT1) | 1 | 1 (Read parameter) | 0 | 10002 | 0 |
| 0x12D | 0x001 | 0x1 | 0x00 | 0x2712 | 0x00 |

Response frame :

| CAN Identifier | | | | CAN Data | | |
|----------------|----------|--------------------|--------|----------|--------|------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 1 | 301 | 1 (Read parameter) | 0 | 10002 | 0 | 80.0 |
| 0x001 | 0x12D | 0x1 | 0x00 | 0x2712 | 0x00 | 0x42A00000 |

The battery charge current of the first VarioTrack is 80.0A.

5.2 WRITE PARAMETER SERVICE

The client side sends a request frame with the following format:

| CAN Identifier | | | | CAN Data | | |
|----------------|----------------|---------|--------|------------------|--------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x2 | 0x00 | 0x0000 to 0xFFFF | 0x00 | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x2 (write parameter service)
FLAGS : 0x0
PARAM ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
PART : 0x00 (write in Flash) or 0x04 (write in RAM only: unsaved value)
VALUE : 0x00000000 to 0xFFFFFFFF

In case of success, The Xcom-CAN responds with a frame with the following format:

| CAN Identifier | | | | CAN Data | | |
|----------------|----------------|---------|--------|------------------|--------|---------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x2 | 0x00 | 0x0000 to 0xFFFF | 0x00 | 0x00 |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x2 (write parameter)
FLAGS : 0x00 (success)
PARAM ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
PART : 0x00 (write in Flash) or 0x04 (write in RAM only: unsaved value)
VALUE : always 0, reserved

In case of error, the Xcom-CAN responds with a frame with the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | ERROR CODE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 0x000 to 0x3FF | 0x000 to 0x3FF | 0x2 | 0x01 | 0x0000 to 0xFFFF | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 0x00 to 0x3FF
SRC ADDR : 0x00 to 0x3FF
SERVICE : 0x2 (write parameter service)
FLAGS : 0x01 (error)
PARAM ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")
ERROR CODE : 0x00000000 to 0xFFFFFFFF (see chapter 6)

5.2.1 Example (write in flash and RAM)

Write the battery charge current parameter on the first VarioTrack.

Request frame :

| CAN Identifier | | | | CAN Data | | |
|----------------|----------|---------|--------|----------|--------|------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 301 (VT1) | 1 | 2 | 0 | 10002 | 0 | 50.0 |
| 0x12D | 0x001 | 0x2 | 0x00 | 0x2712 | 0x00 | 0x42480000 |

Response frame :

| CAN Identifier | | | | CAN Data | | |
|----------------|-----------|---------|--------|----------|--------|---------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 1 | 301 (VT1) | 2 | 0 | 10002 | 0 | 0 |
| 0x12D | 0x001 | 0x2 | 0x00 | 0x2712 | 0x00 | 0x00 |

5.2.2 Example (write in RAM only: unsaved value)

Write the battery charge current parameter on the first VarioTrack **WITHOUT** saving it into the Flash memory.

Request frame :

| CAN Identifier | | | | CAN Data | | |
|----------------|----------|---------|--------|----------|-------------|------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 301 (VT1) | 1 | 2 | 0 | 10002 | 4 | 50.0 |
| 0x12D | 0x001 | 0x2 | 0x00 | 0x2712 | 0x04 | 0x42480000 |

Response frame :

| CAN Identifier | | | | CAN Data | | |
|----------------|-----------|---------|--------|----------|-------------|---------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | PARAM ID | PART | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 8 bits | 32 bits |
| 1 | 301 (VT1) | 2 | 0 | 10002 | 4 | 0 |
| 0x001 | 0x12D | 0x2 | 0x00 | 0x2712 | 0x04 | 0x00 |

5.3 CHANGE VALUE OF PARAMETERS ON THE XTENDER INVERTER

Changing parameters when the inverters are in operation should be done carefully. The modification of parameters can restart the corresponding algorithm inside the inverter. For example, the change of a delay can restarts the timer attached to it.

5.4 CYCLIC WRITE OF PARAMETERS

When you are using the RCC remote control, the Xtender inverter/charger, VarioTrack and VarioString MPPT solar chargers store their parameter values in a non-volatile flash memory. Because of the endurance of this memory, the number of writes on a single parameter is only guaranteed for 1000 write operations.

To allow the cyclic write of parameters without count limit, the "unsaved value" feature has been created. It enables to write directly in RAM (volatile memory) without affecting the flash. You can write parameters in RAM by setting the "Part" field to the value 0x4. We strongly recommend, when you are using the Xcom-CAN Public protocol to control the installation, to write in RAM instead of flash using the "unsaved value" part.

A read operation with the part set to "unsaved value" will give the values stored in the flash memory (not what is in the RAM). Also, after a reset, the values in flash memory will be copied into the RAM.

To use the "unsaved value" feature you must have a release \geq R616 beta. This include:

- XT firmware version \geq 1.6.12
- VT firmware version \geq 1.6.14
- VS firmware version \geq 1.6.14
- BSP firmware version \geq 1.6.12
- RCC and Xcom-232i firmware version \geq 1.6.14

The "unsaved value" functionality is not supported by the following products:

- Xcom-SMS
- Xcom-MS

5.5 DATA ENCODING

In appendix, there are tables presenting the user information and the parameters of each device. The format for each information/parameter is specified. All data are encoded according to the standard format IEEE 754-2008: single precision floating point. There is plenty of documentation regarding this format on the Internet. However, the format is explained for your convenience below:

| sign | exponent | mantissa |
|-------|----------|----------|
| 1 bit | 8 bits | 23 bits |

IEEE 754-2008 single precision 32 bits floating point format

As an example, the decimal value 0.0 is coded 0x00000000 and the decimal value 1.0 is coded 0x3F800000.

