AN11232

USB composite device on LPC11Uxx

Rev. 1 — 1 July 2012

Application note

| **Abstract** | This document describes how to create a USB composite device on NXP Cortex-M0 LPC11Uxx. The composite device may include two or three different (or same) interfaces among HID, MSC, CDC and DFU which are some of the most commonly used USB device classes. |

Revision history

Rev	Date	Description
1	20120701	Initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The LPC11Uxx parts (LPC11U1x, LPC11U2x and LPC11U3x) are available with a USB 2.0 Full-Speed device controller. Extensible on-chip USB drivers are available for LPC11U2x and LPC11U3x. Refer to the User Manual for details.

A Composite Device is defined in the *USB Specification Revision 2.0* as "A device that has multiple interfaces controlled independently of each other".

This document describes how to create a USB composite device on LPC11Uxx parts. For compatibility with all LPC11Uxx parts, the USB on-chip drivers are not used.

The commonly used USB device classes are: Human Interface Device (HID), Mass Storage Class (MSC), and Communication Device Class (CDC). If a user wants to update the firmware, a Device Firmware Upgrade (DFU) class can also be used.

The USB composite device examples created in this Application Note are shown below:

- HID+MSC
- HID+MSC
- MSC+CDC
- HID+MSC+CDC
- DFU+HID
- DFU+MSC
- DFU+MSC
- DFU+MSC+HID

All the examples are tested on a Keil MCB11U00 evaluation board (populated with LPC11U14/201, Rev. A).

2. USB composite device basics

2.1 Introduction

Normally, a USB device presents a single function to the host, such as a mouse, keyboard, serial RS-232 port or removable storage disk. One function occupies one USB cable and port; in order to use multiple functions at the same time, many USB cables and ports are needed. The composite device allows a multi-function device with only one cable and one port.

The USB Specification Revision 2.0 defines a composite device as "A device that has multiple interfaces controlled independently of each other". It has the following features:

- Multiple interfaces and functions
- Only a single device address

Note: A Compound Device can also allow multiple functions through a single USB port, but differs in that "multiple functions are combined with a hub in a single package". Please refer to the USB Specification Revision 2.0 for details.

2.2 Architecture

2.2.1 Communication flow

Fig 1 is the typical communication flow between function (Device side) and client software (Host side). Function is defined as "A USB device that provides a capability to the host" on USB Specification Revision 2.0.
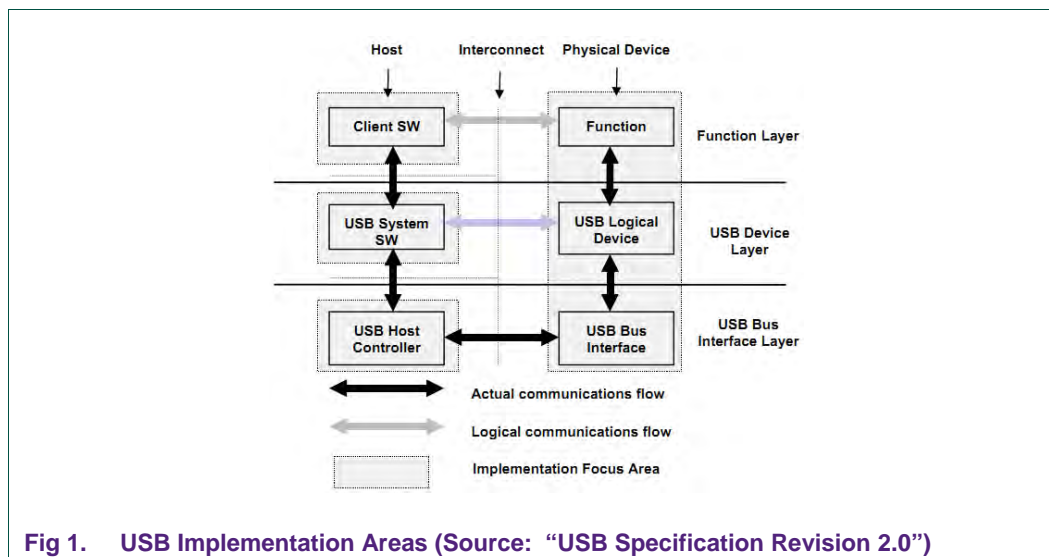

Fig 1.   USB Implementation Areas (Source: "USB Specification Revision 2.0")

Usually there is only one function for a USB device. However, a composite device contains several functions, each of which communicate with the corresponding Client SW (Device Driver) on the Host side. For example, for a composite device which contains HID and MSC, the Host will load a HID and MSC driver respectively on enumeration. A HID driver on the Host will talk to the HID function in the device and an MSC driver will talk to the MSC function in the device simultaneously.

2.2.2 Interfaces

A composite device contains several functions, and each function may have one (e.g. HID, or MSC) or more (e.g. CDC or audio) interfaces. Therefore, special care must be taken for the interface descriptor of a composite device.

For a composite device which contains a single-interface function only, e.g. HID+MSC, the interface descriptor is the concatenation of each function's interface descriptor.

For a composite device which contains multi-interface function, e.g. HID+CDC, an additional Interface Association Descriptor (IAD) should be used.

2.2.3 Endpoints

Beside the default control endpoint (EP0), the composite device should define a number of endpoints equal to the sum of the number of endpoints required for each individual function.

Fig 2 shows the interface and endpoint assignment of the composite device examples in this Application Note.

