# AN2341

# USB4715 FlexConnect Operation

*Authors:* *Andrew Rogers and Connor Chilton*
*Microchip Technology Inc.*

## INTRODUCTION

The Microchip USB4715 smart hub allows any of the four FlexConnect-capable USB ports to assume the role of USB host at any time during hub operation. This host role exchange feature is called FlexConnect.

This functionality can be used in two primary ways:

- **Host Swapping:** This functionality can be achieved through a hub, where a host and a device can agree to swap the host or device relationship. The host becomes a device, and the device becomes a host.
- **Host Sharing:** A USB ecosystem can be shared between multiple hosts. Note that only one host may have access to the USB tree at a time.

FlexConnect can be enabled through any of the following methods:

- **Register Access:** An embedded SMBus master can control the state of the FlexConnect feature through basic write/read operations. This same register can be written to by a host during runtime to initiate FlexConnect.
- **USB Command:** FlexConnect can be initiated via a special USB command directed to the hub's internal Hub Feature Controller (HFC) device.
- **Direct Pin Control:** A select number of available GPIO pins on the hub in configuration 2 and 4 can be assigned the role of a FlexConnect control pin.

  **Note:** It is strongly recommended to view these 3 control mechanisms as mutually exclusive (i.e.: use only one of the methods: SMBus, USB Command, or Direct Pin control). If multiple FlexConnect control mechanisms are implemented with a single application, extra care must be taken to ensure that each controller is aware of the current status of FlexConnect and can be sure that the other controllers in the system coordinate such that there is no control contention.

By default, the USB4715 is configured to accommodate the automotive smart phone integration use-case. In this use-case, FlexConnect is initiated via USB command to the Hub Feature Controller (HFC), an embedded device inside the hub. FlexConnect is terminated automatically when the Flex host is disconnected. For other FlexConnect applications, certain features which make this application possible should be disabled by default. Additional guidance is provided in the respective sections for each control mechanism.

### Sections



### References

Consult the following documents for details on the specific parts referred to in this document:

- *USB4715 Data Sheet*
- *AN2651 Configuration of USB47xx and USB49xx*

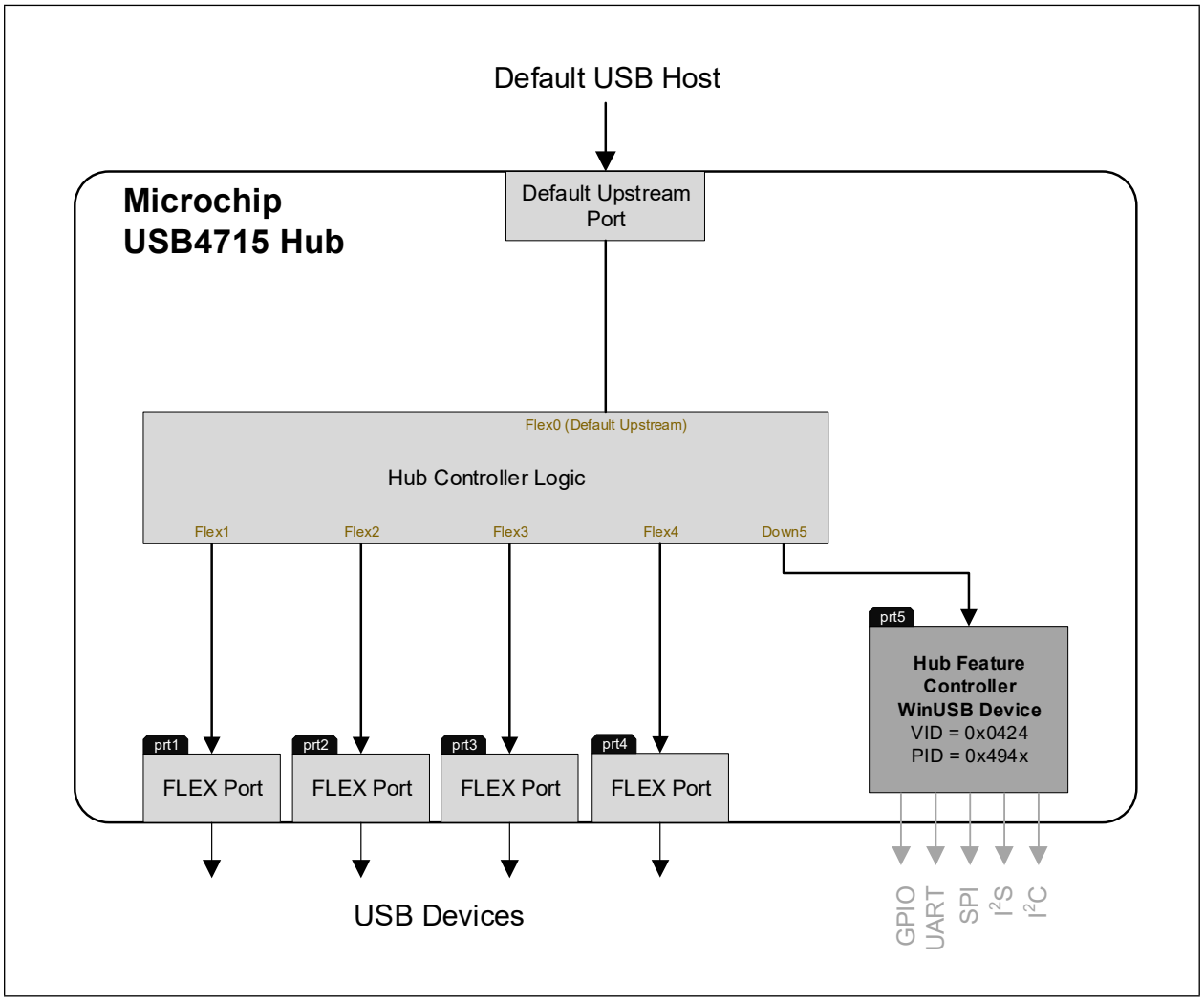
## FUNCTIONAL OVERVIEW

The internal block diagram of Microchip USB4715 is shown in Figure 1.

The USB4715 has an internal USB device (or the HFC), which enables the advanced features of the hub. The HFC device is a standard WinUSB class device with a Product ID (PID) of 0x494x (where “x” depends on the configuration mode selected via the CFG_STRAP[1:0] pins).

The HFC is permanently connected to the internal port 5 on the USB4715.This port is configured as non-removable.

**FIGURE 1: USB4715 INTERNAL BLOCK DIAGRAM**



## FlexConnect Initiation

FlexConnect can be initiated via SMBus control or a USB command to the internal HFC device. The SMBus and runtime register details are located in FlexConnect Register Control (SMBus Control) on page 6. The details of this USB command are shown in FlexConnect USB Command Details on page 11, and an example USB protocol trace capture of the command is detailed in FlexConnect USB Command Example on page 12.

When FlexConnect is initiated, the current host port changes to a device port, and the designated port changes into a host port. This can be used for Host Swapping or Host Sharing type applications.

## FlexConnect Termination

After initiating the FlexConnect mode, FlexConnect can be terminated in one of multiple ways. Different use cases call for different FlexConnect termination mechanisms. For system integrators controlling FlexConnect via any of the mechanisms SMBus, many of these features may be undesirable so instructions for disabling each mechanism are also included.

