Application Note

An OCD ISP for the Z8 Encore! Family Using the Z8F6482 MCU

AN037301-0215

Abstract

This application note describes a boot loader program that uses a Z8F6482 microcontroller to control an On-Chip Debugger (OCD) In-System Programmer (ISP) for debugging and programming the entire family of Z8 Encore! microcontrollers, which includes the Z8 Encore!, Z8 Encore! XP, and Z8 Encore! MC product lines. The three primary areas of the C language-based code developed for this application are:

- A file loader that receives a hexadecimal file in the Intel Hex format via a terminal user interface and saves it to a host MCU's internal or external Flash memory
- An Application Programming Interface (API) between the OCD and a target device for all programming commands
- Sample code for programming a target device

Note: The source code file associated with this application note, AN0373-SC01, is available free for download from the Zilog website. This source code is tested with ZDS II – Z8 Encore! version 5.2.1, Windows XP SP3, Windows 7, and the F6482 Series Development Board. Zilog Developer Studio II (ZDS II) for Z8 Encore! offers debug and development support for Zilog's Z8 Encore! XP and Z8FMC16 product families. Subsequent releases of ZDS II may require you to modify the code supplied with this application note.

Features

The key features of this application include:

- Z8F6482 internal Flash memory that stores up to 8 KB of firmware
- External Flash memory with four program slots; each program slot can store up to 64 KB of firmware
- A firmware hexadecimal file from the terminal user interface that can be directly programmed to the target device
- The following OCD API commands allow you to:
  - Read the OCD revision
  - Read and write to/from the OCD control registers
  - Read and write to/from the registers
  - Read and write to/from program memory
  - Read status

- Send debug commands
- Set the target Flash frequency
- Start Debug Mode
- Lock and unlock the target Flash memory
- Mass erase, program, and verify to/from the target Flash memory

Discussion

The application program runs in two modes: PC Mode and Standalone Mode. In PC Mode, firmware from the PC can either be saved to the host MCU's internal or external Flash memory before being programmed to the target device, or it can be programmed directly to the target device. In Standalone Mode, firmware saved to Program Slot 1 in external Flash memory can be programmed to the target device with the press of a push-button.

Each microcontroller in the Z8 Encore!, Z8 Encore! XP, and Z8 Encore! MC product lines contains an integrated OCD used for project debugging. Its features include:

- Reading and writing of the register file, program memory, and data memory
- Setting of breakpoints
- Executing eZ8 CPU instructions

Four major components are used in the development of this application, as shown in Figure 1.

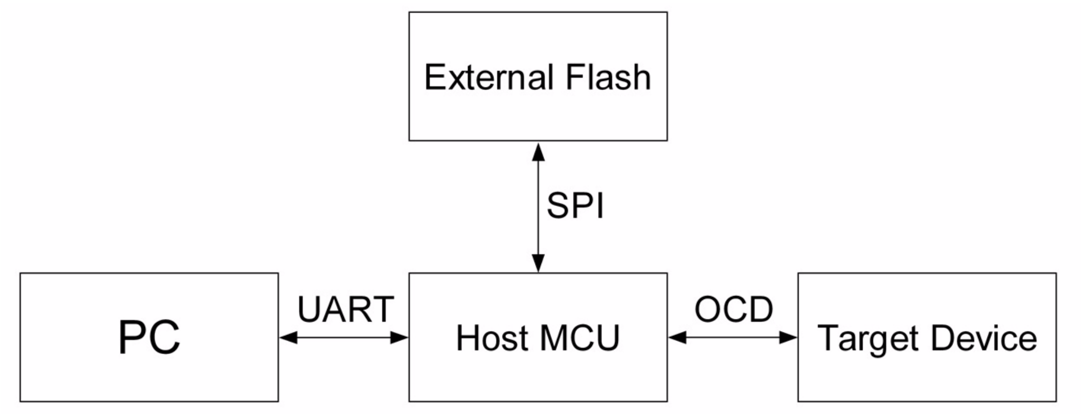


Figure 1. OCD Hardware Setup Block Diagram

The host MCU is the Z8F6482 microcontroller on the F6482 Series Development Board. This application utilizes external Flash memory in the Z8F6482 MCU for storing firmware. In this application, the main functions of this MCU include:

- Providing a user interface

- Obtaining a firmware file from the PC in Intel Hex format
- Saving the firmware to external or internal Flash memory
- Sending firmware from the PC, internal Flash memory, or external Flash memory to the target device

On the PC, users can select from the following list of operations:

- Save firmware to internal Flash memory
- Save firmware to external Flash memory
- Send firmware from the PC to the target device
- Send firmware from internal Flash memory to the target device
- Send firmware from external Flash memory to the target device

Table 1 lists the serial communication interfaces used between components. The target device is a microcontroller from the Z8 Encore!, Z8 Encore! XP, or Z8 Encore! MC Series of MCUs that will be programmed by the host MCU.

Table 1. Serial Communication Interfaces

Component	Serial Communication Interface
PC – Host MCU	UART
Host MCU – External Flash Memory	SPI (Serial Peripheral Interface)
Host MCU – Target Device	UART – OCD

Communicating with the Target Device

The host MCU communicates with the target device by sending OCD commands through the debug interface circuit. The following OCD commands are used in this application:

Read OCD Revision (0x00) . Determines the OCD version that identifies which commands are added, removed, or changed.

Read OCD Status Register (0x02) . Reads the OCD Status Register (OCDSTAT) of the target device.

Write OCD Control Register (0x04) . Writes data to the OCD Control Register (OCDCTL) of the target device.

Read OCD Control Register (0x05). Reads the OCDCTL of the target device.

Write Register (0x08). Writes data to the specified register file address.

Read Register (**0x09).** Reads data from the specified register file address.

Write Program Memory (0x0A) . Writes data to the specified program memory address.

Read Program Memory (0x0B). Reads data from the specified program memory address

Table 2 lists the OCD commands in sequential order and the data flow between the host MCU and the target device.

Table 2. Data Flows Between the Host MCU and Target Device

OCD Command	Data Flow	
	From Host to Target Device	**From Target Device to Host**
Read OCD Revision	0x00	
		Revision Number [high byte]
		Revision Number [low byte]
Read OCD Status	0x02	
		OCDSTAT [1 byte]
Write OCD Control Register	0x04	
	OCDCTL [1 byte]	
Read OCD Control Register	0x05	
		OCDCTL [1 byte]
Write Register	0x08	
	Register Address [high byte]	
	Register Address [low byte]	
	Data Size [1 byte]	
	Data [up to 256 bytes]	
Read Register	0x09	
	Register Address [high byte]	
	Register Address [low byte]	
	Data Size [1 byte]	
		Data [up to 256 bytes]
Write Program Memory	0x0A	
	Program Memory Address [high byte]	
	Program Memory Address [low byte]	
	Program Size [high byte]	
	Program Size [low byte]	
	Data [up to 65,536 bytes]	

Table 2. Data Flows Between the Host MCU and Target Device (Continued)

OCD Command	Data Flow	
	From Host to Target Device	From Target Device to Host
Read Program Memory	0x0B	
	Program Memory Address [high byte]	
	Program Memory Address [low byte]	
	Program Size [high byte]	
	Program Size [low byte]	
		Data [up to 65,536 bytes]

In addition to the command itself, the Write Register and Read Register OCD commands send the register address and data size to the target device. The register address consists of two bytes – high byte and low byte. The upper nibble of the register address high byte is always `0000b`. The data size is the number of bytes that can be written to, or read from a register. The data size value range is 1–256. This value in Hex is `0x01`–`0xFF` for 1–255, and `0x00` for 256.

For example, to read 1 byte of data from Port A Input Data Register (`0xFD2`) of the target Z8F082A MCU, the data to be sent by the host MCU through OCD are `0x09`, `0x0F`, `0xD2` and `0x01`. The Read Register OCD command is `0x09`; the register address high byte is `0x0F` and low byte is `0xD2`; and the data size is `0x01`.

For Write Program Memory, Read Program Memory, Write Data Memory, and Read Data Memory OCD commands, the program memory and data memory addresses are two bytes – high byte and low byte. The data size value range is 1–65,536. This value in Hex is `0x0001`–`0xFFFF` for 1–65,535, and `0x0000` for 65,536.

Writing Firmware to the Target Device

The purpose of this application is to program the firmware to a target device through the OCD, achieved by executing the four actions described in this section.

Setting the Target Device to Debug Mode

To establish communication with the host MCU and execute the operations requested by the host MCU, the target device must operate in Debug Mode.

To set the target device to Debug Mode, the host MCU must perform the following steps:

1. Pull down the RESET pin of the target device.
2. Send a break to the target device by configuring the Send Break setting of U1CTL0 bit [2] of the UART-LDD Control Register.
3. Release the RESET pin of the target device.

4. Stop sending breaks to the target device by clearing the Send Break setting of U1CTL0 bit [2].
5. Send auto-baud character `0x80` to the target device.
6. Set the target device's Flash frequency value by issuing the following data:
   – A write register command (OCD command `0x08`)
   – Register file address `0xFFA` (corresponding to the Flash Programming Frequency Register)
   – A Flash frequency value (two bytes)

To set an 8-pin target device to Debug Mode, the host MCU must perform the following steps:

1. Pull down the RESET pin of the target device.
2. Send the following bytes serially to the 8-pin target device:
   – `0x80` (auto-baud character)
   – `0xEB`
   – `0x5A`
   – `0x70`
   – `0xCD`
3. Release the RESET pin of the target device.
4. Resend auto-baud character `0x80` to the target device.
5. Set the target device to Debug Mode and enable breakpoints by issuing a Write OCD Control Register command (OCD command `0x04`) and an OCD Control Register value of `0xC0`.
6. Set the target device's Flash frequency value by issuing the following data:
   – A write register command (OCD command `0x08`)
   – Register file address `0xFFA` (corresponding to the Flash Programming Frequency Register)
   – A Flash frequency value (two bytes)

If the target device is a Z8F6482 microcontroller, then instead of setting the Flash frequency value, enable the Digitally Controlled Oscillator (DCO) and Frequency Locked Loop (FLL) on the Clock Control 5 register file address (i.e., `0xF87`), then set the Flash programming configuration (`0xFFA`) to Byte Programming Mode (`0x00`).