	HID+MSC	HID+CDC	MSC+CDC	HID+MSC+CDC	DFU+HID	DFU+MSC	DFU+CDC	DFU+HID+MSC
Interface 0	HID	HID	MSC	HID	DFU	DFU	DFU	DFU
Interface 1	MSC	CDC Control	CDC Control	CDC Control	HID	MSC	CDC Control	HID
Interface 2		CDC Data	CDC Data	CDC Data			CDC Data	MSC
Interface 3				MSC				
EP0	Default Control EP (also for DFU)							
EP1		CDC Interrupt IN	CDC Interrupt IN	CDC Interrupt IN			CDC Interrupt IN	
EP2	MSC Bulk IN & OUT		MSC Bulk IN & OUT	MSC Bulk IN & OUT		MSC Bulk IN & OUT		MSC Bulk IN & OUT
EP3		CDC Bulk IN & OUT	CDC Bulk IN & OUT	CDC Bulk IN & OUT			CDC Bulk IN & OUT	
EP4	HID Interrupt IN & OUT	HID Interrupt IN & OUT		HID Interrupt IN & OUT	HID Interrupt IN & OUT			HID Interrupt IN & OUT
EP5								

Fig 2. Interface and endpoint assignment (Note: for a composite device with DFU function, the interface assignment in this table is only for run-time mode)

2.3 Host drivers

Windows recognizes the device as a composite device if it meets the following requirements:

- The device class field of the device descriptor (**bDeviceClass**) must contain a value of zero, or the class (**bDeviceClass**), subclass (**bDeviceSubClass**), and protocol (**bDeviceProtocol**) fields of the device descriptor must have the values 0xEF, 0x02 and 0x01 respectively.
- The device must have multiple interfaces.
- The device must have a single configuration.

Windows first recognizes the device as a composite device, and then loads the device driver for each function.

For HID or MSC, Windows loads a generic HID driver (*hidclass.sys* and *hidusb.sys*) or MSC driver (*usbstor.sys*) respectively.

For the CDC, USB virtual COM-port in this example, Windows loads a *usbser.sys* driver. An **INF** file should be provided that contains the device's Vendor ID and Product ID. Windows will use this INF file to select a driver to load. The INF file for CDC in this Application Note is located in \USBClass\CDC\Driver.

For DFU, Windows doesn't provide a driver for this class. This driver should be provided by the user. Windows will also need an INF file to select a driver to load. The driver and INF file for DFU in this Application Note is located in \USBClass\DFU\Driver.

For more details about the USB driver support on Windows, please refer to http://msdn.microsoft.com/en-us/library/ff538820.aspx

3. Descriptors of USB composite device

3.1 Device descriptor

The device descriptor of a composite device can be divided into two categories:

- Composite device without multi-interface function, e.g. HID+MSC, HID+DFU.
- Composite device with multi-interface function, e.g. HID+CDC, MSC+CDC.

Fig 3 shows the device descriptor of a composite device without a multi-interface function.

```
                  /* USB Standard Device Descriptor */
           __align(4) const uint8_t USB_DeviceDescriptor[] = {
             USB_DEVICE_DESC_SIZE,             /* bLength */
             USB_DEVICE_DESCRIPTOR_TYPE,       /* bDescriptorType */
        #if LPM_SUPPORT
             WBVAL(0x0201), /* 2.01 */         /* bcdUSB */
        #else
             WBVAL(0x0200), /* 2.00 */         /* bcdUSB */
        #endif
             0x00,                             /* bDeviceClass */
             0x00,                             /* bDeviceSubClass */
             0x00,                             /* bDeviceProtocol */
             USB_MAX_PACKET0,                  /* bMaxPacketSize0 */
             WBVAL(0x1FC9),                    /* idVendor */
             WBVAL(0x000B),                    /* idProduct */
             WBVAL(0x0100), /* 1.00 */         /* bcdDevice */
             0x01,                             /* iManufacturer */
             0x02,                             /* iProduct */
             0x03,                             /* iSerialNumber */
             0x01                              /* bNumConfigurations */
           };
```

Fig 3.   Device descriptor of a composite device without multi-interface function

Fig 4 shows the device descriptor of a composite device with multi-interface function.

```
                  /* USB Standard Device Descriptor */
           __align(4) const uint8_t USB_DeviceDescriptor[] = {
             USB_DEVICE_DESC_SIZE,             /* bLength */
             USB_DEVICE_DESCRIPTOR_TYPE,       /* bDescriptorType */
        #if LPM_SUPPORT
             WBVAL(0x0201), /* 2.01 */         /* bcdUSB */
        #else
             WBVAL(0x0200), /* 2.00 */         /* bcdUSB */
        #endif
             USB_DEVICE_CLASS_MISCELLANEOUS,   /* bDeviceClass */
             0x02,                             /* bDeviceSubClass */
             0x01,                             /* bDeviceProtocol */
             USB_MAX_PACKET0,                  /* bMaxPacketSize0 */
             WBVAL(0x1FC9),                    /* idVendor */
             WBVAL(0x000B),                    /* idProduct */
             WBVAL(0x0100), /* 1.00 */         /* bcdDevice */
             0x01,                             /* iManufacturer */
             0x02,                             /* iProduct */
             0x03,                             /* iSerialNumber */
             0x01                              /* bNumConfigurations */
           };
```

Fig 4.   Device descriptor of a composite device with multi-interface function

bDeviceClass, **bDeviceSubClass** and **bDeviceProtocol** must be EFH, 02H and 01H.

3.2 Configuration descriptor

For composite devices with multi-interface functions, an IAD is necessary. Therefore, the length of an IAD should be added into the total length (**wTotalLength**).

Fig 5 shows the configuration descriptor of a composite device (HID+CDC) which must include an IAD.

AN11232

© NXP B.V. 2012. All rights reserved.

