# CodeGuard™ Intermediate Security

This section of the manual contains the following major topics:


1.0 Introduction ..... 2
2.0 Control Registers ..... 2
3.0 Code Segment Organization ..... 3
4.0 Security Privileges and Rules ..... 8
5.0 Dual Boot Security ..... 14
6.0 Flash OTP with ISCP Write Inhibit ..... 18
7.0 Immutable Secure Boot ..... 19
8.0 Design Tips ..... 21
9.0 Related Documents ..... 22
10.0 Revision History ..... 23

## 1.0 INTRODUCTION

CodeGuard™ Intermediate Security provides immutability and access control for the intellectual property stored in Flash program memory. The features can be configured for a wide range of system security use cases, including multiple code authors on a single device, secure immutable boot and secure field updates. These features also provide additional security enhancements in devices that incorporate dual boot program memory.

Depending on the type of device, Flash program memory can be organized into multiple (up to four) code space segments. Each of these segments has an implied security privilege level and system function. Any operation of the system that has the potential to allow discovery of code or data contents is restricted based on the segment from which it originated or the segment it targets. These include:

- Programming, erasing or verifying operations
- Reads and writes of code space
- Program Flow Changes into a secured segment from outside of that segment
- Interrupt vectors into a secured segment

CodeGuard Intermediate Security features apply only to the program memory space. Data memory is not restricted and may be freely accessed from any Code Segment.

## 2.0 CONTROL REGISTERS

The features of program code security are controlled entirely at device start-up by the device Configuration bits. The locations of these bits are a function of the device family. For most dsPIC33 and PIC24 devices, the Configuration bits are located in the FSEC and FBSLIM Flash Configuration registers. For detailed information on a particular device family, refer to the specific device data sheet.

The relevant Configuration bits discussed in this chapter are:

- CSS<2:0> (Configuration Segment Security Configuration)
- CWRP (Configuration Segment Write-Protect)
- BSEN (Boot Segment Enable)
- BSS<1:0> (Boot Segment Security Configuration)
- BWRP (Boot Segment Write-Protect)
- GSS<1:0> (General Segment Security Configuration)
- GWRP (General Segment Write-Protect)
- AIVTDIS (Disable Alternate IVT)
- BSLIM<12:0> (Boot Segment Limit Value)

For devices with dual boot program memory, the BTMOD<1:0> bits (generally found in the FBOOT Configuration register) also modify the behavior of CodeGuard security features depending on the Boot mode selected.

## 3.0 CODE SEGMENT ORGANIZATION

Flash program memory is divided into several segments, each having their own Code Protection (CP) and write protection (WRP) settings. Optionally, a Boot Segment (BS) can also be defined and partitioned from the General Segment (GS). The multiple segment approach allows restriction between segments for all types of access and operation, which enables a chain of trust. When operating in dual boot mode, the code segments also have restrictions between the Active and Inactive partitions.

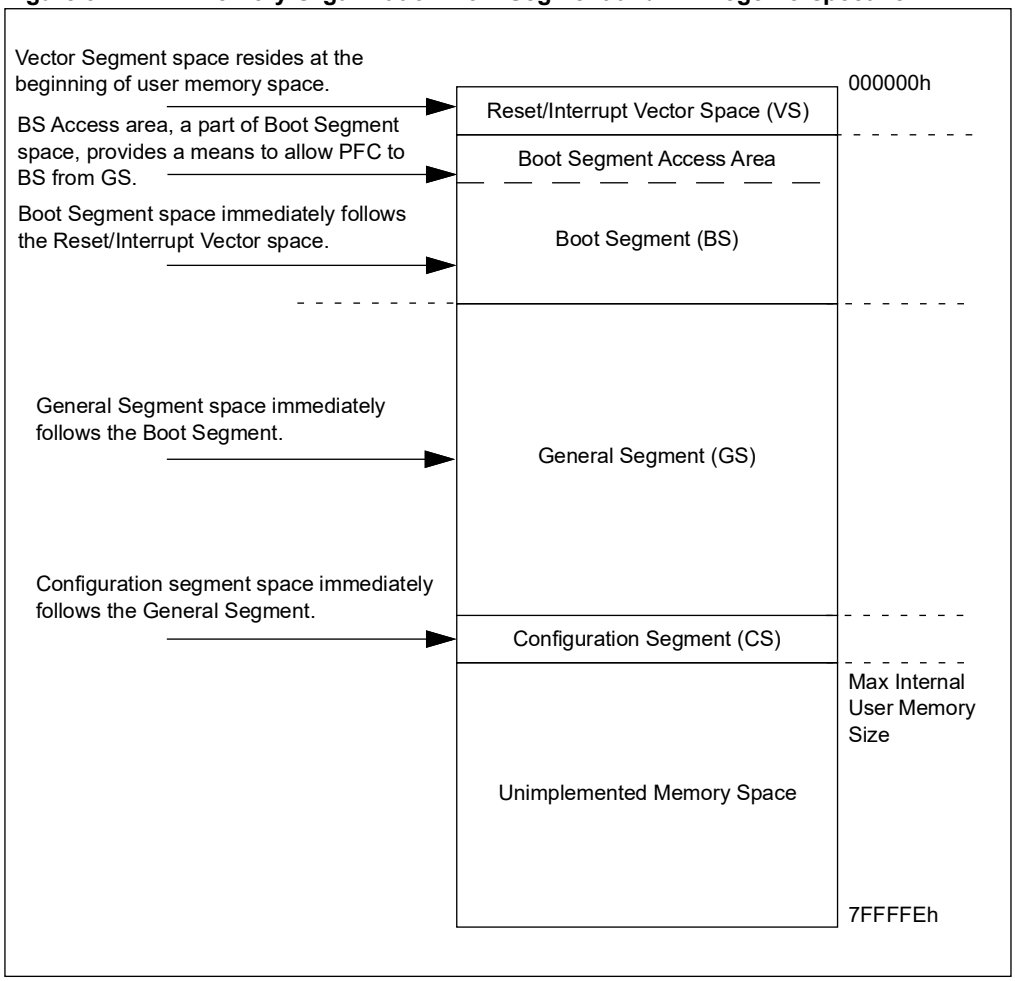
### 3.1 Code Protection Bits

The individual Code Protection features for each segment are controlled by Flash configuration bits located at the end of user program memory. These include Code Protection (CP) bits that define the security level (BSS, GSS or CSS) and Write Protection (WRP) bits, which block all write operations to a particular segment. Like all other Flash Configuration bits, the configuration bits are set (= 1) by default and programmed by clearing (= 0) the individual bits.

Unlike other Flash Configuration bits, Code Protection bits can only be programmed. Attempting to erase a Code Protection bit (from '0' to '1') is not allowed. To erase a Code Protection bit, a chip erase, inactive partition erase or page erase that targets the configuration segment page (when permitted by the existing Code Protection values) must be used to erase all Code Protection bits and delete Code Protection.

During the development stage, enabling the Code Protection can inhibit the entry into the debug mode. Therefore, Code Protection must be disabled before entering the debug mode.

