MICROCHIP

Section 23. CodeGuard™ Security

HIGHLIGHTS

This section of the manual contains the following major topics:

23.1 CODE PROTECTION OVERVIEW

Microchip's CodeGuard™ Security enables multiple parties to securely share resources (memory, interrupts and peripherals) on a single chip. Intellectual Property (IP) vendors, Original Design/Original Equipment Manufacturers (ODM/OEM) and Value-Added Resellers (VAR) now have an opportunity to reap the following benefits using these on-chip code protection features:

- System cost reduction
- Component reduction and associated benefits to inventory management
- Decreased risk of losing IP to unqualified partners
- Increased security during code distribution and Flash memory update

The on-chip program Flash memory in a PIC24H device can be organized into three code space segments. Each of these segments has an implied security privilege level and system function.

1. The Boot Segment (BS) has the highest security privilege level. It has greater access to the other segments. The Boot Segment is intended for secure boot loader and device update functions.
2. The Secure Segment (SS) is the next highest security privilege. This segment is designed for storing proprietary algorithms from algorithm vendors.
3. The General Segment (GS) has the lowest security privilege. This segment is designed for the end user system code.

Segments of user data RAM space of the device can be allocated as secure RAM, which are directly associated with the Boot or Secure segments.

Any operation of the system that potentially allows exposure of the code or data contents is restricted, based on the segment from which the operation originated, or the segment to which the operation targets.

Restricted operations include:

- Programming, Erase or Verify Operations
- Reads or Writes of Code Space
- Reads or Writes of protected Data Space
- Code Flow Change into a Secure Segment from outside the segment
- Interrupt Vectors into a Secure Segment

Configuration bits are provided to enable access to the Secure Segments and their parameters. These bits allow configuration of both the sizes and restrictions of the program Flash memory, and RAM segments.

23.2 DEVICE SPECIFIC CODE PROTECTION FEATURES

Two different subsets of CodeGuard Security features are available on PIC24H devices.

On devices with smaller memory sizes, program memory can be allocated to Boot Segment and General Segment; however, there is no Secure Segment and no Data RAM protection.

On larger memory devices, memory can be allocated to Boot Segment, Secure Segment and General Segment code space. On these larger memory devices, Data RAM can be allocated to Boot and Secure Segments.

Table 23-1 shows the code protection features that are available. Refer to the specific device data sheet to correlate these features with specific devices or product families.

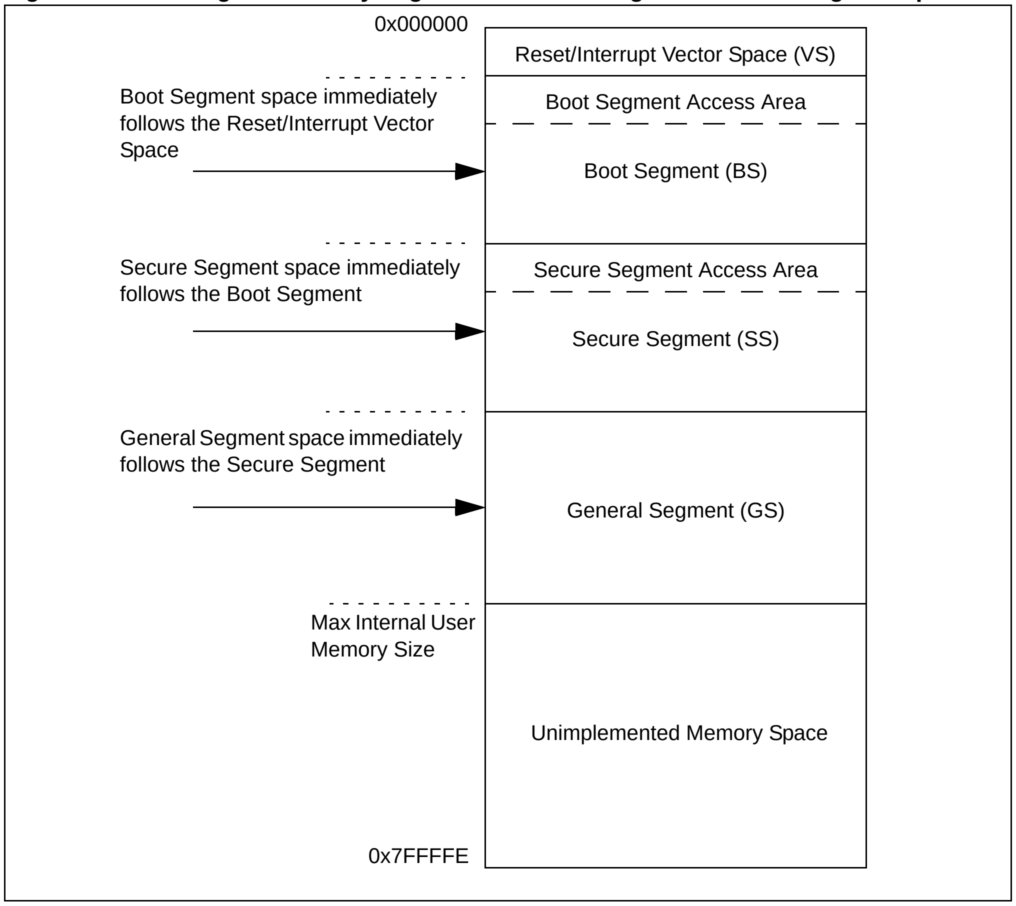
Table 23-1: Code Protection Features

Feature	Smaller Memory Devices	Larger Memory Devices
Assign Code Space as Boot Segment	Yes	Yes
Assign Code Space as Secure Segment	—	Yes
Assign Code Space as General Segment	Yes	Yes
Assign Data Ram to Boot and Secure Segments	—	Yes

23.3 PROGRAM MEMORY ORGANIZATION

The total user program memory can be allocated into one of the three segments. The size of these different segments is determined by Configuration bits. The relative location of the segments does not change, such that a Boot Segment, if present, occupies the memory area just after the device interrupt vector space. The Secure Segment, if present, occupies the space just after the Boot Segment and the General Segment occupies the space just after the Secure Segment (refer to Figure 23-1).

