AN002 BIM M13x_Serial_Com

Published by:

Opternus Components GmbH
Bahnhofstr. 5
D-22941 Bargteheide
Germany

Serial communication with BIM M13x

BIM_M_13x_SerialCom is a free to use and modifiable for own requirements.

It contains the source code for a BIM M 13x application and the C#.NET source code that could be compiled with MS Visual Studio for a windows applications or a device that supports the .NET compact framwork and also with the Linux ported .NET project MONO.

There are four parts in BIM_M_13x_SerialCom:

- LowLevel

- ComObject

- GrpAddress

(- Parameter)

The "LowLevel" part contains a function "SLLSend" on BIM M 13x side and "SendMsg" on .NET side. This function takes an array of bytes and adds to the array a sequence counter, a command byte, a checksum and the length of all data and transmit the data via serial communication. The "LowLevel" part also contains functions that receive data via a serial communication and check the checksum and sequence counter. This functions send ACK or NACK whether the checksum is correct or incorrect or a reset if the sequence counters does not match. If (and only if) a valid message was received the "LowLevel" part will call a function ("SLL_OnDataReceived") and pass the payload data to it. This function has to be implemented and must handle the received data.

The "ComObject" part uses the „LowLevel“ functions and provides the following features:

- sending an indication (from BIM M 13x) that a communication object was updated via bus (message contains the new data) -> (CommObj-)Indication

- sending new data for a communication object (from .NET) that should be transmitted -> (CommObj-)Write request

- sending (from .NET) a request that a group value read request should be send on the bus for a specific communication object -> (CommObj-)Read request

- sending a confirm (from BIM M 13x) for the write and read requests -> (CommObj-)Write confirm and (CommObj-)Read confirm

- sending a request (from .NET) that the actual data of a communication object should be send back -> (CommObj-)Value read request

- sending a response (from BIM M 13x) for the "(CommObj-)Value read request" -> (CommObj-)Value read response

- sending a request (from .NET) to change the ram flags for a communication object -> Set(-CommObj-)RamFlags

- sending a response (from BIM M 13x) with the actual value of the ram flags for a specific communication object -> Set(-CommObj-)RamFlags response

- sending a request (from .NET) to get the type if a specific communication object -> Get(-CommObj-)Type

- sending a response (from BIM M 13x) with the type of a specific communication object (usefull if ETS could change the type of a communication object) -> Get(-CommObj-)Type response.

The exact message structure could be seen in BIM_M_13x_SerialCom.xls that is part of this package.

For a first test load from the 'bin' folder of this package the 'BIM_M_13x_SerialCom_Obj_mod.s19' file into a BIM M 13x or the EVB with the

BIM Tools S19-Downloader and run this application. Start from the 'bin' folder the windows application 'BIM_M_13x_SerialCom.exe'.

The windows application will work with the features of the "ComObject" part. Send group value write telegrams with the BIM Tools on addresses

0x1100 (UINT1)

0x1101 (UINT6)

0x1102 (UINT8)

0x1103 (UINT16)

0x1104 (TIME_DATE)

0x1105 (DATA6)

0x1106 (DATA10)

0x1107 (MAXDATA)

and you should see '(CommObj-)Indication' messages with 'BIM_M_13x_SerialCom.exe'.

The source code for BIM M 13x application is in folder 'BIM_M_13x_Serial_BIM\BIM_M_13x_SerialCom_Obj' of this package and the source code for the windows application is in folder 'BIM_M_13x_Serial_PC'.

To use the "ComObject" and "LowLevel" parts in a BIM M 13x application just follow these steps:

- create a new project with your communication objects with BIM-Tools Wizard
- open project with IAR EmbeddedWorkbench for NEC78K0
- add the following files to your project

application.c

application.h

BIM_M_13x_Serial_ComObject.c

BIM_M_13x_Serial_ComObject.h

BIM_M_13x_Serial_LowLevel.c

BIM_M_13x_Serial_LowLevel.h

utility.c

utility.h

- set NUM_OF_TIMERS in user.h to 2
- add '#include "application.h"' in main.c
- delete the complete function 'AppInit(void)' in main.c
- add 'Application_Task();' within the 'for(;;)...' loop in function 'main(void)' in main.c
- change in 'AppInfoBlock' the 'pUserTimerTab' entry from 'NULL' to '&UserTimerTab'

To use the "ComObject" and "LowLevel" parts in .NET application just add the files

BIM_M_13x_Serial_ComObj.cs

BIM_M_13x_SerialCom_LowLevel.cs

to your own project.

