AP08055

XC88x

XC886/888 CM/CLM

CAN BSL Example for XC88x

Microcontrollers

Never stop thinking

Edition 2006-12

Published by
Infineon Technologies AG
81726 München, Germany

© Infineon Technologies AG 2006.
All Rights Reserved.

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES 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.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

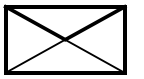
XC88x

Revision History: V1.1, 2006-12

Previous Version(s):
none

Page	Subjects (major changes since last revision)
V1.0	Initial Release
V1.1	Additional Option for Flash Un-Protection (Chapter 5.1)

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com



1 Introduction

A built in bootstrap loader is implemented in the XC88x to provide a mechanism to load code to XRAM via the CAN interface. The CAN bootstrap loader is only available in the variant with a marking “C” (“XC88xCM”, and “XC88xC”).

1.1 Overview

This document provides an example that could be used to program the internal Flash via the CAN interface.

Hex file used:	can_loader.hex
	u2can.hex (to be downloaded to the CAN Interface Board)
Tools:	Keil C51 evaluation version
	XC800 Flash Loader (version 4.0 or higher)
Code generator:	DAvE
Hardware:	XC88x starterkits (2 pcs).

1.2 Hardware Connection

Overview of the hardware connection for this example.

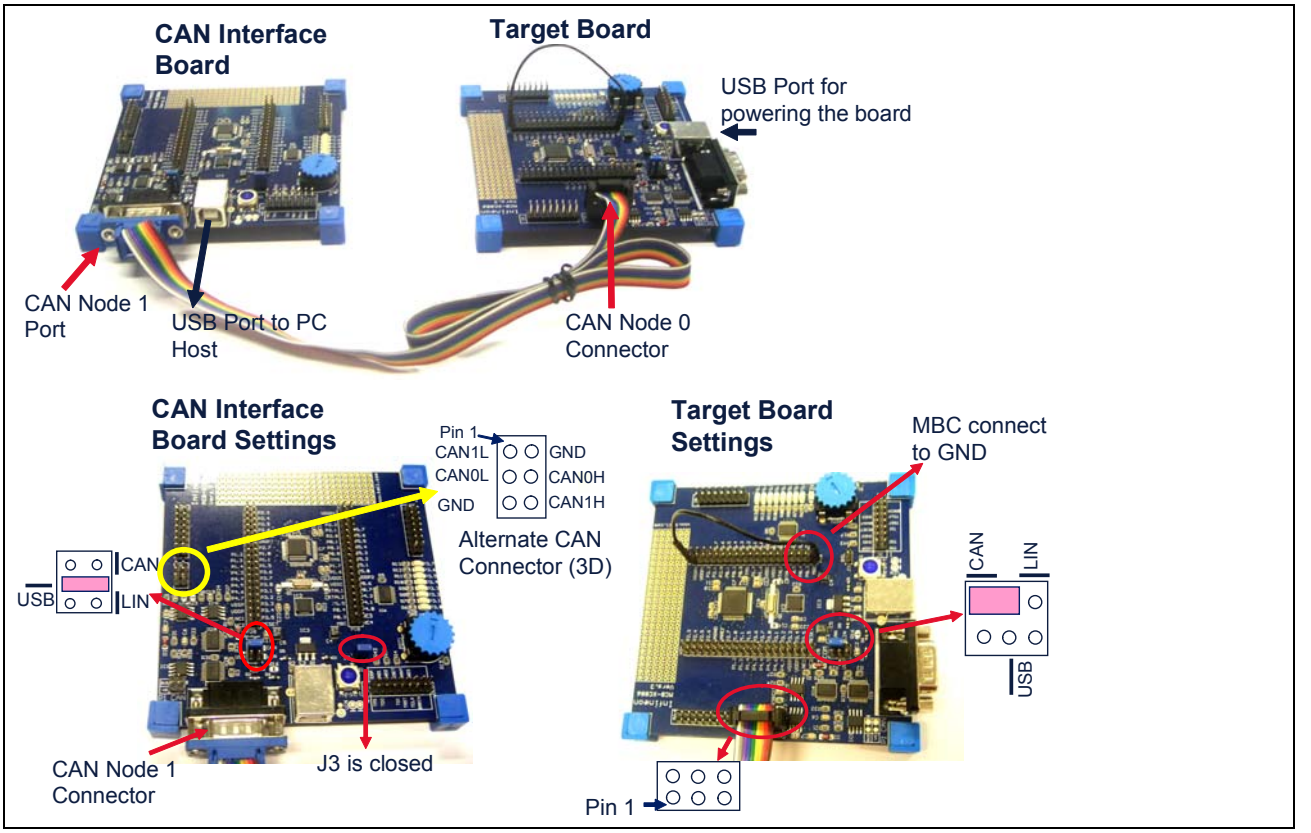
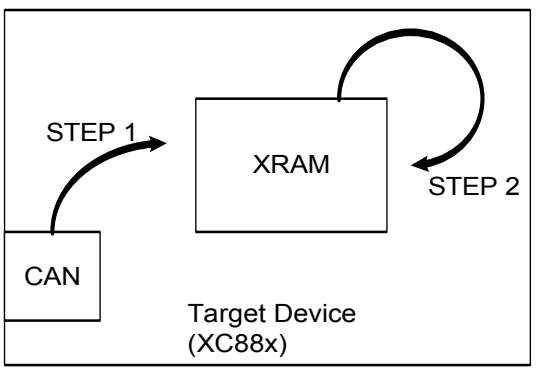


