XC800 Family

AP08108

Programming the BMI value in the XC82x and XC83x products

Application Note

V1.0, 2010-07

Microcontrollers

Edition 2010-07

Published by
Infineon Technologies AG
81726 Munich, Germany

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 the 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 the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only 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.

XC82x,XC83x
Revision History: V1.0 2010-07

Previous Version(s):

Page	Subjects (major changes since last revision)
–	This is the first release …

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing? 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

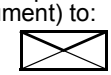


Table of Contents

1 Introduction

This application note is for the XC82x/XC83x products and describes the boot options' entry, usage and programming methodologies. For detailed descriptions of the XC82x/XC83x products, please refer to the XC82x/XC83x User's Manual.

1.1 Overview

The boot configuration for XC82x/XC83x does not depend on pin status during reset. Instead, a Boot Mode Index (BMI) configuration determines the entry to various boot modes such as User Mode, Boot-Loader (BSL) Mode and On-chip Debug (OCDS) Mode. After reset, the BMI value is taken and the respective boot mode entry is executed.

The advantage of using BMI configuration is that the port pins are not dedicated for boot-up and can be used for other purposes. The BMI is configured on-device, as described in the subsequent sections of this document. **Figure 1** shows an overview of boot mode entry methodologies for the XC800 family of products.

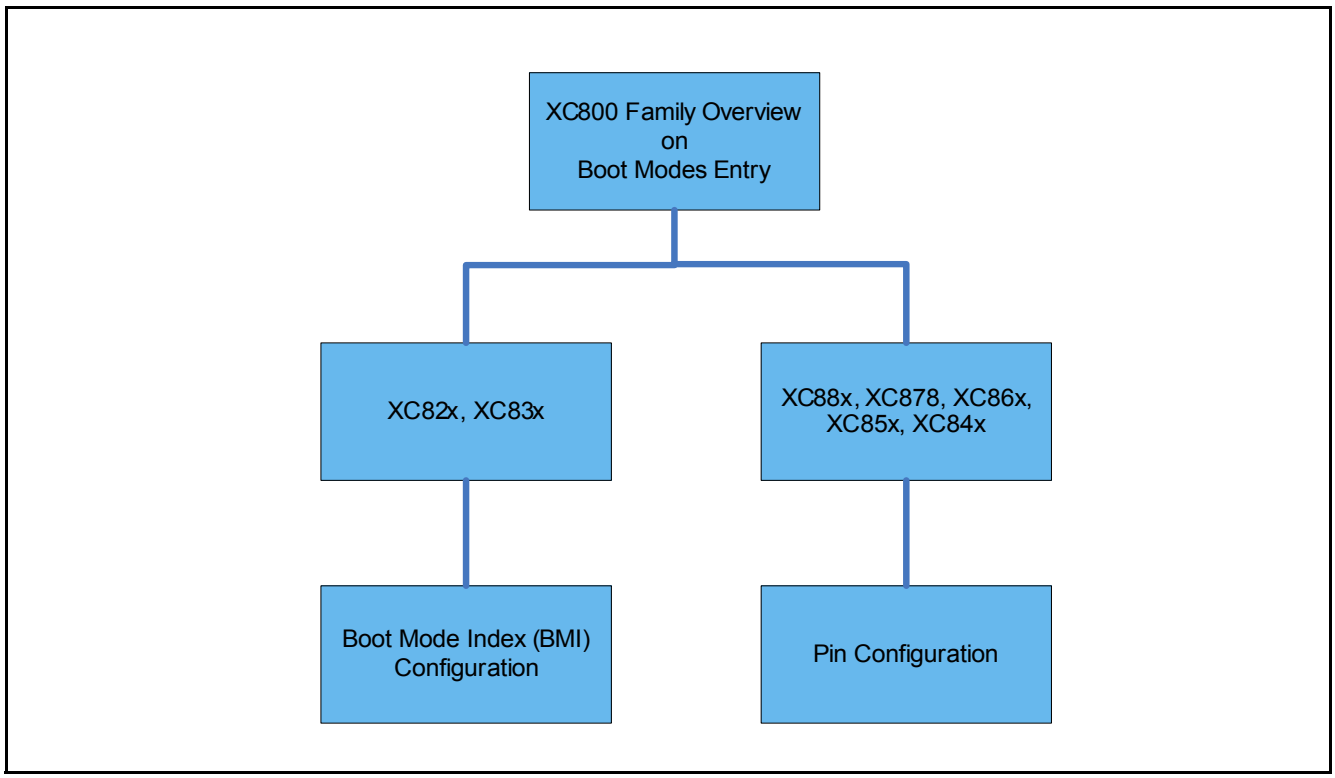


Figure 1 Boot Mode entry of XC800 family

2 Boot Modes

The BMI and $\overline{\text{BMI}}$ values are configured on-chip using a variety of methods (refer to **Chapter 3**). Upon power-up, the BMI and $\overline{\text{BMI}}$ values will be evaluated to determine the boot mode execution after reset. The supported boot modes are shown in **Figure 2** and are described in **Section 2.3**.

