

USB 2.0 to I²C/SPI/UART Protocol Converter with GPIO

The MCP2222 is the next-generation version of the MCP2221A and includes a USB-to-SPI interface in addition to USB-to-UART and USB-to-I²C interfaces.

USB Interface

- USB 2.0 Full-Speed Compliant Interface Operating at 12 Mbps
- Supports 3.3V or 5V Operation
- Enumerates as a USB Composite Device:
 - CDC (Communications Device Class) - ACM (Abstract Control Model) class for USB-to-UART communication
 - Vendor class for USB-to-I²C, USB-to-SPI, GPIO control, and device configuration
- Supports Bus-Powered and Self-Powered Operation
- Configurable Vendor IDs (VIDs), Product IDs (PIDs), USB String Descriptors and Serial Number

Utility Software Support

Microchip provides the following tools to configure and use the MCP2222:

- GUI-based Utility With Device Configurator, I²C Terminal and SPI Terminal Support
 - Command Line Interface
 - A 'C' Library Example
- Note:** For quick adaptation of the MCP2222, an MCP2222 breakout board can be used.

UART Interface

- USB-to-UART Communication Through USB CDC-ACM Class Interface
- Supports Runtime Configuration Through the `SET_LINE_CODING` Command, Including:
 - Supported baud rate range: 2,000 bps to 1 Mbps
 - Standard baud rates: 2,400 bps to 921,600 bps
- UART Pins Include:
 - Default pins: TX and RX
 - Optional hardware flow control: RTS and CTS
 - Optional activity indication pins: RXA and TXA for RX and TX activity

I²C Host Interface

- USB-to-I²C Communication Through Vendor Class Interface
- Supports Clock Speeds of: 100 kHz, 400 kHz, and 1 MHz
- Supports 7-bit Addressing

Note: The device may support selected SMBus™ host functions; however, it does not support all SMBus features, including PEC (Packet Error Checking) and ARA (Alert Response Address).

SPI Interface

- USB-to-SPI Communication Through the Vendor Class Interface
- Supports all Four SPI Modes:
 - Supports SPI clock frequencies of up to 12 MHz
 - Supports up to four configurable chip selects

GPIO Interface

The MCP2222 provides six configurable I/O pins. By default, each pin is configured according to its factory-default function, but it can be reconfigured as its respective alternate function pin.

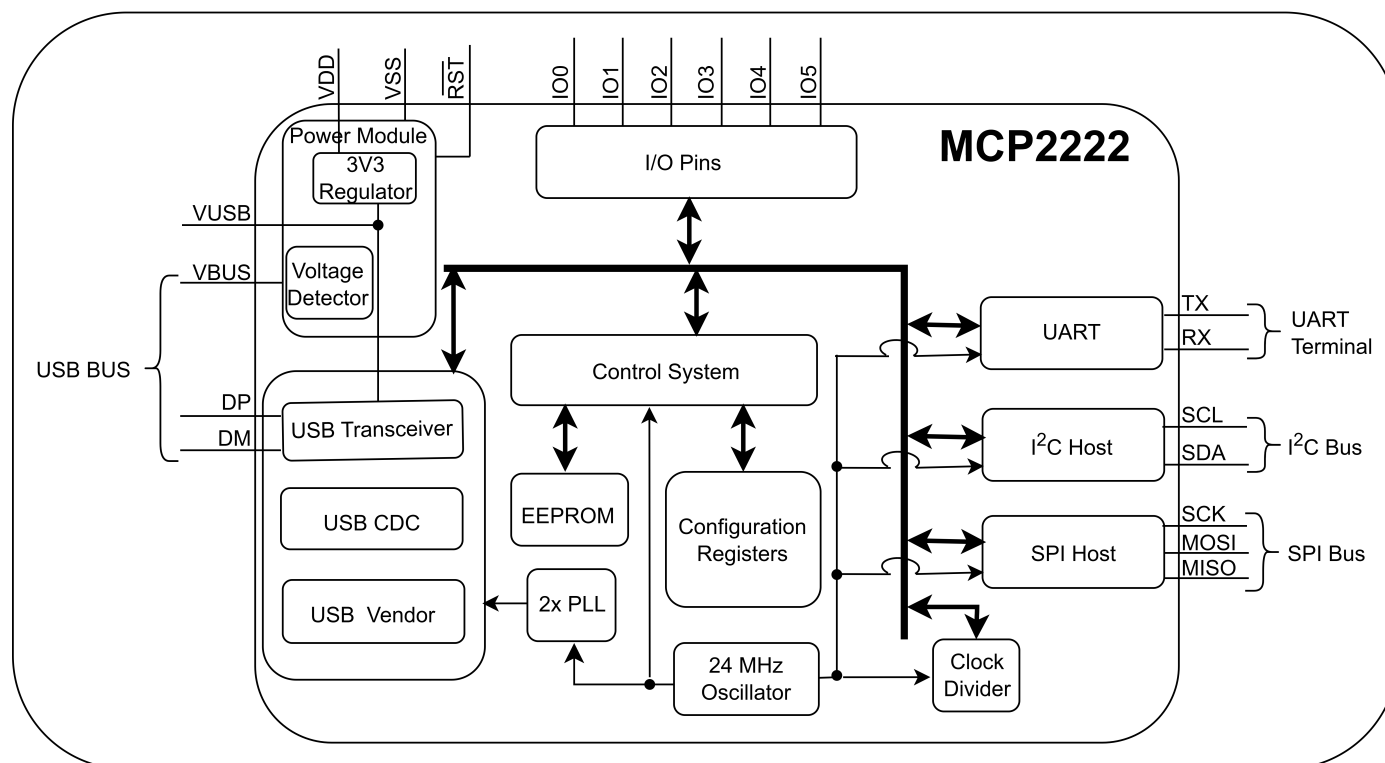
Pins configured as GPIO can be controlled or read through the vendor class interface.

Table 1. Factory Default and Alternate Functions

IO0	GPIO0 , Chip-select 3 (CS3), Alert 0 (ALRT0), Clear-to-send (CTS)
IO1	GPIO1 , Chip-select 2 (CS2), I ² C Activity (I2CA), Request-to-send (RTS)
IO2	GPIO2 , Chip-select 1 (CS1), Alert 1 (ALRT1), RX Activity (RXA)
IO3	Clockout (CLKO) , TX Activity (TXA), I ² C Timeout (I2CTO), GPIO3
IO4	Chip-select 0 (CS0) , USB Enumerated (USBCFG), Alert 2 (ALRT2), GPIO4
IO5	Alert 3 (ALRT3) , USB Suspend (USBSPND), SPI Activity (SPIA), GPIO5

Note: Functions in **bold** are factory-default functions; the rest are alternate functions.

MCP2222 - Block Diagram



MCP2222 - Pinout

Figure 1. 20-Pin SSOP

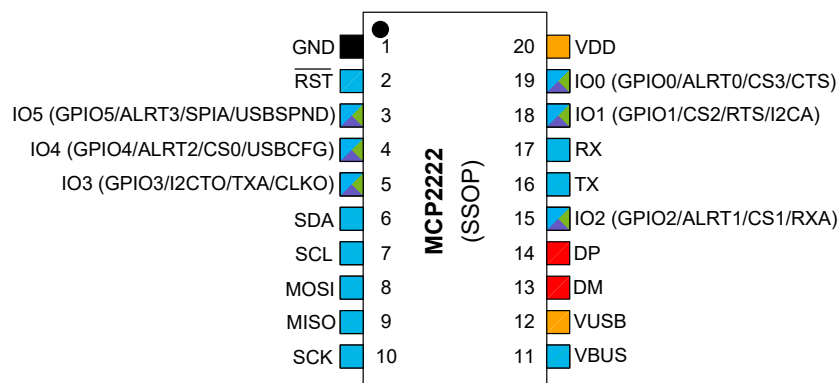


Figure 2. 20-Pin VQFN

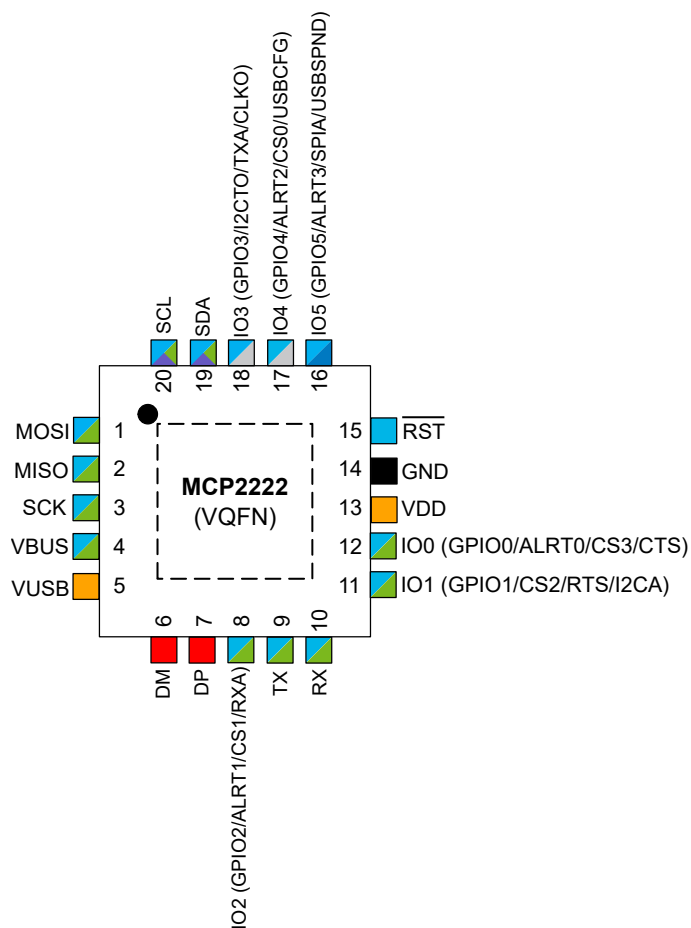


Figure 3. Pinout Legend**Power**

-  Input supply
-  Ground
-  PIN on VDD power domain
-  PIN on VUSB power domain

Functionality

-  Single Function
-  Multiple Functions
-  Multiple Functions

Table of Contents

USB 2.0 to I ² C/SPI/UART Protocol Converter with GPIO.....	1
USB Interface.....	1
Utility Software Support.....	1
UART Interface.....	1
I ² C Host Interface.....	1
SPI Interface.....	2
GPIO Interface	2
MCP2222 - Block Diagram.....	2
MCP2222 - Pinout.....	3
1. Functional Overview.....	7
1.1. Supported Operating Systems.....	8
1.2. Power and Reset Module.....	8
1.3. I/O Pin Module.....	11
1.4. Configuration Registers.....	12
1.5. USB Module.....	12
1.6. USB Transceiver.....	13
1.7. UART Module.....	13
1.8. I ² C Host Module.....	15
1.9. SPI Host Module.....	16
1.10. GPIO Module.....	16
1.11. Clock Reference Module.....	16
1.12. Alert or Interrupt Module.....	17
2. USB Enumeration Process.....	18
2.1. Drivers Required on the USB Host.....	18
2.2. USB CDC-ACM Class Device.....	18
2.3. USB Vendor Class Device.....	19
2.4. USB Re-Enumeration.....	19
3. Device Configuration.....	20
3.1. Device Configuration on Power-up or on Reset.....	21
3.2. USB Parameters.....	21
3.3. I/O Pins Settings.....	21
3.4. Peripheral Settings.....	22
3.5. Configuration Protection Setting.....	24
3.6. Device Configuration Commands and Their Responses	24
3.7. Device Configuration Command Usage.....	40
3.8. Register Summary.....	45
4. UART COMMUNICATION.....	55
4.1. UART Interface Commands and Their Responses	55
5. I ² C COMMUNICATION.....	60
5.1. I ² C Interface Commands and Their Responses.....	60
6. SPI Communication.....	70

6.1. SPI Interface Commands and Their Responses.....	70
7. GPIO CONTROL.....	78
7.1. GPIO Interface Commands and Their Responses.....	78
8. Electrical Characteristics.....	85
8.1. Disclaimer.....	85
8.2. Electrical Specification	85
8.3. Standard Operating Conditions.....	85
9. Ordering Information.....	95
9.1. Product Identification System.....	95
9.2. Package Drawings.....	96
10. Data Sheet Revision History.....	103
Microchip Information.....	104
Trademarks.....	104
Legal Notice.....	104
Microchip Devices Code Protection Feature.....	104

1. Functional Overview

The MCP2222 is a USB-to-UART, USB-to-I²C, and USB-to-SPI bridge device that enables USB connectivity between USB host systems and peripheral devices with legacy UART, I²C, and/or SPI interfaces. The device integrates USB termination resistors and a USB oscillator, reducing the number of required external components.

Note: The MCP2222 is configured through the USB interface, while its UART, I²C, and SPI interfaces enable communication only with external devices. These interfaces cannot be used for the MCP2222 configuration.

The MCP2222 provides six multifunction I/O pins that can be configured to support the following functions:

- General purpose I/O functionality: GPIO[5:0]
- USB status indication: USBCFG, USBSPND
- Clock output: CLKO
- UART hardware flow control: CTS, RTS
- Communication activity indication: TXA, RXA, I2CA, SPIA
- I²C clock low timeout indication: I2CTO
- SPI Chip Select outputs: CS[3:0]
- Interrupt/alert inputs: ALRT[3:0]

The pin functions are summarized in *Table 1-1* and are described in detail in the *I/O Pin Settings* section.

Table 1-1. Pinout Description

Pin Name	SSOP	QFN	Pin Type	Factory Default Function	Alternate Functions
IO0	19	12	I/O	GPIO0 - General purpose I/O 0; input with pull-up enabled	CS3 - SPI chip-select 3 output. ALRT0 - Interrupt0/Alert0 input. CTS - UART clear-to-send input.
IO1	18	11	I/O	GPIO1 - General purpose I/O 1; input with pull-up enabled	CS2 - SPI chip-select 2 output. I2CA - I ² C activity output; open-drain. RTS - UART request-to-send output.
IO2	15	8	I/O	GPIO2 - General purpose I/O 2; input with pull-up enabled	CS1 - SPI chip-select 1 output. ALRT1 - Interrupt1/Alert1 input. RXA - RX activity output; open-drain.
IO3	5	18	I/O	CLKO - Clock reference output; 4 MHz	TXA - TX activity output; open-drain. I2CTO - I ² C clock low timeout output. GPIO3 - General purpose I/O 3.
IO4	4	17	I/O	CS0 - SPI Chip-select 0 output; active low	USBCFG - USB enumerated indication; open-drain. ALRT2 - Interrupt2/Alert2 input. GPIO4 - General purpose I/O 4.
IO5	3	16	I/O	ALRT3 - Interrupt3/Alert3 input; pull-up enabled, falling edge detection	SPIA - SPI activity output. USBSPND - Indicates USB host is suspended. GPIO5 - General purpose I/O 5.
RX	17	10	I	UART RX input	N/A
TX	16	9	O	UART TX output	N/A

Table 1-1. Pinout Description (continued)

Pin Name	SSOP	QFN	Pin Type	Factory Default Function	Alternate Functions
SCL	7	20	I/O	I ² C clock	N/A
SDA	6	19	I/O	I ² C data	N/A
SCK	10	3	O	SPI clock output	N/A
MOSI	8	1	O	SPI data Output	N/A
MISO	9	2	I	SPI data Input	N/A
DM	13	6	USB	USB data minus (D-)	N/A
DP	14	7	USB	USB data plus (D+)	N/A
V _{USB}	12	5	P	USB transceiver power pin. When the V _{DD} voltage is 5V, the pin is connected internally to a 3.3V LDO (externally, a high-quality ceramic bypass capacitor needs to be connected). When the V _{DD} voltage is 3.3V, the pin needs to be connected to 3.3V externally.	N/A
V _{BUS}	11	4	I	USB Bus voltage sense input	N/A
RST	2	15	I	RESET input	N/A
GND	1	14	P	Ground	N/A
V _{DD}	20	13	P	Device power supply, 3V or 5V	N/A
EP	—	21	—	Exposed Thermal Pad (EP)	Connect to GND Rail

1.1. Supported Operating Systems

The MCP2222 supports the following operating systems:

- Windows®—8, 8.1, 10 and 11
- Linux®— distribution with CDC and vendor class support
- MacOS®—10.7 and later

1.2. Power and Reset Module

1.2.1. Power Options

The MCP2222 supports the following power configurations:

- USB bus-powered operation with 5V or derived 3.3V
- Self-powered operation with a 3.3V or 5V supply

The MCP2222 includes an internal 3.3V Low-Dropout (LDO) regulator that powers the USB transceiver and DP/DM pins when the V_{DD} voltage is 5V. When the V_{DD} voltage is 3.3V, the LDO is switched off, and a 3.3V supply must be connected externally to the V_{USB} pin to power the USB transceiver.

The device is held in the Reset state when the voltage on the V_{DD} pin is below 2.85V.

The MCP2222 device can be powered using the USB bus power or any available 3.3V or 5V system rail. The voltage applied to V_{DD} determines the voltage levels on the I/O pins, as well as on the UART, I²C, and SPI pins. When V_{DD} is 5V, all these pins will have a logical '1' of approximately 5V, with the variations specified in the DC Characteristics section.

Although USB provides a 5V bus supply, the USB transceiver requires 3.3V for signaling on the DM and DP lines. The voltage on the DM and DP pins depends on the voltage of the V_{USB} pin. If V_{DD} is greater or equal to 5V, the internal 3.3V LDO regulator is enabled and provides 3.3V to the V_{USB} pin. In this case, a bypass capacitor must be connected externally to the V_{USB} pin. If V_{DD} is 3.3V, a 3.3V supply must be connected externally to the V_{USB} pin.

USB 2.0 devices are required to support Low-Power Suspend mode. During USB Suspend mode, devices must limit current consumption from the 5V USB BUS to no more than 500 μ A (or 2.5 mA for high-powered devices that are remote wake-up capable).

The USB host places a USB device in Suspend mode by stopping USB traffic to that device for more than 3 ms.

1.2.2. Powering the MCP2222 With USB Bus Power (5V Supply)

In USB bus-powered operation, application power is supplied from the USB bus. This is the simplest power configuration for the MCP2222.

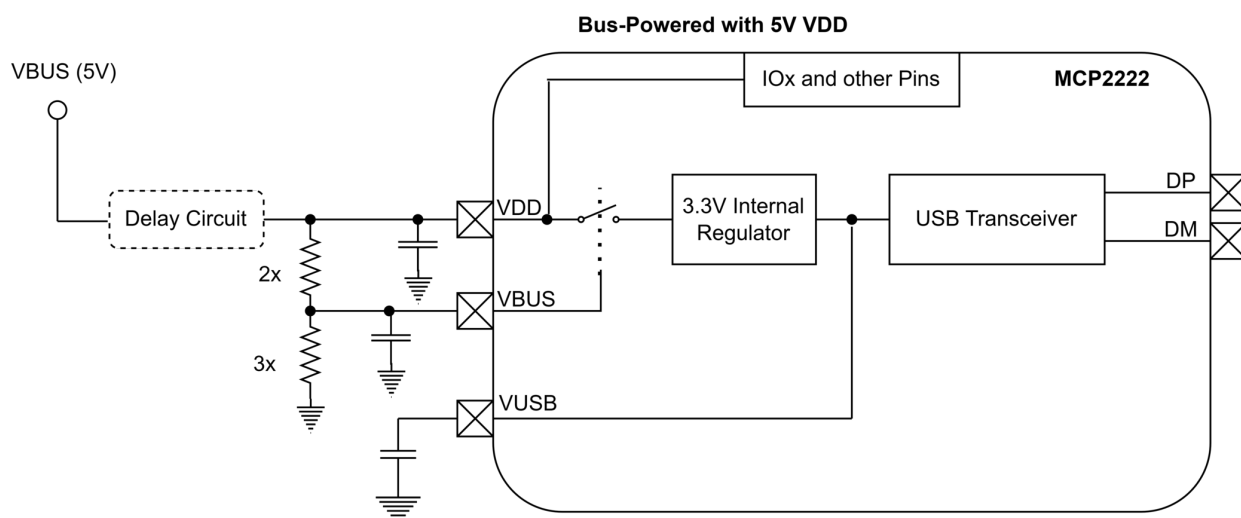
To meet USB 2.0 inrush current requirements, the total effective capacitance appearing across V_{BUS} and ground must not exceed 10 μ F. If the capacitance exceeds this value, an inrush current-limiting circuit is required. Refer to the latest “Universal Serial Bus Specification” for additional information on limiting inrush current.

During USB Suspend mode, the DP or DM pull-up resistor must remain active, which contributes to the allowable suspend current budget (500 μ A or 2.5 mA). The V_{USB} pin requires an external ceramic bypass capacitor with a value between 0.22 and 0.47 μ F.

To power the device with 5V USB bus supply, the V_{BUS} supply from the USB receptacle is connected directly, or through a start-up delay circuit, to the V_{DD} pin. The nominal 5V V_{BUS} supply powers the device, and the internal 3.3V regulator powers the USB transceiver.

Figure 1-1 below shows a typical circuit in which the MCP2222 is powered from USB bus power and its internal LDO supplies 3.3V to the USB transceiver. When the internal LDO is used, a capacitor must be connected to the V_{USB} pin.

Figure 1-1. Power Options With 5V Bus Power

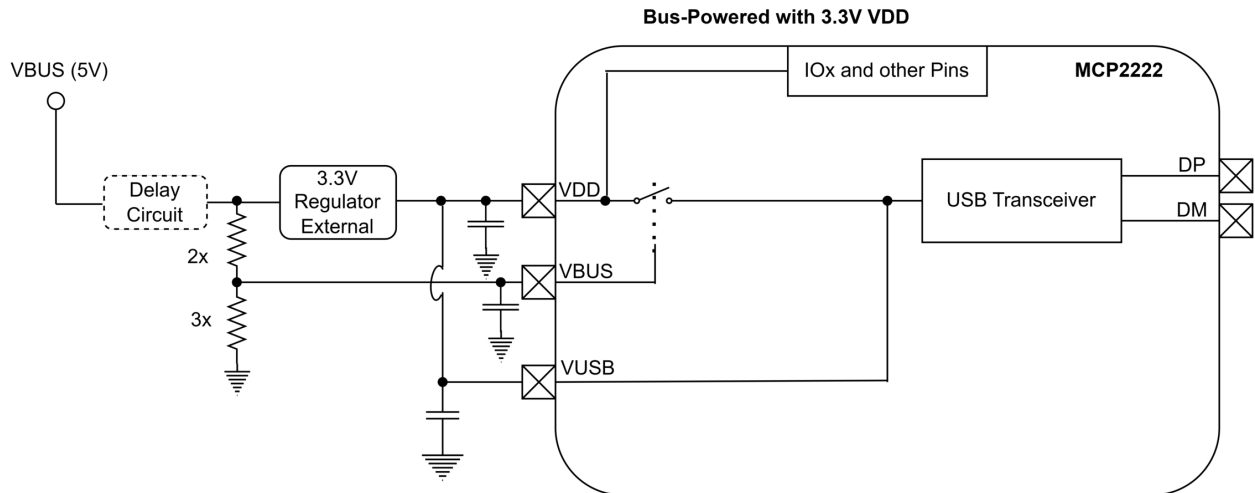


1.2.3. Powering the MCP2222 Using USB Bus Power With an External 3.3V Regulator

Some applications require a 3.3V supply for their devices. To derive 3.3V from the USB bus, an external 3.3V regulator must be used.

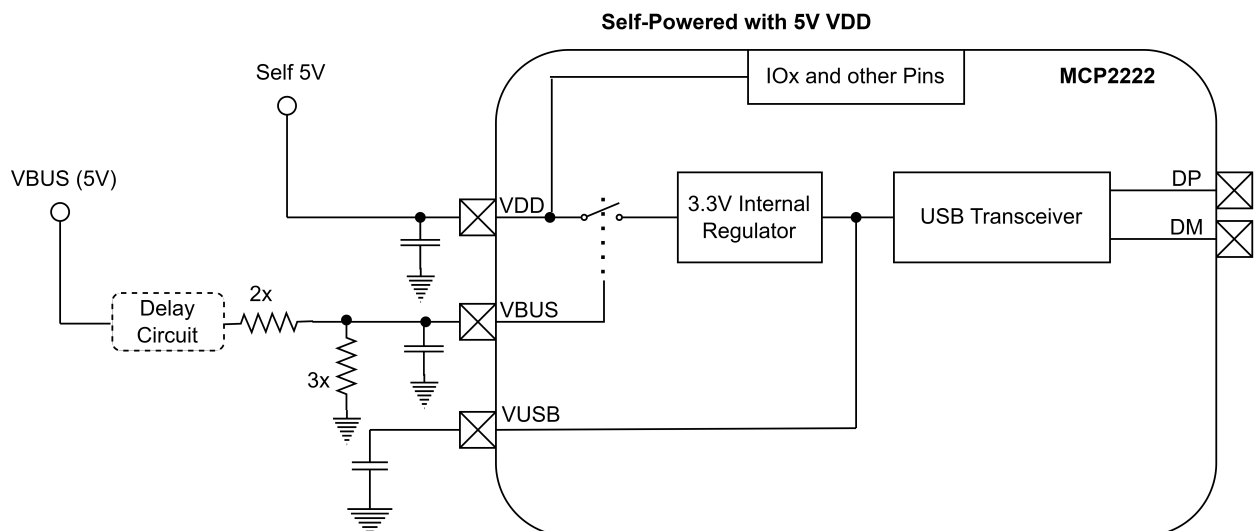
To power the MCP2222 with 3.3V derived from USB bus power, the V_{BUS} supply from the USB receptacle must be connected directly, or through a start-up delay circuit, to the input of an external 3.3V regulator. The regulator output is connected to the MCP2222 V_{DD} pin.

In this configuration, the V_{USB} pin must also be externally connected to a 3.3V supply to provide 3.3V to the USB transceiver.