Figure 23-1: Program Memory Organization From Segment and Privilege Perspective

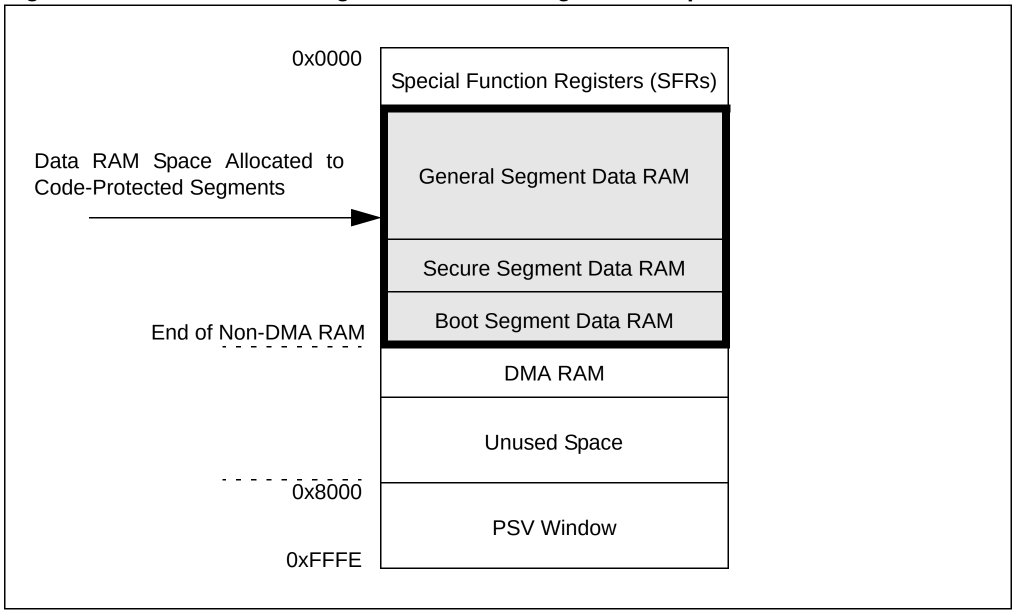

0x000000
Reset/Interrupt Vector Space (VS)
Boot Segment space immediately follows the Reset/Interrupt Vector Space
Boot Segment Access Area
Boot Segment (BS)
Secure Segment space immediately follows the Boot Segment
Secure Segment Access Area
Secure Segment (SS)
General Segment space immediately follows the Secure Segment
General Segment (GS)
Max Internal User Memory Size
Unimplemented Memory Space
0x7FFFFE

23.4 DATA RAM ORGANIZATION

Data RAM memory can also be allocated into code protection segments: Boot, Secure and General. Segment size is primarily specified by Configuration bits. The relative locations of the segments do not change, meaning that a Boot Segment RAM area occupies the memory region at the end of the non-DMA RAM, the Secure Segment RAM occupies the area just before the Boot Segment and the General Segment RAM occupies the remainder of the data RAM space (refer to Figure 23-2).

Note:	DMA RAM is not present on all PIC24H devices. For DMA RAM availability and sizes, refer to the specific device data sheet.

Figure 23-2: DATA RAM Organization From Segment Perspective



23.5 CONTROL REGISTERS

Several Configuration and Special Function Registers control the security functions. On basic and intermediate security implementations, some of these registers do not exist. The key registers for supporting the code security features are:

- **FBS: Boot Segment Configuration Register Byte**
- **BSRAM: Boot Segment RAM Special Function Register**
- **FSS: Secure Segment Configuration Register Byte**
- **SSRAM: Secure Segment RAM Special Function Register**
- **FGS: General Segment Configuration Register**
- **INTTREG: Interrupt Vector and Priority Status Special Function Register**

23.6 THE BOOT SEGMENT (BS)

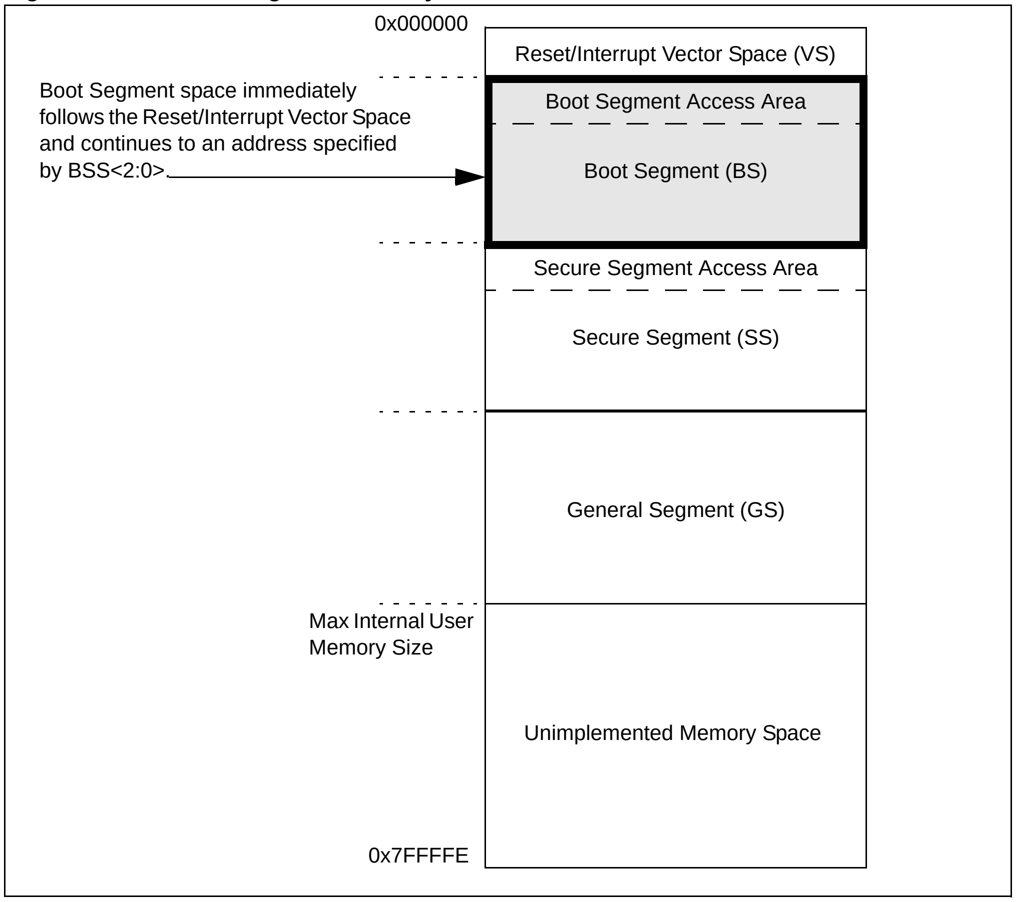
The Boot Segment has the highest privilege. The Boot Segment can be small, allowing a simple yet secure boot loader, or it can be large, enabling it to hold a more sophisticated secure operating system.

The Boot Segment can also rewrite its own locations, enabling it to store and update data such as “encryption keys”.

23.6.1 Allocating the Boot Segment

The existence and size of the Boot Segment are determined by Configuration bits BSS<2:0> (FBS<3:1>). The default option, on an erased non-programmed device, is to exclude the Boot Segment. When implemented, the Boot Segment begins at the end of the interrupt vector space and continues to an address specified by the BSS<2:0> bits.