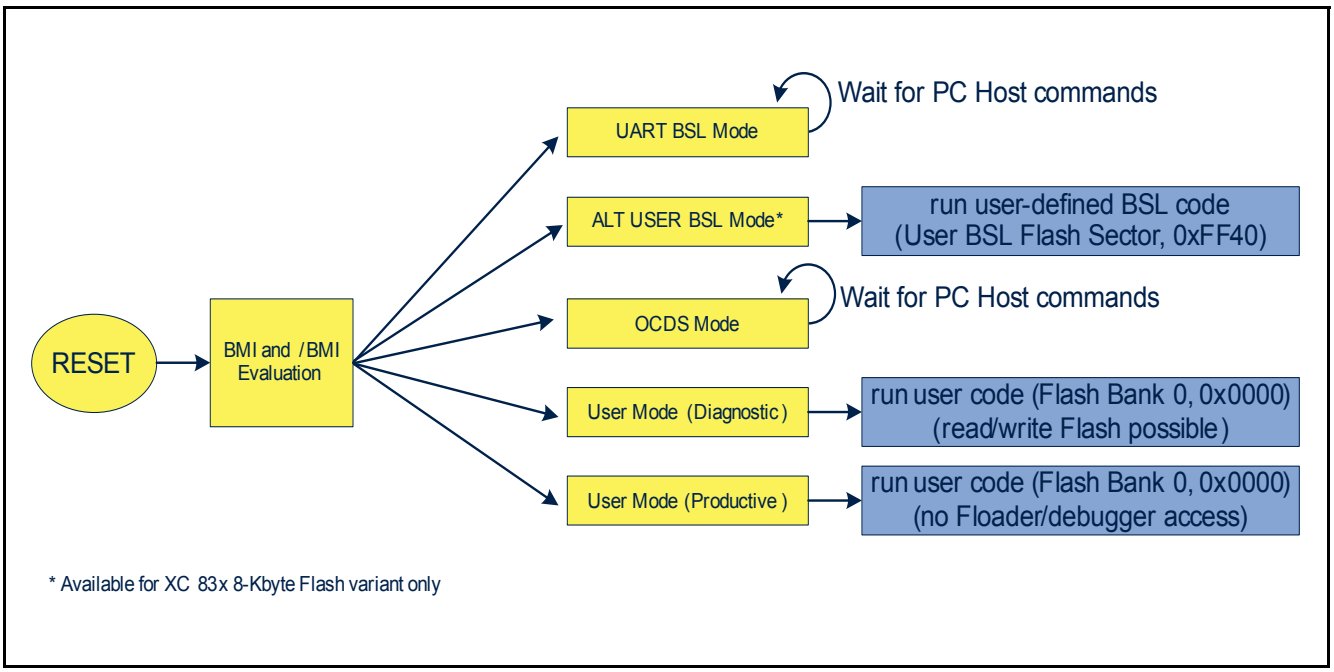


Figure 2 Boot ROM operating mode after power-on reset

2.1 Boot Mode Index

The user needs to program the BMI and $\overline{\text{BMI}}$ values to enter the respective boot mode. Devices are typically delivered with the BMI configured by default to UART Boot-Loader (BSL) Mode. $\overline{\text{BMI}}$ is introduced to verify that the BMI value is valid; i.e. (BMI + $\overline{\text{BMI}}$ + 1 = 0). **Table 1** and **Table 2** show the BMI and $\overline{\text{BMI}}$ values relating to different boot modes on the XC82x and XC83x respectively.

Table 1 Boot Mode Index for XC82x

Boot Mode[1)]	BMI	$\overline{\text{BMI}}$	Description
UART Boot-Loader (BSL) Mode[2)]	00_H	FF_H	Supports programming and erasing of Flash and XRAM via UART (full and half duplex).
	00_H	00_H	
User Mode (Productive)[3)]	10_H	EF_H	Execution of the user code in Flash memory, starting from address 0x0000. Debugging is not supported.
User Mode (Diagnostic) with SPD pin SPD_0	50_H	AF_H	Execution of the user code in Flash memory, starting from address 0x0000. Debugging is supported. Corresponding SPD pin is automatically configured to allow hot-attach.
User Mode (Diagnostic) with SPD pin SPD_1	52_H	AD_H	

Table 1 Boot Mode Index for XC82x (cont'd)

Boot Mode[1)]	BMI	$\overline{BMI}$	Description
OCDS Mode with SPD pin SPD_0	60_H	$9F_H$	Support debugging of the user code in the Flash memory via corresponding single-pin DAP (SPD).
OCDS Mode with SPD pin SPD_1	62_H	$9D_H$	
Reserved	Others	Others	BMI values not defined above are not supported. The user should not configure undefined BMI values.

1) SPD : Single Pin DAP;
2) UART BSL Mode can be entered when BMI is 00 and $\overline{BMI}$ is either 00 or FF.
3) This mode should be used with extra care. If there is a need to re-configure the BMI after entering this mode, it will be necessary to embed an update-code in with the user code to be able to enter other modes from User Mode (Productive). For more details, refer to **Section 2.3.5** and **Chapter 4**.

Table 2 Boot Mode Index for XC83x

Boot Mode[1)]	BMI	$\overline{BMI}$	Description
UART Boot-Loader (BSL) Mode[2)]	00_H	FF_H	Supports programming and erasing of Flash and XRAM via UART (full and half duplex).
	00_H	00_H	
Alternate User Boot-Loader (BSL) Mode[3)]	01_H	FE_H	Execution of the user's own customized BSL code, at address 0xFF40. BSL code is programmed at User BSL Flash sector (0xFF40-0xFF7F).
User Mode (Productive)[4)]	10_H	EF_H	Execution of the user code in Flash memory, starting from address 0x0000. Debugging is not supported.
User Mode (Diagnostic) with SPD pin SPD_0	50_H	AF_H	Execution of the user code in Flash memory, starting from address 0x0000. Debugging is supported. Corresponding SPD pin is automatically configured to allow hot-attach.
User Mode (Diagnostic) with SPD pin SPD_1	52_H	AD_H	
User Mode (Diagnostic) with SPD pin SPD_2	54_H	AB_H	
OCDS Mode with SPD pin SPD_0	60_H	$9F_H$	Support debugging of the user code in the Flash memory via corresponding single-pin DAP (SPD).
OCDS Mode with SPD pin SPD_1	62_H	$9D_H$	
OCDS Mode with SPD pin SPD_2	64_H	$9B_H$	
Reserved	Others	Others	BMI values not defined above are not supported. The user should not configure undefined BMI values.

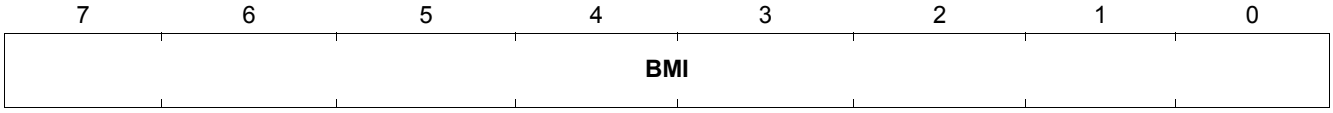
1) SPD : Single Pin DAP;
2) UART BSL Mode can be entered when BMI is 00 and $\overline{BMI}$ is either 00 or FF.
3) Available only in the 8-Kbyte Flash variant.
4) This mode should be used with extra care. If there is a need to re-configure the BMI after entering this mode, it will be necessary to embed an update-code in with the user code to be able to enter other modes from User Mode (Productive). For more details, refer to **Section 2.3.5** and **Chapter 4**.

Note: When BMI is not valid (i.e. BMI + $\overline{BMI}$ + 1 != 0), the default mode is UART BSL Mode, provided that the BMI is not programmed as 0x10 or $\overline{BMI}$ is not programmed as 0xEF. Under those conditions, software protection is initiated and the device will enter a deadlock situation.

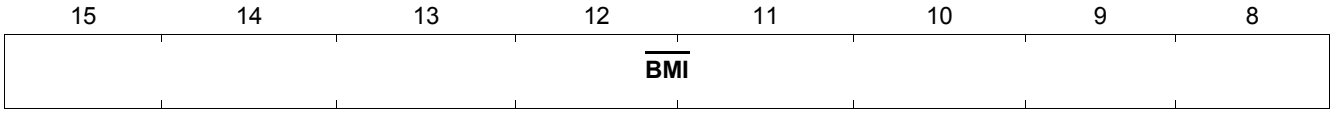
2.2 User Identification Number

The BMI and $\overline{\text{BMI}}$ are part of the User Identification, USER_ID. The USER_ID consists of 4-bytes of data that contain the user specific information, as shown in **Table 3**. To update the BMI and $\overline{\text{BMI}}$ values, the user needs to program the whole 4 bytes of USER_ID.