Figure 1-2. Power Options With 3.3V Bus Power**1.2.4. Powering the MCP2222 Using a 5V System Rail (Self-Powered)**

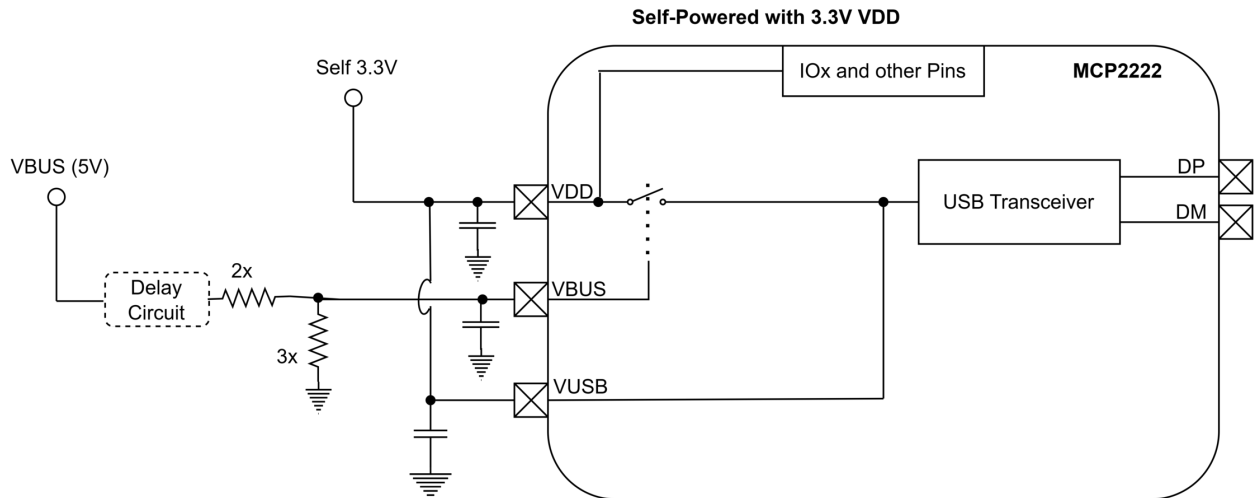
Many embedded systems use a 5V rail to power devices. The MCP2222 can be powered from this 5V supply. When powered from a 5V rail, the internal LDO provides 3.3V to the USB transceiver.

For self-powered applications, the V_{BUS} pin must be used to detect the presence of the USB bus. In this configuration, V_{BUS} should be connected to the USB VBUS supply through a 3:2 resistor divider, as shown in Figure 1-3.

Figure 1-3. Power Options with a 5V Rail**1.2.5. Powering the MCP2222 Using a 3.3V System Rail (Self-Powered)**

The MCP2222 can also be powered from an existing 3.3V system rail. In this configuration, both V_{DD} and V_{USB} are connected to the 3.3V rail.

For self-powered applications, the V_{BUS} pin must be used to detect the presence of the USB bus. V_{BUS} should be connected to the USB VBUS supply through a 3:2 resistor divider, as shown in Figure 1-4.

Figure 1-4. Power Options with 3.3V Rail

1.2.6. RESET ($\overline{\text{RST}}$) Pin

The RST pin provides an option to reset the device externally; when it is active-low, holding the RST pin resets the device. The device then remains in Reset until the required operating conditions are met.

1.3. I/O Pin Module

The I/O Pin module includes an I/O multiplexer that allows various device functions to be assigned to the multifunction pins. The MCP2222 configuration utility can be used to configure these pin functions.

The multifunction I/O pins support the following configurations:

- GPIOx — An I/O pin can be configured as a general-purpose digital input or output
- ALRTx — One of four I/O pins can be configured as an alert or interrupt input. These inputs can wake the MCP2222 and trigger an interrupt event to the USB host when remote wake-up is enabled.
- USBSPND — Indicates the USB suspend state
- USBCFG — Indicates USB configuration status
- CLK0 — Provides a digital clock output. The factory-default frequency is 4 MHz.
- RTS — UART Request to Send output
- CTS — UART Clear to Send input
- RXA — Indicates UART receive activity
- TXA — Indicates UART transmit activity
- I2CA — Indicates I²C bus activity
- SPIA — Indicates SPI bus activity
- I2CTO — Indicates an I²C clock low timeout condition
- CSx — Up to four I/O pins can be configured as SPI chip-select outputs

Input pins include an internal pull-up option, which can be enabled or disabled. Output pins support an open-drain feature that can be enabled or disabled, except for the CLK0 and CSx functions.

1.4. Configuration Registers

The Configuration registers store and manage the MCP2222 device settings, including access protection. The Nonvolatile Memory (NVM) is used to store the power-up/RESET and the factory-default settings.

At power-up or Reset, the module loads the configuration settings from NVM into Volatile memory. These settings define the active device configuration, including the VID, PID, serial number, manufacturer name, product name, and I/O settings.

The user can read or modify both volatile and nonvolatile memory settings through the USB vendor interface.

1.5. USB Module

The MCP2222 includes a full-speed USB 2.0 compliant USB interface. It implements the CDC, ACM and Vendor USB classes.

The MCP2222 enumerates as a composite USB device and supports the following interfaces:

- CDC-ACM class interface: Provides USB-to-UART communication
- Vendor class interface: Provides USB-to-I²C communication, USB-to-SPI communication, GPIO control, and device configuration

The device also supports:

- Configurable VID and PID assignments
- Configurable string descriptors
- Bus-powered or self-powered operation

1.5.1. USB Descriptors

USB descriptors are stored in the MCP2222 and can be modified using the configuration tools.

The device enumerates with these descriptors:

- Vendor ID (VID)
- Product ID (PID)
- Self-powered or bus-powered
- Remote wake-up capable or not
- Enumerates with or without a serial number
- The maximum USB bus current
- Product name
- Manufacturer name
- Serial number

These descriptors can be customized using the Microchip configuration utility or by using a custom application that sends the appropriate device configuration commands to store the new descriptor data in the device.

1.5.2. USB Events

The MCP2222 supports signaling for USB-related events, including USB suspend and resume.

USB suspend and resume signals are used for power management of the MCP2222. The device enters Suspend mode when suspend signaling is detected on the USB bus or when there is no USB activity on the bus for 3 ms.

The MCP2222 exits Suspend mode when one of the following events occurs:

- USB resume signaling is detected or generated
- A USB Reset signal is detected
- A device Reset occurs

If IO5 is configured for the USBSPND function, it indicates the USB suspend or resume state.

If IO4 is configured for the USBCFG function, it indicates successful USB enumeration.

This signal allows the application to enter a low-power state when USB communication is suspended and return to an active state when USB communication resumes.

1.5.3. Remote Wake-up Capability

The MCP2222 supports remote wake-up signalling to the USB host. Remote wake-up is triggered through any of the four Alert/Interrupt function pins when configured as alert input pins.

To use this feature, one of the ALRT[3:0] pins must be designated as an alert input. The interrupt detector must be configured to detect the required edge.

When the USB host is in Suspended mode, the MCP2222 enters Sleep mode. If the remote wake-up is enabled and any configured ALRT[3:0] pin detects the configured edge, the MCP2222 exits Sleep mode and sends a resume signal to the USB host.

1.6. USB Transceiver

The MCP2222 includes an integrated, on-chip USB 2.0 full-speed transceiver connected internally to the USB module.

When V_{DD} is supplied from a 5V source, the internal 3.3V regulator powers the USB transceiver, and V_{USB} pin should be bypassed with a ceramic capacitor.

When V_{DD} is supplied from a 3.3V source, the V_{USB} pin must be externally connected to the 3.3V supply.

1.7. UART Module

The MCP2222 UART interface consists of the TX and RX data signals, with optional hardware flow control through the RTS and CTS pins when configured on IO0 and IO1, respectively.

The UART interface supports configurable parameters that can be set at runtime through the `SET_LINE_CODING` command:

- Baud Rate
- STOP Bits
- Parity
- Data Width

Supported baud rate range: 2,000 bps to 1 Mbps. Standard baud rates from 2,400 bps to 921,600 bps are listed in Table 1-2.

Note: When the data width is set to five bits, the UART communication with no flow control may not operate properly beyond the baud rate of 921,600 bps.

1.7.1. Baud Rate Rounding Errors

Table 1-2. UART Primary Baud Rates

Desired Rate (bps)	Actual Rate (bps)	% Error
921600	914285.71	-0.8%
2400	2400	0.00%
4800	4800	0.00%

Table 1-2. UART Primary Baud Rates (continued)

Desired Rate (bps)	Actual Rate (bps)	% Error
9600	9600	0.00%
19200	19200	0.00%
38400	38400	0.00%
57600	57588.48	-0.02%
115200	115107.91	-0.08%
230400	230215.83	-0.08%
460800	459330.14	-0.32%



The percentage error is calculated assuming the 24 MHz clock source is stable. Any drift in the source clock frequency would affect the percentage error.

The actual baud rate is calculated as follows:

$$\text{Actual Rate} = \frac{64 * 24 \text{ MHz}}{16 * \text{ceiling}(x)}$$

Where,

$$x = \frac{64 * 24 \text{ MHz}}{16 * \text{DesiredRate}}$$

Example: For 115,200 bps,

$$x = (64 \times 24,000,000) / (16 \times 115,200) = 86.805... \text{ } x_{\text{rounded}} = 87$$

$$\text{Actual Rate} = (64 \times 24,000,000) / (16 \times 87) = 115,107.91 \text{ bps}$$

$$\text{Error} = -0.08\%$$

1.7.2. Custom Baud Rates

Custom baud rates are configured using the USB `SET_LINE_CODING` command. Refer to the *UART Communication* section for additional information.

The baud rate error can be calculated using the equation provided in Section 1.7.1.

1.7.3. Configurable Stop Bits

The MCP2222 supports one and two stop bits.

1.7.4. Configurable Parity

The MCP2222 supports the following parity settings:

- None
- Odd
- Even

1.7.5. Configurable Data Width

The MCP2222 supports UART data widths of 5, 6, 7 and 8 bits.

1.7.6. Hardware Flow Control

The MCP2222 supports hardware flow control when the IO0 and IO1 pins are configured as CTS and RTS, respectively.

To enable UART hardware flow control, configure:

- IO0 as CTS
- IO1 as RTS

1.7.7. UART Receive Activity (RXA)

The RXA function indicates UART receive activity.

When the IO2 pin is configured as RXA, the pin becomes active whenever UART receive activity is detected. This allows the application to provide a visual indication of UART RX traffic.

1.7.8. UART Transmit Activity (TXA)

The TXA function indicates UART transmit activity.

When the IO3 pin is configured as TXA, the pin becomes active whenever UART transmit activity is detected. This allows the application to provide a visual indication of UART TX traffic.

1.8. I²C Host Module

The MCP2222 I²C host interface consists of SCL and SDA, which are the clock and data signals. The I²C host interface supports configurable clock speed and clock low timeout periods.

The I²C host is controlled using the I²C interface commands.

The I²C host may support some SMBus host functions.

Note: The MCP2222 does not support all SMBus features, including Packet Error Checking (PEC) and Alert Response Address (ARA).

The I²C host supports the following features:

- Clock frequency: 100 kHz, 400 kHz, and 1 MHz
- 7-bit Addressing mode
- Single transfers of up to 65,535 bytes
- Clock stretching with a configurable timeout from 1 ms to 250 ms

Clock stretching support allows communication with slower I²C client devices.

All user data transmitted or received over the I²C bus is exchanged with the USB host through the USB vendor-class interface.

1.8.1. I²C Module Factory Default Configuration

The factory-default I²C configuration is:

- Clock frequency: 100 kHz
- Clock stretch timeout: 15 ms

1.8.2. I²C Activity (I2CA)

The I2CA function indicates I²C bus activity.

When the IO1 pin is configured as I2CA, the pin becomes active whenever an I²C transaction occurs. This allows the application to provide a visual indication of I²C traffic.

1.8.3. I²C Clock Low Timeout, I2CTO

The I2CTO function indicates an I²C clock low timeout condition caused by clock stretching. The timeout period is configurable from 1 ms to 250 ms.

When the IO3 pin is configured as I2CTO, the pin becomes active when an I²C clock stretch timeout occurs and remains active until the bus is restored or the MCP2222 is reset.

1.9. SPI Host Module

The MCP2222 SPI host interface consists of MOSI, MISO, SCK, and CS, which are the data-out, data-in, clock, and chip select signals, respectively. It is responsible for SPI traffic generation. The SPI host interface supports configurable clock speed, SPI mode, and delay between data bytes.

The SPI host is controlled using SPI interface commands.

The SPI host supports the following features:

- Clock frequency: Up to 12 Mbps
- SPI modes: All four modes: 0 to 3
- Up to four chip select outputs for communication with multiple SPI client devices
- Single transfers of up to 65,535 bytes
- Configurable delay from 0 μ s to 5000 μ s for:
 - Chip-select assertion to the first data byte
 - Delay between data bytes
 - Last data byte to chip-select deassertion

All user data transmitted or received over the SPI bus is exchanged with the USB host through the USB vendor-class interface.

1.9.1. SPI Module Factory-Default Configuration

The factory-default SPI configuration is:

- Clock frequency: 12 MHz
- SPI mode: Mode 0
- Chip-select configuration: CS0 (only CS0 is configured by default)
- Configurable delay: 0 μ s

1.9.2. SPI Activity (SPIA)

The SPIA function indicates SPI bus activity.

When the IO5 pin is configured as SPIA, the pin becomes active whenever an SPI transaction occurs. This allows the application to provide a visual indication of SPI traffic.

1.10. GPIO Module

The GPIO module allows the multifunction I/O pins to be configured and controlled as general-purpose digital inputs or outputs.

When configured for GPIO operation, the pins may be used as digital inputs or digital outputs.

When configured as an output, the GPIO state can be controlled and can also be configured for open-drain operation.

When configured as an input, the logical state of the pin can be read. The internal pull-up resistor for the input pin can be enabled or disabled.

These GPIO operations can be performed using GPIO Interface commands.

1.11. Clock Reference Module

The IO3 pin supports the Clock Out (CLKO) function. When configured as a CLKO, IO3 outputs a clock signal at the selected frequencies.

The factory-default CLKO frequency is 4 MHz. The output frequency can be one of the following:

- 6 MHz

- 4 MHz
- 2 MHz
- 1 MHz

These output frequencies can be changed through device configuration commands.

1.12. Alert or Interrupt Module

The IO0, IO2, IO4 and IO5 pins can be configured as interrupt or alert inputs. When configured as an alert or interrupt input, the pin operates as a digital input with edge-detection capability. Depending on the configuration, the pin can detect:

- Rising edge
- Falling edge
- Both rising and falling edges

All four alert inputs, ALRT[3:0] include internal pull-up resistors that can be enabled or disabled. The IO5 pin is factory-configured as ALRT3 pin.

When an alert condition is detected on any configured ALRT[3:0] pin, the MCP2222 wakes up from Sleep and can trigger a USB remote wake-up event to the USB host, provided that remote wake-up is enabled.

2. USB Enumeration Process

The MCP2222 enumerates as a USB 2.0 full-speed composite device with dual interfaces:

- USB CDC-ACM (Communication Device Class - Abstract Control Model). The communication class interface is on interface 0 and the data class interface is on interface 1.
- Vendor (Interface 2) provides the following functions:
 - I²C communication and data
 - SPI communication and data
 - GPIO control
 - Device configuration

The device and configuration descriptor information is stored in the Nonvolatile Memory (NVM) and used during USB enumeration. These descriptors include:

- Vendor ID (VID)
- Product ID (PID)
- Manufacturer name
- Product name
- Serial number
- Bus-powered or self-powered configuration
- Remote wake-up enable or disable setting
- Maximum bus current

These descriptor values can be modified using the Microchip configuration tool, Python library, or C library. Refer to Section 3.7.1 *USB Descriptor Modification Steps* for the commands used to modify USB descriptor settings.

2.1. Drivers Required on the USB Host

To use the MCP2222, no custom drivers are required. It works with standard drivers that are installed automatically by the host operating system during device enumeration.

- On Linux® and macOS® systems, the MCP2222 USB CDC-ACM class interface registers with the serial subsystem, and the USB vendor-class interface is registered with USB subsystem and can be accessed from user space through the libUSB library.
- On Windows® systems, the MCP2222 USB CDC-ACM class is assigned to the standard Windows serial driver, and USB vendor-class interface is bound to the WinUSB driver.

2.2. USB CDC-ACM Class Device

The MCP2222 CDC-ACM class interface provides USB-to-UART bridge functionality. When used with Windows-based host systems, COM port assignment behavior should be considered.

When the MCP2222 is connected to a PC for the first time, Windows searches for a suitable driver and creates a registry entry for the device. This registry entry stores information about the USB-to-UART adapter, the associated driver, and the assigned COM port. On subsequent connections to the same PC, Windows uses this registry entry to load the appropriate driver and assign the previously associated COM port number.

COM port numbering is a legacy Windows feature that remains supported for compatibility. Historically, COM ports were physical ports on the computer motherboard, each assigned a unique index number. With the introduction of USB-to-UART adapters, Windows extended the COM port model to support USB serial devices.

During enumeration, the MCP2222 can provide a USB serial number. If a serial number is provided, Windows stores it in the registry and uses it to assign the same COM port number to the device, regardless of which USB port is used. If no serial number is provided, Windows may assign different COM port numbers when the device is connected to different USB ports.

Both behaviors are useful depending on the application. Using a serial number provides consistent COM port assignment across USB ports. Omitting the serial number allows Windows to treat the same device as a separate instance when connected to different USB ports.

By default, the MCP2222 includes a factory-configured unique serial number that may be used during USB enumeration. The Microchip configuration utility can be used to enable, disable, or modify the serial number used for enumeration.

All USB-related settings described in this section are part of the device configuration and are stored in NVM.

2.3. USB Vendor Class Device

Except for the USB-to-UART function, which uses the CDC-ACM class interface, all other MCP2222 functions use the USB vendor class interface. These functions include:

- USB-to-I²C communication
- USB-to-SPI communication
- GPIO control
- Device configuration

On Windows systems, the MCP2222 enumerates as a WinUSB vendor-class device and appears in Device Manager as a USB converter. WinUSB is a generic USB driver provided by Microsoft that allows applications to communicate with USB devices without requiring a custom kernel-mode driver. WinUSB includes the kernel-mode driver, `Winusb.sys`, and the user-mode dynamic link library, `Winusb.dll`.

On macOS and Linux systems, the MCP2222 enumerates as a libusb vendor-class device. libusb is a portable library that provides generic access to USB devices and allows application software to communicate with USB hardware across multiple operating systems, including Linux, macOS, and Windows.

Vendor-class communication typically begins when the USB host writes a command packet to one of the OUT endpoints associated with Interface 2. The MCP2222 uses the following endpoints for vendor-class functions:

- Endpoint 3 and 83: I²C communication
- Endpoint 4 and 84: GPIO control
- Endpoint 5 and 85: Device configuration
- Endpoint 6 and 86: SPI communication

The device response is typically a 2-byte packet, although responses of up to 64 bytes may be received from the corresponding IN endpoint, depending on the command.

2.4. USB Re-Enumeration

The MCP2222 can be re-enumerated without physically disconnecting it from the USB bus. To initiate re-enumeration, issue a device Reset that disables and then re-enables the pull-up on the DP and/or DM USB data lines.

This operation can be performed by issuing the `MCP2222_SoftReset` command supported by the MCP2222 device.

3. Device Configuration

The MCP2222 stores device configuration settings in NVM (EEPROM) and/or volatile memory (SRAM):

- SRAM: Settings stored in SRAM are referred as active settings.
- EEPROM: Settings stored in NVM are persistent; they are copied into SRAM and are made active at every device reset or up at power-up. These settings are referred as power-up settings.

The active settings can be modified using the following commands:

- IO_SetConfiguration: Used to configure an I/O pin at run-time
- I2C_Config: Used to configure I²C parameters at run-time
- SPI_Config: Used to configure SPI parameters at run-time
- SET_LINE_CODING (CDC ACM Class): Used to configure UART parameters at run-time

The power-up settings are factory-defined and can be modified using the appropriate device configuration commands.

The configuration settings are classified into the following categories:

USB parameters include:

- USB Vendor ID and Product ID
- USB power attributes, including requested current and bus-powered or self-powered operation
- USB manufacturer, product, and serial number descriptors

USB parameters can be modified using the following commands:

- CNFG_SetUSBParameters
- CNFG_SetManufacturer
- CNFG_SetSerialNumber
- CNFG_SetProductName

I/O pins settings include:

- I/O pin function designation
- I/O pin direction
- I/O pin logic state for output pins
- Open-drain enable for output pins
- Internal pull-up enable for input pins
- Alert trigger edge selection for alert/interrupt pins
- Clock reference output frequency

The available I/O pin function designations and supported configuration options are described in the I/O Designations and Other Pin Settings table.

I/O settings can be modified using the following command: CNFG_SetIOConfiguration

- Peripheral settings include:
 - UART settings
 - I²C settings
 - SPI settings
- Peripheral settings can be modified using the following commands:
 - CNFG_SetI2CSettings

- CNFG_SetSPISettings

Note: No power-up settings are stored for UART, only the active ones.

- Configuration protection settings:
The MCP2222 provides configuration protection for power-up settings, including USB parameters, I/O pin settings, and peripheral settings. Protection levels are described in *Section 3.5: Configuration Protection Settings*.

The configuration protection level and the password if set, are saved in NVM.

3.1. Device Configuration on Power-up or on Reset

During power-up or a device reset, the MCP2222 loads USB descriptor information, and configures the I/O pins and peripherals, according to the EEPROM settings.

3.2. USB Parameters

USB parameters are stored in NVM and loaded into the USB descriptors on power-up or Reset. Any modification to these parameters takes effect only after the device is power-cycled or reset and re-enumerated.

The following table lists the USB parameters stored in NVM.

Table 3-1. USB Parameters

Parameter	Valid Range	Factory Default	Comments
Vendor ID (wVID)	0x0000 - 0xFFFF	0x04D8	USB Vendor ID.
Product ID (wPID)	0x0000 - 0xFFFF	0x0B15	USB Product ID.
Device Config (bDevCfg)	See the Device Configuration Register description	0x01	Device Configuration Register: Bit 3: Self-powered (1)/bus-powered (0). Bit 2: Remote wake-up capable (1)/not capable (0). Bit 1: Unimplemented. Bit 0: Enumerate with serial number (1)/ without (0).
Maximum Current (bBusCrnt)	0x01 (2 mA) - 0xFA (500 mA)	0x32 (100 mA)	Bus current, Maximum current÷2, in 8-bit hexadecimal format
Manufacturer string length (bMfrLen)	0x02 - 0x3C	0x32	Length of the manufacturer name in bytes
Manufacturer string (vMfrName)	Up to 30 ASCII characters	Microchip Technology Inc.	Manufacturer name, in UTF-16LE format
Product string length (bPrdLen)	0x02 - 0x3C	0x24	Length of the product name in bytes
Product string (vPrdName)	Up to 30 ASCII characters	MCP2222 USB Bridge	Product name, in UTF-16LE format
Serial number length (bSrlLen)	0x02 - 0x3C	0x30	Length of the serial number in bytes
Serial number (vSrlNum)	Up to 30 ASCII characters	Unique 24-char code	Serial number, in UTF-16LE format

3.3. I/O Pins Settings

Each I/O pin has an associated IOx settings register, which controls the following parameters for the associated I/O pin:

- Pin function designation
- Pin direction
- Default output state
- Pull-up enable or disable

- Open-drain enable or disable
- Alert edge selection or clock output frequency, where applicable

The supported settings for each I/O pin designation are shown in Table 3-2.

Table 3-2. I/O Designations and Other Pin Settings

I/O Designation	I/O Direction	Open-Drain	Pin Output State	Pull-up	Trigger Edge/Clock frequency
GPIO [5:0]	Input/Output	Enable/Disable (if output)	'0'/'1' (if output)	Enable/Disable (if input)	—
ALRT [3:0]	Input	—	—	Enable/Disable	Falling, rising, both or none
CS [3:0]	Output	—	SPI controlled	—	—
CLKO	Output	—	Clock Module Controlled	—	6 MHz, 4MHz, 2MHz or 1MHz
USBCFG	Output	Enable/Disable	USB controlled	—	—
USBSPND	Output	Enable/Disable	USB controlled	—	—
RTS	Output	Enable/Disable	UART controlled	—	—
CTS	Input	—	—	Disable	—
RXA	Output	Enable/Disable	UART controlled	—	—
TXA	Output	Enable/Disable	UART controlled	—	—
SPIA	Output	Enable/Disable	SPI controlled	—	—
I2CA	Output	Enable/Disable	I ² C controlled	—	—
I2CTO	Output	Enable/Disable	I ² C controlled	—	—

Table 3-3. I/O Pin Settings Map

Byte Index	Register name	Comments
0	IO0SETTING	IO0 pin designation, direction, pull-up enable, open-drain, alert edge, and idle-state selection
1	IO1SETTING	IO1 pin designation, direction, pull-up enable, open-drain, “—”, and idle-state selection
2	IO2SETTING	IO2 pin designation, direction, pull-up enable, open-drain, alert edge, and idle-state selection
3	IO3SETTING	IO3 pin designation, direction, pull-up enable, open-drain, CLKO frequency, and idle-state selection
4	IO4SETTING	IO4 pin designation, direction, pull-up enable, open-drain, alert edge, and idle-state selection
5	IO5SETTING	IO5 pin designation, direction, pull-up enable, open-drain, alert edge, and idle-state selection

The I/O pin power-up settings are made active settings during a device power-up or reset. Changes made to these settings take effect after:

- A power cycle
- A device Reset
- Sending the vendor command, CNFG_SetNVMAAsCurrent

The changes made to the active settings take effect immediately but are lost after a power cycle or a device reset.

3.4. Peripheral Settings

Peripheral power-up settings are made active settings during a device power-up or reset. The changes made to these settings take effect after:

- A power cycle

- A device Reset
- Sending the vendor command, CNFG_SetNVMAAsCurrent

The changes made to active settings take effect immediately but are lost after a power cycle or a device reset.