Figure 1 Hardware connection to run the example.

1.3 Software Overview

The figure below will give an overview of the steps needed to program the target device using the CAN BSL.

The detail of each step and the detailed explanation about can_loader.hex will be given in the next chapters.



STEP 1: Downloading can_loader.hex to XRAM

STEP 2: Running the can_loader.hex from XRAM

Figure 2 Software process overview

1.4 Running the Example

This Application Note should use the tools and files mentioned in the overview chapter (**Chapter 1.1**)

1. Connect the hardware as shown in **Figure 1**.
2. Make sure that all of the jumpers are correctly connected as shown in the **Figure 1**.
3. Reset the Target Board by pressing the Reset Button.
4. Open XC800_Fload (version 4.0 or higher).
5. Click the button “Find Device” until you see the selection of devices.

   There should be at least one XC88x device selected (this will represent the CAN Interface Board)
6. Go to the “Option” menu to choose “Use CAN Board” option.

   Please note that the Baudrate is automatically changed to 115200 (“u2can.hex” is hardcoded with this baudrate, so you can not change it). If you would like to repeat

STEP 5, please make sure to disable the “Use CAN Board” option and change the baudrate back to 57600.

7. Click the “Connect” button to see if the XC800 FLOAD can connect to the Target Board.

   In this step, the “u2can.hex” is automatically loaded to the CAN Interface board. The hex code is hardcoded inside the loader, so the process is transparent to the user.

8. Now you can start to download the hex file to the target board.

   As an example, please try to download the “Hello_World.hex” given in the starterkit or together with this example.

   You can download by clicking “Open File” and choose “Hello_Word.hex”.

   Click the “Download” button to download this file into the Target board.

9. After a successful download, you can either click “Execute Flash” or Reset the Target Board (after removing the MBC jumper).

10. You can also experiment with the Flash Protection.

    Please make sure that you configure the parameter in the file “XC88x_FLADDR.H” correctly (see **Chapter 4.1** and **Chapter 5.1**for important information)

If there is an error during the autobaudrate step (in Step 7), please Reset the Target Board by pressing the Reset Button.

2 CAN Interface Board

In this example, XC88x starterkit board (with external crystal) is used as the interface board.

The board is programmed with the U2CAN (Uart to CAN) software.

This code has it's own protocol on how to communicate with the PC Host and the another CAN Node.

U2CAN is responsible to take the command from the PC Host (through UART) and send it to the other CAN Device through it's CAN Node 1.

U2CAN is also responsible to take the CAN Frame from the other CAN Device (through it's CAN Node 1) and pass the CAN Frame information (ID, Data, Counter, etc) to the PC Host (through UART).

Other feature of U2CAN software (e.g: CAN sniffer through Node 0, Configuration of CAN Node, etc) is not used in this example.

The detail protocol and feature about U2CAN software will not be discussed here, as there is already ApHints for the Uart to CAN software.

This document will concentrate on the can_loader.hex, as this hex file can be loaded by any tools (e.g: CAN Vector interface board, etc), independently from the interface board.

