xCORE-200 DESIGN ADVISORY

XM-014276-DA : 90nm to 65nm Flash Memory Change

Important Notice

This document provides essential information and actions to be taken following a product update to a range of XMOS devices which have integrated flash memory. Failure to implement this advisory could result in incorrect factory programming and the inability to execute in-service DFU over USB upgrade processes.

This Design Advisory should be read in conjunction with Product Change Notice **PCN XM-014218-PN**.

Part numbers covered by this document are all xCORE-200 devices that have embedded flash memories, i.e. part numbers that start with **XLF2xx-, XUF2xx- and XEF2xx-**

XMOS

Bringing technology to life

Contents

1. OVERVIEW

1.1. DESCRIPTION OF CHANGE

Flash memory supplier ISSI has updated the silicon process they use to produce the flash memory device integrated into a range of XMOS xCORE-200 products. The ISSI flash memory devices based on a 90nm process (part number IS25LQ016B) are now at End-of-Life (EOL) and have been replaced by a form, fit and function direct replacement (IS25LP016D) which is manufactured on a 65nm process.

These flash devices are integrated in a range of XMOS devices and from 4Q2020 devices with the IS25LQ016B integrated will be discontinued and replaced with new, compatible parts containing the new IS25LP016D version of the flash memory integrated.

Devices effected by this Design Advisory are xCORE-200 devices with embedded flash memory. These devices have part numbers that start with:

- XLF2xx-
- XUF2xx-
- XEF2xx-

For ordering the new devices, existing part number structures are retained, but the new parts are differentiated with the addition of an 'A' suffix on the replacement part order codes.

This change has been notified in XMOS Product Change Notification (PCN) document XM-014218-PN, which contains the full list of part numbers and their new order codes. This is available on the Advisories section of the XMOS website here: https://www.xmos.ai/advisories-and-notices/The new devices that have the IS25LP016D integrated can be identified from the date code line on the top marking of new 65nm integrated flash devices. The printed date code line will have an 'A' suffix added to the date code as shown in the figure below:

Identification of package with new 65nm embedded flash device

This Design Advisory to details changes required to the USB Audio 2.0 firmware as a result of this product change.

1.2. IMPACT OF CHANGE

The performance and functionality of the replacement parts are unchanged and can continue to be used in existing and new designs. However, the new flash devices have a different JEDEC Standard Identification Code (JEDEC ID), which is a hardware identifier for flash devices.

Old devices : IS25LQ016B – ID 0x9D4015.

New devices: IS25LP016D – ID 0x9D**6**015

In designs based on the XMOS USB Audio 2.0 reference, this JEDEC ID is read by the XMOS firmware that programmes the image onto the flash memory. This ID is then cross-checked with a list of supported devices to verify that a valid flash device is being programmed. The ID reported by the new devices is not contained in this list, so it is not recognised as valid by the programming tools

which causes the programming process to abort. This is a deliberate action to prevent misconfiguration of the flash which could be irreversible.

The same issue affects in service Device Firmware Upgrade (DFU) using XMOS supplied DFU libraries as the same verification process takes place during the DFU operation.

Products developed using the XMOS USB Audio 2.0 reference software, that use the embedded flash version of the xCORE-200 processor, are therefore impacted by this change in two areas:

Area	Products Impacted	Impact of new device	Action Required
Device Firmware Upgrade (DFU)	xCORE-200 parts with embedded flash	Firmware change will abort. Upgrade and factory revert no longer possible	Modify source code as shown in section 2.2.2 and recompile firmware image with new .flash specification. Then install on all product built with the new 'A' parts before running field upgrades
Factory Programming (xflash tools)	xTIMEcomposer 14.3 and earlier	Unable to programme new 'A' devices	Use updated .spispec file for device programming

Note: USB Audio 2.0 reference software used on xcore-200 devices with external flash memory are not impacted by the change described in this advisory.

The following sections describe the specific modifications to the firmware source code required to support the new flash device ID. The required changes in the source code are very constrained and do not impact normal in-service operation of the device. Once implemented the modified firmware will run correctly on both new and old parts.

The source code changes are detailed for the XMOS USB Audio 2.0 reference design. Users with their own custom designs must integrated these changes into their own source code and recompile to create a new firmware image for factory programming.

