AN2932

USB-to-GPIO Bridging with Microchip USB72xx Hubs

Author: Andrew Rogers
Microchip Technology Inc.

INTRODUCTION

The USB-to-GPIO Bridging feature of Microchip's USB72xx family of hubs provides system designers expanded system control and potential BOM reductions. General Purpose Input/Outputs (GPIOs) may be used for any general 3.3V-level digital control and input functions.

Commands may be sent from the USB Host to the internal Hub Feature Controller (HFC) device in the Microchip hub to perform the following functions:

- Set the direction of the GPIO (input or output)
- Enable a pull-up resistor
- Enable a pull-down resistor
- Read the state
- Set the state

SECTIONS

REFERENCES

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

- *Microchip USB7202 Data Sheet*
- *Microchip USB7206 Data Sheet*
- *Microchip USB7250 Data Sheet*
- *Microchip USB7251 Data Sheet*
- *Microchip USB7252 Data Sheet*
- *Microchip USB7256 Data Sheet*
- *Microchip AN2935 Configuration of USB7202/USB7206/USB725x*

GENERAL INFORMATION

Microchip hub USB-to-GPIO Bridging features work via host commands sent to an embedded Hub Feature Controller within the device located on an additional internal USB port. In order for the bridging features to work correctly, this internal Hub Feature Controller must be enabled by default. Table 1 provides details on default Hub Feature Controller settings per device.

TABLE 1: DEFAULT SETTINGS FOR HUB FEATURE CONTROLLER ENABLE

Part Number	Part Summary	Hub Feature Controller Default Setting
USB7202	4-Port USB3.1 Gen2 Hub	Enabled by default on port 6
USB7206	6-Port USB3.1 Gen2 Hub	Enabled by default on port 8
USB7250	4-Port USB3.1 Gen2 Hub with USB power delivery on 3 ports	Enabled by default on port 6
USB7251	4-Port USB3.1 Gen2 Hub with USB power delivery on 2 ports	Enabled by default on port 6
USB7252	4-Port USB3.1 Gen2 Hub with USB power delivery on 1 port	Enabled by default on port 6
USB7256	6-Port USB3.1 Gen2 Hub with USB power delivery on 1 port	Enabled by default on port 8

The Hub Feature Controller is connected to an extra internal port in the hub. It is mapped to the highest numbered port on the hub by default.

The Hub Feature Controller example for the USB7202 is illustrated in Figure 1. USB7206 is shown in Figure 2. USB7250, USB7251, and USB7252 are in Figure 3. USB7256 is shown in Figure 4.