Figure 23-3: Boot Segment Memory Allocation


0x000000
Reset/Interrupt Vector Space (VS)
Boot Segment space immediately follows the Reset/Interrupt Vector Space and continues to an address specified by BSS<2:0>.
Boot Segment Access Area
Boot Segment (BS)
Secure Segment Access Area
Secure Segment (SS)
General Segment (GS)
Max Internal User Memory Size
Unimplemented Memory Space
0x7FFFFE

23.6.1.1 BOOT SEGMENT SIZE OPTIONS

For an example of Boot Segment size options, refer to Table 23-2 (in this case for devices with 64 KB flash memory). The start and end program memory addresses listed are typical. For specific program memory addresses for a given device, refer to the device data sheet.

Table 23-2: Boot Segment Size Example

BSS2:BSS0	Security Level	BS Size	BS Start Address	BS End Address
`x11`	No Boot Program Flash Segment			
`110`	Standard	Small	0x000200	0x0007FE
`010`	High	Small	0x000200	0x0007FE
`101`	Standard	Medium	0x000200	0x001FFE
`001`	High	Medium	0x000200	0x001FFE
`100`	Standard	Large	0x000200	0x003FFE
`000`	High	Large	0x000200	0x003FFE

Register 23-1: FBS: Boot Segment Configuration Register Byte

Lower Third Byte:

R/P	R/P	U	U	R/P	R/P	R/P	R/P
RBS1	RBS0	R	R	BSS2	BSS1	BSS0	BWRP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-6 **RBS<1:0>:** Boot Segment RAM Code Protection bits(1)
`11` = No Boot RAM defined
`10` = Boot RAM is 128 Bytes
`01` = Boot RAM is 256 Bytes
`00` = Boot RAM is 1024 Bytes

bit 5-4 **Reserved bits:** Programming values will have no effect

bit 3-1 **BSS<2:0>:** Boot Segment Program Flash Code Protection bits(2)
`X11` = No Boot program Flash segment
`110` = Standard security, Small Boot Segment
`010` = High security, Small Boot Segment
`101` = Standard security, Medium Boot Segment
`001` = High security, Medium Boot Segment
`100` = Standard security, Large Boot Segment
`000` = High security, Large Boot Segment

bit 0 **BWRP:** Boot Segment Program Flash Write Protection bit
`1` = Boot segment can be written
`0` = Boot segment is write-protected

Note 1: Not all devices have Boot Segment RAM code protection. For device specific information refer to Table 23-3, Table 23-3 and Table 23-4.

2: The exact definitions of Small, Medium, and Large Boot Segment vary from one device to another. For device specific information refer to Table 23-5, Table 23-6 and Table 23-7.

3: If a Boot Segment is not needed, the BWRP bit must be programmed as a “`1`”.

Table 23-3: Data RAM Segment Sizes for Devices with 16 KB RAM

CONFIGURATION BITS	RBS<1:0> = 11 OR RBS<1:0> = 10 AND RL_BSR = 1	RBS<1:0> = 10 AND RL_BSR = 0 OR RBS<1:0> = 01 AND RL_BSR = 1	RBS<1:0> = 01 AND RL_BSR = 0 OR RBS<1:0> = 00 AND RL_BSR = 1	RBS<1:0> = 00 AND RL_BSR = 0
RSS<1:0> = 11 OR RSS<1:0> = 10 and RL_SSR = 1	0x0800 GS RAM = 14336 0x3FFF	0x0800 GS RAM = 14208 0x3F80 BS RAM = 128 0x3FFF	0x0800 GS RAM = 14080 0x3F00 BS RAM = 256 0x3FFF	0x0800 GS RAM = 13312 0x3C00 BS RAM = 1024 0x3FFF
RSS<1:0> = 10 and RL_SSR = 0 or RSS<1:0> = 01 AND RL_SSR = 1	0x0800 GS RAM = 14080 0x3F00 SS RAM = 256 0x3FFF	0x0800 GS RAM = 14080 0x3F00 SS RAM = 128 0x3F80 BS RAM = 128 0x3FFF	0x0800 GS RAM = 14080 0x3F00 BS RAM = 256 0x3FFF	0x0800 GS RAM = 13312 0x3C00 BS RAM = 1024 0x3FFF
RSS<1:0> = 01 AND RL_SSR = 0 OR RSS<1:0> = 00 AND RL_SSR = 1	0x0800 GS RAM = 12288 0x3800 SS RAM = 2048 0x3FFF	0x0800 GS RAM = 12288 0x3800 SS RAM = 1920 0x3F80 BS RAM = 128 0x3FFF	0x0800 GS RAM = 12288 0x3800 SS RAM = 1792 0x3F00 BS RAM = 256 0x3FFF	0x0800 GS RAM = 12288 0x3800 SS RAM = 1024 0x3C00 BS RAM = 1024 0x3FFF
RSS<1:0> = 00 AND RL_SSR = 0	0x0800 GS RAM = 10240 0x3000 SS RAM = 4096 0x3FFF	0x0800 GS RAM = 10240 0x3000 SS RAM = 3968 0x3F80 BS RAM = 128 0x3FFF	0x0800 GS RAM = 10240 0x3000 SS RAM = 3840 0x3F00 BS RAM = 256 0x3FFF	0x0800 GS RAM = 10240 0x3000 SS RAM = 3072 0x3C00 BS RAM = 1024 0x3FFF

Legend: OR = Logical OR, AND = Logical AND

Note: If the defined Boot Segment size is greater than, or equal to, the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.

