XLN-t bvba
Hoogstraat 52
B 2580 Putte-Beerzel
Belgie - Belgium

tel +32 (0) 15 24 92 43
fax +32 (0) 15 25 10 58
RPR Mechelen
BTW BE 423 212 087
Bank 733-2011497-38

BDX118 T1107

Manual

V00.02.001

11 March 2011

Introduction

This document contains the specifications for the BDX118 T1107.

Tip: There are complete examples at the end of the document. It is useful to look at them when the explanation is unclear.

Functionality

When a valid eID is presented to the eID reader, the eID reader will transmit a sequence of fields through the RS232 connection. The sequence is configurable.

There is a special protocol that is used to configure the eID reader.

RS232 protocol specifiations

Baud rate:	9600 baud
Data bits:	8 bits
Parity bits:	no parity
Stop bits:	1 bit
Handshake:	None

Power supply

Nominal voltage:	5 – 12Vdc
Minimal voltage:	4.95Vdc
Maximum voltage:	14Vdc
Polarity protection:	**NOT PRESENT !!**
Current consumption:	50mA

Connections

The connections are on the RJ45 connector.
The top side of the PCB has a indication for pin 8 of the connector.
The bottom side of the PCB has a indication for pin 1 and also the 0V=GND and +12V=Power pin.

Number	Name	Description
1	RxD	Serial input ±12V
2	TxD	Serial output ±8V RS232 formaat
3		Do not connect !
4		Do not connect!
5	CTS	Input ±12V No function
6	RTS	Output ±8V Active(+8V) when valide ID card is present
7	Power	5-12Vdc connection, +
8	0V	Ground connection

Field order

The eID card contains two files, each file has multiple fields. The eID reader can transmit the fields from the eID card in a configurable order.

The configuration contains 19 locations. Each location can be set to a value between 0 and 255. When the location is between 1 and 19, it indicates a specific field from the eID card. The value is then called a Field ID, see next table. A value of 0 or 255 is used to stop the sequence. The values 20..254 are reserved for further use. They will currently be skipped, so nothing is transmitted, but the delay will be done, see Field format.

Field ID (decimal)	Data	Encoding Type	Max nb. of byte (decimal)	Max nb. of UTF-8 characters (decimal)
1	Card number	ASCII	12	
2	Chip number	Binary[1)]	16	
3	Card validity date begin ***DD.MM.YYYY***	ASCII	10	
4	Card validate date end ***DD.MM.YYYY***	ASCII	10	
5	Card delivery municipality	UTF-8	80	42
6	National Number	ASCII	11	
7	Name	UTF-8	110	62
8	2 first given names	UTF-8	95	52
9	First letter of 3rd given name	UTF-8	3	1
10	Nationality	UTF-8	85	50
11	Birth location	UTF-8	80	40
12	Birth date ***DD mmmm YYYY*** or ***DD.mmm.YYYY***(german) See table below	UTF-8	12	12
13	Sex ***M***: man ***F/V/W***: woman	ASCII	1	
14	Noble condition	UTF-8	50	21
15	Document type ***1***: Belgian citizen ***2***: European Community Citizen ***3***: non European Com. Citizen	ASCII	2	
16	Special status ***0***: No status ***1***: White cane (blind people) ***2***: Extended minority ***3***: White cane + extended min. ***4***: Yellow cane (partially sighted people) ***5***: Yellow cane + extended min.	ASCII	2	
17	Street + number	UTF-8	80	60
18	ZIP code	ASCII	6	4
19	Municipality	UTF-8	67	47

1) The chip number is converted to a hexadecimal number in ASCII format.

Example of configuration:

We want to transmit the fields 7, 8, 9, 11, 6.

Location	Description	Value
1	First field to transmit	7
2	Second field to transmit	8
3	3rd field to transmit	9
4	4th field to transmit	12
5	5th field to transmit	6
6	6th field to transmit	0
7	7th field to transmit	255
8	8th field to transmit	255
9	9th field to transmit	255
10	10th field to transmit	255
11	11th field to transmit	255
12	12th field to transmit	255
13	13th field to transmit	255
14	14th field to transmit	255
15	15th field to transmit	255
16	16th field to transmit	255
17	17th field to transmit	255
18	18th field to transmit	255
19	19th field to transmit	255

The reader will transmit the following fields in the given order:

7 Name
8 2 First given names
9 First letter of 3rd given name
11 Birth location
6 National number

Field format

Each field is transmitted in the following format:

<Prefix><Field ID><Sep1><Len><Sep2><Data><Postfix><Crc><Terminator>{Delay}

So each field contains 10 ***parts***.