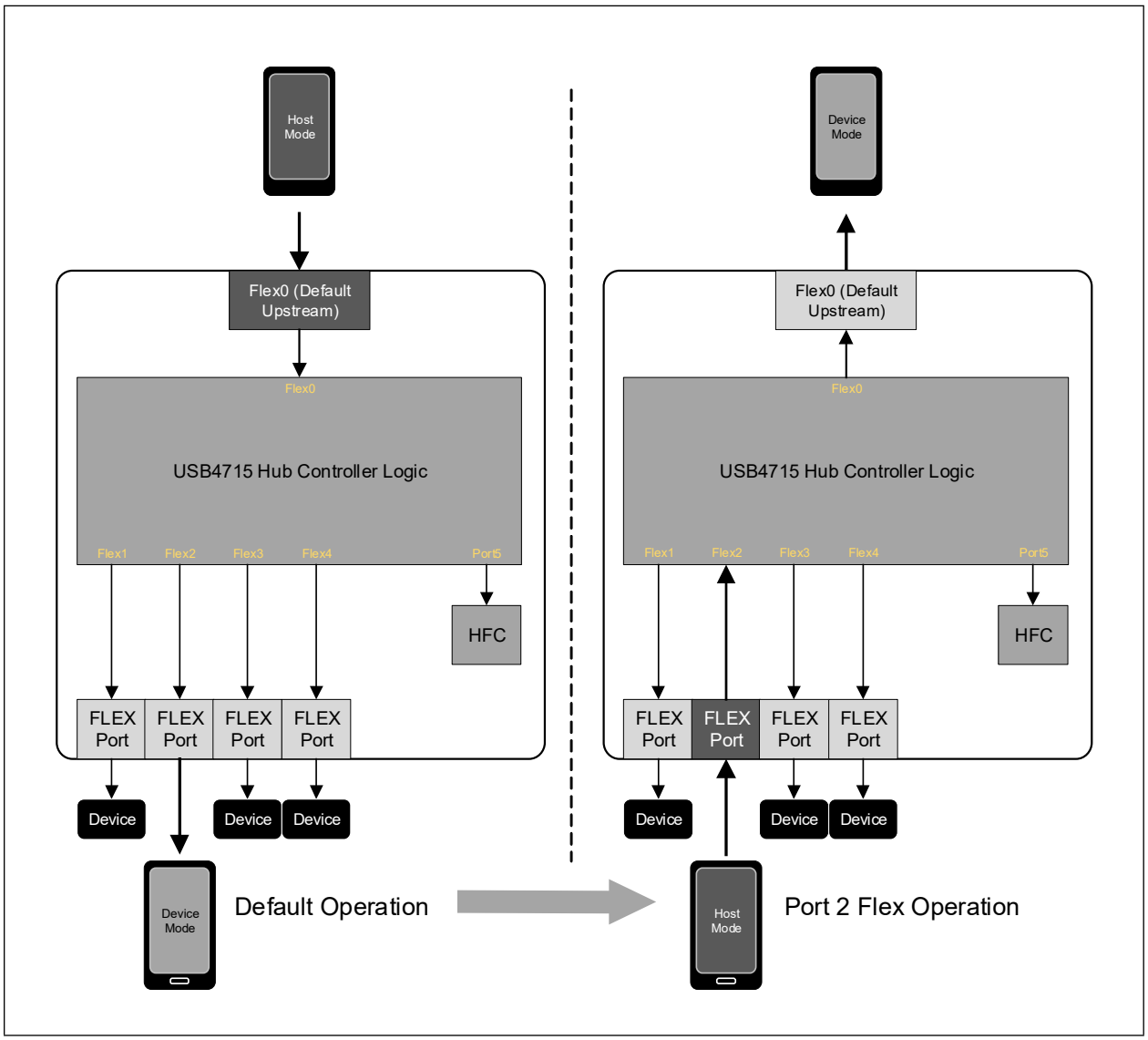
**TABLE 1: FLEXCONNECT TERMINATION (“UNFLEX”) MECHANISMS**

| “unFlex” Mechanism | Method to Disable |
|---|---|
| Flex Host sends a USB command to UnFlex. | To disable USB control, the Hub Feature Controller’s USB interface can be disabled. To do this, set MODE to 02h in Runtime Flags 2 Register (BFD2_341Bh = 02h) |
| FlexConnect GPIO is deasserted. | Do not configure/enable a FlexConnect control GPIO. |
| FlexConnect is terminated through SMBus register access. | Do not configure the hub hardware to enable the SMBus interface via the hardware configuration straps. |
| Flex Host is disconnected any time after the Flex host has enumerated the hub. | Clear bit ENABLE_AUTO_ROLE_REVERT in Runtime Flags 2 Register (i.e.: configuration such that BFD2_3402 bit[1] = 0b) |
| Hub not enumerated by Flex Host before ENUM_TIMEOUT expires. | Set ENUM_TIMEOUT = 000b. Note that ENUM_TIMEOUT is only used when FlexConnect is initiated via USB Command. |
| The VBUS_DET goes low. | To remove VBUS_DET dependency, an internal GPIO can be used to supply the VBUS_DET hardware block with a constant ‘high’ value. This is achieved by setting the following:<br>• Clear bit USB2_PASS_THRU (bit 0) in VBUS Pass Through Register<br>• Set bit GPIO_32_OE (bit 0) in PIO[63:32] OUTPUT ENABLE REGISTER<br>• Set bit GPIO_32_OS (bit 0) in PIO[63:32] OUTPUT REGISTER |

## Host Swapping

In a Host Swapping application, the host role is exchanged between two dual-role-capable devices. This host mode switching can be initiated through ID pin control, protocol handshake, or some other proprietary method. An example is shown in Figure 2.

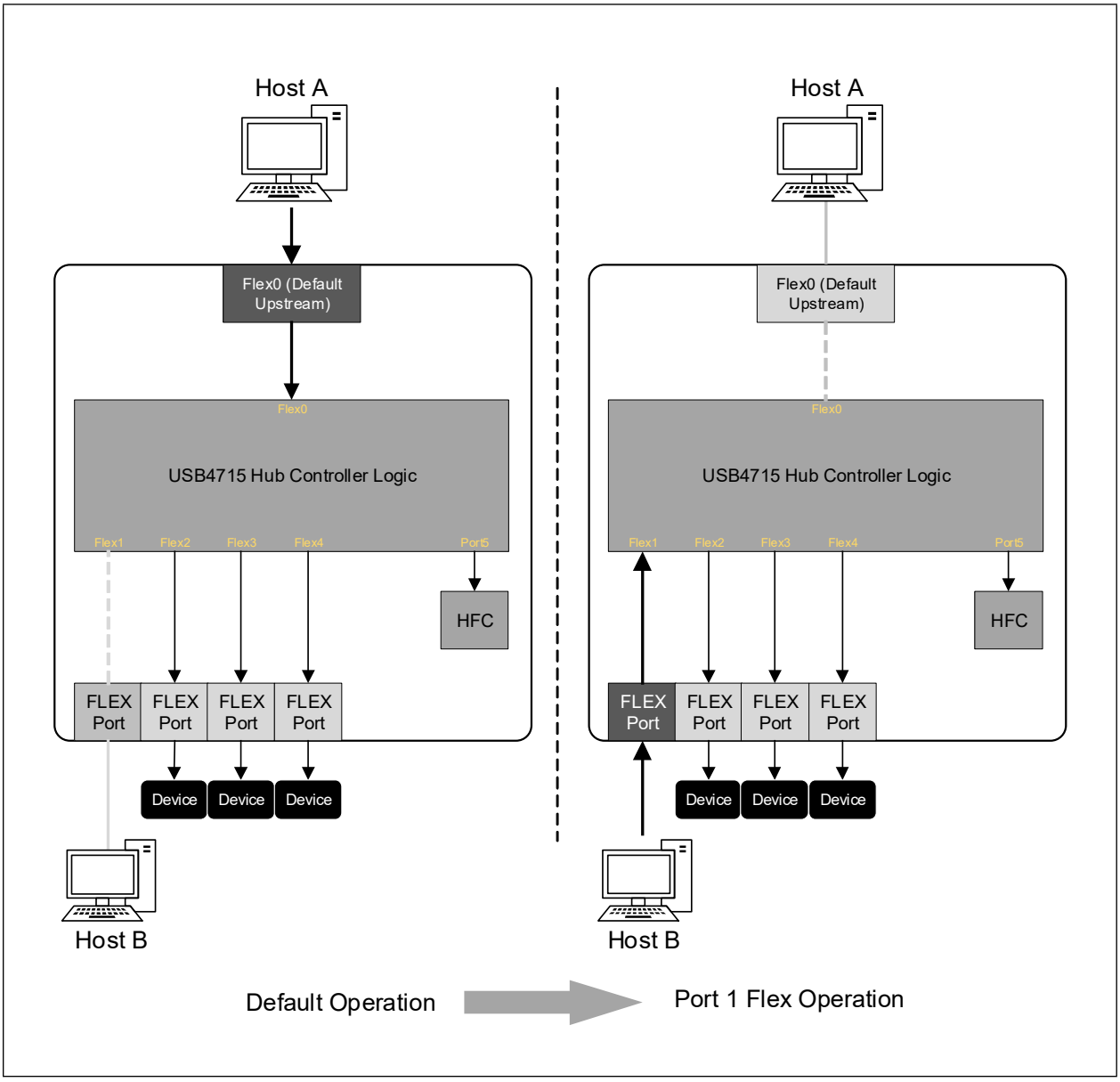
**FIGURE 2: FLEXCONNECT "HOST SWAPPING" EXAMPLE**



## Host Sharing

In a Host Sharing application, multiple hosts are connected to the hub, and only one of these hosts has access to the USB device tree at a time. All other hosts are effectively ‘disconnected’ from the USB tree when they do not control the host port. An example of Host Sharing functionality is shown in Figure 3.

**FIGURE 3: FLEXCONNECT “HOST SHARING” EXAMPLE**



## FLEXCONNECT REGISTER CONTROL (SMBUS CONTROL)

The FlexConnect feature can be controlled and configured via SMBus control. This method is useful for systems with an embedded microcontroller that can control the state of the FlexConnect feature via SMBus. In addition, this same register can be written to by a host during runtime to initiate FlexConnect.

