Table of Contents

1. System Overview 7
2. Pin Definitions 9
2.1. QFN-32 Package Specifications 12
3. Electrical Characteristics 14
3.1. Absolute Maximum Specifications 14
3.2. Electrical Characteristics 15
4. System Development 17
4.1. System Development Process 17
4.2. Firmware Development 18
4.3. System Configurable Parameters 19
4.4. Programming Options 20
5. Typical Connection Diagrams 21
5.1. USB Connections and System Power 21
5.2. Touch Screen Interface 23
5.3. C2 Debug Interface 24
6. CIP-51 Microcontroller 25
6.1. Instruction Set 26
6.2. CIP-51 Register Descriptions 31
7. Memory Organization 35
7.1. Program Memory 36
8. Voltage Regulator and Voltage Supply Monitor 38
8.1. Voltage Regulator 38
8.2. Power-Fail Reset / VDD Monitor 38
8.3. Power-On Reset 38
8.4. External Reset 38
9. USB Interface 40
9.1. API Functionality 40
9.2. Device Descriptors and Customization 41
9.3. Enumeration Process 42
9.4. USB Bootloader 43
10. UART Interface 44
11. Serial Peripheral Interface (SPI) 46
11.1. Signal Descriptions 47
11.2. Serial Clock Phase and Polarity 48
11.3. SPI Timing 49
12. System-Management Bus (SMBus) Interface 51
12.1. SMBus Configuration 52
12.2. SMBus Operation 52
13. General Purpose Input / Output pins (GPIO) 54
Contact Information 56

List of Figures

Figure 1.1. CP2501 Block Diagram 8
Figure 2.1. CP2501 QFN-32 Pinout Diagram (Top View) 11
Figure 2.2. QFN-32 Package Drawing 12
Figure 2.3. QFN-32 Recommended PCB Land Pattern 13
Figure 5.1. USB Bus-Powered 21
Figure 5.2. USB Self-Powered with Regulator Enabled 21
Figure 5.3. USB Self-Powered with Regulator Disabled 22
Figure 5.4. UART Interface 23
Figure 5.5. SPI interface 23
Figure 5.6. SMBus Interface 23
Figure 5.7. GPIO Interface 24
Figure 5.8. C2 Program and Debug Interface 24
Figure 6.1. CIP-51 Block Diagram 25
Figure 7.1. CP2501 Memory Map 35
Figure 8.1. Power-On and VDD Monitor Reset Timing 39
Figure 9.1. USB Interface Block Diagram 40
Figure 10.1. UART System Block Diagram 44
Figure 10.2. 8-Bit UART Timing Diagram 45
Figure 11.1. SPI Block Diagram 47
Figure 11.2. Master Mode Data/Clock Timing 48
Figure 11.3. SPI Master Timing (Clock Phase = 0) 49
Figure 11.4. SPI Master Timing (Clock Phase = 1) 49
Figure 12.1. SMBus Block Diagram 51
Figure 12.2. Typical SMBus Configuration 52
Figure 12.3. SMBus Transaction 53

List of Tables

Table 2.1. Pin Definitions for the CP2501 9
Table 2.2. QFN-32 Package Dimensions 12
Table 2.3. QFN-32 PCB Land Pattern Dimensions 13
Table 3.1. Absolute Maximum Ratings 14
Table 3.2. Global Electrical Characteristics 15
Table 3.3. GPIO and VBUS DC Electrical Characteristics 15
Table 3.4. Reset Electrical Characteristics 16
Table 3.5. Internal Voltage Regulator Electrical Characteristics 16
Table 3.6. Internal High-Frequency Oscillator Electrical Characteristics 16
Table 6.1. CIP-51 Instruction Set Summary 27
Table 7.1. CP2501Memory Allocation 35
Table 9.1. Configurable USB Descriptors 41
Table 9.2. Default USB Bootloader Descriptors 41
Table 10.1. Data Formats and Baud Rates. 44
Table 11.1. PI Clock Speeds and Configurations 46
Table 11.2. Data centered on first or second edge of SCK period 46
Table 11.3. SPI Master Timing Parameters 50
Table 12.1. SMBus Clock Speed 51

List of Registers

SFR Definition 6.1. DPL: Data Pointer Low Byte 31
SFR Definition 6.2. DPH: Data Pointer High Byte 31
SFR Definition 6.3. SP: Stack Pointer 32
SFR Definition 6.4. ACC: Accumulator 32
SFR Definition 6.5. B: B Register 33
SFR Definition 6.6. PSW: Program Status Word 34

1. System Overview

The CP2501 is a programmable, 8051-based bridge device that adds a HID-compliant USB touch-screen interface to multi-touch devices such as digitizers, touch screens and pens. The CP2501 devices are pre-programmed with an easy-to-use firmware API, called the System Firmware, to perform USB communication and interface to touch modules using UART, SPI, or SMBus.

The development tools for the CP2501 include the CP2501 Configuration Wizard, which is a Windows tool that enables quick system development by creating a firmware project, including USB descriptors that match the capabilities of the multi-touch device. The tool also customizes the USB parameters such as the VID, PID and user strings.

The user programmable memory enables a custom interface to a wide range of multi-touch modules. Typical multi-touch modules use UART, SPI, or SMBus interfaces. The user firmware calls functions in the System Firmware to retrieve data from touch module and then sends the touch information to the USB host PC. The functions provided by the System Firmware make firmware development easy and remove the need to write low-level USB or interface firmware.

Highlighted features of the CP2501 are listed below.

- Support for up to 5 multi-touch points and 1 pen device
- High-speed pipelined 8051-compatible microcontroller core (up to 48 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- Precision calibrated 48 MHz or 24 MHz internal oscillator
- 64 kB of on-chip Flash memory
- 4 kB bytes of on-chip RAM
- SMBus/I^2C, UART, and SPI serial interfaces
- On-chip VDD supply monitor
- 16 general purpose I/O

With on-chip power-on reset, V_{DD} monitor, and clock oscillator, the CP2501 devices are truly stand-alone, system-on-a-chip solutions. The Flash memory devices are reprogrammable in-circuit, providing non-volatile data storage. The Flash memory devices also allow field upgrades of the 8051 firmware.

The CP2501 devices include Silicon Laboratories' 2-Wire C2 Debug and Programming interface, which allows for non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production device installed in the final application. This debug logic supports inspection of memory, viewing and modification of special function registers, setting breakpoints, single stepping, and run and halt commands. All features are fully functional while debugging using C2.

Each device is specified for 3.0 to 3.6 V operation over the industrial temperature range (–45 to +85 °C). The Port I/O and $\overline{RST}$ pins are tolerant of input signals up to 5 V. Figure 1.1 is the block diagram for the CP2501.

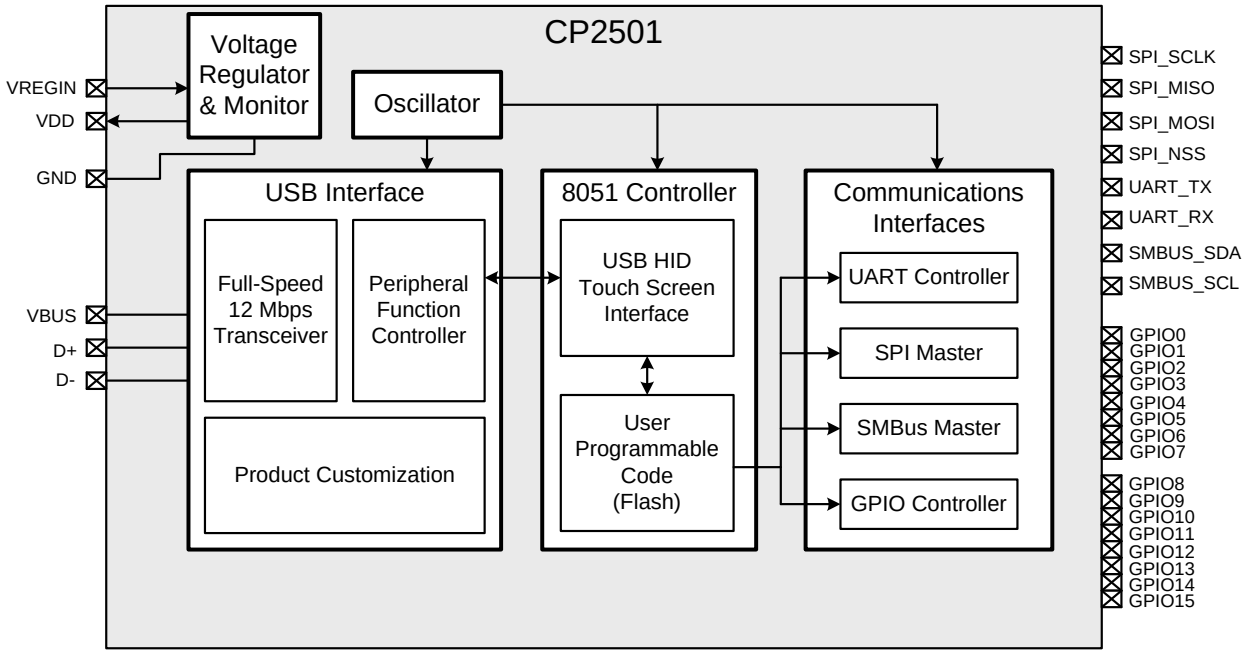


Figure 1.1. CP2501 Block Diagram

2. Pin Definitions

Table 2.1. Pin Definitions for the CP2501

Name	Pin	Type	Description
GND	3	Power	Ground. This ground connection is required. The center pad may optionally be connected to ground as well.
V_{DD}	6	Power	Power Supply Voltage. The output of the on-chip voltage regulator is available on this pin.
$\overline{RST}$ / C2CK	9	Digital I/O Digital I/O	Device Reset. Open-drain output of internal POR or V_{DD} monitor. An external source can initiate a system reset by driving this pin low. Clock signal for the C2 Debug Interface.
C2D	10	D I/O	Bi-directional data signal for the C2 Debug Interface.
VREGIN	7	Power	5 V Regulator Input. This pin is the input to the on-chip voltage regulator.
VBUS	8	Power	VBUS Sense Input. This pin should be connected to the VBUS signal of a USB network. A 5 V signal on this pin indicates a USB network connection.
D+	4	Digital I/O	USB D+.
D-	5	Digital I/O	USB D-.
SPI_SCLK	2*	Digital I/O	SPI Clock.
SPI_MISO	1*	Digital I/O	SPI Master In / Slave Out.
SPI_MOSI	32*	Digital I/O	SPI Master Out / Slave In.
SPI_NSS	31*	Digital I/O	SPI Slave Select.
UART_TX	30*	Digital I/O	UART Transmit.
UART_RX	29*	Digital I/O	UART Receive.
SMBUS_SDA	28*	Digital I/O	SMBus Data.
SMBUS_SCL	27*	Digital I/O	SMBus Clock.
GPIO0	26*	Digital I/O	General Purpose I/O 0.
GPIO1	25*	Digital I/O	General Purpose I/O 1.
GPIO2	24*	Digital I/O	General Purpose I/O 2.
GPIO3	23*	Digital I/O	General Purpose I/O 3.
GPIO4	22*	Digital I/O	General Purpose I/O 4.
Note: Pins can be left unconnected when not used.			

Table 2.1. Pin Definitions for the CP2501 (Continued)

Name	Pin	Type	Description
GPIO5	21*	Digital I/O	General Purpose I/O 5.
GPIO6	20*	Digital I/O	General Purpose I/O 6.
GPIO7	19*	Digital I/O	General Purpose I/O 7.
GPIO8	18*	Digital I/O	General Purpose I/O 8.
GPIO9	17*	Digital I/O	General Purpose I/O 9.
GPIO10	16*	Digital I/O	General Purpose I/O 10.
GPIO11	15*	Digital I/O	General Purpose I/O 11.
GPIO12	14*	Digital I/O	General Purpose I/O 12.
GPIO13	13*	Digital I/O	General Purpose I/O 13.
GPIO14	12*	Digital I/O	General Purpose I/O 14.
GPIO15	11*	Digital I/O	General Purpose I/O 15.
Note: Pins can be left unconnected when not used.			