FIGURE 1: USB7202 HUB FEATURE CONTROLLER EXAMPLE

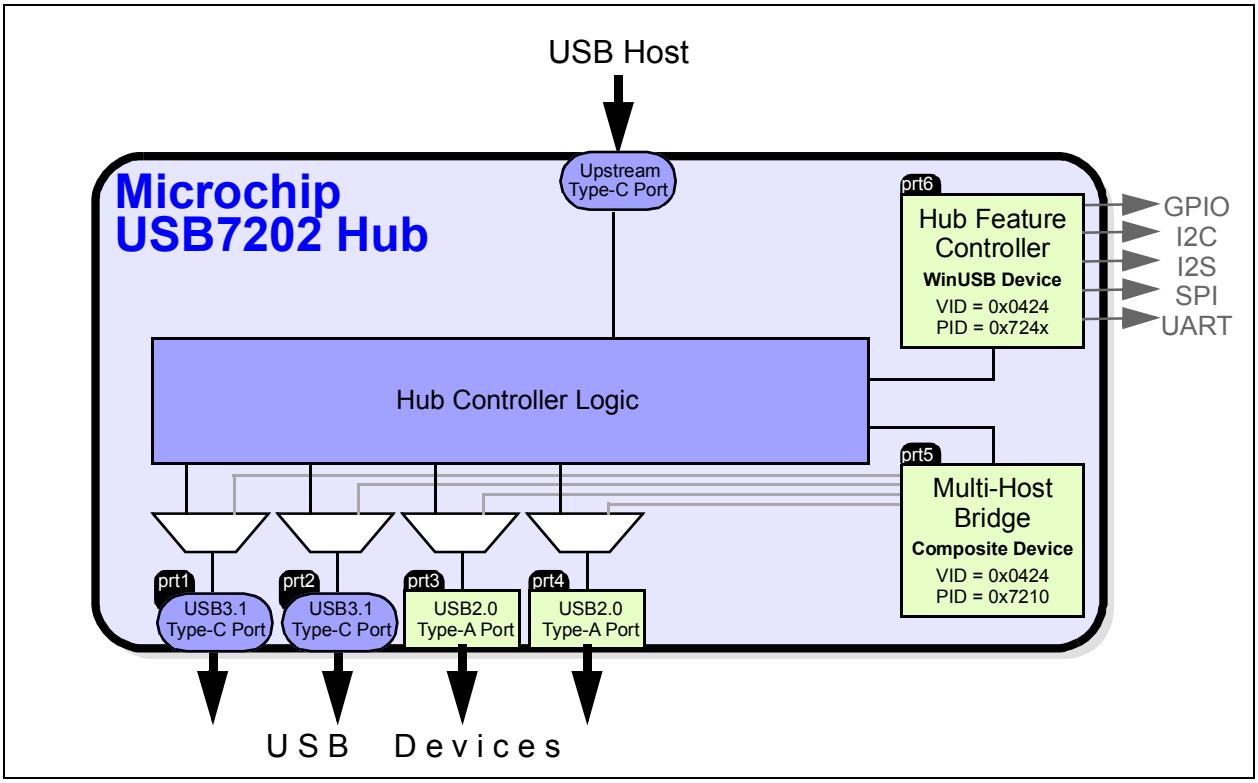


FIGURE 2: USB7206 HUB FEATURE CONTROLLER EXAMPLE

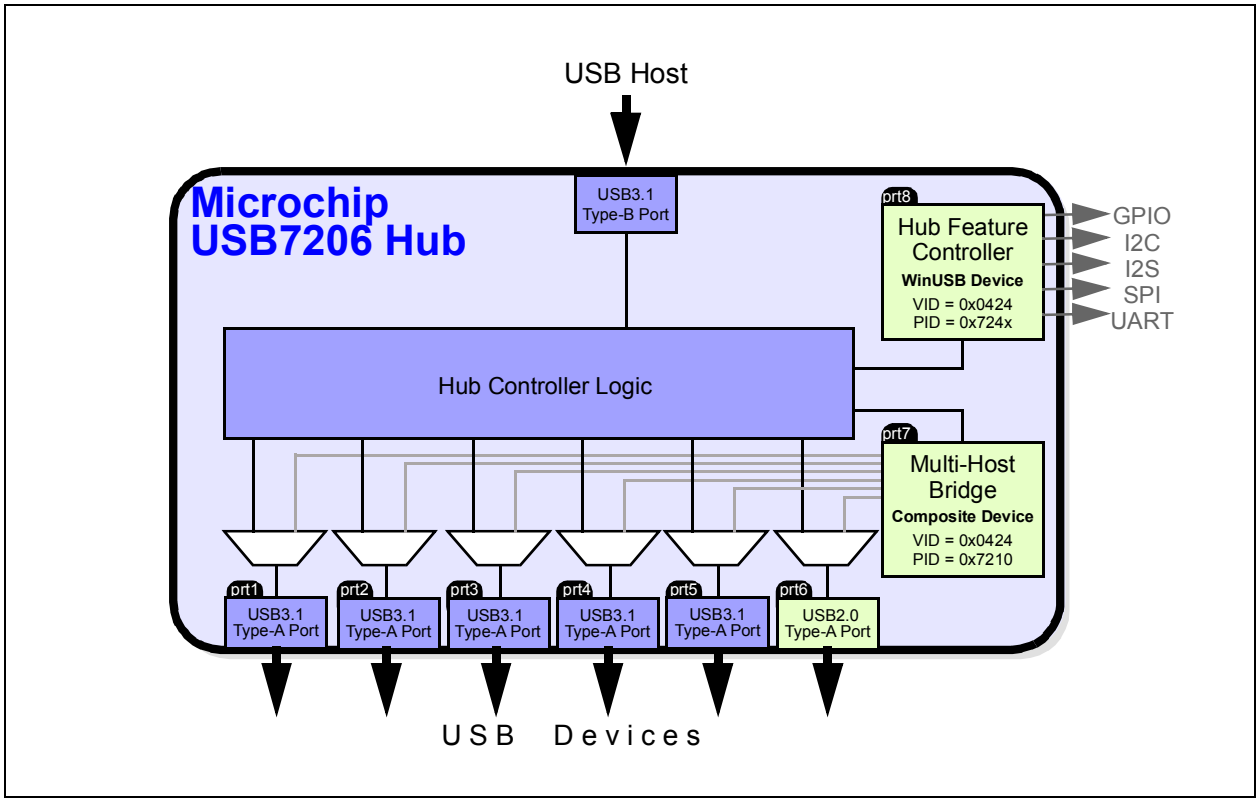


FIGURE 3: USB7250, USB7251, AND USB7252 HUB FEATURE CONTROLLER EXAMPLE

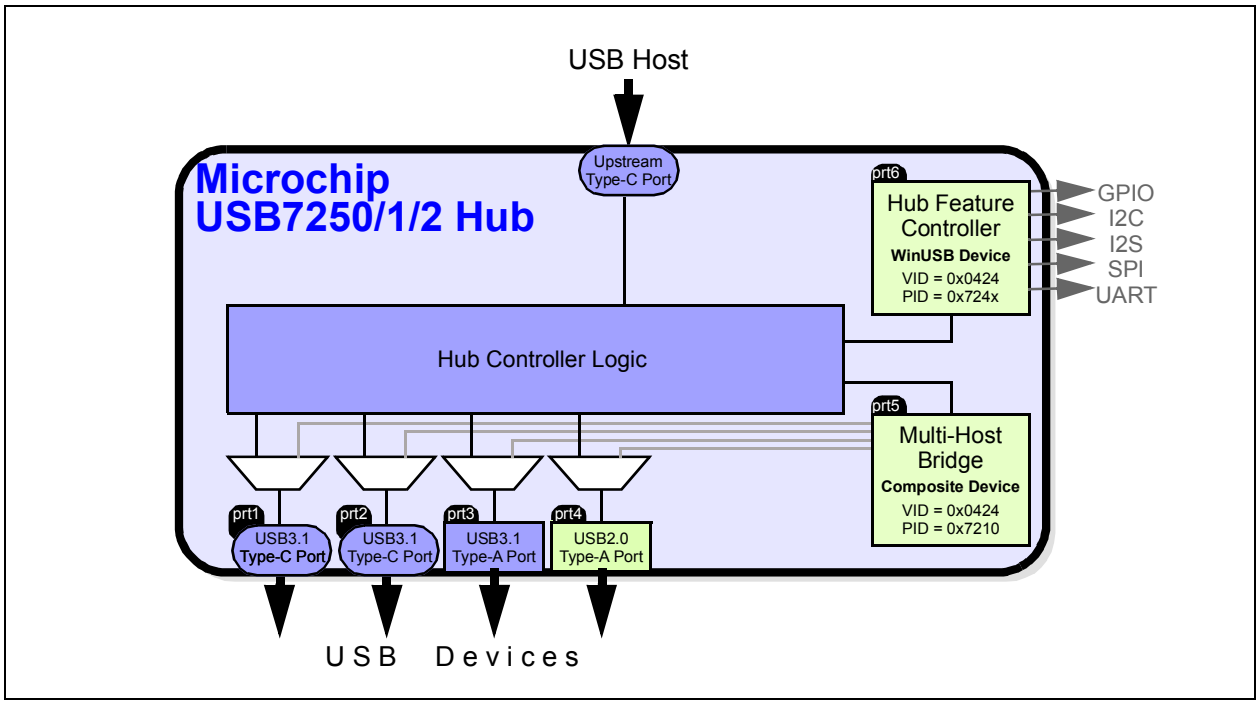
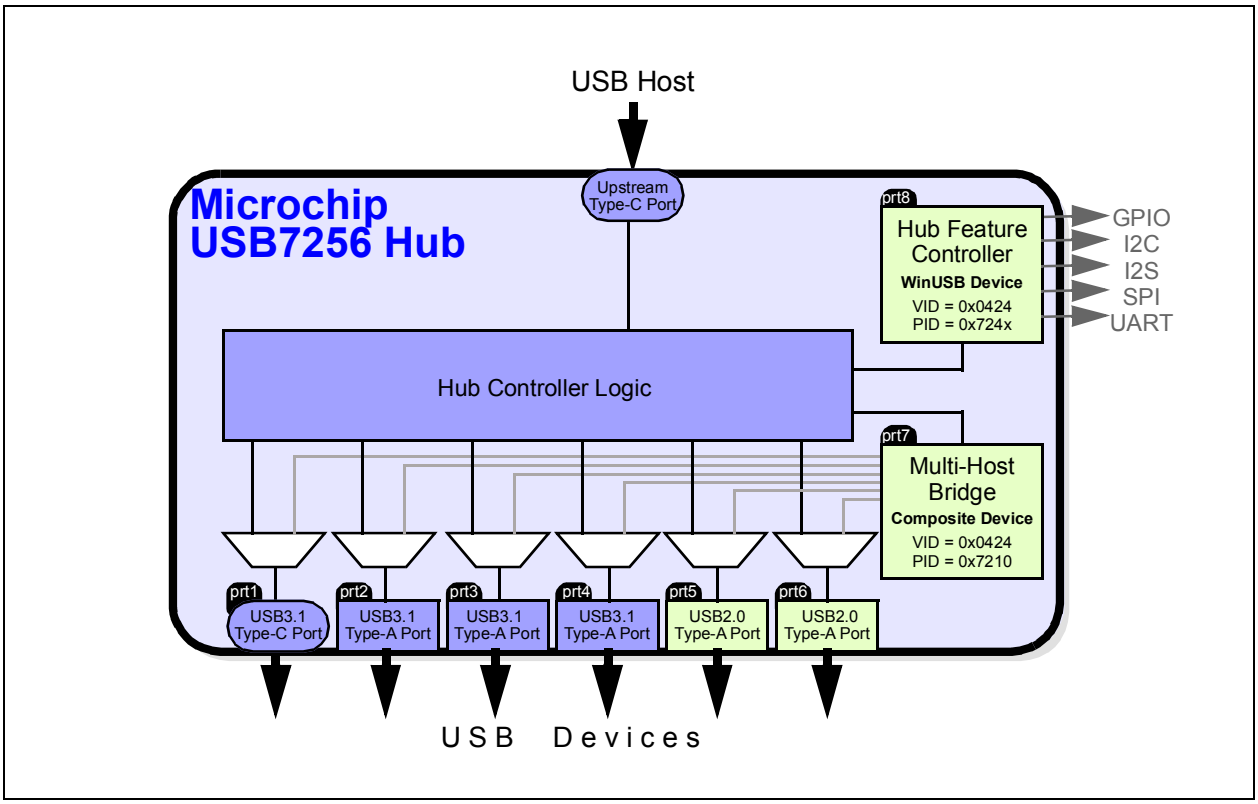