Note: *Flash frequency value* (two bytes) is defined as the system clock frequency of the target device divided by 1000.

Performing a Mass Erase Operation on the Target Device's Flash Memory

A Mass Erase operation is performed prior to writing firmware to the target device Flash memory and it is only enabled through OCD when the target device is in Debug Mode. A mass erase action is executed by issuing a series of write register commands (OCD command `0x08`) to the target device.

To perform a mass erase operation on the target device, the host MCU must perform the following steps:

1. Unlock the target device's Flash memory by writing the unlocking code sequence, `0x73` and `0x8C` respectively, to register file address `0xFF8` in the Flash Control (FCTL) Register of the target device.
2. Initiate a mass erase action by writing mass erase command `0x63` to register file address `0xFF8` in the FCTL of the target device. This action will set all of the values of the target device's Flash memory to `0xFF`.
3. Wait until the mass erase process is completed by continuously reading from register file address `0xFF8` in the Flash Status (FSTAT) Register of the target device. A Flash controller locked status (`0x00`) is returned when the operation is complete.

Programming the Target Device's Flash Memory

To program the target device's Flash memory, send the firmware to the device's Flash memory.

The host MCU must perform the following steps to program the target device with firmware:

1. Set the target device to Debug Mode.
2. Perform a mass erase operation on the target device's Flash memory.
3. Unlock the target device's Flash memory by writing the unlocking code sequence, `0x73` and `0x8C` respectively, to register file address `0xFF8` in the FCTL of the target device.
4. Read the Flash Status Register address (`0xFF8`) of the target device. If Flash memory is unlocked, address `0x03` is sent to the host MCU.
5. Send the following data to the target device:
   - Write program memory command (OCD command `0x0A`)
   - The starting address in Flash memory to which the firmware will be programmed (two bytes)
   - The size of the firmware (i.e., two bytes)
   - Firmware
6. Lock the target device's Flash memory by writing `0x00` to Register File `0xFF8`.

Verifying the Target Device’s Flash Memory

After programming the target device’s Flash memory, firmware from the host MCU’s Flash memory and the target device’s Flash memory is read and compared. Identical data in both locations indicates successful programming of the target device’s Flash memory.

To verify the target device’s Flash memory, the host MCU must perform the following steps:

1. Send a read program memory command (OCD command `0x0B`) to the target device.
2. Send the starting address in Flash memory to be read (two bytes).
3. Send the size of the firmware to be read (two bytes).
4. Compare firmware received from the target device to firmware in the host MCU’s Flash memory.

Hex File Format

The hexadecimal file generated by ZDS II is an Intel Hex file. This file contains the firmware or program that will be sent by the host MCU to the target device.

Intel Hex is a file format with ASCII text lines that contain hexadecimal characters. Each text line or record consists of six fields, as shown in Figure 2.

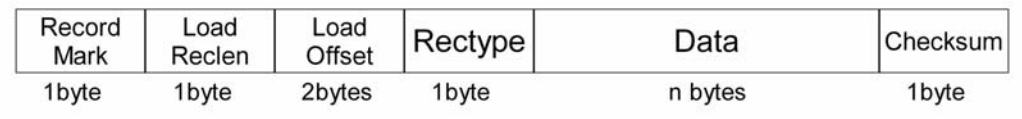


Figure 2. Intel Hex File Format

These six fields of an Intel Hex file are described in this section.

Record Mark or Start Code. This field is an ASCII colon (:) or `0x3A`.

Load Reclen or Byte Count. Indicates the number of bytes of information in the data field. The maximum value for Load Reclen is `0xFF` (255).

Load Offset or Address. Specifies the 16-bit starting address of the data bytes. The physical address of the data can be determined by adding load offset to a previously-established base address.

Rectype or Record Type. Defines the data field, as listed in Table 3.

Table 3. Record Types

Rectype	Description
00	Data Record
01	End of File Record
02	Extended Segment Address Record
03	Start Segment Address Record

Table 3. Record Types

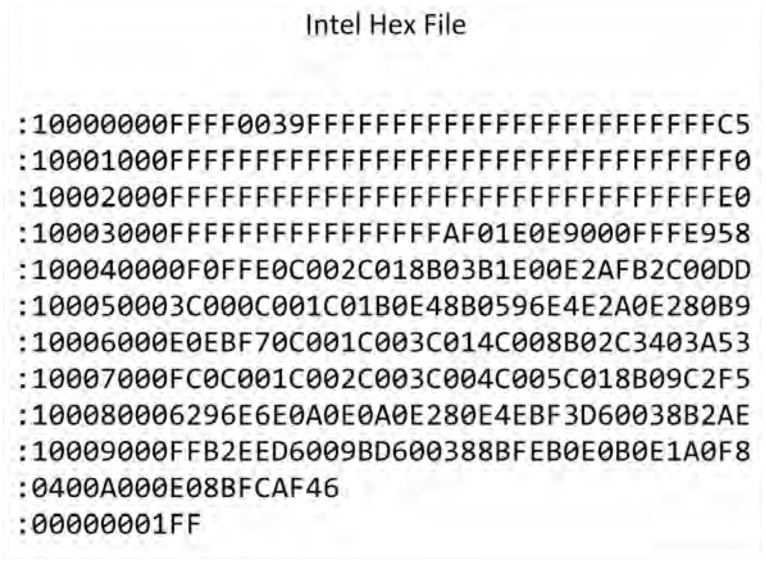
Rectype	Description
04	Extend Linear Address Record
05	Start Linear Address Record

Data or Info. Contains *n* bytes of data.

Checksum. This field is calculated as the two's complement of the sum of the entire text line.

Note: One byte is equal to two ASCII characters.

Figure 3 shows an example of an Intel file.


Intel Hex File

```
:10000000FFFF0039FFFFFFFFFFFFFFFFFFFFFFFFC5
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFAF01E0E9000FFFE958
:100040000F0FFE0C002C018B03B1E00E2AFB2C00DD
:100050003C000C001C01B0E48B0596E4E2A0E280B9
:10006000E0EBF70C001C003C014C008B02C3403A53
:10007000FC0C001C002C003C004C005C018B09C2F5
:100080006296E6E0A0E0A0E280E4EBF3D60038B2AE
:10009000FFB2EED6009BD600388BFEB0E0B0E1A0F8
:0400A000E08BFCAF46
:00000001FF
```

Figure 3. Example of an Intel Hex File

Table 4 lists the six fields of each record from the Intel Hex file in Figure 3.

Table 4. Contents of a Sample Intel Hex File

	Fields					
Record Number	Record Mark	Load Reclen	Load Offset	Rectype	Data	Checksum
1	:	10	0000	00	FFFF0039FFFFFFFFFFFFFFFFFFFFFFFF	C5
2	:	10	0010	00	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	F0

Table 4. Contents of a Sample Intel Hex File

	Fields					
Record Number	**Record Mark**	**Load Reclen**	**Load Offset**	**Rectype**	**Data**	**Checksum**
3	:	10	0020	00	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	E0
4	:	10	0030	00	FFFFFFFFFFFFFFFFAF01E0E9000FFFE9	58
5	:	10	0040	00	0F0FFE0C002C018B03B1E00E2AFB2C00	DD
6	:	10	0050	00	3C000C001C01B0E48B0596E4E2A0E280	B9
7	:	10	0060	00	E0EBF70C001C003C014C008B02C3403A	53
8	:	10	0070	00	FC0C001C002C003C004C005C018B09C2	F5
9	:	10	0080	00	6296E6E0A0E0A0E280E4EBF3D60038B2	AE
10	:	10	0090	00	FFB2EED6009BD600388BFEB0E0B0E1A0	F8
11	:	04	00A0	00	E08BFCAF	46
12	:	00	0000	01		FF

As a further example, consider record number 5, in which the Load Reclen count is 10. As a result, the data field of record number 5 includes the following 16 bytes: `0F`, `0F`, `FE`, `0C`, `00`, `2C`, `01`, `8B`, `03`, `B1`, `E0`, `0E`, `2A`, `FB`, `2C`, and `00`. Because the Load Offset value is `0040`, these 16 bytes of data will be stored in `0040`–`004F` address range. The Rectype value is `00`, which indicates that the data field contains a data record, as shown in Table 3. The Checksum value is the two's complement of the sum of the Load Reclen, Load Offset, Rectype, and Data fields. The sum of these fields is 23, which is calculated in Equation 1.

Equation 1

```
10 + 00 + 40 + 00 + 0F + 0F + FE + 0C + 00 + 2C + 01 + 8B + 03 + B1 + E0 +
0E + 2A + FB + 2C + 00 = 23
```

The Checksum value is DD, which is the two's complement of 23.

The final record occurs only once in every hexadecimal file. In this record, Load Reclen is `0`, which indicates an empty data field, and Load Offset is typically `0000`.