2. DEVICE FIRMWARE UPGRADE (DFU)

2.1. DFU PROCESS AND CUSTOM APPLICATION FLASH ACCESS

The DFU process comprises the host system(e.g. a PC) establishing a USB connection with the XMOS device and sending a request to USB Endpoint 0 that triggers the DFU process. The DFU process (executing on the xCORE-200 device) writes an upgrade image, transferred via the USB, to the flash memory. The existing firmware on the device executing a DFU process or custom application flash access on the xCORE is responsible for managing the co-packaged flash chip.

To support a range of different flash devices, the generic *quadflash* library used by DFU includes a mandatory cross-check of the JEDEC ID, read from the flash chip, against several flash specification files provided with the library.

However, the built-in flash specifications in the *quadflash* library of current firmware builds do NOT INCLUDE the specification for the new flash device used in the XMOS devices covered by this design advisory. When the existing DFU firmware executes on one of these replacement devices it detects an invalid flash device and aborts the update operation.

If the new 'A' parts are factory programmed with the existing firmware image, the consequence of this is that it will no longer be possible to modify the version of firmware stored in these devices in the field via the DFU mechanism.

2.2. ACTION REQUIRED

To ensure that DFU is supported for new parts users will need to update the XMOS device software with a new firmware image as follows.

2.2.1. PREREQUISITES

Please ensure that the correct version of xTIMEcomposer is installed; release 14.3.2 is required for this product release. This tools version can be downloaded from the XMOS website:

https://www.xmos.ai/software-tools/

Before compiling the new firmware, ensure that the correct version of the tools is installed and in the command path by executing the following command in an xTIMEcomposer terminal window.

```
xcc --version
```

If the version is something other than 14.3.2 or the command fails, follow the documentation included in the tools release.

2.2.2. SOURCE CODE MODIFICATIONS

For devices that use custom firmware that is based on the USB Audio 2.0 reference design source code with modifications, this source code must be updated and a new binary image compiled and used for new 'A' devices.

MANDATORY CHANGE

The source files that need to be changed to support the new device are (shown in the path location of the USB Audio 2.0 reference design package):

```
sc_usb_audio/module_usb_audio/flashlib_user.c

sw_usb_audio/app_<application>/src/core/customdefines.h
```

The first file is common to all applications, while the second is application specific. The required changes in these files are shown in Appendix A of this document. These changes can be applied manually by copying the appropriate code fragments into the product source tree and recompiling the firmware.

OPTIONAL CHANGE (UPDATE VERSION NUMBER)

Update the product version by modifying:

```
sc_usb_audio/module_usb_audio/devicedefines.h
```

and increment that value defined by

```
#define BCD_DEVICE_N
```

and update the CHANGELOG with the new unique version number.

Note: If a custom, implementation specific, version number has been used in a design, the appropriate updates should be done instead.

COMPILATION

This source code should then be recompiled to generate a new version of the firmware.

Navigate to the required `app_` directory in `sw_usb_audio` and run:

```
xmake CONFIG=<desired config>.
```

For example, to build the `hifi` config, run

```
xmake CONFIG=hifi
```

The resulting .xe file in

```
bin/hifi
```

now contains the necessary compatibility changes for the new flash part. This can be loaded onto the xCORE device via the `xflash` tool as described in Section 3.2.2 below

2.3. FREQUENTLY ASKED QUESTIONS ON DFU

2.3.1. CAN THE NEW FIRMWARE BE USED ON BOTH ORIGINAL AND NEW XMOS PARTS?

YES. The modification documented in section 2.2.2 above includes the flash specifications for both the original and new XMOS parts. This ensures that the modified firmware (i.e. recompiled following the modification above) can be successfully deployed onto either original or new XMOS parts. In both cases these parts will support DFU and custom application flash access.

2.3.2. CAN EXISTING FIRMWARE BE RETAINED?

NOT RECCOMENDED. If DFU or custom flash access is NOT required in existing applications, then it is possible to use the new parts as a direct replacement for the original parts and continue to deploy that same firmware image to Flash (see section 3 for details on XFLASH tool support)

However, by deploying a firmware image based on unmodified releases onto new parts, it will **never be possible** to make in-the-field changes (i.e. you CANNOT Upgrade, Downgrade or Revert to Factory) to these parts. In order to avoid this XMOS recommends that all existing firmware is updated to support the new devices before they are introduced.