FIGURE 4: USB7256 HUB FEATURE CONTROLLER EXAMPLE



The following GPIO functions are supported:

- Set the GPIO Direction (Input or Output)
- Enable GPIO Internal Pull-Up Resistor
- Enable GPIO Internal Pull-Down Resistor
- GPIO Read State (Input Mode)
- GPIO Set State (Output Mode)

Set the GPIO Direction (Input or Output)

Each GPIO can be configured as either a Schmitt-triggered input or output with an 8 mA sink/source.

Enable GPIO Internal Pull-Up Resistor

Each GPIO can be enabled with a 50 uA (typical) internal pull-up resistor. Internal pull-up resistors prevent unconnected inputs from floating. The pull-up is only 67k, so it may not be strong enough to drive a load of less than 100k. When connected to a load that must be pulled high, an external resistor must be added.

Enable GPIO Internal Pull-Down Resistor

Each GPIO can be enabled with a 50 uA (typical) internal pull-down resistor. Internal pull-down resistors prevent unconnected inputs from floating. The pull-down is only 67k, so it may not be strong enough to drive a load of less than 100k. When connected to a load that must be pulled low, an external resistor must be added.

GPIO Read State (Input Mode)

Read a 0: GPIO is below 0.9V.

Read a 1: GPIO is above 1.9V.

Note: When configured as an input, the GPIOs are digital Schmitt-triggered inputs. The range 0.9V to 1.9V is an indeterminate input state, so 3.3V-to-2.5V signaling is supported.

GPIO Set State (Output Mode)

Set to 0: GPIO Drives to 0.0V. When driven low, an 8 mA sink is enabled, driving the pin to 0.4V or lower.

Set to 1: GPIO Drives to 3.3V. When driven high, an 8 mA source is enabled, driving the pin to VDD33 to 0.4V or higher.

PART NUMBER-SPECIFIC INFORMATION

Part Summary

Table 2 summarizes the total number of available GPIOs by part number. Many of the GPIOs on the hub are only available after configuration. The following methods may be used to configure the hub:

- **MPLAB Connect Configurator:** If configuring via internal One-Time Programmable (OTP) memory or SPI EEPROM with a base firmware file
- **SMBus/I^{2}C Configuration:** If using an embedded SoC/I^{2}C EEPROM to configure the hub at each start-up/reset
- **Pin Strapping:** Many of the GPIOs are made available by specific pin strapping or by simply not populating an SPI EEPROM device.

Table 4 to Table 8 provide a detailed view of the available GPIOs for each configuration strap (CFG_STRAP) option.

TABLE 2: GPIO AVAILABILITY SUMMARY

	USB7202	USB7206	USB7250	USB7251	USB7252	USB7256
Minimum GPIOs Available	4 (CONFIG2)	1 (CONFIG1)	8 (CONFIG1)	4 (CONFIG3)	1 (CONFIG3)	1 (CONFIG2)
Maximum GPIOs Available	12 (CONFIG4)	1 (CONFIG1)	14 (CONFIG4)	11 (CONFIG4)	8 (CONFIG4)	2 (CONFIG1)

TABLE 3: USB7202 GPIOS

	CONFIG1 (I^{2}C)	CONFIG2 (I^{2}S)	CONFIG3 (UART)	CONFIG4 (FLEX)
PF6	GPIO70	GPIO70	*UART_RX*	GPIO70
PF7	GPIO71	*MIC_DET*	*UART_TX*	GPIO71
PF12	GPIO76	—	—	GPIO76
PF14	GPIO78	*I2S_SDI*	*UART_nCTS*	GPIO78
PF18	*MSTR_I2C_CLK*	*I2S_LRCK*	*UART_nDCD*	GPIO82
PF19	*MSTR_I2C_DATA*	*I2S_SDO*	*UART_nRTS*	GPIO83
PF26	*SLV_I2C_CLK*	*I2S_SCK*	*UART_nDSR*	GPIO90
PF27	*SLV_I2C_DATA*	*I2S_MCLK*	*UART_nDTR*	GPIO91
PF28	GPIO92	GPIO92	GPIO92	GPIO92
PF29	GPIO93	GPIO93	GPIO93	GPIO93
PF30	GPIO94	*MSTR_I2C_CLK*	GPIO94	GPIO94
PF31	GPIO95	*MSTR_I2C_DATA*	GPIO95	GPIO95