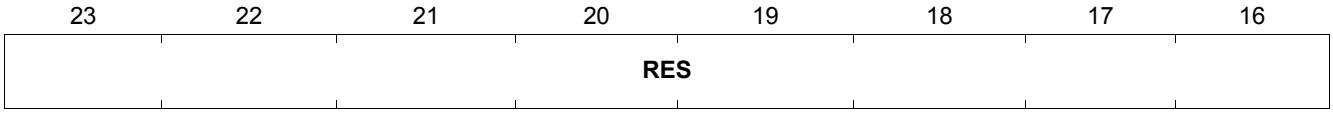
USER_ID, Byte 0



USER_ID, Byte 1



USER_ID, Byte 2



USER_ID, Byte 3

Table 3 USER_ID

USER_ID Field	Byte	Bits	Description
BMI	Byte 0	[7:0]	**Boot Mode Index** The boot option to enter User Mode, UART BSL Mode and OCDS Mode as described in **Section 2.1**
$\overline{\text{BMI}}$	Byte 1	[15:8]	**Inverse of Boot Mode Index** The inverse of BMI. BMI is checked for validity with $\overline{\text{BMI}}$. BMI is valid when BMI + $\overline{\text{BMI}}$ + 1 = 0
RES	Byte 2	[23:16]	**Reserved**

Table 3 USER_ID (cont'd)

USER_ID Field	Byte	Bits	Description
RES	Byte 3	[29:24]	**Reserved**
PERIPHERALS_EN		30	**Peripherals Enable Bit** 0_B Disable all peripherals defined in register PMCON1 1_B Enable all peripherals defined in register PMCON1
CLKMODE_SEL		31	**Clock Mode Selection** 0_B 8 MHz active mode 1_B 24 MHz active mode

2.3 Boot ROM Operating Mode

The respective Boot ROM operating modes are described in this section.

2.3.1 UART Boot-Loader (BSL) Mode

If the Boot-loader (BSL) mode is selected, the BSL software routines are executed and waiting for BSL commands from the PC host. By sending UART_INIT_ID, the PC host can choose to communicate with the target device via UART full-duplex or UART half-duplex. In BSL mode, the PC host can download and execute code in XRAM/Flash, erase code in Flash, program 4-byte USER_ID (which contains the BMI and $\overline{\text{BMI}}$ value), and read the CHIP_ID and USER_ID information. The overview of port pin(s) used in UART BSL Mode is shown in **Table 4**.

Table 4 Port Settings for UART BSL Mode

Device	UART_INIT_ID	Full/Half Duplex	RXD Pin	TXD Pin
XC82x	0xAA	Half duplex	P0.6	P0.6
	0x55	Full duplex	P0.6	P0.5
XC83x	0xAA	Half duplex	P3.2	P3.2
	0x55	Full duplex	P3.2	P0.7

2.3.2 Alternate User Boot-Loader (BSL) Mode

If the Alternate User Boot-loader (BSL) mode is selected, the user-customized BSL code in the User BSL Flash Sector will be executed. This code has to be first programmed into the 64 bytes User BSL Flash Sector, located in the program memory address range 0xFF40 to 0xFF7F, before the Alternate User BSL Mode is selected.

Note: This mode is only available in the XC83x *8-Kbyte Flash variant.*

2.3.3 OCDS Mode

If the OCDS mode is selected, the OCDS hardware is initialized and jump to program memory address 0x0000. The device then waits for SPD commands from PC host. The user can access to full debug functionality using SPD pin. This is the recommended mode when a user starts to develop and debug their own application code. **Table 5** gives an overview of the port pin(s) used in OCDS Mode.

Table 5 Port Settings for OCDS Mode

Device	SPD	Pin used
XC82x	SPD_0	P0.6
	SPD_1	P1.0

Table 5 Port Settings for OCDS Mode (cont'd)

Device	SPD	Pin used
XC83x	SPD_0	P3.2
	SPD_1	P0.6
	SPD_2	P1.0

2.3.4 User Mode (Diagnostic)

If the User Mode (Diagnostic) is selected, the Boot ROM will jump to the program memory address 0x0000 to execute the user code in the Flash memory, and the specified SPD port is automatically configured to allow debugger access. The user can stop the code and read/write to Flash using the SPD pin. This is the recommended mode when the user has completed code debugging and wants to test the code in a real application environment. The user can also update their working code using the Flash loader or debugger. **Table 6** gives an overview of the port pin(s) used in OCDS Mode.

Changing the BMI value to enter another boot mode should be taken into consideration, especially under scenarios, where the user code:

- enters Idle Mode periodically or
- enters Power Down Mode periodically or
- enables WatchDog Timer

See **Chapter 4** for code details.

Table 6 Port Settings for User Mode (Diagnostic)

Device	SPD	Pin used
XC82x	SPD_0	P0.6
	SPD_1	P1.0
XC83x	SPD_0	P3.2
	SPD_1	P0.6
	SPD_2	P1.0

2.3.5 User Mode (Productive)

When User mode (Productive) is selected, the Boot ROM jumps to the program memory address 0x0000 to execute the user code in the Flash memory. The difference between User Mode (Diagnostic) and User Mode (Productive) is that User Mode (Productive) provides Flash memory protection from external access (read/write) from debugger and Flash loader. User Mode (Productive) should be used when the user has fully tested their code and no further code modification is allowed.

Once the device is in User Mode (Productive), changing the BMI value to enter another boot mode could be achieved by programming a specific code embedded in the user code. This specific code is called under user-defined conditions; via an interrupt for example, or via GPIO pin values.

In this specific code, the user can use one of the following instructions to change the BMI value:

- LJMP to BR_UART_BSL Boot ROM user routine (0xDFEA) to re-enter UART BSL Mode (Mode 6) or
- LCALL of *BR_PROG_USER_ID* Boot ROM user routine (0xDFE4) to re-program USER_ID to enter other mode

Please refer to the User's Manual for the input settings for these Boot ROM user routines. For code protection purposes, the user may want to erase all the important code or data in the flash memory before executing that specific routine. See **Chapter 4** for more code details.

3 Programming the BMI value

The default Boot mode for XC82x or XC83x is UART BSL Mode. The BMI value needs to be programmed to allow the devices to work in OCDS Mode for debugging. The BMI also needs to be programmed to User Mode for running the final working application code. The following points should be considered when programming the BMI value:

- The VDDP of the target device should keep stable at 3V or 5V when programming the BMI value.
- A software reset is executed after the BMI value has been updated. Switching off the supply voltage before the software reset has occurred may cause the BMI to be updated incorrectly.
- No additional reset is necessary for the system to use the new BMI value.
- The Watchdog Timer (WDT) should be disabled before programming the BMI value, because if a WDT reset occurs when programming the value, it may cause the BMI to be updated incorrectly.
- Four bytes of USER_ID are to be programmed together in order to write the BMI value.

Note: Great care must be taken when programming the BMI value, because a wrong or invalid BMI could cause the device to enter a deadlock situation.

Software tools are available to facilitate programming the BMI value, such as the DAvE-Bench (XC800_FLOAD) tool from Infineon and the uVision 4 from KEIL. Both software tools require the following components for BMI programming:

- **DAS DLL**: a DLL driver for the software tool to call the DAS API to send SPD/UART signals through a hardware device, DAP miniWiggler.
- **DAP miniWiggler**: a USB powered device controlled by the DAS API to produce SPD/UART signals.