2.3.3. WHAT HAPPENS IF DFU IS ATTEMPTED ON OLD FIRMWARE?

If a DFU is attempted on a new part loaded with the original firmware, it will result in a runtime exception (in the code running on the XMOS part). The device will recover to its factory image following a power-cycle, the flash contents will be un-changed. This is true for download, revert and upload operations, so it will never be possible to change the version of firmware on the device.

To avoid this scenario, XMOS strongly recommends that all production firmware images are updated to the new version before introducing the new parts into a production flow. This will reduce the risk of the older firmware version being loaded as a factory image on these new 'A" parts.

In certain scenarios the host software may not display an error, but in most cases there will be a visible error reported. XMOS recommends checking the behaviour of products in the expected use cases following the DFU operation.

2.3.4. WHAT HAPPENS IF A USER UPDATES TO AN OLDER VERSION OF THE FIRMWARE?

If a user attempts to load an older version of the firmware onto a board with new "A" devices the DFU process will install this older version and boot the device from it. This image will **no longer be able to change the flash device** and it is no longer possible to modify the firmware image via DFU. Any further changes require physical access to the device via the JTAG/ interface and the use of the `xflash` tool.

It is critical that system designers implement version controls at the product level to prevent users loading incorrect versions of software onto this device. E.g. users should be notified that old DFU upgrade images that may exist in the field must no longer be used and instead be replaced with a new DFU image.

2.4. SUMMARY OF PROGRAMMING AND DFU SCENARIOS

Key:

Original firmware image	A firmware image created for the current XMOS parts, prior to the modification.
Future unmodified firmware image	An updated version of the firmware but EXCLUDING the modifications to the firmware (as described in 2.2.2 above)
New firmware image	A firmware image created for the new XMOS parts with the new Flash integrated and including the modifications to the firmware (as described in 2.2.2 above)
Future new firmware image	A subsequent firmware image created for the new XMOS parts (with the new Flash integrated) and including the modifications to the firmware (as described in 2.2.2 above)

ID	PART	FACTORY PROGRAMMED	ATTEMPT UPGRADE TO	OUTCOME
1	Original part	Original firmware image	New firmware image Or Future unmodified firmware image	From the factory, device operates correctly and is upgradeable. Attempts to DFU a "Future firmware image" will succeed.
2	Original part	New firmware image	Future firmware image	From the factory, device operates correctly and is upgradeable. Attempts to DFU a "Future firmware image" will succeed.
3	Original part	New firmware image	Original firmware image	This DFU will succeed and the part remains upgradeable as per (1) or (2) above.
4	New part	Original firmware image	New firmware image	From the factory, device operates correctly but is not upgradeable.  Attempts to DFU the "New firmware image" will fail. See 2.3.3 above Device will revert to operation according to its factory image on a power-cycle.
5	New part	New firmware image	Future new firmware image	From the factory, device operates correctly and is upgradeable. Attempts to DFU a "Future firmware image" will succeed.
6	New part	New firmware image	Original firmware image	The DFU will succeed but will leave the device in the state of (4) above and so  All subsequent attempts to DFU (including revert and down-grade) will fail. See 2.3.4 above After this event the device will work correctly as per the original firmware image.

3. XFLASH

3.1. IMPACT ON XFLASH

The `xflash` tool used for programming the firmware image into the device. When using the replacement XMOS parts, certain combinations of `xflash` version and firmware configuration will cause `xflash` to print an error message during factory programming and stop instead of completing successfully.

To check the version of xflash installed enter the command line:

```
xflash --version
```

3.1.1. XFLASH VERSION 14.4.1

xTIMEcomposer V14.4.1 already incorporates support of the IS25LP016D flash device so, provided the .xe file has been complied with the modifications described above included, programming of the embedded device will occur correctly.

3.1.2. XFLASH 14.3.3 AND EARLIER

With earlier versions of xTIMEcomposer, an error message is displayed by `xflash` when it is run:

```
Error on tile[0]: failed to connect to flash device. Please verify that SQI
type is supported and that the correct SQI ports are defined within your xn
file.
```

If the `--no-inq` command option is used, a different error message is printed:

```
Error: F03013 Failed to run : 0x7ffee7480b50
```

Note: The memory address in the latter error message may differ in specific design configurations.