TABLE 4: USB7206 GPIOS

	CONFIG1
PF29	GPIO93

TABLE 5: USB7250 GPIOS

	CONFIG1 (I^2C)	CONFIG2 (I^2S)	CONFIG3 (UART)	CONFIG4 (FLEX)	CONFIG5
PF2	GPIO66	GPIO66	*UART_nCTS*	GPIO66	GPIO66
PF3	GPIO67	*I2S_SDI*	*UART_nRTS*	GPIO67	GPIO67
PF4	*PD_SPI_CE_N2*	*I2S_SDO*	*UART_nDSR*	GPIO68	GPIO68
PF5	*PD_SPI_CE_N1*	*I2S_SCK*	*UART_nDTR*	GPIO69	GPIO69
PF6	*PD_SPI_CE_N0*	*I2S_LRCK*	*UART_RX*	GPIO70	GPIO70
PF7	*PD_SPI_CLK*	*I2S_MCLK*	*UART_TX*	GPIO71	GPIO71
PF14	GPIO78	GPIO78	GPIO78	GPIO78	GPIO78
PF19	SLV_I2C_DATA	SLV_I2C_DATA	SLV_I2C_DATA	GPIO83	*SLV_I2C_DATA*
PF26	*SLV_I2C_CLK*	*SLV_I2C_CLK*	*SLV_I2C_CLK*	GPIO90	*SLV_I2C_CLK*
PF27	GPIO91	*MIC_DET*	GPIO91	GPIO91	GPIO91
PF28	GPIO92	GPIO92	GPIO92	GPIO92	GPIO92
PF29	GPIO93	GPIO93	GPIO93	GPIO93	GPIO93
PF30	GPIO94	GPIO94	GPIO94	GPIO94	GPIO94
PF31	GPIO95	GPIO95	GPIO95	GPIO95	GPIO95

TABLE 6: USB7251 GPIOS

	CONFIG1 (I^2C)	CONFIG2 (I^2S)	CONFIG3 (UART)	CONFIG4 (FLEX)
PF4	GPIO68	GPIO68	GPIO68	GPIO68
PF6	GPIO70	GPIO70	*UART_RX*	GPIO70
PF7	GPIO71	*MIC_DET*	*UART_TX*	GPIO71
PF14	GPIO78	*I2S_SDI*	*UART_nCTS*	GPIO78
PF19	SLV_I2C_DATA	I2S_SDO	*UART_nRTS*	GPIO83
PF26	*SLV_I2C_CLK*	I2S_SCK	*UART_nDSR*	GPIO90
PF27	GPIO91	I2S_MCLK	*UART_nDTR*	GPIO91
PF28	GPIO92	I2S_LRCK	*UART_nDCD*	GPIO92
PF29	GPIO93	GPIO93	GPIO93	GPIO93
PF30	GPIO94	GPIO94	GPIO94	GPIO94
PF31	GPIO95	GPIO95	GPIO95	GPIO95

TABLE 7: USB7252 GPIOS

	CONFIG1 (I^2C)	CONFIG2 (I^2S)	CONFIG3 (UART)	CONFIG4 (FLEX)
PF6	GPIO70	GPIO70	*UART_RX*	GPIO70
PF7	GPIO71	*MIC_DET*	*UART_TX*	GPIO71
PF14	GPIO78	*I2S_SDI*	*UART_nCTS*	GPIO78
PF19	SLV_I2C_DATA	I2S_SDO	*UART_nRTS*	GPIO83
PF26	*SLV_I2C_CLK*	I2S_SCK	*UART_nDSR*	GPIO90
PF27	GPIO91	I2S_MCLK	*UART_nDTR*	GPIO91
PF28	GPIO92	I2S_LRCK	*UART_nDCD*	GPIO92
PF29	GPIO93	GPIO93	GPIO93	GPIO93

TABLE 8: USB7256 GPIOS

	CONFIG1 (I^2C)	CONFIG2 (I^2S)
PF6	GPIO70	GPIO70
PF7	GPIO71	*MIC_DET*

MICROCHIP SOFTWARE SOLUTIONS

Microchip currently offers two publicly available software solutions to facilitate USB-to-GPIO Bridging in a USB72xx hubs on Windows® and Linux®.

MPLAB Connect Configurator Package (For Windows®)

The MPLAB Connect Configurator (MPLABCC) package consists of both GUI-based and CLI-based tools which support USB-to-GPIO Bridging in a standalone form. In addition to these, it contains a Dynamically Linked Library (DLL) for Windows which can be used for implementing USB-to-GPIO Bridging feature in custom applications using C programming language. The MPLABCC DLL consists of the following:

- User's guide: A detailed description of how to use the DLL API to call each function
- Release notes
- Library files: A .dll and a .lib file
- Example code

Application Code Examples (For Linux®)

For implementing USB-to-GPIO Bridging on Linux, you can use the following USB72xx Linux Application Code Example (ACE):

- ACE007 USB-to-GPIO Bridging: This ACE demonstrates how to use the GPIO Master interface of the hub to perform read/write operations. It also allows the user to select from a range of GPIO clock frequencies.

This application example uses libusb library for Linux to build and send USB packets as described in Manual Implementation. It is a full-feature code example that consists of:

- Example code with minimal abstraction and in-line comments describing the various steps involved
- A Makefile
- README

This ACE can be used as a standalone application and can be integrated into existing applications.

Note:	Visit the product page on www.microchip.com for any of the hubs listed in this document to download the software solution for the desired operating system.

MANUAL IMPLEMENTATION

The USB-to-GPIO Bridging features may be implemented at the lowest level if you have the ability to build USB packets.

All USB-to-GPIO Bridging commands are accomplished with internal register writes and reads. Further details can be found in the Microchip application note, *AN2935 Configuration of USB7202/USB7206/USB725x*. All USB-to-GPIO Bridging commands must be sent directly to Endpoint 0 of the Hub Feature Controller connected to the last downstream port of the Microchip hub.

For details on the register read and write USB SETUP packets, refer to Register Read and Register Write, respectively. The configuration register addresses and contents are detailed in GPIO Configuration Register Map and Register Definitions.

Register Read

To read the state of a GPIO, a register read with the USB SETUP packet in Table 9 must be used:

TABLE 9: REGISTER READ USB SETUP COMMAND

SETUP Parameter	Value	Description
bmRequestType	0xC0	Device-to-host, vendor class, targeted to interface
bRequest	0x04	Register read command: CMD_REG_READ
wValue	Register address LSB	Valid address range: <0x0000> to <0xFFFF> [64KB]
wIndex	Register address MSB	Valid address range: <0x0000> to <0xFFFF> [64KB]
wLength	Data length	Length of the data bytes to be retrieved

REGISTER READ USB TRANSACTION SEQUENCE

Command Phase: The Hub Feature Controller receives the SETUP packet with the parameters specified in Table 9.

Data Phase: The Hub Feature Controller sends the data bytes of length wLength from the specified address.

Status Phase: The Hub Feature Controller sends ACK on the successful completion of register read.