Table 23-4: Data RAM Segment Sizes for Devices with 8 KB RAM

CONFIGURATION BITS	RBS<1:0> = `11` OR RBS<1:0> = `10` AND RL_BSR = `1`	RBS<1:0> = `10` AND RL_BSR = `0` OR RBS<1:0> = `01` AND RL_BSR = `1`	RBS<1:0> = `01` AND RL_BSR = `0` OR RBS<1:0> = `00` AND RL_BSR = `1`	RBS<1:0> = `00` AND RL_BSR = `0`
RSS<1:0> = `11` OR RSS<1:0> = `10` AND RL_SSR = `1`	0x0800 GS RAM = 6144 0x1FFF	0x0800 GS RAM = 6016 0x1F80 BS RAM = 128 0x1FFF	0x0800 GS RAM = 5888 0x1F00 BS RAM = 256 0x1FFF	0x0800 GS RAM = 5120 0x1C00 BS RAM = 1024 0x1FFF
RSS<1:0> = `10` AND RL_SSR = `0` OR RSS<1:0> = `01` AND RL_SSR = `1`	0x0800 GS RAM = 5888 0x1F00 SS RAM = 256 0x1FFF	0x0800 GS RAM = 5888 0x1F00 SS RAM = 128 0x1F80 BS RAM = 128 0x1FFF	0x0800 GS RAM = 5888 0x1F00 BS RAM = 256 0x1FFF	0x0800 GS RAM = 5120 0x1C00 BS RAM = 1024 0x1FFF
RSS<1:0> = `01` AND RL_SSR = `0` OR RSS<1:0> = `00` AND RL_SSR = `1`	0x0800 GS RAM = 4096 0x1800 SS RAM = 2048 0x1FFF	0x0800 GS RAM = 4096 0x1800 SS RAM = 1920 0x1F80 BS RAM = 128 0x1FFF	0x0800 GS RAM = 4096 0x1800 SS RAM = 1792 0x1F00 BS RAM = 256 0x1FFF	0x0800 GS RAM = 4096 0x1800 SS RAM = 1024 0x1C00 BS RAM = 1024 0x1FFF
RSS<1:0> = `00` AND RL_SSR = `0`	0x0800 GS RAM = 2048 0x1000 SS RAM = 4096 0x1FFF	0x0800 GS RAM = 2048 0x1000 SS RAM = 3968 0x1F80 BS RAM = 128 0x1FFF	0x0800 GS RAM = 2048 0x1000 SS RAM = 3840 0x1F00 BS RAM = 256 0x1FFF	0x0800 GS RAM = 2048 0x1000 SS RAM = 3072 0x1C00 BS RAM = 1024 0x1FFF

Legend: OR = Logical OR, AND = Logical AND

Note: If the defined Boot Segment size is greater than, or equal to, the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.

Table 23-5: Program Flash Segment Sizes for 256 KB Devices

CONFIGURATION BITS	BSS<2:0> = x11 0K	BSS<2:0> = x10 1K	BSS<2:0> = x01 4K	BSS<2:0> = x00 8K
SSS<2:0> = x11 0K	0x000000 VS = 256 IW 0x000200 GS = 87296 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 GS = 86528 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 GS = 83456 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 GS = 79360 IW 0x02ABFE
SSS<2:0> = x10 8K	0x000000 VS = 256 IW 0x000200 SS = 7936 IW 0x004000 GS = 79360 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 7168 IW 0x004000 GS = 79360 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 SS = 4096 IW 0x004000 GS = 79360 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 GS = 79360 IW 0x02ABFE
SSS<2:0> = x01 16K	0x000000 VS = 256 IW 0x000200 SS = 16128 IW 0x008000 GS = 71168 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 15360 IW 0x008000 GS = 71168 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 SS = 12288 IW 0x008000 GS = 71168 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 SS = 8192 IW 0x008000 GS = 71168 IW 0x02ABFE
SSS<2:0> = x00 32K	0x000000 VS = 256 IW 0x000200 SS = 32512 IW 0x010000 GS = 54784 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 31744 IW 0x010000 GS = 54784 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 SS = 28672 IW 0x010000 GS = 54784 IW 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 SS = 24576 IW 0x010000 GS = 54784 IW 0x02ABFE

Legend: IW = Instruction Words

Note: If the defined Boot Segment size is greater than, or equal to, the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.

Table 23-6: Program Flash Segment Sizes for 128 KB Devices

CONFIGURATION BITS	BSS<2:0> = x11 0K	BSS<2:0> = x10 1K	BSS<2:0> = x01 4K	BSS<2:0> = x00 8K

SSS<2:0> = x11
0K
	VS = 256 IW 0x000000
	0x000200
	GS = 43776 IW
	0x0157FE
	0x02ABFE
		VS = 256 IW 0x000000
		BS = 768 IW 0x000200
		0x000800
		GS = 43008 IW
		0x0157FE
		0x02ABFE
			VS = 256 IW 0x000000
			BS = 3840 IW 0x000200
			0x002000
			GS = 39936 IW
			0x0157FE
			0x02ABFE
				VS = 256 IW 0x000000
				BS = 7936 IW 0x000200
				0x004000
				GS = 35840 IW
				0x0157FE
				0x02ABFE

SSS<2:0> = x10
8K
	VS = 256 IW 0x000000
	0x000200
	SS = 7936 IW
	0x004000
	GS = 35840 IW
	0x0157FE
	0x02ABFE
		VS = 256 IW 0x000000
		BS = 768 IW 0x000200
		0x000800
		SS = 7168 IW
		0x004000
		GS = 35840 IW
		0x0157FE
		0x02ABFE
			VS = 256 IW 0x000000
			BS = 3840 IW 0x000200
			0x002000
			SS = 4096 IW
			0x004000
			GS = 35840 IW
			0x0157FE
			0x02ABFE
				VS = 256 IW 0x000000
				BS = 7936 IW 0x000200
				0x004000
				GS = 35840 IW
				0x0157FE
				0x02ABFE

SSS<2:0> = x01
16K
	VS = 256 IW 0x000000
	0x000200
	SS = 16128 IW
	0x008000
	GS = 27648 IW
	0x0157FE
	0x02ABFE
		VS = 256 IW 0x000000
		BS = 768 IW 0x000200
		0x000800
		SS = 15360 IW
		0x008000
		GS = 27648 IW
		0x0157FE
		0x02ABFE
			VS = 256 IW 0x000000
			BS = 3840 IW 0x000200
			0x002000
			SS = 12288 IW
			0x008000
			GS = 27648 IW
			0x0157FE
			0x02ABFE
				VS = 256 IW 0x000000
				BS = 7936 IW 0x000200
				0x004000
				SS = 8192 IW
				0x008000
				GS = 27648 IW
				0x0157FE
				0x02ABFE

SSS<2:0> = x00
32K
	VS = 256 IW 0x000000
	0x000200
	SS = 32512 IW
	GS = 11264 IW 0x010000
	0x0157FE
	0x02ABFE
		VS = 256 IW 0x000000
		BS = 768 IW 0x000200
		0x000800
		SS = 31744 IW
		GS = 11264 IW 0x010000
		0x0157FE
		0x02ABFE
			VS = 256 IW 0x000000
			BS = 3840 IW 0x000200
			0x002000
			SS = 28672 IW
			GS = 11264 IW 0x010000
			0x0157FE
			0x02ABFE
				VS = 256 IW 0x000000
				BS = 7936 IW 0x000200
				0x004000
				SS = 24576 IW
				GS = 11264 IW 0x010000
				0x0157FE
				0x02ABFE

Legend: IW = Instruction Words

Note: If the defined Boot Segment size is greater than, or equal to, the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.