3.2. CORRECTING XFLASH ERRORS

The errors in section **Error! Reference source not found.** are generated because the `xflash` tool versions earlier than 14.4.1 do not automatically recognise the flash device. The error messages shown above are removed by creating a new specification file (.spispec file) for the embedded flash device and adding this as a command line argument to `xflash`.

3.2.1. .SPISPEC FILE

The `IS25LP016D.spispec` required for the new 'A' parts must have the following contents:

```
21,                         /* Enum value to identify the flashspec in a list */
256,                        /* page size */
8192,                       /* num pages */
3,                          /* address size */
4,                          /* log2 clock divider */
0x9F,                       /* QSPI_RDID */
0,                          /* id dummy bytes */
3,                          /* id size in bytes */
0x9D6015,                   /* device id */
0x20,                       /* QSPI_SE */
4096,                       /* Sector erase is always 4KB */
0x06,                       /* QSPI_WREN */
0x04,                       /* QSPI_WRDI */
PROT_TYPE_NONE,             /* no protection */
{{0,0},{0x00,0x00}},        /* QSPI_SP, QSPI_SU */
0x02,                       /* QSPI_PP */
0xEB,                       /* QSPI_READ_FAST */
1,                          /* 1 read dummy byte */
SECTOR_LAYOUT_REGULAR,      /* mad sectors */
{4096,{0,{0}}},             /* regular sector sizes */
0x05,                       /* QSPI_RDSR */
0x01,                       /* QSPI_WRSR */
0x01,                       /* QSPI_WIP_BIT_MASK */
```

This .spispec file can be stored anywhere on the computer used for programming, but it is advised to copy it into the current working directory where the `xflash` command is run from.

3.2.2. XFLASH COMMAND LINE CHANGE

A typical factory programming procedure could include the following steps, or similar.

```
xflash --noinq --factory <app_xxx>.xe -o flash.bin
xflash --write-all flash.bin
```

When used with new "A" parts the second command will yield an error.

This is corrected by adding `--spi-spec IS25LP016D.spispec` to the second command, which then becomes:

```
xflash --spi-spec IS25LP016D.spispec --write-all flash.bin
```

This changed command must be made only applied for boards built with the new "A" parts. For the older parts the `xflash` command should be run without the `--spi-spec` option.

4. TESTING

XMOS strongly recommends that any modifications made based on this Design Advisory are to be fully and robustly tested. Customers are responsible for how they manage their firmware updates. Modified software, utilities or tools may impact the advice provided above, and changes must be tested in the final application environment before release.

4.1. VERIFICATION OF MODIFICATION

The following notes provide guidance to implementers on verification that the modifications described in this design advisory have been implemented correctly.

4.1.1. CHECK THE BCD DEVICE VERSION NUMBER

STEP 1 - CHECK CONNECTED USB DEVICES FOR THE UPDATED VERSION NUMBER:

- xmosdfu (Linux & macOS)

  Linux or macOS users can check the BCD device version number with the `xmosdfu` utility by running