Register Write

To configure the direction of a GPIO, pull-up/pull-down resistor settings, or set the output state of a GPIO, a register write command with the USB SETUP packet in Table 10 must be used:

TABLE 10: REGISTER WRITE USB SETUP COMMAND

SETUP Parameter	Value	Description
bmRequestType	0x40	Host-to-device, vendor class, targeted to interface
bRequest	0x03	Register read command: CMD_REG_WRITE
wValue	Register address LSBs	Last four bytes of the 32-bit register address
wIndex	Register address MSBs	First four bytes of the 32-bit register address
wLength	Data length	Length of data bytes to write

REGISTER WRITE USB TRANSACTION SEQUENCE

Command Phase: The Hub Feature Controller receives the SETUP packet with the parameters specified in Table 10.

Data Phase: The Hub Feature Controller receives the data bytes of length wLength to be written to the register starting from the specified address.

Status Phase: The Hub Feature Controller sends ACK on successful completion of register write.

GPIO Configuration Register Map

TABLE 11: CONFIGURATION REGISTER MEMORY MAP

Address	Name	R/W	Function	Default
BF80 0908	PIO96_OEN	R/W	PIO[95:64] Output Enable Register	00h
BF80 0918	PIO96_IEN	R/W	PIO[95:64] Input Enable Register	00h
BF80 0928	PIO96_OUT	R/W	PIO[95:64] Output State Register	00h
BF80 0938	PIO96_IN	R	PIO[95:64] Input State Register	00h
BF80 0948	PIO96_PUE	R/W	PIO[95:64] Pull-up Enable Register	00h
BF80 0958	PIO96_PDE	R/W	PIO[95:64] Pulldown Enable Register	00h

Register Definitions

Table 12 to Table 17 show details for all GPIO-related registers.

TABLE 12: PIO[95:64] OUTPUT ENABLE REGISTER

PIO96_OEN (BF80 0908h)			PIO[95:64] Output Enable Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_OE	R/W	Set bit to enable GPIO95 as an output.
30	GPIO_94_OE	R/W	Set bit to enable GPIO94 as an output.
29	GPIO_93_OE	R/W	Set bit to enable GPIO93 as an output.
28	GPIO_92_OE	R/W	Set bit to enable GPIO92 as an output.
27	GPIO_91_OE	R/W	Set bit to enable GPIO91 as an output.
26	GPIO_90_OE	R/W	Set bit to enable GPIO90 as an output.
25:20	Reserved	R	Reserved
19	GPIO_83_OE	R/W	Set bit to enable GPIO83 as an output.
18	GPIO_82_OE	R/W	Set bit to enable GPIO82 as an output.
17:15	Reserved	R	Reserved
14	GPIO_78_OE	R/W	Set bit to enable GPIO78 as an output.
13	Reserved	R	Reserved
12	GPIO_76_OE	R/W	Set bit to enable GPIO76 as an output.
11:8	Reserved	R	Reserved
7	GPIO_71_OE	R/W	Set bit to enable GPIO71 as an output.
6	GPIO_70_OE	R/W	Set bit to enable GPIO70 as an output.
5	GPIO_69_OE	R/W	Set bit to enable GPIO69 as an output.
4	GPIO_68_OE	R/W	Set bit to enable GPIO68 as an output.
3	GPIO_67_OE	R/W	Set bit to enable GPIO67 as an output.
2	GPIO_66_OE	R/W	Set bit to enable GPIO66 as an output.
1:0	Reserved	R	Reserved
Note:	**BF80_0908h:** GPIO66-71; **BF80_0909h:** GPIO76-78; **BF80_090Ah:** GPIO82-83; **BF80_090Bh:** GPIO 90-92		

TABLE 13: PIO[95:64] INPUT ENABLE REGISTER

PIO96_IEN (BF80 0918h)			PIO[95:64] Input Enable Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_IE	R/W	Set bit to enable GPIO95 as an input.
30	GPIO_94_IE	R/W	Set bit to enable GPIO94 as an input.
29	GPIO_93_IE	R/W	Set bit to enable GPIO93 as an input.
28	GPIO_92_IE	R/W	Set bit to enable GPIO92 as an input.
27	GPIO_91_IE	R/W	Set bit to enable GPIO91 as an input.
26	GPIO_90_IE	R/W	Set bit to enable GPIO90 as an input.
25:20	Reserved	R	Reserved
19	GPIO_83_IE	R/W	Set bit to enable GPIO83 as an input.
18	GPIO_82_IE	R/W	Set bit to enable GPIO82 as an input.
17:15	Reserved	R	Reserved
14	GPIO_78_IE	R/W	Set bit to enable GPIO78 as an input.
13	Reserved	R	Reserved
12	GPIO_76_IE	R/W	Set bit to enable GPIO76 as an input.
11:8	Reserved	R	Reserved
7	GPIO_71_IE	R/W	Set bit to enable GPIO71 as an input.
6	GPIO_70_IE	R/W	Set bit to enable GPIO70 as an input.
5	GPIO_69_IE	R/W	Set bit to enable GPIO69 as an input.
4	GPIO_68_IE	R/W	Set bit to enable GPIO68 as an input.
3	GPIO_67_IE	R/W	Set bit to enable GPIO67 as an input.
2	GPIO_66_IE	R/W	Set bit to enable GPIO66 as an input.
1:0	Reserved	R	Reserved
Note:	**BF80_0918h:** GPIO66-71; **BF80_0919h:** GPIO76-78; **BF80_091Ah:** GPIO82-83; **BF80_091Bh:** GPIO 90-92		

TABLE 14: PIO[95:64] OUTPUT STATE REGISTER

PIO96_OUT (BF80 0928h)			PIO[95:64] Output State Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_OS	R/W	Set bit to drive GPIO95 high. Clear bit to drive GPIO95 low.
30	GPIO_94_OS	R/W	Set bit to drive GPIO94 high. Clear bit to drive GPIO94 low.
29	GPIO_93_OS	R/W	Set bit to drive GPIO93 high. Clear bit to drive GPIO93 low.
28	GPIO_92_OS	R/W	Set bit to drive GPIO92 high. Clear bit to drive GPIO92 low.
27	GPIO_91_OS	R/W	Set bit to drive GPIO91 high. Clear bit to drive GPIO91 low.
26	GPIO_90_OS	R/W	Set bit to drive GPIO90 high. Clear bit to drive GPIO90 low.
25:20	Reserved	R	Reserved
19	GPIO_83_OS	R/W	Set bit to drive GPIO83 high. Clear bit to drive GPIO83 low.
18	GPIO_82_OS	R/W	Set bit to drive GPIO82 high. Clear bit to drive GPIO82 low.
17:15	Reserved	R	Reserved
14	GPIO_78_OS	R/W	Set bit to drive GPIO78 high. Clear bit to drive GPIO78 low.
13	Reserved	R	Reserved
12	GPIO_76_OS	R/W	Set bit to drive GPIO76 high. Clear bit to drive GPIO76 low.
11:8	Reserved	R	Reserved
7	GPIO_71_OS	R/W	Set bit to drive GPIO71 high. Clear bit to drive GPIO71 low.
6	GPIO_70_OS	R/W	Set bit to drive GPIO70 high. Clear bit to drive GPIO70 low.
5	GPIO_69_OS	R/W	Set bit to drive GPIO69 high. Clear bit to drive GPIO69 low.
4	GPIO_68_OS	R/W	Set bit to drive GPIO68 high. Clear bit to drive GPIO68 low.
3	GPIO_67_OS	R/W	Set bit to drive GPIO67 high. Clear bit to drive GPIO67 low.
2	GPIO_66_OS	R/W	Set bit to drive GPIO66 high. Clear bit to drive GPIO66 low.
1:0	Reserved	R	Reserved
Note:	**BF80_0928h:** GPIO66-71; **BF80_0929h:** GPIO76-78; **BF80_092Ah:** GPIO82-83; **BF80_092Bh:** GPIO 90-92		