**Note:** It is strongly recommended to avoid using SMBus control (of any kind) along with either the USB Command or GPIO control FlexConnect methods. When FlexConnect is initiated through any mechanism, the hub undergoes a soft reset. If SMBus control is enabled through the hardware pin straps, the hub will wait in the configuration stage (SOC_CFG) indefinitely for the SMBus controller to indicate that configuration is complete.

Additionally, by default the hub's SMBus interface will not be functional when the hub is placed into USB Suspend; the hub will stretch the SMBus CLK signal if access is attempted which will hold up the bus until the hub is brought out of USB Suspend. Typically, an SMBus host controller will not have any awareness of the USB bus' Suspend status, so many applications will need to force the hub's clocks to stay on during USB Suspend, ensuring that the SMBus interface is always available. Be aware that this will result in elevated power consumption during USB Suspend as the hub will not be able to power down certain hardware blocks. To enable that, set set bit 6 (BYPASS_MCU_SUSPEND) of the RUNTIME_FLAGS register (address BFD2_3500h).

To initiate FlexConnect, modify the registers defined in Table 2 accordingly.

**TABLE 2: FLEXCONNECT FEATURE CONTROL REGISTER 1**

| FLEX_FEATURE (BFD2_3440h) | | | FlexConnect Feature Control Register 1 |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:3 | Reserved | R | Reserved (must always be '0') |
| 2:0 | PORT | R/W | Physical downstream port to initiate or terminate the FlexConnect session<br>000b = No FlexConnect or Terminate FlexConnect session<br>001b = Downstream Port 1 is swapped with the upstream port.<br>010b = Downstream Port 2 is swapped with the upstream port.<br>011b = Downstream Port 3 is swapped with the upstream port.<br>100b = Downstream Port 4 is swapped with the upstream port.<br>All others = ***Invalid*** |

**TABLE 3: RUNTIME FLAGS 2 REGISTER**

| RUNTIME_FLAGS2 (BFD2_3402h) | | | Firmware Runtime Flags 2 |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:2 | Masked | R | Settings Not Related to Multi-Host Endpoint Reflector (do not change). |
| 1 | ENABLE_AUTO_ROLE_REVERT | R/W | 1b – While the ports are in role-switched mode, a disconnection detected on UDC2 forces an automatic role revert back to the default state.<br><br>0b – While the ports are in role-switched mode, only explicit commands issued from the USB host can revert the roles back to the default state. |

**TABLE 3: RUNTIME FLAGS 2 REGISTER**

| RUNTIME_FLAGS2 (BFD2_3402h) | | | Firmware Runtime Flags 2 |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 0 | DISCONNECT_DETECT_USBDATALINES | R/W | This bit is ignored if ENABLE_AUTO_ROLE_REVERT = 0.<br><br>0b – Hub will run a timer when the UDC1, UDC2, or both devices are placed into USB SUSPEND and will auto-detach both UDC1 (if ENABLE_UDC1_PERSISTENT = 0) and UDC2 when the timer expires. This timer duration is configured in the Suspend Timeout register.<br><br>1b – Hub will run a check on the USB data lines to determine if the UDC2 has been physically detached. This is performed by injecting a small current source on the D- pin and checking the line state. If the line state is detected as low during the check, UDC2 will remain attached. If the line state is detected as high during the check (indicating Hi-Z on the line), both UDC1 (if ENABLE_UDC1_PERSISTENT = 0) and UDC2 will auto-detach. |

**TABLE 4: RUNTIME FLAGS REGISTER**

| RUNTIME_FLAGS (BFD2_3500h) | | | Firmware Runtime Flags |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 31:24 | Reserved | R | Reserved |
| 23 | DISABLE_CDP_MULTIPLE_HANDSHAKE | R/W | 1b – Except the primary handshake in CDP mode, subsequent handshake by the downstream device while in CDP mode will be ignored by the firmware.<br><br>0b – Multiple handshake by the downstream device while in CDP mode will be acknowledged by the firmware until a time period of BC_CDP_MULTIPLE_HANDSHAKE_TIMEOUT expires after the first handshake in CDP mode. |
| 22 | ENABLE_QUADSPI | R/W | 0b – Quad SPI support is disabled in the firmware.<br><br>1b – Quad SPI support is enabled in the firmware. |
| 21 | DISABLE_I2CM_PULLUP_CHECK | R/W | 0b – Presence of I2C Pull up resistors will be checked prior to I2C Bridging & in any other operation involving the SoC as I2C controller.<br><br>1b – Presence of I2C Pull up resistors will NOT be checked prior to I2C Bridging & in any other operation involving the SoC as I2C controller. |
| 20 | Masked | R/W | Masked configuration setting. Do not change. |
| 19 | Masked | R/W | Masked configuration setting. Do not change. |
| 18 | ENABLE_USB491X_UDC1_PERSISTENT | R/W | 1b – Multi-Host Endpoint Reflector (UDC1 device) of USB4916/USB4914/USB4912 hub is enabled and exposed to the host be default. Also UDC1's upstream removal will be ignored even if ENABLE_AUTO_ROLE_REVERT=1.<br><br>0b – Multi-Host Endpoint Reflector (UDC1 device) device in USB4916/USB4914/USB4912 will be exposed to the host only when a role switch command is issued to the Hub Feature Controller (UDC0 device). |

**TABLE 4: RUNTIME FLAGS REGISTER**

| RUNTIME_FLAGS (BFD2_3500h) | | | Firmware Runtime Flags |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 17 | ENABLE_AUTO_ROLE_REVERT | R/W | 0b – While ports are in role switched mode (Flexed state in case of USB4715, MHER enabled in case of USB4916/USB4914/USB4912, Secondary hub enabled in case of USB4927/USB4925), only explicit commands from host will switch the role to default state.<br><br>1b – While ports are in role switched mode, following events will revert the role back to normal mode:<br>**USB4715:** Hub's upstream removal while in Flexed state.<br>**USB4916/4914:** UDC2's upstream removal or UDC1's upstream removal while MHER is enabled.<br>**USB4927/4925:** Secondary hub's upstream removal in role switched state. |
| 16 | DISCONNECT_DETECT_USBDATALINES | R/W | This bit is ignored if ENABLE_AUTO_ROLE_REVERT=0.<br><br>0b – Firmware will detect upstream disconnect based on suspend condition prevailing in the upstream for a minimum duration of SUSPEND_TIMEOUT.<br><br>1b – When set, firmware will detect upstream disconnect based on the state of D+/D- lines. |
| 15:14 | Reserved | R | Reserved |
| 13 | ENABLE_CDP_TO_SDP_RECOVERY | R/W | 1b – Toggle downstream VBUS when a downstream device doesn't enumerate in CDP mode.<br><br>0b – Do not toggle downstream VBUS when a downstream device doesn't enumerate in CDP mode.<br><br>**Note:** By default in ROM, the flag is 1 to ensure that downstream device enumeration is prioritized. This bit needs to be cleared to zero to run Battery charging compliance tests. |
| 12 | OTP_LOCK | R/W | When set, this is a soft lock of the OTP. |
| 11 | DISABLE_CDC_REMOTEWAKEUP_FEATURE | R/W | 0b – CDC Interface if enabled, reports as remote wakeup capable.<br><br>1b – CDC Interface if enabled, does not report as remote wakeup capable. |
| 10 | DISABLE_125K_PU | R/W | 0b – 125K Pull up resistors are enabled in China mode battery charging.<br><br>1b – 125K Pull up resistors are disabled in China mode battery charging. |
| 9 | ENABLE_BC_UNIVERSAL | R/W | 0b – Disable BC 1.2 compliant changes to Universal BC algorithm.<br><br>1b – Enable BC 1.2 compliant changes to Universal BC algorithm regardless of BC12_DCP bit in Runtime BC Flags (0x413X).<br><br>If this is set to 1 then the HEARTBEAT_UNIT will be 5ms.<br>If this is set to 0 then the HEARTBEAT_UNIT will be 10ms. |
| 8 | Reserved | R | Reserved |