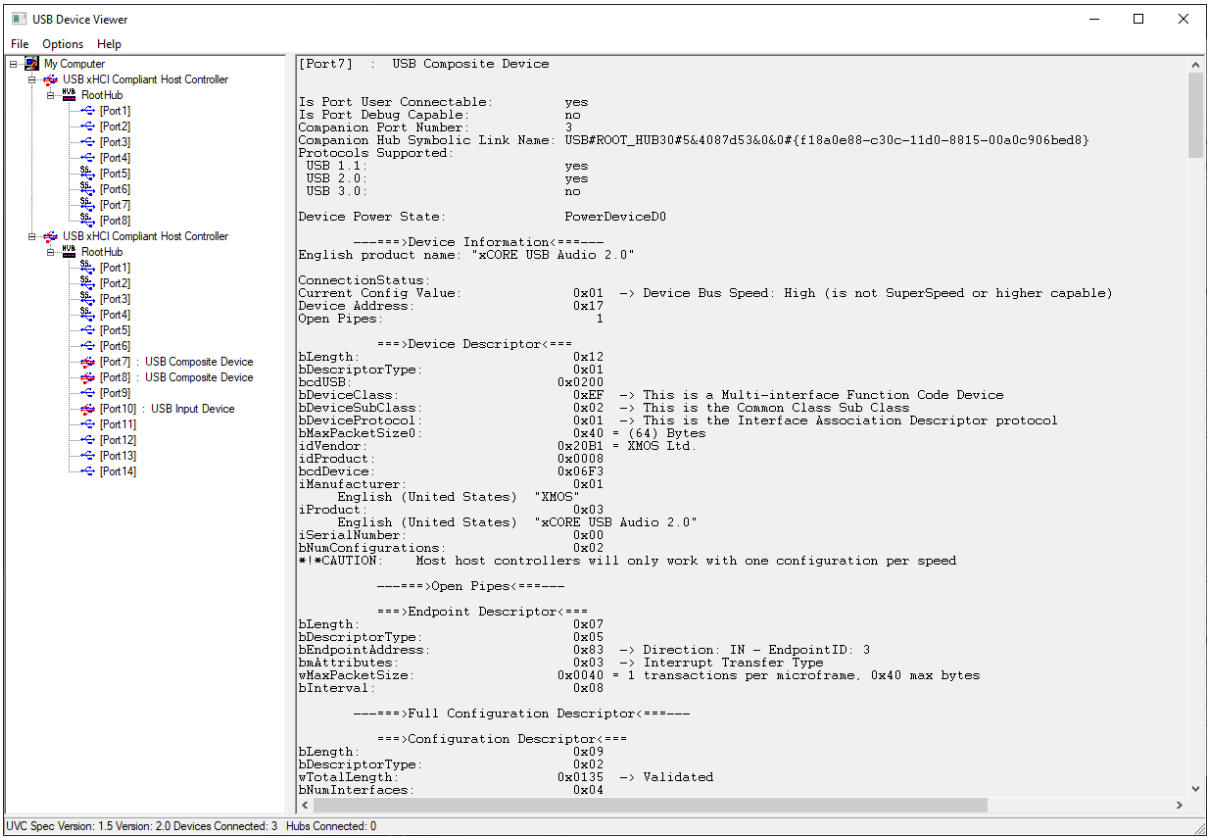
```
xmosdfu --list-devices
```

  - Look for the line in the output where the VID and PID match the following:

```
VID = 0x20b1, PID = 0x11, BCDDevice: <note this value>
```

  - Note the BCDDevice

- USBView (Windows)
  - Download USBView Following the instructions on this page: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/usbview
  - Open USBView in File Explorer: Open the install directory (`C:\Program Files (x86)\Windows Kits\10\Debuggers\x64`) and select `USBView.exe`
  - Find the `xCORE USB Audio 2.0` by selecting devices in the left pane and checking the `iProduct` value under `Device Descriptor` in the right pane.



  - Look in the right pane for `bcdDevice`.

STEP 2 CHECK THE BCDDEVICE VALUE

- If BCDDevice is as changed in section 2.2.2 the firmware has been successfully updated.
- If BCDDevice is 0x6f2 – Firmware is the old , unmodified version of the reference firmware.
- If BCDDevice is 0x9901 – Device is running a DFU test build of the firmware.

Note: These values are for the reference software. If a custom version number scheme has implemented in a specific design, check the BCDDevices codes match the version expected.

4.1.2. VERIFICATION OF COMPILED BINARY IMAGES

To build the updated firmware ensure that the source code has been updated with the required changes described in Section 2.2.2 above and detailed in the Appendix of this document. This source code should then be recompiled and flashed onto the device. If the BCD Device version number has been changed to a custom value, this can be verified using the instructions in section 4.1.1 above.

Re-build the firmware, but this time add `TEST_DFU_1=1` to the `xmake` command.

For example, if the original `xmake` command was:

```
xmake CONFIG=hifi
```

then run the command:

```
xmake CONFIG=hifi TEST_DFU_1=1
```

Rename the resulting .xe file to “dfu.xe”.

Created a DFU binary by running the following `xflash` command:

```
xflash --factory-version 14.3 --upgrade 1 dfu.xe -o dfu.bin --no-compression
```

With the device connected, download the DFU image to the device by running the following `xmosdfu` command:

```
sudo ./xmosdfu XMOS_SU1_AUDIO2_PID --download dfu.bin
```

Check the BCD Device Version number by following instructions in Section 4.1.1 to verify that the device is running the DFU test version of the firmware.