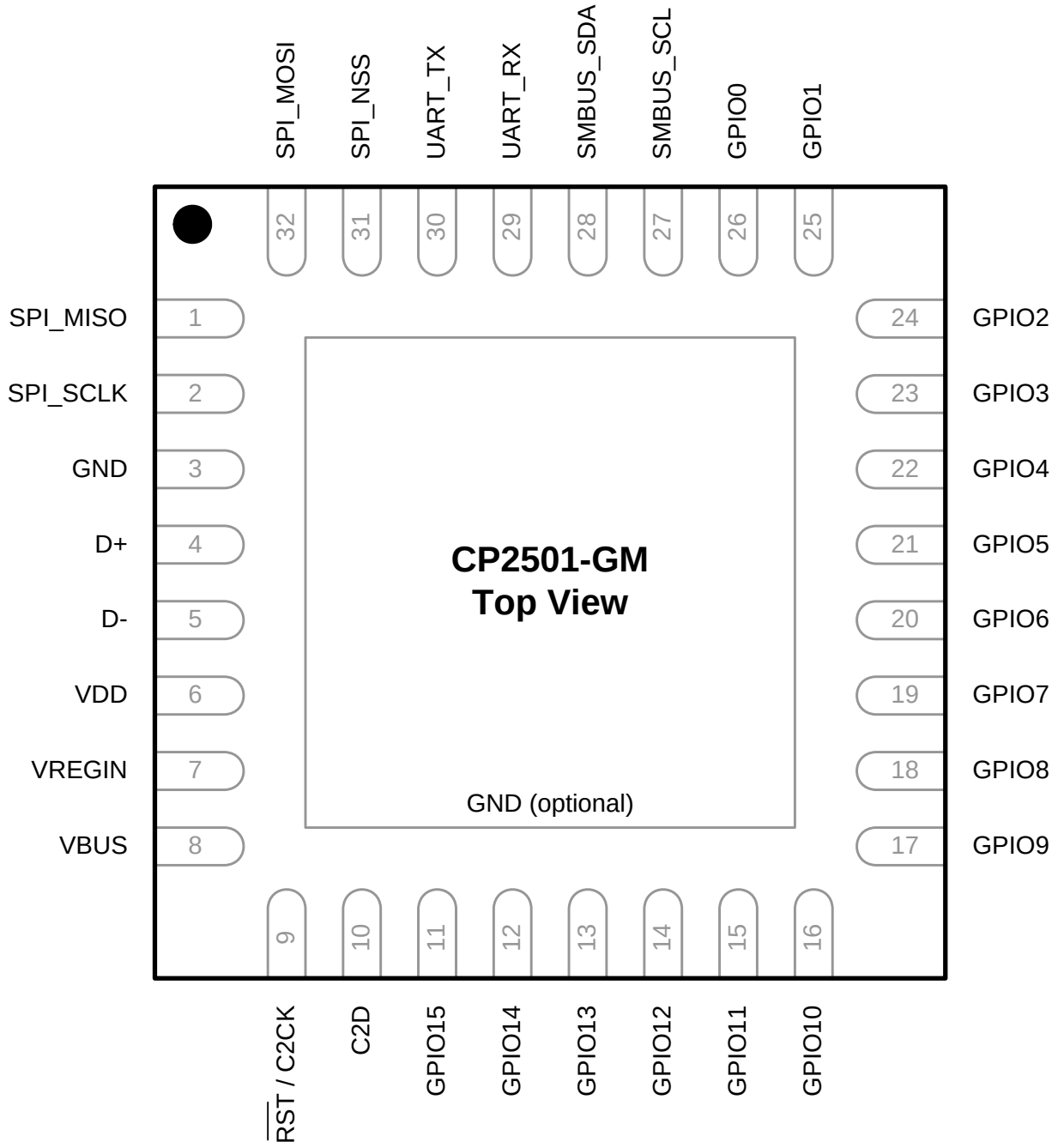


Figure 2.1. CP2501 QFN-32 Pinout Diagram (Top View)

2.1. QFN-32 Package Specifications

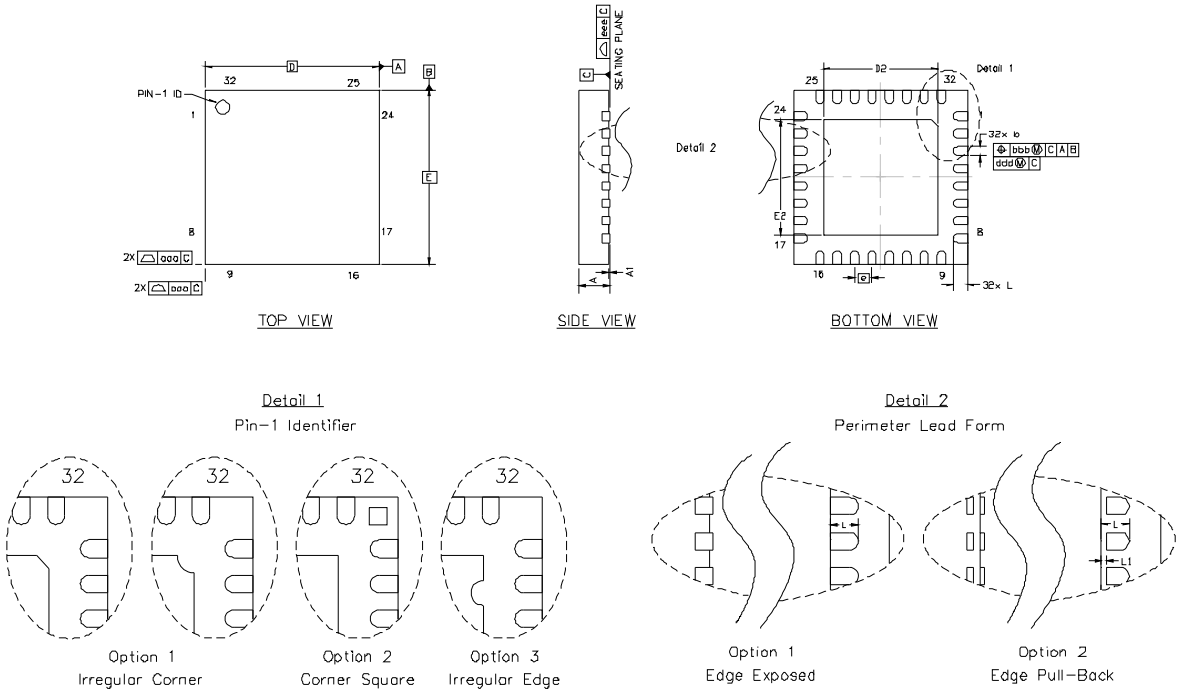


Figure 2.2. QFN-32 Package Drawing

Table 2.2. QFN-32 Package Dimensions

Dimension	Min	Typ	Max
A	0.80	0.90	1.00
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
D		5.00 BSC.	
D2	3.20	3.30	3.40
e		0.50 BSC.	
E		5.00 BSC.	

Dimension	Min	Typ	Max
E2	3.20	3.30	3.40
L	0.30	0.40	0.50
L1	0.00	—	0.15
aaa		0.15	
bbb		0.10	
ddd		0.05	
eee		0.08	

Notes:

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC Solid State Outline MO-220, variation VHHD except for custom features D2, E2, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

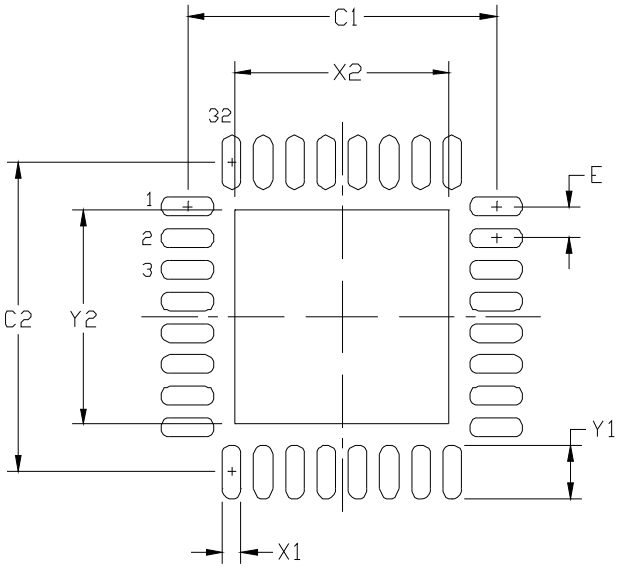


Figure 2.3. QFN-32 Recommended PCB Land Pattern

Table 2.3. QFN-32 PCB Land Pattern Dimensions

Dimension	Min	Max
C1	4.80	4.90
C2	4.80	4.90
E	0.50 BSC	
X1	0.20	0.30

Dimension	Min	Max
X2	3.20	3.40
Y1	0.75	0.85
Y2	3.20	3.40

Notes:

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This Land Pattern Design is based on the IPC-7351 guidelines.

Solder Mask Design

3. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60 μm minimum, all the way around the pad.

Stencil Design

4. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
5. The stencil thickness should be 0.125 mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pins.
7. A 3x3 array of 1.0 mm openings on a 1.2 mm pitch should be used for the center pad to assure the proper paste volume.

Card Assembly

8. A No-Clean, Type-3 solder paste is recommended.
9. The recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

3. Electrical Characteristics

3.1. Absolute Maximum Specifications

Table 3.1. Absolute Maximum Ratings

Parameter	Conditions	Min	Typ	Max	Units
Ambient temperature under bias		–55	—	125	°C
Storage Temperature		–65	—	150	°C
Voltage on $\overline{\text{RST}}$ or any Port I/O Pin with respect to GND		–0.3	—	5.8	V
Voltage on V_{DD} with respect to GND		–0.3	—	4.2	V
Maximum Total current through V_{DD} and GND		—	—	500	mA
Maximum output current sunk by $\overline{\text{RST}}$ or any Port pin		—	—	100	mA

Note: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

3.2. Electrical Characteristics

Table 3.2. Global Electrical Characteristics

–40 to +85 °C

Parameter	Conditions	Min	Typ	Max	Units
Digital Supply Voltage (V_{DD})		3.0	3.3	3.6	V
Supply Current [1]	Normal Operation—48 MHz	—	35	37	mA
	Normal Operation—24 MHz	—	23	25	mA
	Suspended	—	43	54	µA
Supply Current - USB Pull-up[2]			200	228	
Digital Supply RAM Data Retention Voltage		—	1.5	—	V
Specified Operating Temperature Range		–40	—	+85	°C

Notes:

1. If the device is connected to the USB bus, the USB Pull-up Current should be added to the Supply Current to calculate the total required current.
2. The USB Pull-up supply current values are calculated based on the USB specifications.

Table 3.3. GPIO and VBUS DC Electrical Characteristics

V_{DD} = 3.0 to 3.6 V, –40 to +85 °C unless otherwise specified.

Parameters	Conditions	Min	Typ	Max	Units
Output High Voltage	I_{OH} = –10 µA, Port I/O push-pull	V_{DD} – 0.1	—	—	V
	I_{OH} = –3 mA, Port I/O push-pull	V_{DD} - 0.7	—	—	V
	I_{OH} = –10 mA, Port I/O push-pull	—	V_{DD} - 0.8	—	V
Output Low Voltage	I_{OL} = 10 µA	—	—	0.1	V
	I_{OL} = 8.5 mA	—	—	0.6	V
	I_{OL} = 25 mA	—	1.0	—	V
Input High Voltage		2.0	—	—	V
Input Low Voltage		—	—	0.8	V
Input Leakage Current	Weak Pullup Off	–1	—	1	µA
	Weak Pullup On, V_{IN} = 0 V	—	25	50	µA
VBUS Detection Input High Voltage		3.0	—	—	V
VBUS Detection Input Low Voltage		—	—	1.0	V

Table 3.4. Reset Electrical Characteristics

V_{DD} = 1.8 to 3.6 V, –40 to +85 °C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
$\overline{RST}$ Output Low Voltage	I_{OL} = 8.5 mA, V_{DD} = 1.8 V to 3.6 V	—	—	0.6	V
$\overline{RST}$ Input High Voltage		0.7 x V_{DD}	—	—	V
$\overline{RST}$ Input Low Voltage		—	—	0.3 x V_{DD}	V
V_{DD} Monitor Threshold (V_{RST})		2.40	2.55	2.70	V
Minimum $\overline{RST}$ Low Time to Generate a System Reset		15	—	—	µs

Table 3.5. Internal Voltage Regulator Electrical Characteristics

V_{DD} = 3.0 to 3.6 V, –40 to +85 °C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Input Voltage Range[1]		3.0	—	5.25	V
Output Voltage Range		3.0	3.3	3.6	V
Output Current[2]		—	—	100	mA
Dropout Voltage (V_{DO})[3]		—	1	—	mV/mA

1. Input range specified for regulation. When an external regulator is used, VREGIN should be tied to VDD.
2. Output current is total regulator output, including any current required by the CP2501.
3. The minimum input voltage is 3.0 V or VDD + V_{DO} (max load), whichever is greater.