P0.6 of XC82x and P3.2 of XC83x can function both as a UART BSL (half-duplex) pin and a SPD pin, as shown previously in **Table 4** and **Table 5** respectively. Therefore the DAS API supports both UART communication and SPD communication on the same pin. This feature enables the user to use the same pin to perform the following actions:

- Enter UART BSL Mode (default)
- Program the BMI value (via UART BSL Mode 6) from UART BSL Mode to OCDS Mode[1)]
- Debug user application code in OCDS Mode via SPD interface directly

The hardware setups and the software tools for programming the BMI value are described in **Section 3.1** and **Section 3.2** respectively. The links of the board and tools are listed in **Section 3.3**.

3.1 Hardware for BMI Programming

This section will describe the two hardware setups that work with the Infineon DAvE-Bench (XC800_FLOAD) and KEIL uVision 4.

When the user is using their own target board for BMI programming, a DAP miniWiggler setup as described in **Section 3.1.1** would be needed. The XC822-TSSOP16 board (a XC822 Easy Kit board) and XC836-TSSOP board (a XC836 Easy Kit board) have the SPD wiggler integrated on-board, as shown in **Section 3.1.2**. Hence, the user could use the DAvE-Bench(XC800_FLOAD) or KEIL uVision 4 to change the BMI value directly on the Easy Kit board.

1) After the BMI value is changed, the device will automatically perform a software reset and boot up in OCDS Mode

3.1.1 Using a Target Board with DAP miniWiggler (JTAG/DAP/SPD V2.0)

With the miniWiggler, a user can program the XC82x or XC83x device on the target application board by setting up the 3 pins as shown in **Figure 3** and **Figure 4** respectively. The DAP miniwiggler requires the VDDP (5V or 3.3V) from the target board to power the SPD signal on the DAP miniWiggler.