5. FURTHER INFORMATION

DOCUMENT TITLE	DOWNLOAD
xTIMEcomposer Tools (including xflash)	https://www.xmos.ai/software-tools/

Questions regarding this Design Advisory can be addressed by raising a support ticket via https://www.xmos.ai/support/

6. REVISION HISTORY

DOCUMENT VERSION	RELEASE DATE	CHANGE DESCRIPTION
XM-014276-DA-1	15 September 2020	Initial Release

APPENDIX A: DETAILS OF SOURCE CODE CHANGES REQUIRED

This Appendix details the changes required in order to implement support for the new 'A' parts in designs based on the XMOS USB Audio 2.0 software.

There are two changes that must be made exactly as specified in all implementations.

Since specific product designs will have modified the reference design, only the specific sections to be modified as shown. Users will need to apply the specific modifications to their versions of the source code.

The changes are shown below in github diff format; source lines below highlighted in green indicate code added; in red code that should be removed, and also as code blocks that can be copied directly. (sections in blue indicate unchanged code)

File #1 – sc_usb_audio/module_usb_audio/flashlib_user.c

This module is common to all applications. The code needs a change to extend the `fl_connectToDevice` function call to support more than 1 flash specification as defined by the `#define DFU_FLASH_DEVICE` contained in an application specific header file – see File 2 below.

16 module_usb_audio/flashlib_user.c

```
@@ -19,9 +19,15 @@
  #define portout(a,b) {__asm__ __volatile__("out res[%0], %1": : "r" (a) , "r" (b));}

  #ifdef DFU_FLASH_DEVICE
+
+ #ifdef QUAD_SPI_FLASH
+ /* Using specified flash device rather than all supported in tools */
+ fl_QuadDeviceSpec flash_devices[] = {DFU_FLASH_DEVICE};
+ #else
  /* Using specified flash device rather than all supported in tools */
  fl_DeviceSpec flash_devices[] = {DFU_FLASH_DEVICE};
  #endif
+ #endif

  #ifdef QUAD_SPI_FLASH
  /*
@@ -83,7 +89,15 @@ int flash_cmd_enable_ports()
  #endif

  #ifdef DFU_FLASH_DEVICE
-     result = fl_connectToDevice(&p_flash, flash_devices, 1);
+ #ifdef QUAD_SPI_FLASH
+     result = fl_connectToDevice(
+         &p_qflash, flash_devices, sizeof(flash_devices) / sizeof(fl_QuadDeviceSpec)
+     );
+ #else
+     result = fl_connectToDevice(
+         &p_flash, flash_devices, sizeof(flash_devices) / sizeof(fl_DeviceSpec)
+     );
+ #endif
  #else
      /* Use default flash list */
  #ifdef QUAD_SPI_FLASH
```

Specific code changes to flashlib_user.c

```
ADDED Lines 22-26:
#ifdef QUAD_SPI_FLASH
/* Using specified flash device rather than all supported in tools */
fl_QuadDeviceSpec flash_devices[] = {DFU_FLASH_DEVICE};
#else

ADDED Line 30:
#endif

REMOVED line 86:
     result = fl_connectToDevice(&p_flash, flash_devices, 1);

ADDED Lines 92-100:
#ifdef QUAD_SPI_FLASH
    result = fl_connectToDevice(
        &p_qflash, flash_devices, sizeof(flash_devices) / sizeof(fl_QuadDeviceSpec)
    );
#else
    result = fl_connectToDevice(
        &p_flash, flash_devices, sizeof(flash_devices) / sizeof(fl_DeviceSpec)
    );
#endif
```

File #2 – sw_usb_audio/app_<specific application>/src/core/customdefines.h

The specific header file for the application (e.g. app_usb_aud_mic_array) needs to be changed to include the specification of new flash device and add this device into list of supported devices in the `#define DFU_FLASH_DEVICE` that is used in File #1 above.

By defining both flash device specifications in this file the firmware will operate correctly on both old and new parts.