Hardware Setup

Figure 4 shows the schematic diagram used in this application. The debug interface circuit consisting of a Schottky diode and a pull-up 10KΩ resistor is located between a host MCU and a target device. The host MCU and the target device are powered by the same power supply. The power from the host MCU board is shared to the target device through an OCD debug cable.

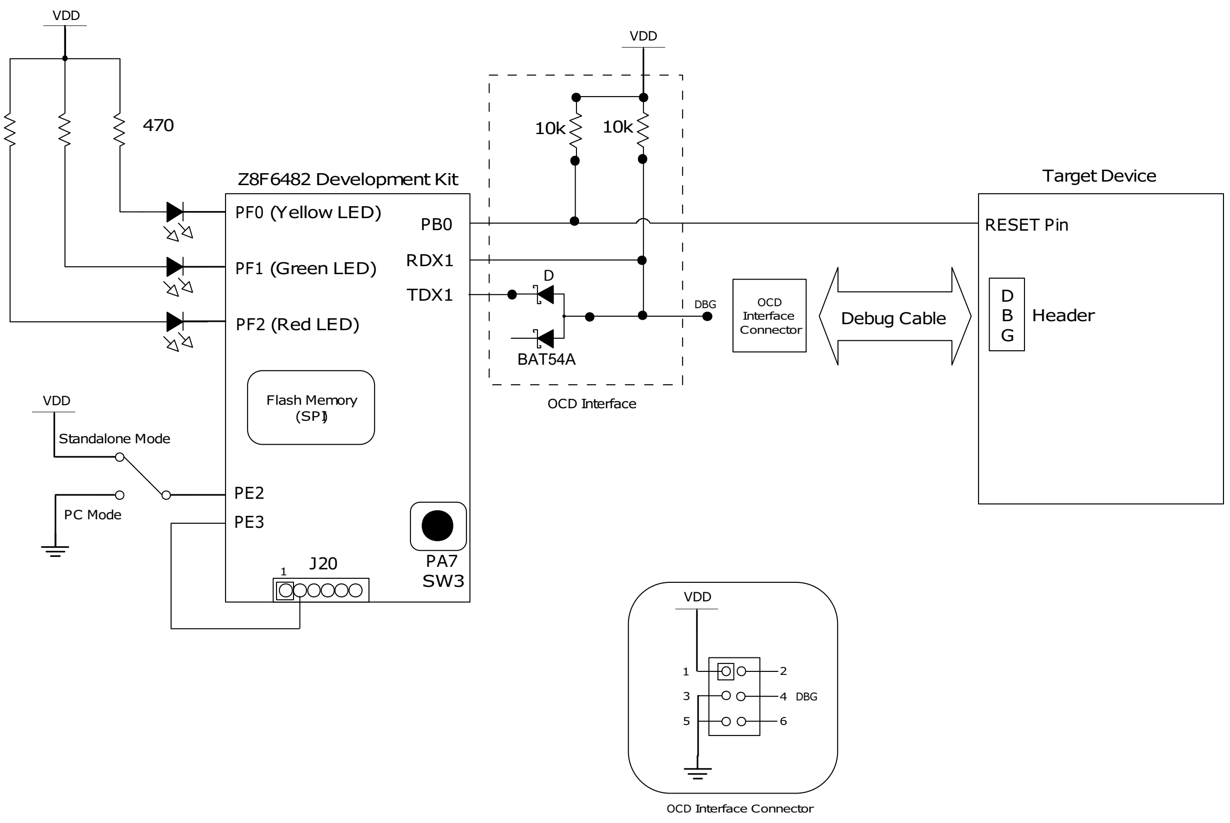


Figure 4. Schematic Diagram

In Figure 4, the target device is a member of the Z8 Encore!, Z8 Encore! XP, or Z8 Encore! MC product lines.

The F6482 Series Development Board includes 32 Mbit external SPI Flash memory (S25FL032P) which is used to store between 4–64 KB of firmware and firmware information. This firmware information is used by the application to identify the number of pages occupied by each firmware file and to verify if external Flash memory contains firmware, as described in the next section, Firmware Design.

Port B [0] is used by the host MCU to control the reset line of the target MCU. This port's connection is not included in the debug header because reset connections are not available on all Zilog development boards.

Port E [2] includes a single-pole double-throw switch used as a mode selector switch to select either Standalone Mode or PC Mode. Port E[3] is connected to pin 2 of J20. It is used as the RTS flow control signal.

Switch 3 (SW3) is a push-button switch connected to Port A[7]. This switch is included on the F6482 Series Development Board and is used to program the target device when the application is in Standalone Mode.

Three LEDs that are used as indicators are connected to Port F[2:0]. The red and green LEDs are indicators for PC Mode and Standalone Mode operations while the yellow LED is an indicator for Standalone Mode only. Table 5 lists the LED status description.

Table 5. LED Status Description

LED Color	Status	Description
Yellow	ON	External memory Program Slot 1 contains firmware
	Blinking	External memory Program Slot 1 does not contain firmware
Red	Blinking	Saving the firmware to Flash memory and programming target device
	ON	An error occurred
Green	ON	Target device successfully programmed

Firmware Design

The firmware developed for this application features two main modes of operation, PC Mode and Standalone Mode. Each of these modes is described in this section.

PC Mode

The following two primary operations can be performed in PC Mode:

- Save firmware to Flash memory
- Send firmware to the target device

Saving Firmware to Flash Memory

To save firmware to Flash memory, the `PCMODE_PCHexFiletoINTFlash()` and `PCMODE_PCHexFiletoEXTFlash()` functions are used. Both of these functions perform a call to the `ui8FLASHLOADER_StartReadHEXFile()` routine that extracts firmware from the hexadecimal file before saving it to Flash memory or sending it to the target device.

The `PCMODE_PCHexFiletoINTFlash()` function is used to save the firmware from the PC to an allocated 8KB of Flash memory in the host MCU, with a starting address of `0xA000`.

The `PCMODE_PCHexFiletoEXTFlash()` function is used to save the firmware from the PC to external Flash memory. Four program slots are available in external Flash memory, as shown in Figure 5. The capacity of each program slot is 64KB.

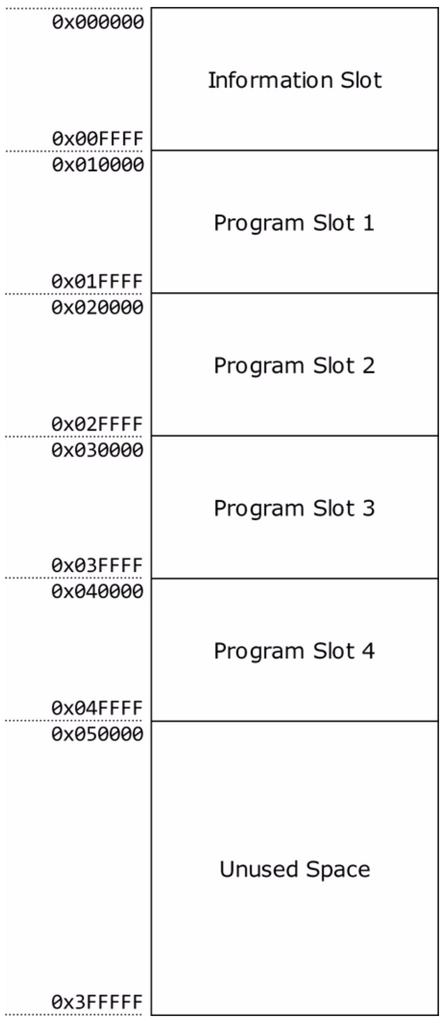
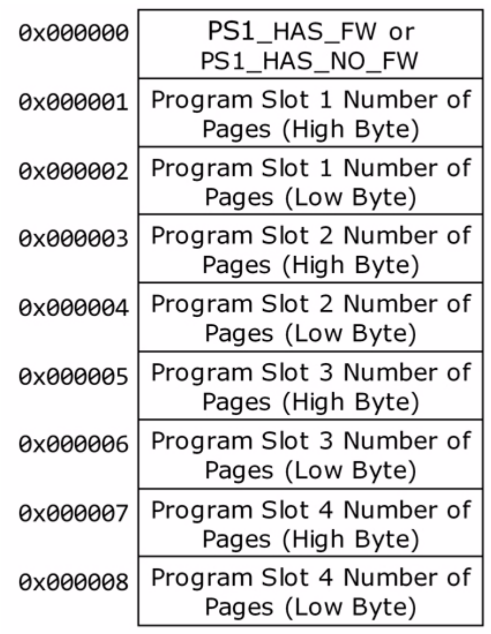


Figure 5. External Flash Memory Map

An information slot is also allocated in external Flash memory which provides information for the Standalone Mode to determine if Program Slot 1 contains firmware. If Program Slot 1 includes firmware, `0xAA` is stored to external Flash memory address `0x000000`. If Program Slot 1 does not contain firmware, `0x00` is stored.

The information slot includes information about the number of pages occupied by the firmware in a program slot, as shown in Figure 6.



0x000000	PS1_HAS_FW or PS1_HAS_NO_FW
0x000001	Program Slot 1 Number of Pages (High Byte)
0x000002	Program Slot 1 Number of Pages (Low Byte)
0x000003	Program Slot 2 Number of Pages (High Byte)
0x000004	Program Slot 2 Number of Pages (Low Byte)
0x000005	Program Slot 3 Number of Pages (High Byte)
0x000006	Program Slot 3 Number of Pages (Low Byte)
0x000007	Program Slot 4 Number of Pages (High Byte)
0x000008	Program Slot 4 Number of Pages (Low Byte)

Figure 6. External Flash Information Slot Memory Map

Sending Firmware to the Target Device

The `PCMODE_INTFlashtoTargetDev()`, `PCMODE_EXTFlashtoTargetDev()`, and `PCMODE_PCHexFiletoTargetDev()` functions are used to program the firmware to the target device.