Application note

Rev. 1 — 1 July 2012

6 of 20

```
const uint8_t USB_ConfigDescriptor[] = {
/* Configuration 1 */
  USB_CONFIGUARTION_DESC_SIZE,        /* bLength */
  USB_CONFIGURATION_DESCRIPTOR_TYPE,  /* bDescriptorType */
  WBVAL(                              /* wTotalLength */
    1*USB_CONFIGUARTION_DESC_SIZE +
    1*USB_INTERFACE_DESC_SIZE      +
    HID_DESC_SIZE                  +
    2*USB_ENDPOINT_DESC_SIZE       +
    1*USB_INTERFACE_ASSOCIATION_DESC_SIZE +  /* interface association */
    1*USB_INTERFACE_DESC_SIZE      +  /* communication interface */
    0x0013                         +  /* CDC functions */
    1*USB_ENDPOINT_DESC_SIZE       +  /* interrupt endpoint */
    1*USB_INTERFACE_DESC_SIZE      +  /* data interface */
    2*USB_ENDPOINT_DESC_SIZE          /* bulk endpoints */

  ),
  0x03,                               /* bNumInterfaces */
  0x01,                               /* bConfigurationValue */
  0x00,                               /* iConfiguration */
#if REMOTE_WAKEUP_ENABLE             /* bmAttributes */
  USB_CONFIG_SELF_POWERED | USB_CONFIG_REMOTE_WAKEUP,
#else
  USB_CONFIG_SELF_POWERED,
#endif
  USB_CONFIG_POWER_MA(100),           /* bMaxPower */
```

Fig 5.   Configuration descriptor of a composite device (HID+CDC)

For composite devices with single-interface functions, the total length (**wTotalLength**) of the configuration descriptor is simply the sum of all the needed descriptors' length.

Fig 6 shows the configuration descriptor of a composite device (HID+MSC) which does not need the IAD.

```
const uint8_t USB_ConfigDescriptor[] = {
/* Configuration 1 */
  USB_CONFIGUARTION_DESC_SIZE,        /* bLength */
  USB_CONFIGURATION_DESCRIPTOR_TYPE,  /* bDescriptorType */
  WBVAL(                              /* wTotalLength */
    1*USB_CONFIGUARTION_DESC_SIZE +
    1*USB_INTERFACE_DESC_SIZE      +
    USB_DFU_DESCRIPTOR_SIZE        +
    1*USB_INTERFACE_DESC_SIZE      +
    HID_DESC_SIZE                  +
    2*USB_ENDPOINT_DESC_SIZE
  ),
  0x02,                               /* bNumInterfaces */
  0x01,                               /* bConfigurationValue */
  0x00,                               /* iConfiguration */
#if REMOTE_WAKEUP_ENABLE             /* bmAttributes */
  USB_CONFIG_SELF_POWERED | USB_CONFIG_REMOTE_WAKEUP,
#else
  USB_CONFIG_SELF_POWERED,
#endif
  USB_CONFIG_POWER_MA(100),           /* bMaxPower */
```

Fig 6.   Configuration descriptor of a composite device (HID+MSC)

For composite devices with DFU functions, the configuration descriptor in run-time mode and DFU mode is different. Two configurations must be defined: USB_ConfigDescriptor and USB_dfuConfigDescriptor.

USB_ConfigDescriptor is used in run-time mode and may contain HID+MSC+DFU functions.

USB_dfuConfigDescriptor is used in DFU mode and only contain DFU function.

For more details about DFU, please refer to http://www.usb.org/developers/devclass_docs/DFU_1.1.pdf

3.3 Interface descriptor

The interface descriptor for each function in a composite device is almost the same with their definitions in a single USB device except the actual interface index.

3.3.1 Interface Association Descriptor (IAD)

For composite device with multi-interface function, IAD is necessary.

Fig 7 shows the IAD in HID+CDC example.

```
                /* IAD to associate the two CDC interfaces */
      USB_INTERFACE_ASSOCIATION_DESC_SIZE,        /* bLength */
      USB_INTERFACE_ASSOCIATION_DESCRIPTOR_TYPE, /* bDescriptorType */
      USB_CDC_CIF_NUM,                    /* bFirstInterface */
      2,                                  /* bInterfaceCount */
      CDC_COMMUNICATION_INTERFACE_CLASS, /* bFunctionClass */
      CDC_ABSTRACT_CONTROL_MODEL,         /* bFunctionSubClass */
      0,                                  /* bFunctionProtocol */
      0x06,                               /* iFunction (Index of string
    /* Interface 1, Alternate Setting 0, Communication class interface
      USB_INTERFACE_DESC_SIZE,            /* bLength */
      USB_INTERFACE_DESCRIPTOR_TYPE,      /* bDescriptorType */
      USB_CDC_CIF_NUM,                    /* bInterfaceNumber: Number o:
```

Fig 7.   IAD in HID+CDC example

bDescriptorType is 0x11 to indicate an Interface Association Descriptor.

bFirstInterface is the Interface Number of the first interface that is associated with this function.

bInterfaceCount is the Number of contiguous interfaces that are associated with this function.

Note: only contiguously numbered interfaces can be associated. Also note that this IAD descriptor must be positioned just before the interfaces it references.

For more details about IAD, please refer to http://www.usb.org/developers/whitepapers/iadclasscode_r10.pdf

3.3.2 Interface descriptor of DFU

Fig 8 shows the DFU interface descriptor on run-time mode.

Offset	Field	Size	Value	Description
0	bLength	1	09h	Size of this descriptor, in bytes.
1	bDescriptorType	1	04h	INTERFACE descriptor type.
2	bInterfaceNumber	1	Number	Number of this interface.
3	bAlternateSetting	1	00h	Alternate setting. Must be zero.
4	bNumEndpoints	1	00h	Only the control pipe is used.
5	bInterfaceClass	1	FEh	Application Specific Class Code
6	bInterfaceSubClass	1	01h	Device Firmware Upgrade Code
7	bInterfaceProtocol	1	01h	Runtime protocol.
8	iInterface	1	Index	Index of string descriptor for this interface.

Fig 8. DFU interface descriptor on Run-time mode

Fig 9 shows the DFU interface descriptor on DFU mode.

Offset	Field	Size	Value	Description
0	bLength	1	09h	Size of this descriptor, in bytes.
1	bDescriptorType	1	04h	INTERFACE descriptor type.
2	bInterfaceNumber	1	00h	Number of this interface.
3	bAlternateSetting	1	Number	Alternate setting. *
4	bNumEndpoints	1	00h	Only the control pipe is used.
5	bInterfaceClass	1	FEh	Application Specific Class Code
6	bInterfaceSubClass	1	01h	Device Firmware Upgrade Code
7	bInterfaceProtocol	1	02h	DFU mode protocol.