**TABLE 4: RUNTIME FLAGS REGISTER**

| RUNTIME_FLAGS (BFD2_3500h) | | | Firmware Runtime Flags |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | ENABLE_SPI_BYTE_-FLASH | R/W | 0b – Generic SPI Flash commands are issued for SPI Flash programming.<br><br>1b – Microchip Byte Flash commands are issued for SPI flash programming. |
| 6 | BYPASS_MCU_SUSPEND | R/W | 0b – Default UDC suspend handling.<br><br>1b – MCU will not handle UDC suspend and will ignore the same. Note that if there's a suspend hook function present, that will be invoked still. |
| 5 | TARGET_OTP | R/W | Set Target OTP:<br><br>0b – Any command targeted to OTP (OTP Read, OTP programming, etc.) is re-directed to the pseudo OTP in SPI flash physical address 0x140000 to 0x41FFF.<br><br>1b – Any command targeted to OTP (OTP Read, OTP programming, etc.) is directed to the OTP. |
| 4 | CFG_FROM_SPI | R/W | Load Config from SPI:<br><br>0b – Configuration will not be loaded from pseudo OTP in SPI.<br><br>1b – Configuration loaded from pseudo OTP in SPI. |
| 3 | CFG_FROM_OTP | R/W | Load Config from OTP:<br><br>0b – Configuration will not be loaded from OTP.<br><br>1b – Firmware will load configuration from OTP. |
| 2 | HUB_CFG_CLK | R/W | Enable Hubcfg Interface power down:<br><br>0b – Clock to hub configuration registers is left on.<br><br>1b – Clock to hub configuration registers is turned off after Hub attach to USB. |
| 1 | Reserved | R | Reserved |
| 0 | DISABLE_BC | R/W | Disable Battery Charging:<br><br>0b – Battery charging logic is left enabled based on other flags.<br><br>1b – Battery charging logic is completely disabled. |

**TABLE 5: OTP UDC ENUMERATION**

| OTP_UDC_ENABLE (BFD2_341Bh) | | | OTP UDC Enumeration |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:0 | MODE | R/W | Control whether the “Hub Feature Controllers” USB Device Controller (UDC) interface is enabled or disabled.<br><br>00h: No change from default ROM behavior (Note: “Hub Feature Controller” UDC is enabled by default in ROM FW)<br>01h: Enable “Hub Feature Controller” UDC enumeration<br>02h: Disable “Hub Feature Controller” UDC enumeration<br>All other values are reserved. |

**TABLE 6: VBUS PASS THROUGH REGISTER**

| VBUS_PASS_THRU (BF80_3C40h) | | | VBUS Pass Through Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:1 | Reserved | R | Always ‘0b’ |
| 0 | USB2_PASS_THRU | R/W | 0 - The VBUS to USB2 Hub comes from internally connected PIO32<br>1 - The VBUS to USB2 Hub comes from the device pin |

**TABLE 7: PIO[63:32] OUTPUT ENABLE REGISTER**

| PIO63_OEN (BF80_0904h to BF80_0907h) | | | PIO[63:32] Output Enable Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 31:1 | Masked | R | Masked - used as unrelated internally connected GPIO functions |
| 0 | GPIO_32_OE | R/W | Set bit to enable GPIO32 as an output. |

**TABLE 8: PIO[63:32] OUTPUT REGISTER**

| PIO63_OUT (BF80_0924h to BF80_0927h) | | | PIO[63:32] Output Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 31:1 | Masked | R | Masked - used as unrelated internally connected GPIO functions |
| 0 | GPIO_32_OS | R/W | Set bit to drive GPIO32 output high. Clear bit to drive GPIO32 low |

## FLEXCONNECT USB COMMAND DETAILS

A special USB command can be issued from the USB host to control the FlexConnect feature.

After sending the special USB command to initiate the FlexConnect mode, the hub automatically performs ‘unflex’ and returns to the default Port 0 Host state if one of the following conditions is met:

- The new Flex host sends a USB command that commands the hub to return to the default Unflexed state.
- The new Flex host is disconnected any time after the Flex host has enumerated the hub.
- The hub has not yet been enumerated by the Flex host after the ENUM_TIMEOUT expires.
- The VBUS_DET on either the old upstream port or the new upstream port goes low.

The USB command is a NO DATA control transfer that must be issued to Endpoint 0 of the internal HFC device. On USB4715, the HFC is the internal device located on port 6. The SETUP command format is shown in Table 9 and Table 10.

**TABLE 9: MULTI-HOST BRIDGE SETUP PACKET**

| Setup Parameter | Value | Description |
|---|---|---|
| bmRequestType | 0x41 | Device-to-host, vendor class, targeted to interface |
| bRequest | 0x90 | SET_ROLE_SWITCH |
| wValue | 0xYYYY | Bits detailed in Table 10 |
| wIndex | 0x0000 | Reserved |
| wLength | 00 | No data |

**TABLE 10: WVALUE DETAIL OF FLEXCONNECT COMMAND SETUP PACKET**

| Bit | Name | Description |
|---|---|---|
| 15:11 | Reserved | Reserved (must always be ‘0’) |
| 10:8 | ENUM_TIMEOUT | These bits control a timer that is started after FlexConnect is initiated.<br><br>If the Flex host does not enumerate the hub before the timer expires, the hub will revert to the Unflexed state.<br><br>If the Flex host enumerates the hub within the time defined, then the hub will remain in the Flexed state.<br><br>000 = No timeout defined (hub will never automatically ‘unflex’)<br>001 = 10 ms<br>010 = 100 ms<br>011 = 500 ms<br>100 = 1 second<br>101 = 5 seconds<br>110 = 10 seconds<br>111 = 20 seconds |
| 7 | Reserved | Reserved (must always be ‘0’) |
| 6:5 | ROLE_SWITCH_TYPE | Must always write ‘01’ for FlexConnect |
| 4 | ROLE_SWITCH_STATE | 1 = Transitions the port defined in PORT bits to the FlexConnect mode<br>0 = Exits the FlexConnect mode currently active on port defined in PORT bits |
| 3:0 | PORT | Physical downstream port to initiate or terminate Multi-Host Bridging session<br>0001 = Port 1<br>0010 = Port 2<br>0011 = Port 3<br>0100 = Port 4<br>All others = ***Invalid*** |

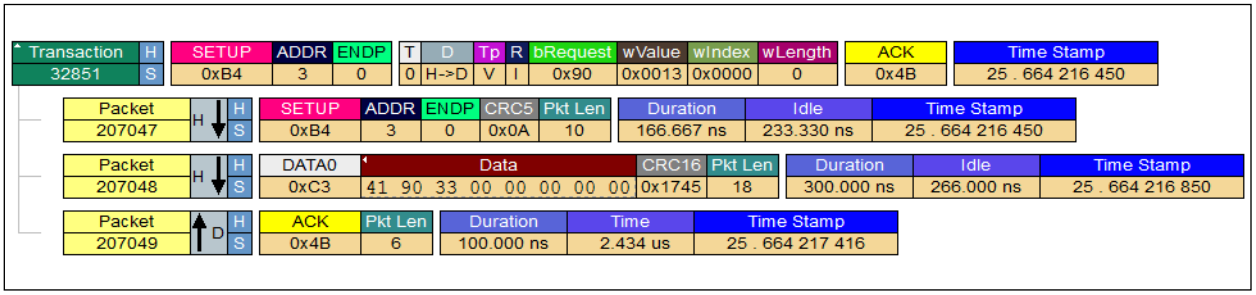
## FlexConnect USB Command Example

An example of a FlexConnect initialization command for port 3 is shown in Table 11. This command is sent to Endpoint 0 of the HFC.

**TABLE 11: FLEXCONNECT SETUP COMMAND EXAMPLE**

| Setup Parameter | Value | Note |
|---|---|---|
| bmRequestType | 0x41 | Device-to-host, vendor class, targeted to interface |
| bRequest | 0x90 | SET_ROLE_SWITCH |
| wValue | 0x0033 | Bits 15:11 = 0000b<br>Bits 10:8 = 000b (no enum timer)<br>Bit 7 = 0b<br>Bits 6:5 = 01b<br>Bit 4 = 1b (enter the FlexConnect mode)<br>Bits 3:0 = 0011b (Port 3) |
| wIndex | 0x0000 | Reserved |
| wLength | 00 | No data |

**FIGURE 4: REGISTER READ SETUP TRANSACTION EXAMPLE**



## FLEXCONNECT PIN CONTROL

The FlexConnect feature can be initiated via direct pin control for systems that have an embedded MCU, which controls the state of FlexConnect. A select number of available PROG_FUNCx/GPIO pins in configuration modes 2 and 4 may be assigned the role of FLEX_IN_PORTx.