Table 3.6. Internal High-Frequency Oscillator Electrical Characteristics

V_{DD} = 3.0 to 3.6 V; –40 to +85 °C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Oscillator Frequency	48 MHz configuration	48 – 0.5%	48	48 + 0.5%	MHz
Oscillator Frequency	24 MHz configuration	24 – 0.5%	24	24 + 0.5%	MHz

4. System Development

The CP2501 family of products are programmable, 8051-based bridge devices that add a Windows-compatible, HID USB touch-screen interface to multi-touch screen devices such as digitizers, touch screens and pens. The CP2501 devices are pre-programmed with an easy-to-use firmware API, called the System Firmware, to perform USB communication and interface to touch modules using UART, SPI, or SMBus.

All multi-touch devices have unique capabilities and interface requirements. Custom user firmware is required to use the functions provided by the System Firmware to initialize the CP2501 and the touch screen. The user firmware also needs to process the data received from the touch screen before sending it to the USB host. Example firmware is available for each communications interface to show how use the API.

The following sections describe the user firmware development process, the development tools, the various configuration options for the CP2501 devices, and how to program the devices once firmware development is complete.

4.1. System Development Process

4.1.1. Relevant Documents and Development Tools

The following documents and development tools are necessary for developing user firmware for CP2501 devices:

- Silicon Labs Integrated Development Environment (IDE)
- 8051 C Compiler
- CP2501 Configuration Wizard
- Application Note 464: CP2501 Programmer's Guide and API Specification.

The document and tools are available with the CP2501 development kit and also available online at www.silabs.com.

4.1.2. Development Procedure Overview

The development process for the CP2501 includes these steps:

1. Determine the touch capabilities and interface specifications for the touch screen device.
2. Use the CP2501 Configuration Wizard to generate a firmware project based on those capabilities.
3. Develop user firmware on top of the base project using the Silicon Labs IDE and an 8051-toolset.
4. Test the project using the Silicon Labs CP2501 development kit hardware. The development kit hardware can interface directly to the touch screen module.

4.1.3. Development Tool Overview

The source code generated for the base firmware project by the CP2501 Configuration Wizard is written in C. An 8051 C compiler is required to build this project. The base firmware is written to be compatible with the following 8051 compilers:

- Raisonance
- Keil
- SDCC

The Silicon Labs IDE and CP2501 Configuration Wizard require Microsoft Windows XP or newer operating system.

4.2. Firmware Development

The user firmware performs three primary tasks:

1. CP2501 device configuration and initialization, including USB initialization.
2. Initialization of the touch screen, using one of the three communications interfaces.
3. Touch screen data retrieval, processing, and communication with the USB host.

The first component, CP2501 device configuration and initialization, is primarily handled by the firmware generated by the CP2501 Configuration Wizard. The system configurable parameters are described in Section 4.3.

The second and third components of the user firmware are customized for each system. The follow sections describe important aspects of the user firmware.

4.2.1. Interface API

All functions performed by the user firmware must use the Interface API, except for the functions that manage data returned from the touch screen. The CP2501 API Specification provides the full details regarding Interface API.

The capabilities of the communications interfaces and the GPIO are described in their respective sections:

Section “10. UART Interface” on page 44

Section “11. Serial Peripheral Interface (SPI)” on page 46

Section “12. System-Management Bus (SMBus) Interface” on page 51.

Section “13. General Purpose Input / Output pins (GPIO)” on page 54.

Example code that uses each of the three communications interfaces is available as part of the development kit. The example code is not targeted towards a specific touch screen interface, but it is intended to show how a typical system would use the communications interface.

4.2.2. Memory Organization

The pre-programmed System Firmware block reserves small portions of the various memory segments (Flash, RAM, and XRAM) for operation. The remaining parts of the memory segments are available for user firmware. The memory organization is detailed in Section “7. Memory Organization” on page 35.

The firmware project generated by the Configuration Wizard includes compiler and linker command line options to prevent user firmware from using the System Firmware’s reserved memory spaces. If these settings are overridden, user firmware must take care not to use any of the reserved memory spaces.

4.2.3. USB Data Format

The user firmware sends the touch screen data to the PC using the Interface API. The System Firmware directly copies the data from the user memory space to the USB FIFO. For the USB host to properly interpret the touch screen data, the data must be sent to host PC in a format that matches the specification defined by the USB descriptors. The Configuration Wizard generates a header file which includes a C struct. The C struct defines the order and the size of the elements for each of the attributes (usage) of each touch point. A separate struct is generated for touch point, pen, and mouse support.

4.2.4. Key Signature

When a CP2501 device is powered-on, code execution is initially handled by the System Firmware. The System Firmware transfers control to the user firmware if it determines that the key signature is present. Without a key signature, device control remains with the System Firmware. The base firmware project generated by the Configuration Wizard includes the key signature. The key signature is not intended as a data integrity signature, such as checksum or CRC, but is only used to indicate that user firmware has been programmed. The key signature should always be programmed after the user firmware.

4.3. System Configurable Parameters

The CP2501 devices include various configurable options. The configuration options for the USB, UART, SPI, SMBus and GPIO are detailed in their respective sections. All other options are detailed in the following sections.

4.3.1. System Clock

The 8051 system clock is configurable using the Interface API to 48 MHz or 24 MHz. The default system clock is 24 MHz. The faster system clock allows for quicker firmware execution, but requires a higher supply current. The faster clock is useful if the touch screen returns data at a high rate or if the data requires a large amount of processing before it is sent to the host PC.

The slower system clock speed executes the firmware at half the speed, but requires less supply current. If the touch screen data can be processed using the slower clock speed, the slower clock speed is recommended. The supply current requirements for the two clock speeds are available in Table 3.2, “Global Electrical Characteristics,” on page 15.

The system clock configuration should not be changed once the device has enumerated.

4.3.2. Voltage Regulator

The CP2501 devices include an on-chip 5 V to 3 V regulator. The regulator is turned on by default. If the CP2501 is powered using the 5 V source from the USB host (VBUS), the voltage regulator should use the default setting. If a 3 V source is available in the system to power the CP2501, the regulator should be turned off using the Interface API in order to reduce supply current. The supply current requirements for the voltage regulator are available in Table 3.5, “Internal Voltage Regulator Electrical Characteristics,” on page 16. The regulator is fully described in Section “8. Voltage Regulator and Voltage Supply Monitor” on page 38.

4.4. Programming Options

There are multiple options for programming CP2501 devices once user firmware development is complete.

- Silicon Labs factory programming
- Programming in-system over the C2 interface
- Programming in-system using the USB bootloader

Silicon Labs factory programming
Once firmware development is complete, the developer can submit the user firmware hex file to Silicon Labs. Silicon Labs will create a custom-part number which is available for order.

Programming in-system over the C2 Interface
If the final system includes external connections for the debug pins, the device is programmable in-system using the USB debug adapter that is provided with the development kit.

The final system can include the standard 10-pin debug header that is included on the development kits to directly interface with the USB debug adapter, or it can include a reduced size connection with only the debug pins and GND. A custom connection can be created between the USB debug adapter and the target system.

To program CP2501 devices in system, the C2CK/RST, C2D, and GND pins must be made available.

Programming in-system using the USB bootloader
The CP2501 user firmware is bootloadable over the USB interface using the Silicon Labs CP2501 Bootloader. The Silicon Labs CP2501 Bootloader is a Windows application that takes the hex file from the user generated firmware as an input.

5. Typical Connection Diagrams

The typical connection diagrams shown in this section partition the CP2501 device schematic into three components: USB & Power, touch screen interface, and debug connection.

5.1. USB Connections and System Power

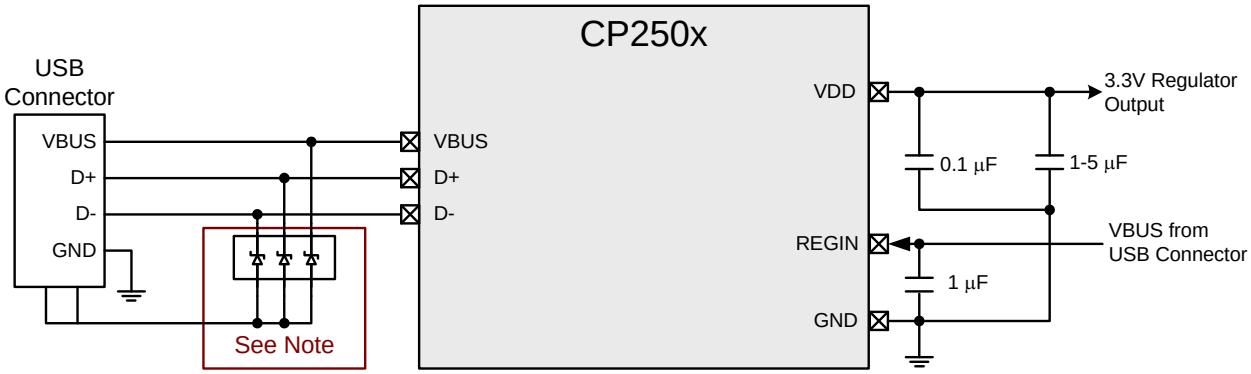


Figure 5.1. USB Bus-Powered

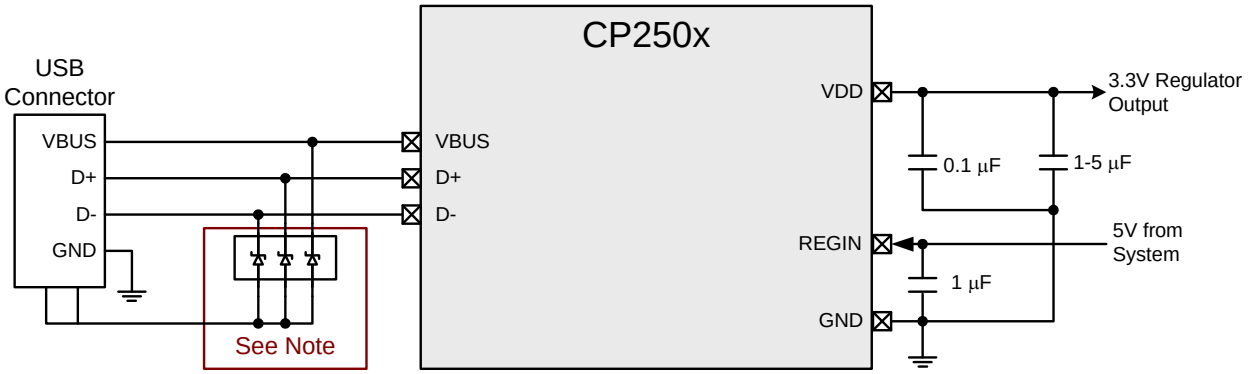


Figure 5.2. USB Self-Powered with Regulator Enabled

Note: Avalanche transient voltage suppression diodes compatible with full-speed USB should be added at the connector for ESD protection. Use Littelfuse p/n SP0503BAHT or equivalent.