The `PCMODE_INTFlashtoTargetDev()` function sends firmware from the host MCU's internal Flash memory, starting at address `0xA000`, to the target device.

The `PCMODE_EXTFlashtoTargetDev()` function sends firmware from the host MCU's external Flash memory to the target device. Users have the option to select a program slot number from which the firmware is sent to the target device. Four program slots are available.

The `PCMODE_PCHexFiletoTargetDev()` function sends firmware from the PC to the target device. However, before sending the firmware to the target device, the firmware is written to an 8KB temporary storage location in the host MCU's internal Flash memory, starting at address `0x8000`.

Up to 8KB of firmware files can be stored in the host MCU's internal Flash memory. The storage size for each program slot in external Flash memory is 64KB. Internal Flash memory allocated for the target device's firmware and program slot sizes, which can be adjusted, depends on the host MCU and target device's Flash memory sizes. Because two 8KB Flash memory sectors are used for firmware storage and application code in the host MCU, Zilog recommends a Z8F6482 microcontroller with at least 32KB of Flash memory as the host MCU.

Standalone Mode

In Standalone Mode, when Program Slot 1 of external Flash memory contains firmware, the Yellow LED turns ON, indicating that the user can continue using this mode. If Program Slot 1 does not contain firmware, the yellow LED blinks.

If Program Slot 1 contains firmware, briefly pressing the push-button switch connected to PA7 sends the firmware to the target device. The function used for this operation is `ui8OCD_API_SendFW_EXTToTarget()`.

The green LED turns ON to indicate that the firmware is successfully programmed to the target device. If an error occurs, the red LED turns ON.

Tools

The following tools were used to develop this application:

- A development kit from the Z8 Encore! Family:
  - Z8 Encore! XP F1680 Series (28-Pin) Development Kit
  - Z8 Encore! XP F0822 Series (28-Pin) Development Kit
  - Z8 Encore! XP F042A Series (8-Pin) Development Kit
  - Z8 Encore! XP F082A Series (28-Pin) Development Kit
  - Z8 Encore! XP F64xx Series Development Kit
  - Z8 Encore! XP F6482 Series Development Kit
  - Z8FMC16100 Series Motor Control Development Kit
  - Z8 Encore! F083A Series (28-Pin) Development Kit
  - ZMOTION Detection and Control (8-Pin and 20-Pin) Development Kits
- Serial terminal emulation program (for example, RealTerm or HyperTerminal)
- Serial cable (RS232)
- USB SmartCable

Note: If you are developing your application with the Z8F0830 MCU, use the Z8 Encore! F083A Series (28-Pin) Development Kit.

Testing the Application

Observe the following steps to install and set up the Z8F6482 Development Kit and the source code for this application. For more in-depth information on how to set-up the Z8F6482 Development Kit (for example, jumper settings), refer to Z8 Encore XP F6482 Series Development Kit User Manual (UM0263).

1. Download and install the ZDSII – Z8 Encore! version 5.2.1 software and documentation. These files are available free for download from the Zilog Store.
2. Download AN0373-SC01.zip from the Zilog website and unzip it to the following directory on your PC; this path is created during the installation process in Step 1.

   ```
   < ZDSII_Z8Encore!_5.2.1 installation folder>\samples
   ```

3. Connect the USB SmartCable to the PC and to the F6482 Series Development Board.
4. Connect the target device to the F6482 Series Development Board; refer to the schematic diagram in Figure 4.
5. Apply power to the F6482 Series Development Board.
6. Launch ZDSII – Z8 Encore! version 5.2.1 and open the project named `AN0373_OCD_Programmer.zdsproj`, which is located in the following directory:

   ```
   <ZDSII_Z8Encore!_5.2.1 installation folder>\samples\AN0373-SC01
   ```

7. Open the `ocd_api.h` file, and make the following adjustments to the code, depending on your target device:
   - If the target device belongs to the F6482 Series of MCUs, uncomment `#define TARGET_F6482`.
   - If the target device is an 8-Pin Z8 Encore! XP MCU, uncomment `#define TARGET_8_PIN`.
8. If the target device does NOT belong to the F6482 Series of MCUs, provide the target device Flash frequency values (i.e., `TARGET_FFREQ_H` and `TARGET_FFREQ_L`).

   Table 6 lists the Flash frequency values for multiple target devices.

Table 6. Target Device Flash Frequency Values

Target Device (MCU Family)	FFREQ_H	FFREQ_L
Z8F082A and Z8F0830	0x15	0x99
Z8F1680 and Z8FMC16100	0x2B	0x33
Z8F64xx and Z8F0822	0x48	0x00
Z8F083A	0x4E	0x20

9. In ZDSII, click the **Save all**, **Rebuild all,** and **Download Code** menu options to enable the F6482 Series Development Board for PC Mode or Standalone Mode operations.

Selecting PC Mode or Standalone Mode

Select the mode of operation by executing the following steps.

1. For Standalone Mode operation, connect Port E[2] On the F6482 Series Development Board to V_{DD}. For PC Mode operation, connect Port E[2] to GND.

2. Reset the Development Board.

Running the Application in PC Mode

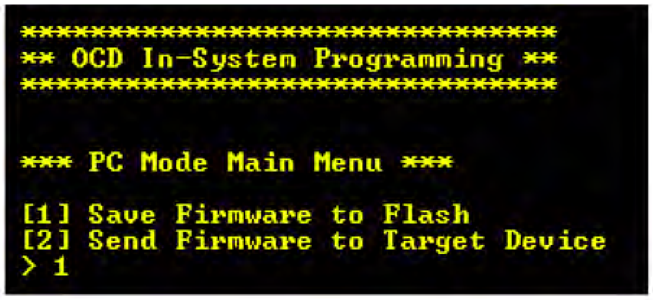
Observe the following steps to set up the serial terminal emulation program:

1. Connect the serial cable to the PC's COM port and to the F6482 Series Development Board.
2. On the PC, launch the serial terminal program.
3. Configure the following settings on the serial terminal program:
   - Baud: 57600
   - Port 1 (this setting depends on the COM port of the PC)
   - Parity: none
   - Data bits: 8
   - Stop bits: 1
   - Flow Control: Hardware – RTS/CTS

Saving Firmware from the Hex File to the Host MCU's Flash Memory

Observe the following steps to save firmware from the Hex file to Flash memory.

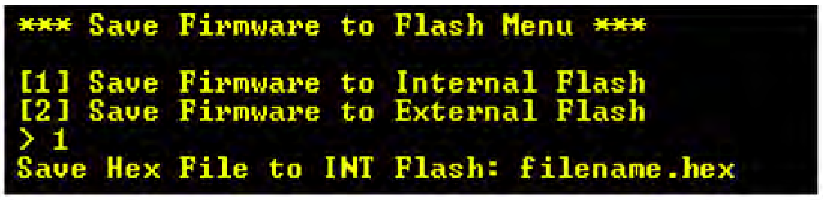
1. From the **PC Mode Main Menu** display, enter **1** to save the firmware to Flash memory, as indicated in Figure 7.



```
********************************
** OCD In-System Programming **
********************************


*** PC Mode Main Menu ***

[1] Save Firmware to Flash
[2] Send Firmware to Target Device
> 1
```

Figure 7. PC Mode Main Menu Selection

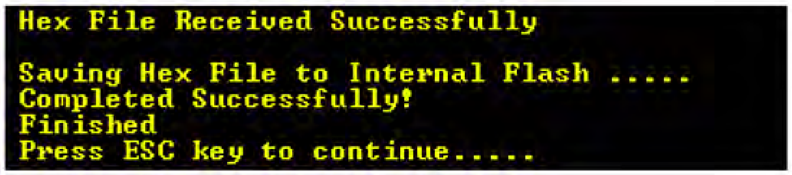
2. From the **Save Firmware to Flash Menu** display, enter **1** to save the firmware to internal Flash memory, as indicated in Figure 8.



```
*** Save Firmware to Flash Menu ***

[1] Save Firmware to Internal Flash
[2] Save Firmware to External Flash
> 1
Save Hex File to INT Flash: filename.hex
```

Figure 8. Save Firmware to Internal Flash Menu Selection

3. In the serial terminal emulation program, browse and select the Hex file. Send it as text file to the MCU. Figure 9 shows that the firmware is successfully stored in the host MCU's internal Flash memory.



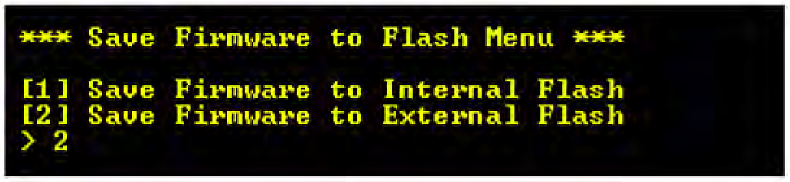
```
Hex File Received Successfully

Saving Hex File to Internal Flash .....
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 9. Successfully Saved Firmware to Internal Flash

Saving Firmware from the Hex File to External Flash Memory

Observe the following steps to save firmware from the Hex file to external Flash memory.

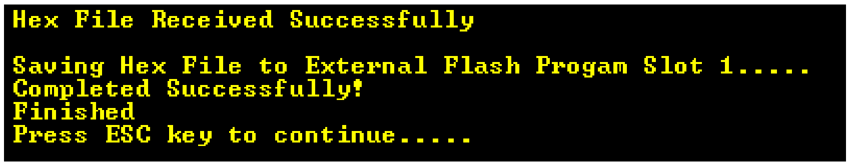
1. From the **PC Mode Main Menu** display, enter **1** to save the firmware to Flash memory, as indicated in Figure 7.
2. From the **Save Firmware to Flash Menu** display, enter **2** to save the firmware to external Flash memory, as indicated in Figure 10.