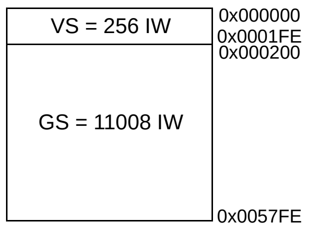
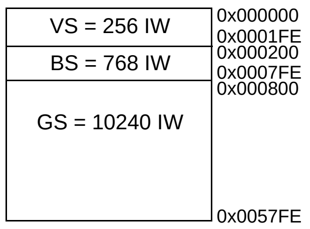
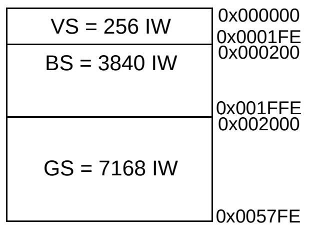
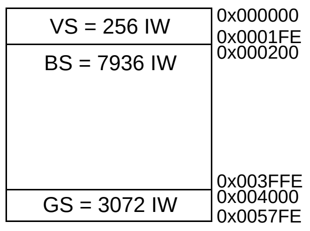
Table 23-7: Program Flash Segment Sizes for 64 KB Devices

CONFIGURATION BITS	BSS<2:0> = `x11` 0K	BSS<2:0> = `x10` 1K	BSS<2:0> = `x01` 4K	BSS<2:0> = `x00` 8K
SSS<2:0> = `x11` **0K**	0x000000 VS = 256 IW GS = 21760 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 GS = 20992 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 GS = 17920 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE
SSS<2:0> = `x10` **4K**	0x000000 VS = 256 IW 0x000200 SS = 3840 IW 0x002000 GS = 17920 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 3072 IW 0x002000 GS = 17920 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 GS = 17920 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE
SSS<2:0> = `x01` **8K**	0x000000 VS = 256 IW 0x000200 SS = 7936 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 7168 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 SS = 4096 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 GS = 13824 IW 0x00ABFE 0x02ABFE
SSS<2:0> = `x00` **16K**	0x000000 VS = 256 IW 0x000200 SS = 16128 IW 0x008000 GS = 5632 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 768 IW 0x000800 SS = 15360 IW 0x008000 GS = 5632 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 3840 IW 0x002000 SS = 12288 IW 0x008000 GS = 5632 IW 0x00ABFE 0x02ABFE	0x000000 VS = 256 IW 0x000200 BS = 7936 IW 0x004000 SS = 8192 IW 0x008000 GS = 5632 IW 0x00ABFE 0x02ABFE

Legend: IW = Instruction Words

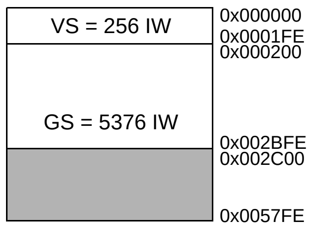
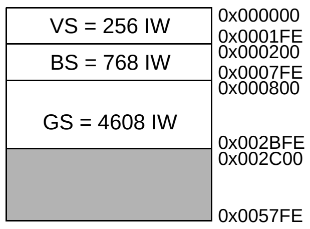
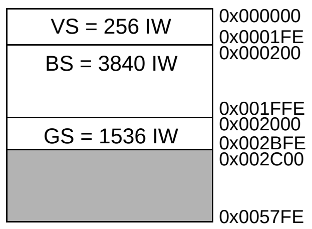
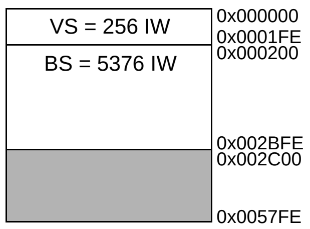
Note: If the defined Boot Segment size is greater than, or equal to, the defined Secure Segment, then the Secure Segment size selection has no effect and the Secure Segment is disabled.

Table 23-8: Program Flash Segment Sizes for 32KB Devices

BSS<2:0> = x11 0K	**BSS<2:0> = x10 1K**
 VS = 256 IW GS = 11008 IW 0x000000 0x0001FE 0x000200 0x0057FE	 VS = 256 IW BS = 768 IW GS = 10240 IW 0x000000 0x0001FE 0x000200 0x0007FE 0x000800 0x0057FE
BSS<2:0> = x01 4K	**BSS<2:0> = x00 8K**
 VS = 256 IW BS = 3840 IW GS = 7168 IW 0x000000 0x0001FE 0x000200 0x001FFE 0x002000 0x0057FE	 VS = 256 IW BS = 7936 IW GS = 3072 IW 0x000000 0x0001FE 0x000200 0x003FFE 0x004000 0x0057FE

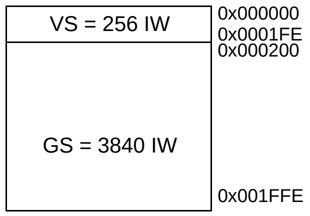
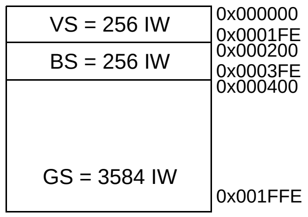
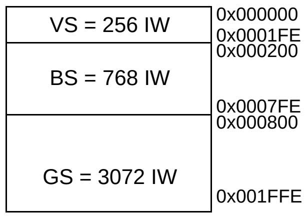
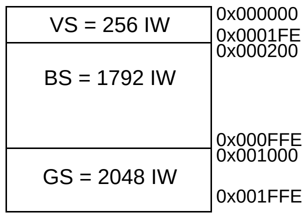
Legend: IW = Instruction Words

Table 23-9: Program Flash Segment Sizes for 16 KB Devices

BSS<2:0> = x11 0K	**BSS<2:0> = x10 1K**
 VS = 256 IW GS = 5376 IW 0x000000 0x0001FE 0x000200 0x002BFE 0x002C00 0x0057FE	 VS = 256 IW BS = 768 IW GS = 4608 IW 0x000000 0x0001FE 0x000200 0x0007FE 0x000800 0x002BFE 0x002C00 0x0057FE
BSS<2:0> = x01 4K	**BSS<2:0> = x00 8K**
 VS = 256 IW BS = 3840 IW GS = 1536 IW 0x000000 0x0001FE 0x000200 0x001FFE 0x002000 0x002BFE 0x002C00 0x0057FE	 VS = 256 IW BS = 5376 IW 0x000000 0x0001FE 0x000200 0x002BFE 0x002C00 0x0057FE

Legend: IW = Instruction Words

Table 23-10: Program Flash Segment Sizes for 12 KB Devices

CONFIG BITS	
BSS<2:0> = x11 0K	 VS = 256 IW GS = 3840 IW 0x000000 0x0001FE 0x000200 0x001FFE
BSS<2:0> = x10 256	 VS = 256 IW BS = 256 IW GS = 3584 IW 0x000000 0x0001FE 0x000200 0x0003FE 0x000400 0x001FFE
BSS<2:0> = x01 768	 VS = 256 IW BS = 768 IW GS = 3072 IW 0x000000 0x0001FE 0x000200 0x0007FE 0x000800 0x001FFE
BSS<2:0> = x00 1792	 VS = 256 IW BS = 1792 IW GS = 2048 IW 0x000000 0x0001FE 0x000200 0x000FFE 0x001000 0x001FFE

Legend: IW = Instruction Words

Note: The segment organizations shown in Table 23-3 through Table 23-10 are typical, but some devices may differ. To verify the memory segment sizes for different settings, refer to the specific device data sheets.