TABLE 15: PIO[95:64] INPUT STATE REGISTER

PIO96_IN (BF80 0938h)			PIO[95:64] Input State Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_IS	R/W	Read bit to determine input state of GPIO95.
30	GPIO_94_IS	R/W	Read bit to determine input state of GPIO94.
29	GPIO_93_IS	R/W	Read bit to determine input state of GPIO93.
28	GPIO_92_IS	R/W	Read bit to determine input state of GPIO92.
27	GPIO_91_IS	R/W	Read bit to determine input state of GPIO91.
26	GPIO_90_IS	R/W	Read bit to determine input state of GPIO90.
25:20	Reserved	R	Reserved
19	GPIO_83_IS	R/W	Read bit to determine input state of GPIO83.
18	GPIO_82_IS	R/W	Read bit to determine input state of GPIO82.
17:15	Reserved	R	Reserved
14	GPIO_78_IS	R/W	Read bit to determine input state of GPIO78.
13	Reserved	R	Reserved
12	GPIO_76_IS	R/W	Read bit to determine input state of GPIO76.
11:8	Reserved	R	Reserved
7	GPIO_71_IS	R/W	Read bit to determine input state of GPIO71.
6	GPIO_70_IS	R/W	Read bit to determine input state of GPIO70.
5	GPIO_69_IS	R/W	Read bit to determine input state of GPIO69.
4	GPIO_68_IS	R/W	Read bit to determine input state of GPIO68.
3	GPIO_67_IS	R/W	Read bit to determine input state of GPIO67.
2	GPIO_66_IS	R/W	Read bit to determine input state of GPIO66.
1:0	Reserved	R	Reserved
Note:	**BF80_0938h:** GPIO66-71; **BF80_0939h:** GPIO76-78; **BF80_093Ah:** GPIO82-83; **BF80_093Bh:** GPIO 90-92		

TABLE 16: PIO[95:64] PULL-UP ENABLE REGISTER

PIO96_PUE (BF80 0948h)			PIO[95:64] Pull-Up Resistor Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO95.
30	GPIO_94_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO94.
29	GPIO_93_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO93.
28	GPIO_92_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO92.
27	GPIO_91_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO91.
26	GPIO_90_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO90.
25:20	Reserved	R	Reserved
19	GPIO_83_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO83.
18	GPIO_82_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO82.
17:15	Reserved	R	Reserved
14	GPIO_78_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO78.
13	Reserved	R	Reserved
12	GPIO_76_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO76.
11:8	Reserved	R	Reserved
7	GPIO_71_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO71.
6	GPIO_70_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO70.
5	GPIO_69_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO69.
4	GPIO_68_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO68.
3	GPIO_67_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO67.
2	GPIO_66_PUE	R/W	Set bit to enable ~62k pull-up resistor on GPIO66.
1:0	Reserved	R	Reserved
Note:	**BF80_0948h:** GPIO66-71; **BF80_0949h:** GPIO76-78; **BF80_094Ah:** GPIO82-83; **BF80_094Bh:** GPIO 90-92		

TABLE 17: PIO[95:64] PULLDOWN ENABLE REGISTER

PIO96_PDE (BF80 0958h)			PIO[95:64] Pull-Down Resistor Register
Bit	**Name**	**R/W**	**Description**
31	GPIO_95_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO95.
30	GPIO_94_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO94.
29	GPIO_93_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO93.
28	GPIO_92_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO92.
27	GPIO_91_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO91.
26	GPIO_90_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO90.
25:20	Reserved	R	Reserved
19	GPIO_83_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO83.
18	GPIO_82_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO82.
17:15	Reserved	R	Reserved
14	GPIO_78_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO78.
13	Reserved	R	Reserved
12	GPIO_76_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO76.
11:8	Reserved	R	Reserved
7	GPIO_71_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO71.
6	GPIO_70_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO70.
5	GPIO_69_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO69.
4	GPIO_68_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO68.
3	GPIO_67_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO67.
2	GPIO_66_PDE	R/W	Set bit to enable ~62k pull-down resistor on GPIO66.
1:0	Reserved	R	Reserved
Note:	**BF80_0958h:** GPIO66-71; **BF80_0959h:** GPIO76-78; **BF80_095Ah:** GPIO82-83; **BF80_095Bh:** GPIO 90-92		

EXAMPLES

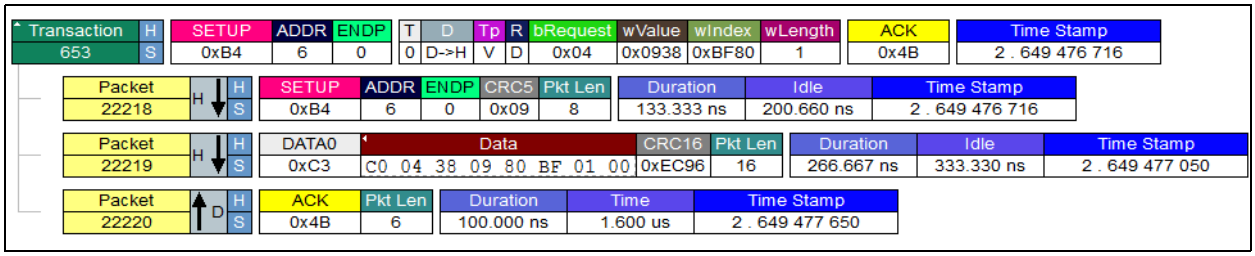
Read the Input State of PF7/GPIO71

1. **Command Phase (SETUP Transaction):** Send the following SETUP Register Read Command to Endpoint 0 of the Hub Feature Controller to read the contents of registers 0xBF80_0938 (PIO[95:64] Input State register) which contains the input state information for PF7/GPIO71 (assuming that the GPIO was already configured as an input in a previous command). See Table 18 and Figure 5.