3.4.1. UART Settings

UART settings include:

- Baud rate
- Number of stop bits
- Parity
- Data width

These settings are not stored in NVM. Each time CDC communication is set up, the UART settings are sent by the USB host using the Set_Line_Coding command.

In addition to the above settings, hardware flow control can also be enabled by configuring the IO0 and IO1 pins as CTS and RTS pins.

Table 3-4. UART Settings Map

Byte Index	Register name	Comments
8-11	dwBaudRate	UART baud rate in 32-bit hexadecimal format
12	bStopBit	Number of stop bits: 0x00 = 1 stop bit; 0x02 = 2 stop bits
13	bParity	Parity selection: 0x00 = none; 0x01 = odd parity; 0x02 = even parity
14	bDataWidth	Data width selection: 0x05 = 5 bits; 0x06 = 6 bits; 0x07 = 7 bits; 0x08 = 8 bits

3.4.2. I²C SETTINGS

I²C settings include:

- Clock speed
- Clock stretch timeout

These settings are shown in Table 3-5.

Table 3-5. I²C Settings Map

Byte Index	Register name	Comments
0	bSpeed	I ² C clock frequency: 0x70 = 100 kHz; 0x16 = 400 kHz; 0x04 = 1 MHz
1	bTimeout	I ² C clock stretch timeout in milliseconds. Valid range: 0x01 to 0xFA , corresponding to 1 ms to 250 ms.

3.4.3. SPI Settings

SPI settings include the following:

- Clock speed
- SPI mode
- Chip-select selection and its active state
- Delay between:
 - Chip-select assertion and to the first data byte
 - Between the data bytes
 - Last data byte and chip-select deassertion

These settings are shown in Table 3-6.

Table 3-6. I/O Pin Settings Map

Byte Index	Register name	Comments
0	bSpeed	SPI clock frequency: 0x30 = 12 MHz; 0x20 = 6 MHz; 0x32 = 3 MHz; 0x22 = 1.5 MHz; 0x34 = 750 kHz; 0x24 = 375 kHz; 0x26 = 187.5 kHz
1	bMode	SPI mode: 0x44 = Mode 0; 0x45 = Mode 1; 0x46 = Mode 2; 0x47 = Mode 3
2	bChipSelect-ActiveState	Chip-select and its active state: 0x00 = CS0 active low; 0x01 = CS0 active high; 0x40 = CS1 active low; 0x41 = CS1 active high; 0x80 = CS2 active low; 0x81 = CS2 active high; 0xC0 = CS3 active low; 0xC1 = CS3 active high
3	wDelay	Delay between chip-select assertion and the first data byte, between two data bytes, and between the last data byte and chip select deassertion: For example: 0x0000 = 0µs; 0x000A = 10 µs; 0x0014 = 20 µs; 0x03E8 = 1000 µs; 0x1388 = 5000 µs

3.5. Configuration Protection Setting

USB parameters, I/O settings, and peripheral settings stored in NVM can be protected from modification by selecting the appropriate protection level. After a power cycle or Reset, the settings stored in NVM are made as active settings.

The MCP2222 supports the following configuration protection levels:

- No Protection – Configuration settings can be modified.
- Password protected (unlocked) – A password is set, but configuration settings can be modified.
- Password protected (locked) – Configuration settings cannot be modified until the device is unlocked with the correct password.
- Permanently locked – Configuration settings cannot be modified, no unlocking is allowed.

3.6. Device Configuration Commands and Their Responses

The MCP2222 supports device configuration through the USB vendor-class interface.

The command and response formats are as given below:

- Command format: bCommandCode + Payload
- Response format: wResponseCode + Payload

The following endpoints are used for configuration commands and responses:

- Endpoint 0x05 (OUT) — Used by the USB host to send configuration commands.
- Endpoint 0x85 (IN) — Used by the USB host to read command responses.

Command	Effect
CNFG_SetUSBParameters	Saves the USB parameters to NVM. The changes are applied during the next enumeration.
CNFG_SetManufacturer	Saves the manufacturer name to NVM. The changes are applied during the next enumeration.
CNFG_SetProductName	Saves the product name to NVM. The changes are applied during the next enumeration.
CNFG_SetSerialNumber	Saves the serial number to NVM. The changes are applied during the next enumeration.
CNFG_GetUSBParameters	Reads the USB parameters from NVM
CNFG_GetManufacturer	Reads the manufacturer name from NVM
CNFG_GetProductName	Reads the product name from NVM

Device Configuration Commands and Their Responses (continued)	
Command	Effect
CNFG_GetSerialNumber	Reads the serial number from NVM
CNFG_SetIOConfiguration	Saves the six I/O pin configuration to NVM
CNFG_GetIOConfiguration	Reads the configuration of the six I/O pins saved in SRAM or NVM
CNFG_SetI2CConfiguration	Saves the I ² C configuration to NVM
CNFG_GetI2CConfiguration	Reads the I ² C configuration from SRAM or NVM
CNFG_SetSPICConfiguration	Saves the SPI configuration to NVM
CNFG_GetSPICConfiguration	Reads the SPI configuration from SRAM or NVM
CNFG_ResetToFactoryDefault	Restores the NVM settings to their factory-default values
CNFG_SetPassword	Enables password protection and saves the password to NVM
CNFG_DisablePassword	Disables password protection
CNFG_LockConfiguration	Locks the password-protected configuration
CNFG_UnlockConfiguration	Unlocks the password-protected configuration
CNFG_ChangePassword	Changes the password and saves the new password to NVM
CNFG_PermanentLockCfg	Permanently locks the device configuration
CNFG_GetCfgProtectionLevel	Reads the current device configuration protection level
CNFG_SetNVMasCurrent	Makes the power-up I/O, I ² C and SPI current settings
CNFG_GetRevID	Reads the device revision ID
CNFG_SoftReset	Resets the device and causes it to re-enumerate

3.6.1. CNFG_SetUSBParameters Command and its Response

The CNFG_SetUSBParameters command (0x81) saves the USB parameters to NVM:

- wVID → 16-bit Vendor ID in hexadecimal format
- wPID → 16-bit Product ID in hexadecimal format
- bPwrSrc-RmtWup-SNEn → Power source, remote wake-up, and serial number enumeration setting:
 - Bit 3: 0 = bus-powered, 1 = self-powered
 - Bit 2: 0 = remote wake-up disabled, 1 = remote wake-up enabled
 - Bit 0: 0 = enumerates without a serial number, 1 = enumerates with a serial number
- **bMaxCurrent** — Maximum bus current:
 - 0x01 = 2 mA
 - 0x02 = 4 mA
 - ...
 - 0xFA = 500 mA

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-7. CNFG_SetUSBParameters - Command

Byte Index	Command		Description
0	0x81	Command code	Saves the device descriptors to NVM

Table 3-7. CNFG_SetUSBParameters - Command (continued)

Byte Index	Command		Description
1	L (wVID)	VID – Lower Byte	Vendor ID: Factory-default is 0x04D8; it can be modified with a valid range of 0x0000 to 0xFFFF
2	H (wVID)	VID – Higher Byte	
3	L (wPID)	PID – Lower Byte	Product ID: Factory-default is 0x0B15; it can be modified with a valid range of 0x0000 to 0xFFFF
4	H (wPID)	PID – Higher Byte	
5	0x00 0x04 0x08 0x0C 0x01 0x05 0x09 0x0D	SrcType-RmtWakUp- EnDisSrlNumEnum: BusPow-RWNo-SNEnNo Bus Pow-RWYes-SNEnNo Self Pow-RWNo-SNEnNo SelfPow-RWYes-SNEnNo BusPow-RWNo-SNEnYes BusPow-RWYes-SNEnYes SelfPow-RWNo-SNEnYes SelfPow-RWYes-SNEnYes	Combines three settings: Bit 3: Bus-powered or self-powered. Bit 2: Wake-Up supported or not. Bit 0: Enumerates with or without a serial number.
6	0x01 0x02 ... 0xFA	Maximum Bus Current: 2 mA 4 mA ... 500 mA	Indicates the maximum current the device draws from the bus. Parameter value = Maximum bus current/2 (in 8-bit hexadecimal format)

Table 3-8. CNFG_SetUSBParameters- Response

Byte Index	Response		Description
0	0x81	Echo command code	Returns whether the CNFG_SetUSBParameters command was accepted
1	0x20	Accepted - successful	Command accepted, and the VID and PID are saved to NVM
	0x60	Rejected - invalid parameters	Command rejected because the parameters are invalid
	0x64	Rejected - locked	The command is rejected because the device configuration is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.2. CNFG_SetManufacturer Command and its Response

The CNFG_SetManufacturer command (0x83) saves the manufacturer name and its length to NVM:

- `bMfrLen` → Length of the manufacturer name, up to 60 bytes.
- `vMfrName` → The manufacturer name in Unicode format. Manufacturer name may contain printable ASCII characters. Because the string is stored in Unicode format, each character requires two bytes. Therefore, up to 30 printable ASCII characters, including spaces, are supported.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-9. CNFG_SetManufacturer - Command

Byte Index	Command		Description
0	0x83	Command code	Saves the manufacturer name to NVM

Table 3-9. CNFG_SetManufacturer - Command (continued)

Byte Index	Command		Description
1	bMfrLen	Number of ASCII characters x 2 (in UTF-16LE format)	Length of manufacturer name in bytes
2	vMfrName	Manufacturer name (up to 30 ASCII characters)	Manufacturer name, in UTF-16LE format

Table 3-10. CNFG_SetManufacturer - Response

Byte Index	Response		Description
0	0x83	Echo command code	Returns whether the CNFG_SetManufacturer command was accepted
1	0x20	Accepted - success	Command accepted, and the manufacturer name is saved to NVM
	0x60	Rejected - too long	Command rejected because the name is too long
	0x64	Rejected - locked	The command is rejected because the device configuration is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.3. CNFG_SetProductName Command and its Response

The CNFG_SetProductName command (0x85) saves the product name and its length to NVM:

- bPrdLen → Length of the product name, maximum 60 bytes.
- vPrdName → Product name in Unicode format. The product name may contain printable ASCII characters. Because the string is stored in Unicode format, each character requires two bytes. Therefore, up to 30 printable ASCII characters are supported, including spaces.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-11. CNFG_SetProductName - Command

Byte Index	Command		Description
0	0x85	Command code	Saves the product name to NVM
1	bPrdLen	Number of ASCII characters x 2 (UTF-16LE format)	Length of the product name in bytes
2	vPrdName	Product name (up to 30 ASCII characters)	Product name, in UTF-16LE format

Table 3-12. CNFG_SetProductName - Response

Byte Index	Response		Description
0	0x85	Echo command code	Returns whether the CNFG_SetProductName command was accepted
1	0x20	Accepted - success	Command accepted and product name is saved to NVM
	0x60	Rejected - too long	The command is rejected because the name is too long
	0x64	Rejected - locked	The command is rejected because the device configuration is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.4. CNFG_SetSerialNumber Command and its Response

The CNFG_SetSerialNumber command (0x87) saves the serial number and its length to NVM:

- `bSrlLen` → Length of the serial number, maximum 60 bytes.
- `vSrlNum` → Serial number in Unicode format. Serial number may contain printable ASCII characters. Because the string is stored in UTF-16LE format, each character requires two bytes. Therefore, up to 30 printable ASCII characters, including spaces, are supported.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-13. `CNFG_SetSerialNumber` - Command

Byte Index	Command		Description
0	0x87	Command code	Saves the serial number to NVM
1	<code>bSrlLen</code>	Number of ASCII characters x 2 (UTF-16LE format)	Length of the serial number in bytes
2	<code>vSrlNum</code>	Serial number (up to 30 ASCII characters)	Serial number, in UTF-16LE format

Table 3-14. `CNFG_SetSerialNumber` - Response

Byte Index	Response		Description
0	0x87	Echo command code	Returns whether the <code>CNFG_SetSerialNumber</code> command was accepted
1	0x20	Accepted - success	The command is accepted, and the serial number is saved to NVM
	0x60	Rejected - too long	Command is rejected because the name is too long
	0x64	Rejected - locked	The command is rejected because the device configuration is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.5. `CNFG_GetUSBParameters` Command and its Response

The `CNFG_GetUSBParameters` command (0x89) reads the USB parameters from NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-15. `CNFG_GetUSBParameters` - Command

Byte Index	Command		Description
0	0x89	Command code	Reads USB parameters from NVM

Table 3-16. `CNFG_GetUSBParameters` () - Response

Byte Index	Response		Description
0	0x89	Echo command code	Returns the command code and USB parameters from NVM
1	0x21	Accepted	Command is accepted
2	L(wVID)	VID – lower byte	Vendor ID
3	H(wVID)	VID – higher byte	
4	L(wPID)	PID – lower byte	Product ID
5	H(wPID)	PID – higher byte	

Table 3-16. CNFG_GetUSBParameters () - Response (continued)

Byte Index	Response		Description
6	0x00	SrcType-RmtWakUp-EnDisSrlNumEnum:	Combines three settings: Bit 3: Self-powered (1) or bus-powered (0). Bit 2: Remote wake-up supported (1) or not (0). Bit 0: Enumerates with a serial number (1) or without (0).
	0x04	BusPow-RWNo-SNEnNo	
	0x08	Bus Pow-RWYes-SNEnNo	
	0x0C	Self Pow-RWNo-SNEnNo	
	0x01	SelfPow-RWYes-SNEnNo	
	0x05	BusPow-RWNo-SNEnYes	
	0x09	BusPow-RWYes-SNEnYes	
	0x0D	SelfPow-RWNo-SNEnYes SelfPow-RWYes-SNEnYes	
7	0x01	Maximum Current: 2 mA	Indicates the maximum current the device would draw from the bus. Maximum bus current = Parameter value x 2
	0x02	4 mA	
	...	...	
	0xFA	500 mA	

3.6.6. CNFG_GetManufacturer Command and its Response

The CNFG_GetManufacturer command (0x8B) reads the manufacturer name from NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-17. CNFG_GetManufacturer - Command

Byte Index	Command		Description
0	0x8B	Command code	Returns the manufacturer name saved in NVM

Table 3-18. CNFG_GetManufacturer - Response

Byte Index	Response		Description
0	0x8B	Echo command code	Returns the command code and the manufacturer name from NVM
1	0x21	Accepted	Command is accepted
2	bMfrLen	Number of ASCII characters x 2 (in UTF-16LE format)	Length of manufacturer name in bytes
3	vMfrName	Manufacturer name (up to 30 ASCII characters)	Manufacturer name, in UTF-16LE format

3.6.7. CNFG_GetProductName Command and its Response

The CNFG_GetProductName command (0x8D) reads the product name from NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-19. CNFG_GetProductName - Command

Byte Index	Command		Description
0	0x8D	Command code	Returns the product name from NVM

Table 3-20. CNFG_GetProductName - Response

Byte Index	Response		Description
0	0x8D	Echo command code	Returns the command code and the product name from NVM
1	0x21	Accepted	Command is accepted

Table 3-20. CNFG_GetProductName - Response (continued)

Byte Index	Response		Description
2	bPrdLen	Number of ASCII characters x 2 (UTF-16LE format)	Length of the product name in bytes
3	vPrdName	Product name (up to 30 ASCII characters)	Product name, in UTF-16LE format

3.6.8. CNFG_GetSerialNumber Command and its Response

The CNFG_GetSerialNumber command (0x8F) reads the serial number from NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-21. CNFG_GetSerialNumber - Command

Byte Index	Command		Description
0	0x8F	Command code	Returns the serial number from NVM

Table 3-22. CNFG_GetSerialNumber - Response

Byte Index	Response		Description
0	0x8F	Echo command code	Returns the command code and the serial number saved in NVM
1	0x21	Accepted	Command is accepted
2	bSrlLen	Number of ASCII characters x 2 (UTF-16LE format)	Length of the serial number in bytes
3	vSrlNum	Serial number (up to 30 ASCII characters)	Serial number, in UTF-16LE format

3.6.9. CNFG_SetIOConfiguration Command and its Response

The CNFG_SetIOConfiguration command (0xA1) saves the I/O configuration to NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-23. CNFG_SetIOConfiguration - Command

Byte Index	Command		Description
0	0xA1	Command code	Saves the I/O pin configuration to NVM
1	bIO0S	IO0 configuration	Refer to the IO0 setting register
2	bIO1S	IO1 configuration	Refer to the IO1 setting register
3	bIO2S	IO2 configuration	Refer to the IO2 setting register
4	bIO3S	IO3 configuration	Refer to the IO3 setting register
5	bIO4S	IO4 configuration	Refer to the IO4 setting register
6	bIO5S	IO5 configuration	Refer to the IO5 setting register

Table 3-24. CNFG_SetIOConfiguration - Response

Byte Index	Response		Description
0	0xA1	Echo command code	Returns whether the CNFG_IOConfiguration command was accepted
1	0x20	Accepted - success	Accepted, and the I/O configuration to NVM is saved
	0x60	Rejected - invalid configuration	Rejected because the configuration is invalid
	0x64	Rejected - locked	Rejected because the device is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.10. CNFG_GetIOConfiguration Command and its Response

The CNFG_GetIOConfiguration command (0xA3) reads the I/O configuration from SRAM or NVM:

- bSource → 0x00 = SRAM; 0x01 = NVM

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-25. CNFG_GetIOConfiguration - Command

Byte Index	Command		Description
0	0xA3	Command code	Returns the I/O Configuration from SRAM or NVM
1	0x00 0x01	bSource: SRAM NVM	

Table 3-26. CNFG_GetIOConfiguration - Response

Byte Index	Response		Description
0	0xA3	Echo command code	Returns whether the CNFG_IOConfiguration command is accepted and returns the I/O configuration from the specified source
1	0x21 0x60	Accepted: Successful Rejected: Invalid Parameters	
2	bIO0S	IO0 configuration	Refer to the IO0 setting register
3	bIO1S	IO1 configuration	Refer to the IO1 setting register
4	bIO2S	IO2 configuration	Refer to the IO2 setting register
5	bIO3S	IO3 configuration	Refer to the IO3 setting register
6	bIO4S	IO4 configuration	Refer to the IO4 setting register
7	bIO5S	IO5 configuration	Refer to the IO5 setting register

3.6.11. CNFG_SetI2CConfiguration Command and its Response

The CNFG_SetI2CConfiguration command (0xA9) saves the I²C Configuration to NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-27. CNFG_SetI2CConfiguration - Command

Byte Index	Command		Description
0	0xA9	Command code	Saves the I ² C configuration to NVM
1	0x70 0x16 0x04	Speed: 100 kHz 400 kHz 1 MHz	I ² C clock frequency
2	0x01 0x02 ... 0xFA	Timeout: 1 ms 2 ms 250 ms	

Table 3-28. CNFG_SetI2CConfiguration - Response

Byte Index	Response		Description
0	0xA9	Echo command code	Returns whether the
1	0x20	Accepted – success	CNFG_SetI2CConfiguration command was accepted
	0x60	Rejected – wrong parameters	Command rejected because the parameters are invalid
	0x64	Rejected – configuration locked	Command rejected because the device configuration is locked
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.12. CNFG_GetI2CConfiguration Command and its Response

The CNFG_GetI2CConfiguration command (0xAB) reads the I²C Configuration from SRAM or NVM:

- bSource → 0x00 = SRAM; 0x01 = NVM

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-29. CNFG_GetI2CConfiguration - Command

Byte Index	Command		Description
0	0xAB	Command code	Returns the I ² C configuration from SRAM or NVM
1	0x00	bSource: SRAM	
	0x01	NVM	

Table 3-30. CNFG_GetI2CConfiguration - Response

Byte Index	Response		Description
0	0xAB	Echo command code	Returns the I ² C configuration from SRAM or NVM
1	0x21	Accepted	Command is accepted
2	0x70	Speed: 100 kHz	I ² C clock frequency
	0x16	400 kHz	
	0x04	1 MHz	
3	0x01	Timeout: 1 ms	I ² C clock-stretch timeout
	0x02	2 ms	
	...		
	0xFA	250 ms	

3.6.13. CNFG_SetSPIConfiguration Command and its Response

The CNFG_SetSPIConfiguration command (0xAD) saves the SPI configuration to NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-31. CNFG_SetSPIConfiguration - Command

Byte Index	Command		Description
0	0xAD	Command code	Configures the SPI module with the following settings:

Table 3-31. CNFG_SetSPIConfiguration - Command (continued)

Byte Index	Command		Description
1	0x30	Speed: 12 MHz	SPI clock frequency
	0x20	6 MHz	
	0x32	3 MHz	
	0x22	1.5 MHz	
	0x34	750 kHz	
	0x24	375 kHz	
	0x26	187.5 kHz	
2	0x44	Mode: 0	SPI mode
	0x45	1	
	0x46	2	
	0x47	3	
3	0x00	CS-ActStat:	Chip-select and its assertion state
	0x01	CS0-ActiveLow	
	0x40	CS0-ActiveHigh	
	0x41	CS1-ActiveLow	
	0x80	CS1-ActiveHigh	
	0x81	CS2-ActiveLow	
	0xC0	CS2-ActiveHigh	
	0xC1	CS3-ActiveLow	
		CS3-ActiveHigh	
4	0x0000	0 μ s	Delay between chip-select assertion and the first data byte
	0x000A	10 μ s	Delay between each data byte
	0x0014	20 μ s	
	0x001E	30 μ s	Delay between the last data byte and chip select deassertion.
	...	...	
	0x03E8	1000 μ s	
	...	...	
	0x1388	5000 μ s	

Table 3-32. CNFG_SetSPIConfiguration - Response

Byte Index	Response		Description
0	0xAD	Echo command code	Returns whether the CNFG_SetSPIConfiguration command was accepted
1	0x20	Accepted	
	0x60	Rejected - wrong parameters	Command rejected because the parameters are invalid
	0x64	Rejected - configuration locked	Command rejected because the device configurations are locked
	0x90	Failure - unsuccessful NVM write	Could not save in NVM; retry

3.6.14. CNFG_GetSPIConfiguration Command and its Response

The CNFG_GetSPIConfiguration command (0xAF) reads the SPI Configuration from SRAM or NVM:

- bSource $\rightarrow$ 0x00 = SRAM; 0x01 = NVM

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-33. CNFG_GetSPIConfiguration - Command

Byte Index	Command		Description
0	0xAF	Command code	Returns the SPI configuration from SRAM or NVM
1	0x00 0x01	bSource: SRAM NVM	

Table 3-34. CNFG_GetSPIConfiguration - Response

Byte Index	Response		Description
0	0xAF	Echo command code	Returns the SPI configuration from SRAM or NVM
1	0x21	Accepted	Command is accepted
2	0x30 0x20 0x32 0x22 0x34 0x24 0x26	Speed: 12 MHz 6 MHz 3 MHz 1.5 MHz 750 kHz 375 kHz 187.5 kHz	SPI clock frequency
3	0x44 0x45 0x46 0x47	Mode: 0 1 2 3	SPI mode
4	0x00 0x01 0x40 0x41 0x80 0x81 0xC0 0xC1	CS-ActStat: CS0-ActiveLow CS0-ActiveHigh CS1-ActiveLow CS1-ActiveHigh CS2-ActiveLow CS2-ActiveHigh CS3-ActiveLow CS3-ActiveHigh	Chip-select and its assertion state
5	0x0000 0x000A 0x0014 0x001E ... 0x03E8 ... 0x1388	0 μ s 10 μ s 20 μ s 30 μ s ... 1000 μ s ... 5000 μ s	Delay between chip-select assertion and the first data byte Delay between each data byte Delay between the last data byte and chip select de-assertion.

3.6.15. CNFG_SetNVMAAsCurrent Command and its Response

The CNFG_SetNVMAAsCurrent command (0x19) loads the I/O, I²C, and SPI configuration stored in NVM into the current SRAM configuration.

This command allows the device to apply the power-on default I/O, I²C, and SPI settings without requiring a power cycle.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-35. CNFG_SetNVMAAsCurrent - Command

Byte Index	Command		Description
0	0x19	Command code	NVM settings are applied as the current IO, I ² C and SPI configuration

Table 3-36. CNFG_SetNVMAAsCurrent - Response

Byte Index	Response		Description
0	0x19	Echo command code	Returns whether the CNFG_SetNVMAAsCurrent was command
1	0x20	Accepted – success	Accepted, and the NVM settings are applied as the current I/O, I ² C and SPI configuration
	0x64	Rejected – locked	Rejected because the device is locked

3.6.16. CNFG_ResetToFactoryDefault Command and its Response

The CNFG_ResetToFactoryDefault command (0xC1) replaces the NVM configuration settings with factory-default values.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-37. CNFG_ResetToFactoryDefault - Command

Byte Index	Command		Description
0	0xC1	Command code	Replaces the NVM configuration settings with factory-default values

Table 3-38. CNFG_ResetToFactoryDefault - Response

Byte Index	Response		Description
0	0xC1	Echo command code	Returns whether the CNFG_ResetToFactoryDefault () command was accepted or rejected
1	0x20	Accepted – success	The NVM configuration settings are replaced with factory-default values
	0x64	Rejected – locked	Rejected because the device is locked

3.6.17. CNFG_SetPassword Command and its Response

The CNFG_SetPassword command (0xC3) saves an eight-byte password to NVM and sets the protection level to 'Password Set - Unlocked':

- qwPassword → 8-bytes of password

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-39. CNFG_SetPassword - Command

Byte Index	Command		Description
0	0xC3	Command code	Saves an eight-byte password to NVM and sets the protection level to Password Set - Unlocked
1-8	qwPassword	Password (eight bytes)	Eight bytes of password

Table 3-40. CNFG_SetPassword - Response

Byte Index	Response		Description
0	0xC3	Echo command code	Returns whether the CNFG_SetPassword command was accepted or not

Table 3-40. CNFG_SetPassword - Response (continued)

Byte Index	Response		Description
1	0x20	Accepted – success	Accepted. The new state of the device configuration is Password protected (unlocked) and the password is saved to NVM.
	0x60	Rejected – wrong parameters	Rejected because the password is not eight bytes long
	0x64	Rejected - permanently locked	Rejected because the configurations are permanently locked
	0x68	Rejected – invalid state	Rejected because the configurations are locked
	0x6C	Rejected - password exists	Rejected because the password is already set
	0x90	Failure - unsuccessful NVM write	Could not save in NVM; retry

3.6.18. CNFG_DisablePassword Command and its Response

The CNFG_DisablePassword command (0xC5) removes the password from NVM and sets the protection level to Unprotected.