**Figure 3-1: Memory Organization from Segment and Privilege Perspective**



## 3.2 Boot Segment (BS)

The Boot Segment (BS) provides a highly secure code space for bootloader code or other intellectual property that needs to be protected from other code executing on the same device or an external interface. It is important that the boot code within the Boot Segment be immutable since interfering with this code can expose the system to attacks that can modify or eliminate the chain of trust. Therefore, the Boot Segment has a higher security privilege compared to the other segments, and it also has access to the other segments. With Boot Segment write-protection not enabled, Boot Segment can rewrite its own locations, enabling it to store and update ephemeral keys.

### 3.2.1 ALLOCATION OF THE BS

The Boot Segment is created by programming the BSEN Configuration bit (= 0) and defining a greater-than-zero page size with the BSLIMx Configuration bits (FBSLIM<12:0>). The value of the BSLIMx bits is the inverse of the number of Boot Segment pages. This is done to prevent shrinking the size of the current Boot Segment by clearing any of the bits, which would effectively place the existing boot code in the lower security general segment.

Like the Code Protection bits, the BSLIMx bits are program only and are also “write-once” bits. If the value loaded from the Flash during the Reset sequence is not erased (all ‘1’s), then programming of the FBSLIM is prohibited; an attempt to do so will fail and have no effect.

### 3.2.2 SELECTING THE SECURITY LEVEL

The security level of the Boot Segment is set using the BWRP and BSS<1:0> Configuration bits, and is used to restrict access to the Boot Segment from the other segments. Two security options (standard and high), as well as no protection, are supported. See **Section 4.1 “Rules Concerning Program Flow”** for more information.

**Figure 3-2: Boot Segment Allocation**

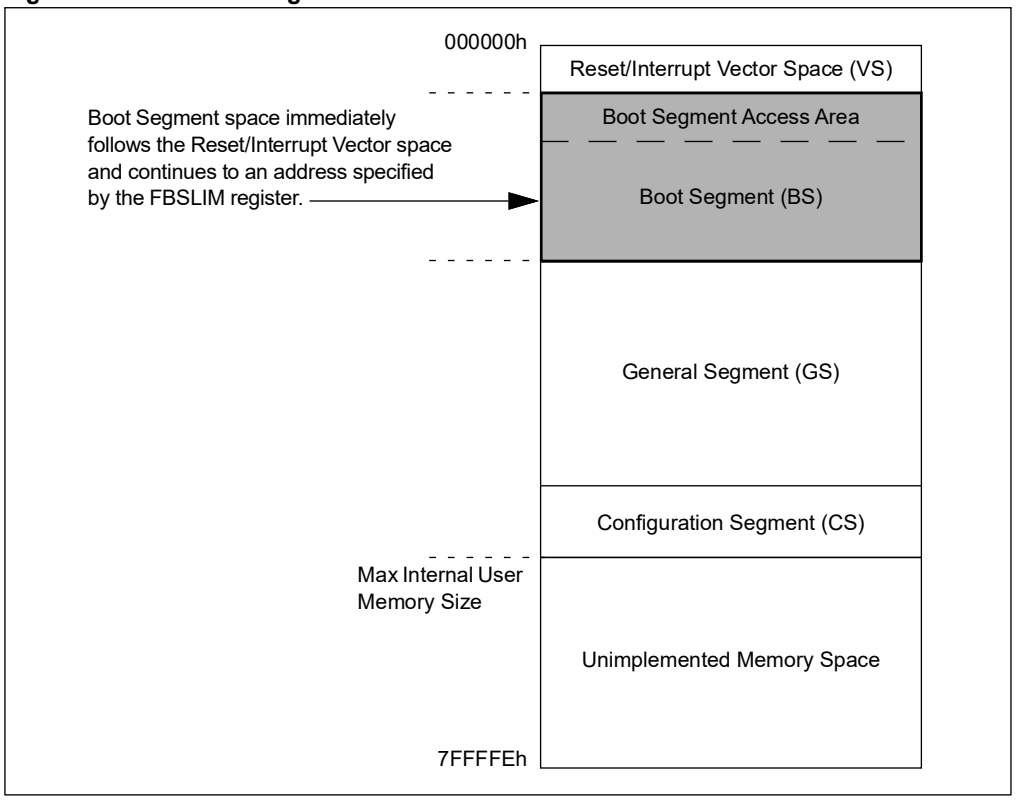


**Table 3-1: Boot Segment Configurations**

| BSEN | BSS<1:0> | Security Level |
|---|---|---|
| 1 | xx | No Boot Segment Defined |
| 0 | 11 | No Security (other than optional write-protect) |
| 0 | 10 | Standard Security |
| 0 | 0x | High Security |

## 3.3 General Segment (GS)

The General Segment (GS) has a lower level of security privilege than the Boot Segment. Typically, the General Segment contains the majority of the application code. The General Segment begins at the page boundary after the Vector Segment or at the page boundary after the Boot Segment if the Boot Segment is implemented.

### 3.3.1 SECURITY LEVEL OF THE GS

Depending on the device, there are up to three levels of security to choose from for the General Segment. Configuration bits, GSS<1:0>, determine the level of protection for this segment (see Table 3-2). Two security options (standard and high), as well as no protection, are supported. See **Section 4.1 "Rules Concerning Program Flow"** for more information.

### 3.3.2 WRITE PROTECTION OF THE GS

The General Segment can be write-protected by programming the GWRP Configuration bit, similar to the write protection of the Boot Segment. Write protection is disabled when the bit is unprogrammed (= 1). Programming the bit enables write protection for the General Segment.

**Figure 3-3: General Segment Allocation**

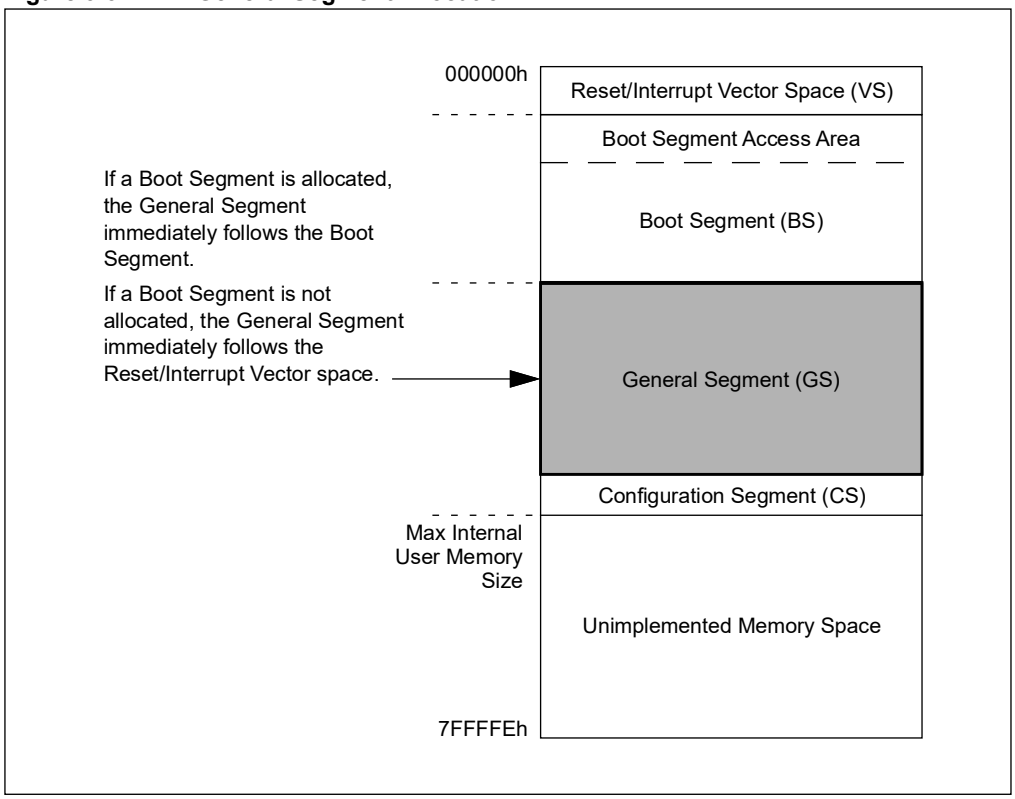


**Table 3-2: General Segment Configurations**

| GSS<1:0> | Security Level |
|---|---|
| 11 | No Security (other than optional write-protect) |
| 10 | Standard Security |
| 0x | High Security |