Fig 9. DFU interface descriptor on DFU mode

3.4 Endpoint descriptor

All the endpoint descriptors are the same as their definitions in a single USB device.

Note: DFU function only use default control endpoint (EP0) to transfer data.

4. Composite device examples

The composite device examples include: HID+MSC, HID+CDC, MSC+CDC, HID+MSC+CDC, DFU+HID, DFU+CDC, DFU+MSC, DFU+HID+MSC.

All the examples on tested on Keil MCB11U00 evaluation board (LPC11U14/201, Rev. A). For more details about this board, please refer to http://www.keil.com/mcb1000/

For simplicity, only HID+MSC+CDC and DFU+HID+MSC are described here.

Note: To avoid the malfunction of USB devices with the same VID&PID (Windows loads the device driver based on the VID&PID), each application example here uses a different PID (VID stays unchanged).

AN11232
Application note

All information provided in this document is subject to legal disclaimers.

Rev. 1 — 1 July 2012

© NXP B.V. 2012. All rights reserved.

9 of 20

4.1 HID+MSC+CDC

The USB configuration for this example can be found in **usbcfg.h** as shown in Fig 10.
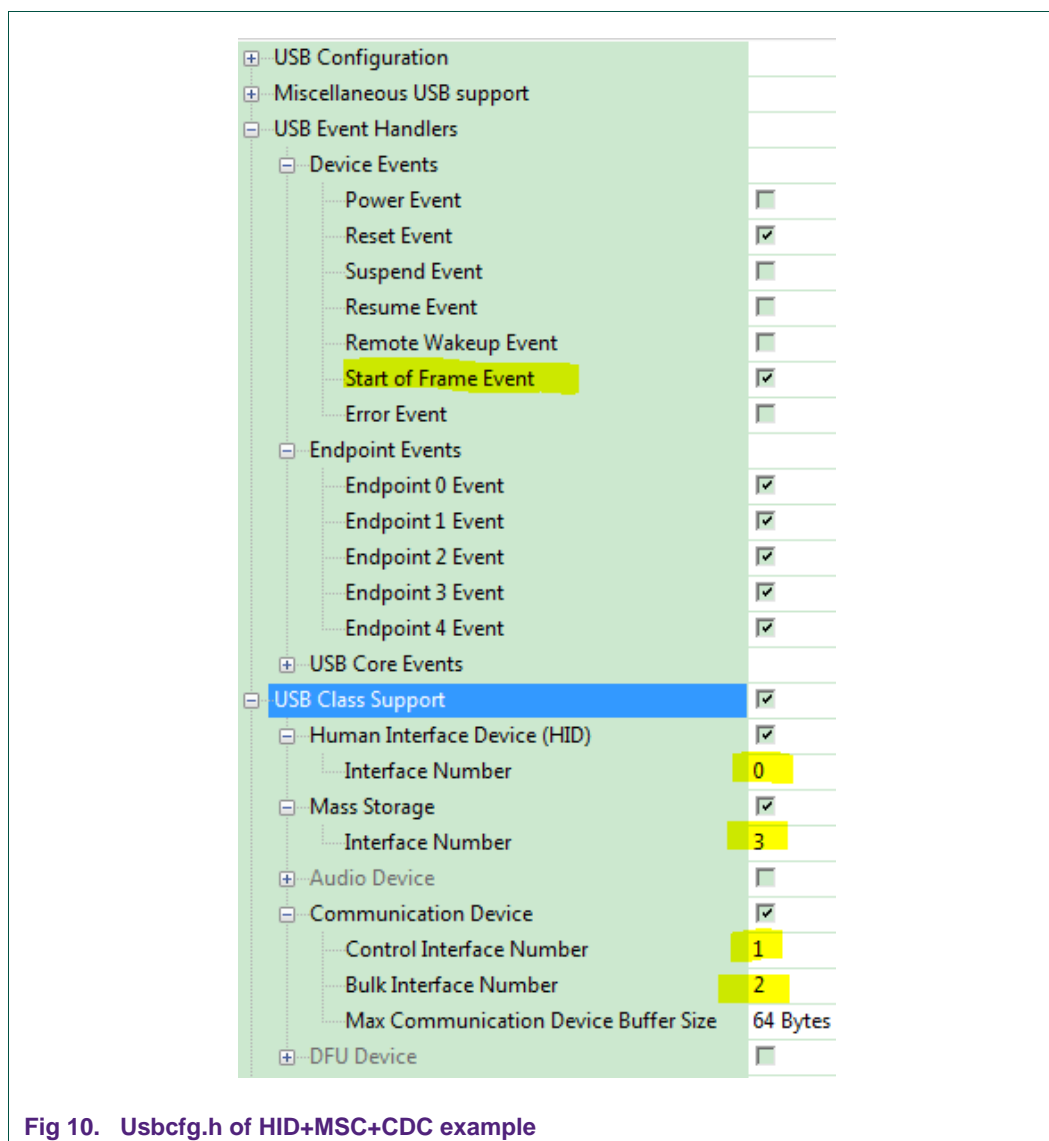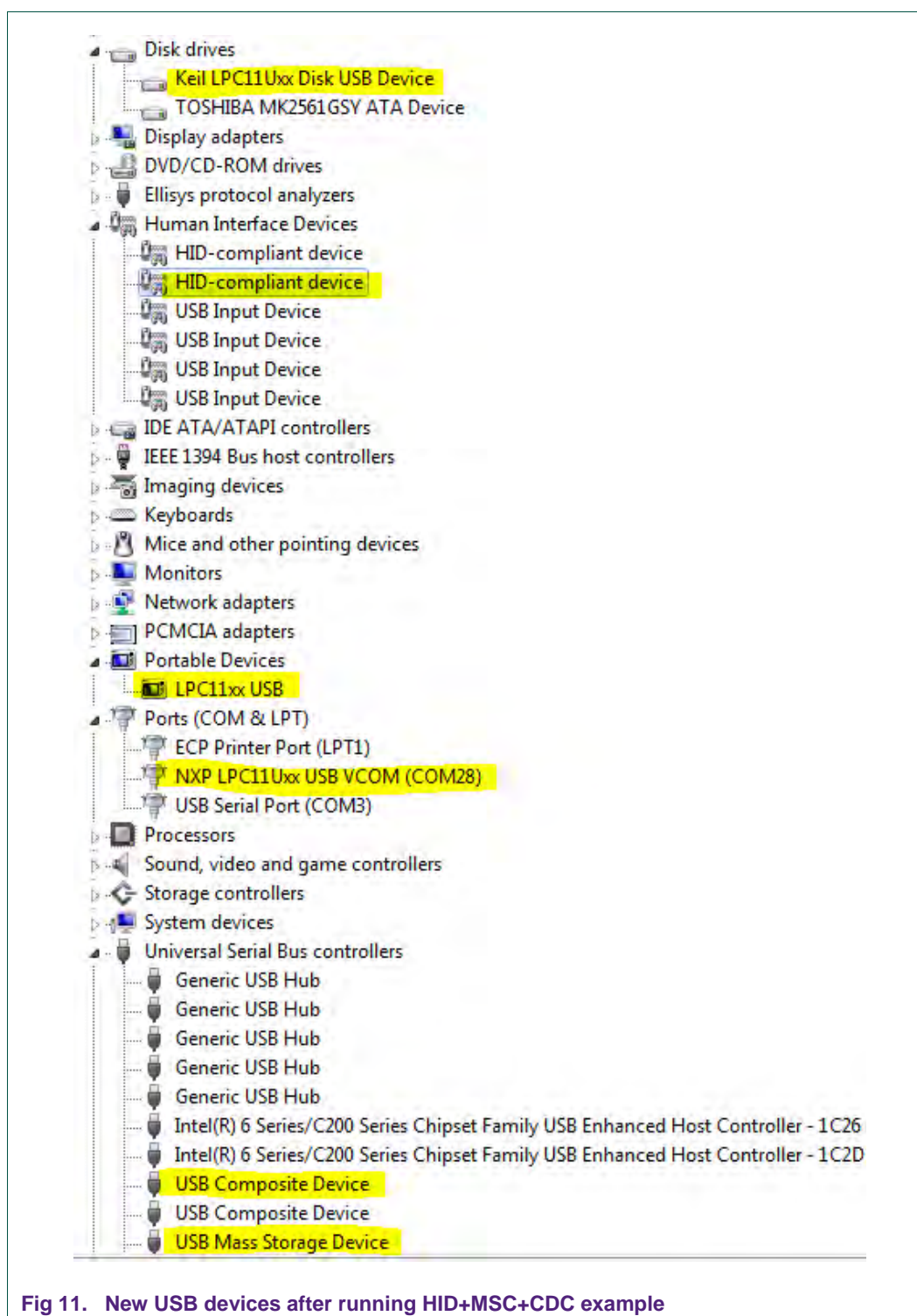