<Prefix>

This is a optional and configurable part. The configuration contains 9 locations (bytes) for the configuration. The first byte is a length that indicates the number of characters of the prefix that must be transmitted. The next 8 locations are the characters. When the length is 0 or 255, no characters will be transmitted. When the length is >= 8 and <= 254, all 8 characters will be transmitted.

Example of configuration:

Location	Description	Value
20	Prefix length	5
21	Prefix character 1	0x01
22	Prefix character 2	0x45
23	Prefix character 3	0x00
24	Prefix character 4	0x00
25	Prefix character 5	0xFF
26	Prefix character 6	255
27	Prefix character 7	255
28	Prefix character 8	255

The reader will transmit: 0x01 0x45 0x00 0x00 0xFF

<Field ID>

This is a optional part. The configuration contains one location to configure this part. When the location has the value 1, the part is transmitted.

The part will be transmitted in ASCII format as 3 decimal numbers with leading zero's. The part will contain the Field ID

Example of configuration:

Location	Description	Value
29	Field ID On/Off	1

The reader will transmit the Field ID: “007”
The value 007 indicates the Name.

<Sep1>

This is a optional and configurable field. The configuration contains one location to configure the field. When the location value is 0 or 255, the separator is not transmitted. When the value is >= 1 and <= 254, the given value is transmitted.

Example of configuration:

Location	Description	Value
30	Sep 1	44

The reader will transmit a comma: “,”

<Len>

This is a optional part. The configuration contains one location to configure this part. When the location has the value 1, the part is added, otherwise it is not added.

The part will be transmitted in ASCII format as 3 decimal numbers with leading zero. The part will contain the length of the data. This is the number of bytes.

Example of configuration:

Location	Description	Value
31	Len On/Off	1

The reader will transmit the Len: “005”
The value 005 indicates the length of the data.

<Sep2>

This is a optional and configurable field. The configuration contains one location to configure the field. When the location value is 0 or 255, the separator is not transmitted. When the value is >= 1 and <= 254, the given value is transmitted.

Example of configuration:

Location	Description	Value
32	Sep 2	44

The reader will transmit a comma ","

<Data>

This part is always present. It contains the data of the field in the format read from the eID card, except for the chip number. In case of the chip number, the value is converted to a ASCII hexadecimal number. Each byte of the card number is translated to two characters.

There is no translation for UTF-8 fields.

<Postfix>

This is a optional and configurable part. The configuration contains 9 locations for the configuration. The first byte is a length that indicates the number of characters in the postfix that must be transmitted. The next 8 locations are the characters. When the length is 0 or 255, no characters will be transmitted. When the length is >= 8 and <= 254, all 8 characters will be transmitted.

Example on configuration:

Location	Description	Value
33	Postfix length	2
34	Postfix character 1	0x0A
35	Postfix character 2	0x0D
36	Postfix character 3	255
37	Postfix character 4	255
38	Postfix character 5	255
39	Postfix character 6	255
40	Postfix character 7	255
41	Postfix character 8	255

The reader will transmit: 0x0A 0x0D

{Delay}

This is a configurable part. The configuration contains one location for the configuration. The delay is equal to 10milliseconds multiplied with the value. This means that when the value is 0, there is no delay. When the value is 100, there is a delay of 100ms.

Very important

The delay is added to the time needed to process the eID card data. This means that the configured delay is a minimum delay. When the delay is measured it will certainly be greater.

Example of configuration:

Location	Description	Value
42	Delay, 10ms resolution	20

The reader will wait 200 milliseconds after sending the field and before sending the next field.

<Crc>

This is a optional and configurable part. The configuration contains one location to configure the field. When the location value is 1 or 2, the crc is transmitted. In any other case no crc is transmitted.