TABLE 18: REGISTER READ SETUP COMMAND EXAMPLE

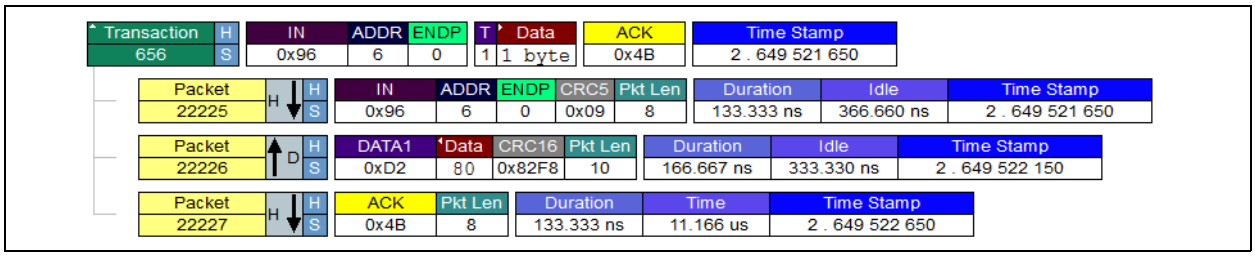
SETUP Parameter	Value	Note
bmRequestType	0xC0	—
bRequest	0x04	—
wValue	0x0938	Last four bytes of the register address
wIndex	0xBF80	First four bytes of the register address
wLength	0x0001	One register to be read

FIGURE 5: REGISTER READ SETUP TRANSACTION EXAMPLE



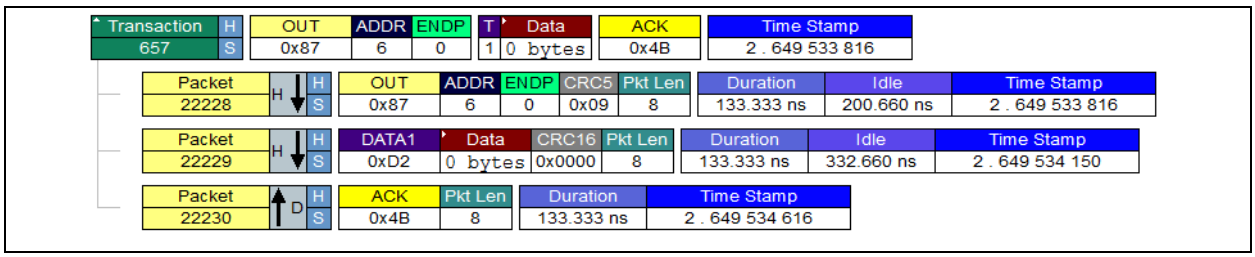
2. **Data Phase (IN Transaction):** The Hub Feature Controller sends the data bytes of length wLength starting from the specified address after receiving an IN packet. The returned value is 0x80, which indicates that PF7/GPIO71 is high. See Figure 6.

FIGURE 6: REGISTER READ IN TRANSACTION EXAMPLE



3. **Status Phase (OUT Transaction):** The Host sends an OUT packet to complete the USB Transfer. The Hub Feature Controller responds with a zero-length data packet. Refer to Figure 7.

FIGURE 7: REGISTER READ OUT TRANSACTION EXAMPLE



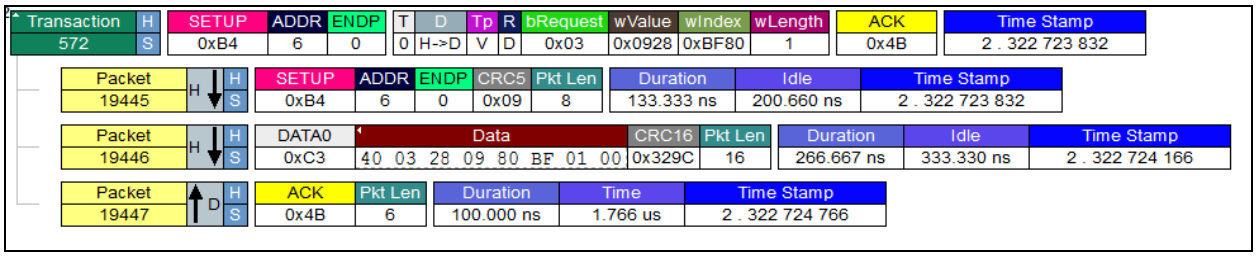
Write Registers to Set PF7/GPIO71 Output State as High

1. **Command Phase (SETUP Transaction):** Send the following SETUP Register Write Command to Endpoint 0 of the Hub Feature Controller to write the contents of register 0xBF80_0928 (PIO[95:64] Output State register). In this example, PF7/GPIO71 is set high (assuming that the GPIO was already configured as an output in a previous command). See Table 19 and Figure 8.

TABLE 19: REGISTER WRITE SETUP COMMAND EXAMPLE

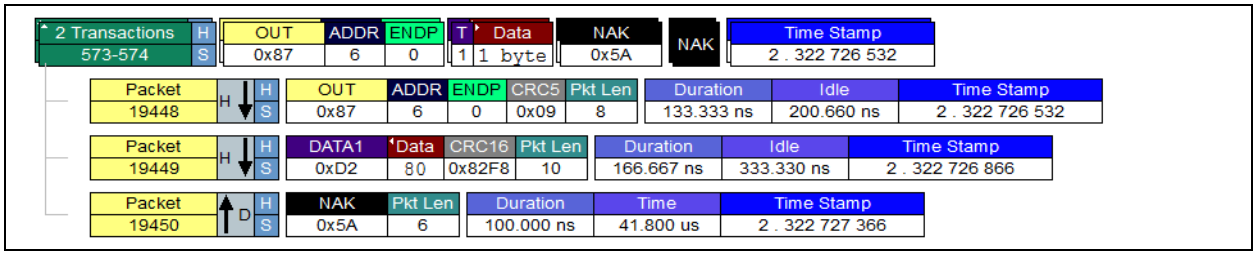
SETUP Parameter	Value	Note
bmRequestType	0x40	—
bRequest	0x03	—
wValue	0x0928	Last 4 bytes of the register address
wIndex	0xBF80	First 4 bytes of the register address
wLength	0x0001	One register is to be read

FIGURE 8: REGISTER WRITE SETUP TRANSACTION EXAMPLE



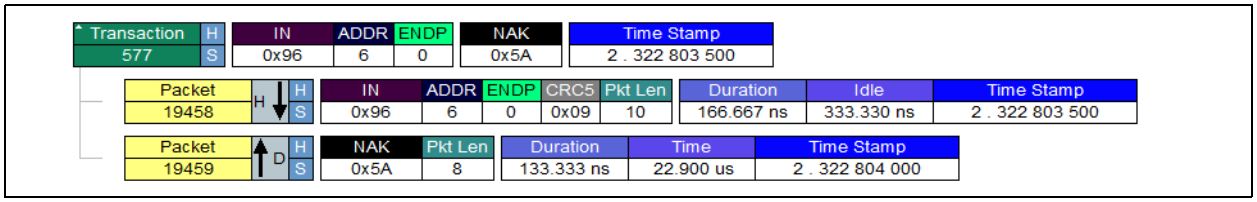
2. **Data Phase (OUT Transaction):** The Host sends the data byte to set 0xBF80_0928 = 0x80 from the specified address after sending the OUT packet to set the PF7/GPIO71 output as high. Refer to Figure 9.