- qwPassword → 8-byte password

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-41. CNFG_DisablePassword - Command

Byte Index	Command		Description
0	0xC5	Command code	Removes the password
1-8	qwPassword	Password (8 bytes)	Eight bytes of password

Table 3-42. CNFG_DisablePassword - Response

Byte Index	Response		Description
0	0xC5	Echo command code	Returns whether the CNFG_DisablePassword command was accepted or rejected
1	0x20	Accepted – success	Accepted. The new state of the device configuration is Unprotected.
	0x60	Rejected – wrong parameters	Rejected because the password length did not match
	0x64	Rejected – permanently locked	Rejected because the configuration is permanently locked
	0x68	Rejected - invalid state	Rejected because the configuration is locked
	0x6C	Rejected - no operation	Rejected because the password is not set
	0x80	Rejected - wrong parameters	Rejected because the password did not match
	0x90	Failure - unsuccessful NVM write	Could not save in NVM; retry

3.6.19. CNFG_LockConfiguration Command and its Response

The CNFG_LockConfiguration command (0xC7) changes the protection level to 'Password Set - Locked':

- qwPassword → 8-byte password

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-43. CNFG_LockConfiguration - Command

Byte Index	Command		Description
0	0xC7	Command code	Changes the protection level to 'Password Set - Locked'
1-8	qwPassword	Password (8 bytes)	Eight bytes of password

Table 3-44. CNFG_LockConfiguration - Response

Byte Index	Response		Description
0	0xC7	Echo command code	Returns whether the CNFG_LockConfiguration command was accepted or rejected
1	0x20	Accepted – success	Accepted. The new state of the device configuration is Password protected (locked).
	0x80	Rejected – wrong parameters	Rejected because the password did not match
	0x64	Rejected - permanently locked	Rejected because the configuration is permanently locked
	0x68	Rejected - invalid state	Rejected because the password is not set
	0x6C	Rejected - no operation	Rejected because the configuration is already locked
	0x90	Failure - unsuccessful NVM write	Could not save in NVM; retry

3.6.20. CNFG_UnlockConfiguration Command and its Response

The CNFG_UnlockConfiguration command (0xC9) changes the protection level to 'Password Set - Unlocked':

- qwPassword → 8-byte password

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-45. CNFG_UnlockConfiguration - Command

Byte Index	Command		Description
0	0xC9	Command code	Changes the protection level as 'Password Set - Unlocked'
1-8	qwPassword	Password (8 bytes)	Eight bytes of password

Table 3-46. CNFG_UnlockConfiguration - Response

Byte Index	Response		Description
0	0xC9	Echo command code	Returns whether the CNFG_UnlockConfiguration command was accepted or rejected
1	0x20	Accepted – success	Accepted. The new state of the device configuration is Password-protected (unLocked).
	0x80	Rejected – wrong parameters	Rejected because the password did not match
	0x64	Rejected – password mismatch	Rejected because the configuration is permanently locked
	0x68	Rejected - invalid state	Rejected because the password is not set
	0x6C	Rejected - no operation	Rejected because the configuration is already unlocked
	0x90	Failure - unsuccessful EEPROM write	Could not save to EEPROM; retry

3.6.21. CNFG_ChangePassword Command and its Response

The CNFG_ChangePassword command (0xCB) replaces the existing password with new password and retains the protection level as 'Password Set - Unlocked':

- qwOldPassword → 8-byte old password
- qwNewPassword → 8-byte new password

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-47. CNFG_ChangePassword - Command

Byte Index	Command		Description
0	0xCB	Command code	Replaces the old password with the new password
1-8	qwOldPassword	Old password (eight bytes)	Eight bytes of the old password
9-16	qwNewPassword	New password (eight bytes)	Eight bytes of the new password

Table 3-48. CNFG_ChangePassword - Response

Byte Index	Response		Description
0	0xCB	Echo command code	Returns whether the CNFG_ChangePassword command was accepted
1	0x20	Accepted – success	Accepted. The state of the device configuration remains 'Password Set - Unlocked' and the new password is saved to NVM.
	0x60	Rejected – wrong parameters	Rejected because the new password length did not match
	0x64	Rejected – permanently locked	Rejected because the configuration is permanently locked
	0x68	Rejected – invalid state	Rejected because the configuration is locked
	0x6C	Rejected – no operation	Rejected because the password is not set
	0x80	Rejected – no operation	Rejected because the old password did not match
	0x90	Failure - unsuccessful NVM write	Saving to NVM failed; retry

3.6.22. CNFG_PermanentLockCfg Command and its Response

The CNFG_PermanentLockCfg command (0xCD) sets the protection level to Configuration - Permanently Locked.

- bSrlLen → Length of the serial number, up to 60 bytes
- vSrlNum → Serial number in Unicode format

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-49. CNFG_PermanentLockCfg - Command

Byte Index	Command		Description
0	0xCD	Command code	Permanently locks the device configuration
1	bSrlLen	Number of ASCII characters x 2 (in UTF-16LE format)	Length of the serial number in bytes
2-N+1	vSrlNum	Serial number (up to 30 ASCII characters)	Serial number, in UTF-16LE format

Table 3-50. CNFG_PermanentLockCfg - Response

Byte Index	Response		Description
0	0xCD	Echo command code	Returns whether the CNFG_PermanentLockCfg command was accepted
1	0x20	Accepted – success	The device configuration is permanently locked.
	0x60	Rejected – serial number mismatch.	Rejected because the serial number did not match.
	0x90	Failure - unsuccessful NVM write.	Saving into NVM failed; retry.

3.6.23. CNFG_GetCfgProtectionLevel Command and its Response

The CNFG_GetCfgProtectionLevel command (0xCF) reads the configuration protection level from NVM.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-51. CNFG_GetCfgProtectionLevel - Command

Byte Index	Command		Description
0	0xCF	Command code	Returns the configured protection level

Table 3-52. CNFG_GetCfgProtectionLevel - Response

Byte Index	Response		Description
0	0xCF	Echo command code	Returns whether the CNFG_GetCfgProtectionLevel command was accepted
1	0x21	Accepted – success	Accepted returns the configuration protection level
2	0x00	Password disabled	Protection Level: Device configuration is modifiable; no password is set.
	0x40	Password set – unlocked	A password is set, but the device configuration is modifiable
	0x80	Password set – locked	A password is set, but the device configuration is not modifiable
	0xC0	Permanently locked	The device configuration is permanently protected, and cannot be modified

3.6.24. MCP2222_SoftReset Command and Its Response

The MCP2222_SoftReset command (0x17) resets the MCP2222 device and causes it to enumerate again on the USB bus.

This command is sent through Endpoint 0x05 (OUT); the format is given in the following table.

Table 3-53. MCP2222_SoftReset - Command

Byte Index	Command		Description
0	0x17	Command code	Resets the MCP2222 and causes it to enumerate again

3.6.25. CNFG_GetRevID Command and its Response

The CNFG_GetRevID command (0xB1) reads the MCP2222 device revision ID.

This command is sent through Endpoint 0x05 (OUT). The response is read through Endpoint 0x85 (IN).

Table 3-54. CNFG_GetRevID - Command

Byte Index	Command		Description
0	0xB1	Command code	Returns the device revision ID

Table 3-55. CNFG_GetRevID - Response

Byte Index	Response		Description
0	0xB1	Echo command code	Returns the CNFG_GetRevID command code and the device revision ID
1	0x21	Accepted – returns the device revision ID	Accepted – returns the device revision ID

Table 3-55. CNFG_GetRevID - Response (continued)

Byte Index	Response		Description
2	0x10	Rev A0	Device revision ID
	0x11	Rev A1	
	...	...	
	0x20	Rev B0	
	0x21	Rev B1	
	...	...	

3.7. Device Configuration Command Usage

This section describes typical command sequences used to modify and read MCP2222 configuration settings.

3.7.1. USB Descriptors Modification Procedure

Use the following procedure to configure USB descriptor parameters such as the VID, PID, power attributes, manufacturer name, product name, and serial number.

1. Configure USB Parameters:

- Send the CNFG_SetUSBParameters command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The responses can be one of the following:

- The command is accepted and executed successfully
- The command is rejected because the device configuration is locked
- The command is rejected because the parameters are invalid

If the response is rejected because parameters are invalid, retry with valid parameters.

If the response is rejected as “Configurations are locked”, try again after unlocking the configuration.

If the response is accepted and successful, proceed to step 2.

2. Configure the Manufacturer’s Name:

- Send the CNFG_SetManufacturer command through Endpoint 0x05 (OUT)
 - Read the response through Endpoint 0x85 (IN)

The responses can be one of the following:

- The command is accepted and executed successfully
- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid

If the response is rejected because parameters are invalid, retry with valid parameters

If the response is rejected as “Configurations are Locked”, try again after unlocking the configuration.

If the response is accepted and successful, proceed to Step 3.

3. Configure the Product Name:

- Send the CNFG_SetProductName command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The responses can be one of the following:

- The command is accepted and executed successfully

- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid

If the response is rejected because parameters are invalid, retry with valid parameters

If the response is rejected as "Configurations are Locked", try again after unlocking the configuration.

If the response is accepted and successful, proceed to Step 4.

4. **Configure the Serial Number:**

- Send the CNFG_SetSerialNumber command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The responses can be one of the following:

- The command is accepted and executed successfully
- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid

If the response is rejected because parameters are invalid, retry with valid parameters

If the response is rejected as "Configurations are Locked", try again after unlocking the configuration.

If the response is accepted and successful, proceed to Step 5.

5. **Re-Enumerate the MCP2222 Device**

Send the MCP2222_SoftReset command through Endpoint 0x05 (OUT).

This command re-enumerates the MCP2222 with the new USB descriptors.

3.7.2. **I/O Pins Configuration Procedure**

Use the following procedure to configure the I/O pins:

- Referring to all IOxSettings registers, prepare the data to be sent with the CNFG_SetIOConfiguration command
- Send the CNFG_SetIOConfiguration command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The possible responses are as follows:

- The command is accepted and executed successfully
- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid

If the response is rejected because parameters are invalid, retry with valid parameters.

If the response is rejected as "Configurations are Locked", try again after unlocking the configuration.

If the response is accepted and successful, the I/O pins configurations are saved in EEPROM.

To make these I/O setting values the current setting:

- Send the CNFG_SetEEPROMAsCurrent command through Endpoint 0x05 (OUT)

3.7.3. **I²C Configuration Modification Procedure**

Use the following procedure to configure I²C:

- Send the CNFG_SetI2CConfiguration command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The possible responses are as follows:

- The command is accepted and executed successfully
- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid
If the response is rejected because parameters are invalid, retry with valid parameters.
If the response is rejected as “Configurations are Locked”, try again after unlocking the configuration.
If the response is accepted and successful, the I/O pins configurations are saved in EEPROM.

To make these I²C setting values the current setting:

- Send the CNFG_SetEEPROMAsCurrent command through Endpoint 0x05 (OUT)

3.7.4. SPI Configuration Modification Procedure

Use the following procedure to configure SPI:

- Send the CNFG_SetSPIConfiguration command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

The possible responses are as follows:

- The command is accepted and executed successfully
- The command is rejected because the configurations are locked
- The command is rejected because the parameters are invalid
If the response is rejected because parameters are invalid, retry with valid parameters.
If the response is rejected as “Configurations are Locked”, try again after unlocking the configuration.
If the response is accepted and successful, the I/O pins configurations are saved in EEPROM.

To make these SPI setting values the current setting:

- Send the CNFG_SetEEPROMAsCurrent command through Endpoint 0x05 (OUT)

3.7.5. Restoring to the Factory-Default Configuration

To restore the power-on default configuration to the factory-default values:

1. Send the CNFG_ResetToFactoryDefault command through Endpoint 0x05 (OUT).
2. Read the response through Endpoint 0x85 (IN).

A successful response indicates that the NVM power-up settings have been replaced with the factory-default configuration.

3.7.6. Loading NVM Power-up Settings as Current Configuration

To apply the I/O, I²C, and SPI settings stored in NVM as the current configuration, do the following steps:

1. Send the CNFG_SetNVMSCurrent command through Endpoint 0x05 (OUT).
2. Read the response through Endpoint 0x85 (IN).

Possible responses include the following:

- The command is accepted and the NVM power-up settings are made current settings
- The command is rejected because the configuration is locked

If the command is rejected because the configuration is locked, retry after unlocking the configuration.

If the response is accepted and successful, the I/O, I²C and SPI interfaces are configured according to the settings stored in NVM.

3.7.7. Setting or Changing the Password

Set Password

To set a password, follow these steps:

1. Send the CNFG_SetPassword command through Endpoint 0x05 (OUT).
2. Provide an eight-byte password.
3. Read the response through Endpoint 0x85 (IN).

Possible responses include the following:

- The command is accepted and the password is saved
- The command is rejected due to invalid password parameters
- The command is rejected because the configuration is permanently locked
- The command is rejected because a password already exists

If the response is rejected because parameters are invalid, retry with a password that has eight characters.

If the response is rejected as password already exists, try recalling the previously set password.

If the response is rejected as configuration is permanently locked, neither the password can be set nor the configuration can be changed.

If the response is accepted, the password is set. Once all configuration is complete, lock the configuration.

Change Password

To change an existing password, follow these steps:

1. Send the CNFG_ChangePassword command through Endpoint 0x05 (OUT).
2. Provide:
 - The current eight-byte password
 - The new eight-byte password
3. Read the response through Endpoint 0x85 (IN).

If the configuration is locked, unlock it before attempting to change the password.

3.7.8. Locking, Unlocking, and Permanently Locking the Device Configuration

Lock Configuration

To lock the configuration, follow these steps:

1. Send the CNFG_LockConfiguration command through Endpoint 0x05 (OUT).
2. Provide the eight-byte password.
3. Read the response through Endpoint 0x85 (IN).

A successful response indicates that the configuration is password-protected and locked.

Unlock Configuration

To unlock the configuration, follow these steps:

1. Send the CNFG_UnlockConfiguration command through Endpoint 0x05 (OUT).
2. Provide the eight-byte password.
3. Read the response through Endpoint 0x85 (IN).

A successful response indicates that the configuration is unlocked and can be modified.

Permanently Lock Configuration

Caution: Permanently locking the configuration is irreversible. Verify that all configuration settings are correct before issuing this command.

If the response is “accepted and successful (0x20)”, the configuration is permanently locked and can never be unlocked or modified.

If the response is rejected as “serial number mismatch (0x60)”, retry with the correct serial number.

To read the serial number:

1. Send the CNFG_GetSerialNumber command through Endpoint 0x05 (OUT).
2. Read the response through Endpoint 0x85 (IN).

The response would be as follows:

- 0x8F → CNFG_GetSerialNumber Command
- 0x21 → The command is accepted and executed successfully
- bSrlLen → Length of the serial number in bytes (maximum 60 bytes)
- vSrlNum → The serial number is in Unicode format and can include up to 30 printable ASCII characters

Use this serial number to permanently lock the device configuration.

3.7.9. Checking the Configuration Protection Status

To read the current configuration protection level:

- Send the CNFG_GetCfgProtectionLevel command through Endpoint 0x05 (OUT)
- Read the response through Endpoint 0x85 (IN)

A successful response returns the protection level:

- 0x00 — Password disabled; the configuration is modifiable
- 0x40 — Password set; the configuration is unlocked
- 0x80 — Password set; the configuration is locked
- 0xC0 — The configuration is permanently locked

3.8. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	Reserved									
0x01	DEVCFG	7:0					SLFPWR	RMTWKP		SNEN
0x02	IO0SETTING	7:0	IDLST0	ALRT0EG[1:0]		OD0SEL	PU0SEL	IO0DIR	IO0DES[1:0]	
0x03	IO1SETTING	7:0	IDLST1			OD1SEL	PU1SEL	IO1DIR	IO1DES[1:0]	
0x04	IO2SETTING	7:0	IDLST2	ALRT1EG[1:0]		OD2SEL	PU2SEL	IO2DIR	IO2DES[1:0]	
0x05	IO3SETTING	7:0	IDLST3	CLKDIV[1:0]		OD3SEL	PU3SEL	IO3DIR	IO3DES[1:0]	
0x06	IO4SETTING	7:0	IDLST4	ALRT2EG[1:0]		OD4SEL	PU4SEL	IO4DIR	IO4DES[1:0]	
0x07	IO5SETTING	7:0	IDLST5	ALRT3EG[1:0]		OD5SEL	PU5SEL	IO5DIR	IO5DES[1:0]	
0x08	CFGPRTN	7:0	CFGPRO[1:0]							

3.8.1. Device Configuration

Name: DEVCFG
Offset: 0x01
Property: 1

Bit	7	6	5	4	3	2	1	0
					SLFPWR	RMTWKP		SNEN
Access					R/W	R/W		R/W
Reset					0	0		1

Bit 3 – SLFPWR USB Self-Powered Attribute

Value	Description
0	Bus-powered: The device draws power from USB VBUS (factory default)
1	Self-powered: The device has an external power supply. The VBUS pin must be connected to USB VBUS through a potential divider

Bit 2 – RMTWKP USB Remote Wake-up Capability

Value	Description
0	No remote wake-up: The device cannot wake the USB host from suspend. The ALRT pins can only wake the MCP2222 device (factory default).
1	Remote wake-up enabled: The ALRT pins configured for edge detection and can wake the MCP2222 device as well as the USB host.

Bit 0 – SNEN USB CDC Serial Number Enable

Value	Description
0	No serial number: The device enumerates without a serial number. Windows may assign different COM ports when the device is connected to different USB ports.
1	Serial number enabled: Windows maintains consistent COM port assignment (factory default).

3.8.2. IO0SETTING

Name: IO0SETTING

Offset: 0x02

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST0	ALRT0EG[1:0]		OD0SEL	PU0SEL	IO0DIR	IO0DES[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	1	0	0

Bit 7 – IDLST0

IO0 Pin Idle or default output state selection bit (valid only when the pin is configured as an output pin).

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1'

Bits 6:5 – ALRT0EG[1:0] Alert0 or Interrupt0 Pin Edge Selection

This is valid only when the pin is configured as the ALRT0 pin.

Value	Description
00	Alert or interrupt disabled
01	Falling edge
10	Rising edge
11	Both edge

Bit 4 – OD0SEL IO0 Pin Open-Drain Selection

This is valid only when the pin is configured as an output and not as a CSx pin.

Value	Description
0	The output pin is not an open-drain pin
1	The output pin is an open-drain pin

Bit 3 – PU0SEL IO0 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled (factory default)

Bit 2 – IO0DIR IO0 Direction

Input or output.

Value	Description
0	GPIO output
1	GPIO input (factory default)

Bits 1:0 – IO0DES[1:0] IO0 Designation

Input or output.

Value	Description
00	GPIO0 pin (GPIO0) (factory default)
01	Clear-to-Send (CTS): UART handshaking signal Pin (CTS)
10	SPI client chip select 3 pin (CS3)

Value	Description
11	Alert0 or Interrupt0 pin (ALRT0)

3.8.3. IO1SETTING

Name: IO1SETTING

Offset: 0x03

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST1			OD1SEL	PU1SEL	IO1DIR	IO1DES[1:0]	
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	1			0	0	1	0	0

Bit 7 – IDLST1 IO1 Pin Idle or Default Output State Selection

This is valid only when the pin is configured as an output pin.

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1' (factory default)

Bit 4 – OD1SEL IO1 Pin Open-Drain Selection

This is valid only when the pin is configured as an output and not as a CSx pin.

Value	Description
0	The output pin is not an open-drain pin (factory default)
1	The output pin is an open-drain pin

Bit 3 – PU1SEL IO1 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled

Bit 2 – IO1DIR IO1 Direction

This is valid only when the pin is configured as an input.

Value	Description
0	GPIO output
1	GPIO input (factory default)

Bits 1:0 – IO1DES[1:0] IO1 Designation

Input or output.

Value	Description
00	GPIO1 pin (GPIO1) (factory default)
01	Request-to-send (RTS): UART handshaking signal pin
10	SPI client chip select 2 pin (CS2)
11	I ² C activity (I2CA)

3.8.4. IO2SETTING

Name: IO2SETTING

Offset: 0x04

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST2	ALRT1EG[1:0]		OD2SEL	PU2SEL	IO2DIR	IO2DES[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	1	0	0

Bit 7 – IDLST2 IO2 Pin Idle or Default Output State Selection

This is valid only when the pin is configured as an output pin.

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1'

Bits 6:5 – ALRT1EG[1:0] Alert1 or Interrupt1 Pin Edge Selection

This is valid only when the pin is configured as the ALRT1 pin.

Value	Description
00	Alert or interrupt disabled (factory default)
01	Falling edge
10	Rising edge
11	Both edge

Bit 4 – OD2SEL IO2 Pin Open-Drain Selection

This is valid only when the pin is configured as an output and not as a CSx pin.

Value	Description
0	The output pin is not an open-drain pin
1	The output pin is an open-drain pin

Bit 3 – PU2SEL IO2 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled (factory default)

Bit 2 – IO2DIR IO2 Direction

Input or output.

Value	Description
0	GPIO output
1	GPIO input (factory default)

Bits 1:0 – IO2DES[1:0] IO2 Designation

Input or output.

Value	Description
00	GPIO2 pin (GPIO2) (factory default)
01	UART receive activity (RXA)
10	SPI client chip select 1 pin (CS1)
11	Alert1 or interrupt1 pin (ALRT1)

3.8.5. IO3SETTING

Name: IO3SETTING

Offset: 0x05

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST3	CLKDIV[1:0]		OD3SEL	PU3SEL	IO3DIR	IO3DES[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	1	1

Bit 7 – IDLST3 IO3 Pin Idle or Default Output State Selection

This is valid only when the pin is configured as an output pin.

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1'

Bits 6:5 – CLKDIV[1:0] Clock Reference Out Frequency Selection

This is valid only when the pin is configured as the CLKO pin.

Value	Description
00	The clock-out frequency is 6 MHz
01	The clock-out frequency is 4 MHz (factory default)
10	The clock-out frequency is 2 MHz
11	The clock-out frequency is 1 MHz

Bit 4 – OD3SEL IO3 Pin Open-Drain Selection

This is valid only when the pin is configured as an output and not as a CLKO pin.

Value	Description
0	The output pin is not an open-drain pin
1	The output pin is an open-drain pin

Bit 3 – PU3SEL IO3 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled

Bit 2 – IO3DIR IO3 Direction

Input or output.

Value	Description
0	GPIO output (factory default)
1	GPIO input

Bits 1:0 – IO3DES[1:0] IO3 Designation

Input or output.

Value	Description
00	GPIO3 pin (GPIO3)
01	UART transmit activity (TXA)
10	I ² C clock-line low timeout (I2CTO)
11	Clock reference out (CLKO) (factory default)

3.8.6. IO4SETTING

Name: IO4SETTING

Offset: 0x06

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST4	ALRT2EG[1:0]		OD4SEL	PU4SEL	IO4DIR	IO4DES[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	1	0

Bit 7 – IDLST4 IO4 Pin Idle or Default Output State Selection

This is valid only when the pin is configured as an output pin.

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1' (factory default)

Bits 6:5 – ALRT2EG[1:0] Alert2 or Interrupt2 Pin Edge Selection

This is valid only when the pin is configured as the ALRT2 pin.

Value	Description
00	Alert or interrupt disabled
01	Falling edge
10	Rising edge
11	Both edge

Bit 4 – OD4SEL IO4 Pin Open-Drain Selection

This is valid only when the pin is configured as an output and not as a CSx pin.

Value	Description
0	The output pin is not an open-drain pin
1	The output pin is an open-drain pin

Bit 3 – PU4SEL IO4 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled

Bit 2 – IO4DIR IO4 Direction

Input or output.

Value	Description
0	GPIO output (factory default)
1	GPIO input

Bits 1:0 – IO4DES[1:0] IO4 Designation

Input or output.

Value	Description
00	GPIO4 pin (GPIO4)
01	USB configuration (enumeration) done
10	SPI client chip select 0 pin (CS0) (factory default)
11	Alert2 or Interrupt2 pin (ALRT2)

3.8.7. IO5SETTING

Name: IO5SETTING

Offset: 0x07

Property: -

Bit	7	6	5	4	3	2	1	0
	IDLST5	ALRT3EG[1:0]		OD5SEL	PU5SEL	IO5DIR	IO5DES[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	1	1	1	1

Bit 7 – IDLST5 IO5 Pin Idle or Default Output State Selection

This is valid only when the pin is configured as an output pin.

Value	Description
0	Idle state default output value is logic '0'
1	Idle state or default output value is logic '1' (factory default)

Bits 6:5 – ALRT3EG[1:0] Alert3 or Interrupt3 Pin Edge Selection

This is valid only when the pin is configured as the ALRT3 pin.

Value	Description
00	Alert or interrupt disabled
01	Falling edge (factory default)
10	Rising edge
11	Both edge

Bit 4 – OD5SEL IO5 Pin Open-Drain Selection

This is valid only when the pin is configured as an output.

Value	Description
0	The output pin is not an open-drain pin
1	The output pin is an open-drain pin

Bit 3 – PU5SEL IO5 Pin Pull-Up Enable

This is valid only when the pin is configured as an input.

Value	Description
0	The internal pull-up is disabled
1	The internal pull-up is enabled (factory default)

Bit 2 – IO5DIR IO5 Direction

Input or output.

Value	Description
0	GPIO output
1	GPIO input (factory default)

Bits 1:0 – IO5DES[1:0] IO5 Designation

Input or output.