3 STEP1: Downloading can_loader.hex to XRAM

To start with STEP1, the target device will need to be reset with the following reset configuration:

MBC= ‘0’ and TMS = ‘0’

After reset, the Boot ROM firmware will run with the protocol shown in the figure below to download the can_loader.hex to XRAM. Once the download is completed, the Boot ROM will automatically jump to XRAM and run the can_loader.hex from there.

No acknowledgement is given to the PC Host after the completion of the download.

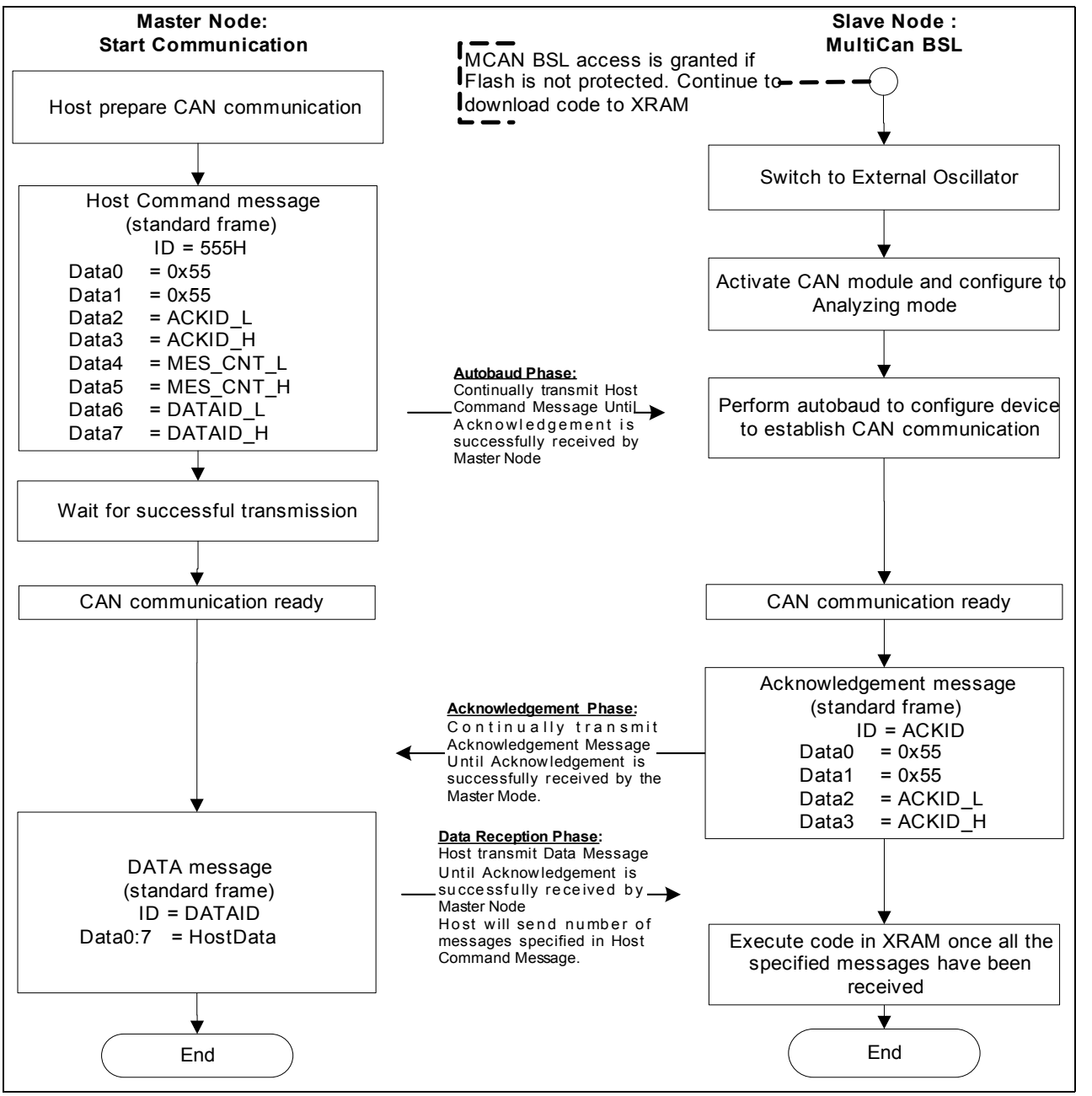


Figure 3 CAN BSL Protocol

4 STEP2: Running can_loader.hex program

4.1 Re-configuration

can_loader.hex is created from DAvE, so the modification / re-configuration is very easy.