Crc16

When the value is 1, a 16 bit CRC is transmitted that includes all characters that are send for the field up to the start of this field. The CRC uses polynomial:

X^16 + X^12 + X^5 + X^1

This is the CRC-CCITT/XModem. The initial value is 0. Example code is provided at the end of the document.

Sum16

When the value is 2, a 16 bit checksum is transmitted that is the sum of all characters that are send for the field up to the start of this field. The initial value is 0.

Transmitted as hex

Both the Crc16 and the Sum16 are transmitted as four hexadecimal digits. The first transmitted digit is the most significant nibble.

Example of configuration:

Location	Description	Value
43	Crc On/Off	0

The reader will not transmit a CRC.

Example of CRC's

Suppose that field 7 is transmitted without any other part and the value of the field is **Meert**.

Crc16 `Meert`**`F536`**
Sum16 `Meert`**`01FD`**

<Terminator>

This is a optional and configurable part. The configuration contains 9 locations for the configuration. The first byte is a length that indicates the number of characters in the terminator that must be transmitted. The next 8 locations are the characters. When the length is 0 or 255, no characters will be transmitted. When the length is >= 8 and <= 254, all 8 characters will be transmitted.

Example on configuration:

Location	Description	Value
44	Terminator length	0
45	Terminator character 1	255
46	Terminator character 2	255
47	Terminator character 3	255
48	Terminator character 4	255
49	Terminator character 5	255
50	Terminator character 6	255
51	Terminator character 7	255
52	Terminator character 8	255

The reader will not transmit a terminator.

Configuration

The configuration contains 39 locations. The value of each location can be changed using a very simple RS232 protocol.

To get the value of a location, given the location number followed by a carriage return. The eID reader will respond with the location number, followed by a colon, the value, carriage return and linefeed.

1<CR>
1:5<CR><LF>

To set the value of a location, give the location number followed by a colon, the new value and a carriage return. The eID reader will respond in the same format as with the get.

1:6<CR>
1:6<CR><LF>

The protocol is enabled as long as no card is presented to the reader. From the moment a card is presented, the protocol is disabled.

The protocol is also protected. Before the values of other locations can be updated, the protection must be disabled. This is done by writing the value 1 to location 123.

123:1<CR>
123:1<CR><LF>

The protection can be re-enabled by setting the value of location 123 to any value other then 1.

123:0<CR>
123:0<CR><LF>

The eID reader will only transmit a answer when the location number is valid and a carriage return is received.

It is possible that the answer does not contain the given value because the value has a bad format.

Example

The following sequence is used to setup the eID reader in for the configuration given in the previous examples. The Carriage return and Line feeds are not shown:

123:1 **Disable protection**
123:1

1:7	Location 1 = 7
1:7	Transmit Name field
2:8	Location 2 = 8
2:8	Transmit First Name field
3:9	Location 3 = 9
3:9	Transmit first letter third name
4:11	Location 4 = 12
4:11	Transmit birth date
5:6	Location 5 = 6
5:6	Transmit national number
6:0	Location 6 = 0
6:0	Stop transmitting fields
20:5	Location 20 = 5
20:5	Prefix length is 5
21:1	Location 21 = 1
21:1	Prefix character 1: 1=0x01
22:69	Location 22 = 69
22:69	Prefix character 2: 69=0x45
23:0	Location 23 = 0
23:0	Prefix character 3: 0=0x00
24:0	Location 24 = 0
24:0	Prefix character 4: 0=0x00
25:255	Location 25 = 255
25:255	Prefix character 5: 255=0xFF
29:1	Location 29 = 1
29:1	Transmit TagID

30:44	Location 30 = 44
30:44	Transmit separator 1: 44=’,’
31:1	Location 31 = 1
31:1	Transmit field length
32:44	Location 32 = 44
32:44	Transmit separator 3: 44=’,’
33:2	Location 22 = 2
20:2	Postfix length is 2
34:10	Location 21 = 10
34:10	Postfix character 1: 10=0x0A
35:13	Location 22 = 13
35:13	Postfix character 2: 13=0x0D
42:20	Location 42 = 20
42:20	Delay is 20 = 200ms
123:0	**Enable protection**
123:0	

Output example

When the configuration is done as in the given examples and a eID card is presented, the following output will be generated (the field data contains arbitrary values).