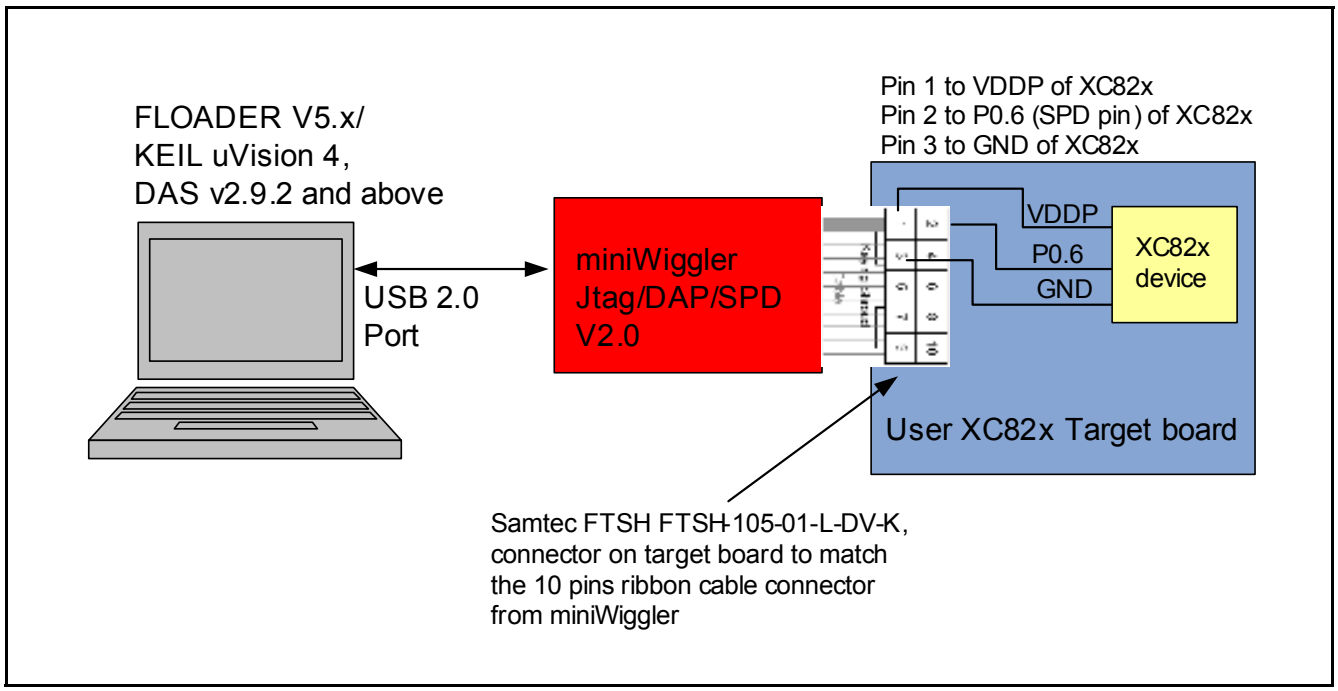


Figure 3 Hardware Setup of DAP miniWiggler with the User's XC82x Target Board

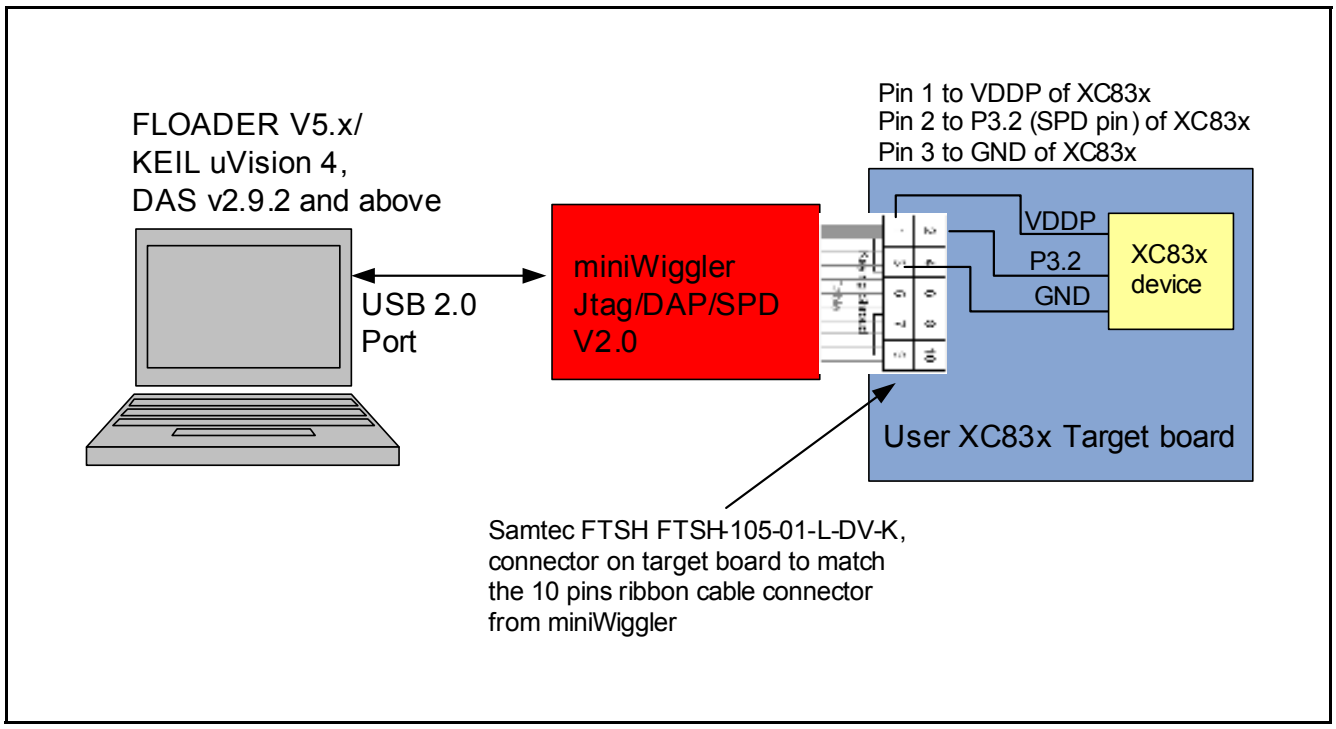


Figure 4 Hardware Setup of DAP miniWiggler with the User's XC83x Target Board

3.1.2 Using the Easy Kit Board

The SPD wiggler is integrated on the XC822/XC836 Easy Kit board. The SPD pin of the on-board wiggler is connected to P0.6 of XC822 Easy Kit board and P3.2 of the XC836 Easy Kit board, by using the jumper setting at the COM_SEL, as shown in **Figure 5** and **Figure 6** respectively.

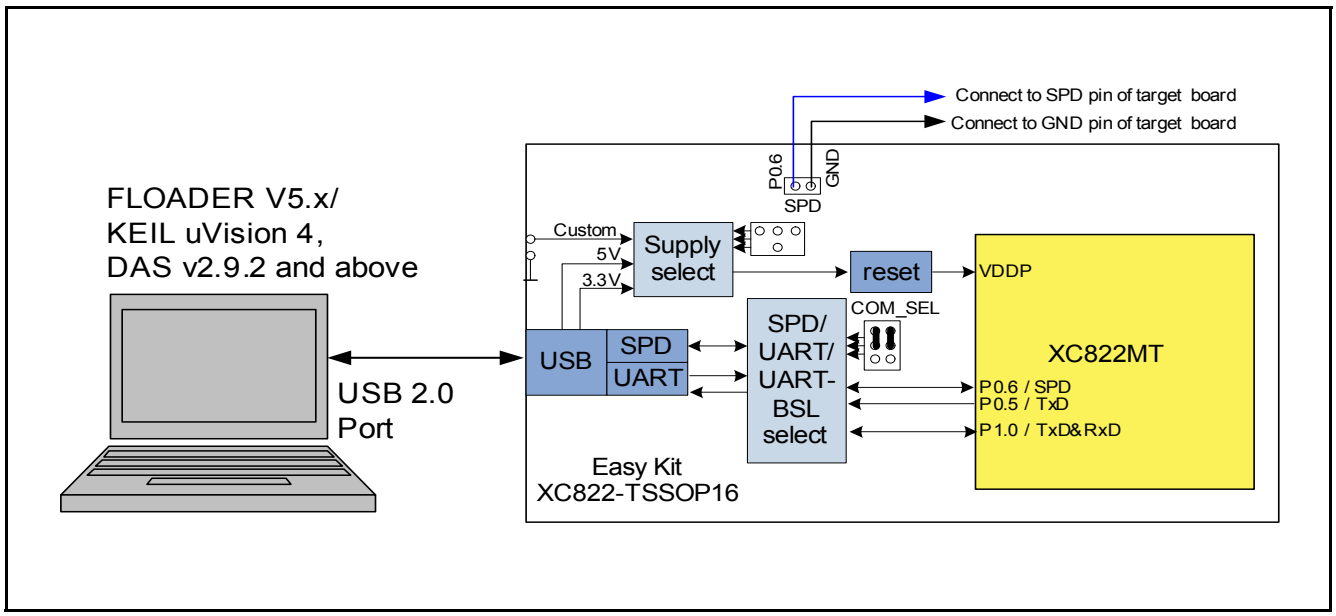


Figure 5 Hardware Setup using the XC822 Easy Kit Board

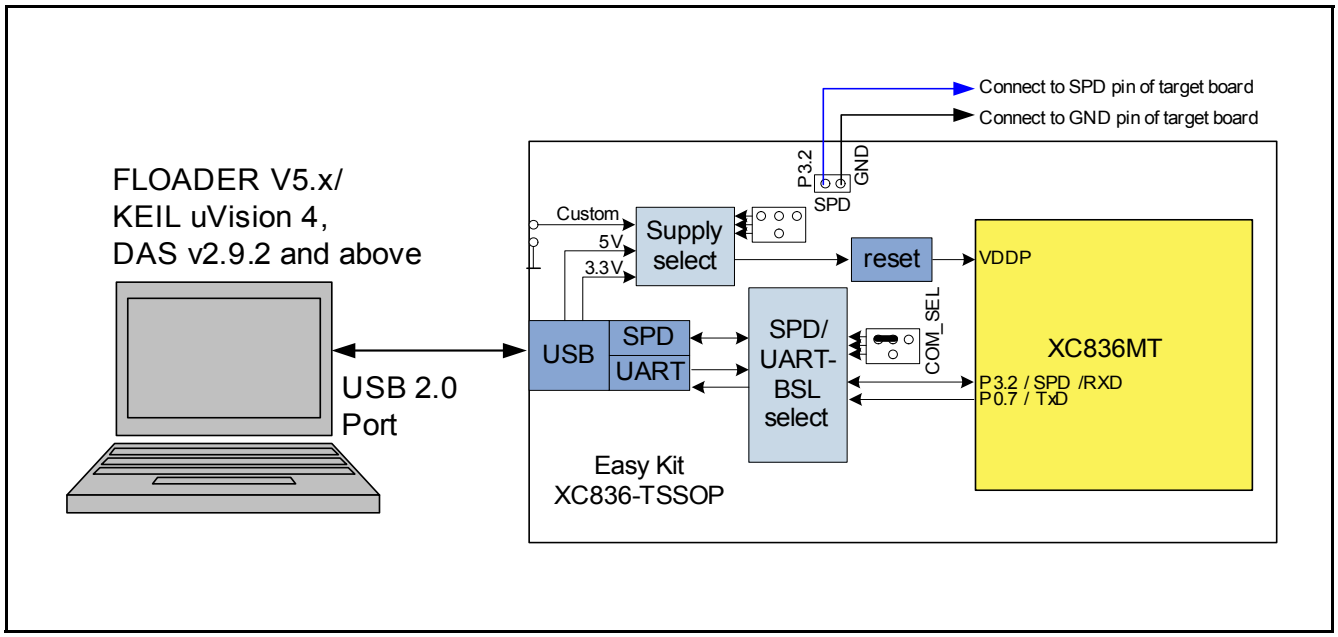


Figure 6 Hardware Setup using the XC836 Easy Kit Board

Note: The XC822/XC836 Easy Kit board can also act as SPD wiggler for connection to the user target board, enabling SPD debug functions and/or single-pin half-duplex UART BSL functions. By acting as SPD wiggler, the Easy Kit board can be used to program the BMI value of the XC82x/XC83x device on the user target board; i.e. there is no need to purchase a separate wiggler. When using it as SPD wiggler, the XC822/XC836 device on the Easy Kit board has to be removed.