## 3.4 Configuration Segment (CS)

The Configuration Segment (CS) is located in the last page of implemented program memory. The Configuration Segment holds all configuration data in the Flash program memory, which is automatically read and loaded into the device configuration registers during the Reset sequence. The Configuration Segment does not contain independently executable code, so it has no special privilege level as compared to Boot Segment or General Segment. However, it does implement security and Write Protection that are independent from General Segment or Boot Segment.

### 3.4.1 SECURITY LEVEL OF THE CS

The security level is set to one of four levels with the CSS<2:0> configuration bits (see Table 3-3). The Configuration Segment features an additional level of security (Enhanced) to provide more flexibility in controlling page erase operations, independently of other program memory accesses, because the Configuration Segment contains data that is critical to the security and write protection of the device. See **Section 4.1 "Rules Concerning Program Flow"** for more information.

### 3.4.2 WRITE PROTECTION OF THE CS

The Configuration Segment can be write-protected by programming the CWRP Configuration bit, similar to write-protecting the Boot Segment. Write protection is disabled when the bit is unprogrammed (= 1). Programming the bit enables write protection for the Configuration Segment.

**Figure 3-4: Configuration Segment Allocation**

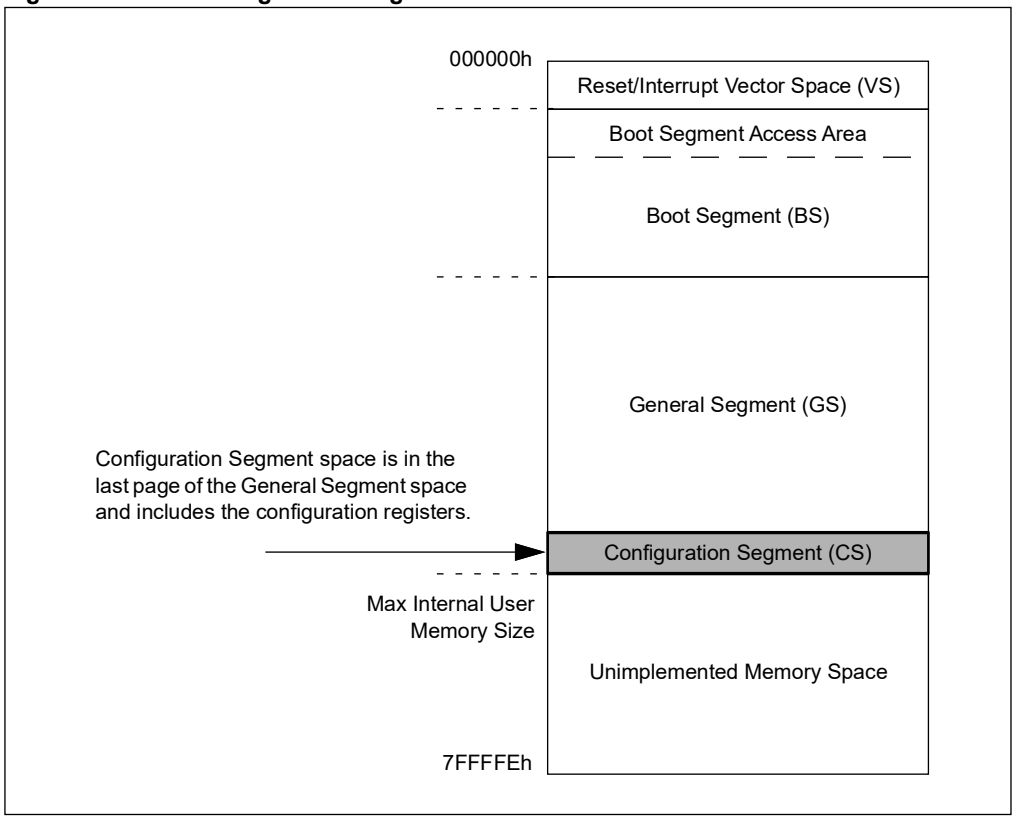


**Table 3-3: Configuration Segment Configurations**

| CSS<2:0> | Security Level |
|---|---|
| 111 | No Security (other than optional write-protect) |
| 110 | Standard Security |
| 10x | Enhanced Security |
| 0xx | High Security |

## 3.5 Vector Segment (VS)

The Vector Segment (VS) contains Reset, Trap and Interrupt Service Routine (ISR) vectors. For PIC24 devices, this is the first 256 instruction words of Flash memory; for dsPIC33 devices, this is the first 512 words. If a Boot Segment is defined, an optional Alternate Interrupt Vector Table (AIVT) may be used. Like the Configuration Segment, the Vector Segment does not independently execute code, so it has no special privilege level in comparison to other segments.

Protection of the Vector Segment depends upon the state of the Boot Segment or General Segment security settings. If a Boot Segment is defined, the Vector Segment will assume the security level and Write Protection level of the BS. If no Boot Segment is defined, then the Vector Segment will assume the security level and Write Protection level of the General Segment.

**Note:** The VS may be modified when programmed for high security. This allows programming the IVT and AIVT during a field update. See **Section 4.0 "Security Privileges and Rules"** for access conditions.

### 3.5.1 ALTERNATE INTERRUPT VECTOR TABLE (AIVT)

Devices with CodeGuard Intermediate Security have the option to implement a second IVT, or Alternate IVT, inside the Boot Segment. The AIVT is enabled by programming the AIVTDIS Configuration bit. To support the AIVT, the Boot Segment must be configured for a minimum size of at least two pages: one for the IVT and BS and one for the AIVT. The AIVT will be located in the last page of the Boot Segment defined by the BSLIMx Configuration bits. Once the AIVT is enabled, the user may choose to direct exceptions to vector from the IVT or AIVT with the AIVTEN control bit (INTCON2<8>).

The AIVT inherits the Boot Segment security settings. If Boot Segment Code Protection is set to high security, all interrupts that occur while executing within the Boot Segment will vector to a single secure vector location within the Boot Segment at the address: [BS Base Address + 40h]. This feature provides the Boot Segment the opportunity to establish a chain of trust by protecting the Boot Segment context and return address prior to allowing a General Segment ISR to execute. See **Section 4.2 "Rules Concerning Interrupts"** for details.