Create a new instance of the class 'BIM_M_13x_Serial_ComObj', call the 'Connect(...)' member and register to the events you need, for example:

```
private BIM_M_13x_Serial_ComObj m_bsc;
.
.
.
...
{
m_bsc = new BIM_M_13x_Serial_ComObj();
m_bsc.OnComObjIndication += new BIM_M_13x_Serial_ComObj.OnComObjIndicationEventHandler(m_bsc_OnComObjIndication);
}
```

```
.

.

.

private void m_bsc_OnComObjIndication(int ObjNum, byte[] data)

{

    // do something with 'ObjNum' and 'data'

}
```

or call one of the features, for example:

```
m_bsc.SendComObjWriteRequest(ObjNum, bytearray);
```

The "GrpAddress" part uses the "LowLevel" functions and provide the following features:

- sending data with a group address (from .NET) that should be transmitted -> (GrpAddr-)Write request
- sending a confirm (from BIM M 13x) that the (GrpAddr-)Write request was sent on bus -> (GrpAddr-)Write confirm
- sending an indication (from BIM M 13x) that a group addressed (write) telegram was on the bus -> (Grpaddr-)Write indication

- sending a read request with a group address (from .NET) that should be transmitted -> (GrpAddr-)Read request
- sending a confirm (from BIM M 13x) that the (GrpAddr-)Read request was sent on bus -> (GrpAddr-)Read confirm
- sending an indication (from BIM M 13x) that a group addressed (read) telegram was on the bus -> (Grpaddr-)Read Indication

- sending a response request with a group address (from .NET) that should be transmitted -> (GrpAddr-)Response request
- sending a confirm (from BIM M 13x) that the (GrpAddr-)Response request was sent on bus -> (GrpAddr-)Response confirm
- sending an indication (from BIM M 13x) that a group addressed (response) telegram was on the bus -> (Grpaddr-)Response Indication

The exact message structure could be seen in BIM_M_13x_SerialCom.xls that is part of this package.

For a first test load from the 'bin' folder of this package the 'BIM_M_13x_SerialCom_Grp_mod.s19' file into a BIM M 13x or the EVB with the BIM Tools S19-Downloader and run this application. Start from the 'bin' folder the windows application 'BIM_M_13x_SerialCom.exe'.

The windows application will work with the features of the "GrpAddress" part on tab 'Group addresses'. Send group value write telegrams with the BIM Tools and you should see '(GrpAddr-)Write Indication' messages with 'BIM_M_13x_SerialCom.exe'.
Enter in 'BIM_M_13x_SerialCom.exe' a group address (hexadecimal), enter payload bytes and press send. You should see the group value write telegrams with BIM Tools.

The source code for BIM M 13x application is in folder 'BIM_M_13x_Serial_BIM\BIM_M_13x_SerialCom_Grp' of this package and the source code for the windows application is in folder 'BIM_M_13x_Serial_PC'.

The 'GroupAddress' BIM M 13x application will only send group addressed telegrams (without broadcast telegtrams) via serial communication to the external device.

To use the "GrpAddress" and "LowLevel" parts in .NET application just add the files

    BIM_M_13x_Serial_GrpAddr.cs

    BIM_M_13x_SerialCom_LowLevel.cs

to your own project.

Create a new instance of the class 'BIM_M_13x_Serial_GrpAddr', call the 'Connect(...)' member and register to the events you need, for example:

```
private BIM_M_13x_Serial_GrpAddr m_bsg;

.

.

.

...

{
```

```
m_bsg = new BIM_M_13x_Serial_GrpAddr();

m_bsg.OnGrpAddrWriteIndication += new
BIM_M_13x_Serial_GrpAddr.OnGrpAddrWriteIndicationEventHandler(m_bsg_OnGrpAddrWriteIndication);

}

.

.

.

private void m_bsg_OnGrpAddrWriteIndication(ushort GrpAddr, bool Less7Bit, byte[] data)

{

    // do something with 'GroupAddr' and 'data' (and check Less7Bit)

}
```

or call one of the features, for example:

```
m_bsg.SendGrpAddrWriteRequest(GrpAddr, true, bytearray);
```

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE BIM M13x / CHIPSET COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF CERTAIN FUNCTIONALITY. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. OPTERNUS COMPONENTS GMBH HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.