The notation 0xYY is used to indicate a single byte in non ASCII format. This is also surrounded with quotes: “, the quotes are <u>not</u> part of the transmission.

```
“0x01 0x45 0x00 0x00 0xFF”007,005,MEERT”0x0A 0x0D”
“0x01 0x45 0x00 0x00 0xFF”008,008,CHRISTEL”0x0A 0x0D”
```

```
“0x01 0x45 0x00 0x00 0xFF”009,000,”0x0A 0x0D”
“0x01 0x45 0x00 0x00 0xFF”011,012,29.MAAR.1975”0x0A 0x0D”
“0x01 0x45 0x00 0x00 0xFF”006,011,75032900123”0x0A 0x0D”
```

After each field there is a minimal delay of 200 milliseconds.

Error format

It is possible that the BDX118 T1107 detects a error while processing the card. In that case it can be useful that the reader transmits a error message. The reader supports this feature with three different formats. The enabling of this feature and selecting the format is done on location 53.

Location	Description	Value
53	Error On/Off	1

Value	
1	Field format
2	Special field format
3	Short format
Any other value	No error

1=Field format

This uses the same format as a normal field, but the Field ID will be equal to 254 and the Data will contain the word “Error”.

The same configuration as for normal fields is applied.

After transmitting the field no other fields will be transmitted, until a new eID card is presented.

Example when configuration is done as in the given examples:

```
“0x01 0x45 0x00 0x00 0xFF”254,005,ERROR”0x0A 0x0D”
```

2=Special Field format

When the value on location 53 is 2, the reader will transmit following parts in case of a error:

<Errorprefix><Field ID><Sep1><Len><Sep2><Errorcode><Postfix><Crc><Terminator>{Delay}

The <Field ID> will be 254 and the <Len> will be 006.

The difference with option 1=Field format, is the special <Errorprefix> and the <Errorcode> in stead of the word ERROR.

3=Short format

When the value on location 53 is 1, the reader will transmit following parts in case of a error:

<Errorprefix><Errorcode><Postfix><Crc><Terminator>{Delay}

<Errorprefix>

This is a optional and configurable part. The configuration contains 9 locations (bytes) for the configuration. The first byte is a length that indicates the number of characters of the prefix that must be transmitted. The next 8 locations are the characters. When the length is 0 or 255, no characters will be transmitted. When the length is >= 8 and <= 254, all 8 characters will be transmitted.

Example of configuration:

Location	Description	Value
54	Error prefix length	6
55	Error prefix character 1	‘E’
56	Error prefix character 2	‘R’
57	Error prefix character 3	‘R’
58	Error prefix character 4	‘O’
59	Error prefix character 5	‘R’
60	Error prefix character 6	‘:’
61	Error prefix character 7	255
62	Error prefix character 8	255

The reader will transmit: ERROR:

<Errorcode>

This is value that contains six hexadecimal digits. The value is very difficult to interpret because it is some internally generated code.

Other parts

The other part are the same as for a normal field.

Valid eID card

The eID Dräger doesn’t check of the eID card is valid. This would require a network connection with a validation server. The eID Dräger only reads the card data, that must be in a valid format, with any interpretation or writing to non-volatile memory.

The user of the system must check of the eID card is valid.

Configuration overview

Location	Description	Default value
1	First field to transmit	7
2	Second field to transmit	8
3	3rd field to transmit	12
4	4th field to transmit	17
5	5th field to transmit	18
6	6th field to transmit	19
7	7th field to transmit	0
8	8th field to transmit	255
9	9th field to transmit	255
10	10th field to transmit	255
11	11th field to transmit	255
12	12th field to transmit	255
13	13th field to transmit	255
14	14th field to transmit	255
15	15th field to transmit	255
16	16th field to transmit	255
17	17th field to transmit	255
18	18th field to transmit	255
19	19th field to transmit	255
20	Prefix length	0
21	Prefix character 1	255
22	Prefix character 2	255
23	Prefix character 3	255
24	Prefix character 4	255
25	Prefix character 5	255
26	Prefix character 6	255
27	Prefix character 7	255
28	Prefix character 8	255
29	Field ID On/Off	0
30	Sep 1	0
31	Len On/Off	0
32	Sep 2	0