Value	Description
00	GPIO5 pin (GPIO5)
01	USB host is suspended (USBSPND)
10	SPI activity (SPIA)
11	Alert3 or Interrupt3 pin (ALRT3) (factory default)

3.8.8. Configuration Protection

Name: CFGPRTN

Offset: 0x08

Property: -

Bit	7	6	5	4	3	2	1	0
	CFGPRO[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bits 7:6 – CFGPRO[1:0] Device Configuration Protection Level

Value	Description
00	Unprotected (factory default)
01	Password protected - unlocked
10	Password protected - locked
11	Permanently locked

4. UART COMMUNICATION

The MCP2222 supports UART communication using the CDC (Communication Device Class) protocol. It also supports the CDC commands listed in the *Supported CDC UART Commands* table.

4.1. UART Interface Commands and Their Responses

The MCP2222 supports CDC class commands for UART operations and UART configuration.

- Control commands are sent through, and the responses are read through Endpoint 0x00
- Notification interrupts are received through Endpoint 0x01 (IN)
- Data is transmitted through Endpoint 0x02 (OUT)
- Data is received through Endpoint 0x82 (IN)

Table 4-1. Supported CDC Class Interface Commands for UART Functions

Command/Interrupt	Effect
SET_LINE_CODING	Configures the UART baud rate, parity stop bits and data width
GET_LINE_CODING	Reads the current UART configuration, including baud rate, parity stop bits, and data width
SET_CONTROL_LINE_STATE	Controls the state of the UART peripheral and the RTS line
SERIAL_STATE	Sends the error state (parity error and framing error) to the Host

4.1.1. SET_LINE_CODING Command

The SET_LINE_CODING command configures the UART with the specified parameters.

Send this request through Endpoint 0x00, with the parameters described in the following table, to configure the UART baud rate, parity stop bits, and data width.

Note: Avoid issuing a SET_LINE_CODING request while the UART data transfer is in progress.

Table 4-2. SET_LINE_CODING - Command

Byte Index	Command	Description
0	0x21 bmRequestType	Host to device
1	0x20 bRequest (Command code)	Configure the UART baud rate, stop bits, parity, and data width
2	0x00 wValue – lower byte	Reserved
3	0x00 wValue – higher byte	
4	0x00 wIndex – lower byte	Interface 0
5	0x00 wIndex – higher byte	
6	0x07 wLength – lower byte	Payload length in bytes
7	0x00 wLength – higher byte	
8	L (dwBaud) dwBaud – lowest byte	Baud rate in 32-bit hexadecimal format
9	H1 (dwBaud) dwBaud – higher byte	
10	H2 (dwBaud) dwBaud – next higher byte	
11	H3 (dwBaud) dwBaud – highest byte	
12	0x00 0x02 1 Stop bit 2 Stop bits	Number of stop bits
13	0x00 0x01 0x02 No parity Odd parity Even parity	Data parity

Table 4-2. SET_LINE_CODING - Command (continued)

Byte Index	Command		Description
14	0x05	5-bit ⁽¹⁾ data width	Data byte width
	0x06	6-bit data width	
	0x07	7-bit data width	
	0x08	8-bit data width	



CAUTION When the data width is set to five bits, the UART communication with no flow control may not operate properly beyond the baud rate of 921,600 bps.

4.1.2. SET_CONTROL_LINE_STATE Command

The SET_CONTROL_LINE_STATE command:

- Enables or disables the UART peripheral
- Asserts or deasserts the RTS line

Send this request through Endpoint 0x00, with the parameters described in the following table, to enable or disable the UART or to assert or deassert the RTS line.

Table 4-3. SET_CONTROL_LINE_STATE - Command

Byte Index	Command		Description
0	0x21	bmRequestType	Host to device, class-specific, Interface 0
1	0x22	bRequest	Set flow-control line state
2	0x00	Close UART	wValue
	0x01	De-assert RTS line	0x0000 → Disables the UART peripheral
	0x02	Reserved	0x0001 → Deasserts the RTS line (reception is disabled)
	0x03	Assert RTS line	0x0002 → Reserved
3	0x00	wValue – higher byte	0x0003 → Asserts the RTS line (Reception is enabled)
4	0x00	wIndex – lower byte	Interface 0
5	0x00	wIndex – higher byte	
6	0x00	wLength – lower byte	Payload length in bytes
7	0x00	wLength – higher byte	

4.1.3. GET_LINE_CODING Command and its Response

The GET_LINE_CODING command retrieves the current UART configuration.

Send this request through Endpoint 0x0, with the parameters described in the GET_LINE_CODING - Command table, to retrieve the current UART configuration. The configuration can be read from Endpoint 0x00 in the format given in the GET_LINE_CODING - Response table.

Table 4-4. GET_LINE_CODING - Command

Byte Index	Command		Description
0	0xA1	bmRequestType	Host to device, class-specific, Interface 0
1	0x21	bRequest	Read the UART baud rate, stop bits, parity, and data width
2	0x00	wValue – lower byte	Reserved
3	0x00	wValue – higher byte	
4	0x00	wIndex – lower byte	Interface 0
5	0x00	wIndex – higher byte	

Table 4-4. GET_LINE_CODING - Command (continued)

Byte Index	Command	Description
6	0x07	wLength – lower byte
7	0x00	wLength – higher byte

Table 4-5. GET_LINE_CODING - Response

Byte Index	Response	Description
0	L (Baud)	dwBaud – lowest byte
1	H1 (Baud)	dwBaud – higher byte
2	H2 (Baud)	dwBaud – next higher byte
3	H3 (Baud)	dwBaud – highest byte
4	0x00 0x01 0x02	1 Stop bit 1.5 Stop bit 2 Stop bits
5	0x00 0x01 0x02	No parity Odd parity Even parity
6	0x05 0x06 0x07 0x08	5-bit data width 6-bit data width 7-bit data width 8-bit data width

4.1.4. SERIAL_STATE Notification

The MCP2222 notifies parity or framing errors through Interrupt Endpoint 0x01 (IN). The format is given in the following table.

Table 4-6. SERIAL_STATE - Notification

Byte Index	Notification	Description
0	0xA1	vmRequestType
1	0x20	bRequest
2	0x00	wValue – lower byte
3	0x00	wValue – higher byte
4	0x00	wIndex – lower byte
5	0x00	wIndex – higher byte
6	0x02	wLength – lower byte
7	0x00	wLength – higher byte
8	0x1x 0x2x	Framing error Parity error
9	0x00	wState - higher byte

4.1.5. UART Communication Procedure

This section describes the typical command sequence for using the MCP2222 for I²C communication.

If hardware flow control needs to be enabled, ensure that IO0 and IO1 are configured as CTS and RTS pins, respectively.

Configure UART:

Send the SET_LINE_CODING request through Endpoint 0x00 with the following parameters:

Parameter	Value	Description
bRequest type	0x21	Host to device
bRequest code	0x20	SET_LINE_CODING request code
wValue	0x0000	
wIndex	0x0000	Interface 0
wPayload length	0x0007	Payload length
dwBaudrate	0xdwBaudrate	Baudrate in 32-bit hexadecimal format
bNumber of Stop bits	0x00 0x02	One Stop bit Two Stop bits
bParity type	0x00 0x01 0x02	No parity Odd parity Even parity
bData width in bits	0x05 - 0x8	Data width in bits

Send the SET_CONTROL_LINE_STATE request through Endpoint 0x00 with the following parameters:

Parameter	Value	Description
bRequest type	0x21	Host to device
bRequest code	0x22	SET_CONTROL_LINE_STATE request code
wValue	0x0003	RTS line asserted
wIndex	0x0000	Interface 0
wPayload length	0x0000	Payload length

When terminal emulators, such as PuTTY, Tera Term, VS Code Serial Monitor, or any other applications are used, they take care of sending the above requests.

Data Transfer**Transmit:**

1. Wait until CTS is asserted (skip this step if flow control is not used).
2. Write to Endpoint 0x02 (OUT).

Continue the above steps until all the data are transmitted.

Receive:

1. Read from Endpoint 0x82 (IN)

Repeat the above step until Endpoint 0x82 (IN) is empty.

Abort/Close Communication:

Send the SET_CONTROL_LINE_STATE request through Endpoint 0x00 with the following parameters:

Parameter	Value	Description
bRequest type	0x21	Host to device
bRequest code	0x22	SET_CONTROL_LINE_STATE request code
wValue	0x0000	RTS line deasserted
wIndex	0x0000	Interface 0
wPayload length	0x0000	Payload length

Error Notification:

If any error occurs during communication (reception), it is reported using the SERIAL_STATE notification through Interrupt Endpoint 0x01 (IN). The byte indexed 8 represents the error status.

- 0x1x → Framing error
- 0x2x → Parity error

5. I²C COMMUNICATION

The MCP2222 supports I²C communication through the USB vendor bulk interface.

I²C commands, transmitted data, and received data are exchanged using the vendor endpoints assigned to the I²C interface.

The general command and response formats are as follows:

- Command format: bCommandCode + Payload
- Response format: wResponseCode + Payload

5.1. I²C Interface Commands and Their Responses

The MCP2222 supports I²C functionality through vendor class, providing commands that allow the USB host to configure the I²C module, write data to an I²C client device, read data from an I²C client device, perform combined write/read transactions, check SCL/SDA line states, and attempt bus recovery.

The following endpoints are used for I²C communication:

- Endpoint 0x03 (OUT) — Used to send I²C commands and transmit I²C data
- Endpoint 0x83 (IN) — Used to read I²C command responses and received I²C data

Table 5-1. Supported I²C Interface Commands

Command	Description
I2C_Config	Configures the I ² C module with the following parameters: 1. Specified clock speed. 2. Specified clock-stretching timeout.
I2C_Write	Initiates an I ² C_Write transaction of "wWrDLen" bytes of "vDatPac" to the I ² C client device addressed by "bAddress"
I2C_Read	Initiates an I ² C_Read transaction of "wRdDLen" of bytes from the I ² C client device addressed by "bAddress"
I2C_WriteRead	Initiates an I ² C_Write transaction of "wWrDLen" bytes of "vDatPac", followed by an I ² C_Read transaction of "wRdDLen" bytes from the I ² C client device addressed by "bAddress"
I2C_GetSCLSDAStatus	Returns the current logic state of the SCL and SDA lines
I2C_BusRecovery	Initiates up to nine clock pulses on the SCL line and attempts to issue an I ² C_Stop condition

5.1.1. I²C_Config Command and its Response

The I2C_Config command (0x40) configures the I²C peripheral with the specified clock speed and clock-stretch timeout.

This command is sent through Endpoint 0x03 (OUT). The response is read through Endpoint 0x83 (IN).

Table 5-2. I2C_Config - Command

Byte Index	Command	Description
0	0x40	Command code
1	0x70	Speed: 100 kHz
	0x16	400 kHz
	0x04	1 MHz

Table 5-2. I2C_Config - Command (continued)

Byte Index	Command		Description
2	0x01	Timeout: 1 ms	I ² C clock-stretch timeout
	0x02	2 ms	
	...	...	
	0xFA	250 ms	

Table 5-3. I2C_Config - Response

Byte Index	Response		Description
0	0x40	Return command code	Returns whether the I2C_Config command was accepted or rejected
1	0x20	Accepted – success	The command is accepted
	0x60	Rejected – wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected – I ² C is busy	The command is rejected because the I ² C module is busy

5.1.2. I2C_Write Command and its Response

The I2C_Write command (0x44) initiates an I²C write transaction. The command writes `wWrDLen` bytes of the data, `vDatPac` to the I²C client device addressed by `bAddress`.

This command is sent through Endpoint 0x03 (OUT). The response is read through Endpoint 0x83 (IN).

If the write data length exceeds the payload capacity of a single USB packet, multiple packets are used. The first packet contains up to 60 data bytes, and each additional continuation packets contains up to 62 data bytes.

A response is provided after the initial command packet, after each continuation packet, and after the write transaction completes or is aborted.

Table 5-4. I2C_Write - Command

Serial No.	Byte Index	Command		Description
1	0	0x44	Command code	Initiates the I ² C write transaction
2	1	bAddress	I ² C client address	7-bit I ² C client address
3	2	L(wWrDtLen)	Write data length (lower byte)	Number of data bytes; maximum length: 65,535 bytes
4	3	H(wWrDtLen)	Write data length (upper byte)	
5	4	b0(vDatPac)	First data byte	The first 60 data bytes
6	5	b1(vDatPac)	Second data byte	
7-63	...	...	Additional data bytes	
64	63	b59(vDatPac)	60 th data byte	
65	Wait for the first response from the MCP2222 to determine whether the command is accepted. If the data length is more than 60 bytes, continue writing.			
66	0	0x44	Repeat the command code	Write is continued
67	1	0xA0	Write is continued	The next 62 data bytes
68	2	b60(vDatPac)	The 61 st data byte	
69-126	...	...	Additional data bytes	
127	63	b121(vDatPac)	The 122 nd data byte	
128	Wait for the next response			
129	If all data bytes have not been written, continue writing by repeating the above steps from serial number 66 to 128 until all data bytes are written. The final packet may contain fewer than 62 data bytes.			

Table 5-4. I2C_Write - Command (continued)

Serial No.	Byte Index	Command	Description
130		Continue polling for the write-completion response	

Table 5-5. I2C_Write - First Response

Byte Index	First Response		Description
0	0x44	Return command code	Returns whether the I ² C Write command was accepted or rejected
1	0x22	Accepted - initiated	The command is accepted, and an I ² C write transaction is initiated
	0x60	Rejected - wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected - I ² C is busy	The command is rejected because the I ² C module is busy

Table 5-6. I2C_Write - Subsequent Response

Byte Index	Subsequent Responses		Description
0	0x44	Return command code	The I ² C write is continued.
1	0x22	Accepted - write continued	
	0x68	Rejected - buffer full	The Write buffer is full. Retry sending the same packet until it is accepted.
	0x80	Failure - address NACK	An address NACK was received. Verify the I ² C client address.
	0x84	Failure - data NACK	A data NACK was received. Verify the client device requirements and the data length.
	0x88	Failure - clock-stretch timeout	A clock-stretch timeout occurred. Recover the bus and retry, if required.
	0x8C	Failure - arbitration lost	Bus arbitration was lost; another I ² C host may be using the bus.
	0x20	Success - write completed	The I2C_Write transaction is completed.
	0x90	Failure - bus error	An I ² C bus error occurred.

5.1.3. I2C_Read Command and its Response

The I2C_Read command (0x48) initiates an I²C read transaction. The command reads `wRdDtLen` bytes of data from the I²C client device addressed by `bAddress`.

This command is sent through Endpoint 0x03 (OUT) and the response is read through Endpoint 0x83 (IN).

Depending on the requested read length, multiple response packets may be returned. The first response may contain up to 60 data bytes and subsequent response packets may contain up to 62 data bytes each.

Table 5-7. I2C_Read - Command

Sl. No.	Byte Index	Command		Description
1	0	0x48	Command code	Initiates the I ² C read transaction
2	1	bAddress	I ² C client address	7-bit I ² C client address
3	2	L(wRdDtLen)	Read data length (lower byte)	Number of data bytes; maximum length 65535 bytes
4	3	H(wRdDtLen)	Read data length (upper byte)	
5	Wait for the first response from the MCP2222 to determine whether the command is accepted			
6	If accepted, continue reading until the required number of bytes are read or until a failure occurs.			

Table 5-8. I2C_Read - First Response

Byte Index	First Response		Description
0	0x48	Return command code	Returns whether the I2C_Read command was accepted.
1	0x22 0x60 0x64	Accepted – initiated Rejected – wrong parameters Rejected – I2C is busy	The command is accepted and the read transaction is initiated. The command is rejected because the parameters are invalid. The command is rejected because the I2C module is busy.

Table 5-9. I2C_Read - Second Response

Byte Index	Second Response		Description
0	0x48	Return command code	The I2C read is continued.
1	0x21 0x80 0x88 0x8C 0x90	Read is continued Failure - address NACK Failure - clock-stretch timeout Failure - arbitration lost Failure - an I2C bus error occurred	
2	L(wRdDtLen)	Data length (lower byte)	Number of data bytes to be read (maximum 65,535 bytes).
3	H(wRdDtLen)	Data length (higher byte)	
4	b0(vDatPac)	First data byte	The first 60 data bytes are read
5	b1(vDatPac)	Second data byte	
...	...	...	
63	b59(vDatPac)	The 60 th data byte	

Table 5-10. I2C_Read - Subsequent Responses

Byte Index	Subsequent Responses		Description
0	0x48	Return command code	The I2C_Read is continued
1	0x21 0x88 0x8C 0x90	Read is continued Failure - clock-stretch timeout Failure - arbitration lost Failure - an I2C bus error occurred	
2	b60(vDatPac)	The 61 st data byte	The next 62 data bytes are read
...	...	...	
63	b121(vDatPac)	The 122 nd data byte	

Repeat the subsequent response sequence until all requested bytes have been received or an error occurs.

5.1.4. I2C_WriteRead Command and its Response

The I2C_WriteRead command (0x4A) initiates a combined I2C write followed by an I2C read. The command writes wWrDtLen bytes of vDatPac and then reads wRdDtLen bytes from the I2C client device addressed by bAddress.

This command is sent through Endpoint 0x03 (OUT) and the response is read through Endpoint 0x83 (IN).

If the write data length exceeds the available payload space in the first command packet, additional write data is transmitted using continuation packets. A response is available after each transmitted packet.

Table 5-11. I2C_WriteRead - Command

Sl. No.	Byte Index	Command		Description
1	0	0x4A	Command code	Initiates the I ² C write-read transaction
2	1	bAddress	I ² C client address	7-bit I ² C client address
3	2	L (wWrDtLen)	Write data length (lower byte)	Number of data bytes to be written; maximum length 65535 bytes
4	3	H (wWrDtLen)	Write data length (upper byte)	
5	4	L (wRdDtLen)	Read data length (lower byte)	Number of data bytes to be read; maximum length 65535 bytes
6	5	H (wRdDtLen)	Read data length upper byte	
7	6	b0 (vDatPac)	First data byte	The N (up to 58) data bytes Note: The write data length can be up to 65,535 bytes. If the length exceeds 58 bytes, use multiple data packets, as done for I2C_Write.
8	7	b1 (vDatPac)	Second data byte	
9	...	...	...	
10	n+5	bn-1 (vDatPac)	The N th data byte	
11	Wait for the first response to determine whether the command was accepted			
12	If accepted, continue polling for read data until all requested bytes are received or an error occurs.			

Table 5-12. I2C_WriteRead - First Response

Byte Index	Response		Description
0	0x4A	Return command code	Returns whether the I2C_WriteRead command was accepted.
1	0x22	Accepted – initiated	The command is accepted, an I ² C write is initiated, and then an I ² C read is performed.
	0x60	Rejected – wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected – I ² C is busy	The command is rejected because the I ² C module is busy

Table 5-13. I2C_WriteRead - Second Response

Byte Index	Second Response		Description
0	0x4A	Return command code	The I2C_WriteRead command is continued
1	0x21	I2C_WriteRead is continued	
	0x80	Failure - address NACK	An address NACK was received. Verify the I ² C client address.
	0x88	Failure - clock-stretch timeout	A clock-stretch timeout occurred. Recover the bus and retry, if required.
	0x8C	Failure - arbitration lost	Bus arbitration was lost
	0x90	Failure - an I ² C bus error occurred	A bus error occurred
2	L(wRdDtLen)	Read data length (lower byte)	Number of data bytes to be read (maximum 65,535 bytes)
3	H(wRdDtLen)	Read data length (upper byte)	

Table 5-13. I2C_WriteRead - Second Response (continued)

Byte Index	Second Response		Description
4	b0 (vDatPac)	First data byte	The first 60 data bytes are read
5	b1 (vDatPac)	Second data byte	
...	...	...	
63	b59 (vDatPac)	The 60 th data byte	

Table 5-14. I2C_WriteRead - Subsequent Response

Byte Index	Subsequent Response		Description	
0	0x4A	Return command code	The I ² C_WriteRead command is continued	
1	0x21	The I2C_WriteRead operation is continued		
	0x88	Failure - clock-stretch timeout		A clock-stretch timeout occurred
	0x8C	Failure - arbitration lost		Bus arbitration was lost
4	b60 (vDatPac)	The 61 st data byte	The next 62 data bytes are read	
...	. . .	...		
63	b121 (vDatPac)	The 122 nd data byte		
The subsequent response sequence is repeated until all requested bytes have been received or an error occurs				

5.1.5. I2C_GetSCLSDAStatus Command and Its Response

The I2C_GetSCLSDAStatus command (0x4C) reads the current logic states of the SCL and SDA lines.

This command is sent through Endpoint 0x03 (OUT) and the response is read through Endpoint 0x83 (IN).

Table 5-15. I2C_GetSCLSDAStatus - Command

Byte Index	Command		Description
0	0x4C	Command code	Returns the state of the SCL and SDA lines

Table 5-16. I2C_GetSCLSDAStatus - Response

Byte Index	Response		Description
0	0x4C	Return command code	Returns the command code
1	0x21	Accepted - success	The I2C_GetSCLSDAStatus command is accepted
2	0x11	SCL = SDA = High	Returns the current logical state of the SCL and SDA lines
	0x01	SCL = High , SDA = Low	
	0x10	SCL = Low, SDA = High	
	0x00	SCL = SDA = Low	

5.1.6. I2C_BusRecovery Command and its Response

The I2C_BusRecovery command (0x4E) generates up to nine clock pulses on the SCL line and attempts to issue an I2C Stop condition.

This command is typically used after a clock-stretch timeout or when the I2C bus does not return to the Idle state.

This command is sent through Endpoint 0x03 (OUT) and the response is read through Endpoint 0x83 (IN).

Table 5-17. I2C_BusRecovery - Command

Byte Index	Command		Description
0	0x4E	Command code	Attempts I ² C bus recovery by generating up to nine SCL clock pulses and issuing an I ² C STOP condition

Table 5-18. I2C_BusRecovery - Response

Byte Index	Response		Description
0	0x4E	Return command code	Returns whether the command code was accepted
1	0x20	Accepted – success	Bus recovery completed successfully; the I ² C bus is released and in the idle state
	0x60	Rejected – no ClkLowTimeout	Rejected because no clock-stretch timeout has occurred
	0x8C	Accepted – failure	The I ² C bus is not released and is not in the idle state

5.1.7. I²C Communication Procedure

This section describes the typical command sequence for using the MCP2222 for I²C communication.

Writing to an I²C client device from a specified location:

1. Configure I²C:

- Send the I2C_Config command through Endpoint 0x03 (OUT) with the following parameters:

Parameter	Value	Description
Command	0x40	Configure I ² C
Clock speed	0x70 0x16 0x04	100 kHz clock speed. 400 kHz clock speed. 1 MHz clock speed.
Clock stretch timeout	0x01 - 0xFA	Clock-stretch timeout in milliseconds (8-bit hexadecimal)

- Read the response through Endpoint 0x83 (IN). The responses can be one of the following:

Response code	Status	Action
0x40	Response to I ² C Config	Continue reading
0x20	I ² C is configured.	Proceed with the write transaction.
0x60	Invalid parameters.	Retry with valid parameters.
0x64	I ² C module busy.	Wait and retry later.

2. Write transaction

- Send the I2C_Write command through Endpoint 0x03 (OUT) with the following parameters:

Parameter	Value	Description
Command	0x44	Write to the I ² C client device
I ² C Client address	bAddress	7-bit I ² C client address
Write data length	wWrDtLen	Number of bytes to be written including the location address, encoded as a 16-bit hexadecimal value
Location address	vLocAdd	Starting location from where the data should be written
Data bytes	vDatPac	Data to be written (up to 60 bytes, including the location address)

- Read the initial response from Endpoint 0x83 (IN). Possible initial responses are:

Response code	Status	Action
0x44	Response to I ² C Write	Continue reading

I2C Communication Procedure (continued)

Response code	Status	Action
0x22 0x60 0x64	The I ² C write transaction is initiated. Invalid parameters. The I ² C module is busy.	Proceed to continue the write transaction. Retry with valid parameters. Wait and retry later.

- If the number of data bytes to write exceeds 60 bytes, send continuation packets through Endpoint 0x03 (OUT). Each continuation packet includes:

Parameter	Value	Description
Command	0x44	Continue writing to the I ² C client device
Continuation flag	0xA0	
Data bytes	vDatPac	Data to be written (up to 62 bytes)

- After each continuation packet, and after the write transaction is completed, read the response from Endpoint 0x83 (IN). The response can be one of the following:

Response code	Status	Action
0x44 NACK	Response to I ² C Write. No response available yet.	Continue reading. Continue polling.
0x22 0x68 0x80 0x84 0x88 0x8C 0x20	The I ² C write transaction is continued. The buffer is full. An address NACK occurred. A data NACK occurred. A clock-stretch timeout occurred. I ² C bus arbitration was lost. The I ² C write transaction is successfully completed.	Proceed to continue the write transaction. Retry until the packet is accepted. Retry with a valid address. Use a smaller data packet. Run bus recovery. Wait and retry (I ² C bus in use).

- Repeat sending continuation packet through Endpoint 0x03 (OUT) and polling Endpoint 0x83 (IN) until all required data bytes are written.

Reading data from an I²C client device from a specified location:

- Configure I²C:
 - Refer to the Configure I²C section of Writing to a I²C client device from a specified location.
- Read data from specific locations of an I²C client device:
 - Send the I2C_WriteRead command through Endpoint 0x03 (OUT) with the following parameters.

Parameter	Value	Description
Command	0x4A	Write to and read from an I ² C client device
I ² C Client address	bAddress	7-bit I ² C client address
Write data length	wWrDtLen	Number of bytes in the location address, encoded as a 16-bit hexadecimal value
Read data length	wRdDtLen	Number of bytes to be read, encoded as a 16-bit hexadecimal value
Location address	vLocAdd	Starting location from where the data should be read

- Read the initial response from Endpoint 0x83 (IN). Possible initial responses are:

Response code	Status	Action
0x4A	Response to I ² C WriteRead	Continue reading

I2C Communication Procedure (continued)

Response code	Status	Action
0x22	I ² C write-read transaction is initiated.	Proceed to continue polling for the next response.
0x60	Invalid parameters	Retry with valid parameters.
0x64	The I ² C module is busy.	Wait and retry later.