**Note:** The Reset vector is not duplicated within the AIVT, so Resets always vector to 000000h.

### 3.5.2 AIVT CONSIDERATIONS FOR DUAL BOOT MODES

Both the IVT and AIVT may be read from the Active partition. This may present a security issue when updating code in the Inactive partition. To address this concern, the Inactive partition's AIVT can be disabled with the AIVTDIS control bit, effectively applying Boot Segment security to the code to block access from Code Segments in the Active partition. Code in the secure bootloader can then enable the AIVT before it is mapped to the Active partition.

# 4.0 SECURITY PRIVILEGES AND RULES

It is important to understand the relative privilege levels of the two Code Protection segments. Operations can be described as being relative to higher or lower privileged segments. The lower privileged segment can only access code from the higher segment by issuing calls. Rules governing access privileges are discussed in the following sections. Table 4-1 through Table 4-5 present a summary overview of these rules during normal run-time operation.

## 4.1 Rules Concerning Program Flow

Program flow refers to the execution sequence of program instructions in program memory. Normally, instructions are executed sequentially as the Program Counter (PC) increments.

Program Flow Change (PFC) occurs when the Program Counter is reloaded as a result of a branch instruction, allowing the program flow to follow an alternate path. These instructions include Call, Jump, Computed Jump, Return or Return from Subroutine. Branches within the same segment are unrestricted, while branches to a higher security level segment are only possible through restricted PFC. A restricted PFC allows the program to branch to a higher security segment through a special segment access area.

Vector Flow Change (VFC) occurs when the Program Counter is reloaded as the result of an interrupt request or hardware exception trap. These are primarily interrupt or trap vectors.

Jumping into secure code at unintended locations can expose the code to algorithm detection. Therefore, PFC and VFC operations are restricted if they violate the privilege hierarchy. PFCs within a segment are unrestricted. PFCs and VFCs from one segment to another are also not restricted, except when Boot Segment security is set to high. In that case, PFCs and VFCs between segments have the following restrictions:

- To ensure the integrity of the operations of code within the Boot Segment, the user must restrict program flow options to this segment by setting the security level to high
- Program flow can be limited to only allow the secure segment access areas to be a branch target
- The secure segment access areas are the first 32 instruction locations of the Boot Segment

Figure 4-1 illustrates normal and restricted program flow.

The owners of the code inside of the Boot Segment can ensure that the access area contains branches to specified sections of the application code, verified to not expose the algorithm.

If a PFC or VFC targets a restricted location, that operation will cause a security Reset. The device will reset and set the IOPUWR (RCON<14>) status bit, indicating an illegal operation.

In addition to this specific security Reset, there are also program flow checks that are built into all devices. If a PFC or VFC targets unimplemented program memory space, an address error trap occurs.

Code execution from the Vector Segment, other than the instruction at the Reset location, is not allowed. If it is attempted, an address error trap results.

### 4.1.1 FLOW CHANGES INTO THE INACTIVE PARTITION

In Dual Boot mode, an attempted PFC to an Inactive partition address space is regarded as a flow change into an illegal address. This is because execution from Inactive partition address space is not possible and an illegal address trap will result.

**Table 4-1: VS (Active Partition) Access Rules**

| Boot Segment | | Undefined (GSS Security) | | | | | | Defined (BSS Security) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Segment Security Level | | None | | Standard | | High | | None | | Standard | | High | |
| Write Protection | | N | Y | N | Y | N | Y | N | Y | N | Y | N | Y |
| Requested Operation | | | | | | | | | | | | | |
| Read of VS from | BS | N/A | | | | | | Yes | | | | | |
| | GS | Yes | | | | | | Yes | | | | | |
| Program/Page Erase of VS from | BS | N/A | | | | | | Y | (1) | Y | (1) | Y | (1) |
| | GS | Y | N | Y | N | Y | N | Y | No | | | | |

**Note 1:** Operations from IVT are not permitted; operations from AIVT are permitted.

**Table 4-2: BS (Active Partition) Access Rules**

| Segment Security Level | | None | | Standard | | High | |
|---|---|---|---|---|---|---|---|
| Write Protection | | No | Yes | No | Yes | No | Yes |
| Requested Operation: | | | | | | | |
| Read of BS from | BS | Yes | | | | Yes | |
| | GS | Yes | | No | | | |
| Program/Page Erase of BS from | BS | Yes | No | Yes | No | Yes | No |
| | GS | Yes | No | | | | |

**Table 4-3: GS (Active Partition) Access Rules**

| Segment Security Level | | None | | Standard | | High | |
|---|---|---|---|---|---|---|---|
| Write Protection | | No | Yes | No | Yes | No | Yes |
| Requested Operation: | | | | | | | |
| Read of GS from | BS | Yes | | | | No | |
| | GS | Yes | | | | | |
| Program/Page Erase of GS from | BS | Yes(1) | No | Yes(1) | No | | |
| | GS | | | | No | Yes | No |

**Note 1:** Page Erase of the last page of GS is defined by the security level set by CSS<2:0>.

**Table 4-4: CS Access Rules**

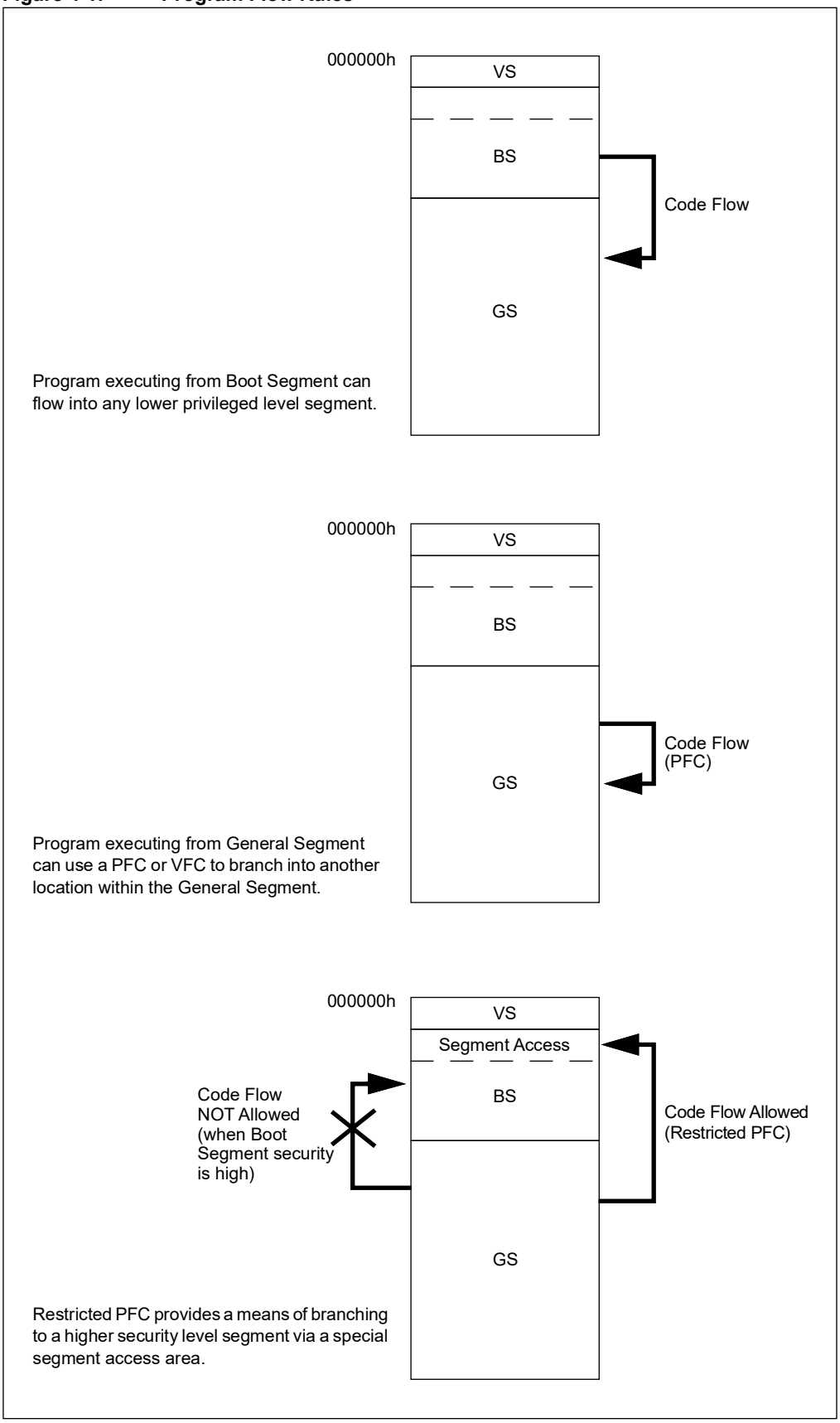
| Active CS Security Level | | None | | Standard | | Enhanced | | High | |
|---|---|---|---|---|---|---|---|---|---|
| Write Protection | | No | Yes | No | Yes | No | Yes | No | Yes |
| Requested Operation: | | | | | | | | | |
| Read of CS from | BS | Yes | | | | | | | |
| | GS | Yes | | | | | | | |
| Program of CS from | BS | Yes | No | Yes | No | Yes | No | Yes | No |
| | GS | | | | No | | | | |
| Page Erase of CS from | BS | Yes | No | Yes | No | Yes | No | | |
| | GS | | No | | | | | | |