35 app_usb_aud_mic_array/src/core/customdefines.h

```
@@ -82,4 +82,39 @@ void genclock();
 #define PRODUCT_STR_A1     "XMOS Microphone Array UAC1.0"
 //:

+#define FL_QUADDEVICE_ISSI_IS25LP016D \
+{ \
+    22,                     /* Enum value to identify the flashspec in a list */ \
+    256,                    /* page size */ \
+    8192,                   /* num pages */ \
+    3,                      /* address size */                                   \
+    3,                      /* log2 clock divider */                             \
+    0x9F,                   /* QSPI_RDID */ \
+    0,                      /* id dummy bytes */ \
+    3,                      /* id size in bytes */ \
+    0x9D6015,               /* device id */ \
+    0x20,                   /* QSPI_SE */ \
+    4096,                   /* Sector erase is always 4KB */ \
+    0x06,                   /* QSPI_WREN */ \
+    0x04,                   /* QSPI_WRDI */ \
+    PROT_TYPE_NONE,         /* no protection */ \
+    {{0,0},{0x00,0x00}},    /* QSPI_SP, QSPI_SU */ \
+    0x02,                   /* QSPI_PP */ \
+    0xEB,                   /* QSPI_READ_FAST */ \
+    1,                      /* 1 read dummy byte */ \
+    SECTOR_LAYOUT_REGULAR,  /* mad sectors */ \
+    {4096,{0,{0}}},        /* regular sector sizes */ \
+    0x05,                   /* QSPI_RDSR */ \
+    0x01,                   /* QSPI_WRSR */ \
+    0x01,                   /* QSPI_WIP_BIT_MASK */ \
+}
+
+// DFU_FLASH_DEVICE is a comma-separated list of flash spec structures
+// FL_QUADDEVICE_ISSI_IS25LQ016B: Deprecated integrated flash part
+// FL_QUADDEVICE_ISSI_IS25LP016D: New integrated flash part
+// This define is used in sc_usb_audio/module_usb_audio/flashlib_user.c
+#define DFU_FLASH_DEVICE \
+    FL_QUADDEVICE_ISSI_IS25LQ016B, \
+    FL_QUADDEVICE_ISSI_IS25LP016D,
+
 #endif
```

Note: It is not necessary to have an explicit flash spec for the IS25LQ016B part as this is already built into the XMOS tools.

Specific code changes to customdefines.h in the target application folder.

```
ADDED Lines 85-119:
#define FL_QUADDEVICE_ISSI_IS25LP016D \
{ \
    22,                     /* Enum value to identify the flashspec in a list */ \
    256,                    /* page size */ \
    8192,                   /* num pages */ \
    3,                      /* address size */                                   \
    3,                      /* log2 clock divider */                             \
    0x9F,                   /* QSPI_RDID */ \
    0,                      /* id dummy bytes */ \
    3,                      /* id size in bytes */ \
    0x9D6015,               /* device id */ \
    0x20,                   /* QSPI_SE */ \
    4096,                   /* Sector erase is always 4KB */ \
    0x06,                   /* QSPI_WREN */ \
    0x04,                   /* QSPI_WRDI */ \
    PROT_TYPE_NONE,         /* no protection */ \
    {{0,0},{0x00,0x00}},    /* QSPI_SP, QSPI_SU */ \
    0x02,                   /* QSPI_PP */ \
    0xEB,                   /* QSPI_READ_FAST */ \
    1,                      /* 1 read dummy byte */ \
    SECTOR_LAYOUT_REGULAR,  /* mad sectors */ \
    {4096,{0,{0}}},        /* regular sector sizes */ \
    0x05,                   /* QSPI_RDSR */ \
    0x01,                   /* QSPI_WRSR */ \
    0x01,                   /* QSPI_WIP_BIT_MASK */ \
}

// DFU_FLASH_DEVICE is a comma-separated list of flash spec structures
// FL_QUADDEVICE_ISSI_IS25LQ016B: Deprecated integrated flash part
// FL_QUADDEVICE_ISSI_IS25LP016D: New integrated flash part
// This define is used in sc_usb_audio/module_usb_audio/flashlib_user.c
#define DFU_FLASH_DEVICE \
    FL_QUADDEVICE_ISSI_IS25LQ016B, \
    FL_QUADDEVICE_ISSI_IS25LP016D,
```