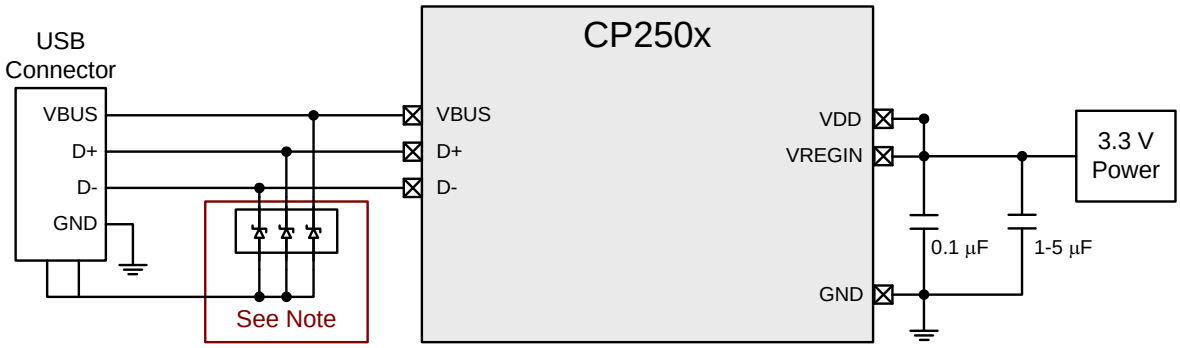


Figure 5.3. USB Self-Powered with Regulator Disabled

Note: Avalanche transient voltage suppression diodes compatible with full-speed USB should be added at the connector for ESD protection. Use Littelfuse p/n SP0503BAHT or equivalent.

5.2. Touch Screen Interface

The CP2501 devices support the use of only one of the three touch-screen interfaces at any time, and systems should only connect one of these interfaces.

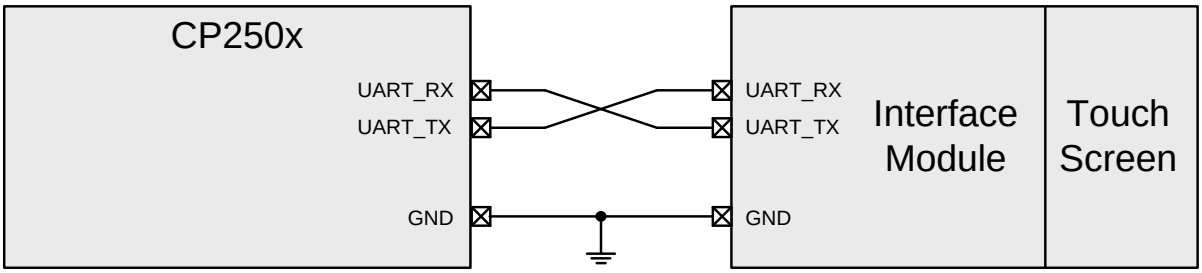


Figure 5.4. UART Interface

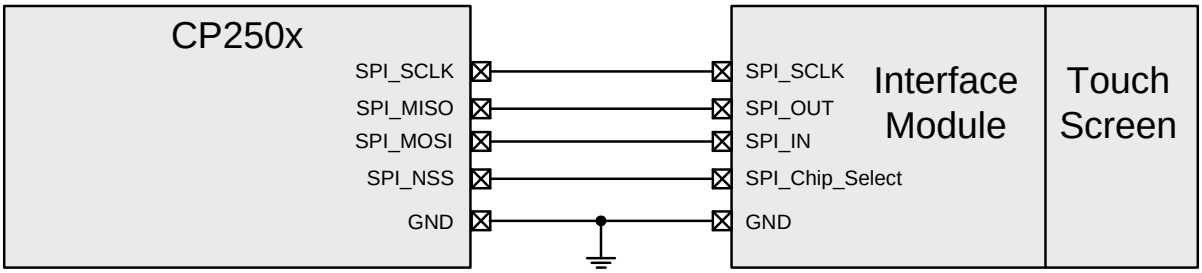


Figure 5.5. SPI interface

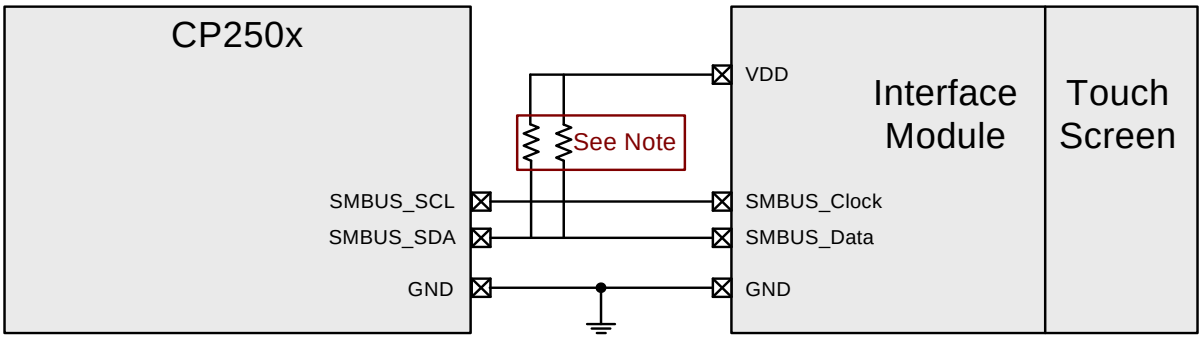


Figure 5.6. SMBus Interface

Note: External pull-up resistors are required if they are not already present on the bus. The pull-up resistors should be chosen according to the I^2C specification.

The GPIO pins are optional connections, that can be interfaced to the touch screen module or to other circuitry. Unused GPIO pins can be left unconnected. Figure 5.7 shows some example connections for the GPIO pins.

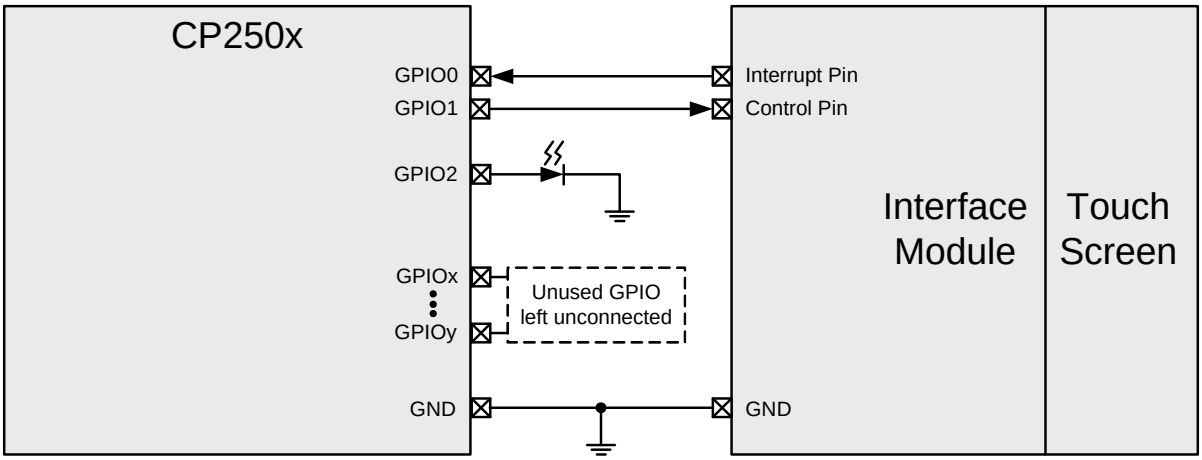


Figure 5.7. GPIO Interface

5.3. C2 Debug Interface

This debug connection is only necessary if the devices are programmed in-system. If the devices are programmed at the factory or elsewhere, external connections for a debug header are not necessary.

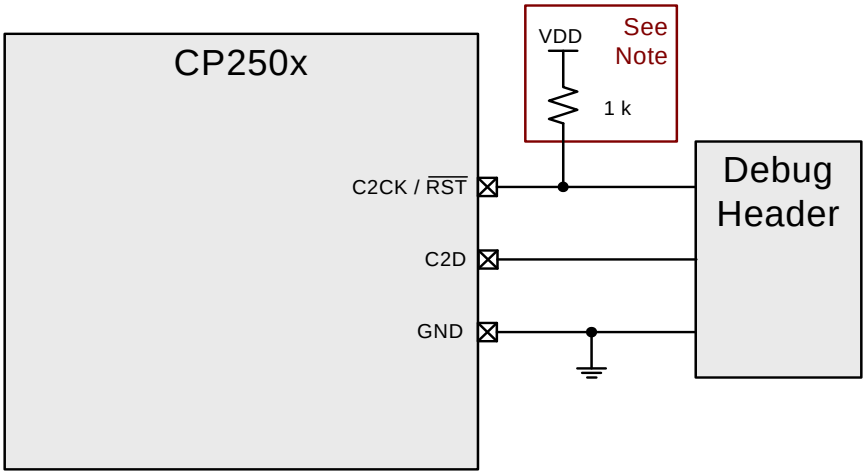


Figure 5.8. C2 Program and Debug Interface

Note: An external pull-up is not required, but is recommended for noise immunity. This pull-up is recommended even if a debug connection is not required.

6. CIP-51 Microcontroller

The system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The CIP-51 includes on-chip debug hardware, and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 6.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 48 MIPS Peak Throughput with 48 MHz Clock
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

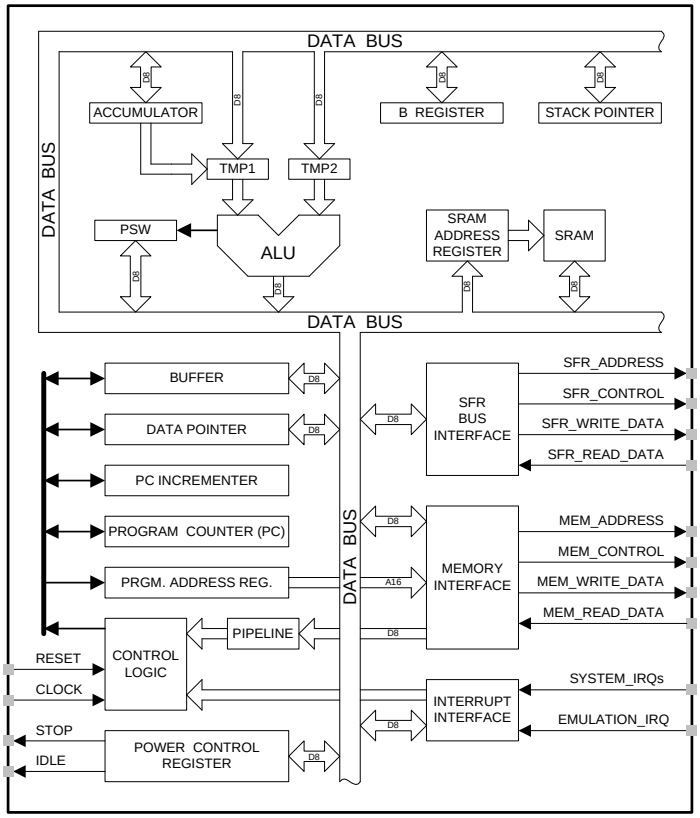


Figure 6.1. CIP-51 Block Diagram

With the CIP-51's maximum system clock at 48 MHz, it has a peak throughput of 48 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instructions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	7	3	1	2	1

6.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

6.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 6.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

Table 6.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2

Table 6.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	2
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2

Table 6.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
ANL C, bit	AND direct bit to Carry	2	2
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/4
JNC rel	Jump if Carry is not set	2	2/4
JB bit, rel	Jump if direct bit is set	3	3/5
JNB bit, rel	Jump if direct bit is not set	3	3/5
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/5
Program Branching			
ACALL addr11	Absolute subroutine call	2	4
LCALL addr16	Long subroutine call	3	5
RET	Return from subroutine	1	6
RETI	Return from interrupt	1	6
AJMP addr11	Absolute jump	2	4
LJMP addr16	Long jump	3	5
SJMP rel	Short jump (relative address)	2	4
JMP @A+DPTR	Jump indirect relative to DPTR	1	4
JZ rel	Jump if A equals zero	2	2/4
JNZ rel	Jump if A does not equal zero	2	2/4
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/5
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/5
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/5
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/6
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/4
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/5
NOP	No operation	1	1

Notes on Registers, Operands and Addressing Modes:
Rn—Register R0–R7 of the currently selected register bank.
@Ri—Data RAM location addressed indirectly through R0 or R1.
rel—8-bit, signed (twos complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.
direct—8-bit internal data location's address. This could be a direct-access Data RAM location (0x00–0x7F) or an SFR (0x80–0xFF).
#data—8-bit constant
#data16—16-bit constant
bit—Direct-accessed bit in Data RAM or SFR
addr11—11-bit destination address used by ACALL and AJMP. The destination must be within the same 2 kB page of program memory as the first byte of the following instruction.
addr16—16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8 kB program memory space.
There is one unused opcode (0xA5) that performs the same function as NOP. All mnemonics copyrighted © Intel Corporation 1980.