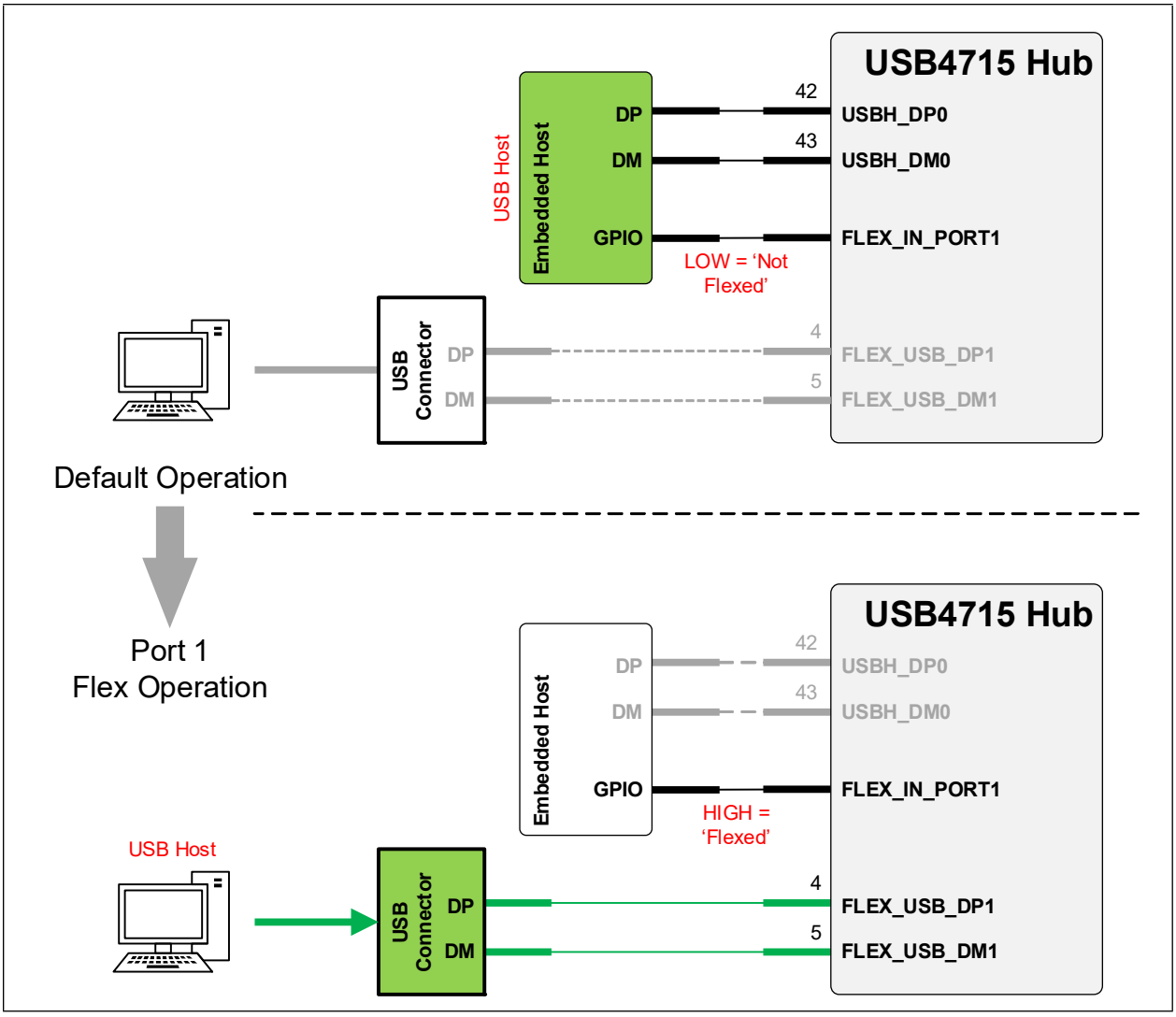
**Note:** It is strongly recommended to avoid using GPIO FlexConnect control along with either the USB Command or SMBus FlexConnect methods. When FlexConnect is initiated through any mechanism, the hub undergoes a soft reset. If SMBus control is enabled through the hardware pin straps, the hub will wait in the configuration stage (SOC_CFG) indefinitely for the SMBus controller to indicate that configuration is complete and that the hub should enter normal runtime operation. Using GPIO based FlexConnect along with SMBus control of the hub requires that the controller that controls the FlexConnect GPIO remain in sync with the SMBus controller so that the hub does not get stuck in the SMBus configuration stage each time the FlexConnect state is changed.

If using the direct pin control method, each port that utilizes the FlexConnect feature must also be assigned its own FLEX_IN_PORTx signal.

The FLEX_IN_PORTx configuration setting must be programmed into the hub's OTP configuration memory (if running from the hub's default internal ROM-based firmware image) or into 'pseudo-OTP' configuration memory located on an external SPI Flash device (if running the hub firmware from an optional external SPI-flash device).

An example of how an embedded USB host may use the direct pin control method is shown in Figure 5. .

**FIGURE 5: FLEXCONNECT PIN CONTROL EXAMPLE**



By default, FLEX_IN_PORT[1:4] pins are not assigned and hence not active. Only ports that are to be controlled via direct pin control must have a PROG_FUNC/GPIO pin assigned. Care must be taken to ensure that the assigned pin does not already have a critical pin function assigned to it.

Once one or more FLEX_IN_PORT[1:4] pins have been assigned, the pins behave as shown in Table 12. Note that only one FLEX_IN_PORT[1:4] should be asserted at once.

**TABLE 12: EXAMPLE FLEXCONNECT STATE TRUTH TABLE**

| FLEX_IN_PORT4 | FLEX_IN_PORT3 | FLEX_IN_PORT2 | FLEX_IN_PORT1 | FlexConnect State |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Port 0 = Host Port |
| 0 | 0 | 0 | **1** | Port 1 = Host Port |
| 0 | 0 | **1** | 0 | Port 2 = Host Port |
| 0 | **1** | 0 | 0 | Port 3 = Host Port |
| **1** | 0 | 0 | 0 | Port 4 = Host Port |
| All Others | | | | INVALID STATE |

**Note:** The SMBus and USB Command methods both have specific conditions where the hub will automatically perform ‘unflex’ and return to the default Port 0 Host state. When using the direct pin control method, the hub will always follow the state of the pins and will never attempt to automatically perform ‘unflex’.

The pin assignment registers are detailed in Table 13 to Table 16. These can be modified via the hub’s internal OTP memory, or through the SMBus interface during hub start-up configuration. These pins cannot be modified via runtime register writes.

**TABLE 13: PORT 1 FLEXCONNECT TRIGGER PIO CONFIGURATION REGISTER**

| FLEX_IN_PORT1 (BFD2_3442h) | | | FlexConnect Trigger PIO Configuration Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | FLEX_IN_EN | R/W | 1 - Enables FlexConnect trigger for port 1 through GPIO control. The specific GPIO pin is selected in bits [2:0] of this register.<br><br>The FlexConnect state is entered upon the rising edge of the selected GPIO. The FlexConnect state is exited upon the falling edge of the selected GPIO.<br><br>0 - Disables FlexConnect pin trigger for port 1. All other bits in this register are ignored. |
| 6:3 | Reserved | R | Reserved |
| 2:0 | FLEX_IN_IO | R/W | Selects the GPIO to be used for the Port 1 FlexConnect trigger.<br><br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 14: PORT 2 FLEXCONNECT TRIGGER PIO CONFIGURATION REGISTER**

| FLEX_IN_PORT2 (BFD2_3443h) | | | FlexConnect Trigger PIO Configuration Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | FLEX_IN_EN | R/W | 1 - Enables FlexConnect trigger for port 2 through GPIO control. The specific GPIO pin is selected in bits [2:0] of this register.<br><br>The FlexConnect state is entered upon the rising edge of the selected GPIO. The FlexConnect state is exited upon the falling edge of the selected GPIO.<br><br>0 - Disables FlexConnect pin trigger for port 2. All other bits in this register are ignored. |
| 6:3 | Reserved | R | Reserved |
| 2:0 | FLEX_IN_IO | R/W | Selects the GPIO to be used for the Port 2 FlexConnect trigger.<br><br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 15: PORT 3 FLEXCONNECT TRIGGER PIO CONFIGURATION REGISTER**

| FLEX_IN_PORT3 (BFD2_3444h) | | | FlexConnect Trigger PIO Configuration Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | FLEX_IN_EN | R/W | 1 - Enables FlexConnect trigger for port 3 through GPIO control. The specific GPIO pin is selected in bits [2:0] of this register.<br><br>The FlexConnect state is entered upon the rising edge of the selected GPIO. The FlexConnect state is exited upon the falling edge of the selected GPIO.<br><br>0 - Disables FlexConnect pin trigger for port 3. All other bits in this register are ignored. |
| 6:3 | Reserved | R | Reserved |
| 2:0 | FLEX_IN_IO | R/W | Selects the GPIO to be used for the Port 3 FlexConnect trigger.<br><br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 16: PORT 4 FLEXCONNECT TRIGGER PIO CONFIGURATION REGISTER**

| FLEX_IN_PORT4 (BFD2_3445h) | | | FlexConnect Trigger PIO Configuration Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | FLEX_IN_EN | R/W | 1 - Enables FlexConnect trigger for port 4 through GPIO control. The specific GPIO pin is selected in bits [2:0] of this register.<br><br>The FlexConnect state is entered upon the rising edge of the selected GPIO. The FlexConnect state is exited upon the falling edge of the selected GPIO.<br><br>0 - Disables FlexConnect pin trigger for port 4. All other bits in this register are ignored. |
| 6:3 | Reserved | R | Reserved |
| 2:0 | FLEX_IN_IO | R/W | Selects the GPIO to be used for the Port 4 FlexConnect trigger.<br><br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

## FLEX_CONFIGURATION SPACE

The Flex_Configuration (FLEXCFG) space is a space within the hub that allows programming of registers. The features defined in these registers will only be implemented once the hub flexed. Example features include port enable/disable, battery charging configuration, and more. The FLEXCFG space follows a similar structural format as One-Time Programmable (OTP). Table 17, Table 18, and Table 19 show the registers associated with FLEXCFG.