3.2 Software Tools for BMI Programming

Software tools are available for BMI programming. The DAvE-Bench (XC800_FLOAD) tool from Infineon for example, and uVision 4 from KEIL, are described in **Section 3.2.1** and **Section 3.2.2** respectively.

3.2.1 Using the DAvE-Bench's Flash loader (XC800_FLOAD) Tool

The section describes using the DAvE-Bench (XC800_FLOAD) tool from Infineon to program the BMI value, depending on the current XC82x/XC83x boot mode. Please refer to the Hardware manual in "XC82x/XC83x Easy Kit CD Content" of the XC82x/XC83x Easy Kit website for the COM_SEL jumper setting.

First, run the DAvE-Bench Application and click on the XC800_FLOAD icon as shown in **Figure 7**. The XC800_FLOAD application will be executed and the user should follow the procedures shown below for BMI value programming.

DAvE-Bench (XC800_FLOAD) procedure to program BMI in UART BSL Mode:

- Select XC800_FLOAD
- Select Protocol -> "UART" or "JTAG/SPD"
- Select Physical Interface:
  - For "UART" protocol, select between Full Duplex and Half Duplex/Ignore Echo[1)]
  - For "JTAG/SPD" protocol, "UDAS/JTAG over USB" is the only option available [2)]
- Select Target Device -> XC83x-2F (for example)
- Enter COM port number if "UART" protocol is selected
- Click "Write User ID" button
- Select Boot Mode, Peripherals Enable/Disable and Clock Mode once a connection is established successfully
- Click "OK" after selection is made -> Wait for "BMI switched" Message

DAvE-Bench (XC800_FLOAD) procedure to program BMI in User Mode (Diagnostic) or OCDS Mode:

- Select XC800_FLOAD
- Select Protocol -> "JTAG/SPD" [2)]
- Select Physical Interface -> "UDAS/JTAG over USB"
- Select Target Device -> XC83x-2F (for example)
- Click "Write User ID" button
- Select Boot Mode, Peripherals Enable/Disable and Clock Mode once a connection is established successfully
- Click "OK" after selection is made -> Wait for "BMI switched" Message

Note: If the XC82x/XC83x *is in UART BSL Mode and the selected physical interface is "JTAG/SPD", a message box appears asking the user for permission to switch the device to OCDS mode SPD_0. Select "Yes".*

The DAvE-Bench (XC800_FLOAD) GUI for programming of the BMI value is shown in **Figure 7**.

1) This selection works with the **Figure 5** and **Figure 6** setup but COM_SEL on the Easy Kit board must be set to UART BSL Mode.

2) This selection works with the **Figure 3**, **Figure 4**, **Figure 5** and **Figure 6** setup, but COM_SEL on **Figure 5** and **Figure 6** must be set to SPD mode.

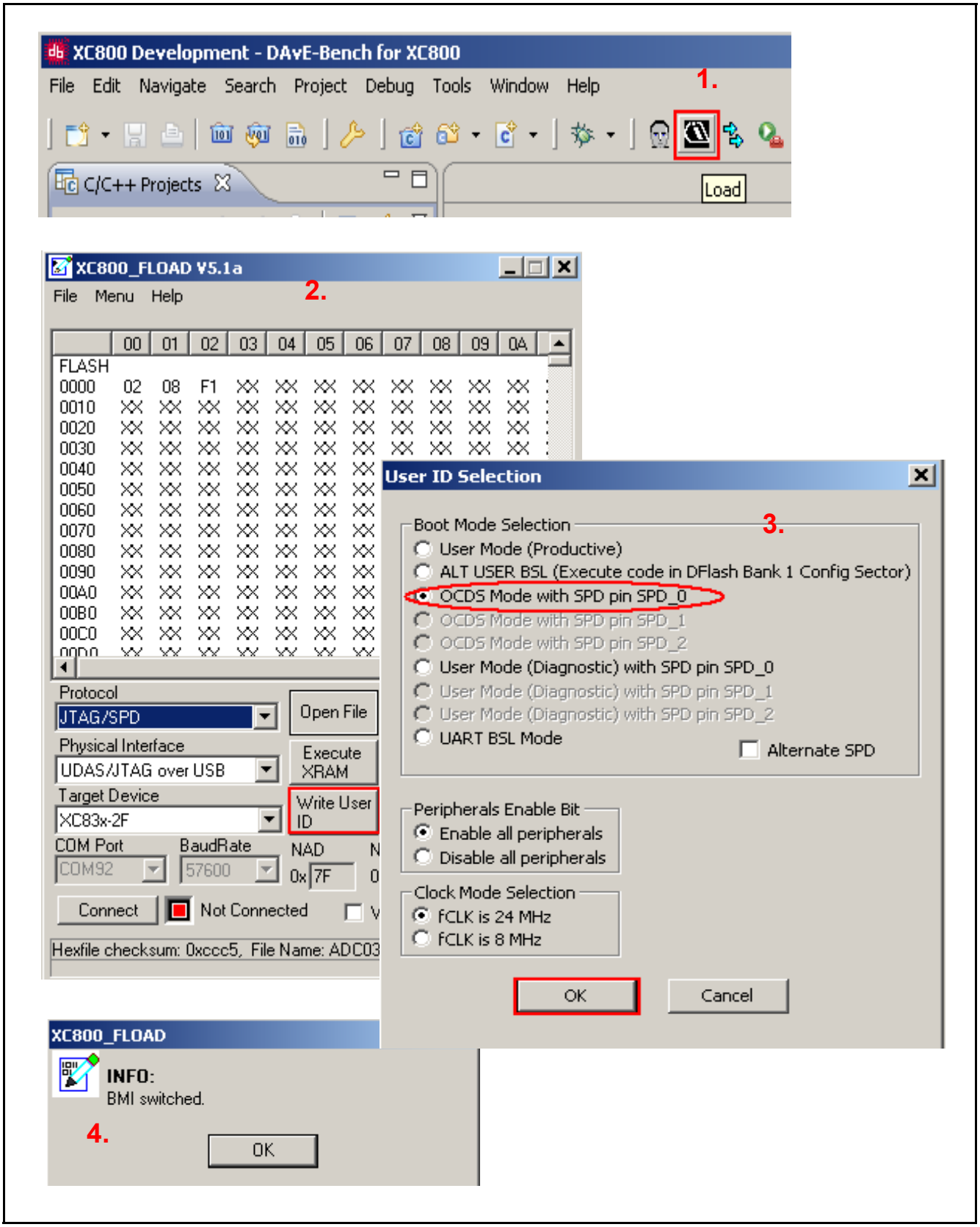


Figure 7 DAvE-Bench (XC800_FLOAD) GUI for Programming the BMI Value

3.2.2 Using the KEIL uVision 4 Tool

This section describes using the KEIL uVision 4 tool to program the BMI value. KEIL uVision 4 works with **Figure 3**, **Figure 4**, **Figure 5**, and **Figure 6** setup, but COM_SEL for **Figure 5** and **Figure 6** setup must be set to SPD mode. Please refer to the Hardware manual in the “XC82x/XC83x Easy Kit CD Content” of the XC82x/XC83x Easy Kit website for the COM_SEL jumper setting. The GUI of KEIL uVision 4 of programming of the BMI value is shown in **Figure 8**.