**Table 4-5: Partition Erase Rules**

| Requested Operation | From Segment | |
|---|---|---|
| | BS | GS |
| Chip Erase (Active Partition)(1) | No | No |

**Note 1:** Chip Erase erases all user space and is only permitted when programming the device in ICSP mode (refer to **Section 4.3.4 "Rules for In-Circuit Serial Programming™ (ICSP™)"**).

**Figure 4-1: Program Flow Rules**



## 4.2 Rules Concerning Interrupts

Interrupt handling is restricted for the following reasons:

- A return from interrupt is one way to corrupt an intended program flow (by changing the return address in the stack).
- The secure code should have the opportunity to clear sensitive information from registers and RAM before responding to an interrupt.

When Boot Segment Code Protection is high, all interrupts that occur while executing within the Boot Segment vector can be contained in a *single* secure vector location at address: [BS Start Address + 40h]. This feature provides the Boot Segment the opportunity to protect the Boot Segment context and the return address prior to allowing a General Segment ISR to execute.

### 4.2.1 SECURE INTERRUPT HANDLING SEQUENCE

The objective of a secure handling sequence is to preserve the chain of trust by deleting any secure information contained in the W registers or data memory (to be restored later), prior to servicing a General Segment exception that occurs while executing from a Boot Segment configured for high security. This avoids any possibility to corrupt an intended program flow (that could be done by changing the return address in the stack).

When an interrupt or hardware trap occurs while Boot Segment code is executing with high security, the return address is pushed onto the stack and PC is loaded with the address in secure vector location [Base Address + 40h] instead of the usual interrupt vector; this points to a special ISR for the Boot Segment.

1. The special Boot Segment ISR must execute the following:
   a) Any secure information in the W registers or data memory is deleted.
   b) The actual return address from the stack is retrieved and saved in data memory (and encrypted if necessary).
   c) The actual return address is replaced with a new return address, located between the start of the Boot Segment and the Boot Segment Start Address + 03Eh (i.e. the first 32 instruction locations).
   d) The INTTREG is read to determine which interrupt vector to jump to.
   e) The interrupt vector is read from the Interrupt Vector Table and an indirect jump is executed.
2. Execute the application's ISR in the General Segment, execute user code and return from interrupt. This returns the application to the "New Return Address", which will be (or jump to) the recovery routine.
3. Read the actual return address from data memory.

## 4.3 Rules for Flash Access

### 4.3.1 RULES FOR FLASH READS

`TBLRD` and instructions that address program memory through the PSV addressing may be restricted. An unauthorized read of a protected program memory location will read as all '`0`'s. The General Segment cannot read the Boot Segment unless the Boot Segment is configured for no security (BSS<1:0> = `11`). The Boot Segment can read the General Segment unless the General Segment is configured for high security (BSS<1:0> = `0x`). The Configuration Segment and Vector Segment are always readable from the Boot Segment and General Segment, regardless of the security levels of these segments.

In devices that support the Dual Boot mode, the above rules apply between the Active and Inactive partition address spaces.

### 4.3.2 RULES FOR RUN-TIME SELF-PROGRAMMING (RTSP)

Run-Time Self-Programming (RTSP) is performed by first erasing a portion of Flash and then writing the new data to the write latches. The security features prevent the actual write operation based on the segment rules. If segment Write Protection is enabled, the write is blocked.

In devices that support Dual Boot mode, Write Protection is ignored for target segments that reside within the Inactive partition. The exception is the Protected Dual Boot mode, which is discussed in **Section 5.2.2 "Protected Dual Boot Mode"**. Privileged Dual Boot mode allows an Inactive partition to be erased; however, the security features will force the inactive BSLIMx bits to that of the active BSLIMx bits.

### 4.3.3 ERASING A SEGMENT AND CLEARING CODE PROTECTION

The Configuration Segment contains all device Code Protection control bits and resides in user space immediately following the General Segment. The only way to release Code Protection of a segment is to erase the Configuration Segment subject to the restrictions outlined above. It should be noted that the Configuration Segment will be considerably smaller than a Flash page, so the General Segment is permitted to exist all the way up to the start of the Configuration Segment in order to maximize available Flash space. Consequently, users should keep in mind that a Configuration Segment Page Erase will:

- Assume the Configuration Segment security level, which may be higher than that of the General Segment.
- Erase any General Segment code within the same page.

Therefore, Configuration Segment updates (made by the Boot Segment) would also require the General Segment within the Configuration Segment page to be rewritten.

### 4.3.4 RULES FOR IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

When the device is connected to a device programmer, the allowable operations are limited to erasing, programming and verifying the device code and data Flash memory. The device programmer will use Chip, Partition or Page Erase commands to erase the device and clear the Code Protection. ICSP programming may only proceed on an unprotected General Segment, which is not write-protected. Attempts to verify code-protected segments within the device will return '0's. Once the device is programmed with the desired code and Boot mode, the Configuration bits are written to enable the Code Protection level. After this operation, the only way to change the device code is by the code itself or by ICSP erasure and clearing the Code Protection once more.

# 5.0 DUAL BOOT SECURITY

**Note:** Dual boot operation is not present on all dsPIC33 or PIC24 devices. Please refer to the **"Memory Organization"** chapter in the specific device data sheet.