```
*** Save Firmware to Flash Menu ***

[1] Save Firmware to Internal Flash
[2] Save Firmware to External Flash
> 2
```

Figure 10. Save Firmware to Flash Menu with Item 2 Selected

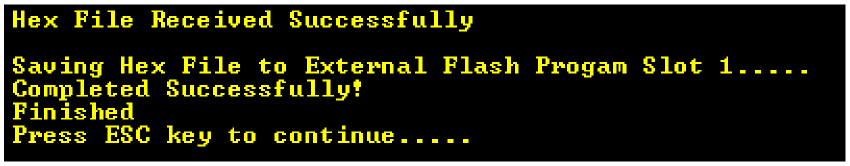
3. Select a program slot option from the four options that are displayed. In the Figure 11 example, Program Slot 1 is selected.



```
Hex File Received Successfully

Saving Hex File to External Flash Progam Slot 1.....
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 11. Program Slot Menu Selection

4. In the serial terminal emulation program, browse and select the Hex file. Send it as a text file to external Flash memory. Figure 12 shows that the firmware is successfully saved to Program Slot 1.



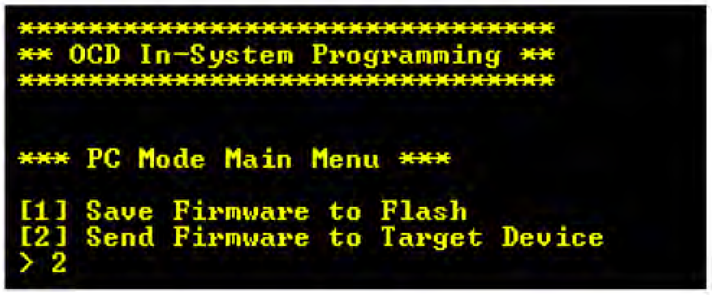
```
Hex File Received Successfully

Saving Hex File to External Flash Progam Slot 1.....
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 12. Firmware Successfully Saved to External Flash Program Slot 1

Sending Firmware from the Host MCU's Flash Memory to the Target Device's Flash Memory

Prior to sending firmware from the host to the target device, ensure that the firmware is saved in the host MCU's Flash memory. If required, observe the procedure in the Saving Firmware from the Hex File to the Host MCU's Flash Memory section on page 17, then continue with the following steps.

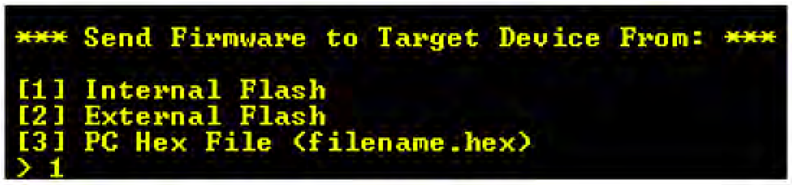
1. From the **PC Mode Main Menu** display, enter **2** to send the firmware to the target device, as indicated in Figure 13.



```
********************************
** OCD In-System Programming **
********************************


*** PC Mode Main Menu ***

[1] Save Firmware to Flash
[2] Send Firmware to Target Device
> 2
```

Figure 13. PC Mode Main Menu with Item 2 Selected

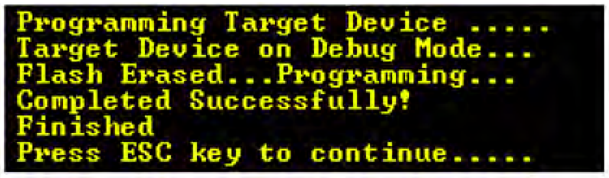
2. From the **Send Firmware to Target Device From:** menu, enter **1** to send the firmware from internal Flash memory, as indicated in Figure 14.



```
*** Send Firmware to Target Device From: ***

[1] Internal Flash
[2] External Flash
[3] PC Hex File (filename.hex)
> 1
```

Figure 14. Send Firmware to Target Device from Internal Flash

3. Figure 15 indicates that the firmware is successfully programmed to the target device.



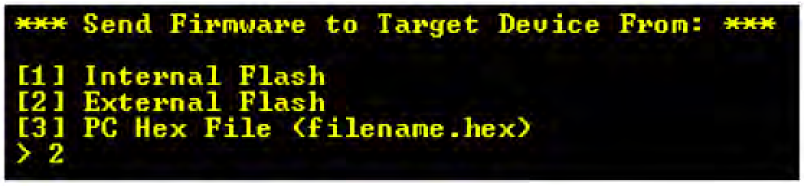
```
Programming Target Device .....
Target Device on Debug Mode...
Flash Erased...Programming...
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 15. Successfully Programmed the Target Device with Firmware from Internal Flash

Sending Firmware from External Flash Memory to the Target Device's Flash Memory

Prior to sending firmware from external Flash memory to the target device, ensure that the firmware for the target device is installed in external Flash memory. If required, follow the steps in the Saving Firmware from the Hex File to External Flash Memory section on page 18. Observe the following procedure.

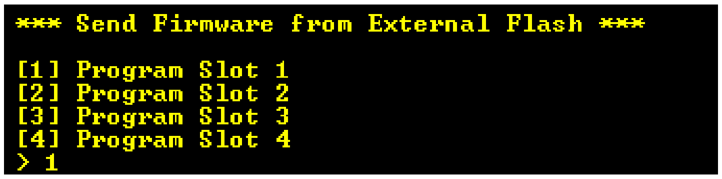
1. From the **PC Mode Main Menu** display, enter **2** to send the firmware to the target device, as indicated in Figure 7 on page 17.
2. From the **Send Firmware to Target Device From:** menu, enter **2** to send the firmware to the target device from external Flash memory, as indicated in Figure 16.



```
*** Send Firmware to Target Device From: ***

[1] Internal Flash
[2] External Flash
[3] PC Hex File (filename.hex)
> 2
```

Figure 16. Send Firmware to Target Device from External Flash

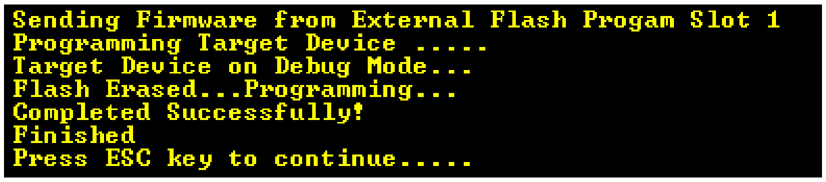
3. Select a program slot from the **Send Firmware from External Flash** menu, ~~as~~ shown in Figure 17.



```
*** Send Firmware from External Flash ***

[1] Program Slot 1
[2] Program Slot 2
[3] Program Slot 3
[4] Program Slot 4
> 1
```

Figure 17. External Flash Program Slots Display

4. Figure 18 indicates that the firmware is successfully programmed into the target device. In this example, the firmware is selected from Program Slot 1 of external Flash memory.



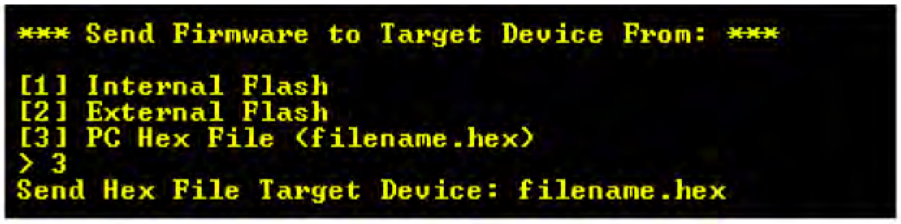
```
Sending Firmware from External Flash Progam Slot 1
Programming Target Device ......
Target Device on Debug Mode...
Flash Erased...Programming...
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 18. Programming Target Device with Firmware from External Flash Successful

Sending Firmware from the PC to the Target Device's Flash Memory

Observe the following steps to send firmware from the PC to the target device.

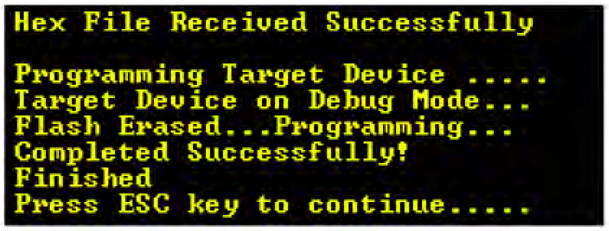
1. From the **PC Mode Main Menu** display, enter **2** to send the firmware to the target device, as indicated in Figure 7 on page 17.
2. From the **Send Firmware to Target Device From:** menu, enter **3** to send the firmware to the target device from a hex file on the PC, as indicated in Figure 19.



```
*** Send Firmware to Target Device From: ***

[1] Internal Flash
[2] External Flash
[3] PC Hex File (filename.hex)
> 3
Send Hex File Target Device: filename.hex
```

Figure 19. Send Firmware to Target Device from PC Hex File

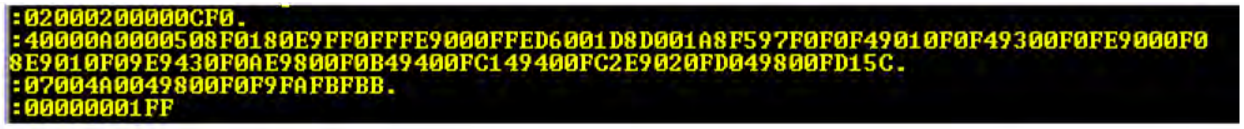
3. Figure 20 indicates that the firmware is successfully programmed to the target device.