Fig 10. Usbcfg.h of HID+MSC+CDC example

The Start of Frame Event is necessary for CDC Virtual COM-port and should be enabled.

The implementation of a handler for each event and non-control endpoint can be found on **usbuser.c**.

After successful enumeration, Windows will request to install the driver for Virtual COM-port using an INF file which is located in \USBClass\CDC\Driver. After installation, the composite device will present three functions which can be viewed in the device manager. Windows system will load the generic HID and MSC driver automatically. Refer to Fig 11.

AN11232

© NXP B.V. 2012. All rights reserved.

Application note

Rev. 1 — 1 July 2012

10 of 20

Fig 11. New USB devices after running HID+MSC+CDC example

Next, check if these three devices work.

For HID, run the HID Client utility **HIDClient.exe** (\USBClass\HID\Utility\). Refer to Fig 12.

Fig 12. HID client utility (the HID device has been detected and selected)

The HID device works if:

- "**LPC11Uxx HID**" can be found and selected.
- Bit[1..0] of the Input Report can correctly reflect the status of Buttons S4 and S3 respectively on Keil MCB11U00 board.
- Bit[7..0] of the Output Report can correctly control the eight LEDs on the Keil MCB11U00 board.

The MSC device works if:

- A removable storage disk appeared with the label of "**LPC11Uxx USB**".
- There is one file (**README.TXT**) in this disk.

To verify the virtual COM-port functionality, you need to open two Tera Term programs. One is for the virtual COM-port, the other is for the local RS-232 port or USB-to-RS232 port. The virtual COM-port works if the content typed in one Tera Term windows can be echoed in another window. Refer to Fig 13.
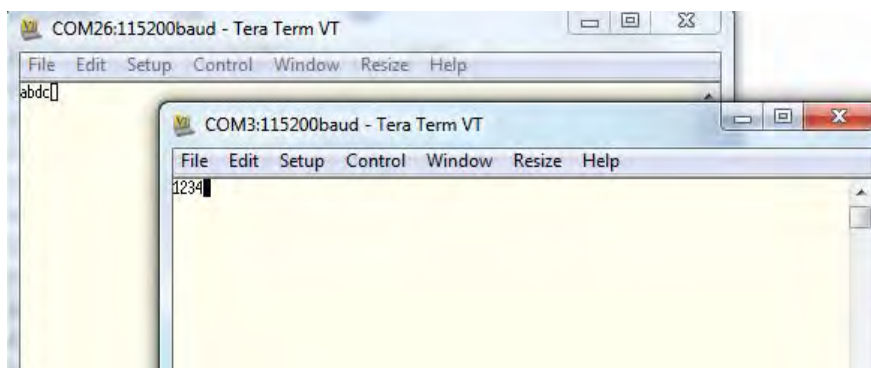
Fig 13. Verify virtual COM-port functionality (Note: The content typed in its own window is not displayed)

4.2 DFU+HID+MSC

The USB configuration for this example can be found in **usbcfg.h** as shown in Fig 14.