Additional security features are available for applications that use Dual Boot modes. In addition to the segment-to-segment privileges in the Active partition, there are also restrictions placed on operation from code executing in the Active partition into the Inactive partition. There are also two special Dual Boot modes that further enhance Code Protection. Code cannot be executed from any segment in the Inactive partition, including any form of erase or program operation.

## 5.1 Dual Boot Overview

**Note:** For specific details regarding dual boot operation, refer to the **"Flash Program Memory"** section of the specific device data sheet.

When the device is in one of the Dual Boot modes, the memory can be programmed with two independent applications, each in its own partition (also referred to as Partition 1 and Partition 2). During device initialization, the partition with the lower Boot sequence number is mapped to Active partition and is executed. If the Boot sequence numbers of both partitions are equal, partition 1 is mapped to Active partition. Active and Inactive partitions can be swapped, either during run time or initiated by changing the Boot Sequence Numbers and executing a device Reset.

Each code partition (i.e., Partitions 1 and 2) has its own independent Code Protection settings. This includes the Boot Segment size, security level and Write Protection that resides in each partition's CS. Write Protection is only applied to code when it is located in the Active partition and is ignored for code mapped to the Inactive partition (where it cannot be executed). This permits a write-protected segment within the Active partition to program or erase a segment in the Inactive partition, even if it is configured to be write-protected when moved into the Active partition.

**Note:** If a partition swap is initiated using the `BOOTSWP` instruction, all configurations, including the BSLIMx bits and Code Protection values, will not be reconfigured based on the newly Active partition's configuration. A device Reset is needed to reassign Code Protection configuration data to the newly Active partition if it is different. Alternatively, they can be programmed identically to prevent security gaps when soft swapping.

**Table 5-1: VS (Inactive Partition) Access Rules**

| Boot Segment Status | | Undefined (GSS Security) | | | Defined (BSS Security) | | |
|---|---|---|---|---|---|---|---|
| **Segment Security Level** | | **None** | **Standard** | **High** | **None** | **Standard** | **High** |
| Requested Operation from Active Segment: | | | | | | | |
| Read of VS from | BS | N/A | | | Yes | | |
| | GS | Yes | | | Yes | | |
| Program/Page Erase of VS from | BS | N/A | | | Yes | | |
| | GS | Yes | | | Yes | No | |

**Table 5-2: BS and GS (Inactive Partition) Access Rules**

| Segment Security Level | | None | Standard | High |
|---|---|---|---|---|
| Requested Operation from Active Segment: | | | | |
| Read of BS from | BS | Yes | | |
| | GS | Yes | No | |
| Program/Page Erase of BS from | BS | Yes | | |
| | GS | Yes | No | |
| Chip Erase from | BS | No | | |
| | GS | No | | |
| Inactive Partition Erase from(1) | BS | Yes | | |
| | GS | Yes | No | |
| Read of GS from | BS | Yes | | No |
| | GS | Yes | | |
| Program/Page Erase of GS from | BS | Yes | | No |
| | GS | Yes | | |

**Note 1:** An Inactive Panel erase command may be executed from code in either BS or GS (if BS security level is ‘none’).

**Table 5-3: CS (Inactive Partition) Access Rules**

| Inactive CS Security Level | | None | Standard | Enhanced | High |
|---|---|---|---|---|---|
| Requested Operation from Active Segment: | | | | | |
| Read of CS from | BS | Yes | | | |
| | GS | Yes | | | |
| Program of CS from | BS | Yes | | | |
| | GS | Yes | | No | |
| Page Erase of CS from | BS | Yes | | | No |
| | GS | Yes | No | | |

## 5.2 Security Modes for Dual Boot

In Dual Boot mode, there are three security modes available based on the BTMODE setting:

- Dual Boot mode
- Protected Dual Boot mode
- Privileged Dual Boot mode

### 5.2.1 DUAL BOOT MODE

When the device operates in Dual Boot mode, the only security restrictions that are applied are those defined by the Code Protection bits. Write Protection for code in the Inactive partition is always ignored and can always be programmed at any time. As with the Active partition, the Inactive partition VS inherits the General Segment privileges when no Boot Segment is defined, and it inherits Boot Segment privileges when a Boot Segment is defined. Table 5-1 through Table 5-3 show the interaction from the Active partition operation on the Inactive partition's given Code Protection settings.

### 5.2.2 PROTECTED DUAL BOOT MODE

Protected Dual Boot mode adds the additional capability for a "factory default" image in Partition 1 to become permanently erase/write-protected. When in Protected Dual Boot mode, Partition 1 is always write-protected when it is mapped to the Inactive partition, irrespective of its security settings. When it is mapped to the Active partition, the security of the code in Partition 1 is defined by the configuration of the write and Code Protection bits.

#### 5.2.2.1 Protected Dual Boot Mode Example

In the following example of Protected Dual Boot mode, the factory default code image in Partition 1 is mapped to the Inactive partition. Partition 2 contains the active code image to be executed. Factory default code should contain any code required to validate the non-factory code, as well as any procedures required for its recovery.

To achieve this configuration:

1. Configure the device for Protected Dual Boot mode.
2. Configure Partition 1 to enable a Boot Segment and with a Boot Sequence Number of FFFh.
3. Program the factory default code image.
4. Enable Write Protection for all Code Segments on Partition 1, unless it is necessary for the factory code to self-modify when it is mapped to the Active partition.
5. Program the desired application code image into (Inactive) Partition 2. Configure the partition to include a Boot Segment if the application code needs to self-modify.
6. Promote Partition 2 to the Active partition by programming the Partition 2 Boot Sequence value to be less than FFFh.

When updating the active code in the field, erase the Active partition first to reset the Boot Sequence Number, such that the factory default will be used if an error occurs.

| **Note:** | In Protected Dual Boot mode, enabling Write Protection for Partition 1 fully protects the code from all writes and erases, except for a full Chip Erase. |
|---|---|

### 5.2.3 PRIVILEGED DUAL BOOT MODE

Privileged Dual Boot mode adds additional security protection to allow for protection of intellectual property when multiple parties have software within the device by enforcing the boot size limit. Privileged Dual Boot mode is not available in all dual boot devices; refer to the specific device data sheet for more information.

When in Privileged Dual Boot mode, an Inactive partition erase operation forces the boot size limit to be automatically copied from the Active partition's Configuration Word to the Inactive partition's boot size limit. This prevents malicious code from being able to alter the inactive boot size, effectively placing Boot Segment code in the General Segment space where it may be accessed.

When operating in Privileged Dual Boot mode, it is recommended that the Boot Segment owner creates a Boot Segment of the same size in both the Active and Inactive partition space, even if nothing is to be programmed into the Inactive partition Boot Segment. This prevents the General Segment owner from writing malicious code into the Inactive partition General Segment via a lower sequence number and creating an Inactive partition Boot Segment to encompass it. The General Segment will have write access to the Configuration Segment at this stage and the Trojan can become active at the next Reset, and could then read and dump the contents of the Boot Segment owner's code.

#### 5.2.3.1 Privileged Dual Boot Mode Example

In the following example, application code from two separate parties is programmed into a single device that can be updated in the field without code stalling. For this example, Party 1 is the author of a proprietary algorithm and the bootloader code; Party 2 is the author of the main application code that should not have access to Party 1's proprietary algorithm. Ideally, the bootloader should use an encrypted communication scheme for field updated equipment with the encryption code secured in the Boot Segment.