The following are some of the parameters that can be modified easily by DAvE:

1. Oscillator setup (internal or external oscillator).
2. CAN ID (Acceptance Filter for receive / Transmit ID)
3. CAN Node baudrate
4. Message Object and LIST configuration.

The example is delivered with the following configuration:

1. External Oscillator is used.
2. Transmit ID = 0x008 and Receive Acceptance Filter and ID = 0x000 (accept all Frame)
3. CAN Node 0 baudrate is set to 100 KBps
4. Message Object 0 - 2 is set for receive FIFO and Message Object 8 is set for transmit object.

There are other parameters inside the file "**XC88x_FLADDR.H**" that needs to be modified (Default Setting is XC88xAA = 1). This is important for the Flash Protection feature:

```
#define XC88xAA 1  // Must be set to '1' (for AA step)
#define XC88xAB 0  // Must be set to '1' (set to '1' for AB step)
#define XC88xAC 0  // Must be set to '1' (Set to '1' for AC step)
```

Another configuration that can be done by user is located in the file "**MAIN.C**". These 2 files can be seen from the uVision project.

```
#define RX0_BASE_FIFO  0   // Define the Base Message Object for
                           // Receive FIFO
#define RX0_TOP_FIFO   2   // Define the Top Message Object for
                           // Receive FIFO
#define TX_MSGOBJ      8   // Define the Message Object for
                           // Transmit (No FIFO is supported for
                           // Transmit)
```

Please see the detail on the **Linker Option** for the Code area and Reserved Memory, as the is **limited code space** for the program (0xF000 - 0xF5FF) and special **reserved memory** for Flash Protection.

4.2 Main program flow

The main flow of the can_loader.hex is shown in the flow chart below.

The detail about the MODE will be shown in the chapter about the protocol of can_loader.hex.