6.2. CIP-51 Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should always be written to the value indicated in the SFR description. Future product versions may use these bits to implement new features in which case the reset value of the bit will be the indicated value, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the data sheet associated with their corresponding system function.

SFR Definition 6.1. DPL: Data Pointer Low Byte

Bit	7	6	5	4	3	2	1	0
Name	DPL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x82

Bit	Name	Function
7:0	DPL[7:0]	**Data Pointer Low.** The DPL register is the low byte of the 16-bit DPTR.

SFR Definition 6.2. DPH: Data Pointer High Byte

Bit	7	6	5	4	3	2	1	0
Name	DPH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x83

Bit	Name	Function
7:0	DPH[7:0]	**Data Pointer High.** The DPH register is the high byte of the 16-bit DPTR.

SFR Definition 6.3. SP: Stack Pointer

Bit	7	6	5	4	3	2	1	0
Name	SP[7:0]							
Type	R/W							
Reset	0	0	0	0	0	1	1	1

SFR Address = 0x81

Bit	Name	Function
7:0	SP[7:0]	**Stack Pointer.** The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset.

SFR Definition 6.4. ACC: Accumulator

Bit	7	6	5	4	3	2	1	0
Name	ACC[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE0; Bit-Addressable

Bit	Name	Function
7:0	ACC[7:0]	**Accumulator.** This register is the accumulator for arithmetic operations.

SFR Definition 6.5. B: B Register

Bit	7	6	5	4	3	2	1	0
Name	B[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF0; Bit-Addressable

Bit	Name	Function
7:0	B[7:0]	**B Register.** This register serves as a second accumulator for certain arithmetic operations.

SFR Definition 6.6. PSW: Program Status Word

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS[1:0]		OV	F1	PARITY
Type	R/W	R/W	R/W	R/W		R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD0; Bit-Addressable

Bit	Name	Function
7	CY	**Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.
6	AC	**Auxiliary Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.
5	F0	**User Flag 0.** This is a bit-addressable, general purpose flag for use under software control.
4:3	RS[1:0]	**Register Bank Select.** These bits select which register bank is used during register accesses. 00: Bank 0, Addresses 0x00-0x07 01: Bank 1, Addresses 0x08-0x0F 10: Bank 2, Addresses 0x10-0x17 11: Bank 3, Addresses 0x18-0x1F
2	OV	**Overflow Flag.** This bit is set to 1 under the following circumstances: • An ADD, ADDC, or SUBB instruction causes a sign-change overflow. • A MUL instruction results in an overflow (result is greater than 255). • A DIV instruction causes a divide-by-zero condition. The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases.
1	F1	**User Flag 1.** This is a bit-addressable, general purpose flag for use under software control.
0	PARITY	**Parity Flag.** This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even.

7. Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The memory organization of the CP2501 device family is shown in Figure 7.1

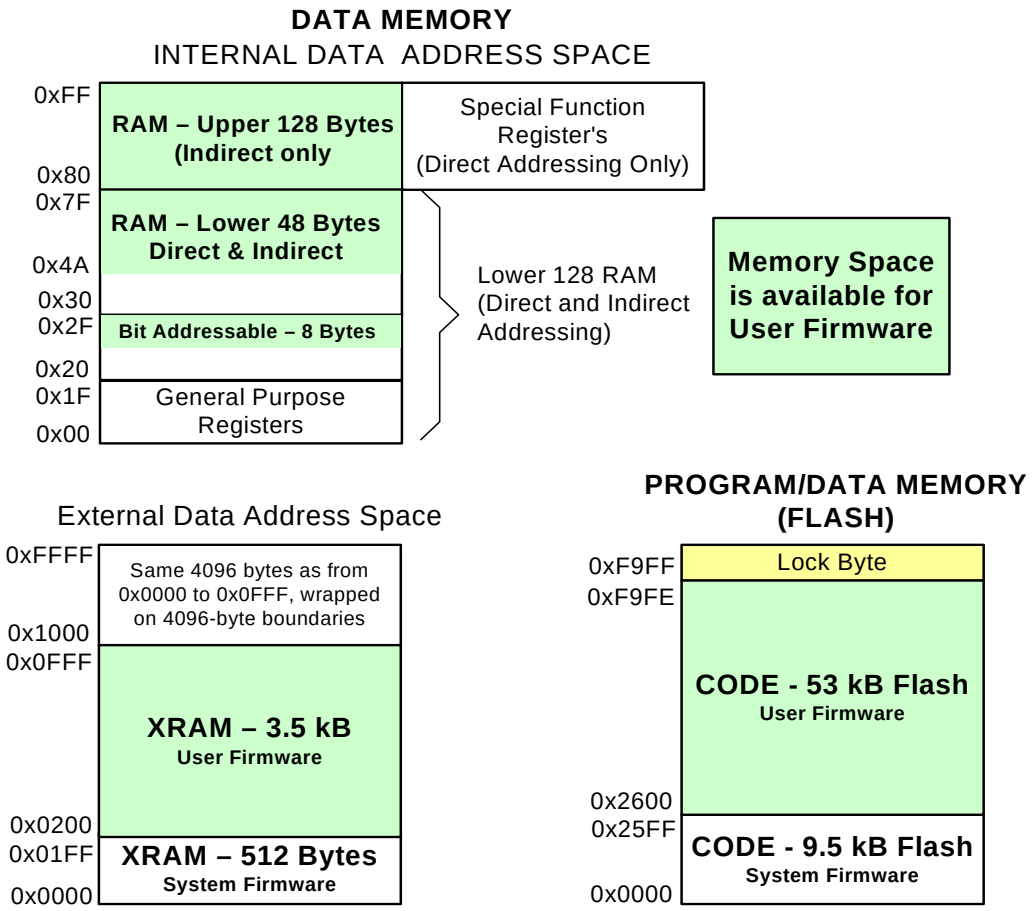


Figure 7.1. CP2501 Memory Map

The System Firmware requires small portions of the Flash, RAM, and XRAM and the unused memory is available for user firmware. Table 7.1 indicates how much of each memory space is available for user firmware.

Table 7.1. CP2501Memory Allocation

Memory Space	Full Address Range	In Use By System Firmware	User Available Range	User Available Size (Bytes)
Code	0x0000– 0xF9FF	0x0000–0x25FF	0x2600–0xF9FF	54272
Internal RAM	0x00–0xFF	0x00–0x27 0x30–0x49	0x28–0x2F 0x4A–0xFF	190
External RAM (XRAM)	0x0000–0x0FFF	0x0000–0x01FF	0x0200–0x0FFF	3584

7.1. Program Memory

The CIP-51 core has 64 kB program memory space and the CP2501 implements all 53 kB of this program memory space as in-system, re-programmable Flash memory. The address range of Flash available for user firmware is provided in Table 7.1. Addresses outside of this range

7.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the CP2501 devices, the MOVX instruction is normally used to read and write on-chip XRAM.

7.1.2. Data Memory

The CP2501 devices family contain a total of 4352 bytes of RAM data memory. On the CP2501 devices, 256 bytes of this memory is mapped into the internal RAM space of the 8051. The other 4096 bytes memory is the on-chip “external” memory, or XRAM. The data memory map is shown in Figure 7.1 for reference.

7.1.3. Internal RAM

There are 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 7.1 illustrates the data memory organization of the CP2501.

7.1.3.1. General Purpose Registers

The lower 32 bytes of data memory, locations 0x00 through 0x1F, may be addressed as four banks of general-purpose registers. Each bank consists of eight byte-wide registers designated R0 through R7. Only one of these banks may be enabled at a time. Two bits in the program status word, RS0 (PSW.3) and RS1 (PSW.4), select the active register bank (see description of the PSW in SFR Definition 6.6). This allows fast context switching when entering subroutines. Indirect addressing modes use registers R0 and R1 as index registers. The System Firmware uses all RAM locations from 0x08 through 0x1F and so the register banks are not available to user firmware.

7.1.3.2. Bit Addressable Locations

In addition to direct access to data memory organized as bytes, the sixteen data memory locations at 0x20 through 0x2F are also accessible as 128 individually addressable bits. Each bit has a bit address from 0x00 to 0x7F. Bit 0 of the byte at 0x20 has bit address 0x00 while bit7 of the byte at 0x20 has bit address 0x07. Bit 7 of the byte at 0x2F has bit address 0x7F. A bit access is distinguished from a full byte access by the type of instruction used (bit source or destination operands as opposed to a byte source or destination). See Table 7.1 for the user firmware accessible range of bit-addressable memory.

The MCS-51™ assembly language allows an alternate notation for bit addressing of the form XX.B where XX is the byte address and B is the bit position within the byte. For example, the instruction:

```
MOV    C, 22.3h
```

moves the Boolean value at 0x13 (bit 3 of the byte at location 0x22) into the Carry flag.

7.1.3.3. Stack

A programmer's stack can be located anywhere in the 256-byte data memory. The stack area is designated using the Stack Pointer (SP) SFR. The SP will point to the last location used. The next value pushed on the stack is placed at SP+1 and then SP is incremented. Upon entry into the user firmware from System Firmware, the stack pointer is initialized to location 0x4A. Therefore, the first value pushed on the stack is placed at location 0x4B, which is also the first register (R0) of register bank 1. Thus, if more than one register bank is to be used, the SP should be initialized to a location in the data memory not being used for data storage. The stack depth can extend up to 256 bytes.

8. Voltage Regulator and Voltage Supply Monitor

8.1. Voltage Regulator

CP2501 devices include an on-chip 5 V to 3.3 V regulator. This allows the devices to be configured as either a USB bus-powered device or a USB self-powered device. The connections diagrams for the various options are shown in Section “5. Typical Connection Diagrams” on page 21. When enabled, the voltage regulator output appears on the V_{DD} pin and can be used to power external devices. Electrical characteristics for the on-chip regulator are specified in Table 3.5 on page 16

The USB max power and power attributes descriptor must match the device power usage and configuration See Application Note “AN465: CP250x Configuration Wizard Guide” for more instructions on how to customize these parameters.

8.2. Power-Fail Reset / V_{DD} Monitor

When a power-down transition or power irregularity causes V_{DD} to drop below V_{RST}, the power supply monitor will drive the $\overline{RST}$ pin low and hold the CP2501 device in a reset state (see Figure 8.1). When V_{DD} returns to a level above V_{RST}, the device will be released from the reset state. The V_{DD} monitor is automatically enabled after power-on resets.

8.3. Power-On Reset

During power-up, the device is held in a reset state and the $\overline{RST}$ pin is driven low until V_{DD} settles above V_{RST}. A delay occurs before the device is released from reset; the delay decreases as the V_{DD} ramp time increases (V_{DD} ramp time is defined as how fast V_{DD} ramps from 0 V to V_{RST}). Figure 8.1. plots the power-on and V_{DD} monitor event timing. The maximum V_{DD} ramp time is 1 ms; slower ramp times may cause the device to be released from reset before V_{DD} reaches the V_{RST} level. For ramp times less than 1 ms, the power-on reset delay ($T_{PORDelay}$) is typically less than 0.3 ms.

8.4. External Reset

The external $\overline{RST}$ pin provides a means for external circuitry to force the device into a reset state. Asserting an active-low signal on the $\overline{RST}$ pin generates a reset; an external pullup and/or decoupling of the $\overline{RST}$ pin may be necessary to avoid erroneous noise-induced resets.