23.6.2 Selecting the Security Level of the Boot Segment

The security level of the Boot Segment is determined by the Configuration bit BSS2 (FBS<3>):

`1` = Standard security

`0` = High security

When the Boot Segment is configured for high security, the number of access methods is more limited than with standard security. The differences are noted in the following paragraphs. For additional information, refer to **Section 23.11 “Rules Concerning Program Flow”**.

23.6.3 Write Protection of the Boot Segment

The Boot Segment can be write-protected by programming Configuration bit BWRP (FBS<0>):

`1` = Boot segment can be written

`0` = Boot segment is write-protected

When write-protected, page erase and programming operations targeting the Boot Segment of program Flash are disabled. Setting the WR bit within the NVMCON Special Function Register will not start an operation. Erase operations that erase the entire Boot Segment are allowed; however, the Secure and General Segments are also erased.

23.6.4 Allocating Boot Segment RAM

The Boot Segment can also allocate a portion of the data RAM memory of the device for exclusive access by code executing within the Boot Segment. This protects the data integrity of algorithms executing within the Boot Segment.

If a Boot Segment is not allocated, BSS<2:0> = `x11`(FBS<3:1>), then a RAM segment cannot be allocated. The existence and size of Boot Segment RAM are determined by the RBS<1:0> (FBS<7:6>) Configuration bits.

One of the options is to exclude the Boot Segment RAM, which is the default option on an erased, non-programmed device.

Figure 23-4 shows that the Boot Segment RAM is located at the end of data RAM, or at the last location before the DMA memory area. The Boot Segment RAM starts at an address specified by the RBS<1:0> bits.

Note: DMA RAM is not present on all PIC24H devices. For DMA RAM availability and sizes, refer to the specific device data sheet.

Figure 23-4: Secure Segment Data RAM Allocation

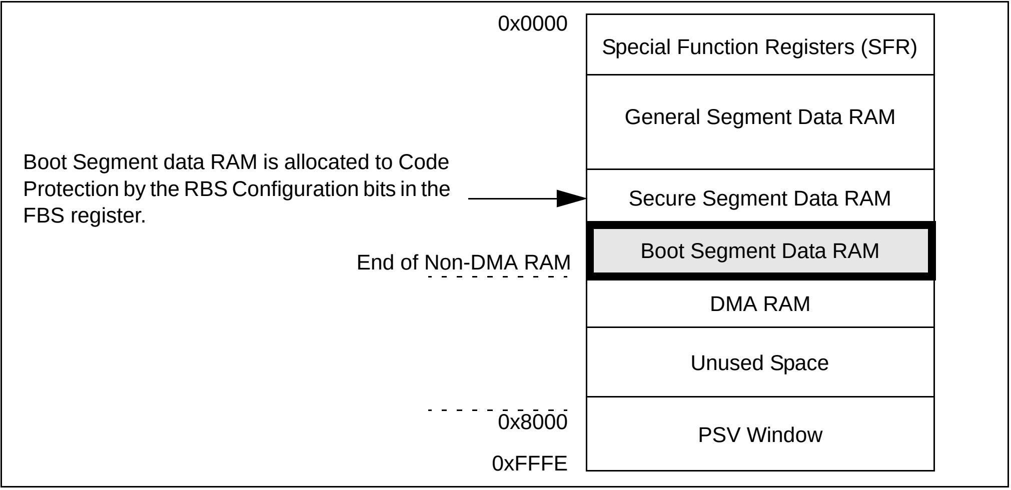

0x0000
Special Function Registers (SFR)
General Segment Data RAM
Boot Segment data RAM is allocated to Code Protection by the RBS Configuration bits in the FBS register.
Secure Segment Data RAM
Boot Segment Data RAM
End of Non-DMA RAM
DMA RAM
Unused Space
0x8000
PSV Window
0xFFFE

Table 23-11 shows an example of Boot Segment RAM size options for PIC24H devices. The start addresses listed are typical. For specific addresses, refer to the device data sheet.

Table 23-11: Boot Segment RAM Size Example

RBS<1:0>	BS Size / Bytes	BS Start Address	BS End Address
11	No Boot Segment		
10	Small/128	EOM-0x007F	EOM
01	Medium/256	EOM-0x00FF	EOM
00	Large/1024	EOM-0x03FF	EOM

Note: EOM refers to the last location of data RAM excluding DMA RAM.

23.6.5 Run Time Release of Boot Segment RAM

When an algorithm within the Boot Segment completes its task and is preparing to return execution to code within a lower priority segment, it can be helpful to release some of the RAM allocated to the Boot Segment. The Boot Segment RAM control Special Function Register (SFR) contains the RL_BSR (BSRAM<0>) bit (refer to Register 23-2). When this bit is set, the system releases a portion of the BS RAM back to the next lower priority segment defined. Table 23-12 is an example of RAM mapped with RL_BSR = '0' and RL_BSR = '1'.

Table 23-12: Boot Segment RAM Release

RBS<1:0>	BS Size when:	
	RL_BSR = 0	RL_BSR = 1
11	No Boot Segment	
10	Small	No Boot Segment
01	Medium	Small
00	Large	Medium

23.6.6 Releasing the Secure RAM for General Use

The secure code segments can release some allocation of its secure RAM for general use at any time during operation. For example, the minimum allocation can be reserved for the BS to store sensitive volatile variables during General Segment code run time. Then, when the code branches to the BS segment to execute an algorithm, the BS code can clear the RL_BSR bit to secure the maximum allocation of the secure RAM for its use. After the BS code execution completes, it can set the RL_BSR bit to again minimize the allocated secure RAM.

Both the Boot and Secure Segments have an associated BSRAM and SSRAM register, which contains the RL_BSR or RL_SSR bits respectively. Only BS segment has write access to the BSRAM register, and only Secure Segment has write access to the SSRAM register.

Note: On any reset, the maximum allocations are in a secure state because the RL_BSR and RL_SSR bits are Reset.

The RAM security bits determine whether RAM is secured or not.