Fig 14. Usbcfg.h of DFU+HID+MSC example

Note: The interface index of the DFU device in the examples is set to 0.

After successful enumeration, Windows requests installation of the driver for the DFU device located in \USBClass\DFU\Driver. After installation, the composite device presents three functions which can be viewed in the device manager. Windows loads the generic HID and MSC driver automatically. Refer to Fig 15.

AN11232

© NXP B.V. 2012. All rights reserved.

Application note **Rev. 1 — 1 July 2012** **13 of 20**

Fig 15. New USB devices after running the DFU+HID+MSC example
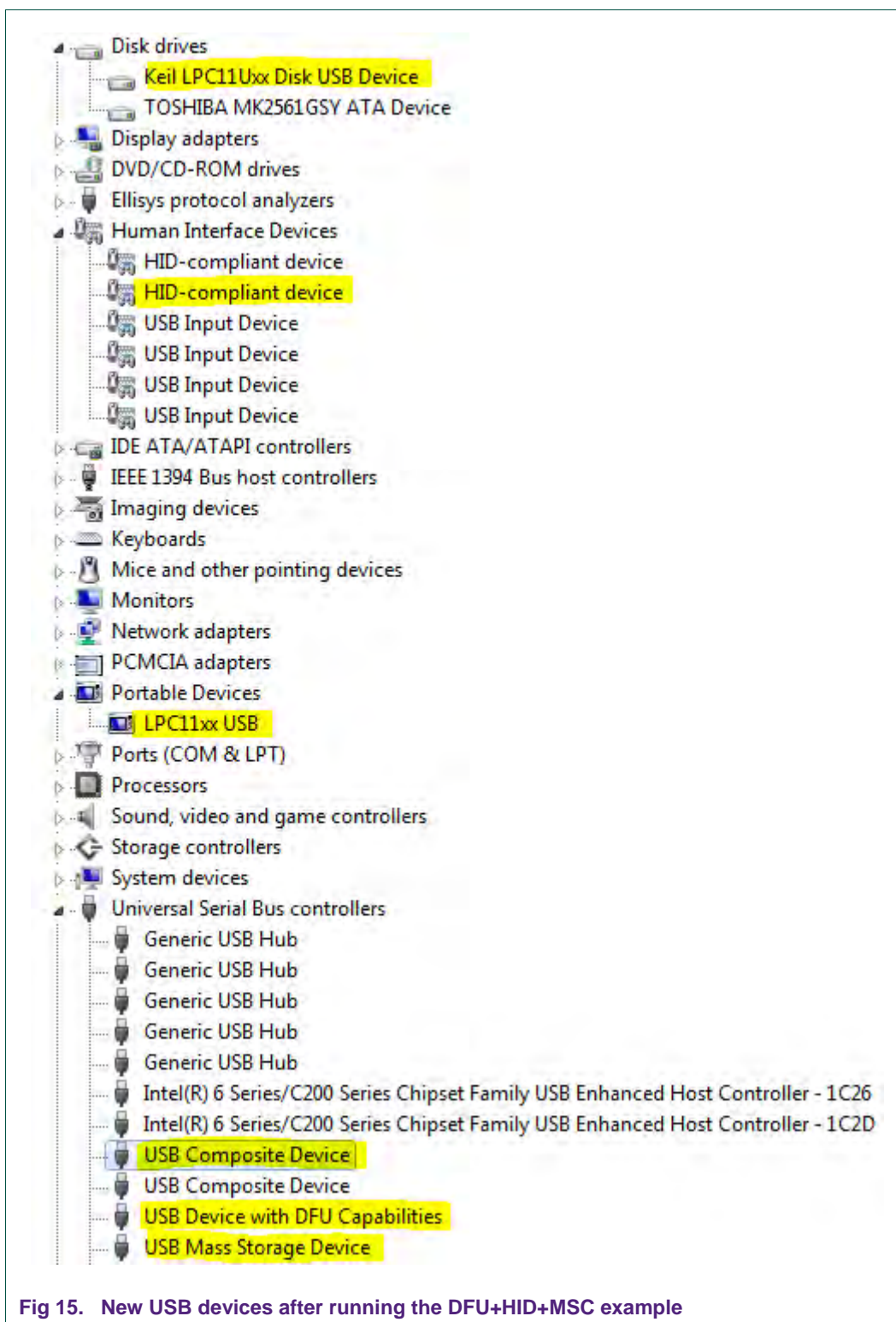
The verification of the HID and MSC device is the same with the HID+MSC+CDC example mentioned above.

For verification of a DFU device, run **DFUAPP.exe** (\USBClass\DFU\Utility\) and the correct information should be displayed. Refer to Fig 16.

AN11232

© NXP B.V. 2012. All rights reserved.