33	Postfix length	2
34	Postfix character 1	0x0A
35	Postfix character 2	0x0D
36	Postfix character 3	255
37	Postfix character 4	255
38	Postfix character 5	255
39	Postfix character 6	255
40	Postfix character 7	255
41	Postfix character 8	255
42	Delay, 10ms resolution	0
43	Crc On/Off	0
44	Terminator length	0
45	Terminator character 1	255
46	Terminator character 2	255
47	Terminator character 3	255
48	Terminator character 4	255
49	Terminator character 5	255
50	Terminator character 6	255
51	Terminator character 7	255
52	Terminator character 8	255
53	Error On/Off	1
54	Error prefix length	0
55	Error prefix character 1	255
56	Error prefix character 2	255
57	Error prefix character 3	255
58	Error prefix character 4	255
59	Error prefix character 5	255
60	Error prefix character 6	255
61	Error prefix character 7	255
62	Error prefix character 8	255

When the default configuration is used and a eID card is presented, the following output will be generated (the field data contains arbitrary values).

The notation 0xYY is used to indicate a single byte in non ASCII format. This is also surrounded with quotes: “, the quotes are not part of the transmission.

```
MEERT”0x0A 0x0D”
CHRISTEL”0x0A 0x0D”
29.MAAR.1975”0x0A 0x0D”
TRAMMELIE 3456”0x0A 0x0D”
2580”0x0A 0x0D”
Putte”0x0A 0x0D”
```

In case of a error, the following is transmitted:

```
ERROR”0x0A 0x0D”
```

Leds

The reader has two leds, a green and a red.

Power up

During power up both leds will be on. During this time the boot loader is running.

Waiting for card

When the reader is waiting for a card, the green led will flash. The red led will be off.

Card present

When a card is presented, the green led will be off. The red led will be on when power is supplied to the card.

RTS

The card reader has a RTS output signal with RS232 levels.

This signal will be active (+12V) when a valid eID card has been presented. It stays active until the card is removed.

Boot loader

The card reader has a boot loader that can be used to update the firmware. There is a separated document that describes this feature.

CRC16

De next code shows how the CRC16 value can be calculated in C.

Crc16.h

```
#ifndef CRC16_H
#define CRC16_H

#ifdef __cplusplus
extern "C"
{
#endif

extern unsigned short CRC16_Update(unsigned short currentcrc,
                                   unsigned char newbyte);

#ifdef __cplusplus
}
#endif

#define CRC16_STARTVALUE(crc)       crc = 0
#define CRC16_UPDATE(crc, newbyte)  crc = CRC16_Update(crc, newbyte)
```

```
#endif /* CRC16_H */
```

Crc16.c

```
#include "CRC16.h"

/* Define to use the table
 * When undefined, the formula is used.
 */
#define CRC16_USING_TABLE

#ifdef CRC16_USING_TABLE

/*
 * Calculate CRC using a table
 *  - Fast
 *  - Uses 512 bytes of memory for the table
 *
 * Use this on a slow processor
 */
const unsigned short CRC16_Table[256] =
{
  0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
  0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
  0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
  0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
  0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
  0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
  0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
  0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
  0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
  0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
  0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
  0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
  0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
  0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
  0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
```

```
  0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
  0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
  0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
  0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
  0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
  0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
  0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
  0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
  0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
  0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
  0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
  0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
  0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
  0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
  0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
  0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
  0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};


unsigned short CRC16_Update(unsigned short currentcrc, unsigned char newbyte)
{
  return (currentcrc << 8) ^ CRC16_Table[ (currentcrc >> 8) ^ newbyte ];
}

#else

/*
 * Calculate CRC using the formula
 *  - Slower
 *  - Uses no memory for table
 *
 * Use this on a fast processor that can handle 16-bit words
 */
unsigned short CRC16_Update(unsigned short currentcrc, unsigned char newbyte)
{
  unsigned char  j;
  unsigned short msg;
```

```
  unsigned short crc;

  crc = currentcrc;
  msg = (newbyte << 8);

  for(j = 0 ; j < 8 ; j++)
  {
        if ((msg ^ crc) >> 15)
              crc = (crc << 1) ^ 0x1021;
        else
              crc <<= 1;

        msg <<= 1;
  }

  return crc;
}

#endif
```