- If RSS<1:0> = 11 (FSS<7:6>), no boot RAM is allocated and the RL_SSR bit is ignored
- If RBS<1:0> = 11 (FBS<7:6>), no boot RAM is allocated and the RL_BSR bit is ignored

Register 23-2: BSRAM: Boot Segment RAM Special Function Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R/W-0
—	—	—	—	—	IW_BSR	IR_BSR	RL_BSR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2 **IW_BSR** (Read-Only Status bit)
`0` = No illegal write has been attempted since this register was last read
`1` = At least one illegal write has been attempted since last read of this register

IW_BSR bit is cleared on any Reset. It is also cleared ***AFTER*** the BSRAM register is read while executing in BS.

bit 1 **IR_BSR** (Read-Only Status bit)
`0` = No invalid read of protected BS RAM section has occurred since this register was last read
`1` = At least one invalid read has occurred since last read of this register

IR_BSR bit is cleared on any Reset. It is also cleared ***AFTER*** the BSRAM register is read while executing in BS.

bit 0 **RL_BSR**
`0` = BSRAM is held secure for BS only
`1` = BS has released the secure RAM for general use. All but the highest 128 bytes are released

RL_BSR bit is cleared to zero on any Reset.

23.7 THE SECURE SEGMENT

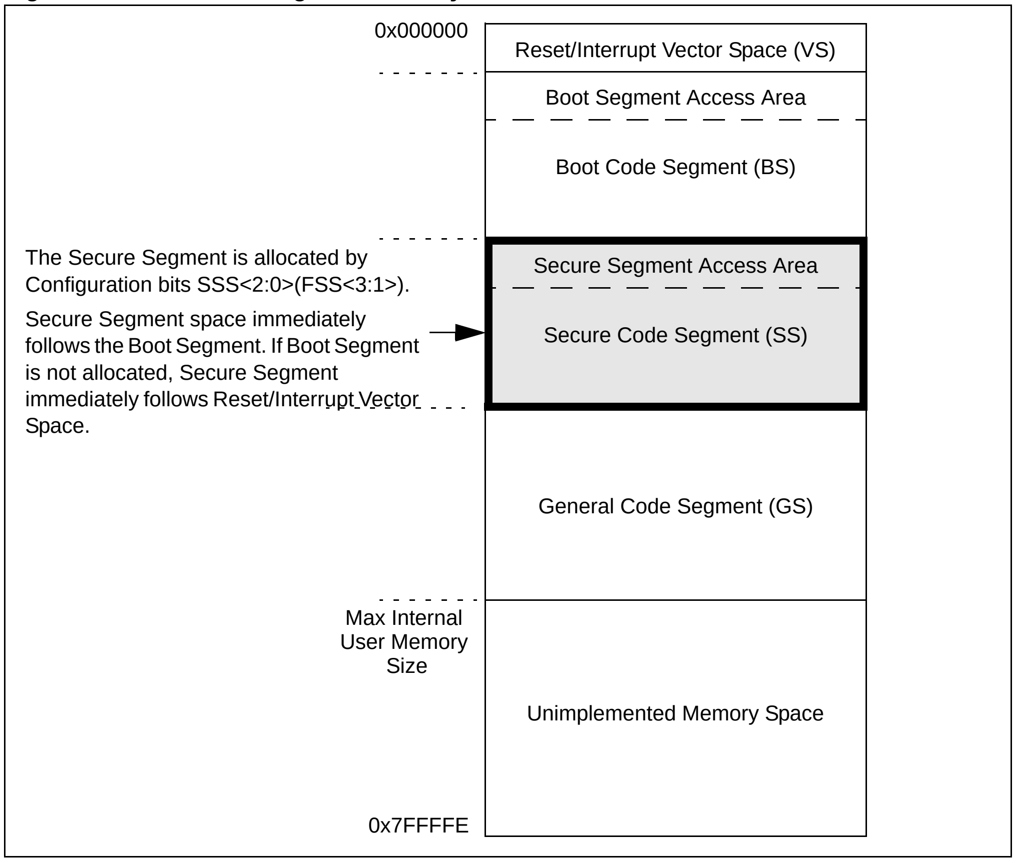
The Secure Segment has the second highest privilege and is ideal for storing proprietary algorithm routines. Access to the Secure Segment from lower priority segments are limited to "calls" to the Secure Segment.

23.7.1 Allocating the Secure Segment

The Secure Segment is allocated by the Configuration bits SSS<2:0> (FSS<3:1>).

Figure 23-5 shows the Secure Segment begins immediately following the Boot Segment. If there is no Boot Segment, the Secure Segment starts at the end of the Reset/Interrupt Vector Space. The default is to exclude a Secure Segment.

Figure 23-5: Secure Segment Memory Allocation


0x000000
Reset/Interrupt Vector Space (VS)
Boot Segment Access Area
Boot Code Segment (BS)
The Secure Segment is allocated by Configuration bits SSS<2:0>(FSS<3:1>).
Secure Segment space immediately follows the Boot Segment. If Boot Segment is not allocated, Secure Segment immediately follows Reset/Interrupt Vector Space.
Secure Segment Access Area
Secure Code Segment (SS)
General Code Segment (GS)
Max Internal User Memory Size
Unimplemented Memory Space
0x7FFFFE

The Secure Segment continues to an address specified by the SSS<2:0> bits. Table 23-13 shows an example of Secure Segment options (in this case, for devices with 64 KB flash memory). The end addresses listed in these tables are typical. Refer to the device data sheet for specific addresses for a given device.

Table 23-13: Secure Segment Size Example

SSS<2:0>	Security Level	SS Size	SS Start Address	SS End Address
`x11`	No Secure Program Flash Segment			
`110`	Standard	Small	E.O. BS + 1	0x001FFE
`010`	High	Small	E.O. BS + 1	0x001FFE
`101`	Standard	Medium	E.O. BS + 1	0x003FFE
`001`	High	Medium	E.O. BS + 1	0x003FFE
`100`	Standard	Large	E.O. BS + 1	0x007FFE
`000`	High	Large	E.O. BS + 1	0x007FFE

Note: E.O. BS refers to the last location of the Boot Segment.

Register 23-3: FSS: Secure Segment Configuration Register Byte

R/P	R/P	U	U	R/P	R/P	R/P	R/P
RSS1	RSS0	R	R	SSS2	SSS1	SSS0	SWRP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-6 **RSS<1:0>:** Secure Segment RAM Code Protection bits(1)
`11` = No Secure RAM defined
`10` = Secure RAM is 256 Bytes less BS RAM
`01` = Secure RAM is 2048 Bytes less BS RAM
`00` = Secure RAM is 4096 Bytes less BS RAM

bit 5-4 **Reserved:** Programming values will have no effect

bit 3-1 **SSS<2:0>:** Secure Segment Program Flash Code Protection bits(2)
`x11` = No Secure program Flash segment
`110` = Standard security, Small Secure Segment
`010` = High security, Small Secure Segment
`101` = Standard security, Medium Secure Segment
`001` = High security, Medium Secure Segment
`100` = Standard security, Large Secure Segment
`000` = High security, Large Secure Segment

For device specific information see Table 23-7.

bit 0 **SWRP:** Secure Segment Program Flash Write Protection
`1` = Secure Segment can be written
`0` = Secure Segment is write-protected

Note 1: Not all devices have Secure Segment RAM code protection. For device specific information refer to Table 23-3, Table 23-3 and Table 23-4.