- Keep polling Endpoint 0x83 (IN) until read data is returned. Possible poll results:

Response code	Status	Action
0x4A	Response to I ² C WriteRead.	Continue reading.
NACK	No response available yet.	Continue polling.
0x21	The I ² C write-read transaction continued.	Continue reading until the entire packet is read, then proceed to poll for the next response.
0x84	A data NACK occurred.	Retry with the correct location address.
0x88	A clock-stretch timeout occurred.	Run bus recovery.
0x8C	I ² C bus arbitration was lost.	Wait and retry (I ² C bus in use).
wRdDtLen	The first 60 bytes of data are read	Continue reading
vDatPac		Read all the available data bytes from the endpoint. Continue reading until all packets have been read.

- Repeat polling Endpoint 0x83 (IN), until all required data bytes are read:

Response code	Status	Action
0x4A	Response to I ² C WriteRead.	Continue reading.
NACK	No response available yet.	Continue polling.
0x21	The I ² C write-read transaction continued.	Continue reading until the entire packet is read, then proceed to poll for the next response.
0x88	A clock-stretch timeout occurred.	Run bus recovery.
0x8C	I ² C bus arbitration was lost.	Wait and retry (I ² C bus in use).
vDatPac	The next 62 bytes of data are read.	Read all available data bytes from the endpoint. Continue reading until all packets have been read.

Recovering the I²C Bus on a Clock-Stretch Timeout

Send the I2C_BusRecovery command through Endpoint 0x03 (OUT) if a clock-stretch timeout occurs during a write, read, or writ-read transaction.

Parameter	Value	Description
Command	0x4E	Recover the I ² C bus

Read the response from Endpoint 0x83 (IN).

Response code	Status	Action
0x4E	Response to I ² C bus recovery	Continue reading
0x20	The I ² C bus is recovered.	The I ² C bus is in the idle state.
0x60	No clock-stretch timeout was detected.	The bus is available for transaction.
0x80	Bus recovery failed.	Retry by sending the command again.

Reading the logic states of the SCL and SDA lines:

To check the states of the SCL and SDA lines, follow these steps:

1. Send the I2C_GetSCLSDAStatus (0x4C) through Endpoint 0x03 (OUT).

Parameter	Value	Description
Command	0x4C	Recover the I ² C bus

2. Read the response from Endpoint 0x83(IN).

Response code	Status	Action
0x4C	Response to I ² C bus recovery	Continue reading
0x21	SCL and SDA states are read	Continue reading
0x00 0x00 0x00 0x00	The SCL is in a low state, and the SDA is also in a low state The SCL is in a high state, and the SDA is in a low state The SCL is in a low state, and the SDA is in a high state The SCL is in a high state, and the SDA is also in a high state	Take the necessary action depending on the logical state read

6. SPI Communication

The MCP2222 supports SPI communication through the USB vendor bulk interface.

SPI commands, transmitted data, and received data are exchanged using the vendor endpoints assigned to the SPI interface.

The general command and response formats are as follows:

- Command format: bCommandCode + Payload
- Response format: wResponseCode + Payload

6.1. SPI Interface Commands and Their Responses

The MCP2222 supports SPI functionality through the vendor class, providing commands that allow the USB host to configure the SPI module, write data to an SPI client device, read data from an SPI client device, and perform full-duplex write-and-read transactions.

The following endpoints are used for SPI communication:

- Endpoint 0x06 (OUT) — Used to send SPI commands and transmit SPI data
- Endpoint 0x86 (IN) — Used to read SPI command responses and received SPI data

Table 6-1. Supported SPI Interface Commands

Command	Description
SPI_Config	Configures the SPI module with: 1. A specified clock speed. 2. A specified SPI mode. 3. The specified chip-select set to idle state. 4. Transaction delays between: • Chip-select assertion and the first data byte • Two consecutive data bytes • The last data byte and chip-select deassertion
SPI_Write	Asserts the configured chip-select, applies the configured delay, writes “wDatLen” bytes of “vDatPac” to the SPI client device, and then de-asserts the chip-select after the configured delay
SPI_Read	Asserts the configured chip-select, applies the configured delay, reads “wDatLen” bytes of data from the SPI client device, and then de-asserts the chip-select after the configured delay
SPI_Exchange	Performs a full-duplex SPI transaction by writing wDatLen bytes from vDatPac while simultaneously reading wDatLen bytes from the SPI client device

6.1.1. SPI_Config Command and its Response

The `SPI_Config` command (0x20) configures the SPI peripheral with the specified clock speed, SPI mode, chip-select and its active state, and the delays between chip-select assertion and the first data byte between the data bytes, and between the last data byte and de-assertion of the chip-select.

Note: An additional internal delay of approximately 6 μ s (± 2 μ s) is always present, independent of the configured delay value. The total delay equals the configured Delay plus approximately 6 μ s. This variance must be considered for timing-sensitive applications.

This command is sent through Endpoint 0x06 (OUT). The response is read through Endpoint 0x86 (IN).

Table 6-2. SPI_Config - Command

Byte Index	Command		Description
0	0x20	Command code	Configures the SPI module
1	0x30	Speed: 12 MHz	SPI clock speed
	0x20	6 MHz	
	0x32	3 MHz	
	0x22	1.5 MHz	
	0x34	750 kHz	
	0x24	375 kHz	
	0x26	187.5 kHz	
2	0x44	Mode: 0	SPI mode
	0x45	1	
	0x46	2	
	0x47	3	
3	0x00	CS-ActStat:	Chip-select and its assertion state
	0x01	CS0 active low	
	0x40	CS0 active high	
	0x41	CS1 active low	
	0x80	CS1 active high	
	0x81	CS2 active low	
	0xC0	CS2 active high	
	0xC1	CS3 active low	
		CS3 active high	
4-5	0x0000	Delay: 0 μ s	Delay between: - Chip-select assertion and the first data byte - Between each data byte - The last data byte and chip-select de-assertion.
	0x000A	10 μ s	
	0x0014	20 μ s	
	...	...	
	0x03E8	1000 μ s	
	...	...	
	0x1388	5000 μ s	

Table 6-3. SPI_Config - Response

Byte Index	Response		Description
0	0x20	Return command code	Returns whether the SPI_Config command code was accepted
1	0x20	Accepted - success	The command is accepted
	0x60	Rejected – wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected – SPI is busy	The command is rejected because the SPI module is busy
	0x6C	Rejected – wrong CS pin	The command is rejected because the selected chip-select pin is not configured as a CS function. Configure the corresponding I/O pin as a chip-select pin and resend the command.

6.1.2. SPI_Write Command and its Response

The SPI_Write command (0x24) initiates an SPI write transaction. The command writes wDatLen bytes of data, vWrDtPac, to the SPI client device selected by the active SPI configuration.

This command is sent through Endpoint 0x06 (OUT). The response is read through Endpoint 0x86 (IN).

If the write data length exceeds the payload capacity of a single USB packet, multiple packets are used. The first packet contains up to 60 data bytes, and each additional packet containing up to 62 data bytes.

A response is provided after the initial command packet, after each continuation packet, and after the write transaction completes.

Table 6-4. SPI_Write - Command

Serial No.	Byte Index	Command		Description
1	0	0x24	Command code	Initiates the SPI write transaction
2	1	0xAF	Command code extension byte	
3	2	L(wDatLen)	Data length (lower byte)	Number of data bytes. Maximum length: 65,535 bytes
4	3	H(wDatLen)	Data length (upper byte)	
5	4	b0(vWrDtPac)	First data byte	The first 60 bytes
6	5	b1(vWrDtPac)	Second data byte	
7-63	6-62	...	Additional data bytes	
64	63	b59(vWrDtPac)	The 60 th data byte	
65	Wait for the first response from the MCP2222 to determine whether the command is accepted. If the data length is more than 60 bytes, continue writing.			
66	0	0x24	Repeat command code	Write is continued
67	1	0xA0	Write is continued	
68	2	b60(vDatPac)	The 61 st data byte	The next 62 data bytes
69-126	3-62	...	Additional data bytes	
127	63	b121(vDatPac)	The 122 nd data byte	
128	Wait for the next response			
129	If all data bytes are not written, continue writing, by repeating the above steps from serial number 66 to 128 until all data bytes are written. The final packet may contain fewer than 62 data bytes.			
130	Continue polling for the write-completion response.			

Table 6-5. SPI_Write - First Response

Byte Index	Response		Description
0	0x24	Return command code	Returns whether the SPI_Write command was accepted
1	0x22	Accepted - initiated	The command is accepted, and an SPI write is initiated
	0x60	Rejected - wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected - SPI is busy	The command is rejected because the SPI module is busy

Table 6-6. SPI_Write - Subsequent Response After Every Packet Written

Byte Index	Response		Description
0	0x24	Return command code	The SPI write is continued
1	0x22	Accepted - write continued	
	0x68	Rejected - buffer full	The write buffer is full. Retry the same packet until it is accepted.
	0x20	Success - write completed	The SPI_Write transaction is completed

6.1.3. SPI_Read Command and its Response

The SPI_Read command (0x28) initiates an SPI read transaction. The command reads wDatLen bytes of data from the SPI client device selected by the active SPI configuration.

This command is sent through Endpoint 0x06 (OUT). The response is read through Endpoint 0x86 (IN).

If the read data length exceeds the payload capacity of a single USB packet, multiple packets are used. The first packet contains up to 60 data bytes, and each additional continuation packet contains up to 62 data bytes.

Depending on the requested read length, multiple response packets may be returned. The first data response may contain up to 60 bytes, and subsequent response packets may contain up to 62 bytes each.

Table 6-7. SPI_Read - Command

Byte Index	Command		Description
0	0x28	Command code	Initiates the SPI read transaction
1	0xAF	Command code extension byte	
2	L(wDatLen)	Data length (lower byte)	Number of data bytes. Maximum length: 65535 bytes.
3	H(wDatLen)	Data length (upper byte)	
Wait for the first response from the MCP2222 to determine whether the command is accepted			
If accepted, continue reading until the required number of bytes are read			

Table 6-8. SPI_Read - First Response

Byte Index	Response		Description
0	0x28	Return command code	Returns whether the SPI_Read command was accepted
1	0x22	Accepted – initiated	The command is accepted, and an SPI read is initiated
	0x60	Rejected – wrong parameters	The command is rejected because the parameters are invalid
	0x64	Rejected – SPI is busy	The command is rejected because the SPI module is busy

Table 6-9. SPI_Read - Second Response

Byte Index	Response		Description
0	0x28	Return command code	The SPI read is continued
1	0x21	Read continued	
2	L(wDatLen)	Data length (lower byte)	Number of data bytes to be read (maximum 65,535 bytes)
3	H(wDatLen)	Data length (higher byte)	
4	b0(vDatPac)	First data byte	The first 60 bytes are read
5	b1(vDatPac)	Second data byte	
...	...	...	
63	b59(vDatPac)	The 60 th data byte	

Table 6-10. SPI_Read - Subsequent Response

Byte Index	Response (Endpoint 0x86)		Description
0	0x28	Return command code	The SPI read is continued
1	0x21	Read continued	
2	b60 (vDatPac)	The 61 st Data Byte	The next 62 data bytes are read
...	...	...	
63	b121 (vDatPac)	The 122 nd Data Byte	
Repeat the subsequent response sequence until all requested bytes have been received.			

6.1.4. SPI_Exchange Command and its Response

The SPI_Exchangecommand (0x2C) initiates a full-duplex SPI transaction. The command writes wDatLen bytes from vWrDtPac to the SPI client device while simultaneously reading wDatLen bytes.

This command is sent through Endpoint 0x06 (OUT). The response is read through Endpoint 0x86 (IN).

If the exchange data length exceeds the payload capacity of a single USB packet, multiple packets are used. The first packet contains up to 60 write data bytes, and each additional continuation packet contains up to 62 write data bytes.

Table 6-11. SPI_Exchange - Command

Serial No.	Byte Index	Command		Description
1	0	0x2C	Command code	Initiates a full-duplex SPI transaction
2	1	0xAF	Command code extension byte	
3	2	L(wDatLen)	Data length (lower byte)	Number of data bytes. Maximum length: 65,535 bytes.
4	3	H(wDatLen)	Data length (upper byte)	
5	4	b0(vWrDtPac)	First data byte	The first 60 bytes
6	5	b1(vWrDtPac)	Second data byte	
7-63	6-62	...	...	
64	63	b59(vWrDtPac)	The 60 th data byte	
65	Wait for the first response from the MCP2222 to determine whether the command is accepted. If the data length is more than 60 bytes, continue writing.			
66		0x2C	Repeat command code	The exchange is continued
67	1	0xA0	The exchange is continued	The next 62 data bytes
68	2	b60(vWrDtPac)	The 61 st data byte	
69-126	...	...	Additional data bytes	
127	63	b121(vWrDtPac)	The 122 nd data byte	
128	Wait for the next response, which includes the data bytes read, and read them			
129	If all data bytes are not written, continue writing, by repeating the above steps from serial number 66 to 128 until all data bytes are written. The final packet may contain fewer than 62 data bytes.			
130	Read the final response			

Table 6-12. SPI_Exchange - First Response

Byte Index	Response		Description
0	0x2C	Returns command code	Returns whether the SPI_Exchange command was accepted
1	0x22	Accepted – initiated	The command is accepted, and a full-duplex SPI transaction is initiated
	0x60	Rejected – wrong parameters	The command is rejected because parameters are invalid
	0x64	Rejected – SPI is busy	The command is rejected because the SPI module is busy

Table 6-13. SPI_Exchange - Second Response

Byte Index	Response		Description
0	0x2C	Returns command code	The SPI full-duplex transaction is continued
1	0x21	Exchange is continued	
2	L(wDatLen)	Data length (lower byte)	Number of data bytes. Maximum length: 65,535 bytes
3	H(wDatLen)	Data length (upper byte)	
4	b0(vRdDtPac)	First data byte	The first 60 bytes are read
5	b1(vRdDtPac)	Second data byte	
...	...	...	
63	b59(vRdDtPac)	The 60 th data byte	

Table 6-14. SPI_Exchange - Subsequent Response After Packet Data is Read

Byte Index	Response		Description
0	0x2C	Return command code	The SPI full-duplex transaction is continued
1	0x21	Exchange is continued	
2	b60 (vRdDtPac)	The 61 st data byte	The next 62 data bytes are read
...	...	...	
63	b121 (vRdDtPac)	The 122 nd data byte	
The subsequent response sequence is repeated until all requested bytes have been exchanged			

6.1.5. SPI Communication Procedure

This section describes the typical command sequence for using the MCP2222 for SPI communication.

Writing to an SPI client device from a specified location:

1. Configure the SPI interface:

- Send SPI_Configure command through Endpoint 0x06, OUT with the following parameters:

Parameter	Value	Description
Command	0x20	Configure the SPI
Clock speed	0x30	12 MHz clock speed
	0x20	6 MHz clock speed
	0x32	3 MHz clock speed
	0x22	1.5 MHz clock speed
	0x34	750 kHz clock speed
	0x24	375 kHz clock speed
	0x26	187.5 kHz clock speed
SPI mode	0x44	SPI mode 0
	0x45	SPI mode 1
	0x46	SPI mode 2
	0x47	SPI mode 3
Chip-select and its active state	0x00	CS0 active low
	0x01	CS0 active high
	0x40	CS1 active low
	0x41	CS1 active high
	0x80	CS2 active low
	0x81	CS2 active high
	0xC0	CS3 active low
	0xC1	CS3 active high
Delay between:	0x0000	0 μs
• Chip-select assertion and the first data byte	0x000A	10 μs
• Each data bytes	0x0014	20 μs
	...	...
• The last data byte and chip-select de-assertion	0x03E8	1 ms
	...	...
	0x1388	5 ms

- Read the response through Endpoint 0x86 (IN). The response can be one of the following:

Response code	Status	Action
0x20	Response to SPI Config	Continue reading
0x20 0x60 0x64	The SPI interface is configured. Invalid parameters. The SPI module is busy.	Proceed to the write transaction. Retry with valid parameters. Wait and retry later.

Write transaction

- Send the SPI_Write command through Endpoint 0x06 (OUT) with the following parameters:

Parameter	Value	Description
Command	0x24 0xAF	Write to the SPI client device
Write data length	wDatLen	Number of bytes to be written including the location address, encoded as a 16-bit hexadecimal value
Location address	vLocAdd	Starting location from where the data should be written
Data bytes	vWrDtPac	Data to write (up to 60 bytes, including the location address)

- Read the response from Endpoint 0x86 (IN). Possible initial responses are:

Response code	Status	Action
0x24	Response to SPI Write	Continue reading
0x22 0x60 0x64	The SPI write transaction is initiated. Invalid parameters. The SPI module is busy.	Proceed to continue the write transaction. Retry with valid parameters. Wait and retry later.

- If the number of data bytes to write exceeds 60 bytes, send continuation packets through Endpoint 0x06 (OUT). Each continuation packet includes:

Parameter	Value	Description
Command	0x24	Write to the SPI client device is continued.
Continuation flag	0xA0	
Data bytes	vWrDtPac	Data to write (up to 62 bytes)

- After each continuation packet, and after the write transaction is completed, read the response from Endpoint 0x86 (IN). The response can be one of the following:

Response code	Status	Action
0x24 NACK	Response to SPI Write. No response available yet.	Continue reading. Continue polling.
0x22 0x68 0x20	The SPI write transaction is continued. The buffer is full. The SPI write transaction is successfully completed.	Proceed to continue the write transaction. Retry until the packet is accepted.

- Repeat sending the continuation packets through Endpoint 0x06 (OUT) and polling Endpoint 0x86 (IN) until all the required data bytes are written

Reading data from an SPI client device from a specified location:

- Configure the SPI interface:
 - Refer to the Configure SPI section of Writing to an SPI client device from a specified location.
- Read data from the specific locations of a SPI client device.
 - Send the SPI_Exchange command through Endpoint 0x06 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x2C	Write to and read from an SPI client device
	0xAF	
Read data length + 1 + bytes in the location address	wDatLen + 1 + bytes (vLocAdd)	The sum of the number of bytes to be read, the bytes in the read command, and the bytes in the location address, encoded as a 16-bit hexadecimal value.
SPI client Read command	bRdComnd	The read command specified by the SPI client device.
Location address	vLocAdd	Starting location from where the data should be read
(59 - bytes in vLocAdd) bytes of dummy bytes.	0x00	To generate the SPI clock for reading

- Read the initial response from Endpoint 0x86 (IN). Possible initial responses are:

Response code	Status	Action
0x2C	Response to SPI Exchange	Continue reading.
0x22 0x60	SPI Exchange transaction initiated. Invalid parameters.	Proceed to continue polling for the next response. Retry with valid parameters.
0x64	The SPI module is busy.	Wait and retry later.

- If the number of data bytes to read exceeds (59 bytes in vLocAdd) bytes, send continuation packets through Endpoint 0x06 (OUT). Each continuation packet includes:

Parameter	Value	Description
Command	0x2C	The exchange transaction is continued.
Continuation flag	0xA0	
62 bytes of dummy data	0x00	Dummy write (up to 62 bytes).

- Read the response from Endpoint 0x86 (IN). The response can be one of the following:

Response code	Status	Action
0x2C NACK	Response to SPI Exchange. No response available yet.	Continue reading. Continue polling.
0x21 0x68	The SPI exchange transaction is continued. The buffer is full.	Proceed to continue the write transaction. Retry until the packet is accepted.
wDatLen	The first 60 bytes of data are read	Continue reading
vRdDtPac		Read all the data bytes available in the Endpoint. Continue sending continuation packet until all packets are read.

- Repeat sending continuation packet through Endpoint 0x06 and polling the response packets through Endpoint 0x86 until the required data bytes are read:

Parameter	Value	Description
Command	0x2C	The exchange transaction is continued.
Continuation flag	0xA0	
62 bytes of dummy data	0x00	Dummy write (up to 62 bytes)

- Keep polling Endpoint 0x86 (IN) until final read data is returned. Possible poll results:

Response code	Status	Action
0x2C NACK	Response to SPI Exchange. No response available yet.	Continue reading. Continue polling.
0x21	The SPI Read transaction is continued.	Continue reading until the entire packet is read.
vRdDtPac	The next 62 bytes of data are read	Read all the data bytes available in the Endpoint

7. GPIO CONTROL

The MCP2222 supports GPIO control through the USB vendor-class interface.

GPIO commands, output states, input states, and I/O pin configuration settings are exchanged using the vendor endpoints assigned to the GPIO interface.

The general command and response formats are as follows:

- Command format: bCommandCode + Payload
- Response format: wResponseCode + Payload

7.1. GPIO Interface Commands and Their Responses

The MCP2222 supports GPIO control through the vendor class, providing commands that allow the USB host to configure and control the device's I/O pins.

The following endpoints are used for GPIO control:

- Endpoint 0x04 (OUT) → Used to send GPIO commands
- Endpoint 0x84 (IN) → Used to read GPIO command responses and input pin states

Table 7-1. GPIO Interface Commands

Command	Effect
IO_SetConfiguration	Configures the selected I/O pin
GPIO_Write	Sets the selected GPIO output pin to the selected logic state
GPIO_Read	Reads and returns the logic state of the selected GPIO pin
IO_GetConfiguration	Reads and returns the configuration of the selected I/O pin

7.1.1. IO_SetConfiguration Command and its Response

The `IO_SetConfiguration` command (0x60) configures the specified I/O pin with the selected settings:

The `bPinSetting` value includes the following bit fields:

- Bit 7, `Idle-State` – valid only for an output pin
 - 0 → Idle state is Low
 - 1 → Idle state is High
- Bit 6:5, `AlertEdge` – valid only for an alert pin
 - 00 → Alert/Interrupt disabled
 - 01 → Alert/Interrupt on falling edge
 - 10 → Alert/Interrupt on rising edge
 - 11 → Alert/Interrupt on both the edges

`ClkDiv` – valid only for the CLK0 pin

- 00 → 6 MHz clock
 - 01 → 4 MHz clock
 - 10 → 2 MHz
 - 11 → 1 MHz
- Bit 4, `Open-Drain` – valid only for an output pin
 - 0 → The pin is not a open-drain pin
 - 1 → The pin is a open-drain pin

- Bit 3, *PUSel* – valid only for an input pin
 - 0 → The pin is not pulled up internally
 - 1 → The pin is pulled up internally
- Bit 2, *IODir*
 - 0 → The pin is a output pin
 - 1 → The pin is a input pin
- Bit 1:0, *IODesignation* – varies with the I/O pin selected
 - 00 → GPIO function
 - 01 → CTS/RTS/RXA/TXA/USBCFG/USBSPND function
 - 10 → CS3/CS2/CS1/I2CTO/CS0/SPIA function
 - 11 → Alert0/I2CA/Alert1/CLKO/Alert2/Alert3 function

This command is sent through Endpoint 0x04 (OUT). The response is read through Endpoint 0x84 (IN).

Table 7-2. *IO_SetConfiguration - Command*

Byte Index	Command		Description
0	0x60	Command code	Configures the specified I/O pin
1	0x00	IONum: IO0	I/O number
	0x10	IO1	
	0x20	IO2	
	0x30	IO3	
	0x40	IO4	
	0x50	IO5	
2	bPinSetn	IO setting register value	I/O setting register value for the selected pin

Table 7-3. *IO_SetConfiguration - Response*

Byte Index	Response		Description
0	0x60	Returns the command code	Returns whether the status of the command was accepted
1	0x20	Accepted - successful	The command is accepted and acted upon
	0x60	Rejected – invalid configuration	The command rejected because the configuration is invalid
	0x64	Rejected – locked	The command rejected because the device configurations are locked

7.1.2. GPIO_Write Command and its Response

The *GPIO_Write* command (0x64) configures the specified GPIO output pin to the selected logic state.

This command is sent through Endpoint 0x04 (OUT). The response is read through Endpoint 0x84 (IN).

Table 7-4. *GPIO_Write - Command*

Byte Index	Command		Description
0	0x64	Command code	Sets the specified GPIO pin to the specified state

Table 7-4. GPIO_Write - Command (continued)

Byte Index	Command		Description
1	0x00	GPIONum-GPIOSat:	GPIO pin and its output state
	0x01	GPIO0 Low	
	0x10	GPIO0 High	
	0x11	GPIO1 Low	
	0x20	GPIO1 High	
	0x21	GPIO2 Low	
	0x30	GPIO2 High	
	0x31	GPIO3 Low	
	0x40	GPIO3 High	
	0x41	GPIO4 Low	
	0x50	GPIO4 High	
	0x51	GPIO5 Low	
		GPIO5 High	

Table 7-5. GPIO_Write - Response

Byte Index	Response		Description
0	0x64	Returns command code	Returns whether the GPIO_Write command was accepted
1	0x20	Accepted - success	Accepted, and the output state is written
	0x60	Rejected - invalid pin number	Rejected because the pin number is invalid
	0x68	Rejected - not a GPIO	Rejected because the selected pin is not configured as a GPIO

7.1.3. GPIO_Read Command and its Response

The GPIO_Read command (0x68) reads the logic state of the specified GPIO pin.

This command is sent through Endpoint 0x04 (OUT). The response is read through Endpoint 0x84 (IN).

Table 7-6. GPIO_Read - Command

Byte Index	Command		Description
0	0x68	Command code	Reads the specified GPIO pin and returns the state
1	0x00	GPIO0	GPIO pin
	0x10	GPIO1	
	0x20	GPIO2	
	0x30	GPIO3	
	0x40	GPIO4	
	0x50	GPIO5	

Table 7-7. GPIO_Read - Response

Byte Index	Response		Description
0	0x68	Returns the command code	Returns whether the GPIO_Read command was accepted along with the pin state
1	0x21	Accepted - success	Accepted, and the pin is read.
	0x60	Rejected - not a GPIO pin .	Rejected because the selected pin is not configured as a GPIO.
2	0x00	The GPIO pin state is LOW.	The GPIO pin state
	0x01	The GPIO pin state is HIGH.	