uVision 4 procedure to program BMI:

- Click on “Debug” Tab. Select “Use: -> “Infineon DAS Client for XC800””
- Click “Setting” button -> Infineon XC800 DAS Driver Setup GUI
- Select DAS Server -> “UDAS”
- Select Device: -> “XC800-Family”
- Click “BMI (XC82x/XC83x only)” Tab -> Choose BMI selection for desired boot mode
- Click “Set BMI now” button

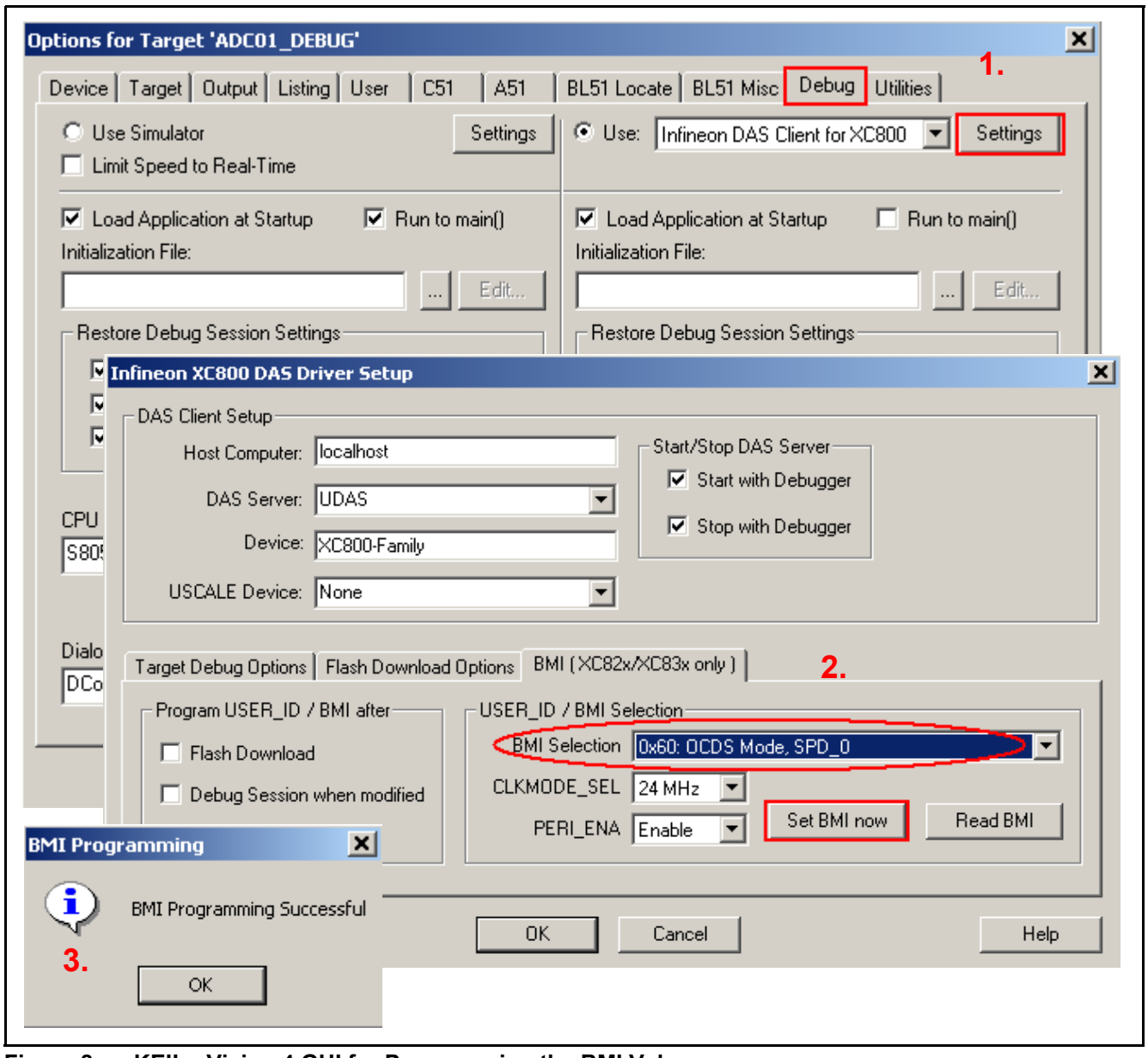


Figure 8 KEIL uVision 4 GUI for Programming the BMI Value

3.3 Tool Links

For more information on using the software tools, miniWiggler, boards and driver, please refer to the links shown in Table 7.

Table 7 BMI Programming Tools

Tool	Version	Link
DAvE-Bench (Flash loader XC800_FLOAD)	V1.0.2 and above	**www.infineon.com/davebench**
KEIL uVision 4	V9.01 and above	**www.keil.com/demo/eval/c51.htm**
DAP miniWiggler	V2.0	**www.infineon.com/miniwiggler**
Easy Kit XC82x	V1.0 and above	**www.infineon.com/xc82x-easykit**
Easy Kit XC83x	V1.0 and above	**www.infineon.com/xc83x-easykit**
DAS DLL	V2.9.2 and above	**www.infineon.com/das**

4 Handling Tips

In UART BSL Mode, the BMI can be easily programmed via UART BSL Mode 6 (using the XC800_FLOAD tool). In OCDS Mode or User Mode (Diagnostic), the BMI can be updated through software tools (XC800_FLOAD and Keil uVision 4, for example) as described in **Chapter 3**. However, to program the BMI in **User Mode (Productive)**, an ***update-code*** needs to be embedded in the user code.

The ***update-code***, whether in **User Mode (Productive)** or **User Mode (Diagnostic)**, must take into account those cases where the user code:

- enters Idle Mode periodically
- enters Power Down Mode periodically
- enables WatchDog Timer (WDT)

In User Mode (Diagnostic), trigger of the SPD pin is not effective when the device is in Idle Mode or Power Down Mode. An interrupt, for example, can be triggered to exit Idle Mode or Power Down Mode permanently, or to disable the WDT before programming the BMI value.

The flow is:

- Include the ***update-code*** in the user code with ***user-defined condition***
- Compile and get hex file
- Program the user code in Flash
- Program BMI to User Mode (Productive)
- No external access on XRAM/Flash is allowed once BMI is programmed to User Mode (Productive)
- To update the BMI or the user code for upgrading process, trigger the ***user-defined condition***
- Depending on the ***update-code***, BMI will be programmed or UART BSL Mode will be entered for updating the user code or the BMI value

The ***user-defined condition*** could be interrupt or polling based, or can use any other method that the user deems applicable to their application. The ***update-code*** is either:

- LJMP to BR_UART_BSL Boot ROM user routine (0xDFEA) to re-enter BSL Mode (Mode 6) or
- LCALL of *BR_PROG_USER_ID* Boot ROM user routine (0xDFE4) to re-program USER_ID to enter other mode

An comparison of update-code methods is shown in **Table 8**. The safer approach, LJMP to BR_UART_BSL is recommended. This method also allows the user to update the user code without changing the BMI value.