FIGURE 9: REGISTER WRITE OUT TRANSACTION EXAMPLE



3. **Status Phase (OUT Transaction):** The Host sends an IN packet to complete the USB Transfer. The Hub Feature Controller responds with a zero-length data packet. See Figure 10.

FIGURE 10: REGISTER WRITE IN TRANSACTION EXAMPLE



GPIO DEFAULT STATES

Many applications may require a particular default state (such as input with a pull-up or pull-down enabled, or output with drive high or drive low). If particular GPIO default states are required, these can be assigned to the pin via OTP configuration patch.

Use the following simplified guide to manually create an OTP patch.

Step 1: String together as many register WRITE_BYTE, SET_BIT, or CLEAR_BIT commands as required for the application. See Table 20, Table 21, and Table 22.

The WRITE_BYTE command sets all bits within the register to the exact value selected.

TABLE 20: WRITE_BYTE COMMAND

Set Address Command	Register Address	CMD	Length	Value(s)
80h	XXh XXh XXh XXh	FEh 00h	01h	XXh

The SET_BIT command on works as a mask which sets the bits = 1 within the register and leaves bits = 0 untouched.

TABLE 21: SET_BIT COMMAND

Set Address Command	Register Address	CMD	Length	Value(s)
80h	XXh XXh XXh XXh	FEh 01h	01h	XXh

The CLEAR_BIT command on works as a mask which clears the bits = 1 within the register and leaves bits = 0 untouched.

TABLE 22: CLEAR_BIT COMMAND

Set Address Command	Register Address	CMD	Length	Value(s)
80h	XXh XXh XXh XXh	FEh 02h	01h	XXh

Note: Refer to *AN2935 Configuration of USB7202/USB7206/USB725x* for a complete explanation on the OTP command structures.

Step 2: Add one “FFh” byte at the end of all of the commands to terminate the sequence of commands.

Step 3: The patch can be saved in basic binary format with a ‘.cfg’ file extension

Step 4: Program configuration file to the hub using MPLAB Connect Configurator tool. See Example 1.

EXAMPLE 1: SET GPIO 70 TO OUTPUT AND DRIVE HIGH BY DEFAULT FOR USB72XX

USB72xx GPIO70 Output Enable Register is **BF80_0908h**, and bit offset is 6. (Hence, 40h is used to select only bit 6 for modification.) Use the SET_BIT command to ensure that other bits within the register are not affected.

```
80 BF 80 09 08 FE 01 01 40
```

USB72xx GPIO1 Output State Register is **BF80_0928h**, and bit offset is 6. (Hence, 40h is used to select only bit 6 for modification.) Use the SET_BIT command to ensure that other bits within the register are not affected.

```
80 BF 80 09 28 FE 01 01 40
```

Complete OTP Patch binary data with ‘FFh’ byte to end the series of commands:

```
80 BF 80 09 08 FE 01 01 40 80 BF 80 09 28 FE 01 01 40 FF
```

Note: The order of these two register commands is not critical. The end result is the same if they are reversed.

GPIO CONTROL VIA SMBUS INTERFACE

GPIOs may be controlled from an embedded MCU/SoC through the hub SMBus slave interface. Note that the hub must be specifically configured to enable the SMBus slave interface. Not all configuration modes have an available SMBus slave interface.

The GPIOs can be controlled in either the Configuration stage (SOC_CFG) or during the hub runtime stage, after issuing the AAh 56h 00h command within the SOC_CFG stage, which instructs the hub to enter the active runtime stage with the SMBus slave interface still active. The SMBus address during SOC_CFG is 2Dh, while the address during runtime is 2Ch.See Example 2 and Example 3.

Note: OTP memory is loaded after the SOC_CFG stage, while the hub is transitioning into the runtime stage. Any changes made to registers will be overwritten if those same registers are manipulated in OTP configuration. Refer to *AN2935 Configuration of USB7202/USB7206/USB725x* for a complete explanation on the SMBus command structures.

EXAMPLE 2: USB72XX SET GPIO 70 TO OUTPUT AND DRIVE HIGH (DURING RUNTIME STAGE)

Send 'USB Attach with SMBus' command to exit configuration stage and enter hub runtime stage

`5A AA 56 00`, Slave address 2Dh

GPIO1 Direction Register is **BF80_0908h**, and bit offset is 6.

`58 00 00 07 00 01 BF 80 09 28 40` // Write BF80_0908h = 40h, Slave address 2Ch

`58 99 37 00` // Register Access Command, Slave address 2Ch

GPIO1 Output Register is **BF80_0928h** and bit offset is 6.

`58 00 00 07 00 01 BF 80 09 28 40` // Write BF80_0928h = 40h, Slave address 2Ch

`58 99 37 00` // Register Access Command, Slave address 2Ch

EXAMPLE 3: USB5806 SET GPIO 71 TO INPUT AND READ INPUT STATE (DURING RUNTIME STAGE)

Send 'USB Attach with SMBus' command to exit configuration stage and enter hub runtime stage

`5A AA 56 00`, Slave address 2Dh

GPIO1 Direction Register is **BF80_0918h** and bit offset is 7.

`58 00 00 07 00 01 BF 80 09 18 80` // Write BF80_0918h = 80h, Slave address 2Ch

`58 99 37 00` // Register Access Command, Slave address 2Ch

GPIO1 Input Register is **BF80_0938h** and bit offset is 7.

`58 00 00 06 01 01 BF 80 09 38` // Read BF80_0938h, Slave address 2Ch

`58 99 37 00` // Register Access Command, Slave address 2Ch

`58 00 06` // Return value is placed in hub memory offset 00h 06h

`59 80 XX` // Read return value, Slave address 2Ch, Ignore the 80h Byte, XXh is the returned value.

APPENDIX A: APPLICATION NOTE REVISION HISTORY

TABLE A-1: REVISION HISTORY

Revision Level and Date	Section/Figure/Entry	Correction
DS00002932A (3-18-2019)	All	Initial release

THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website 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, and Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, and 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 website 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 website at: http://microchip.com/support

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

ISBN: 978-1-5224-4282-0

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820