7.1.4. IO_GetConfiguration Command and its Response

The `IO_GetConfiguration` command (0x6C) reads the configuration settings of the specified I/O pin.

This command is sent through Endpoint 0x04 (OUT). The response is read through Endpoint 0x84 (IN).

Table 7-8. `IO_GetConfiguration` - Command

Byte Index	Command		Description
0	0x6C	Command code	Reads the selected I/O pin configuration.
1	0x00	IO0	IO pin
	0x10	IO1	
	0x20	IO2	
	0x30	IO3	
	0x40	IO4	
	0x50	IO5	

Table 7-9. `IO_GetConfiguration` - Response

Byte Index	Response		Description
0	0x6C	Returns the command code	Returns whether the command was accepted, along with the configuration
1	0x21 0x60	Accepted - success Rejected - wrong parameters	Accepted, and the I/O pin configuration is read. Rejected because the I/O number is invalid.
2	bPinSetn	I/On setting register value.	I/O setting register value for the selected pin

7.1.5. GPIO Controlling Procedure

This section describes the typical command sequence for using the MCP2222 GPIO functions.

Configure GPIO

Configuring an I/O pin as a normal output pin:

- Send the `IO_SetConfiguration` command through Endpoint 0x04 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x60	Configure I/O pin
I/O number	0x00 0x10 0x20 ... 0x50	IO0 IO1 IO2 ... IO5
I/O setting	0x00	Refer to the corresponding I/O Setting register for the setting value.

- Read the response from Endpoint 0x84 (IN). The response can be one of the following:

Response Code	Status	Action
0x60	Response to the I/O Set Configuration	Continue reading
0x20 0x60 0x64	The I/O pin is configured Invalid parameters. Device configuration is locked.	Retry with valid parameters. Unlock the device configuration and retry.

Configuring an I/O pin as an open-drain output pin:

- Send the IO_SetConfiguration command through Endpoint 0x04 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x60	Configure I/O pin
I/O number	0x00	IO0
	0x10	IO1
	0x20	IO2
	...	...
	0x50	IO5
I/O setting	0x10	Refer to the corresponding I/O setting register for the setting value

- Read the response from Endpoint 0x84 (IN). The response can be one of the following:

Response Code	Status	Action
0x60	Response to the I/O Set Configuration	Continue reading
0x20	I/O is configured.	Retry with valid parameters.
0x60	Invalid parameters.	Unlock the device configuration and retry.
0x64	Device configuration is locked.	

Configuring an I/O pin as a normal input pin:

- Send the IO_SetConfiguration command through Endpoint 0x04 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x60	Configure I/O pin
I/O number	0x00	IO0
	0x10	IO1
	0x20	IO2
	...	...
	0x50	IO5
I/O setting	0x04	Refer to the corresponding I/O setting register for the setting value

- Read the response from Endpoint 0x84 (IN). The responses can be one of the following:

Response Code	Status	Action
0x60	Response to the I/O Set Configuration	Continue reading
0x20	I/O is configured.	Retry with valid parameters.
0x60	Invalid parameters.	Unlock the device configuration and retry.
0x64	Device configuration is locked.	

Configuring an I/O pin as an internally pulled-up input pin

- Send the IO_SetConfiguration command through Endpoint 0x04 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x60	Configure I/O pin

GPIO Controlling Procedure (continued)

Parameter	Value	Description
I/O number	0x00	IO0
	0x10	IO1
	0x20	IO2
	...	...
	0x50	IO5
I/O setting	0x0C	Refer to the corresponding IO setting register for the setting value

- Read the response from Endpoint 0x84 (IN). The response can be one of the following:

Response code	Status	Action
0x60	Response to the I/O set configuration	Continue reading
0x20	The I/O is configured.	Retry with valid parameters.
0x60	Invalid parameters.	Unlock the device configuration and retry.
0x64	Device configuration is locked.	

Controlling a GPIO:**Setting or clearing an output pin:**

- Send the GPIO_Write command through Endpoint 0x04, OUT, with the following parameters:

Parameter	Value	Description
Command	0x64	Configure I/O pin
I/O number and output state	0x00	IO0 Low
	0x01	IO0 High
	0x10	IO1 Low
	0x11	IO1 High
	...	...
	0x50	IO5 Low
	0x51	IO5 High

- Read the response from Endpoint 0x84 (IN). The response can be one of the following:

Response Code	Status	Action
0x64	Response to the I/O Set Configuration	Continue reading
0x20	The I/O is written.	Retry with valid parameters.
0x60	Invalid parameters.	Reconfigure and retry.
0x68	Pin is not configured as a GPIO output.	

Reading an input pin:

- Send the GPIO_Read command through Endpoint 0x04 (OUT), with the following parameters:

Parameter	Value	Description
Command	0x68	Configure I/O pin
The I/O number	0x00	IO0
	0x10	IO1
	...	...
	0x50	IO5

- Read the response from Endpoint 0x84 (IN). The response can be one of the following:

Response Code	Status	Action
0x68	Response to the I/O set configuration	Continue reading
0x21 0x60 0x68	The I/O is read. Invalid parameters. The pin is not configured as a GPIO pin.	Retry with valid parameters. Reconfigure and retry.
0x00 0x01	The GPIO pin is at a logical LOW level. GPIO pin is logical high.	Logical state of the pin

8. Electrical Characteristics

8.1. Disclaimer

All typical values are measured at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$ unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage ranges unless otherwise specified.

The given typical values must be considered for design guidance only, and the variation around these values is expected.

8.2. Electrical Specification

Stresses beyond those listed in this chapter may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the operational chapters of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 8-1. Absolute Maximum Ratings

Parameter	Condition	Rating	Units
Ambient temperature under bias		-40 to +85	$^\circ\text{C}$
Storage temperature		-65 to +150	$^\circ\text{C}$
Voltage on Pins with Respect to GND			
On the V_{DD} pin		-0.3 to +5.5	V
On the V_{USB} pin		-0.3 to V_{DD}	V
On the V_{BUS} pin		-0.3 to V_{DD}	V
On the RESET pin		-0.3 to +5.5	V
On all other pins		-0.3 to ($V_{DD} + 0.3$)	V
Maximum Current			
On the GND pin ⁽¹⁾		350	mA
On the V_{DD} ⁽¹⁾ pin		350	mA
On any standard I/O pin		± 50	mA
Clamp current, I_K ($V_{PIN} < 0$ or $V_{PIN} > V_{DD}$)		± 20	mA
Total power dissipation ⁽²⁾		800	mW
Notes:			
1. The maximum current rating requires an even load distribution across the I/O pins. The device package's power dissipation characteristics may limit the maximum current rating. Refer to the <i>Thermal Specifications</i> section to calculate the device specifications.			
2. Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OI})$			

8.3. Standard Operating Conditions

For all other device characteristics to be valid, the device must operate within the ratings listed in this section.

Table 8-2. Standard Operating Conditions

Parameter		Ratings	Units
V_{DD} - Operating Supply Voltage⁽¹⁾			
Industrial temperature	V_{DD} (3.3V)	+3.135 to +3.465	V
	V_{DD} (5.0V)	+4.75 to +5.25	V
V_{DDUREG} - Operating Supply Voltage to use internal USB Regulator^(1,3)			

Table 8-2. Standard Operating Conditions (continued)

Parameter		Ratings	Units
Industrial temperature	V _{DDUREGMIN}	+3.9	V
	V _{DDUREGMAX}	+5.25	V
V_{USB} - Supply Voltage for the USB Transceiver^(1,3)			
Industrial temperature	V _{USB}	+3.3	V
V_{BUS} - Voltage for detecting whether the device is on USB Bus or not⁽¹⁾			
Industrial temperature	V _{BUSMIN}	+2.85	V
	V _{BUSMAX}	+3.15	V
f_{USB} - USB Input Clock Frequency			
Industrial temperature	f _{USBMIN}	47.88	MHz
	f _{USBMAX}	48.12	MHz
C_{VUSB} - External Capacitor Value on V_{USB}⁽²⁾			
Industrial temperature	C _{VUSB}	470	nF
T_A - Operating Ambient Temperature Range			
	T _{A_MIN}	-40	°C
	T _{A_MAX}	+85	°C
Notes:			
1. Refer to the Supply Voltage parameter in the <i>Supply Voltage</i> section.			
2. Use a ceramic or film capacitor to ensure a sufficiently low ESR between 0.5Ω and 4.0Ω.			
3. When 3.9V ≤ V _{DD} ≤ 5.25V, the internal USB regulator is ON, and the V _{USB} pin must be connected to an external capacitor (C _{VUSB}). When 3.135V ≤ V _{DD} < 3.9V, 3.3V must be provided externally on the V _{USB} pin.			

8.3.1. Supply Voltage

Table 8-3. Supply Voltage

Symbol		Min.	Typ. †	Max.	Units	Conditions
Supply Voltage						
V _{DD}	3.3V	3.135	3.3	3.465	V	
	5V	4.75	5	5.25	V	
Slew Rate		—	—	1.5	V/μs	2.85V ≤ V _{DD} ≤ 5.25V
V_{DD} Range for Operating the USB Voltage Regulator						
V _{DDUREG}		3.9	—	V _{DD_MAX}	V	V _{DDUREG} = V _{DD}
Supply Voltage for the USB Transceiver						
V _{USB}		—	3.3	—	V	
Voltage on V_{BUS} Pin						
V _{BUS}		2.85	3.0	3.15	V	
Quiescent Input Current of the USB Regulator						
I _Q		—	180	—	μA	USB regulator without load
Input Leakage on the Regulator Output (V_{USB} Pin)						
I _{VUSB}	Sink current	—	-10	—	mA	This applies to the fault condition when the USB regulator output voltage is driven externally at or above V _{USB_SINK}
	Source Current	—	30	—	mA	3.9 ≤ V _{DDU} ≤ V _{DD_MAX}
Sink Function Trigger Point						
V _{USB_SINK}		—	6	—	%	Percentage above V _{USB}
RAM Data Retention⁽¹⁾						
V _{DR}		2.85	—	—	V	Device in Power-Down mode

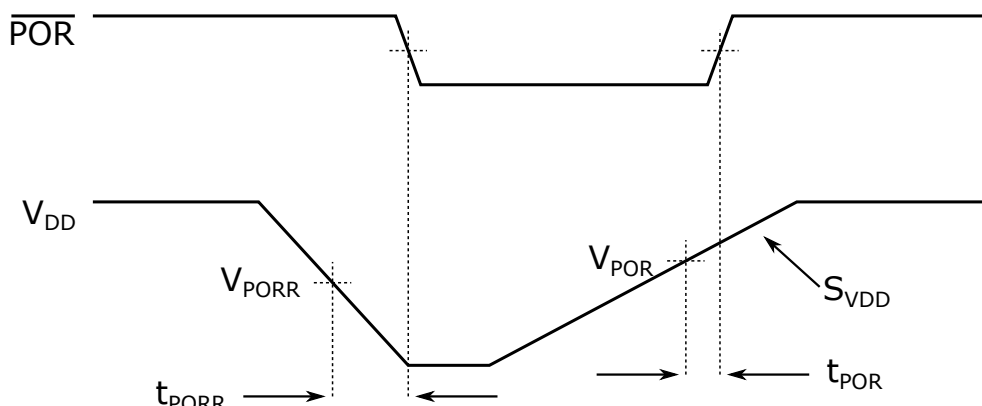
Table 8-3. Supply Voltage (continued)

Symbol	Min.	Typ. †	Max.	Units	Conditions
Power-on Reset Release Voltage					
V_{PORR}	2.7	2.85	3.05	V	
t_{PORR}	—	3	—	μ s	
Power-on Reset Re-Arm Voltage					
V_{PORR}	—	2.85	—	V	
t_{PORR}	—	2.7	—	μ s	
V_{DD} Rise Rate to Ensure Internal Power-on Reset Signal					
S_{VDD}	0.05	—	—	V/ms	

† Data in the “Typ.” column is specified at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$ unless otherwise noted. These parameters are not tested and are for design guidance only.

Note:

1. This is the limit to which V_{DD} can be lowered in a sleep mode without losing RAM data.

Figure 8-1. $\overline{\text{POR}}$ and $\overline{\text{PORR}}$ with Slow Rising V_{DD} 

Note: When $\overline{\text{POR}}$ is low, the device is held in Reset.

8.3.2. Power Consumption

Table 8-4. Device Power Consumption

Operating conditions: $V_{DD} = 3.3\text{V}$ $T_A = 25^\circ\text{C}$ System power consumption measured with peripherals disabled and I/O ports driven low, with inputs disabled					
Symbol	Description	Min.	Typ.†	Units	Conditions
I_{DD}	Active power consumption	—	4.4	mA	
$I_{DD_BASE (1)}$	Minimum power consumption	—	0.65	μ A	Power-Down or Standby mode, with all peripherals disabled. $V_{DD} = 3.3\text{V}$
I_{RST}	Reset power consumption	—	170	μ A	RESET line pulled low

Note:

1. Single Supply mode.

8.3.3. Reset Control

Table 8-5. Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-out Detector Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
t_{RST}^*	RESET pin pulse-width low to ensure a Reset	2.5	—	—	μs	
$R_{RST_UP}^*$	RESET pin pull-up resistor	—	35	—	$k\Omega$	
$T_{OST}^{(1)}$	Oscillator start-up timer period ⁽¹⁾	—	1024	—	cycles	

† Data in the “Typ.” column is specified at $T_A = 25^\circ C$ and $V_{DD} = 3.3V$ unless otherwise noted. These parameters are not tested and are provided for design guidance only.

* These parameters are characterized but not tested in production.

Note:

1. By design, the Oscillator Start-up Timer (T_{OST}) counts the first 1024 cycles, independent of frequency. To ensure these voltage tolerances, V_{DD} and AV_{DD} must be capacitively decoupled to GND as close to the device as possible. Refer to the *Connection for Power Supply* section of the *Hardware Guidelines* section for details.

8.3.4. Peripherals Power Consumption

Use the table below when calculating the additional current consumption for the different I/O peripherals in various operating modes. Some peripherals request the clock to be enabled when operating in Standby sleep mode. Refer to the specific peripheral section for further information.

Table 8-6. Peripherals Power Consumption⁽¹⁾

Operating conditions: $V_{DD} = 3.3V$ $T_A = 25^\circ C$ OSC at 24 MHz used as the clock source for the device in Standby sleep mode						
Symbol	Description	Min.	Typ.†	Max. 85°C	Unit	Conditions
I_{DD_OSC}	Internal Oscillator (OSC)	—	185	—	μA	OSC at 24 MHz
I_{DD_USB}	Universal Serial Bus (USB)	—	2.1	—	mA	$V_{DD}=5.0V$
I_{DD_UART}	Universal Synchronous and Asynchronous Receiver and Transmitter (USART)	—	8.2	—	μA	USART Enabled @9600 Baud
I_{DD_SPI}	Serial Peripheral Interface (SPI)	—	4	—	μA	SPI Host @100 kHz
I_{DD_I2C}	Inter-IC communication (I ² C)	—	8	—	μA	I ² C Host @100 kHz
$I_{DD_NVM_ERASE}$	Flash Programming Erase	—	6.8	—	mA	
$I_{DD_NVM_WRITE}$	Flash Programming Write	—	9.2	—	mA	
† Data in the “Typ.” column is specified at $T_A = 25^\circ C$ and $V_{DD} = 3.3V$, unless otherwise noted. These parameters are not tested and are provided for design guidance only.						
Note: 1. Current consumption of the module only. To calculate the total internal power consumption of the MCP2222, add the power consumption values of all peripherals and clock sources used to the base power consumption given in the <i>Power Consumption</i> section.						

8.3.5. I/O Pins

Table 8-7. General I/O Pins

Symbol	Description	Min.	Typ.†	Max.	Unit	Conditions
Input Low Voltage						

Table 8-7. General I/O Pins (continued)

Symbol	Description	Min.	Typ.†	Max.	Unit	Conditions
V_{IL}	I/O PINS:					
	IO5:IO0	—	—	$0.2 \times V_{DD}$	V	INLVL selecting $S_T^{(1)}$
	V_{BUS}	—	—	0.9	V	INLVL selecting $TTL^{(1)}$
	with I ² C levels	—	—	$0.3 \times V_{DD}$	V	
	RESET pin	—	—	$0.2 \times V_{DD}$	V	
Input High Voltage						
V_{IH}	I/O PINS					
	IO5:IO0	$0.8 \times V_{DD}$	—	—	V	INLVL selecting $S_T^{(1)}$
	V_{BUS}	1.2	—	—	V	INLVL selecting $TTL^{(1)}$
	with I ² C levels	$0.7 \times V_{DD}$	—	—	V	
	RESET pin	$0.8 \times V_{DD}$	—	—	V	
Input Leakage Current⁽¹⁾						
I_{IL}	I/O Pins	—	+/- 5	+/- 125	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85^\circ C$
	RESET pin ⁽²⁾	—	+/- 50	+/- 200	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85^\circ C$
Pull-up Current						
I_{PUR}		—	150	200	μA	$V_{DD} = 3.3V$, $V_{PIN} = GND$
Output Low Voltage						
V_{OL}	Standard I/O pins	—	—	0.6	V	$I_{OL} = 10\text{ mA}$, $V_{DD} = 3.3V$
Output High Voltage						
V_{OH}	Standard I/O pins	$V_{DD} - 0.7$	—	—	V	$I_{OH} = 6\text{ mA}$, $V_{DD} = 3.3V$
I/O Rise Time						
t_{SR}	Rising	—	22	—	ns	
	Falling	—	16	—	ns	
Pin Capacitance						
C_{IO}	I/O Pins	—	5	—	pF	

† Data in the “Typ.” column is specified at $T_A = 25^\circ C$ and $V_{DD} = 3.3V$, unless otherwise specified. These parameters are not tested and are provided for design guidance only.

Notes:

1. The negative current is defined as the current sourced by the pin.
2. The leakage current on the RESET pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

USB Pins

Table 8-8. USB Pin Specifications

Symbol	Description	Min.	Typ.†	Max.	Unit	Conditions
Input Leakage Current						
I_{ILUD}	Input leakage on USB data pins DM	—	+/- 5	+/- 125	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85^\circ C$
	Input leakage on USB data pins DP	—	+/- 5	+/- 125	nA	$GND \leq V_{PIN} \leq V_{DD}$, pin at high-impedance, $T_A = 85^\circ C$
I/O Rise Time						

Table 8-8. USB Pin Specifications (continued)

Symbol	Description	Min.	Typ.†	Max.	Unit	Conditions
	DP Rising	—	13	—	ns	
	DM Rising	—	13	—	ns	
	DP Falling	—	14	—	ns	
	DM Falling	—	14	—	ns	
Input Capacitance						
C _{UD}	Input capacitance on USB data pins DM	—	15	100	pF	Pin at high-impedance
	Input capacitance on USB data pins DP	—	15	100	pF	Pin at high-impedance

† Data in the “Typ.” column is specified at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$, unless otherwise specified. These parameters are not tested and are provided for design guidance only.

Notes:

1. The negative current is defined as the current sourced by the pin.
2. The leakage current on the RESET pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

8.3.6. Memory Programming Specifications**Table 8-9.** Memory Programming Specifications

Symbol	Description	Min.	Typ.†	Max.	Unit	Conditions
Data EEPROM Memory Specifications						
E _D *	Data EEPROM byte endurance	100k	—	—	Erase/Write cycles	$-40^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$
t _{D_RET}	Characteristic retention	—	40	—	Year	
V _{D_RW}	V _{DD} for Read or Erase/Write operation	V _{DDMIN}	—	V _{DDMAX}	V	
N _{D_REF} *	Total Erase/Write cycles before refresh ⁽¹⁾	1M	4M	—	Erase/Write cycles	$-40^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$
t _{D_CE}	Byte/Multi-byte/Full EEPROM Erasetime	—	10	11.7	ms	
t _{D_WRE}	Byte Write time	—	70	75	μs	
t _{D_BEW}	Byte Erase and Write time	—	10.07	—	ms	
Program Flash Memory Specifications						
E _P *	Flash memory cell endurance	1k	—	—	Erase/Write cycles	
t _{P_RET}	Characteristic retention	—	40	—	Year	
V _{P_RD}	V _{DD} for Read operation	V _{DDMIN}	—	V _{DDMAX}	V	
V _{P_REW}	V _{DD} for Erase/Write operation	2.85 ⁽¹⁾	—	V _{DDMAX}	V	
t _{P_CE}	Chip Erase time	—	11	11.6	ms	
t _{P_PE}	Page Erase time	—	10	11.7	ms	
t _{P_WRD}	Byte/Word Write time	—	70	75	μs	

† Data in the “Typ.” column is specified at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$, unless otherwise specified. These parameters are not tested and are provided for design guidance only.

* These parameters are characterized but not tested in production.

Note:

1. The number of times a single location can be erased or written before a full refresh (erase/write) of the EEPROM array is required.

8.3.7. Thermal Specifications

Table 8-10. Thermal Specifications

Symbol	Description	Typ.	Unit	Conditions
θ_{JA}	Thermal Resistance Junction to Ambient	75	°C/W	20-pin SSOP package (SS)
		53	°C/W	20-pin VQFN package (REB)
T_{JMAX}	Maximum Junction Temperature	Refer to the <i>Absolute Maximum Ratings</i> section.		
Notes:				
- Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$. - Internal Power Dissipation is calculated as follows: $P_{INTERNAL} = I_{DD} \times V_{DD}$, where I_{DD} is the current required to run the chip alone, without driving any load on the output pins. - Derated Power is calculated as follows: $P_{DER} = P_{D_{MAX}} (T_J - T_A) / \theta_{JA}$, where T_A is the ambient temperature and T_J is the junction temperature.				

8.3.8. Clock Control

Table 8-11. System Clock: Internal Oscillator Specifications⁽¹⁾

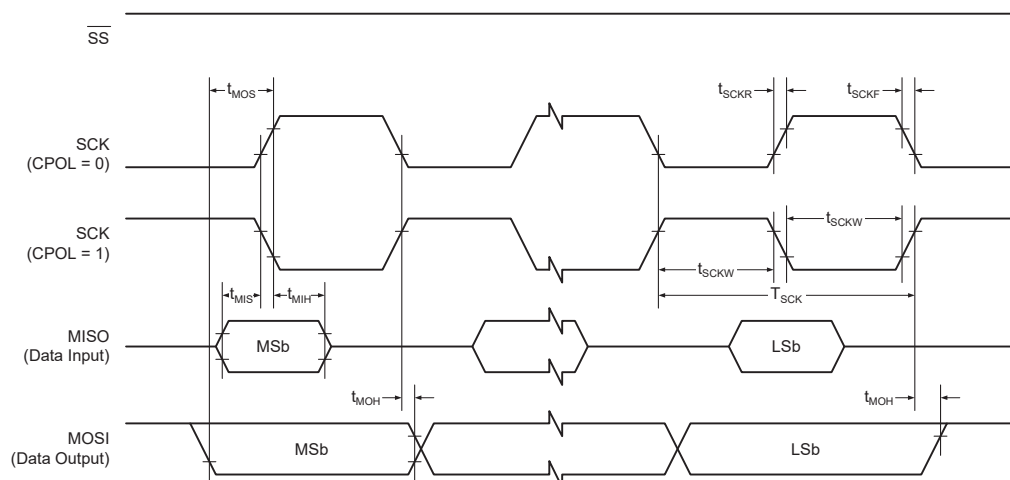
Symbol	Description	Min.	Typ. †	Max.	Unit	Conditions
f_{CLK}	Precision calibrated Clock frequency	—	24	—	MHz	
T_{OSC_ST}	OSC wake-up from Sleep start-up time	—	220	600	µs	Device in sleep mode,

† Data in the “Typ.” column is specified at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$ unless otherwise specified. These parameters are not tested and are provided for design guidance only.

Note:

- To ensure these oscillator frequency tolerances, V_{DD} and GND must be capacitively decoupled as close to the device as possible.

8.3.9. SPI

Figure 8-2. SPI - Timing Requirements

Table 8-12. SPI - Timing Characteristics

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
$f_{SCK}^{(1)}$	SCK clock frequency	—	—	$f_{CLK_PER}/2$	MHz	
$T_{SCK}^{(1)}$	SCK period	$2 \times T_{CLK_PER}$	—	—	ns	

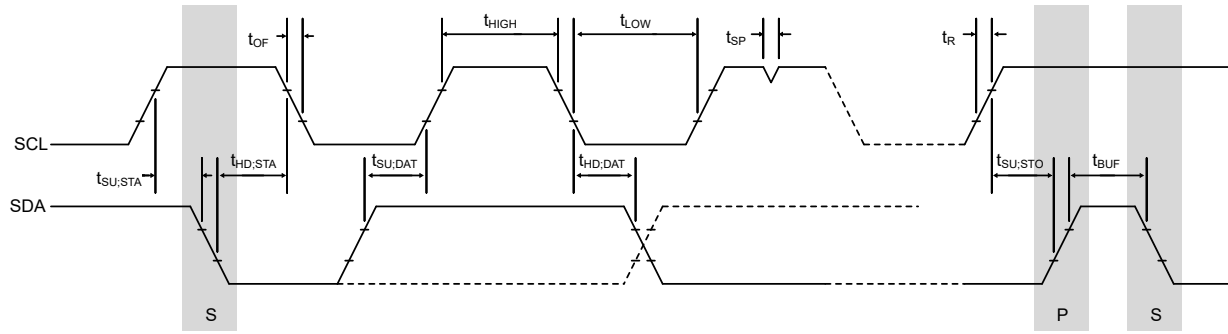
Table 8-12. SPI - Timing Characteristics (continued)

Symbol	Description	Min.	Typ. †	Max.	Units	Conditions
t_{SCKW}	SCK high/low width	—	$0.5 \times T_{SCK}$	—	ns	
t_{MOS}	MOSI valid before SCK	—	$0.5 \times T_{SCK}$	—	ns	
t_{MOH}	MOSI hold after SCK	—	$0.5 \times T_{SCK}$	—	ns	
t_{MIS}	MISO setup to SCK	—	T_{SCK}	—	ns	
t_{MIH}	MISO hold after SCK	—	0	—	ns	

† Data in the "Typ." column is specified at $T_A = 25^\circ\text{C}$ and $V_{DD} = 3.3\text{V}$ unless otherwise specified. These parameters are not tested and are provided for design guidance only.