Below we have briefly explained all Studer specific formats.

5.5.1 "ENUM" format encoded as a float

An "ENUM" is an enumeration and can basically take one value in between a set of possible values. Each value has its own signification. For parameter, ENUM values are always power of two values (e.g. 0, 1, 2, 4, 8, etc.). For user info, ENUM values are always incremental values (e.g. 0, 1, 2, 3, 4, 5, 6, etc.). Remember that these values are encoded as a float, so :

0 will be encoded as 0.0 = 0x00000000,
1 will be encoded as 1.0 = 0x3F800000,
2 will be encoded as 2.0 = 0x40000000,
3 will be encoded as 3.0 = 0x40400000,
etc...

5.5.2 "BOOL" format encoded as a float

The "BOOL" format represents "TRUE" and "FALSE", encoded respectively as "+1.0" and "0.0". So:

"TRUE" will be encoded as 1.0 = 0x3F800000,
"FALSE" will be encoded as 0.0 = 0x00000000

5.5.3 "SIGNAL" format encoded as a float

The Signal format (e.g. parameter {1468}) is coded as a float. To send an active signal, you must write the value "1.0" (encoded 0x3F800000) as the parameter value.

5.5.4 "HOUR" format encoded as a float

The hour format encoding is in minutes beginning at 00:00 and terminating at 23:59. There is no field available for seconds. So:

13:41 = 13*60 + 41 = 821 minutes will be encoded as 821.0 = 0x444D4000

5.5.5 "DAYS of WEEK" format encoded as a float

The days of the week (e.g. parameter {1205}) is coded as a bit field in a float. To select a day, set its corresponding bit to 1.

| Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|----------|--------|----------|-----------|---------|--------|
| Sunday | Saturday | Friday | Thursday | Wednesday | Tuesday | Monday |

So:

Monday will be encoded as $2.0^0 = 1.0 = 0x3f800000$

Sunday will be encoded as $2.0^6 = 64.0 = 0x42800000$

Thursday + Saturday will be encoded as $2.0^3 + 2.0^5 = 40.0 = 0x42200000$

All days will be encoded as $2.0^6 + 2.0^5 + 2.0^4 + 2.0^3 + 2.0^2 + 2.0^1 + 2.0^0 = 127.0 = 0x42fe0000$

5.6 MESSAGE NOTIFICATIONS

Message notifications are frames sent by one of the Studer devices and transmitted to the DTE via the Xcom-CAN module.

5.7 MESSAGE NOTIFICATION SERVICE

The Xcom-CAN notify a new message with a frame in the following format:

| CAN Identifier | | | | CAN Data | |
|----------------|----------------|---------|--------|------------------|--------------------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | MESSAGE ID | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 500 or 0 | 0x000 to 0x3FF | 0x3 | 0x00 | 0x0000 to 0xFFFF | 0x00000000 to 0xFFFFFFFF |

DST ADDR : 500 (RCC group address) or 0 (broadcast)
This field can be ignored in message notification

SRC ADDR : 0x00 to 0x3FF

SERVICE : 0x3 (message notification service)

FLAGS : 0x00

MESSAGE ID : 0x0000 to 0xFFFF (see "Xtender serial protocol appendix")

VALUE : 0x00000000 to 0xFFFFFFFF

5.7.1 Example

A "Battery low" notification from the first XT.

| CAN Identifier | | | | CAN Data | |
|----------------|----------|---------|--------|------------|------------|
| DST ADDR | SRC ADDR | SERVICE | FLAGS | MESSAGE ID | VALUE |
| 10 bits | 10 bits | 3 bits | 6 bits | 16 bits | 32 bits |
| 500 | 101 | 3 | 0 | 0 | 0 |
| 0x1F4 | 0x65 | 0x3 | 0x00 | 0x0000 | 0x00000000 |

6 ERROR CODES

The following error codes apply for "User Info read service" and "Parameter read/write service". They are coded in the response in case of a request failure.

| Name | Error code 32 bits | Meaning |
|------------------------------|-----------------------|--|
| INVALID_FRAME | 0x00000001 | Malformed frame |
| DEVICE_NOT_FOUND | 0x00000002 | Wrong DST_ADDR field |
| RESPONSE_TIMEOUT | 0x00000003 | No response from the server |
| INVALID_SERVICE_ARGUMENT | 0x00000012 | Wrong service data |
| GATEWAY_BUSY | 0x00000013 | Gateway busy |
| OBJECT_ID_NOT_FOUND | 0x00000022 | No object with this info or param id was found |
| INVALID_DATA_LENGTH | 0x00000024 | The data field has an invalid number of bytes |
| PROPERTY_IS_READ_ONLY | 0x00000025 | A writing to this property is not allowed |
| INVALID_DATA | 0x00000026 | This value is impossible for this property |
| DATA_TOO_SMALL | 0x00000027 | The value is below the minimum limit |
| DATA_TOO_BIG | 0x00000028 | The value is above the maximum limit |
| WRITE_PROPERTY_FAILED | 0x00000029 | Writing is possible, but failed |
| READ_PROPERTY_FAILED | 0x0000002A | Reading is possible, but failed |
| ACCESS_DENIED | 0x0000002B | Insufficient user access |
| MULTICAST_READ_NOT_SUPPORTED | 0x0000002D | Read operation is not supported when used on multicast addresses |