**TABLE 17: FLEX CONFIGURATION CONTROL REGISTER**

| FLEXCFG_CT (BFD2_345Bh) | | | Flex Configuration Control Register |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:1 | Reserved | R | Reserved |
| 0 | APPLY_FLEXCFG | R/W | 1 - FLEXCFG configuration of length specified in the FLEXCFG_LEN register will be applied during the Flex state transition.<br><br>0 - FLEXCFG configuration will not be applied on the next Flex state transition. |

**TABLE 18: FLEX CONFIGURATION LENGTH REGISTERS**

| FLEXCFG_LEN (BFD2_345Ch-BFD2_345D) | | | Flex Configuration Length Registers |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 15:10 | Reserved | R | Reserved |
| 9:0 | FLEXCFG_LEN | R/W | This field indicates the length of the FLEXCFG data. |

**TABLE 19: FLEX CONFIGURATION DATA REGISTERS**

| FLEXCFG_DATA (BFD2_3800h-BFD2_39FF) | | | Flex Configuration Data Registers |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7:0 | FLEXCFG_DATA | R/W | This is where bytes from FLEXCFG configuration are programmed. |

The recommended procedure for implementing FLEXCFG is as follows:

1. Create a configuration file with all the options that will be applied after flexing.
2. Program the bytes from the configuration file starting at the first register of FLEXCFG_DATA (BFD2_3800h).
3. Program the length of the data in the configuration file/FLEXCFG_DATA area into the FLEXCFG_LEG register.
4. Set the APPLY_FLEXCFG bit in FLEXCFG_CTL_REG.
5. Send the FlexConnect command.

At this point, the data from FLEXCFG_DATA registers of FLEXCFG_LEN length are applied to the hub and is visible to the new flexed host. The ports revert to their original state before flexing and after unflexing.

If the same configuration will be programmed when the next FlexConnect occurs, rewrite the length of the FLEXCFG patch in FLEXCFG_LEN and send the FlexConnect command. If a new data will be programmed, clear out the data in the FLEXCFG_DATA area then repeat and follow the recommended procedure in steps 1 to 5 above.

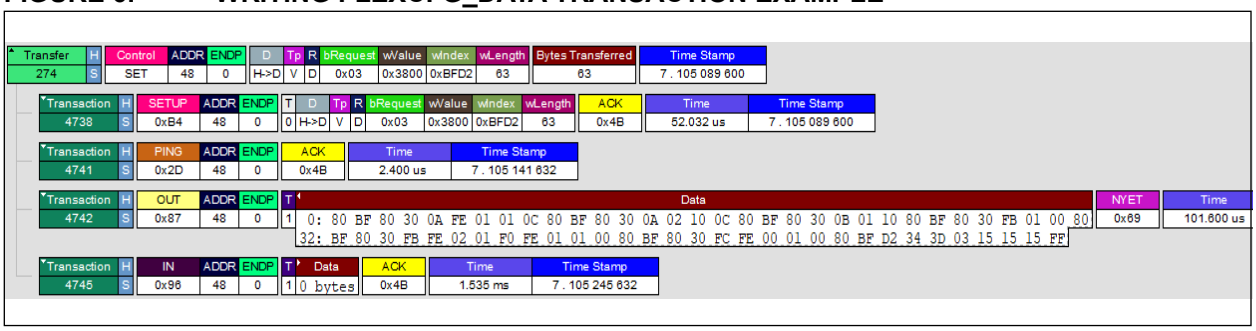
The following is an example configuration that might be programmed into the FLEXCFG space, as well as a USB trace of the process.

Step 1: Create a configuration file with all the options that will be applied after flexing.

In this example, FlexConnect is sent to port 1. This configuration file disables ports 2-4 and sets them to be dedicated charging ports (DCP). After flexing, the new host sees a 2-port hub with the other three ports disabled and set to DCP.

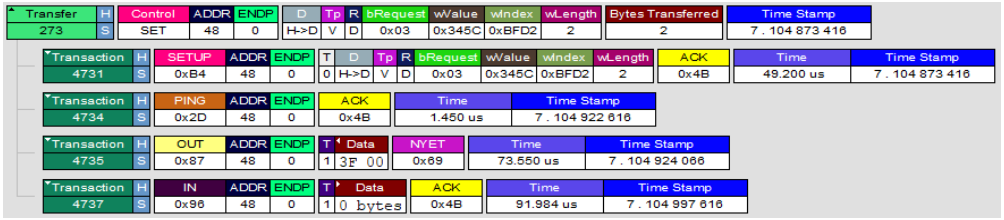
Step 2: Program the bytes from the configuration file starting at the first register of FLEXCFG_DATA (BFD2_3800h).

**FIGURE 6: WRITING FLEXCFG_DATA TRANSACTION EXAMPLE**



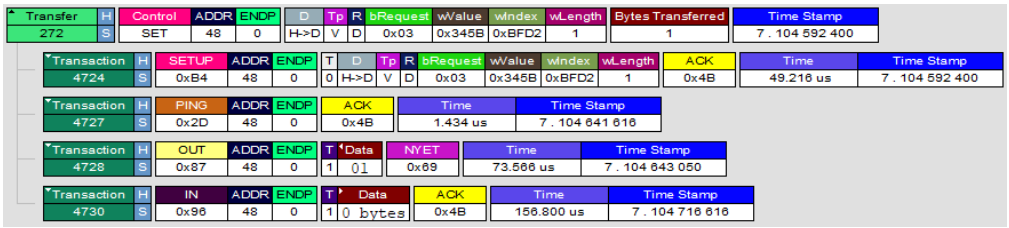

Step 3: Program the length of the data in the configuration file/FLEXCFG_DATA area into the FLEXCFG_LEG register.

**FIGURE 7: WRITING FLEXCFG_LEN TRANSACTION EXAMPLE**



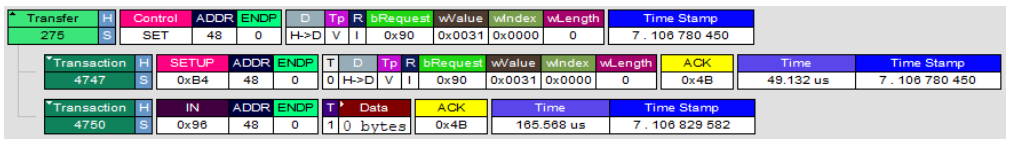

Step 4: Set the APPLY_FLEXCFG bit in FLEXCFG_CTL_REG.

**FIGURE 8: WRITING FLEXCFG_CTL TRANSACTION EXAMPLE**




Step 5: Send the FlexConnect command.

**FIGURE 9: SENDING FLEXCONNECT COMMAND TO PORT 1 TRANSACTION EXAMPLE**




| **Note:** | If battery charging will be enabled on a port via the FLEXCFG area, as in the example above, battery charging **must** also be enabled via OTP or strapping in the original unflexed configuration. For example, if port 2 will be configured as a battery charging port in the FLEXCFG area then battery charging must be enabled via the CFG_BC_EN strap or OTP write in the default Unflexed state. |
|---|---|

## ADDITIONAL HARDWARE CONSIDERATIONS

Schematics for FlexConnect applications are implementation-specific, and hence do not necessarily follow any common framework. The following sections describe the common considerations that must be made for each design.