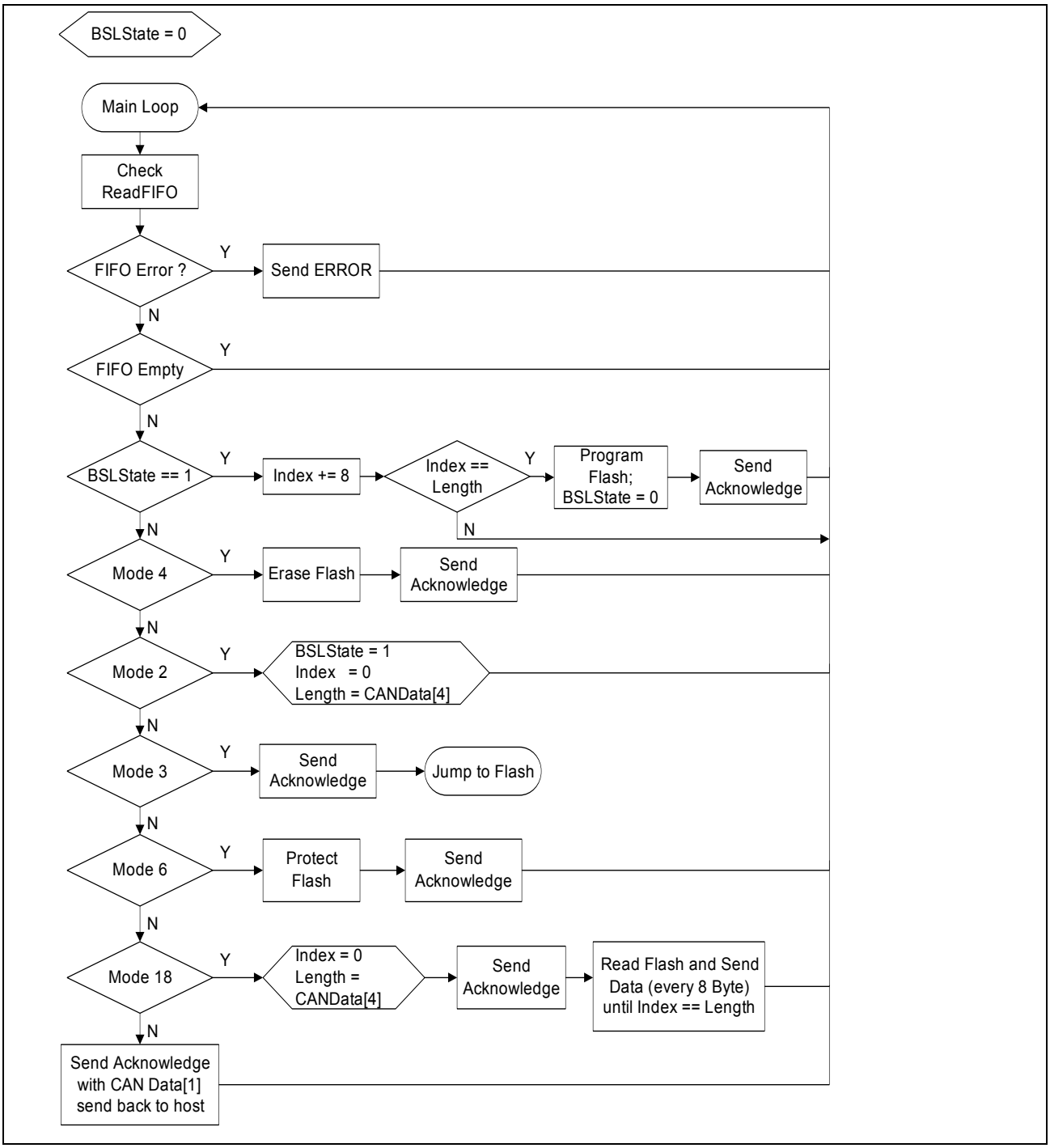


Figure 4 can_loader.hex main flow chart

4.3 CAN Loader Protocol

CAN ID:

CAN Loader can accept any ID, as it's receive message object acceptance filter is set to 0s.

Transmit ID is set to 0x08, but it is re-configurable as mentioned in **Chapter 4.1**

CAN Data:

CAN Loader always uses all 8 data bytes of a CAN Data Frame for it's communication. This way, the DLC (Data Length Code) can always be set as 8.

The meaning of the data bytes depends on the MODE.

Communication Protocol:

The communication is always started from the PC Host, by sending a CAN Data Frame.

The target will respond by either sending an acknowledge or data message (in case of Flash Read) back to the host.

There are 6 working MODEs that are supported by the can_loader.hex.

4.3.1 MODE 2

This mode is used to program the flash.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**02h**	**Addr H**	**Addr L**	**Data Length**	**Not Used**		

Addr H: High Byte of the address that will be programmed with the following CAN Data frame.

Addr L: Low byte of the address that will be programmed with the following CAN Data frame.

Data Length: The number of bytes that will be programmed. These bytes will be supplied by the following CAN Frames.

The PFlash will need Data Length = 64

The DFlash will need Data Length = 32

If Data Length = 64, there will be 8 CAN Frames (Data Block) that will follow the MODE Block.

If Data Length = 32, there will be 4 CAN Frames (Data Block) that will follow the MODE Block.

Data Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**

Byte n: Data Byte n

At the end of the last Data Block, the target will send the Acknowledge Response after approx. 5 ms (programming time)

If no response is sent, it means that the Flash Programming has failed and the target is reset.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
55h	**Data Length**	**Not Used (Ignored)**					

Data Length: The same data as the one sent by Mode Block.

4.3.2 MODE 3

This mode is used to execute the code from Flash.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**03h**	**Addr H**	**Addr L**	**Not Used**			

Addr H: High Byte of the address, where the program execution will be started.

Addr L: Low byte of the address, where the program execution will be started.

The target will send the acknowledge response before it jumps to the desired address. If no response is received by PC Host, something wrong has happened.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
55h	**00h**	**Not Used (Ignored)**					

4.3.3 MODE 4

This mode is used to erase one or more flash sectors.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**04h**	**PFlash Bank_Pair 0**	**PFlash Bank_Pair 1**	**PFlash Bank_Pair 2**	**Not Used**	**Option = 00h**	**Not Used**

When option is 00h, the target will use the information to erase PFlash Banks

Bank_Pair0: Bit0 = 1: Sector 0 will be erased
Bit1 = 1: Sector 1 will be erased
Bit 2 = 1: Sector 2 will be erased.

The same description applies for Bank_Pair1 and Bank_Pair2.

Thus, different PFlash banks can be erased at the same time.