```
Hex File Received Successfully

Programming Target Device .....
Target Device on Debug Mode...
Flash Erased...Programming...
Completed Successfully!
Finished
Press ESC key to continue.....
```

Figure 20. Target Device Successfully Programmed with Firmware from the PC

4. When saving firmware to Flash memory and when sending firmware from the PC to the target device, hexadecimal data is displayed in the serial terminal program. An example is shown in Figure 21.



```
:02000200000CF0.
:40000A0000508F0180E9FF0FFFE9000FFED6001D8D001A8F597F0F0F49010F0F49300F0FE9000F0
8E9010F09E9430F0AE9800F0B49400FC149400FC2E9020FD049800FD15C.
:07004A0049800F0F9FAFBFBB.
:00000001FF
```

Figure 21. Example Hex File Display

Running the Application in Standalone Mode

After Standalone Mode is selected through Port E[2] (see the Selecting PC Mode or Standalone Mode section on page 16), press the SW3 pushbutton located on the F6482 Series Development Board. Pressing this pushbutton programs the target device with firmware from Program Slot 1 in external Flash memory.

If the target device is successfully programmed, the green LED turns ON. If programming of the target device is unsuccessful, the red LED turns ON.

Note: Standalone Mode is only operational if firmware is stored in Program Slot 1 in external Flash memory.

Results

The code developed for this application easily and successfully programs the Z8 Encore! Family of microcontrollers through an on-chip debugger by using the Z8F6482 MCU as a host.

Summary

This application programs the Z8 Encore! Family of microcontrollers through an on-chip debugger (OCD) using a host Z8F6482 MCU, which is mounted on the F6482 Series Development Board. Firmware received from a PC in Intel Hex format is programmed to the target device.

The application program runs in two modes: PC Mode and Standalone Mode. In PC Mode, firmware from the PC can either be saved to the host MCU's internal or external Flash memory before being programmed to the target device, or it can be programmed directly to the target device. In Standalone Mode, firmware saved to Program Slot 1 in external Flash memory can be programmed to the target device with the press of a push-button.

References

The documents associated with the Z8 Encore! Family of MCUs are listed in Table 7. Each of these documents can be obtained from the Zilog website by clicking the link associated with its Document.

Table 7. F6482 Series MCU Documentation

Document ID	Description
PS0294	F6482 Series Product Specification
PS0199	F64xx Series Product Specification
PS0250	F1680 Product Specification
PS0263	F083A Series Product Specification
PS0228	F082A Series Product Specification
PS0251	F0830 Series Product Specification
PS0246	Z8FMC16100 Series Product Specification
UM0263	F6482 Series Development Kit User Manual
UM0151	F64XX Series Development Kit User Manual
UM0203	F1680 28-Pin Series Development Kit
UM0206	F083A Series Development Kit User Manual
UM0186	F082A Series Development Kit User Manual
UM0192	Z8FMC16100 Series Motor Control Development Kit User Manual

Appendix A. OCD API Functions

Table 8 lists the OCD API functions used in this application.

Table 8. OCD API Functions