Application note **Rev. 1 — 1 July 2012** **14 of 20**
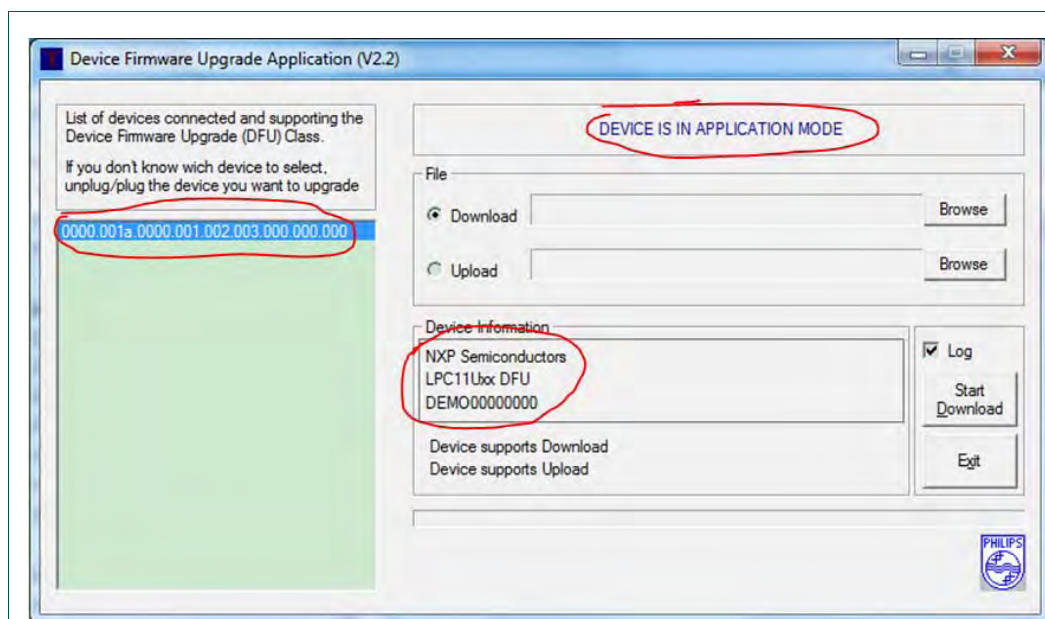
Fig 16. View of DFUAPP.EXE after it is opened

Next, select an image file and press "Start Download" to download to flash. The device switches from run-time mode to DFU mode and starts to transfer the data and program to internal flash on the specified address. Refer to Fig 17 and Fig 18.
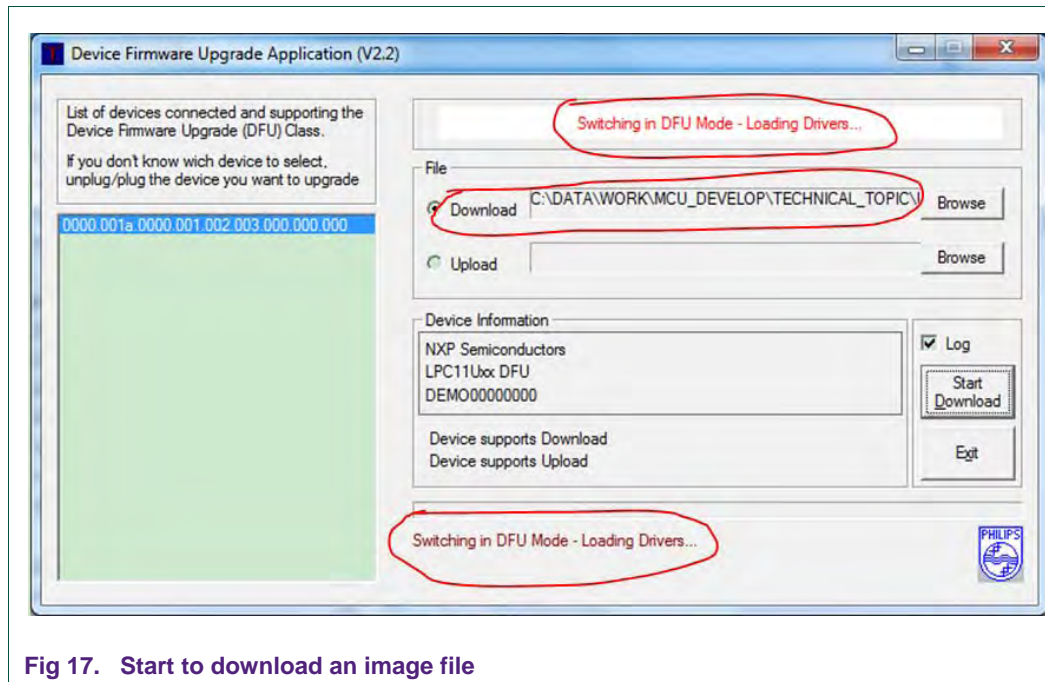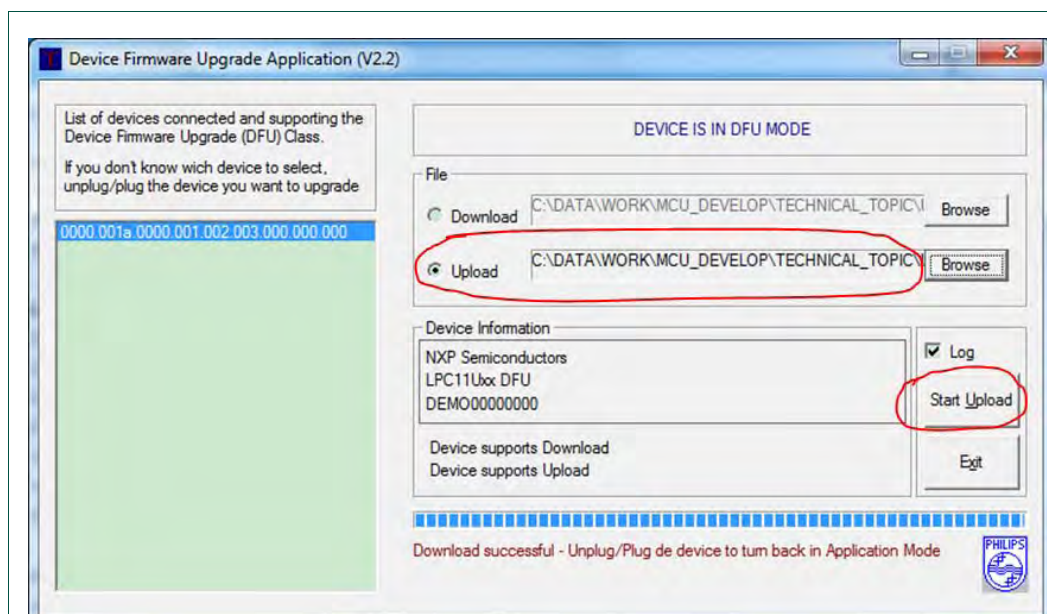