OR

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**04h**	**DFlash Bank0 L**	**DFlash Bank0 H**	**DFlash Bank1 L**	**DFlash Bank1 H**	**Option = 40h**	**Not Used**

When option is 40h, the target will use the information to erase DFlash Banks

Bank0 L: Bit0 - Bit7 will represent Sector0 - Sector7 of DFlash Bank0.

Bank0 H: Bit0 - Bit1 will represent Sector8 - Sector9 of DFlash Bank0.

Setting '1' to any of this bit will erase the corresponding sector.

The same description applies for DFlash Bank1

OR

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**04h**	**Not Used**				**Option = C0h**	**Not Used**

When option is C0h, the target will erase all sectors in all PFlash and DFlash banks.

After a successful erase (approx. 105 ms), the target will send the Acknowledge Response.

If no response is sent, it means that the Flash Erasing has failed and the target is reset.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
55h	**00h**	**Not Used (Ignored)**					

4.3.4 MODE 6

This mode is used to install the password in the flash.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**06h**	**Password**	**Not Used**				

Password: 1 byte password that will be installed in the Flash Config Sector.

After the successful password installation (approx. 110 ms), the target will send the Acknowledge Response.

If no response is sent, it means that the Flash Protection has failed and the device state is unknown.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
STATUS	**00h**	**Not Used (Ignored)**					

STATUS: If STATUS = 55h, Flash Protection is done successfully
If STATUS = FDh, Flash Protection is not done.

4.3.5 MODE 18

This mode is used to read the flash.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**12h**	**Addr H**	**Addr L**	**Data Length**	**Not Used**		

Addr H: High Byte of the address that will be read.
Addr L: Low byte of the address that will be read.
Data Length: The length of the byte that will be read (max: 248)

Upon successful reception of the Mode Block, the target will send the acknowledgement response. If no response is sent, the state of the target device is un-known.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
55h	**00h**	**Not Used (Ignored)**					

After sending the acknowledge response, the target will send the read data 8 bytes at a time (minimum data sent is 8 bytes) with the Data Block, until all the requested data has been sent (e.g: 248 Data Length will result in 31 Data Blocks from Target to PC Host)

Data Block (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**

Byte n: Data Byte n

4.3.6 Loop Back Mode

This mode could be used to check the connection with the target device.

Mode Block (8 Byte) from PC Host to Target

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
00h	**XXh**	**Not Used**					

XX: Any number except the valid mode

Upon receiving this mode, the target will send the acknowledgment block with XXh in Data 1.

Acknowledge Response (8 Byte) from Target to PC Host

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7
55h	**XXh**	**Not Used (Ignored)**					

5 Important Note

5.1 Flash (Un) Protection

Starting from the AB-step, the device will be locked when the flash is protected.

This will mean that the CAN BSL mode can not be entered anymore.

Few options should be taken care of by user:

Option1: User code handles the protocol to un-protect the flash inside its code.

Once user code needs to un-protect the flash, it can call the routine:

bit _FlUnprotect(unsigned char password)

password = The password to un-protect the flash.

If the un-protection is successful, the whole PFlash will be erased.

Additional code =~ 27 Bytes

Option2: User code can include the following routine at the beginning of the code:

void _Check_MCANBSL (void)

This routine will check if the device is reset with BSL configuration (MBC = TMS = 0) or not. If it detects the BSL configuration, it will jump to the MCAN BSL and user can perform the same sequence as that for MCAN BSL.

Additional code =~ 31 Bytes

Option 3: User code can include the following routine at the beginning of the code:

void _Check_MCANBSL_FlUnprotect (void)

This routine will check if the device is reset with BSL configuration (MBC = TMS = 0) or not. If it detects the BSL configuration, it will execute the MCAN BSL initialization and check the CAN Frame. If the CAN Frame has the valid PASSWORD, it will un-protect the flash (the content of the protected flash bank will be erased, as described in the User Manual Chapter 18.1.2.6 and Chapter 3.4).