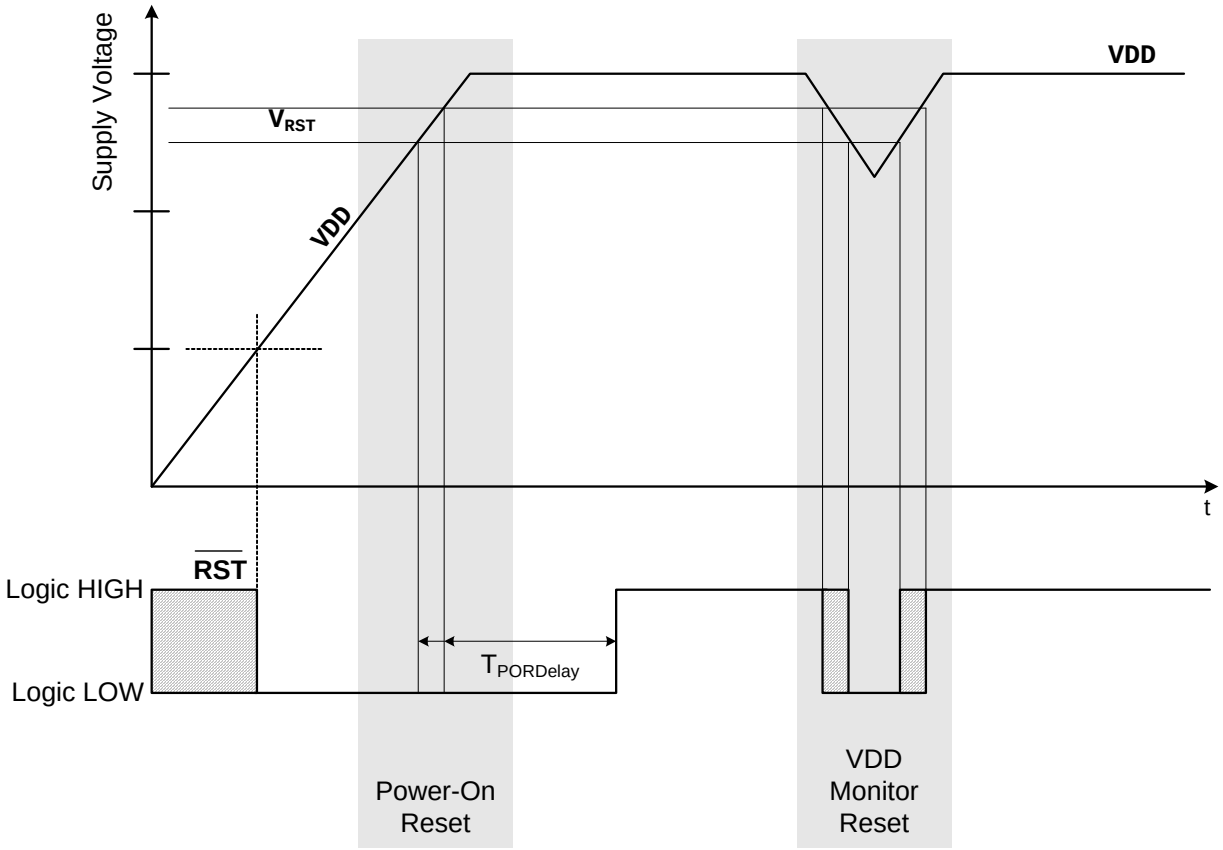


Figure 8.1. Power-On and V_{DD} Monitor Reset Timing

9. USB Interface

The Universal Serial Bus (USB) controller in the CP2501 devices is a USB 2.0 compliant full-speed device with integrated transceiver and on-chip matching and pull-up resistors. The USB function controller manages all data transfers on the USB bus, as well as command requests generated by the USB host controller. The user firmware uses the interface provided by the System Firmware to initialize and communicate using USB. The CP2501 devices also include a USB bootloader to easily update the user firmware.

The following sections describe the USB enumeration process and descriptors, and the USB bootloader capability.

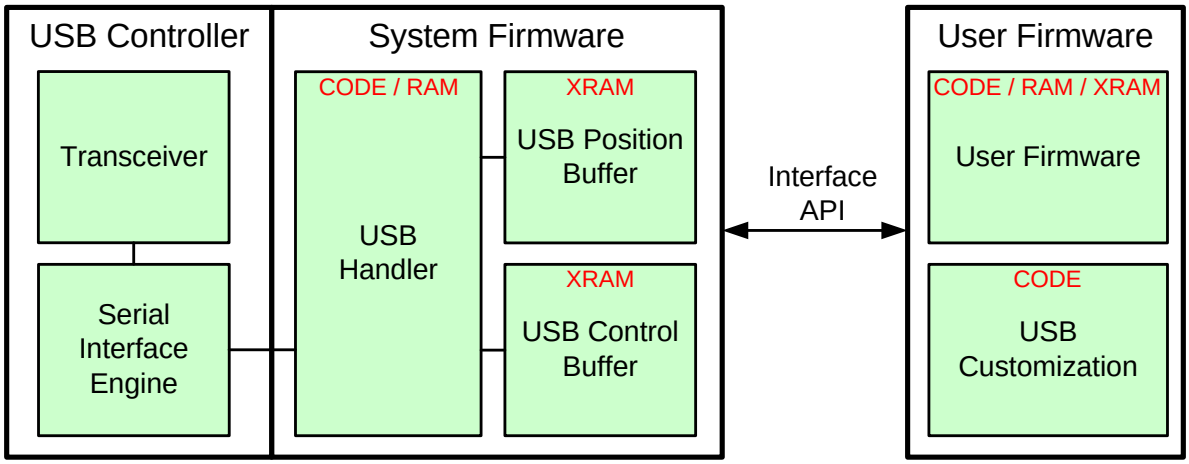


Figure 9.1. USB Interface Block Diagram

9.1. API Functionality

User code is required to use the Interface API to initialize the USB interface and also to send touch data to the host PC. The Interface API is fully documented in Application Note “AN464: CP250x API Specification”

Touch data sent to the host PC must be sent in a specific format. In addition to generating the basic firmware project, the CP2501 Configuration Wizard tool also generates a header file which includes C struct definitions. These structs make it easy for the user firmware to correctly format the data before it is passed to the System Firmware buffer.

9.2. Device Descriptors and Customization

The USB descriptors used to enumerate the CP2501 device are customizable using the CP2501 Configuration Wizard tool. These customized parameters are output to the USB descriptor source file and are compiled as part of the user project.

Table 9.1. Configurable USB Descriptors

USB Descriptor	Range
Vendor ID (VID)	2-byte integer
Product ID (PID)	2-byte integer
Manufacturer String	64 Unicode Byte String
Product String	64 Unicode Byte String
Power Requested	Up to 500 mA
Power Mode	Bus-power or Self-Power
Release Version (Major and Minor)	1 byte each for Major and Minor

A unique VID/PID combination will prevent the device from conflicting with any other USB device. A vendor ID can be obtained from www.usb.org or Silicon Laboratories can provide a free PID for the OEM product that can be used with the Silicon Laboratories VID. All CP2501 devices are pre-programmed with a unique serial number which allows multiple CP2501-based devices to be connected simultaneously to the same PC.

If the CP2501 device is enumerating as a bootloader (See Section 9.4), the following default values are used for the descriptors. These default values are included in the System Firmware block and are not user configurable.

Table 9.2. Default USB Bootloader Descriptors

USB Descriptor	Max Length / Type
Vendor ID (VID)	0x10C4
Product ID (PID)	0xEAA0
Manufacturer String	Silicon Laboratories Inc.
Product String	CP2501 HID Bootloader
Power Requested	100 mA (value is 0x32)
Bus-power or Self-Power	Bus-power
Release Version (Major and Minor)	1.0

While the device is in Bootloader mode, the on-chip regulator is enabled. If a system is designed to operate in self-powered mode, note that output of the voltage regulator will appear on the V_{DD} while the device is in bootloader mode.

9.3. Enumeration Process

On the Flash-based devices, the System Firmware checks the user firmware space for the key signature. If the signature is present, USB enumeration is handled by the user firmware using the Interface API. When the user firmware initializes the USB interface using the Interface API, the System Firmware uses the user-customized descriptors to enumerate as a USB mouse or a touch-screen device. If the key signature is not present, the System Firmware assumes that user firmware has not been programmed and uses the default bootloader descriptors to enumerate. With the default bootloader descriptors, the CP2501 device appears as a standard HID class device that supports only the features required for bootloading.

9.3.1. User-Firmware Enumeration

Once the System Firmware determines that the key signature is present, USB enumeration is handled by the user firmware. The user firmware has a choice of including basic HID mouse support along with touch-screen support. The USB mouse support is useful for when the operating system does not support touch-screens or when the PC has not yet configured the device to report touch screen data.

If USB mouse support is included, the device will initially enumerate as a USB mouse, even if touch-screen capability is available. When configured as a USB mouse, the user firmware should only send USB mouse data. During the enumeration process, the host PC will request the touch screen capabilities from the CP2501 device. After this information is received, the operating system will inform the device of its touch screen capabilities. The CP2501 user firmware can then start sending touch data that corresponds to the operating system's capabilities. If the operating system is not capable of interpreting touch data, the user firmware must continue to send only HID mouse data.

If USB mouse support is not included, the device will enumerate only as a touch screen, and wait for the host PC to indicate its touch screen capabilities.

9.4. USB Bootloader

The CP2501 devices support bootloading the user firmware over the USB interface. This feature is useful for updating the user firmware in the field or for initially programming the user firmware during production. The USB bootloader is part of the System Firmware and is accessed from the PC using the CP2501 USB Bootloader Windows application.

As described in Section 9.3, the CP2501 Flash-based devices will automatically enter bootloader mode if no user firmware is present. If user firmware is present, the CP2501 USB Bootloader can force the device to reset and enumerate as a USB bootloader. While a CP2501 is in bootloader mode, touch screen support is not available.

The CP2501 USB Bootloader application programs the CP2501 device using the hex file that is generated when compiling and building the firmware project. The application performs a boundary check on the user code and will not program a hex file that includes code located in the same code space as the System Firmware.

While the loading of the firmware is in progress, the power supply to the CP2501 should not be disturbed. If the bootloading process is stopped in the middle for any reason, the CP2501 USB Bootloader application will not program the key signature and the device will operate in bootloader mode upon the next reset. Once the bootloading process is complete, a final reset of the device is necessary before it will enumerate using the user firmware.

The CP2501 USB Bootloader application is fully described in Application Note “AN464: CP250x Programmer’s Guide and API Specification.”

10. UART Interface

The UART interface of the CP2501 devices consists of the UART_TX (transmit) and UART_RX (receive) data signals and is programmable to support a wide range of data formats and baud rates. Table 10.1 lists all of the supported data formats and baud rates.

Table 10.1. Data Formats and Baud Rates.

UART Parameter	Options
Data Bits	5, 6, 7, 8
Stop Bits	1, 1.5*, 2
Parity Type	None, Even, Odd, Mark, Space
Baud Rate	1200 baud to 500,000 baud
Note: 5-bit Mode Only	

User code initializes the UART interface, sends data, and receives data through the Interface API. The Interface API is fully documented in Application Note “AN464: CP250x API Specification”.

Data is sent over the UART interface up to 128 bytes at a time. Data is received asynchronously and user code must poll the interface waiting for a byte to arrive. The UART interface includes an 128 byte receive buffer. If a byte is received when the buffer is full, it is discarded.

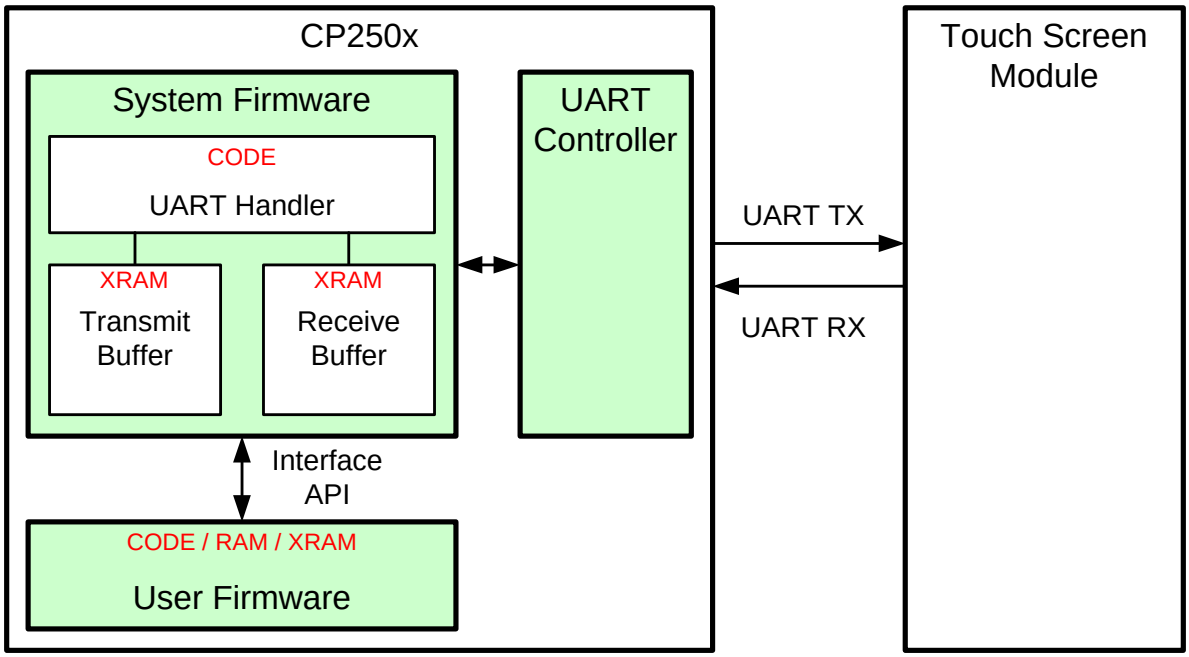


Figure 10.1. UART System Block Diagram

The timing diagram for typical configuration of 8 data bits, no parity, and 1 stop bit is shown in Table 10.2

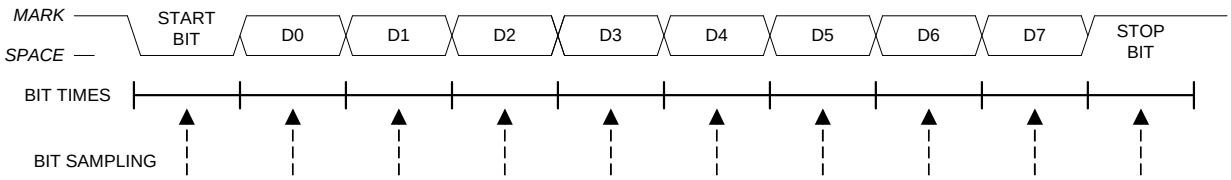


Figure 10.2. 8-Bit UART Timing Diagram

11. Serial Peripheral Interface (SPI)

The Serial Peripheral Interface provides access to a flexible, full-duplex synchronous serial bus. The SPI operates as a master device in 4-wire mode. The slave-select signal is an optional signal that can be left floating when interfacing to a 3-wire SPI device. Additional GPIO pins can be used to select multiple slave devices in master mode.

The SPI interface of the CP2501 devices consists of the SPI_SCLK (clock), SPI_MOSI (master out), SPI_MISO (master in), and SPI_NSS (slave select) signals and is programmable to support a wide range of clock speeds and configurations. Table 11.1 lists all of the supported configurations.

Table 11.1. PI Clock Speeds and Configurations

SPI Parameter	Options
Clock Speed	100 kHz to 4 MHz
Clock Phase	Data centered on first or second edge of SCK period
Clock Polarity	Low or High
NSS Logic Level	Active Low

User code initializes the SPI interface, sends data, and receives data through the Interface API. The Interface API is fully documented in Application Note “AN464: CP250x API Specification”.

Data is sent over the SPI interface up to 128 bytes at a time. Data can be requested from the SPI interface up to 128 bytes at a time. SPI transfers with the external devices are blocking actions and the user code must wait for the sends and receives to complete before proceeding.

As the SPI master, the CP2501 device initiates all data transfers on a SPI bus. Initiating a SPI Write will cause the CP2501 device to immediately shift out the data serially on the SPI_MOSI line while providing the serial clock on SPI_SCLK. While the CP2501 transfers data to a slave on the SPI_MOSI line, the addressed SPI slave device simultaneously transfers the contents of its shift register to the SPI master on the SPI_MISO line in a full-duplex operation. The SPI_NSS pin is manually controlled by user firmware to select the slave device. Additional slave devices can be addressed using general-purpose I/O pins. See Figure 5.5 for a typical connection diagram of the SPI signals. Figure 11.1 shows a block diagram for the SPI Interface.

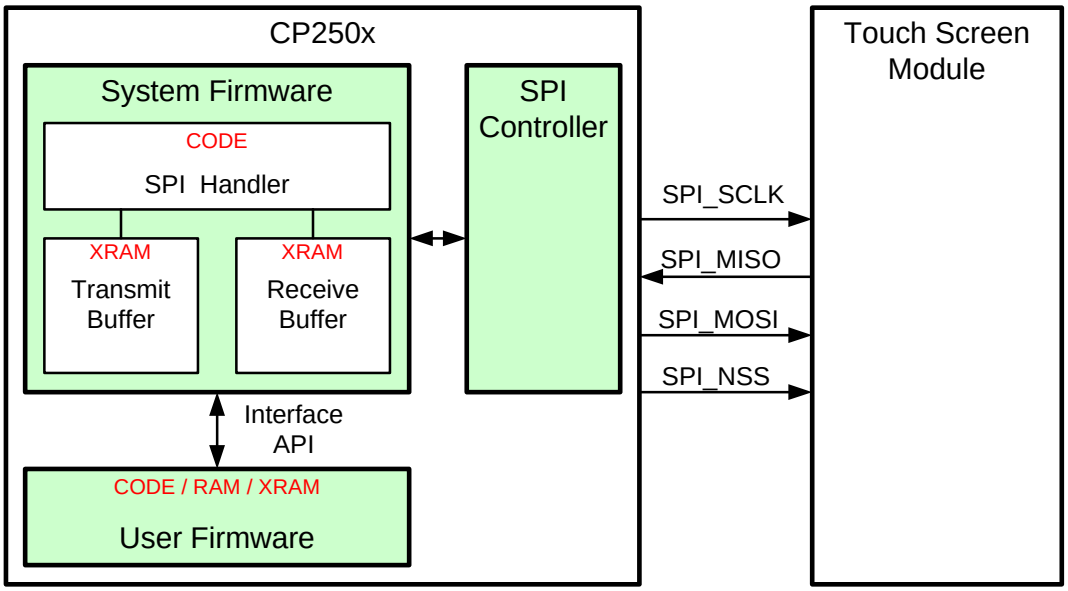


Figure 11.1. SPI Block Diagram

11.1. Signal Descriptions

The four signals used by the SPI interface (SPI_MOSI, SPI_MISO, SPI_SCK, SPI_NSS) are described below.

11.1.1. Master Out, Slave In (SPI_MOSI)

The master-out, slave-in signal is an output from the CP2501 device. It is used to serially transfer data from the master to the slave. Data is transferred most-significant bit first.

11.1.2. Master In, Slave Out (SPI_MISO)

The master-in, slave-out signal is an input to the CP2501 device. It is used to serially transfer data from the slave to the master. Data is transferred most-significant bit first. The MISO pin is placed in a high-impedance state when the SPI module is not enabled.

11.1.3. Serial Clock (SPI_SCLK)

The serial clock signal is an output from the master device. It is used to synchronize the transfer of data between the master and slave on the MOSI and MISO lines. The CP2501 device generates this signal.

11.1.4. Slave Select (SPI_NSS)

The slave-select signal is enabled as an output. The logic level of this signal is configurable using the API interface.

11.2. Serial Clock Phase and Polarity

Four combinations of serial clock phase and polarity are available. The CKPHA bit selects one of two clock phases (edge used to latch the data). The CKPOL bit selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. The clock and data line relationships are shown in Figure 11.2.

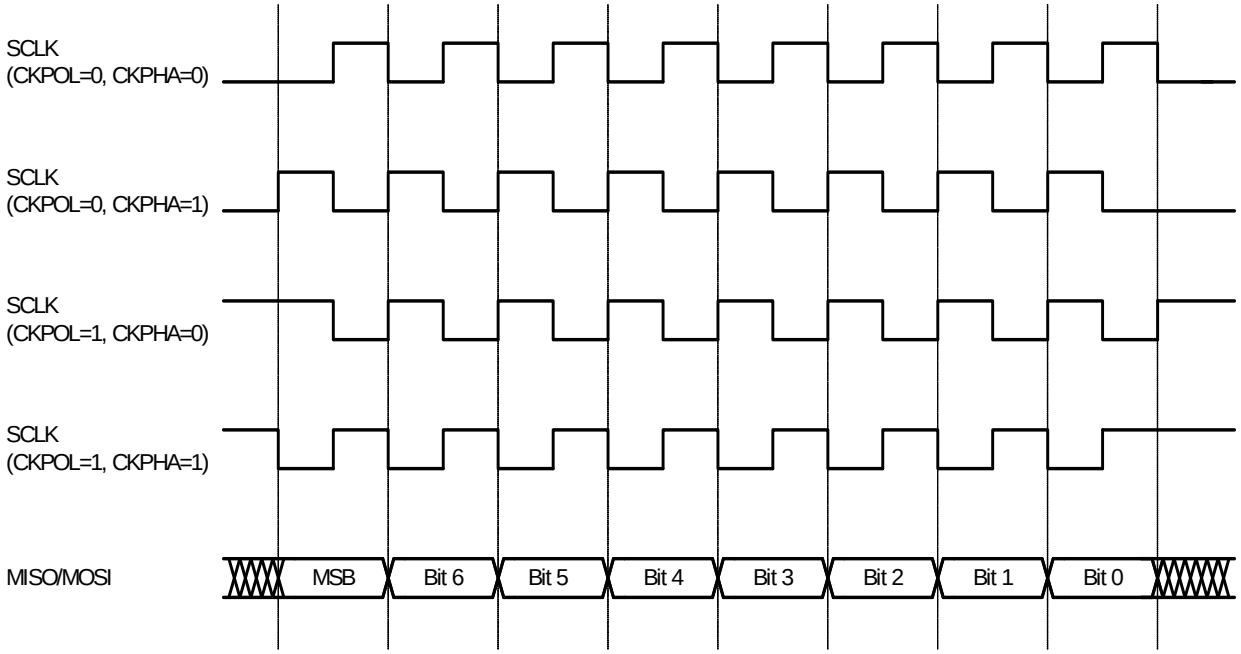
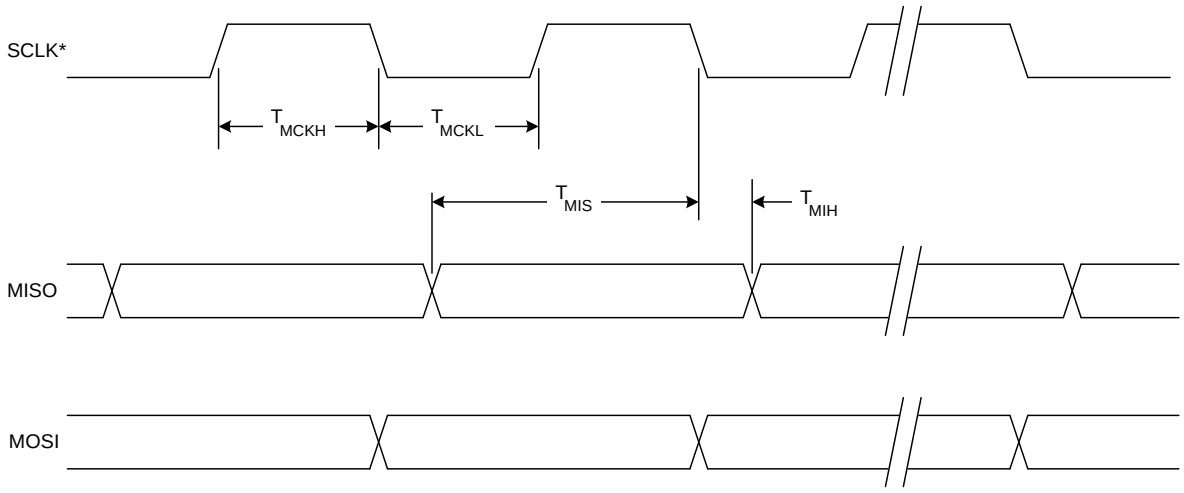