### VBUS_DET

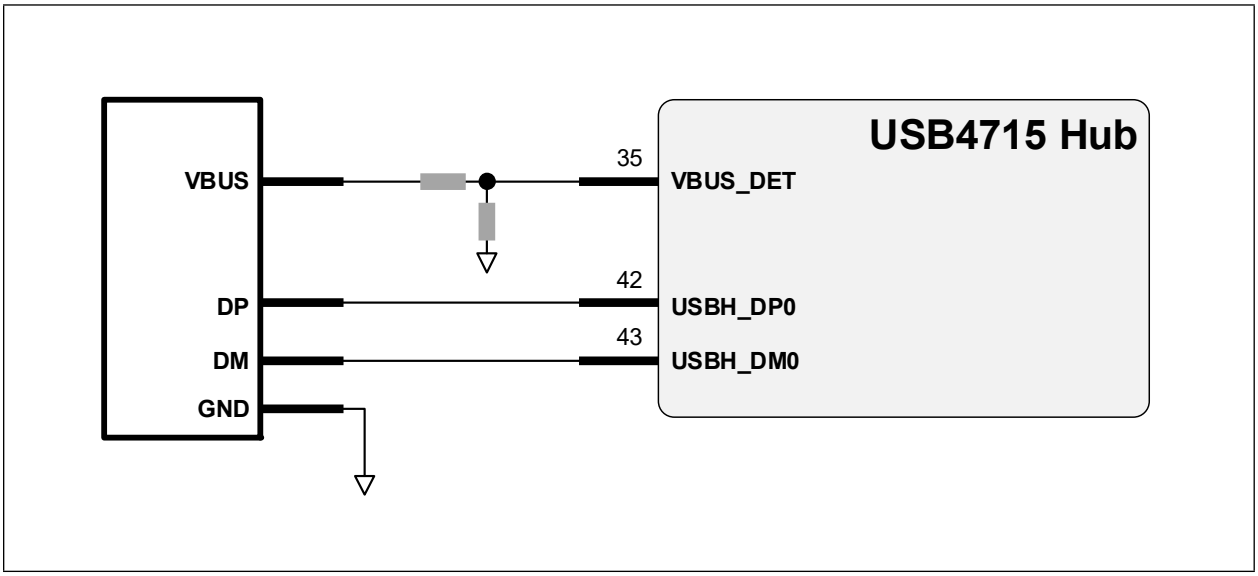
VBUS_DET is a signal used by the hub to determine when a USB host is connected. When VBUS_DET is not present, the hub enters a low-power state to conserve energy. This causes the hub to disconnect when enumerated either through the upstream port or through the new FlexConnect upstream port.

#### NO POWER ROLE SWAP VBUS_DET IMPLEMENTATION

If power roles do not change when initiating FlexConnect, it might be appropriate to connect VBUS_DET directly to the VBUS pin of the default USB host through a resistor divider. This is the standard VBUS_DET implementation, as shown in Figure 10. This is a popular implementation in the automotive application space as the head unit is always connected to the hub.

| | |
|---|---|
| **Note:** | Ensure that the default USB host will not toggle its VBUS supply at any time when FlexConnect is initiated. Otherwise, the hub will be reset and revert to its default state. |

**FIGURE 10: NO POWER ROLE SWAP VBUS_DET CONFIGURATION**




#### CONNECT VBUS_DET DIRECTLY TO A FIXED 3.3V

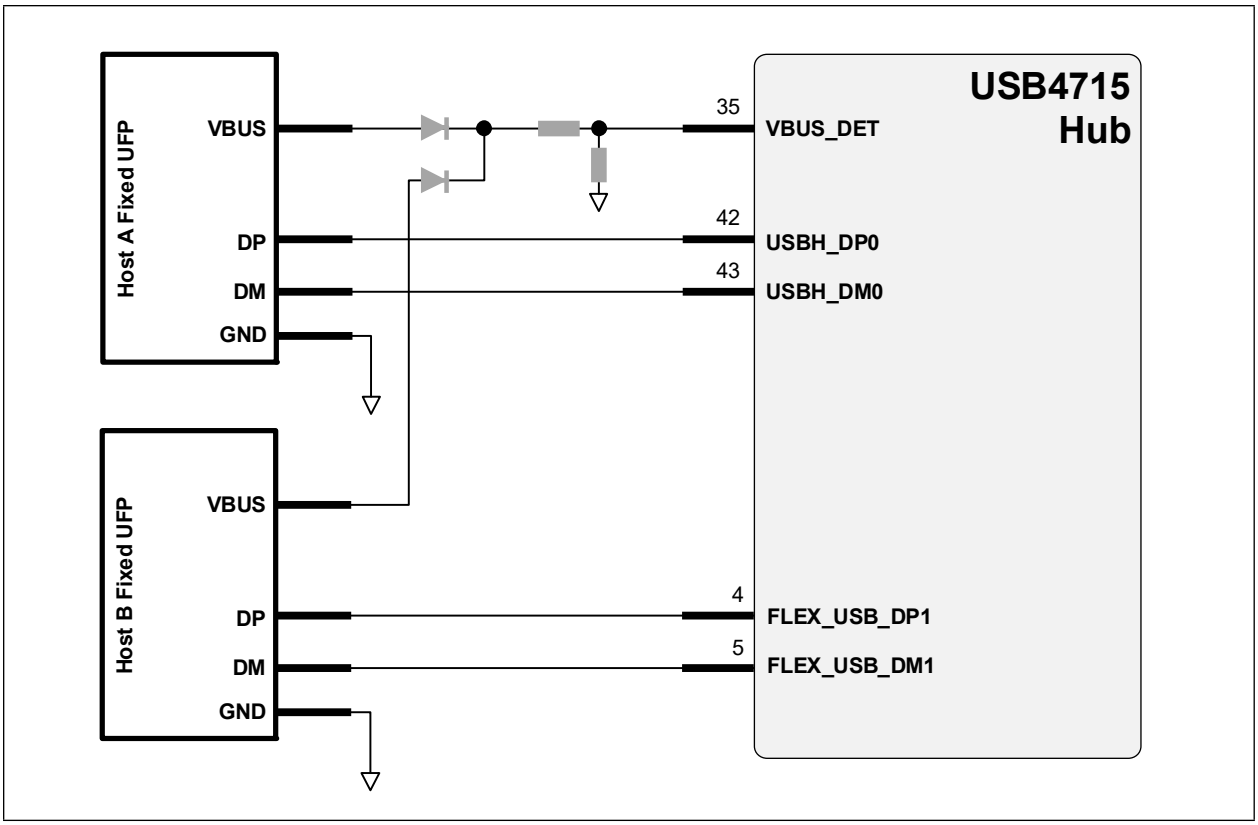
If low-power states are not required when no USB host is present, it is acceptable to connect VBUS_DET to a fixed 3.3V supply on the PCB. This method is only recommended for designs in which the hub is connected to an embedded host IC.

| | |
|---|---|
| **Note:** | In some instances, a USB host might attempt to force a hard reset on a device by toggling VBUS. In this instance, the hub will not reset. |

‘OR’ ALL HOST VBUS CONNECTIONS TO VBUS_DET

In Host Sharing type systems, ‘OR’ing all Host VBUS connections would be an appropriate solution, as shown in Figure 11. This can be used when the “one host at a time” implementation is used. The drawback with this type of solution is that there is no way to distinguish which host port is actually supplying VBUS.

**FIGURE 11: DIODE-OR VBUS_DET SIGNALS**




**Note:** The resistor divider values change depending on the characteristics of the diodes used.

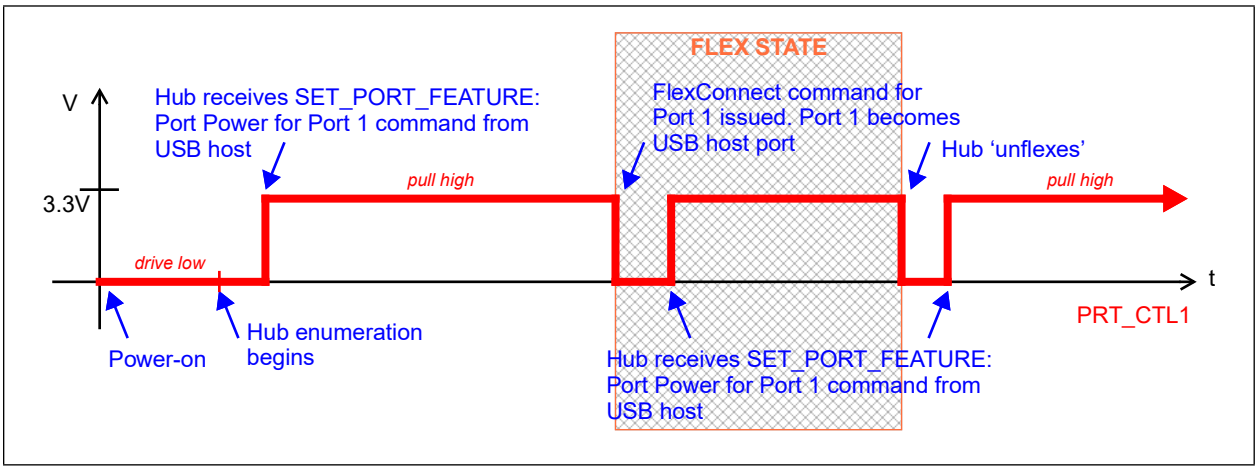
## Port 1-4 Port Power Control

The required behavior of the PRT_CTLx pins should be considered before and after the FlexConnect mode is initiated during the system architecture design phase of the application.

By default, the PRT_CTLx pin continues to respond as the port control pin for its respective logical port in both the Unflexed and Flexed states.

Figure 12 describes how a PRT_CTLx pin operates by default when the FlexConnect mode is entered or exited. The example shown is for port 1 (PRT_CTL1), but the behavior is identical for all ports.