Please see the flow chart in **Figure 5**

Additional code =~ 74 Bytes

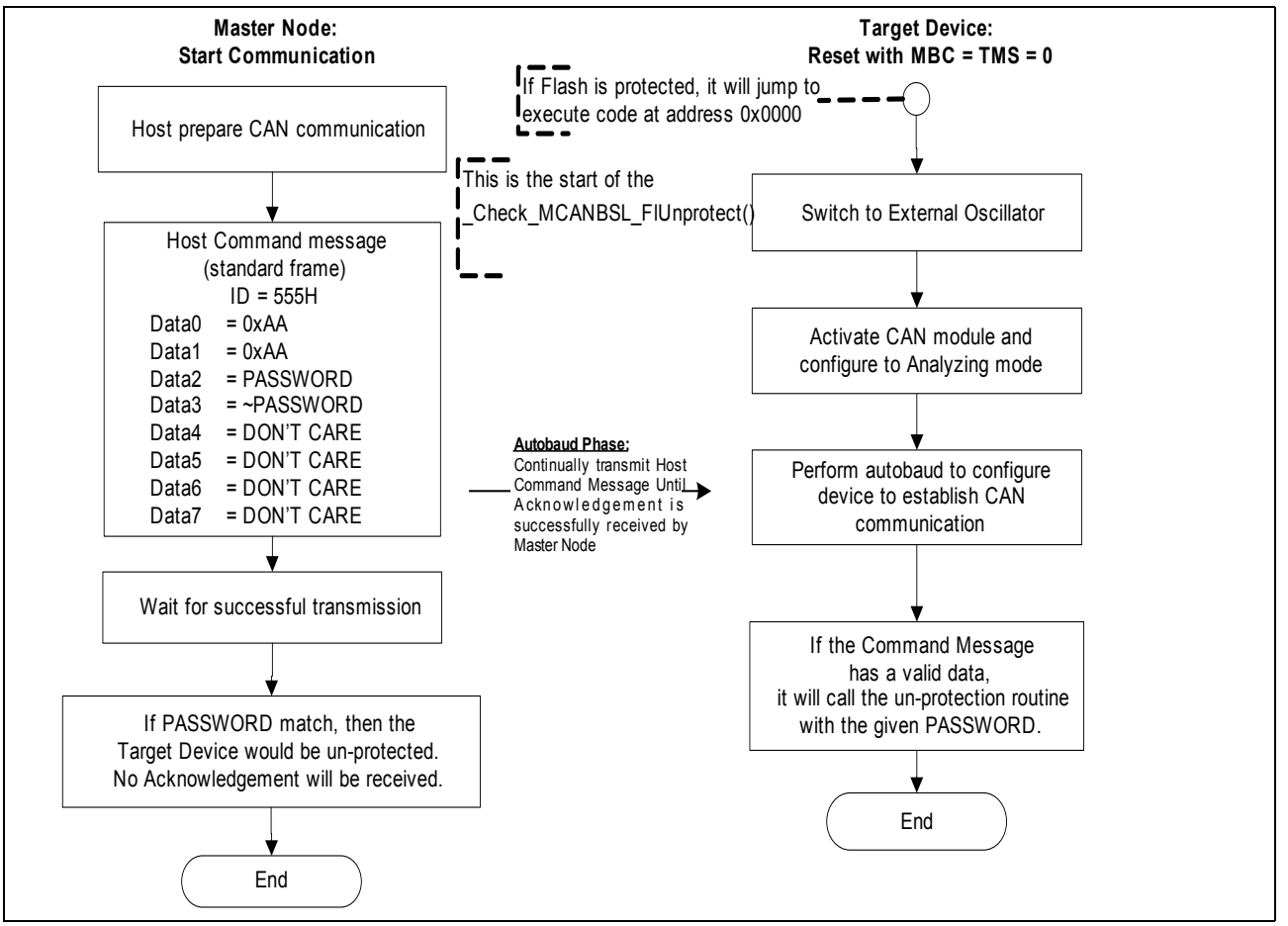


Figure 5 Un-protection flow

From the security point of view, Option 1 is recommended, as user has the full control of the access to the device. User can create his own protocol or protection to allow access to the device.

With Option2, anyone can erase and destroy the PFlash (with no possibility to read the PFlash content).

Option 3 is the best compromise between any of the options above. It does not have a risk on the security, as there is no access to the target device is possible, once the device is protected.

The file called: FlUnProtect_CAN.asm contains the 3routines above and can be inserted in the uVision project (at the moment, only Keil uViison is supported).

5.2 Auto Baud Rate

During the autobaudrate, it is recommended that the minimum baud rate is 100KBps.

For AA-step device, the maximum baudrate recommended is also 100KBps.

For AB-step and above, user can use a maximum baudrate of 1MBps.

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG