AN127: Flash Programming via the C2 Interface

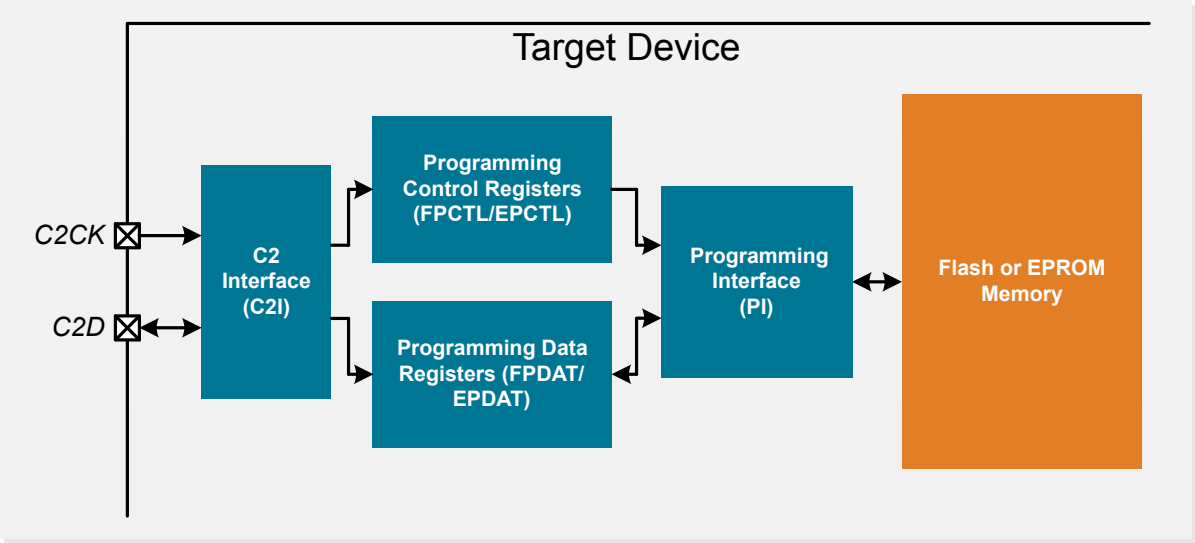
This application note describes how to program the flash or EPROM memory on Silicon Labs microcontroller devices that use the C2 interface (e.g., C8051F30x or C8051T61x). Example software is included.

C2 devices have a Programming Interface (PI) that is accessed via the C2 Interface (C2I) and a set of programming registers. Figure 1 shows the flash access block diagram.

To access the flash or EPROM memory of a C2 device, the programmer must comprehend the C2 interface, the programming registers, and the programming interface.

KEY POINTS

- The C2 interface is a two-wire interface used by most Silicon Labs 8-bit MCUs.
- This interface uses a command-based protocol to modify flash and SFR contents.
- Some devices require additional configuration (i.e. enabling the VDD monitor) to program.



1. C2 Interface

The Silicon Labs 2-Wire Interface (C2) is a two-wire serial communication protocol designed to enable in-system programming and debugging on low pin-count Silicon Labs devices. C2 communication involves an interface master (the programmer/debugger/tester) and an interface target (the device to be programmed/debugged/tested). The two wires used in C2 communication are C2 Data (C2D) and C2 Clock (C2CK).

C2 facilitates a pin-sharing scheme, where the C2 pins on the target device are available for user functions while C2 communication is idle. Each C2 frame is initiated with a START condition on the C2CK pin that signals the target device to configure its C2D pin for C2 communication. Each C2 frame terminates with a STOP condition on the C2CK signal that allows the target device to restore its C2D pin to its user-defined state. The C2CK signal is typically shared with an active-low reset signal (RST) signal on the target device. In this configuration, the width of a low strobe is used to differentiate between a C2 communication strobe and a reset event.

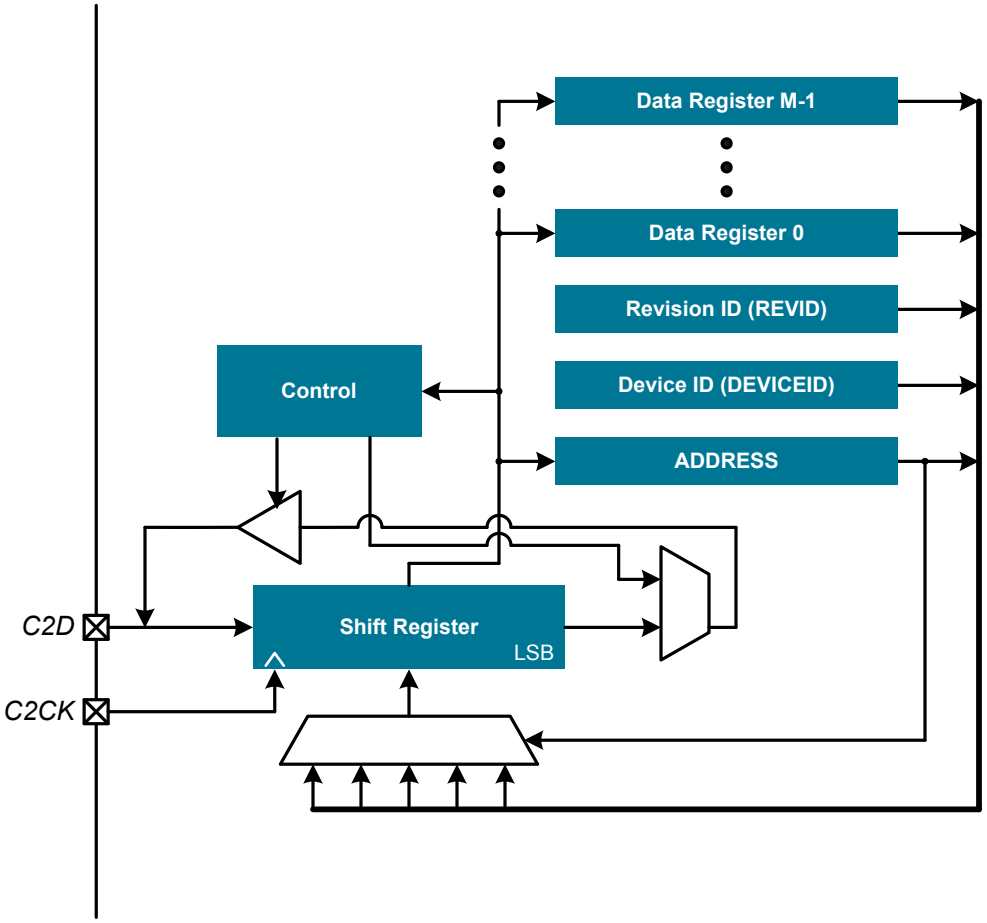


Figure 1.1. C2 Interface Block Diagram

1.1 C2 Basics

The C2 interface operates similar to JTAG with the three JTAG data signals (TDI, TDO, TMS) mapped into one bidirectional C2 data signal (C2D). The signal direction of C2D is strictly specified by the instruction protocol such that contention between the target device and interface master is never allowed. All data is transmitted and received LSB first.

The C2 interface provides access to on-chip programming and debug hardware through a single Address register and a set of Data registers. The Address register defines which Data register will be accessed during Data register read/write instructions (analogous to the JTAG Instruction register). Data registers provide access to various device-specific functions (note: it is not required that all Data registers be both readable and writable). Read and write access to all registers is performed through a common shift register that serves as a serial-to-parallel-to-serial converter.

All C2 devices include an 8-bit Device ID register and an 8-bit Revision ID register. These registers are read-only. Following a device reset, the C2 Address register defaults to 0x00, selecting the 8-bit Device ID register. This allows a C2 master to perform a Device ID register read without knowing the length of the target device's Address register. The length of the target Address register can then be determined using the Device ID register content.

1.2 C2 Registers

1.2.1 ADDRESS: Address Register

The C2 Address register (ADDRESS) serves two purposes in C2 flash or EPROM programming:

1. ADDRESS selects which C2 Data register is accessed during C2 Data Read/Write frames.
2. During Address Read frames, ADDRESS provides PI status information.

Address Reads are used frequently during C2 programming as a handshaking scheme between the programmer and the PI.

The Address Read command returns an 8-bit status code, formatted as shown in the table below.

Table 1.1. C2 Address Register Status Bits

Bit	Name	Description
7	EBusy or FLBusy	This bit indicates when the EPROM or Flash is busy completing an operation.
6	EError	This bit is set to 1 when the EPROM encounters an error.
5:2	—	Unused
1	InBusy	This bit is set to 1 by the C2 Interface following a write to FPDAT. It is cleared to 0 when the PI acknowledges the write to FPDAT.
0	OutReady	This bit is set to 1 by the PI when output data is available in the FPDAT register.

The InBusy bit should be polled following any write to FPDAT, and the OutReady bit should be polled before any reads of FPDAT.

1.2.2 DEVICEID: Device ID Register

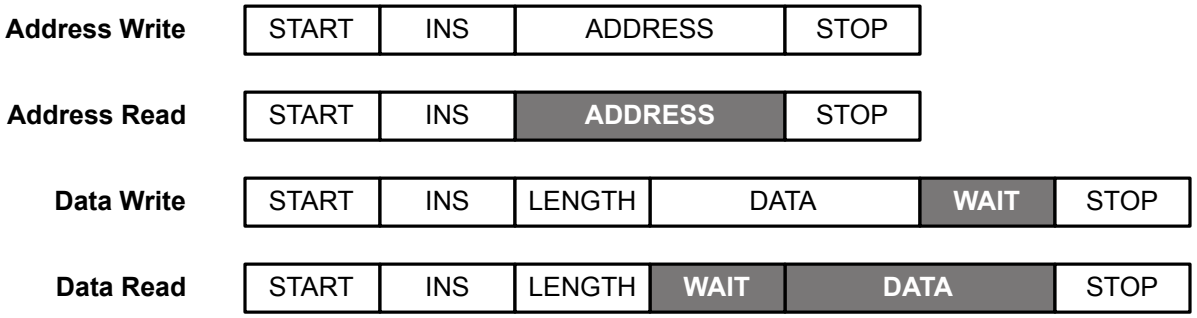
The Device ID register (DEVICEID) is a read-only C2 Data register containing the 8-bit device identifier of the target C2 device. The C2 address for register DEVICEID is 0x00.

1.2.3 REVID: Revision ID Register

The Revision ID register (REVID) is a read-only C2 Data register containing the 8-bit revision identifier of the target C2 device. The C2 address for register REVID is 0x01.

1.3 C2 Instruction Frames

A C2 master accesses the target C2 device via a set of four basic C2 frame formats: Address Write, Address Read, Data Write, and Data Read.



Note: During shaded fields, the C2D signal is driven by the target device.

Figure 1.2. C2 Frame Summary

Note that the master initiates each frame with the START and INS (Instruction) fields. The content of the INS field defines the frame format.

Table 1.2. C2 Instructions

Instruction	INS Code
Data Read	00b
Address Read	10b
Data Write	01b
Address Write	11b

Table 1.3. C2 Bit Field Descriptions

	Field	Description
Master Only	START	A START condition initiates a C2 frame. The master generates this condition by leaving its C2D driver disabled and generating an active-low strobe on C2CK. All C2 frames begin with the START field.
	INS	The INS field is a 2-bit code specifying the current C2 instruction. The four valid C2 instructions are shown in Table 2. All C2 frames include the INS field.
	STOP	A STOP condition ends a C2 frame. The master generates this condition by disabling its C2D driver and generating an active-low C2CK strobe. The slave returns C2D to its user-defined state on the rising edge of this C2CK strobe. All C2 frames are terminated with the STOP field.
	LENGTH	The LENGTH field is a 2-bit code indicating the number of bytes to be read or written during Data register accesses. The number of bytes to transfer is LENGTH + 1 (for example, LENGTH = 01b results in a 2-byte transfer).
Master or Slave	ADDRESS	The ADDRESS field is used to transfer data during Address register accesses. The length of this field must be the same length as the slave device's Address register. The Address register defaults to all zero's following any reset, selecting the Device ID register.
	DATA	The DATA field appears in Data register accesses; the length of this field is determined by the LENGTH field as described above.

	Field	Description
Slave Only	WAIT	A WAIT field appears during Data Read and Data Write frames to allow the slave device to access slower registers or memories. This variable-length field consists of a series of zero or more 0's transmitted by the slave device, terminated by a single 1.
Note: All fields are transmitted LSB first.		

1.3.1 Address Write Frame

An Address Write frame loads the target Address register.

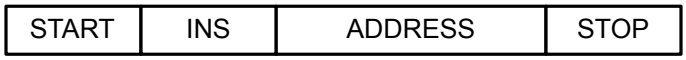


Figure 1.3. Address Write Sequence

The length of the ADDRESS field must always be the length of the slave device's Address register. Following a device reset, the target device's Address register defaults to all zeros, selecting the Device ID register.
Note: All fields are transmitted LSB first.

1.3.2 Address Read Frame

An Address Read frame returns status information or Address register contents from the target device. This instruction is typically used to quickly access status information, though the function of the Address Read instruction is specific to each target device.

Figure 1.4. Address Read Sequence

The length of the ADDRESS field must always be the length of the slave device's Address register.
Note: All fields are transmitted LSB first.

1.3.3 Data Write Frame

A Data Write frame writes a specified value to the target Data register, as selected by the target Address register.

Figure 1.5. Data Write Sequence

LENGTH is a 2-bit field that specifies the length of the DATA field as follows:
DATA length in bytes = LENGTH + 1

The DATA field length must be a multiple of 8 bits. For example, a LENGTH of 01b indicates a DATA length of 2 bytes. The length of the DATA field is not required to be the same length as the target Data register. For example, to write only the eight MSBs of a 10-bit register, LENGTH is set to 00b and DATA specifies only 8 bits of data to be written to the 8 MSBs of the target register. The remaining register bits are undefined. To write all 10 bits of data, LENGTH should be 01b; in this case (shown in Figure 7), the 10 MSBs of the 16-bit DATA field are written to the target register.
Note: All fields are transmitted LSB first.

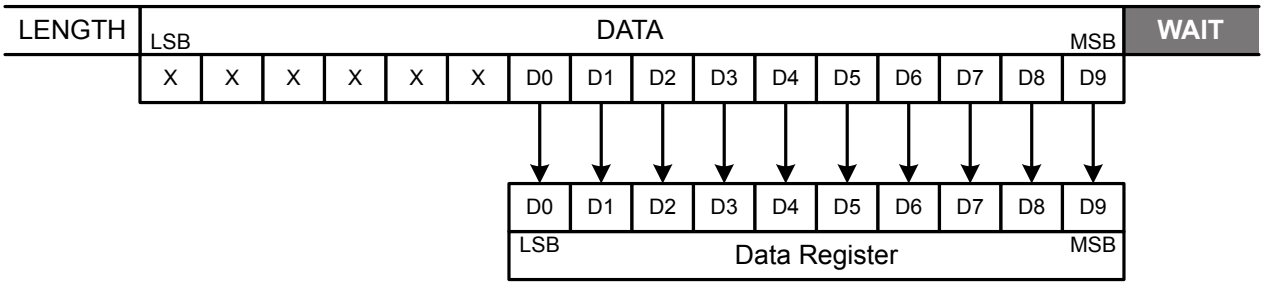


Figure 1.6. DATA Field for a 10-bit Data Register Write

The length of the WAIT field is controlled by the target device. During the WAIT field, the target device transmits 0's on the C2D pin until it has finished writing to the target Data register. To indicate the write complete status, the target device transmits a 1 to terminate the WAIT field.

1.3.4 Data Read Frame

A Data Read frame reads the contents of the target Data register, as selected by the target Address register.

Figure 1.7. Data Read Sequence

LENGTH is a 2-bit field that specifies the length of the DATA field as follows:
DATA length in bytes = LENGTH + 1

The DATA field length must be a multiple of 8 bits. For example, a LENGTH of 01b indicates a DATA length of 2 bytes. As with the Data Write frame, the length of the DATA field is not required to match the length of the target Data register. In this case, the read data is right justified in the DATA field. For example, if LENGTH is 00b (1 byte) and the target register is 12-bits, the 8 LSBs of the target register are read into the DATA field. If LENGTH is 01b (2 bytes) and the target register is 12-bits, the 12-bit Data register makes up the 12 LSBs of the DATA field; the remaining 4 bits are undefined.
Note: All fields are transmitted LSB first.

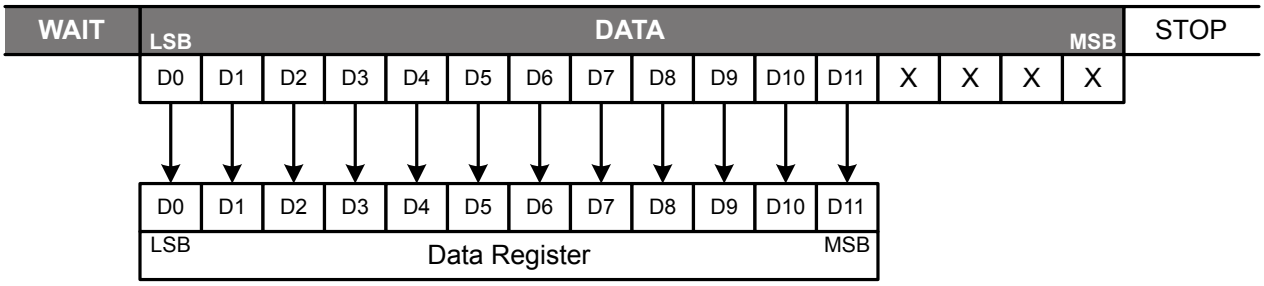


Figure 1.8. DATA Field for a 10-bit Data Register Read

The length of the WAIT field is controlled by the target device. During the WAIT field, the target device transmits 0's on the C2D pin until it has finished reading the target Data register and is ready to shift out data. To indicate the ready status, the target device transmits a 1 to terminate the WAIT field.

1.4 C2 Timing Specifications

This section illustrates the timing sequence for each of the four C2 frame formats and the device reset command.

Table 1.4. C2 Timing Requirements

Parameter	Description	Min	Max
t_{RD}	C2CK low time for a device reset	20 µs	—
t_{SD}	Start bit delay after a device reset	2 µs	—
t_{CL}	C2CK low time for bit transfers	20 ns	5000 ns
t_{CH}	C2CK high time	20 ns	—
t_{DS}	C2D setup time	10 ns	—
t_{DH}	C2D hold time	10 ns	—
t_{ZS}	C2D High-Z setup time	0 ns	—
t_{DV}	C2D valid	—	20 ns
t_{ZV}	C2D High-Z valid	—	20 ns

Because C2CK and RST functions share the same pin, they are distinguished by the length of time the pin is held low. For RST, the pin must be held low for at least 20 µs. For C2CK, the pin cannot be low for longer than 5 µs. If the pin is held low between 5 and 20 µs, the response of the device is undefined.

Data is sampled on C2D on the rising edge of C2CK. C2D changes (as an output) shortly after the rising edge of C2CK. Typical setup and data ready times for C2D are tens of nanoseconds; however some devices may prefer more time (such as the C8051F41x). In general, if C2D is set up before C2CK goes low, C2CK low time is between 80 ns and 5 µs, and C2D is read at least 120 ns after C2CK rising, C2 interface timing will be satisfied.

If the C2CK master is an MCU, care should be taken to disable interrupts while C2CK is low in order to prevent a clock low time extending beyond 5 µs and potentially causing a target device reset.

Shaded C2D bits in these section diagrams indicate times when the master's C2D driver must be disabled.

1.4.1 Device Reset Timing

During C2 instructions, C2CK must not be held low longer than t_{CL}. This requirement allows the device to be reset by holding C2CK low for t_{RD}. The START field of the first C2 instruction must begin at least t_{SD} after C2CK returns high following a device reset.

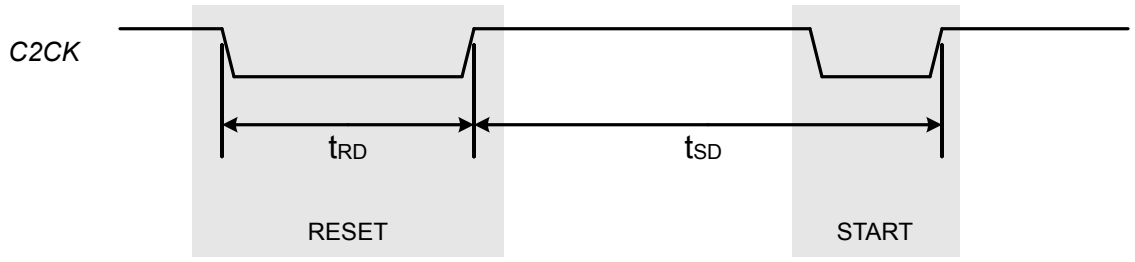


Figure 1.9. Device Reset Timing

1.4.2 Address Write Timing

The 8-bit Address Register Write frame begins with a START (rising edge on C2CK). Note that during a START condition, the master's C2D driver should be disabled. Following the START, the interface master must enable its C2D driver to transmit the INS and ADDRESS bits. Following the last ADDRESS bit, the master disables its C2D driver and strobes C2CK one last time for the STOP field; the slave device returns the C2D pin to its user-defined state following the last rising edge on C2CK.

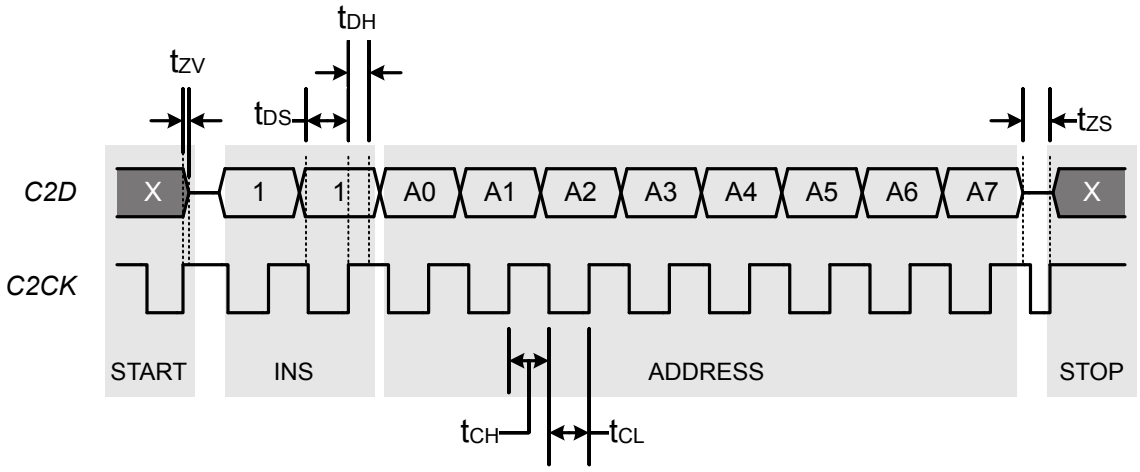


Figure 1.10. Address Write Timing

1.4.3 Address Read Timing

The 8-bit Address Register Read frame begins with a START followed by the 2-bit INS field. Following the INS bits, the interface master disables its C2D driver and strobes C2CK; the slave device outputs the LSB of its Address register on the rising edge of C2CK. Seven more C2CK strobes are required to complete the ADDRESS field. Following the last ADDRESS bit, the master strobes C2CK one last time for the STOP field; the slave device returns the C2D pin to its user-defined state following the last rising edge on C2CK.

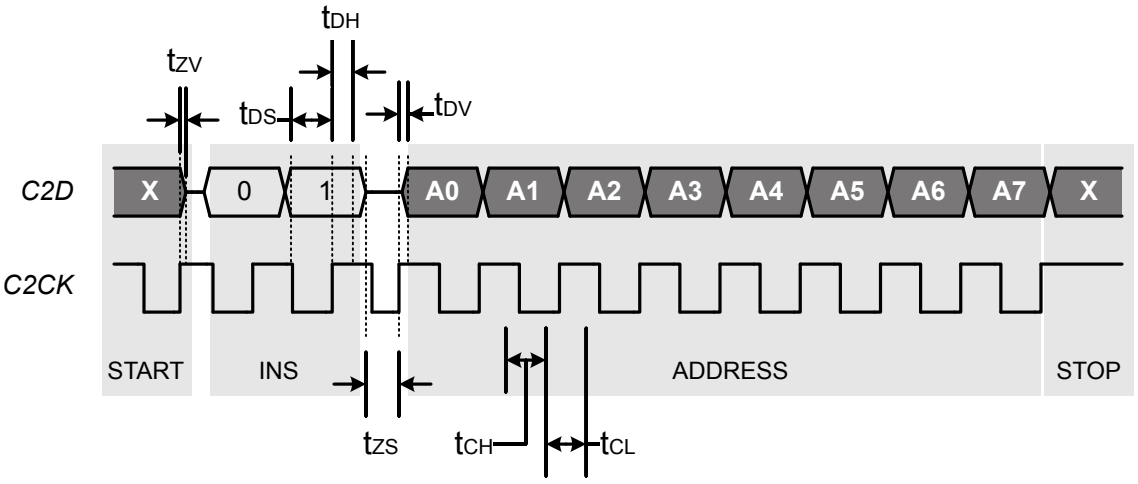


Figure 1.11. Address Read Timing

1.4.4 Data Write Timing

The 1-byte Data Register Write frame begins with a START followed by the 2-bit INS and 2-bit LENGTH fields. In this example, the LENGTH field is 00b indicating a 1-byte transfer. The master transmits the 8-bits of data; following the last DATA bit, the master disables its C2D driver for the WAIT field. In this example the slave transmits only one 0 during the WAIT field. Following the WAIT field, the master strobes C2CK one last time for the STOP field; the slave device returns the C2D pin to its user-defined state following the last rising edge on C2CK.

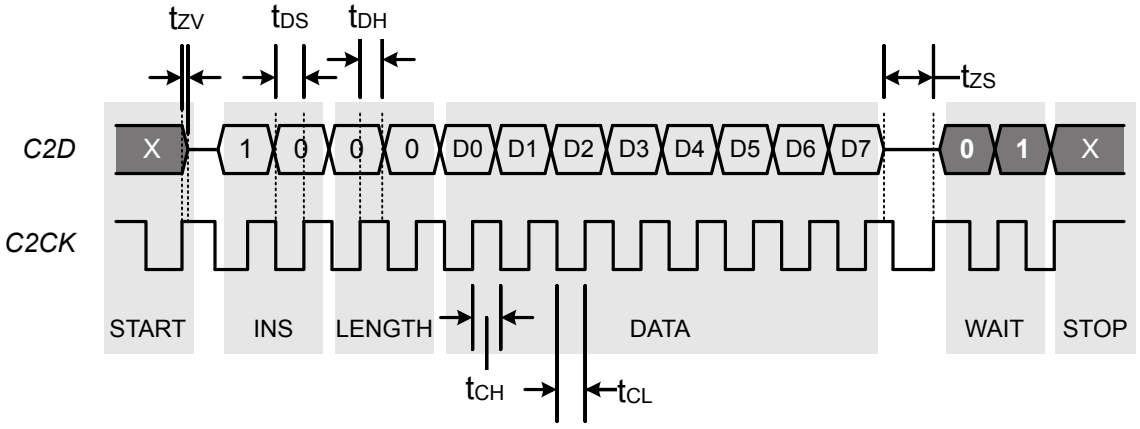


Figure 1.12. Data Write Timing

1.4.5 Data Read Timing

The 1-byte Data Register Read frame begins with a START followed by the 2-bit INS and 2-bit LENGTH fields. In this example, the LENGTH field is 00b indicating a 1-byte transfer. After the last bit of the LENGTH field is transmitted, the master disables its C2D driver for the WAIT field. In this example only one 0 is transmitted during the WAIT field. Following the WAIT field, the master strobes C2CK and the slave shifts out the DATA field. Following the last DATA bit, the master strobes C2CK one last time for the STOP field; the slave device returns the C2D pin to its user-defined state following the last rising edge on C2CK.

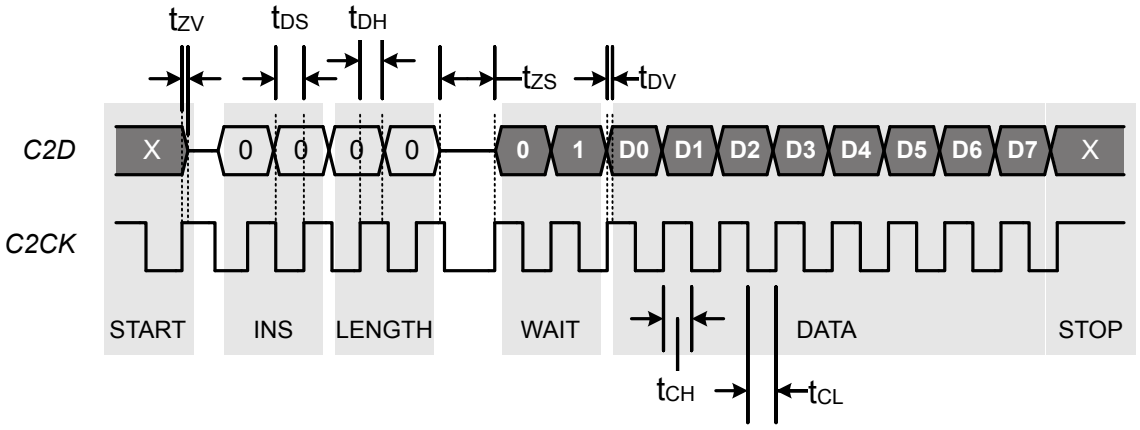


Figure 1.13. Data Read Timing

1.5 C2 Primitives

This section discusses the specific steps for generating various C2 operations.

1.5.1 Device Reset

To generate the reset timing:

1. Turn the C2CK driver on.
2. Force C2CK low.
3. Wait at least 20 µs.
4. Force C2CK high.
5. Wait at least 2 µs.
6. (Optional) Turn the C2CK driver off.

1.5.2 C2CK Clock Strobes

To generate C2CK clock strobes with a microcontroller-based programmer:

1. Turn the C2CK driver on.
2. Wait at least 40 ns. This helps ensure C2D data setup time.
3. Force C2CK low. Ensure interrupts are disabled at this point.
4. Wait between 80 and 5000 ns.
5. Force C2CK high.
6. Wait at least 120 ns. This helps ensure C2D data valid time.

1.5.3 Performing the Address Write Instruction

The C2CK strobes mentioned in this section refer to the steps described in 1.5.2 C2CK Clock Strobes. To write to a target device's ADDRESS register:

1. Disable interrupts.
2. Turn the C2CK driver on.
3. Strobe C2CK. This will generate the START field and forces the C2D pin on the target device to become an input.
4. Turn on the C2D driver.
5. Force C2D high. This sets C2D for the 2-bit INS field, and the Address Write instruction is 11b.
6. Strobe C2CK. This transfers the first bit of the INS field.
7. Strobe C2CK. This transfers the second bit of the INS field.
8. Force C2D to each bit of the address value being written to ADDRESS (starting with bit 0) and strobe C2CK for each bit.
9. Turn off the C2D driver. This prepares C2D to possibly become an output from the target device.
10. Strobe C2CK to generate the STOP field.
11. (Optional) Turn off the C2CK driver.
12. Re-enable interrupts.

1.5.4 Performing the Address Read Instruction

The C2CK strobes mentioned in this section refer to the steps described in 1.5.2 C2CK Clock Strobes. The OutReady and InBusy bits returned by the Address Read instruction are described in 1.2.1 ADDRESS: Address Register.

To read the status code from a target device's ADDRESS register:

1. Disable interrupts.
2. Turn the C2CK driver on.
3. Strobe C2CK. This will generate the START field and forces the C2D pin on the target device to become an input.
4. Turn on the C2D driver.
5. Force C2D low. This sets C2D for the first part of the 2-bit INS field, and the Address Read instruction is 01b.
6. Strobe C2CK. This transfers the first bit of the INS field.
7. Force C2D high. This sets C2D for the second part of the 2-bit INS field.
8. Strobe C2CK. This transfers the second bit of the INS field.
9. Turn off the C2D driver. This prepares C2D to become an output from the target device.
10. Strobe C2CK and read C2D from the target device. The device will return the 8-bit status code returned from an Address Read instruction bit by bit starting with bit 0. C2CK must be strobed after each read of C2D.
11. Strobe C2CK to generate the STOP field.
12. (Optional) Turn off the C2CK driver.
13. Re-enable interrupts.

1.5.5 Performing the Data Write Instruction

The C2CK strobes mentioned in this section refer to the steps described in 1.5.2 C2CK Clock Strobes. To generate a Data Write instruction:

1. Disable interrupts.
2. Turn on the C2CK driver.
3. Strobe C2CK. This will generate the START field and forces the C2D pin on the target device to become an input.
4. Turn on the C2D driver.
5. Force C2D high. This sets C2D for the first part of the 2-bit INS field, and the Data Write instruction is 10b.
6. Strobe C2CK. This transfers the first bit of the INS field.
7. Force C2D low. This sets C2D for the second part of the 2-bit INS field.
8. Strobe C2CK. This transfers the second bit of the INS field.
9. Leave C2D low. This is the start of the LENGTH field, which will be 00b in this example.
10. Strobe C2CK to start the LENGTH field.
11. Strobe C2CK to finish transferring the LENGTH field.
12. Force C2D to each bit of the data value being written to the data register (starting with bit 0) and strobe C2CK for each bit.
13. Turn off the C2D driver. This prepares C2D to become an output from the target device.
14. Srobe C2CK.
15. Check C2D for a value of 1b. If C2D becomes 1, the WAIT field is over, as it can contain zero or more 0's and exactly one 1b. If C2D is not 1b, strobe C2CK and check the status again.
16. At this point, C2D is driving a 1, indicating the end of the WAIT field. Strobe C2CK to generate the STOP field.
17. (Optional) Turn off the C2CK driver.
18. Re-enable interrupts.

1.5.6 Performing the Data Read Instruction

The C2CK strobes mentioned in this section refer to the steps described in 1.5.2 C2CK Clock Strobes. To generate a Data Read instruction:

1. Disable interrupts.
2. Turn on the C2CK driver.
3. Strobe C2CK. This will generate the START field and forces the C2D pin on the target device to become an input.
4. Turn on the C2D driver.
5. Force C2D low. This sets C2D for the first part of the 2-bit INS field, and the Data Read instruction is 00b.
6. Strobe C2CK to transfer the first bit of the INS field.
7. Strobe C2CK to transfer the second bit of the INS field.
8. Leave C2D low. This is the start of the LENGTH field, which will be 00b in this example.
9. Strobe C2CK to start the LENGTH field.
10. Strobe C2CK to finish transferring the LENGTH field.
11. Turn off the C2D driver. This prepares C2D to become an output from the target device.
12. Strobe C2CK.
13. Check C2D for a value of 1b. If C2D becomes 1, the WAIT field is over, as it can contain zero or more 0's and exactly one 1b. If C2D is not 1b, strobe C2CK and check the status again.
14. At this point, C2D is driving a 1, indicating the end of the WAIT field.
15. Strobe C2CK and read C2D from the target device. The device will return the contents of the data register bit by bit starting with bit 0. C2CK must be strobed after each read of C2D.
16. Strobe C2CK to generate the STOP field.
17. (Optional) Turn off the C2CK driver.
18. Re-enable interrupts.

2. Programming Registers

Communication between the PI and C2I is accomplished via two flash programming registers: FPCTL and FPDAT. EPROM devices have additional EPCTL, EPDAT, EPADDRH, EPADDRL, and EPSTAT registers.

2.1 FPCTL: Flash Programming Control Register

The Flash Programming Control register (FPCTL) serves to enable C2 flash or EPROM programming. To enable C2 programming, the following key codes must be written to FPCTL in the following sequence:

1. 0x02
2. 0x04
3. 0x01

The key codes must be written in this sequence following a target device reset. Once the key codes have been written to FPCTL, the target device is halted until the next device reset.

2.2 FPDAT: Flash Programming Data Register

The Flash Programming Data register (FPDAT) is used to pass all data between the C2I and PI on Flash devices. Information passed via FPDAT includes:

- All PI Commands (C2I-to-PI)
- PI Status Information (PI-to-C2I)
- Flash Addresses (C2I-to-PI)
- Flash Data (both directions)

2.3 EPCTL: EPROM Programming Control Register

The EPROM Programming Control register (EPCTL) contains additional controls for EPROM programming over the C2 interface. To enable C2 programming, the following key codes must be written to EPCTL:

1. 0x40
2. 0x58

The key codes must be written in this sequence after the initial writes to FPCTL.

2.4 EPDAT: EPROM Programming Data Register

The EPROM Programming Data register (EPDAT) is used to pass all data between the C2I and PI on EPROM devices. Information passed via EPDAT includes:

- All PI Commands (C2I-to-PI)
- PI Status Information (PI-to-C2I)
- EPROM Addresses (C2I-to-PI)
- EPROM Data (both directions)

2.5 EPADDRH and EPADDRL: EPROM Programming Address

The EPROM Programming Address registers (EPADDRH and EPADDRL) contain the address written to by the PI on the EPROM device.

2.6 EPSTAT: EPROM Programming Status

The EPROM Status register (EPSTAT) provides additional status information when programming the EPROM of a device.

Table 2.1. EPROM Programming Status Bits

Bit	Name	Description
7	WLOCK	This bit indicates the device is locked from EPROM writes.
6	RLOCK	This bit indicates the device is locked from EPROM reads.
5:4	—	Reserved
3:1	—	Unused
0	ERROR	This bit is set to 1 by the PI of an EPROM device detects an error.

3. Programming Interface

The Programming Interface (PI) performs a set of programming commands. Each command is executed using a sequence of reads and writes of the FPDAT register.

Table 3.1. C2 Flash Programming Commands

Command	Code	Supported Devices
Block Write	0x07	all (flash or EPROM)
Block Read	0x06	all (flash or EPROM)
Page Erase	0x08	flash devices only
Device Erase	0x03	flash devices only
Get Version	0x01	all (flash or EPROM)
Get Derivative	0x02	all (flash or EPROM)
Direct Read	0x09	all (flash or EPROM)
Direct Write	0x0A	all (flash or EPROM)
Indirect Read	0x0B	all (flash or EPROM)
Indirect Write	0x0C	all (flash or EPROM)

The most-commonly used commands for flash or EPROM programming are Block Write, Block Read, Page Erase, and Device Erase.

The PI must be initialized with the following sequence:

1. Drive C2CK low.
2. Wait at least 20 µs.
3. Drive C2CK high.
4. Wait at least 2 µs.
5. Perform an Address Write instruction targeting the FPCTL register (0x02).
6. Perform a Data Write instruction sending a value of 0x02.
7. Perform a Data Write instruction sending a value of 0x04 to halt the core.
8. Perform a Data Write instruction sending a value of 0x01.
9. Wait at least 20 ms.

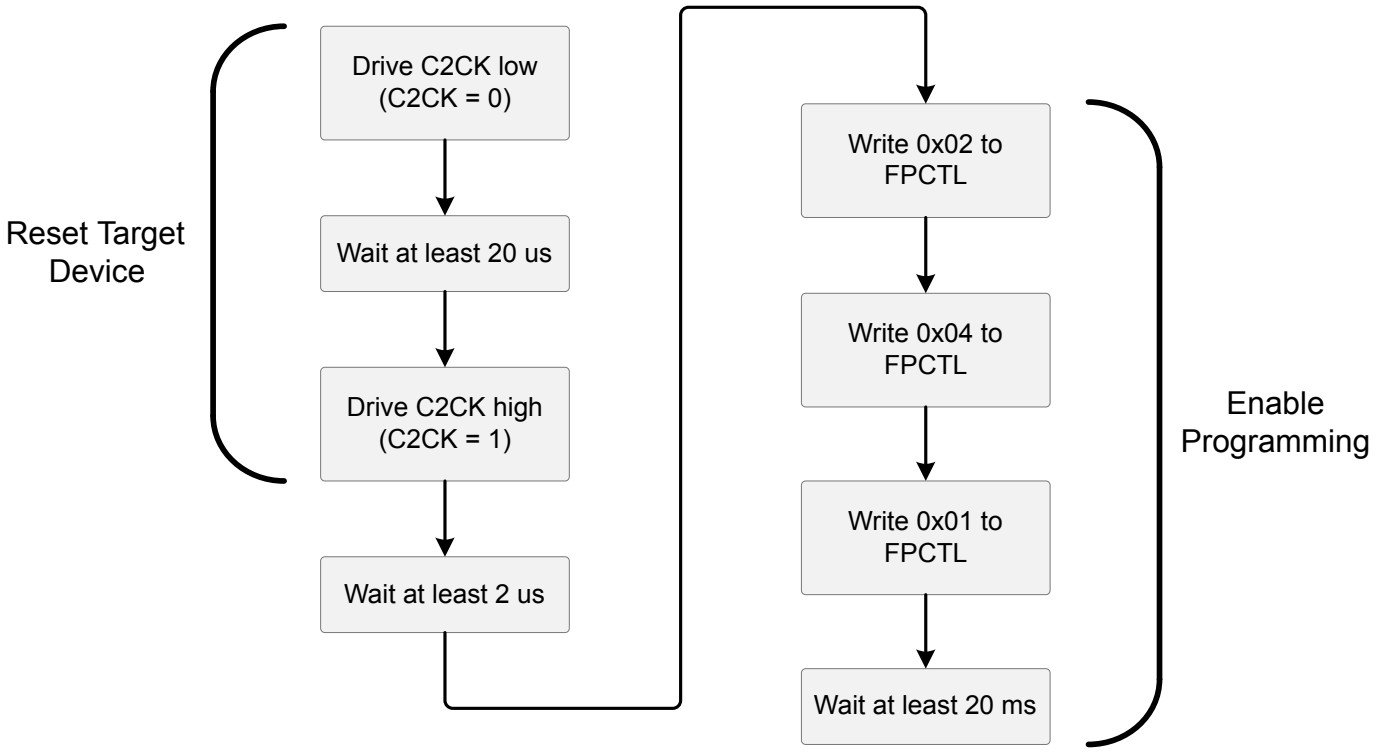


Figure 3.1. PI Initialization Sequence

Some devices need additional configuration before executing write or erase operations. See 3.8 Device-Specific Configurations for more information.

3.1 Writing a Flash or EPROM Block

All flash writes are performed with the Block Write command. The size of the block can be 1-to-256 bytes and is user-defined during the Block Write sequence. The 8-bit data length code is formatted as shown below.

Table 3.2. Block Write Length Formatting

8-bit Data Length Code	Number of Bytes
1 - 255	block size equal to length code
0	256

Figure 3.2 Basic Flash Block Write Sequence on page 16 shows the basic Block Write sequence and assumes the PI has already been initialized by the sequence in Figure 3.1 PI Initialization Sequence on page 15.

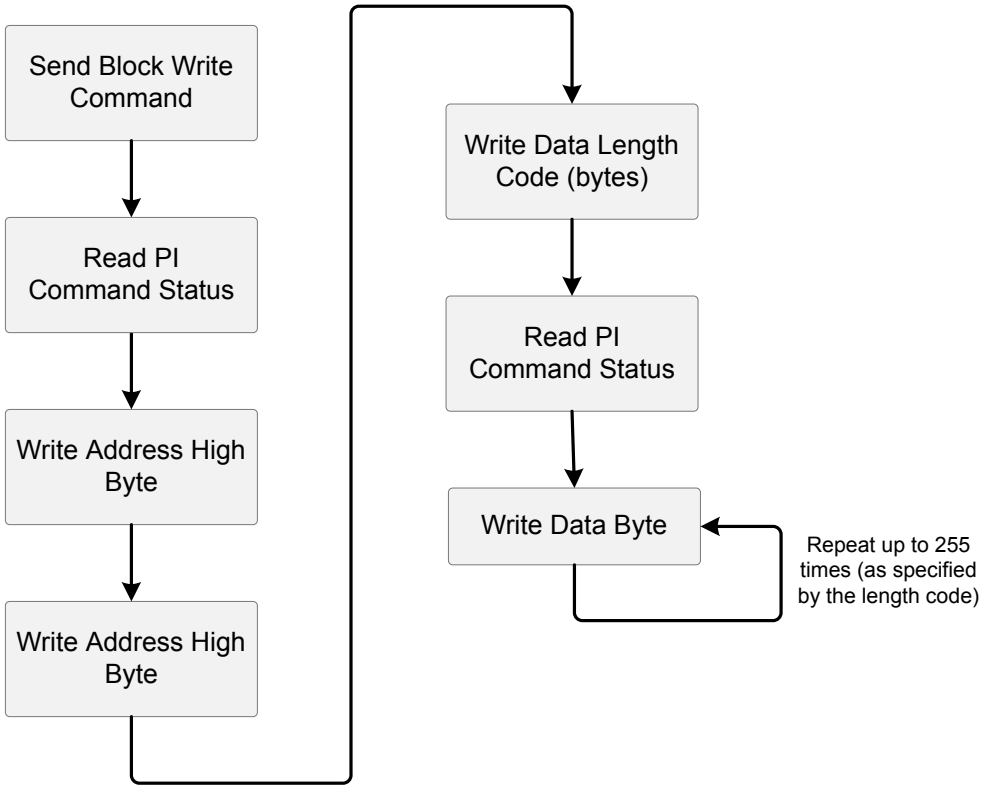


Figure 3.2. Basic Flash Block Write Sequence

To program a flash block:

1. Perform an Address Write with a value of FPDAT.
2. Perform a Data Write with the Block Write command.
3. Poll on InBusy using Address Read until the bit clears.
4. Poll on OutReady using Address Read until the bit set.
5. Perform a Data Read instruction. A value of 0x0D is okay.
6. Perform a Data Write with the high byte of the address.
7. Poll on InBusy using Address Read until the bit clears.
8. Perform a Data Write with the low byte of the address.
9. Poll on InBusy using Address Read until the bit clears.
10. Perform a Data Write with the length.
11. Poll on InBusy using Address Read until the bit clears.
12. Perform a Data Write with the data. This will write the data to the flash. Repeat steps 11 and 12 for each byte specified by the length field.

13. Poll on OutReady using Address Read until the bit set.
14. Perform a Data Read instruction. A value of 0x0D is okay.

To write to an EPROM block:

1. Write 0x04 to the FPCTL register.
2. Write 0x40 to EPCTL.
3. Write 0x58 to EPCTL.
4. Write the high byte of the address to EPADDRH.
5. Write the low byte of the address to address EPADDRL.
6. Perform an Address Write with a value of EPDAT.
7. Turn on VPP.
8. Wait for the VPP settling time.
9. Write the data to the device using a Data Write.
10. Perform Address Read instructions until the value returned is not 0x80 and the EPROM is no longer busy.
11. Repeat steps 9 and 10 until all bytes are written.
12. Turn off VPP. Note that VPP can only be applied for a maximum lifetime amount, and this value is specified in the device data sheet.
13. Write 0x40 to EPCTL.
14. Write 0x00 to EPCTL.
15. Write 0x02 to FPCTL.
16. Write 0x04 to FPCTL.
17. Write 0x01 to FPCTL.

3.2 Reading a Flash or EPROM Block

All flash reads are performed with the Block Read command. The size of the block can be 1-to-256 bytes and is user-defined in the Block Read sequence. The 8-bit data length code is formatted as shown in the table below.

Table 3.3. Block Read Length Formatting

8-bit Data Length Code	Number of Bytes
1 - 255	block size equal to length code
0	256

The Block Read sequence is shown in Figure 3.3 Basic Flash Block Read Sequence on page 18. The flow diagram and sequences described assume the PI has already been initialized by the sequence in Figure 3.1 PI Initialization Sequence on page 15.

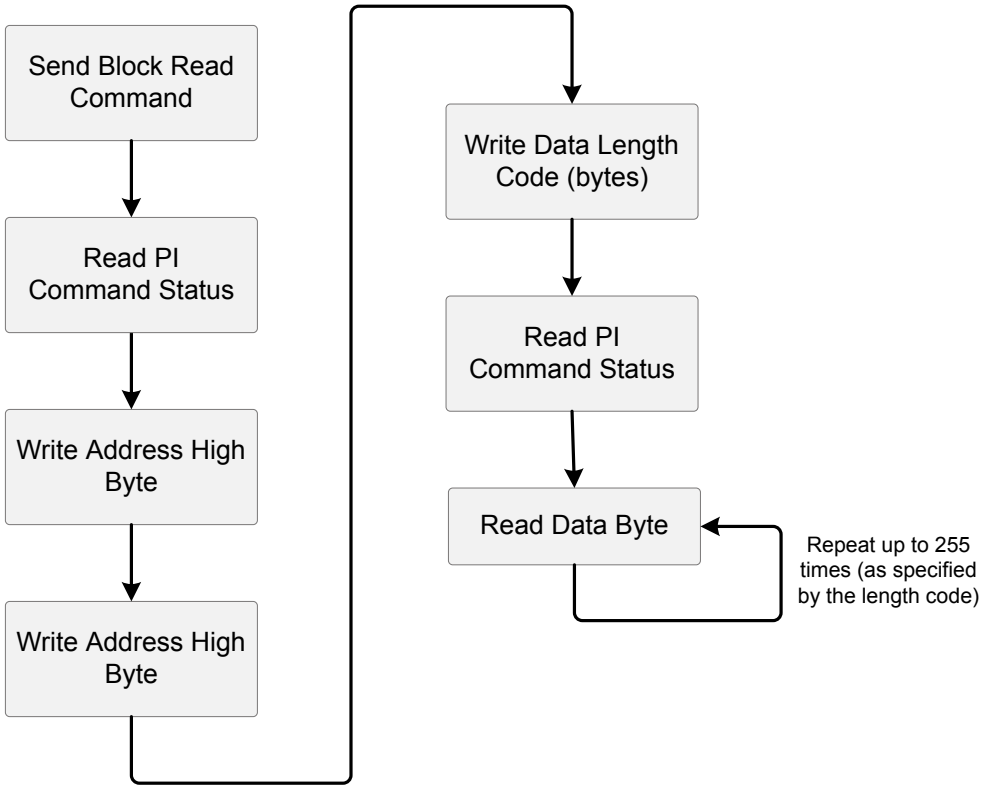


Figure 3.3. Basic Flash Block Read Sequence

To read a flash block:

1. Perform an Address Write with a value of FPDAT.
2. Perform a Data Write with the Block Read command.
3. Poll on InBusy using Address Read until the bit clears.
4. Poll on OutReady using Address Read until the bit set.
5. Perform a Data Read instruction. A value of 0x0D is okay.
6. Perform a Data Write with the high byte of the address.
7. Poll on InBusy using Address Read until the bit clears.
8. Perform a Data Write with the low byte of the address.
9. Poll on InBusy using Address Read until the bit clears.
10. Perform a Data Write with the length.
11. Poll on InBusy using Address Read until the bit clears.
12. Poll on OutReady using Address Read until the bit set.

13. Perform a Data Read instruction. This will read the data from the flash. Repeat step 12 and 13 for each byte specified by the length field.

To read an EPROM block:

1. Write 0x04 to the FPCTL register.
2. Write 0x00 to EPCTL.
3. Write 0x58 to EPCTL.
4. Write the high byte of the address to EPADDRH.
5. Write the low byte of the address to address EPADDRL.
6. Perform an Address Write with a value of EPDAT.
7. Perform Address Read instructions until the value returned is not 0x80 and the EPROM is no longer busy.
8. Read the byte using the Data Read instruction.
9. Repeat steps 7 and 8 until all bytes are read.
10. Write 0x40 to EPCTL.
11. Write 0x00 to EPCTL.
12. Write 0x02 to FPCTL.
13. Write 0x04 to FPCTL.
14. Write 0x01 to FPCTL.

To perform additional status checks during the EPROM read operation, insert these steps in front of step 7 in the previous sequence:

1. Perform an Address Write operation with a value of EPSTAT.
2. Perform a Data Read operation and check the bits of the EPSTAT register.
3. Perform an Address Write operation with a value of EPDAT.

In addition, insert these steps in front of step 8 in the EPROM read sequence:

1. Perform an Address Write operation with a value of EPSTAT.
2. Perform a Data Read operation and check the ERROR bit in the EPSTAT register.
3. Perform an Address Write operation with a value of EPDAT.

3.3 Erasing a Flash Page

Flash memory is erased in pages using the Page Erase command. The size of the page varies depending on the device family. The page to be erased is selected in the Page Erase procedure shown in the figure below.

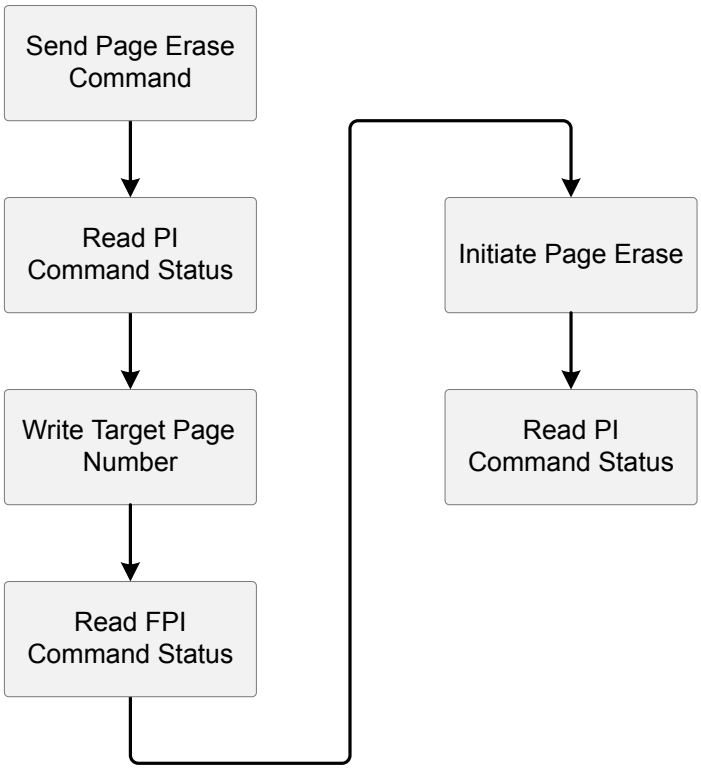


Figure 3.4. Basic Page Erase Sequence

This flow diagram assumes the PI has already been initialized by the sequence in Figure 3.1 PI Initialization Sequence on page 15.

To erase a page of flash:

1. Perform an Address Write with a value of FPDAT.
2. Perform a Data Write with the Page Erase command.
3. Poll on InBusy using Address Read until the bit clears.
4. Poll on OutReady using Address Read until the bit set.
5. Perform a Data Read instruction. A value of 0x0D is okay.
6. Perform a Data Write with the page number.
7. Poll on InBusy using Address Read until the bit clears.
8. Poll on OutReady using Address Read until the bit clears.
9. Perform a Data Read instruction. A value of 0x0D is okay.
10. Perform a Data Write with the a value of 0x00.
11. Poll on InBusy using Address Read until the bit clears.
12. Poll on OutReady using Address Read until the bit set.
13. Perform a Data Read instruction. A value of 0x0D is okay.

3.4 Erasing the Device

The Device Erase command performs a mass erase of the flash memory and erases all pages in the device. A three-byte arming sequence must be written to the PI to enable this procedure:

1. 0xDE
2. 0xAD
3. 0xA5

The erase operation executes following the last byte of the arming sequence. The Device Erase programming sequence is shown in the figure below. This flow diagram assumes the PI has already been initialized by the sequence in Figure 3.1 PI Initialization Sequence on page 15.

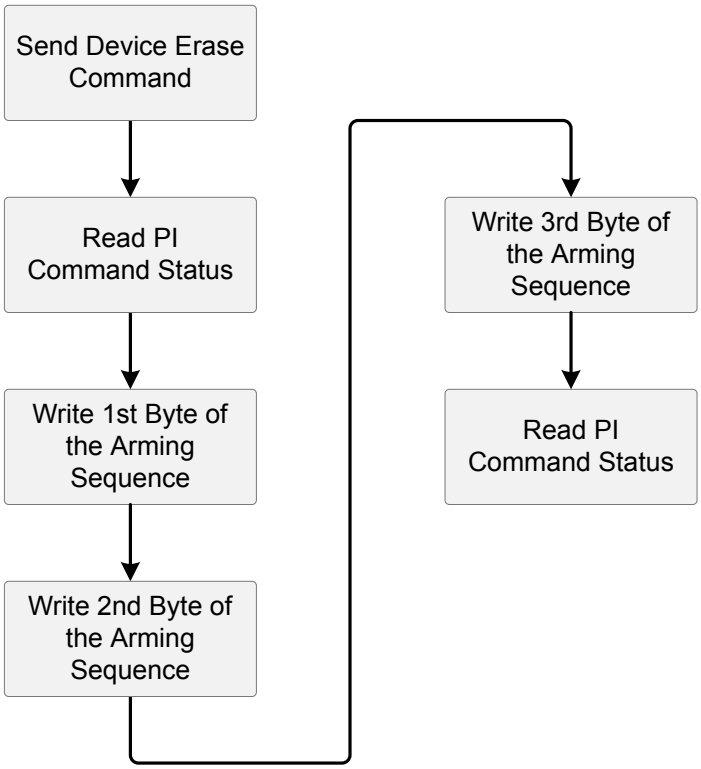


Figure 3.5. Basic Device Erase Sequence

To erase a flash device:

1. Perform an Address Write with a value of FPDAT.
2. Perform a Data Write with the Device Erase command.
3. Poll on InBusy using Address Read until the bit clears.
4. Poll on OutReady using Address Read until the bit set.
5. Perform a Data Read instruction. A value of 0x0D is okay.
6. Perform a Data Write with a value of 0xDE.
7. Poll on InBusy using Address Read until the bit clears.
8. Perform a Data Write with a value of 0xAD.
9. Poll on InBusy using Address Read until the bit clears.
10. Perform a Data Write with a value of 0xA5.
11. Poll on InBusy using Address Read until the bit clears.
12. Poll on OutReady using Address Read until the bit set.
13. Perform a Data Read instruction. A value of 0x0D is okay.

3.5 Writing to an SFR

To write to an SFR on a device that does not have SFR paging:

1. Write the SFR address to the device using the Address Write instruction.
2. Write the SFR value to the device using the Data Write instruction.

For devices with SFR paging, direct writes through the PI using the Direct Write command are recommended to ensure the SFR Page is managed properly.

To write to an SFR from a device with SFR paging:

1. Perform an Address Write with a value of FPDAT.
2. Write the Direct Write command (0x0A) using a Data Write.
3. Poll InBusy until the data is processed by the PI.
4. Poll OutReady it sets to 1.
5. Perform a Data Read to ensure a return value of 0x0D (no errors).
6. Perform a Data Write with a value of the SFR address.
7. Poll InBusy until the data is processed by the PI.
8. Perform a Data Write with a value of 0x01.
9. Poll InBusy until the data is processed by the PI.
10. Perform a Data Write with the new SFR value.
11. Poll InBusy until the data is processed by the PI.

3.6 Reading from an SFR

To read from an SFR on a device that does not have SFR paging:

1. Write the SFR address to the device using the Address Write instruction.
2. Read the SFR value from the device using the Data Read instruction.

For devices with SFR paging, direct reads through the PI using the Direct Read command are recommended to ensure the SFR Page is managed properly.

To read an SFR from a device with SFR paging:

1. Perform an Address Write with a value of FPDAT.
2. Write the Direct Read command (0x09) using a Data Write.
3. Poll InBusy until the data is processed by the PI.
4. Poll OutReady until it sets to 1.
5. Perform a Data Read to ensure a return value of 0x0D (no errors).
6. Perform a Data Write with a value of the SFR address.
7. Poll InBusy until the data is processed by the PI.
8. Perform a Data Write with a value of 0x01.
9. Poll InBusy until the data is processed by the PI.
10. Poll OutReady until it sets to 0.
11. Read the SFR value from the device using the Data Read instruction.

3.7 Reading and Writing Flash on devices with More than 64 KB of Memory

Reading and writing flash memory on devices with more than 64 KB of code space requires translating the linear target address into a 16-bit banked address. Note that erasing a page does not require such a translation because the page erase operation accepts page numbers, not byte addresses.

Prior to starting the actual flash read or write, the PSBANK and COBANK bits must be set appropriately for the bank to be targeted. Ideally this operation should occur prior to writing the address bytes. The mapping of linear address to banked address, along with the correct value of PSBANK, are shown in the table below.

Table 3.4. Translating Linear Addresses to Banked Addresses

Linear Address Range (17 bits)	Banked Address Range (16 bits)	PSBANK Register Value
0x00000–0x0FFFF	0x0000–0xFFFF	0x11 (default)
0x10000–0x17FFF	0x8000–0xFFFF[1]	0x22
0x18000–0x1FFFF	0x8000–0xFFFF	0x33

Note:
1. Set b15 to translate linear address into banked address for linear range 0x10000 – 0x17FF.

PSBANK should be restored to its default value after the flash operation has completed successfully. The PSBANK and COBANK registers are located at different addresses on each device with banked addressing.

On ‘F58x devices, PSBANK is located at SFR address 0xF5. It appears on all SFR pages and can be accessed with the WriteSFR() operation. On ‘F96x devices, PSBANK is located at SFR address 0x84 and appears on all SFR pages.

3.8 Device-Specific Configurations

Some devices require configuration prior to the start of write or erase operations. The required operations include:

1. Setting up the flash timing register.
2. Setting up the voltage regulator (programming it to a high setting).
3. Enabling the VDD monitor and enabling the VDD monitor as a reset source.

Optionally, the system clock of the device can be increased. This will slightly reduce flash write times but will have a very favorable impact on flash read times.

The following table outlines the device-specific programming information, including the device ID values for each family and the location of the FPDAT register.

The procedures for configuring the various options for specific device families are given in Table 11 using the common functions described in 4.1 Common Functions. If a table entry is blank, that means that particular device does not require any special configuration. In general, the sequences should be executed in the order that they are listed in the table from left to right.
Note: The WriteDirect(...) function calls are used on devices with SFR Paging to ensure the SFR Page is managed correctly.

Table 3.5. Device-Specific Programming Information

Family Name	Family (DEVID)	FPDAT Address	Page Size
‘F30x	0x04	0xB4	512
‘F31x	0x08	0xB4	512
‘F32x	0x09	0xB4	512
‘F326/7	0x0D	0xB4	512
‘F33x	0x0A	0xB4	512
‘F336/7	0x14	0xB4	512
‘F34x	0x0F	0xAD	512
‘F35x	0x0B	0xB4	512
‘F36x	0x12	0xB4	1024
‘F38x	0x28	0xAD	512
‘F39x/’F37x	0x2B	0xB4	512
‘F41x	0x0C	0xB4	512
‘F50x/’F51x	0x1C	0xB4	512
‘F52x/’F53x	0x11	0xB4	512
‘F54x	0x22	0xB4	512
’F55x/’F56x/’F57x	0x22	0xB4	512
‘F58x/’F59x	0x20	0xB4	512
‘F70x/’F71x	0x1E	0xB4	512
‘F80x/’F81x/’F82x/’F83x	0x23	0xB4	512
'F85x/'F86x	0x30	0xB4	512
‘F90x/’F91x	0x1F	0xB4	512
‘F92x/’F93x	0x16	0xB4	1024
‘F96x	0x2A	0xB4	1024
‘F99x	0x25	0xB4	512
‘T60x	0x10	0xB4	512

Family Name	Family (DEVID)	FPDAT Address	Page Size
‘T606	0x1B	0xB4	512
‘T61x	0x13	0xB4	512
‘T62x/’T32x	0x18	0xAD	512
‘T622/’T623/‘T326/’T327	0x19	0xAD	512
‘T63x	0x17	0xB4	512
EFM8BB1	0x30	0xB4	512
EFM8BB2	0x32	0xB4	512
EFM8BB3	0x34	0xB4	512
EFM8LB1	0x34	0xB4	512
EFM8SB1	0x25	0xB4	512
EFM8SB2	0x16	0xB4	1024
EFM8UB1	0x32	0xB4	512
EFM8UB2	0x28	0xAD	512

Table 3.6. Device-Specific Programming Sequences

Family Name	Flash Timing	Voltage Regulator Initialization	VDD Monitor Initialization	Oscillator Initialization
‘F30x				WriteSFR(0xB2, 0x07)
‘F31x				WriteDirect(0xEF, 0x00) WriteDirect(0xB2, 0x83)
‘F32x				WriteSFR(0xB2, 0x83)
‘F326/7				WriteSFR(0xB2, 0x83)
‘F33x				WriteSFR(0xB2, 0x83)
‘F336/7				WriteSFR(0xB2, 0x83)
‘F34x	WriteSFR(0xB6, 0x90)		WriteSFR(0xFF, 0x80) WriteSFR(0xEF, 0x02)	WriteSFR(0xB2, 0x83)
‘F35x	WriteSFR(0xB6, 0x10)			WriteSFR(0xB2, 0x83)
‘F36x	WriteDirect (0xA7, 0x0F) WriteDirect (0x84, 0x00) WriteDirect (0xA7, 0x00) WriteDirect (0xB6, 0x00)			WriteDirect(0xA7,0x0F) WriteDirect(0xB7, 0x83) WriteDirect(0xA7, 0x00)
‘F38x	WriteSFR(0xB6, 0x90)		WriteSFR(0xFF, 0x80) WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x03)
‘F39x/’F37x			WriteSFR(0xFF, 0x80) WriteSFR(0xEF, 0x02)	WriteSFR(0xB2, 0x83)
‘F41x	WriteSFR(0xB6, 0x10)	WriteSFR(0xC9, 0x10)	WriteSFR(0xFF, 0xA0) WriteSFR(0xEF, 0x02)	WriteSFR(0xB2, 0x87)

Family Name	Flash Timing	Voltage Regulator Initialization	VDD Monitor Initialization	Oscillator Initialization
‘F50x/’F51x			WriteDirect(0xFF, 0xA0) Delay 100 µs WriteDirect(0xEF, 0x02)	WriteDirect(0xA7, 0x0F) WriteDirect(0xA1, 0xC7) WriteDirect(0x8F, 0x00) WriteDirect(0xA7, 0x00)
‘F52x/’F53x			WriteSFR(0xFF, 0xA0)	WriteSFR(0xB2, 0x87)
‘F54x			WriteDirect(0xFF, 0xA0) Delay 100 µs WriteDirect(0xEF, 0x02)	WriteDirect(0xA7, 0x0F) WriteDirect(0xA1, 0xC7) WriteDirect(0x8F, 0x00) WriteDirect(0xA7, 0x00)
’F55x/’F56x/’F57x			WriteDirect(0xFF, 0xA0) Delay 100 µs WriteDirect(0xEF, 0x02)	WriteDirect(0xA7, 0x0F) WriteDirect(0xA1, 0xC7) WriteDirect(0x8F, 0x00) WriteDirect(0xA7, 0x00)
‘F58x/’F59x	WriteDirect(0xB6, 0x02)		WriteDirect(0xFF, 0xA0) Delay 100 µs WriteDirect(0xEF, 0x02)	WriteDirect(0xA7, 0x0F) WriteDirect(0xA1, 0xC7) WriteDirect(0xA7, 0x00)
‘F70x/’F71x				WriteDirect(0xA7, 0x0F) WriteDirect(0xA9, 0x83) WriteDirect(0xBD, 0x00) WriteDirect(0xA7, 0x00)
‘F80x/’F81x/’F82x/’F83x				WriteSFR(0xB2, 0x83)
'F85x/'F86x			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
‘F90x/’F91x				WriteDirect(0xA7, 0x00) WriteDirect(0xB2, 0x8F) WriteDirect(0xA9, 0x00)
‘F92x/’F93x				WriteDirect(0xA7, 0x00) WriteDirect(0xB2, 0x8F) WriteDirect(0xA9, 0x00)
‘F96x	WriteDirect(0xA7, 0x0F) WriteDirect(0xB6, 0x00) WriteDirect(0xA7, 0x00)		WriteDirect(0xFF, 0x88) WriteDirect(0xEF, 0x02)	WriteDirect(0xA7, 0x00) WriteDirect(0xA9, 0x04)
‘F99x	WriteDirect(0xB6, 0x40)		WriteDirect(0xFF, 0x80) WriteDirect(0xEF, 0x02)	WriteDirect(0xA9, 0x04)
‘T60x				WriteSFR(0xB2, 0x07)

Family Name	Flash Timing	Voltage Regulator Initialization	VDD Monitor Initialization	Oscillator Initialization
‘T606				WriteSFR(0xB2, 0x07)
‘T61x				WriteSFR(0xB2, 0x83)
‘T62x/’T32x				WriteSFR(0xB2, 0x83)
‘T622/’T623/‘T326/’T327				WriteSFR(0xB2, 0x83)
‘T63x				WriteDirect(0xB2, 0x83)
EFM8BB1			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
EFM8BB2			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
EFM8BB3			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
EFM8LB1			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
EFM8SB1	WriteDirect(0xB6, 0x40)		WriteDirect(0xFF, 0x80) WriteDirect(0xEF, 0x02)	WriteDirect(0xA9, 0x04)
EFM8SB2				WriteDirect(0xA7, 0x00) WriteDirect(0xB2, 0x8F) WriteDirect(0xA9, 0x00)
EFM8UB1			WriteSFR(0xFF, 0x80) Delay 5 µs WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x00)
EFM8UB2	WriteSFR(0xB6, 0x90)		WriteSFR(0xFF, 0x80) WriteSFR(0xEF, 0x02)	WriteSFR(0xA9, 0x03)

4. Software Example

The software example included with this application note is written for a C8051F38x device acting as the programmer. The software can be ported to any other Silicon Labs C8051Fxxx device with some modification. The interconnection diagram used with the example software is shown in the figure below. This connection diagram assumes that the C2CK and/or C2D pins on the target device are not used by the target application.

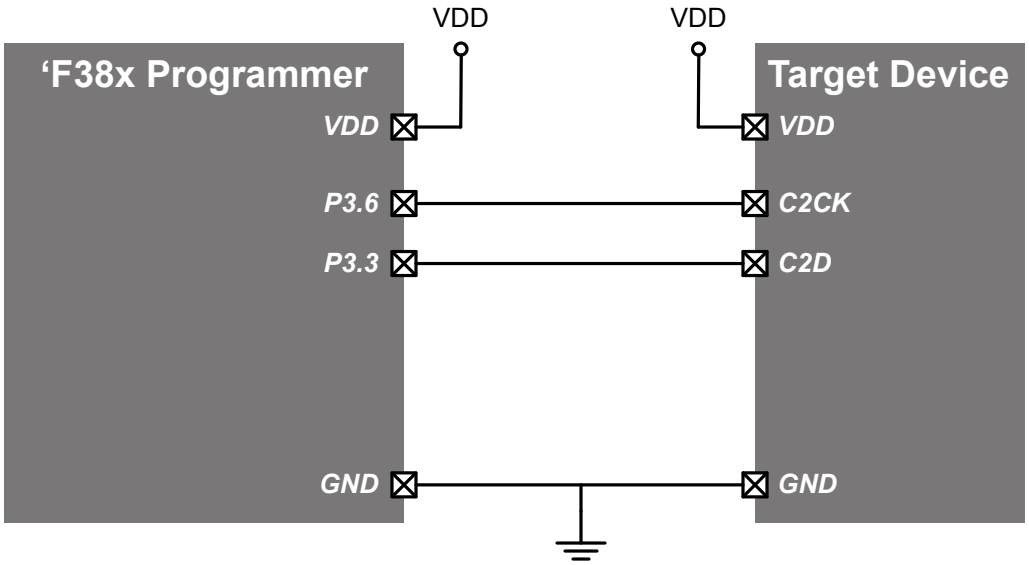


Figure 4.1. C2 Programmer Connection Diagram

If the C2 programming is to be performed on a target device installed in the user application, C2 isolation circuitry may be necessary. See application note "AN124: Pin Sharing Techniques for the C2 Interface" on the Silicon Labs website: http://www.silabs.com.

Table 4.1. C8051F38x Programming Pins

USB Debug Adapter Connector Pin	Signal	C8051F38x (Target Board) Pin
1	+3 V	P3.0 (set high)
2	GND	P3.1 (set low)
3	GND	P3.2 (set low)
4	C2D	P3.3
5	RST pin sharing	P3.4
6	C2D pin sharing	P3.5
7	C2CK	P3.6
8	VPP	P3.7
9	GND	N/A (must cut the TB stake header)
10	VBUS	no connect

4.1 Common Functions

This section outlines common steps grouped into functions in the firmware device programmer example.

4.1.1 WriteSFR

The `WriteSFR(addr,data)` function directly writes an SFR and consists of the following steps:

1. `AddressWrite(addr)`
2. `DataWrite(data)`

4.1.2 ReadSFR

The `ReadSFR(addr)` function directly reads an SFR and consists of the following steps:

1. `AddressWrite(addr)`
2. return `DataRead()`

4.1.3 WriteCommand

The `WriteCommand(command)` function writes a command to the PI. It consists of the following steps:

1. `DataWrite(command)`
2. `Poll_InBusy()`

4.1.4 ReadData

The `ReadData()` function reads data from the PI and has the following steps:

1. `Poll_OutReady()`
2. return `DataRead()`

4.1.5 WriteDirect

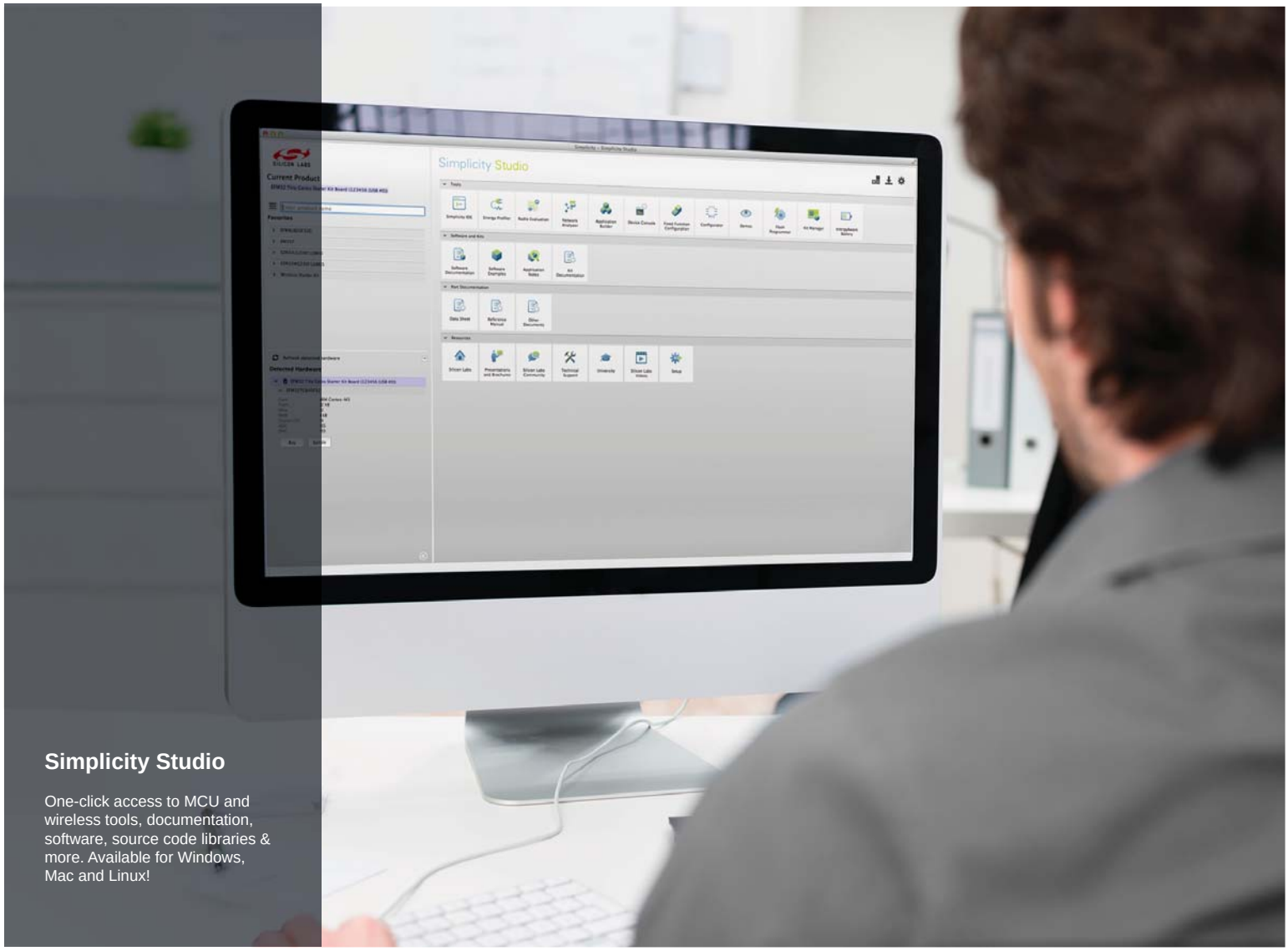
The `WriteDirect(addr,data)` function allows writes to SFRs on devices that have SFR Paging and ensures that the SFR page is managed correctly.

1. `AddressWrite(FPDAT)`
2. `WriteCommand(0x0A) // Direct write`
3. `ReadData() // 0x0D indicates success, all other return values are errors`
4. `WriteCommand(addr)`
5. `WriteCommand(0x01)`
6. `WriteCommand(data)`

4.1.6 ReadDirect

The `ReadDirect(addr)` function allows reads from SFRs on devices that have SFR Paging and ensures that the SFR page is managed correctly.

1. `AddressWrite(FPDAT)`
2. `WriteCommand(0x09) // Direct read`
3. `ReadData() // 0x0D indicates success, all other return values are errors`
4. `WriteCommand(addr)`
5. `WriteCommand(0x01)`
6. return `ReadData()`



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com