**FIGURE 12: PRT_CTLX DURING FLEXCONNECT OPERATION WHEN FLEX_PRTCTL_EN = 0**




## FlexConnect State Indicator Output

GPIOs may also be assigned the role of FlexConnect status indicator outputs. The use of these outputs is completely optional. These outputs can be connected to external LEDs or an embedded microcontroller to communicate the current FlexConnect state.

The GPIO can be configured as drive-high or drive-low or open drain, and the polarity can be configured. This mode should not be used in combination with the FlexConnect GPIO input mode.

**TABLE 20: PORT 1 FLEXCONNECT STATE INDICATOR GPIO CONFIGURATION REGISTER**

| FLEX_OUT_PORT1 (BFD2_3446h) | | | FlexConnect State Indicator GPIO Configuration |
|---|---|---|---|
| **Bit** | **Name** | **R/W** | **Description** |
| 7 | FLEX_OUT_EN | R/W | 1 - Enables the FlexConnect state indicator for port 1 through GPIO. The specific GPIO is selected in bits [2:0] of this register.<br>0 - Disables the FlexConnect state indicator for port 1 through GPIO. All other bits in this register are ignored. |
| 6 | FLEX_OUT_ACTIVE_HIGH | R/W | 1 - The selected GPIO is driven active-high while in the FlexConnect state; otherwise, it is driven low.<br>0 - The selected GPIO is driven active low while in the FlexConnect state; otherwise, it is driven high. |
| 5 | FLEX_OUT_OD | R/W | 1 - The selected GPIO is in the open-drain output mode.<br>0 - The selected GPIO is in the drive-high or drive-low output mode. |
| 4:3 | Reserved | R | Reserved |
| 2:0 | FLEX_OUT_IO | R/W | Selects the GPIO to be used for the Port 1 FlexConnect state indicator.<br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (Only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (Only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (Only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (Only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 21: PORT 2 FLEXCONNECT STATE INDICATOR GPIO CONFIGURATION REGISTER**

| FLEX_OUT_PORT2 (BFD2_3447h) | | | FlexConnect State Indicator GPIO Configuration |
|---|---|---|---|
| **BIT** | **Name** | **R/W** | **Description** |
| 7 | FLEX_OUT_EN | R/W | 1 - Enables the FlexConnect state indicator for port 2 through GPIO. The specific GPIO is selected in bits [2:0] of this register.<br>0 - Disables the FlexConnect state indicator for port 2 through GPIO. All other bits in this register are ignored. |
| 6 | FLEX_OUT_ACTIVE_HIGH | R/W | 1 - The selected GPIO is driven active-high while in the FlexConnect state; otherwise, it is driven low.<br>0 - The selected GPIO is driven active-low while in the FlexConnect state; otherwise, it is driven high. |
| 5 | FLEX_OUT_OD | R/W | 1 - The selected GPIO is in the open-drain output mode.<br>0 - The selected GPIO is in the drive-high or drive-low output mode. |
| 4:3 | Reserved | R | Reserved |
| 2:0 | FLEX_OUT_IO | R/W | Selects the GPIO to be used for the Port 2 FlexConnect state indicator.<br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (Only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (Only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (Only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (Only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 22: PORT 3 FLEXCONNECT STATE INDICATOR GPIO CONFIGURATION REGISTER**

| FLEX_OUT_PORT3 (BFD2_3448h) | | | FlexConnect State Indicator GPIO Configuration |
|---|---|---|---|
| **BIT** | **Name** | **R/W** | **Description** |
| 7 | FLEX_OUT_EN | R/W | 1 - Enables the FlexConnect state indicator for port 3 through GPIO. The specific GPIO is selected in bits [2:0] of this register.<br>0 - Disables the FlexConnect state indicator for port 3 through GPIO. All other bits in this register are ignored. |
| 6 | FLEX_OUT_ACTIVE_HIGH | R/W | 1 - The selected GPIO is driven active-high while in the FlexConnect state; otherwise, it is driven low.<br>0 - The selected GPIO is driven active-low while in the FlexConnect state; otherwise, it is driven high. |
| 5 | FLEX_OUT_OD | R/W | 1 - The selected GPIO is in the open-drain output mode.<br>0 - The selected GPIO is in the drive-high or drive-low output mode. |
| 4:3 | Reserved | R | Reserved |
| 2:0 | FLEX_OUT_IO | R/W | Selects the GPIO to be used for the Port 3 FlexConnect state indicator.<br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (Only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (Only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (Only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (Only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

**TABLE 23: PORT 4 FLEXCONNECT STATE INDICATOR GPIO CONFIGURATION REGISTER**

| FLEX_OUT_PORT4 (BFD2_3449h) | | | FlexConnect State Indicator GPIO Configuration |
|---|---|---|---|
| **BIT** | **Name** | **R/W** | **Description** |
| 7 | FLEX_OUT_EN | R/W | 1 - Enables the FlexConnect state indicator for port 4 through GPIO. The specific GPIO is selected in bits [2:0] of this register.<br>0 - Disables the FlexConnect state indicator for port 4 through GPIO. All other bits in this register are ignored. |
| 6 | FLEX_OUT_ACTIVE_HIGH | R/W | 1 - The selected GPIO is driven active-high while in the FlexConnect state; otherwise, it is driven low.<br>0 - The selected GPIO is driven active-low while in the FlexConnect state; otherwise, it is driven high. |
| 5 | FLEX_OUT_OD | R/W | 1 - The selected GPIO is in the open-drain output mode.<br>0 - The selected GPIO is in the drive-high or drive-low output mode. |
| 4:3 | Reserved | R | Reserved |
| 2:0 | FLEX_OUT_IO | R/W | Selects the GPIO to be used for the Port 4 FlexConnect state indicator.<br>000 - Reserved<br>001 - Reserved<br>010 - PROG_FUNC4/GPIO6 (Only in CONFIG2/4 modes)<br>011 - PROG_FUNC5/GPIO8 (Only in CONFIG2/4 modes)<br>100 - PROG_FUNC6/GPIO10 (Only in CONFIG2/4 modes)<br>101 - PROG_FUNC7/GPIO11 (Only in CONFIG2/4 modes)<br>110 - Reserved<br>111 - Reserved |

## APPENDIX A: APPLICATION NOTE REVISION HISTORY

**TABLE A-1: REVISION HISTORY**

| Revision Level & Date | Section/Figure/Entry | Correction |
|---|---|---|
| DS00002341D (05-19-25) | Introduction on page 1 | Added additional text explaining that each FlexConnect control mechanism should be treated as mutually exclusive. |
| | FlexConnect Termination on page 3 | Added new section describing all the ways FlexConnect may be terminated. |
| | FlexConnect Register Control (SMBus Control) on page 6 | Added a note discussing the conflicts that can arise when using SMBus configuration along with USB command or GPIO-based FlexConnect control. |
| | FlexConnect Register Control (SMBus Control) on page 6 | Added a new table with information on different ways the hub can be unFlexed and how to disable some of default FlexConnect configuration settings. |
| | FlexConnect Register Control (SMBus Control) on page 6 | Added Table 4, "Runtime Flags Register". |
| | FlexConnect Pin Control on page 13 | Added a note discussing the conflicts that can arise when using SMBus configuration along with USB command or GPIO-based FlexConnect control. |
| DS00002341C (12-12-18) | All | Mentions of SMBus replaced with Register.<br>Minor text changes throughout.<br>Enhancement of graphics. |
| | FlexConnect Register Control (SMBus Control) on page 6 | Updated this section. Updated Table 2 updated, and the removed “FlexConnect Feature Control Register 2” table. |
| | Table 13, Table 14, and Table 15 | Updated the description of FLEX_IN_IO bit. |
| | Flex_Configuration Space on page 17 | Added this section. |
| | FlexConnect USB Command Details on page 11, FlexConnect Pin Control on page 13, and Additional Hardware Considerations on page 19 | Updated these sections.<br>Under Additional Hardware Considerations:<br>• Section title changed to No Power Role Swap VBUS_DET Implementation.<br>• Updated Figure 11 and added a new note under this figure.<br>• Removed “Assign a New VBUS_DET Pin to an Available GPIO” section.<br>• Removed “PRT_CTLX/PRT_CTL0 DURING FLEXCONNECT OPERATION WHEN FLEX-_PRTCTL_EN = 1”.<br>• Added FlexConnect State Indicator Output section. |
| | Table 20, Table 21, Table 22, and Table 23 | Updated the description of FLEX_OUT_IO bit. |
| DS00002341B (08-22-17) | Table 13 to Table 23 | Updated FLEX_IN_IO description. |
| | All | Corrected minor typos and pagination issues. |
| | Table 11 | Updated wValue description. |
| | Figure 2 | Corrected text on green Port 2 device. |
| DS00002341A (03-28-17) | All | Initial release. |

## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the web site at: http://microchip.com/support**

# Microchip Information

## Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legalinformation/microchip-trademarks.

ISBN: 979-8-3371-1185-8

## Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.