Note: The BMI value should be changed on entering UART BSL Mode, instead of updating the user code, especially when downloading 'untested' code. This is to prevent the download of the user code that does not include the update-code necessary to re-program BMI or re-enter UART BSL Mode.

Table 8 Comparison of Update-Code Methods

BR_UART_BSL	BR_PROG_USER_ID
The user can access device via UART BSL mode immediately after the LJMP without changing the BMI value.	BMI value is changed accordingly.
External access only via UART pin. Refer to **Table 4**.	External access depends on BMI setting. Refer to **Section 2.3**.
Unintentional execution will not affect the BMI value. If BMI is unchanged, User Mode (Productive) is re-entered after a reset.	Unintentional execution will affect the BMI value. Device can only boot up properly with valid BMI configured. There must be a user-defined condition to ensure this LCALL cannot be unintentionally executed.

4.1 Program BMI via BR_UART_BSL

This section includes example code which uses an interrupt to enter UART BSL Mode via the BootROM user routine BR_UART_BSL. The code uses P2.0 to emulate as a Reset pin (falling edge) to trigger the external interrupt 0 where a softreset will be executed. P2.0 and P2.1 are checked when the device restarts. When the condition is met, the BR_UART_BSL user routine is called and UART BSL Mode is entered. The user can program the BMI or update the user code in Flash via XC800_FLOAD.

Note: For code protection purpose, the user may want to erase all the important code or data in the flash memory before re-entering UART BSL Mode. Therefore, this code must be located at a different Flash sector so that other important code or data can be erased, while this code remains. This code ENTER_UART_BSL() can be located at the smallest Flash sector.

Note: This BR_UART_BSL user routine does not return back to the user code.

```
unsigned int idata j=0x1800;
void ENTER_UART_BSL (void)
{
 // This code located at different flash sector
 // Erase important code or data if needed before entering UART BSL Mode
 ......
 // Enter UART BSL Mode to program BMI or update user code in Flash
 EA = 0;              // disable interrupts
 NMICON = 0x00;       // disable NMIs
 SP = 0xE0;           // Configured SP
 BR_UART_BSL();       // start UART BSL
}

void main(void)
{
 if( (!(P2_DATAIN & 0x02)) && (P2_DATAIN & 0x01) )
 {
       while (--j); // impose delay for system to be ready
       if( !(P2_DATAIN & 0x02) ) //Only enter BSL when P2.1 is low
       {
       ENTER_UART_BSL();
       }
 }
 MAIN_vInit();
//user code begins
}

void INT_vInit(void)
{
 NMICON = 0x00; //NMI Control Register
 IEN0 |= 0x01; // interrupt enable register 0
 IEN1 |= 0x00; // interrupt enable register 1
 EXICON0 = 0xF3; // External interrupt Control Register 0
 SFR_PAGE(_su3, noSST); // switch to page 3 without saving
 MODPISEL1 |= 0x18; // peripheral input select register 1
 SFR_PAGE(_su0, noSST); // switch to page 0 without saving
}// End of function INT_vInit
```

```
void INT_viExt0(void) interrupt EXT0INT
{
 // USER CODE BEGIN (IR_IsrExt0,2)
 // emulate Reset pin an falling edge
 SFR_PAGE(_su1, noSST);
 while ( !(P2_DATAIN & 0x01) );
 MAIN_vSoftReset();
 // USER CODE END
...
} // End of function INT_viExt0
```

4.2 Program BMI via BR_PROG_USER_ID

This section includes example code which uses an interrupt to program the BMI value directly via the BootROM user routine BR_PROG_USER_ID. The code uses P2.0 to trigger the external interrupt 0, which calls this routine. In this example, the BMI value is programmed to enter OCDS Mode.

Unintentional execution of the BR_PROG_USER_ID user routine will affect the BMI value and the device may not be able to boot up properly because of an invalid BMI. Therefore a user-defined condition must be included to prevent an unintentional jump to this routine. An extra password (PASSWORD_BMI) is included in this example.

Note: For code protection purposes, the user may want to erase all the important code or data in the flash memory before re-entering UART BSL Mode. Therefore, this code must be located at a different Flash sector so that other important code or data can be erased, while this code remains. This code `PROGRAM_BMI()` can be located at the smallest Flash sector.

Note: The user should take care of the WatchDog service routine and to disable the WDT before calling this user routine, as it does not return back to the user code. A soft reset will be triggered by this user routine and the device will boot up based on the programmed BMI value.

```
unsigned char xdata PASSWORD_BMI _at_ 0xF0A0; //define a password for programming BMI
unsigned char idata USER_ID[4] _at_ 0x30;

void PROGRAM_BMI(void)
{
 bit bBMIStat;
 if( PASSWORD_BMI == 0xAA )
 {
 // Erase important code or data if needed before programming BMI value
 // ......
 // Prepare USER_ID (for OCDS Mode SPD_0)
 USER_ID[0]          =  0x00; // USER_ID_3
 USER_ID[1]          =  0x00; // USER_ID_2
 USER_ID[2]          =  0x9F; // USER_ID_1 (!BMI)
 USER_ID[3]          =  0x60; // USER_ID_0 (BMI)
 // Program BMI value to OCDS Mode (SPD_0)
 SP = 0x50;           // define stack pointer
 bBMIStat = BR_FEATURE_SETTING(0x01, 0x30);// Option = 0x01. USER_ID add = 0x30
  //   Check the status for BMI programming
  // If BMI programming is successful, no return data. Automatically soft-reset
  if (bBMIStat != 0 )
  {
      //   Failure in correct option selection or NMISR != 0x00
```

```
  }}}

void INT_vInit(void)
{
 NMICON = 0x00; //NMI Control Register
 IEN0 |= 0x01; // interrupt enable register 0
 IEN1 |= 0x00; // interrupt enable register 1
 EXICON0 = 0xF3; // External interrupt Control Register 0
 SFR_PAGE(_su3, noSST); // switch to page 3 without saving
 MODPISEL1 |= 0x18; // peripheral input select register 1
 SFR_PAGE(_su0, noSST); // switch to page 0 without saving
}// End of function INT_vInit

void INT_viExt0(void) interrupt EXT0INT
{
 // emulate Reset pin an falling edge
 SFR_PAGE(_su1, noSST);
 PASSWORD_BMI = 0xAA;        // User-defined password
 while ( !(P2_DATAIN & 0x01) );
 PROGRAM_BMI();
...
} // End of function INT_viExt0
```

5 Summary

The Infineon XC82x/XC83x microcontrollers do not use any pins to enter different boot modes. The BMI configuration boot-up methodology is extremely useful in situations where port pins are very limited. This application note demonstrates how the user can program and update the BMI value under various operating conditions, with ease.

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

Published by Infineon Technologies AG