Function	**Description**
Function:	ui8OCD_API_StartDebugMode
Parameter:	void
Return:	TRUE or FALSE
Description:	This function sets the target device to Debug Mode.
Function:	ui8OCD_API_SendCommandAndData
Parameter:	void
Return:	TRUE or FALSE
Description:	Sends an OCD command and data, then waits for a reply from the target device.
Function:	ui8OCD_API_SendDebugCommand
Parameter:	void
Return:	TRUE or FALSE
Description:	This function runs ui8OCD_API_SendCommandAndData() and handles errors.
Function:	ui8OCD_API_SetTargetFlashFreq
Parameter:	void
Return:	TRUE or FALSE
Description:	This function sets the target device Flash frequency.
Function:	ui8OCD_API_UnlockTargetFlash
Parameter:	ui8OCDFlashPage - page to unlock
Return:	TRUE or FALSE
Description:	This function sends an unlock sequence to access the target device Flash.
Function:	ui8OCD_API_LockTargetFlash
Parameter:	void
Return:	TRUE or FALSE
Description:	This function locks the target device Flash.
Function:	ui8OCD_API_MassEraseTargetFlash
Parameter:	void
Return:	TRUE or FALSE
Description:	This function requests a mass erase from the target device Flash controller.
Function:	ui8OCD_API_ProgramTargetFlash
Parameter:	uiTargetFlashStartAddr – starting address of target Flash memory aui8TargetFlashData – location of data to be sent to target Flash uiTargetFlashDataLength – length of data to be sent to target Flash
Return:	TRUE or FALSE

Table 8. OCD API Functions (Continued)

Description:	This function flashes the entire target device Flash memory.
Function:	ui8OCD_API_VerifyTargetFlash
Parameter:	uiTargetFlashStartAddr – target Flash start address aui8TargetFlashData – program code uiTargetFlashDataLength – length of code
Return:	TRUE or FALSE
Description:	This function verifies the target Flash content with the programmed data.
Function:	ui8OCD_API_SendFW_INTToTarget
Parameter:	aui8TargetFlashData – firmware from internal Flash to target device uiTargetFlashStartAddr – address of the Flash target device
Return:	TRUE or FALSE
Description:	This function sends firmware from the host MCU’s internal Flash to the target device’s Flash memory.
Function:	ui8OCD_API_SendFW_EXTToTarget
Parameter:	ui8EXTFlashProgramSlot – program slot (1 - 4) aui8TargetFlashData location of firmware to be sent to target device
Return:	TRUE or FALSE
Description:	This function sends firmware from external Flash to the target device’s Flash memory.
Function:	OCD_API_ReceiveByte
Parameter:	ui8Data – byte received from target device
Return:	void
Description:	This function handles incoming data from the target device.
Function:	ui8OCD_API_ReadOCDReg
Parameter:	ui8CommandByte – OCD command byte ui8RXDataLength – length of data from target device aui8Result – location to store results
Return:	TRUE or FALSE
Description:	This function reads the OCD register and returns the result.
Function:	ui8OCD_API_WriteControlReg
Parameter:	ui8WriteControlRegValue – value to be written to OCD control register
Return:	TRUE or FALSE
Description:	This function writes a value to the OCD control register.
Function:	ui8OCD_API_ReadControlReg
Parameter:	aui8Result - location to store results
Return:	TRUE or FALSE
Description:	This function reads the OCD control register and returns the result.