In this scenario, Party 1 does the following:

1. Party 1 configures the device for Privileged Dual Boot mode with equally sized boot spaces defined in both partitions. Boot Segment Code Protection bits are configured as high; Configuration Segment code is configured as standard, allowing the General Segment to write to, but not to erase (lower), security.
2. The device is programmed with the bootloader and proprietary algorithm into the BS; after this, Write Protection is enabled for the Boot Segment. The Configuration Segment can now only be programmed from the Boot Segment and is effectively secured from Party 2 code in the General Segment .
3. The partially programmed device is shipped to Party 2.

Party 2 then programs the main application code into the General Segment, and it sets the General Segment and Configuration Segment security as high. Now the General Segment cannot be erased or programmed by the Boot Segment.

When field updates are required, the application code recognizes and authenticates the update request, then vectors to the bootloader in the Boot Segment. Code updates can be programmed into the Inactive partition's Boot Segment from the Active partition's Boot Segment.

| **Note:** | In the event that the updated application requires a resized Boot Segment, the new partition size (defined by the BSLIMx bits) and the Code Protection bits must be programmed before the application is programmed in case of a device Reset. |
|---|---|

If the update is destined for the General Segment, store the data in RAM, and then have the Boot Segment flag the General Segment that an update is required. A jump to a predefined location in the General Segment is done so that the General Segment can update itself.

The Inactive partition can be made Active and a software Reset is then executed.

# 6.0 FLASH OTP WITH ISCP WRITE INHIBIT

ICSP Write Inhibit is an access restriction feature that, when activated, write-protects entire Flash memory. Once activated, ICSP Write Inhibit permanently prevents ICSP Flash programming and erase operations and cannot be deactivated. This feature is intended to prevent alteration of Flash memory contents, with behavior similar to One-Time-Programmable (OTP) devices.

## 6.1 Activating ICSP Write Inhibit

ICSP write inhibit can be activated by programming two Flash words in configuration memory space with predefined 16-bit activation values per word. Refer to the device specific data sheet for the target NVM addresses and values required for the activation. Once both addresses are programmed with their activation values, ICSP Write Inhibit will take permanent effect on the next device Reset. Once ICSP Write Inhibit is successfully activated, these addresses can not be erased or modified by any means.

The addresses can be programmed in any order and also during separate ICSP/Enhanced ICSP/RTSP sessions, but any attempt to program an incorrect 16-bit value or use a row programming operation to program the values will be aborted without altering the existing data.

## 6.2 Enhanced Security with ICSP Write Inhibit

ICSP write inhibit locks the device from further ICSP operations, restricting any further attempt to reprogram or erase the contents of device Flash.

RTSP operation, including erase and programming operations, is not restricted when ICSP Write Inhibit is activated; however, code to perform these actions must be programmed into the device before ICSP Write Inhibit is activated. This allows for a bootloader application to update the Flash contents even when the ICSP Write Inhibit is activated.

## 6.3 Disable Entry Into Debug

Entry to debug mode shall be permanently disabled once the ICSP Write Inhibit mode is turned on in release mode.

If the ICSP Write Inhibit mode is activated in Debug mode, it will not be possible to exit debug mode since the Write Bit will be locked and erasing and writing to the Flash will be completely restricted. ICSP Write Inhibit should therefore only be activated on devices programmed for production.

# 7.0 IMMUTABLE SECURE BOOT

A device boot code is the first code to be executed after a reset and may be used to create a secure “root-of-trust” environment. It is essential for any “root of trust” area to be immutable in order to prevent any code alterations to the root of trust segment. A secure root of trust provides a firm basis to develop security as it is used to validate the authenticity and integrity of software before it is executed. Additionally, it may also support features such as secure field updates, immutable storage of public keys and certificates and facilitate secure boot.

CodeGuard™ Intermediate Security creates an immutable environment for the secure Boot Segment by preventing code in the General Segment from accessing and erasing it and by preventing the code in the Boot Segment from modifying itself. CodeGuard provides three levels of security to the Boot Segment that may contain secure boot code or a proprietary algorithm. With Boot Segment defined to have ‘high’ security, the allowable operations are more restricted than with standard/low security. At this security level, CodeGuard also enables a secure vector (at Boot Segment Base Address + 40h) to the Boot Segment in order to recognize and safeguard any secure information in the W registers or RAM prior to servicing a General Segment exception that occurs while executing from ‘high’ security Boot Segment.

The Boot Segment can also be Write Protected by programming the BWRP configuration bit. When write protected, page erase and row or word programming operations targeting Boot Segment are inhibited.

ICSP write inhibit protects the device from ICSP operations, restricting any further effort to reprogram or erase the contents of device, especially the Boot Segment Flash by ICSP.

With all of these capabilities, CodeGuard™ Intermediate Security together with ICSP Write Inhibit establishes an immutable environment for the secure root of trust code.

## 7.1 Boot Loading a Device

A typical scenario where Code Protection is required is in a system that can be upgraded in the field. This section describes a scenario that shows how a tier-1 manufacturer can secure their proprietary code/IP in Boot Segment, while shipping the product to OEM. The OEM can then develop their application in General Segment and make use of APIs or functions implemented by the tier-1 manufacturer in Boot Segment.

In this scenario, the tier-1 manufacturer programs the device with their code during the manufacturing process and then secures it by setting appropriate Code Protection values for Boot Segment as described in **Section 7.1.1 “Tier-1 Manufacturer Programming”**. This code may include the Tier-1 Manufacturer's proprietary algorithms and secure bootloader code to receive and verify the field updates. The OEM must then program and secure their main application code (as described in **Section 7.1.2 “OEM Programming”**) into the end product at their facility. Finally, the application code (and possibly the tier-1 manufacturer's code) can apply the received patch update through RTSP for it to be capable of being updated in the field. Ultimately, neither the OEM nor tier-1 manufacturer will be able to see or modify each other's code. This example assumes a Single Boot model and uses both Boot Segment and General Segment for code.

### 7.1.1 TIER-1 MANUFACTURER PROGRAMMING

The tier-1 manufacturer programs Boot Segment with their proprietary code and then sets the Boot Segment security level to ‘high’, and configuration segment security to ‘standard.’ Configuration segment can still be written to (but not erased) by General Segment; therefore, inhibiting any attempt by General Segment to lower the Code Protection security of Boot Segment and access the boot code. Boot Segment may also include boot firmware to support a secure field update. The product is then shipped to the OEM.

### 7.1.2 OEM PROGRAMMING

The OEM programs General Segment through secure bootloading and then sets both the General Segment and configuration segment security level to ‘high.’ General segment can now no longer be read by Boot Segment, and configuration segment can now no longer be erased or programmed by either Boot Segment or General Segment. Boot Segment and General Segment are now secure from each other.

### 7.1.3 FIELD UPDATES

As the system is operating in the field, a technician connects a re-programming tool to the system. The application recognizes this connection and branches to a location within the Boot Segment access area. Note that this branch needs to be through the Boot Segment access area and that attempts to modify this branch will result in a device Reset.

The Boot Segment contains the code allowing authenticated communication with the tool. The Boot Segment may also contain a public key used for signature verification using an asymmetric cryptographic algorithm such as ECC-P256 based ECSDA. Since, public keys are placed in Boot Segment with Boot Segmented Code Protection set to High, only the secure bootloader code will have access to these public keys.