Fig 17. Start to download an image file
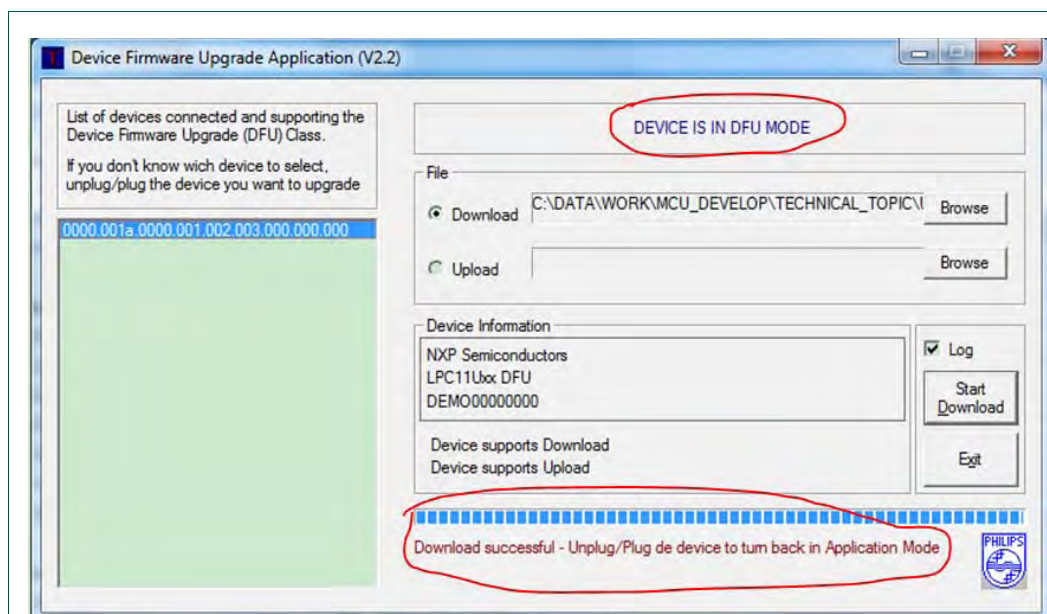
Fig 18.   Successful download

Next, upload the content at the specified address in internal flash to an image file. Select an image file and press the "Start to Upload" button. Refer to Fig 19 and Fig 20.

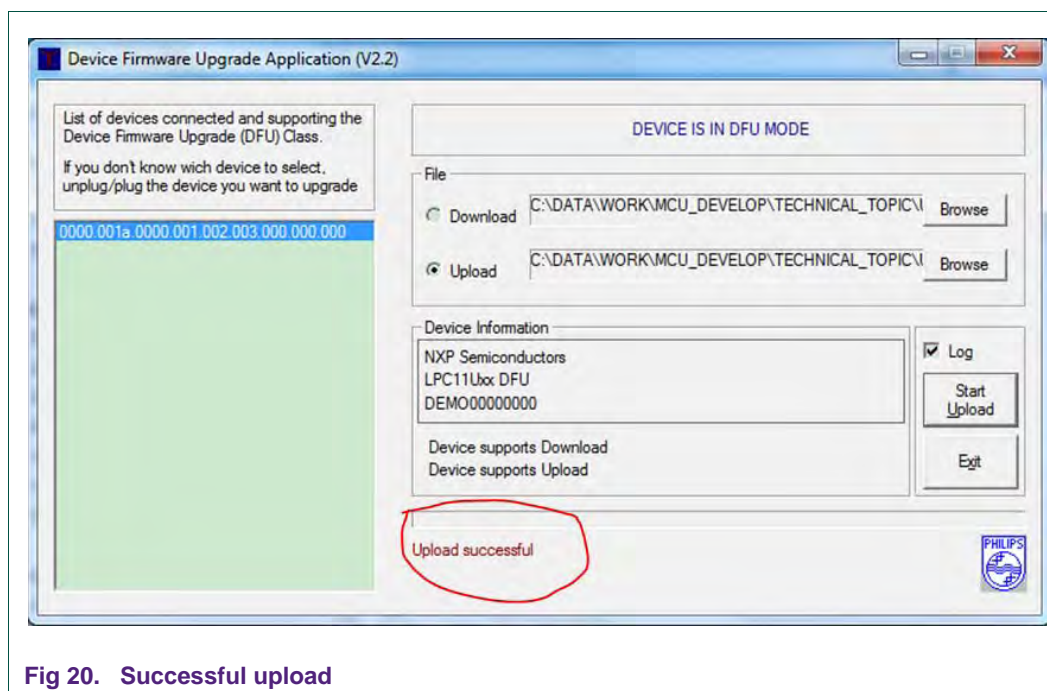**Fig 19.   Select an image file to upload**

Fig 20.   Successful upload

Note: the device will stay in DFU mode after download or upload operation. A reset or a power cycle is needed to make it return to run-time mode.

5. References

[1] LPC11Uxx User Manual UM10462 (Rev. 2.1), NXP Semiconductors, Jan. 13, 2012

[2] AN11018: USB composite device on the LPC134x, NXP Semiconductors, Dec. 17, 2010

[3] Universal Serial Bus Revision 2.0 specification

[4] Interface Association Descriptors (USB ENGINEERING CHANGE NOTICE)

[5] USB Interface Association Descriptor, Device Class Code and Use Model, Rev 1.0

[6] Device Class Definition for Human Interface Devices (HID), Version 1.11, June 27, 2001

[7] Universal Serial Bus Mass Storage Class Specification Overview, Revision 1.4, February 19, 2010

[8] Universal Serial Bus Mass Storage Class Bulk-Only Transport, Revision 1.0, September 31, 1999

[9] Communications Device Class, Revision 1.2

[10] Universal Serial Bus Device Class Specification for Device Firmware Upgrade, Version 1.1, Aug 5, 2004

AN11232

Application note **Rev. 1 — 1 July 2012** **18 of 20**

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP

Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. Contents