Function:	ui8OCD_API_ReadRevisionReg
Parameter:	aui8Revision – location to store results
Return:	TRUE or FALSE

Table 8. OCD API Functions (Continued)

Description:	This function reads the OCD revision register and returns the result.
Function:	ui8OCD_API_ReadStatusReg
Parameter:	aui8Result – location to store results
Return:	TRUE or FALSE
Description:	This function reads the OCD status register and returns the result.
Function:	ui8OCD_API_ReadReg
Parameter:	uiTargetDeviceRegisterAddr - target device register aui8Result – location to store results
Return:	TRUE or FALSE
Description	This function reads the target device registers through the OCD and returns the results.
Function:	ui8OCD_API_WriteReg
Parameter:	uiTargetDeviceRegisterAddr – target device register auiData – location of data to write to register ui8CommandDataLength – length of data to be sent (max 4 bytes)
Return:	TRUE or FALSE
Description:	This function writes data to the target device registers through OCD and returns the results.
Function:	ui8OCD_API_WriteTargetFlash
Parameter:	uiTargetFlashStartAddr – start address on the target device to write data aui8TargetDeviceFlashData – location of bytes to program uiTargetDeviceFlashDataLength – number of bytes to program
Return:	TRUE or FALSE
Description:	This function writes data to target device Flash through OCD and returns the results.
Function:	ui8OCD_API_ReadTargetFlash
Parameter:	uiTargetFlashStartAddr – start address on the target device to read data aui8TargetDeviceFlashData – location to store the bytes uiTargetDeviceFlashDataLength – number of bytes to read
Return:	TRUE or FALSE
Description:	This function reads data from target device Flash through OCD and returns the results.
Function:	OCD_API_HandleBreak
Parameter:	void
Return:	void
Description:	This function handles errors with a single repeat.

Appendix B. Flow Charts

Figure 22 shows the main flow of the OCD programmer.

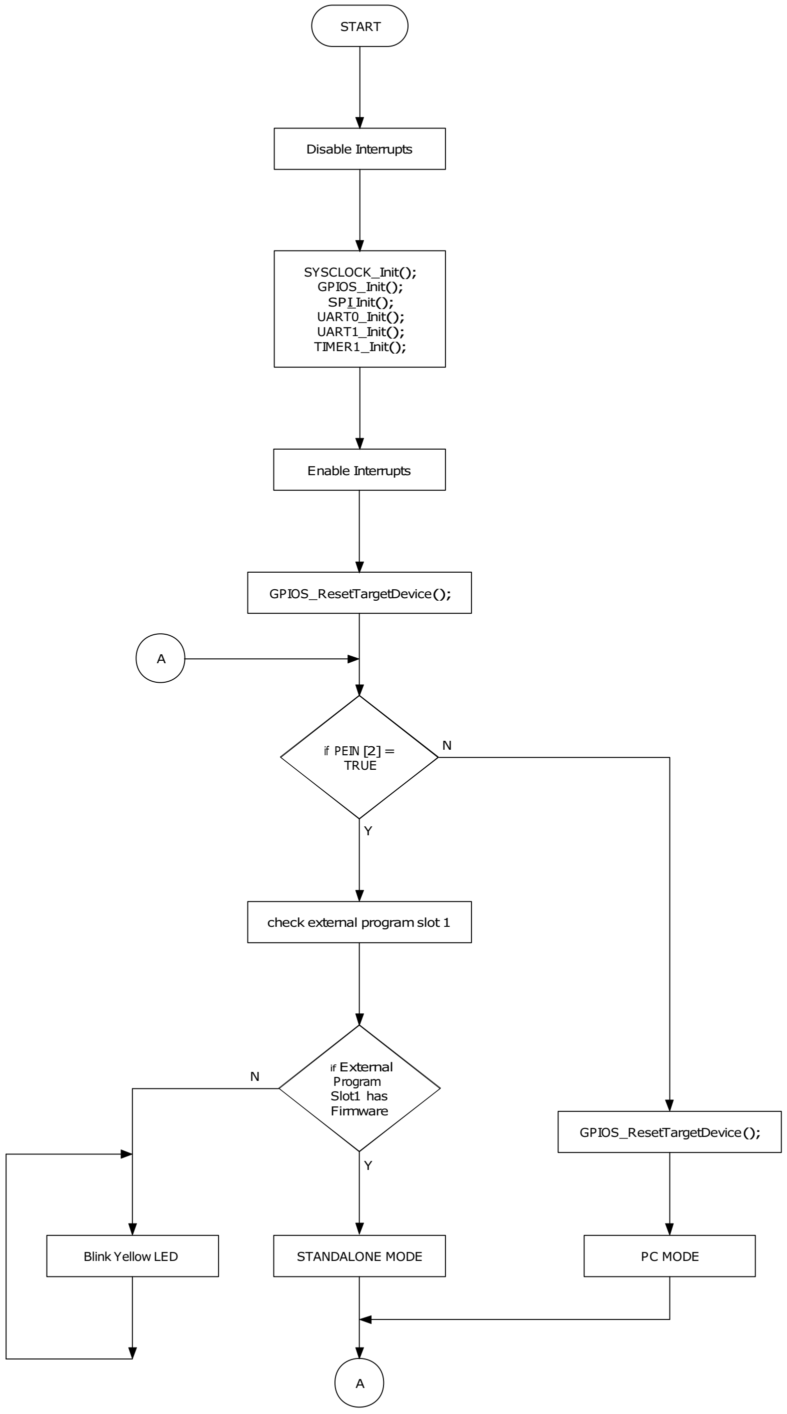


Figure 22. Main Flow of the OCD Programmer

Figures 23 and 24 show flows of the OCD programmer in Standalone Mode and PC Mode respectively.

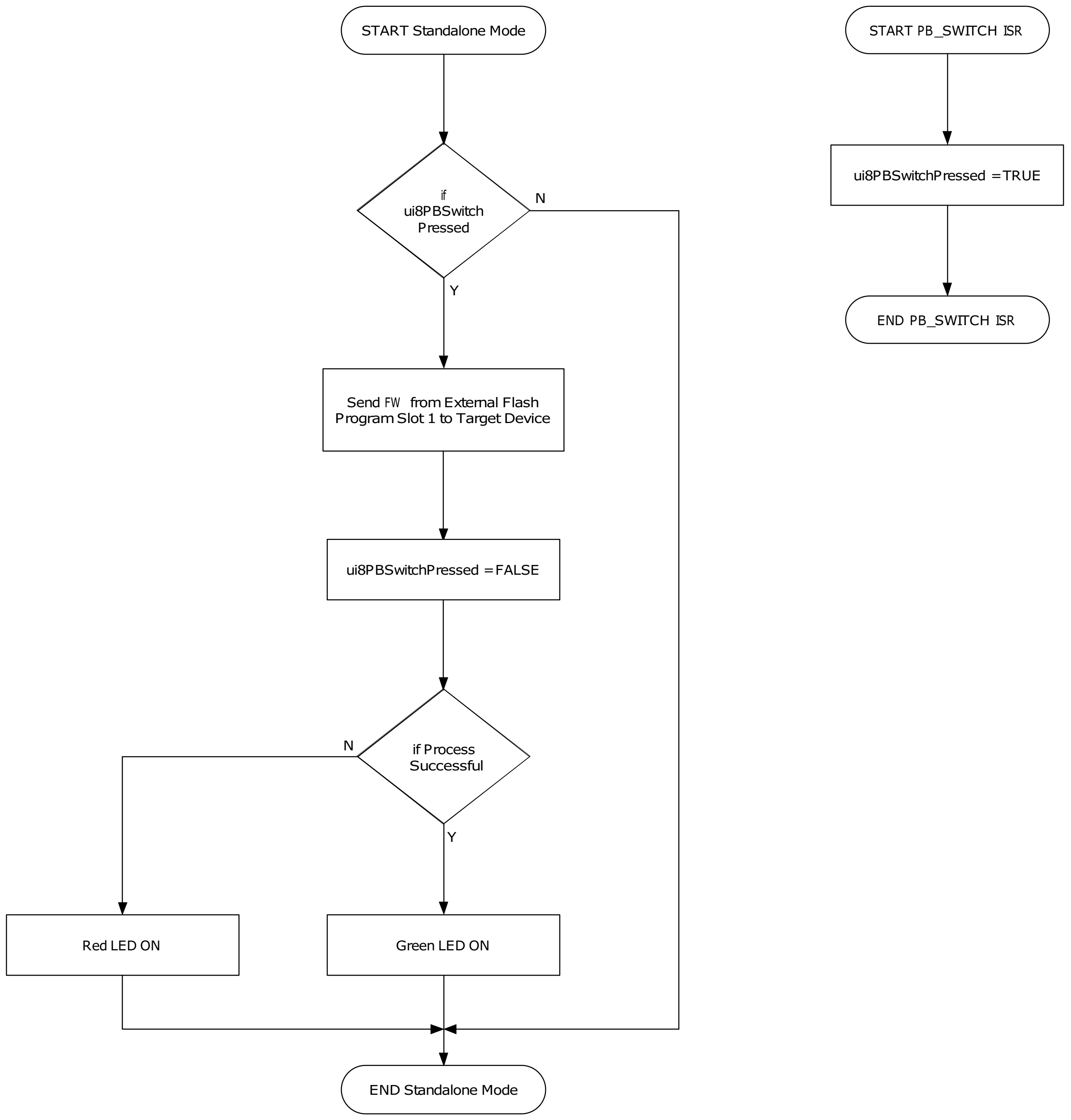


Figure 23. Standalone Mode Flow Chart

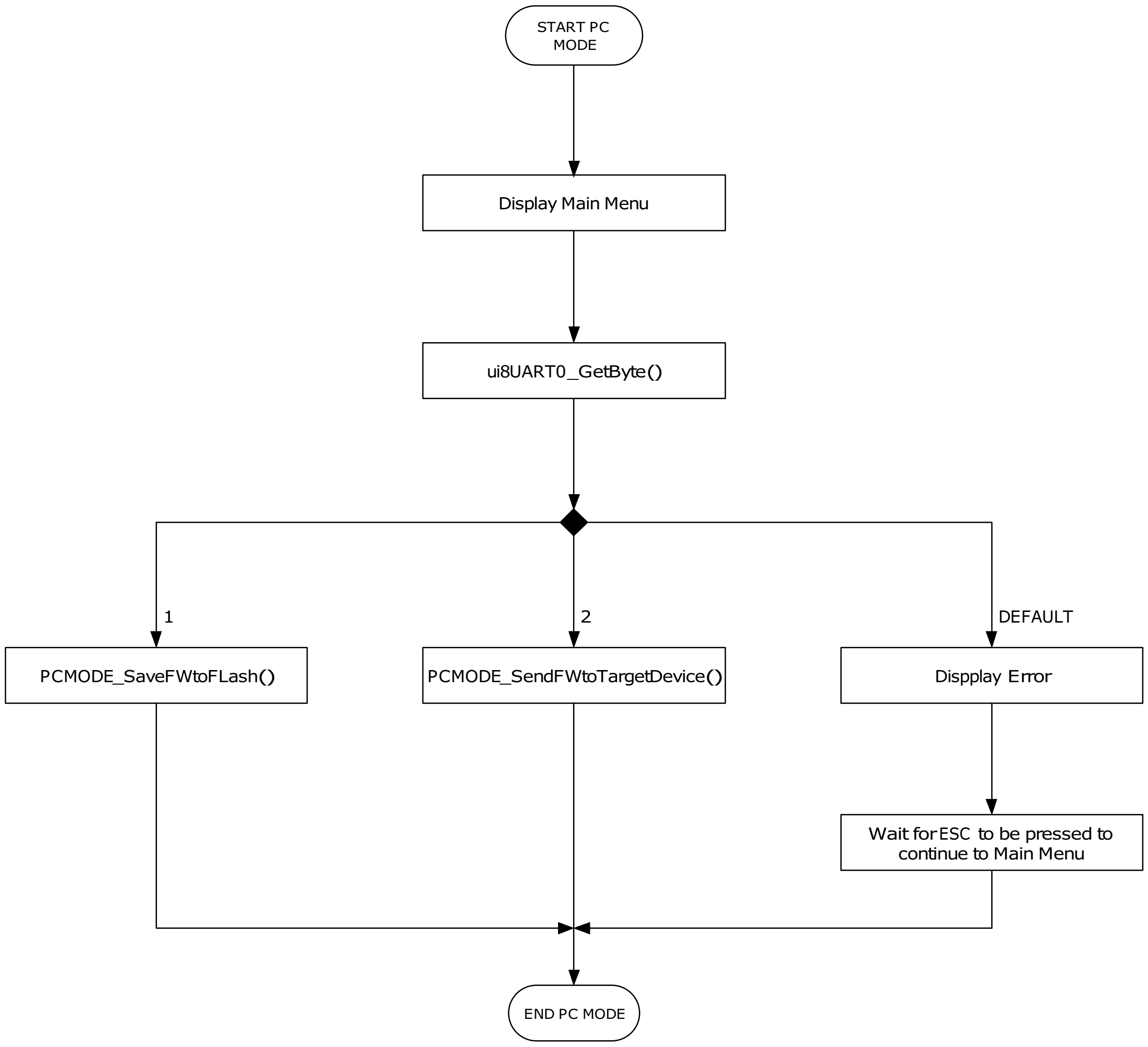


Figure 24. PC Mode Flow Chart

Figure 25 depicts the flow of saving the firmware to Flash memory.

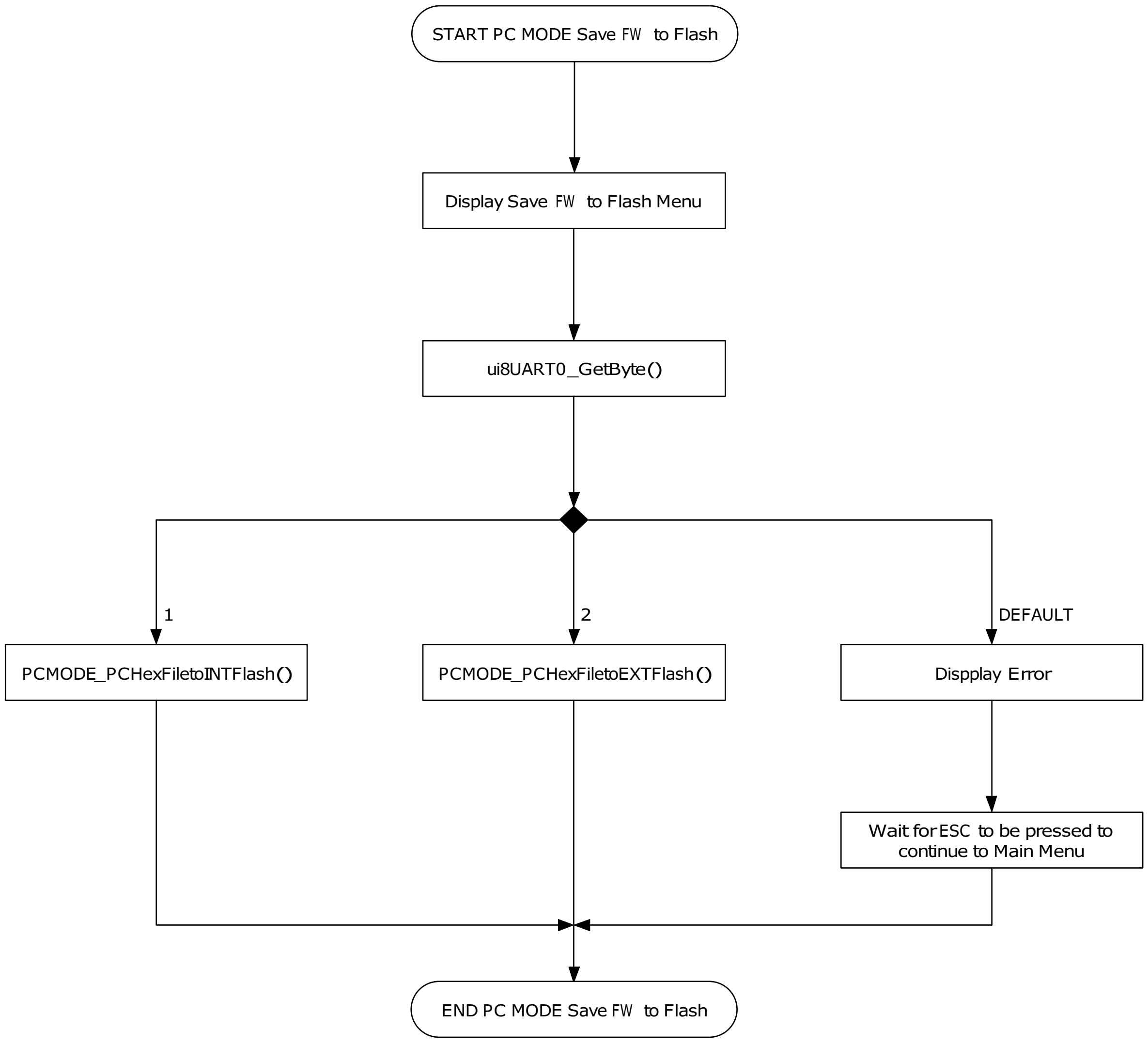

START PC MODE Save FW to Flash
Display Save FW to Flash Menu
ui8UART0_GetByte()
1
2
DEFAULT
PCMODE_PCHexFiletoINTFlash()
PCMODE_PCHexFiletoEXTFlash()
Dispplay Error
Wait for ESC to be pressed to continue to Main Menu
END PC MODE Save FW to Flash

Figure 25. Flow Chart for Saving Firmware to Flash Memory

The process of sending the firmware to the target device is shown in Figure 26.

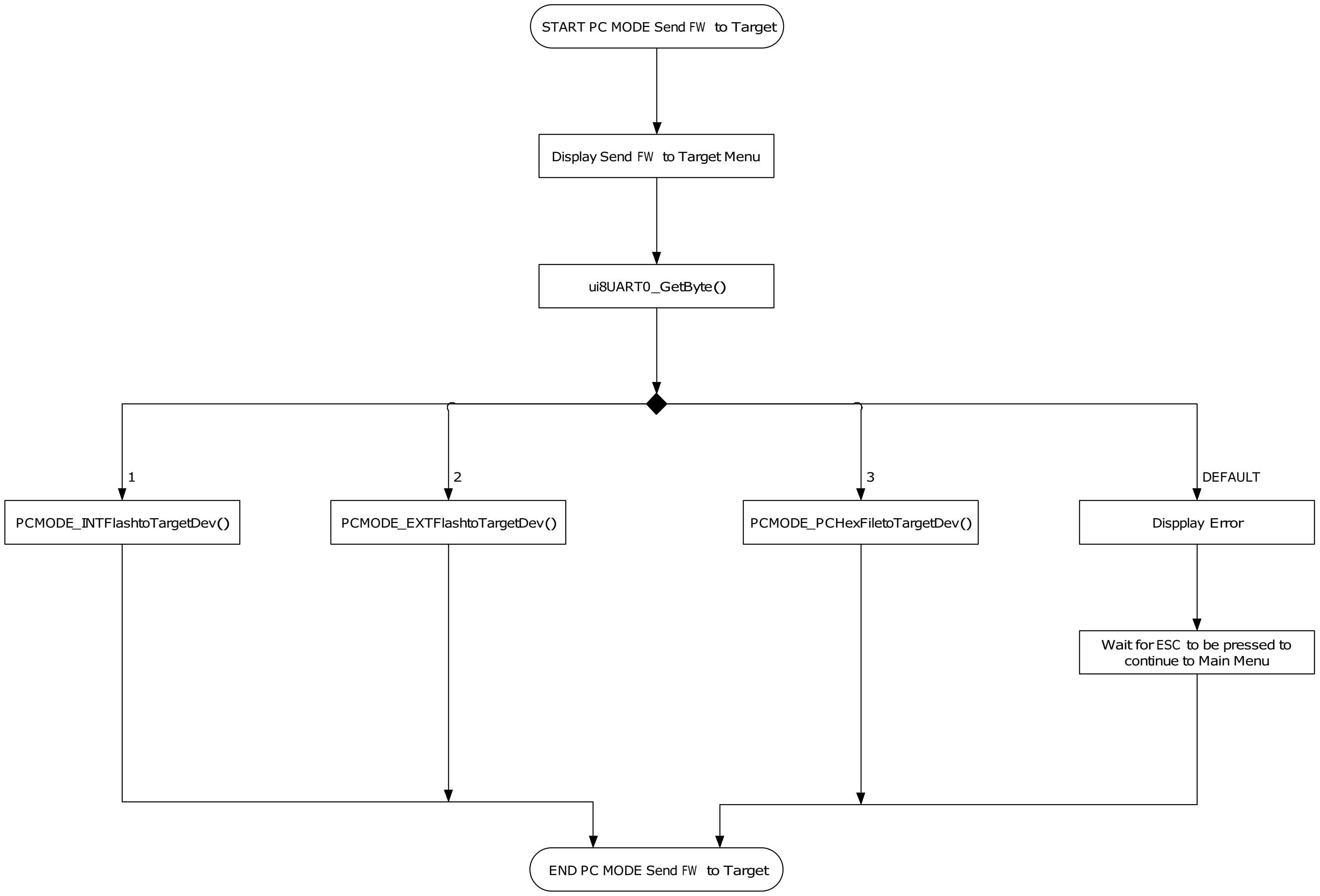

START PC MODE Send FW to Target
Display Send FW to Target Menu
ui8UART0_GetByte()
1
2
3
DEFAULT
PCMODE_INTFlashtoTargetDev()
PCMODE_EXTFlashtoTargetDev()
PCMODE_PCHexFiletoTargetDev()
Dispplay Error
Wait for ESC to be pressed to continue to Main Menu
END PC MODE Send FW to Target

Figure 26. Flow Chart for Sending Firmware to the Target Device

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.