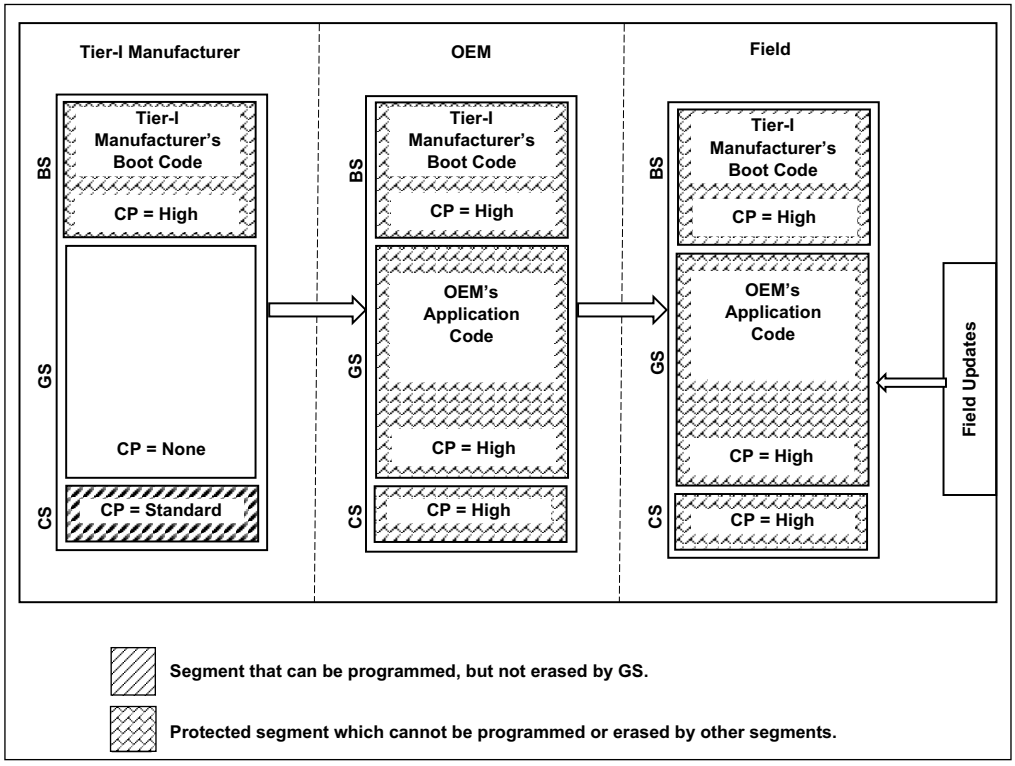
Once the boot loader verifies valid communication with the tool, it can then receive the code update from the tool and store it in a mutually agreed upon area of RAM.

If the update is destined for General Segment, Boot Segment then flags the General Segment code that an update is required and jumps back into General Segment code at a predefined location. The General Segment code will see the flag and then re-flash (Page Erase, program and verify) the area in need of a patch. General Segment is in ‘high’ security, so the re-flash can only be executed within General Segment.

If the update is destined for Boot Segment, then Boot Segment will apply the patch.

As the boot loader is running, it is immune from disruption from interrupts or traps as it can vector those to the secure vector location within the boot loader itself. After an update, the application may or may not be software Reset, depending upon the patch.

**FIGURE 7-1: PROGRAM MEMORY SEGMENTS WITH VALUES OF CP BITS IN EACH STAGE**



## 8.0 DESIGN TIPS

***Question 1:*** ***Can I bootload a device with basic Code Protection?***

**Answer:** Remember that devices with basic Code Protection only have one segment: the General Segment. Because there is only one segment, it is not possible to erase the segment and clear Code Protection without also erasing any bootloader that might be resident within the General Segment.

This limits the options for booting but does not prevent it. The bootloader needs to erase and reprogram Flash in “less than segment” partitions, and the loader cannot select Write Protection for the General Segment. It is also not possible to protect the loaded code from compromises caused by the bootloader itself.

***Question 2:*** ***Can the system load part of the code now and the rest of the code later?***

**Answer:** As long as neither Write Protection nor high security is selected for the segment, “incremental” loads are possible. Incremental loads are still possible in high-security segments as long as the loader resides within that segment. However, once the segment is write-protected, it cannot be changed until the entire segment is erased and Code Protection is cleared by a segment erase command.

You can choose to locate a jump table for interrupt vectors in an unprotected segment and update the jump table with changing interrupt vectors. This allows Boot Segment Write Protection.

## 9.0 RELATED DOCUMENTS

This section lists documents that are related to this section of the manual. These documents may not be written specifically for the dsPIC33 or PIC24 product families, but the concepts are pertinent and could be used with modification and possible limitations.

The current documents related to CodeGuard™ Intermediate Security are:

| Title | Document # |
|---|---|
| CodeGuard™ Security: Protecting Intellectual Property in Collaborative System Designs | DS70179 |

**Note:** Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC33 and PIC24 families of devices.

## 10.0 REVISION HISTORY

### Revision A (May 2014)

This is the initial released version of this document.

### Revision B (April 2023)

This revision includes the following updates:

- Sections:
  - Added **Section 6.0 “Flash OTP with ISCP Write Inhibit”** and **Section 7.0 “Immutable Secure Boot”**.
  - Updated **Section 1.0 “Introduction”**, **Section 3.0 “Code Segment Organization”**, **Section 3.1 “Code Protection Bits”**, **Section 3.2 “Boot Segment (BS)”**, **Section 3.2.1 “Allocation of the BS”**, **Section 3.2.2 “Selecting the Security Level”**, **Section 3.3.2 “Write Protection of the GS”**, **Section 3.5.1 “Alternate Interrupt Vector Table (AIVT)”**, **Section 4.1 “Rules Concerning Program Flow”**, **Section 4.2 “Rules Concerning Interrupts”**, **Section 4.2.1 “Secure Interrupt Handling Sequence”** and **Section 5.1 “Dual Boot Overview”**.
- Figures:
  - Figure 3-1
- Tables:
  - Table 4-2, Table 4-3 and Table 5-2.

**NOTES:**

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

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



*For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.*

**Trademarks**



ISBN: 978-1-6683-2306-9

# Worldwide Sales and Service

## AMERICAS

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

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

**Austin, TX**
Tel: 512-257-3370

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

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

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

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

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

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

**Raleigh, NC**
Tel: 919-844-7510

**New York, NY**
Tel: 631-435-6000

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

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

## ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**China - Xiamen**
Tel: 86-592-2388138

**China - Zhuhai**
Tel: 86-756-3210040

## ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

**Singapore**
Tel: 65-6334-8870

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

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

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

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

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

## EUROPE

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

**Denmark - Copenhagen**
Tel: 45-4485-5910
Fax: 45-4485-2829

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

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

**Germany - Garching**
Tel: 49-8931-9700

**Germany - Haan**
Tel: 49-2129-3766400

**Germany - Heilbronn**
Tel: 49-7131-72400

**Germany - Karlsruhe**
Tel: 49-721-625370

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

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

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

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

**Italy - Padova**
Tel: 39-049-7625286

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

**Norway - Trondheim**
Tel: 47-7288-4388

**Poland - Warsaw**
Tel: 48-22-3325737

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

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

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

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

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