Note:

- These parameters are characterized but not tested in production.

8.3.10. I²C**Figure 8-3.** I²C - Timing Requirements**Table 8-13.** I²C - Timing Specifications

Symbol	Description	Min.	Typ. †	Max.	Unit	Condition
f_{SCL}	SCL clock frequency	0	—	1000	kHz	Max. frequency requires system clock at 10 MHz
		0	—	400	kHz	For $V_{DD} < 2.2\text{V}$
V_{IH}	Input high voltage	$0.7 \times V_{DD}$	—	—	V	
V_{IL}	Input low voltage	—	—	$0.3 \times V_{DD}$	V	
V_{HYS}	Hysteresis of Schmitt Trigger inputs	$0.1 \times V_{DD}$	—	$0.4 \times V_{DD}$	V	
V_{OL}	Output low voltage	—	—	$0.2 \times V_{DD}$	V	$I_{load} = 20\text{ mA}$, Fast mode+
		—	—	$0.4 \times V_{DD}$		$I_{load} = 3\text{ mA}$, Normal mode, $V_{DD} > 2\text{V}$
		—	—	$0.2 \times V_{DD}$		$I_{load} = 3\text{ mA}$, $V_{DD} \leq 2\text{V}$
I_{OL}	Low level output current	3	—	—	mA	$f_{SCL} \leq 400\text{ kHz}$, $V_{OL} = 0.4\text{V}$
		20	—	—		$f_{SCL} \leq 1\text{ MHz}$, $V_{OL} = 0.4\text{V}$
C_B	Capacitive load for each bus line	—	—	400	pF	$f_{SCL} \leq 100\text{ kHz}$
		—	—	400	pF	$f_{SCL} \leq 400\text{ kHz}$
		—	—	550	pF	$f_{SCL} \leq 1\text{ MHz}$
t_R	Rise time for both SDA and SCL	—	—	1000	ns	$f_{SCL} \leq 100\text{ kHz}$
		20	—	300	ns	$f_{SCL} \leq 400\text{ kHz}$
		—	—	120	ns	$f_{SCL} \leq 1\text{ MHz}$

Table 8-13. I²C - Timing Specifications (continued)

Symbol	Description	Min.	Typ.†	Max.	Unit	Condition
t _{OF}	Output fall time from V _{IHMIN} to V _{ILMAX}	—	—	250	ns	f _{SCL} ≤ 100 kHz 10pF < C _B < 400 pF
		20×(V _{DD} /5.25V)	—	250	ns	f _{SCL} ≤ 400 kHz 10pF < C _B < 400 pF
		20×(V _{DD} /5.25V)	—	120	ns	f _{SCL} ≤ 1 MHz 10pF < C _B < 400 pF
t _{SP}	Spikes suppressed by the input filter	0	—	50	ns	
I _L	Input current for each I/O pin	—	—	1	μA	0.1×V _{DD} < V _I < 0.9×V _{DD}
C _I	Capacitance for each I/O pin	—	—	10	pF	
R _P	Value of pull-up resistor	(V _{DD} - V _{OL(max)}) / I _{OL}	—	1000 ns / (0.8473×C _B)	Ω	f _{SCL} ≤ 100 kHz
		—	—	300 ns / (0.8473×C _B)	Ω	f _{SCL} ≤ 400 kHz
		—	—	120 ns / (0.8473×C _B)	Ω	f _{SCL} ≤ 1 MHz
t _{HD_STA}	Hold time (repeated) Start condition	0.6	—	—	μs	f _{SCL} ≤ 400 kHz
		0.26	—	—	μs	f _{SCL} ≤ 1 MHz
		—	2.1	—	T _{SCL}	Start
		—	3.1	—	T _{SCL}	Repeated Start
T _{LOW}	Low period of SCL Clock	4.7	—	—	μs	f _{SCL} ≤ 100 kHz
		1.3	—	—	μs	f _{SCL} ≤ 400 MHz
		0.5	—	—	μs	f _{SCL} ≤ 1 MHz
T _{HIGH}	High period of SCL Clock	4.0	—	—	μs	f _{SCL} ≤ 100 kHz
		0.6	—	—	μs	f _{SCL} ≤ 400 kHz
		0.26	—	—	μs	f _{SCL} ≤ 1 MHz
t _{SU_STA}	Setup time for a repeated Start condition	4.7	—	—	μs	f _{SCL} ≤ 100 kHz
		0.6	—	—	μs	f _{SCL} ≤ 400 kHz
		0.26	—	—	μs	f _{SCL} ≤ 1 MHz
		—	3	—	T _{SCL}	
t _{HD_DAT}	Data hold time	—	0	—	ns	SDAHOLD[1:0]= 0x0
		30	—	300	ns	SDAHOLD[1:0]= 0x1
		120	—	420	ns	SDAHOLD[1:0]= 0x2
		300	—	900	ns	SDAHOLD[1:0]= 0x3
t _{SU_DAT}	Data setup time	250	—	—	ns	f _{SCL} ≤ 100 kHz
		100	—	—	ns	f _{SCL} ≤ 400 kHz
		50	—	—	ns	f _{SCL} ≤ 1 MHz
t _{SU_STO}	Setup time for Stop condition	4	—	—	μs	f _{SCL} ≤ 100 kHz
		0.6	—	—	μs	f _{SCL} ≤ 400 kHz
		0.26	—	—	μs	f _{SCL} ≤ 1 MHz
		—	2	—	T _{SCL}	

Table 8-13. I²C - Timing Specifications (continued)

Symbol	Description	Min.	Typ.†	Max.	Unit	Condition
t _{BUF}	Bus free time between a Stop and Start condition	4.7	—	—	μs	f _{SCL} ≤ 100 kHz
		1.3	—	—	μs	f _{SCL} ≤ 400 kHz
		0.5	—	—	μs	f _{SCL} ≤ 1 MHz
		—	2	—	T _{SCL}	

† Data in the "Typ." column is specified at T_A = 25°C and V_{DD} = 3.3V unless otherwise specified. These parameters are not tested and are provided for design guidance only.

Note:

1. I2CFm+ is supported only for values above 2.7V.

8.3.11. Characteristics Graphs

Characteristics Graphs are not available at this time.

9. Ordering Information

Available ordering options can be found by:

- Clicking on one of the following product page links:
- Searching by product name at [microchipdirect.com](https://www.microchipdirect.com)
- Contacting the Microchip local sales representative

Note: Automotive-grade ordering codes (VAO suffix) are set up on request and are not listed in the table below. Contact your local Microchip sales representative to request VAO ordering codes which are not shown on the respective product page.

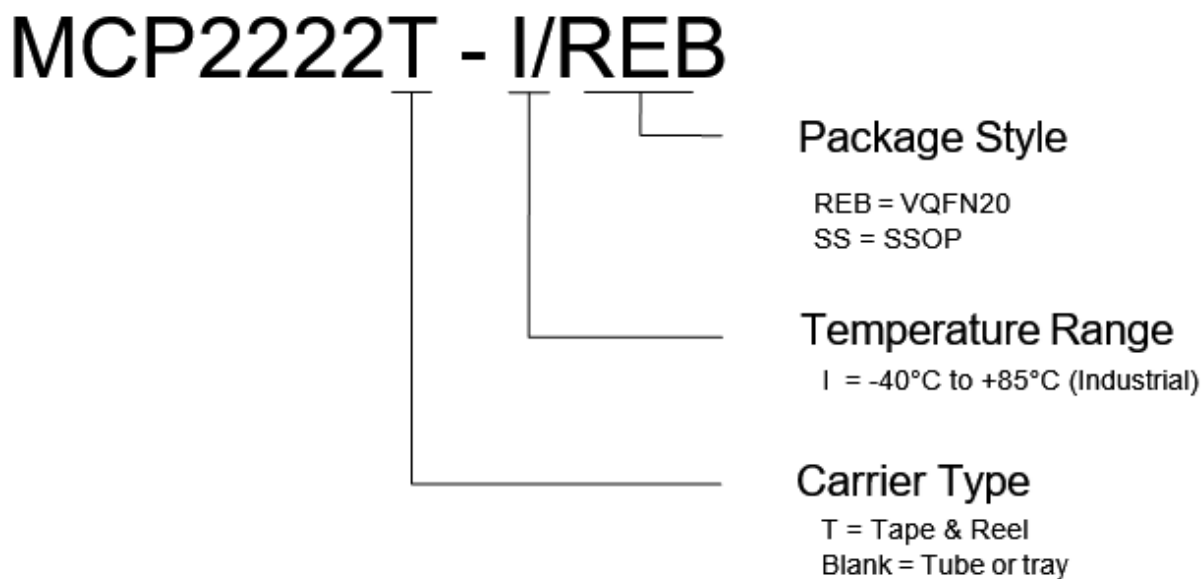
Table 9-1. Available Product Numbers

CPN	SRAM	EEPROM	Pin Count	Package	Temperature Range	Carrier Type
MCP2222T-I/REB	4 KB	256 bytes	20	VQFN	-40°C to +85°C	Tape & Reel
MCP2222-I/REB	4 KB	256 bytes	20	VQFN	-40°C to +85°C	Tube
MCP2222T-I/SS	4 KB	256 bytes	20	SSOP	-40°C to +85°C	Tape & Reel
MCP2222-I/SS	4 KB	256 bytes	20	SSOP	-40°C to +85°C	Tube

9.1. Product Identification System

To order or obtain information, such as pricing or delivery details, refer to the factory or the listed sales office.

Figure 9-1. MCP2222 Product Information



Note: The Tape & Reel identifier appears only in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with the Microchip Sales Office for package availability with the Tape & Reel option.

9.2. Package Drawings

9.2.1. Online Package Drawings

For the most recent package drawings:

1. Go to www.microchip.com/packaging.
2. Go to the package type-specific page, for example, VQFN.
3. Search for the drawing number and style to find the most recent package drawings.

Table 9-2. Package Drawings

Pin Count	Package Type	Drawing Number	Style
20	SSOP	C04-00072	SS
20	VQFN	C04-21380	REB

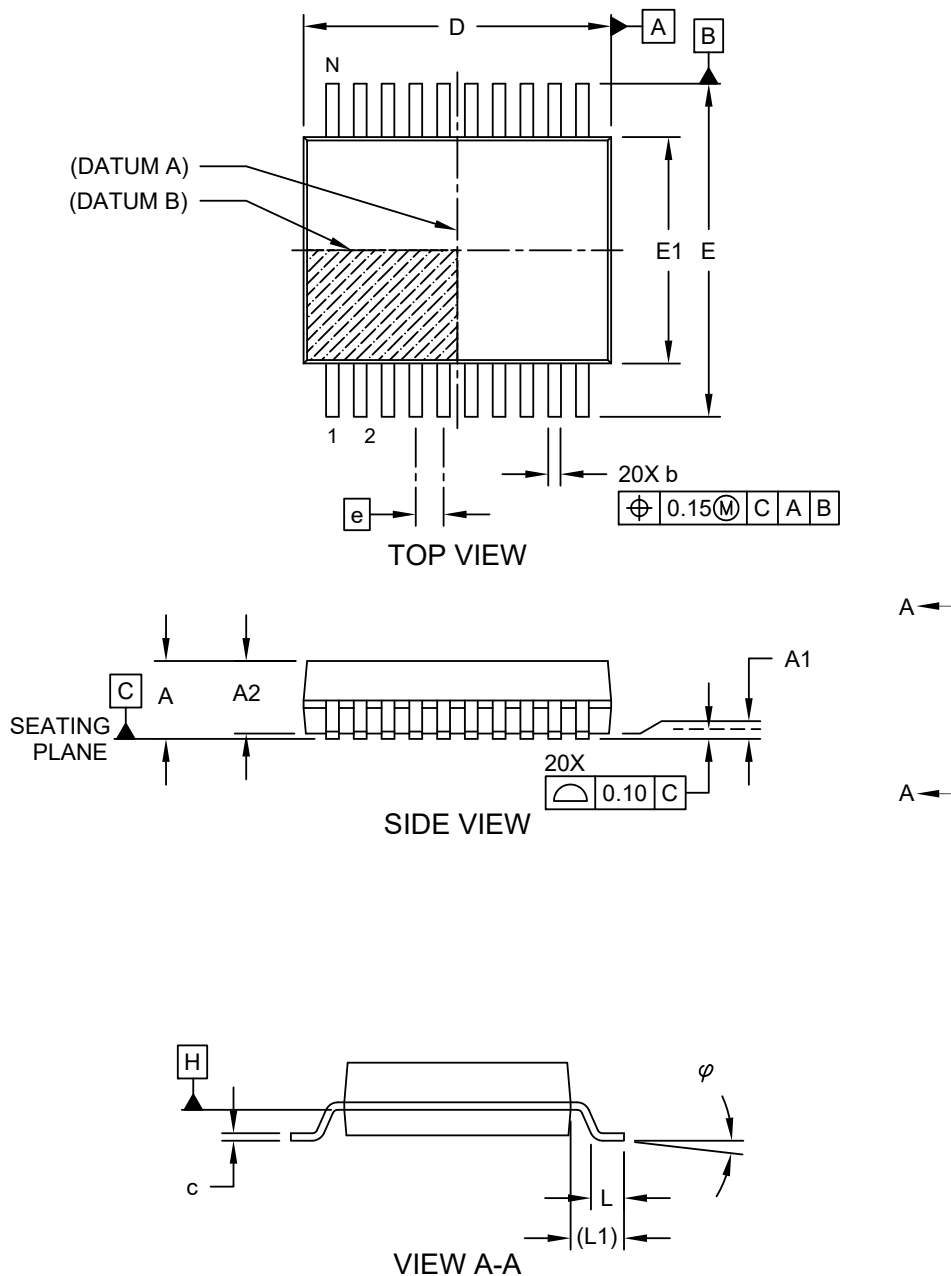
9.2.2. Package Marking Information

Note: If the full Microchip part number cannot be marked on one line, it will be carried over to the next line, which limits the number of available characters for customer-specific information.

9.2.2.1. 20-Pin SSOP

20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

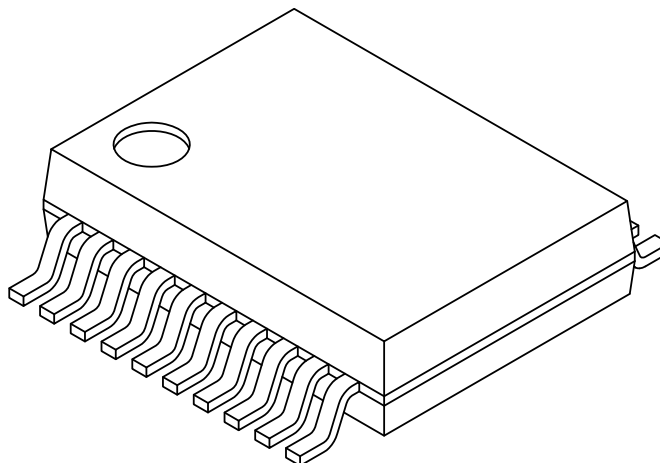
Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-072 Rev C Sheet 1 of 2

20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	20		
Pitch	e	0.65 BSC		
Overall Height	A	-	-	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	-	-
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	6.90	7.20	7.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1	1.25 REF		
Lead Thickness	c	0.09	-	0.25
Foot Angle	ϕ	0°	4°	8°
Lead Width	b	0.22	-	0.38

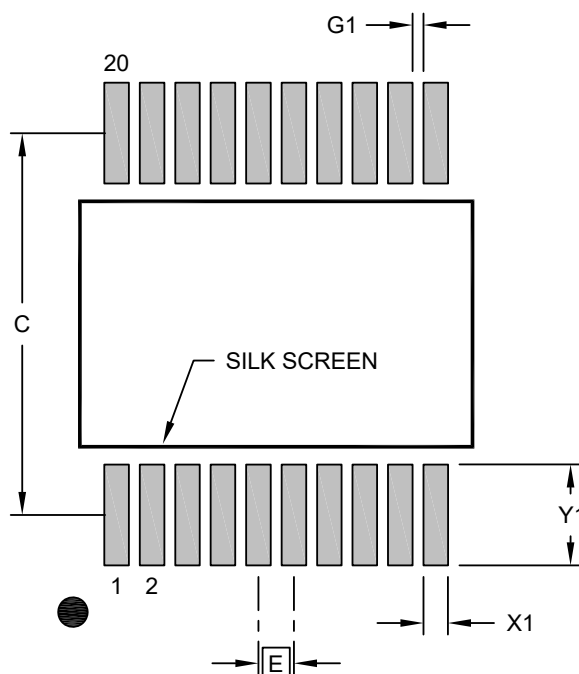
Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20mm per side.
- Dimensioning and tolerancing per ASME Y14.5M
 - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
 - REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072 Rev C Sheet 2 of 2

20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.00	
Contact Pad Width (X20)	X1			0.45
Contact Pad Length (X20)	Y1			1.85
Contact Pad to Center Pad (X18)	G1	0.20		

Notes:

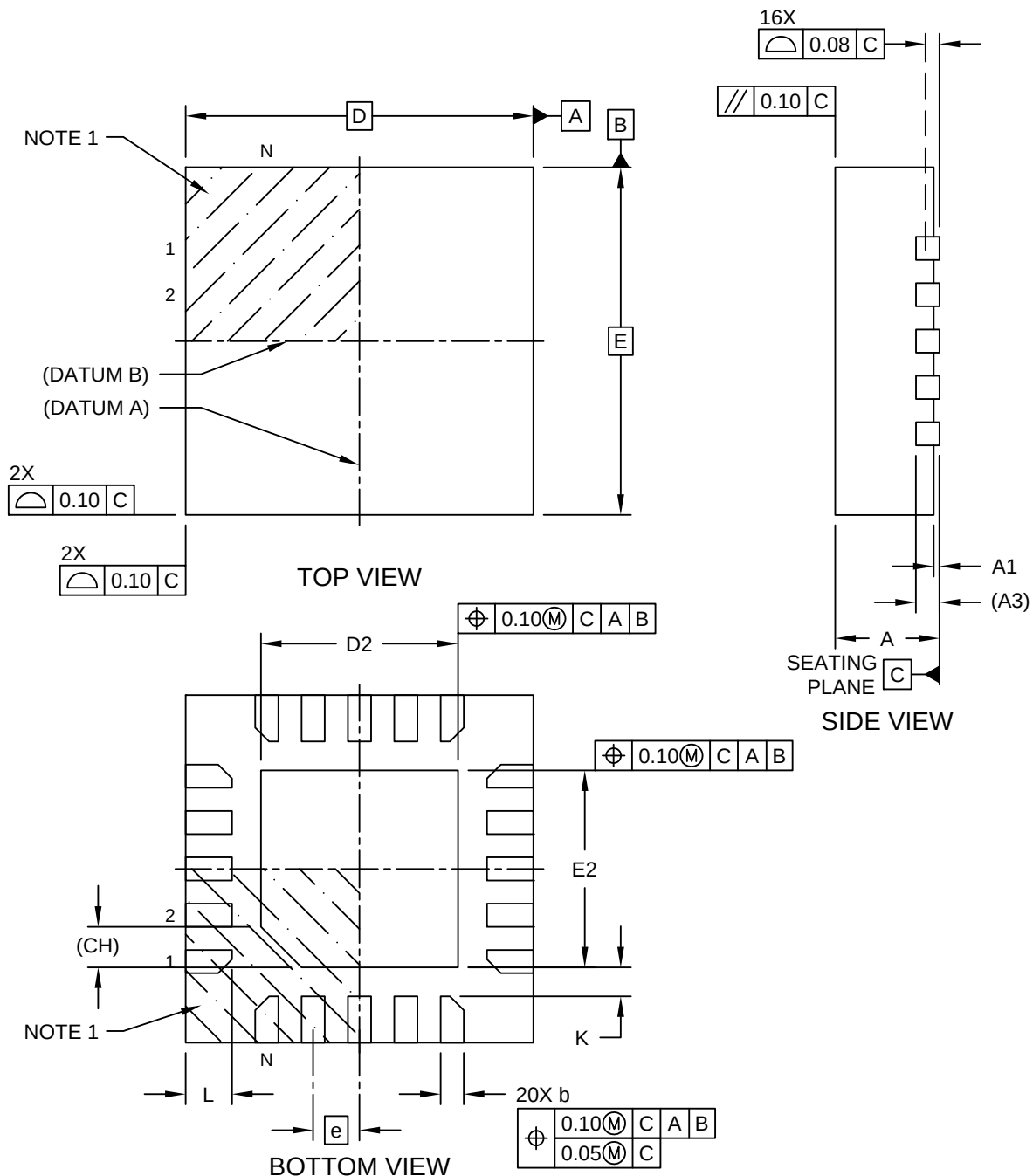
- Dimensioning and tolerancing per ASME Y14.5M
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
- For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2072 Rev C

9.2.2.2. 20-Pin VQFN

**20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN]
With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL**

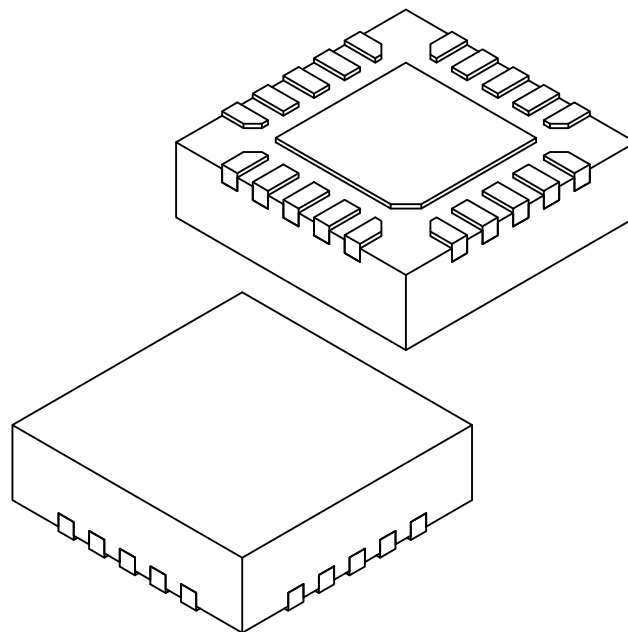
Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-21380 Rev A Sheet 1 of 2

20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Number of Terminals	N		20		
Pitch	e		0.40 BSC		
Overall Height	A		0.80	0.85	0.90
Standoff	A1		0.00	0.035	0.05
Terminal Thickness	A3		0.203 REF		
Overall Length	D		3.00 BSC		
Exposed Pad Length	D2		1.60	1.70	1.80
Overall Width	E		3.00 BSC		
Exposed Pad Width	E2		1.60	1.70	1.80
Terminal Width	b		0.15	0.20	0.25
Terminal Length	L		0.35	0.40	0.45
Terminal-to-Exposed-Pad	K		0.20	-	-
Pin 1 Index Chamfer	CH		0.35 REF		

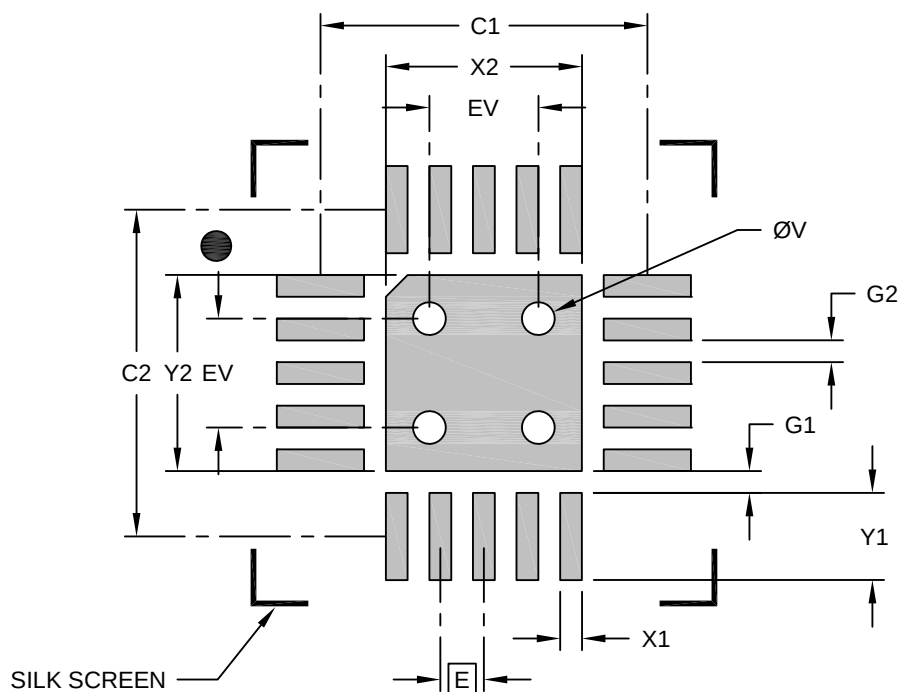
Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated
- Dimensioning and tolerancing per ASME Y14.5M
 - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
 - REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21380 Rev A Sheet 2 of 2

20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E	0.40 BSC		
Optional Center Pad Width	X2			1.80
Optional Center Pad Length	Y2			1.80
Contact Pad Spacing	C1		3.00	
Contact Pad Spacing	C2		3.00	
Contact Pad Width (X20)	X1			0.20
Contact Pad Length (X20)	Y1			0.80
Contact Pad to Center Pad (X20)	G1	0.20		
Contact Pad to Contact Pad (X16)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

- Dimensioning and tolerancing per ASME Y14.5M
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
- For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23380 Rev A

10. Data Sheet Revision History

Note: The data sheet revision is independent of the die revision and the device variant (the last letter of the ordering number).

Table 10-1. Revision History

Document Revision	Date	Comments
A	07/2026	Initial release of the document

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/legal-information/microchip-trademarks>.

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 products are 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.