Figure 11.2. Master Mode Data/Clock Timing

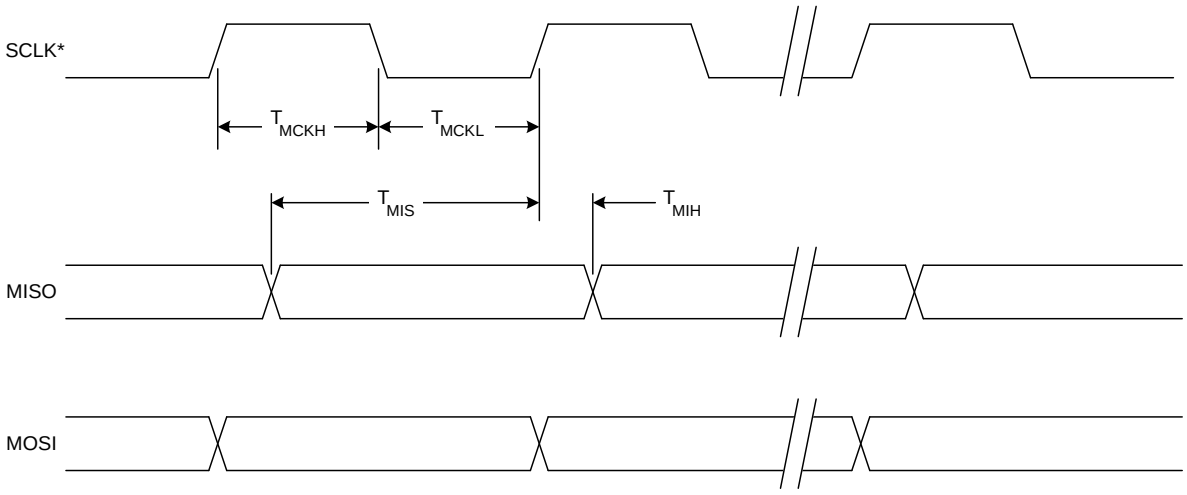
11.3. SPI Timing

The timing for the SPI interface is shown in Figure 11.3 and Figure 11.4.



* SCLK is shown for CKPOL = 0. SCLK is the opposite polarity for CKPOL = 1.

Figure 11.3. SPI Master Timing (Clock Phase = 0)



* SCLK is shown for CKPOL = 0. SCLK is the opposite polarity for CKPOL = 1.

Figure 11.4. SPI Master Timing (Clock Phase = 1)

Table 11.2. SPI Master Timing Parameters

Parameter	Description	Min	Max	Units
T_{MCKH}	SPI_SCLK High Time	1 x T_{SYSCLK}	—	ns
T_{MCKL}	SPI_SCLK Low Time	1 x T_{SYSCLK}	—	ns
T_{MIS}	SPI_MISO Valid to SPI_SCLK Shift Edge	1 x T_{SYSCLK} + 20	—	ns
T_{MIH}	SPI_SCLK Shift Edge to SPI_MISO Change	0	—	ns
Note: T_{SYSCLK} is equal to one period of the device system clock (SYSCLK).				

12. System-Management Bus (SMBus) Interface

The SMBus interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the I^2C serial bus. Reads and writes to the interface are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. A method of extending the clock-low duration is available to accommodate devices with different speed capabilities on the same bus.

The SMBus interface of the CP2501 devices consists of the SMBus_SCL (serial clock) and SMBus_SDA (serial data out) signals and is programmable to support a wide range of clock speeds which is indicated in Table 12.1.

Table 12.1. SMBus Clock Speed

SMBus Parameter	Options
Clock Speed	10 to 400 kHz

User code initializes the SMBus interface, sends data, and receives data through the Interface API. The Interface API is fully documented in Application Note “AN464: CP250x API Specification”.

Data is sent over the SMBus interface up to 128 bytes at a time. Data can be requested from the SMBus interface up to 128 bytes at a time. SMBus transfers with the external devices are blocking actions and the user code must wait for the sends and receives to complete before proceeding.

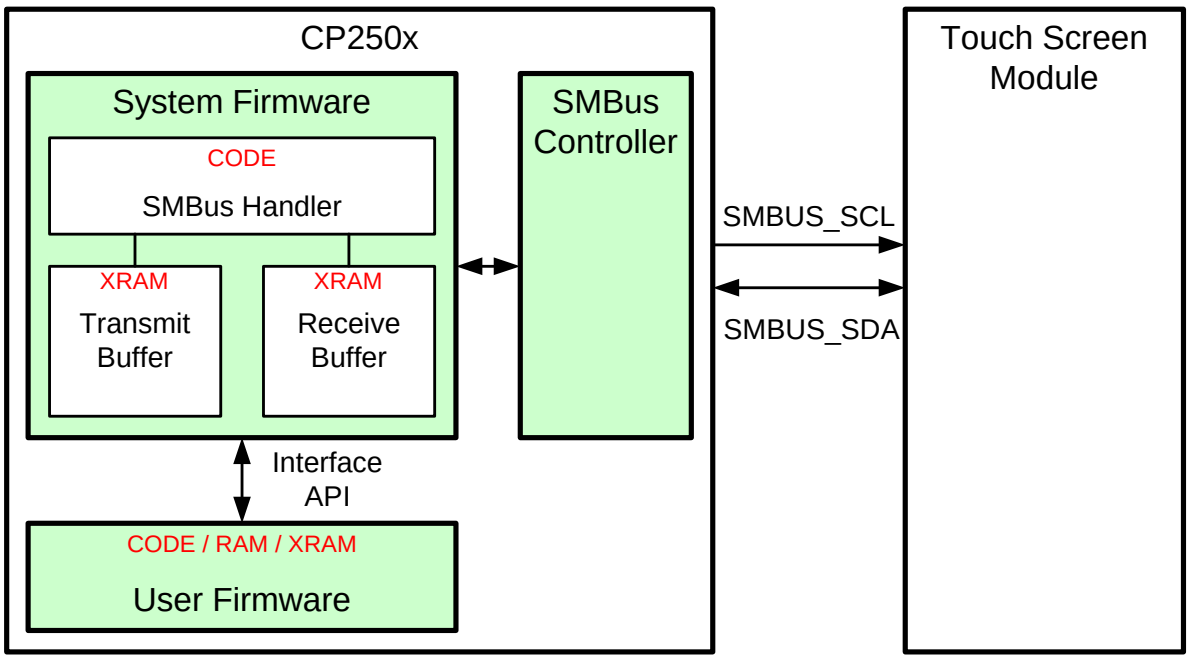


Figure 12.1. SMBus Block Diagram

12.1. SMBus Configuration

Figure 12.2 shows a typical SMBus configuration. The SMBus specification allows any recessive voltage between 3.0 V and 5.0 V; different devices on the bus may operate at different voltage levels. The bi-directional SMBUS_SCL (serial clock) and SMBUS_SDA (serial data) lines must be connected to a positive power supply voltage through a pullup resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both of the signals, so that both are pulled high (recessive state) when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus not exceed 300 ns and 1000 ns, respectively.

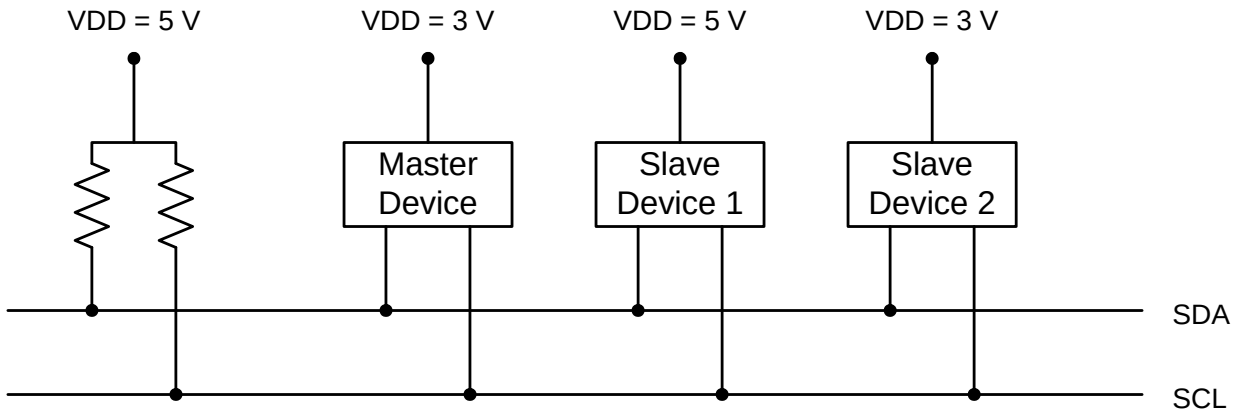


Figure 12.2. Typical SMBus Configuration

12.2. SMBus Operation

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). As a master device, the CP2501 initiates both types of data transfers and provides the serial clock pulses on SCL. Multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device who transmits a START and a slave address becomes the master for the duration of that transfer.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 12.3 illustrates a typical SMBus transaction.

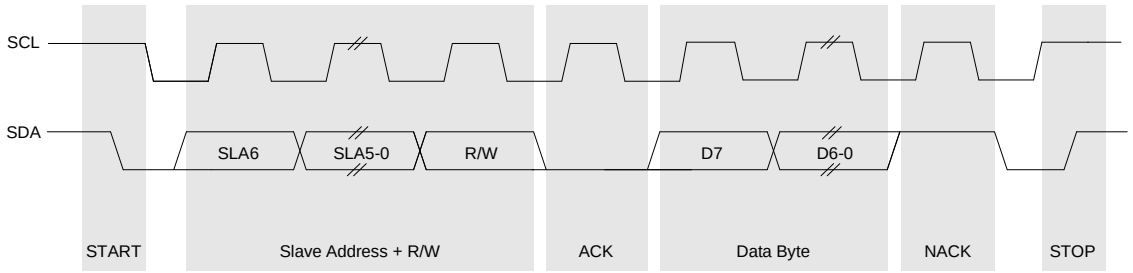


Figure 12.3. SMBus Transaction

12.2.1. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time. In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is non-destructive: one device always wins, and no data is lost.

12.2.2. Clock Low Extension

The CP2501's SMBus interface provides a clock synchronization mechanism, similar to I2C, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SMBUS_SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

12.2.3. SMBUS_SCL Low Timeout

If the SMBUS_SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a "timeout" condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition. The CP2501 devices return a timeout through the Interface API.

13. General Purpose Input / Output pins (GPIO)

The CP2501 supports 16 GPIO pins. These 16 GPIO pins are accessible as individual pins, or as two 8-wire buses.

The operating mode of each of the GPIO pins is individually configurable as open-drain output, push-pull output, or input. The difference between an open-drain output and a push-pull output is when the GPIO output is driven to logic high. A logic high, open-drain output pulls the pin to the V_{DD} rail through an internal, pull-up resistor. A logic high, push-pull output directly connects the pin to the V_{DD} voltage. Open-drain outputs are typically used when interfacing to logic at a higher voltage than the V_{DD} pin. These pins can be safely pulled to the higher, external voltage through an external pull-up resistor. The maximum external pull-up voltage is 5 V.

User code is required to initialize the GPIO pins and their operating mode using the Interface API. The Interface API is fully documented in Application Note AN464: CP250x API Specification.

The GPIO pins can be used to poll for interrupt signals from the external modules, drive LEDs, or perform other control functions required by the system. The timing of the GPIO signalling is determined by the 8051 instruction used to access the GPIO pin. See Table 6.1, “CIP-51 Instruction Set Summary,” on page 27 for information regarding the cycle counts for various instructions.

NOTES:

CONTACT INFORMATION

Silicon Laboratories Inc.

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.