2: The exact definitions of Small, Medium, and Large Secure Segment vary from one device to another. For device specific information refer to Table 23-5, Table 23-6 and Table 23-7.

3: If a Secure Segment is not needed, the SWRP bit must be programmed as a “`1`”.

23.7.2 Selecting the Security Level of the Secure Segment

The security level of the secure code segment is determined by the Configuration bit SSS2 (FSS<3>):

`1` = Standard security

`0` = High security

When the Secure Segment is configured for high security, the number of access methods is more limited than with standard security. The differences are noted in the following paragraphs.

23.7.3 Write Protection of the Secure Segment

The Secure Segment can be write-protected by programming the SWRP (FSS<0>) Configuration bit.

`1` = Secure Segment can be written

`0` = Secure Segment is write-protected

When write-protected, or page erase and programming operations targeting the Secure Segment of program Flash are disabled, setting the WR bit within the NVMCON Special Function Register will not start an operation. Erase operations that erase the entire Secure Segment are allowed; however, the General Segment is also erased.

23.7.4 Allocating Secure Segment RAM

The Secure Segment can also allocate a portion of the data RAM for code protection. However, if a Secure Segment is not allocated, SSS<2:0> = `x11(`FSS<3:1>), then a RAM segment cannot be allocated. The existence and size of Secure Segment RAM are determined by the RSS<1:0> (FSS<7:6>) Configuration bits.

One of the options is to exclude the Secure Segment RAM, which is the default option on an erased, non-programmed device.

The Secure Segment RAM ends at the last location before the Boot Segment RAM. The Secure Segment RAM starts at an address specified by the RSS<1:0> bits.

Figure 23-6: Secure Segment Data RAM Allocation

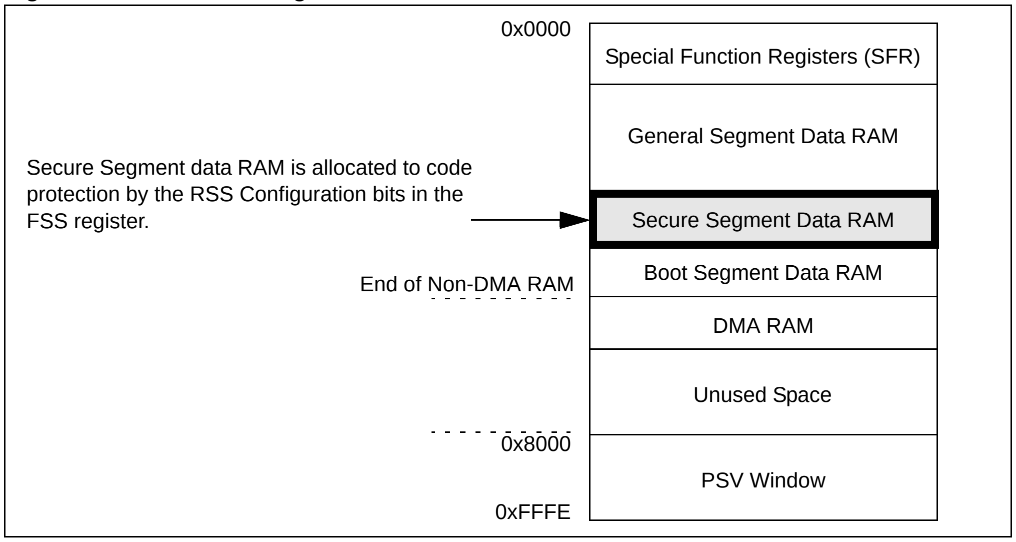

0x0000
Special Function Registers (SFR)
General Segment Data RAM
Secure Segment data RAM is allocated to code protection by the RSS Configuration bits in the FSS register.
Secure Segment Data RAM
Boot Segment Data RAM
End of Non-DMA RAM
DMA RAM
Unused Space
0x8000
PSV Window
0xFFFE

Table 23-14 shows an example of Secure Segment RAM allocations. The start addresses listed are typical. Refer to the device data sheet for specific addresses for a given device.

Table 23-14: Secure Segment RAM Size Example

RSS<1:0>	SS Size	SS Start Address	SS End Address
11	No Secure Segment		
10	Small	S.O.BS-0x0100	S.O.BS-1
01	Medium	S.O.BS-0x0800	S.O.BS-1
00	Large	S.O.BS-0x1000	S.O.BS-1

Note: S.O. BS refers to the first location of basic segment data RAM.

23.7.5 Run Time Release of Secure Segment RAM

Like the Boot Segment, the Secure Segment can allocate and release RAM. The Secure Segment RAM control Special Function Register (SFR) contains the RL_SSR (SSRAM<0>) bit (refer to Register 23-4). When this bit is set, the system releases a portion of the Secure Segment RAM back to the next lower priority segment defined. Table 23-15 is an example of RAM mapped with RL_SSR = '0' and RL_SSR = '1'.

Table 23-15: Secure Segment RAM Release

RSS<1:0>	Secure Segment Size When:	
	RL_SSR = 0	**RL_SSR = 1**
11	No Secure Segment	
10	Small	No Secure Segment
01	Medium	Small
00	Large	Medium

23.7.6 Releasing Secure Segment RAM for General Use

The secure code segments can release some allocation of its secure RAM for general use at any time during operation. For example, the minimum allocation can be reserved for the Secure Segment to store sensitive volatile variables during General Segment code run time. When the code branches to the Secure Segment to execute an algorithm, the Secure Segment code can clear the RL_SSR bit to secure the maximum allocation of the secure RAM for its use. After the Secure Segment code execution completes, it can set the RL_SSR bit to again minimize the allocated secure RAM.

Both the Boot and Secure Segments have an associated BSRAM and SSRAM register, which contains the RL_BSR or RL_SSR bits, respectively. Only the Boot Segment has write access to the BSRAM register, and only the Secure Segment has write access to the SSRAM register.

Note that on any Reset, because the RL_SSR bit is Reset, the maximum allocations are in a secure state.

The RAM security bits determine whether RAM is secured or not. If RSS<1:0> = 11 (FSS<7:6>), no secure RAM is allocated and the RL_SSR bit is "don't care."

Register 23-4: SSRAM: Secure Segment RAM Special Function Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R-0	R-0
—	—	—	—	—	IW_SSR	IR_SSR	RL_SSR
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2 **IW_SSR** (Read-Only Status bit)
0 = No illegal write of protected SSRAM has been attempted since this register was last read
1 = At least one illegal write has been attempted since last read of this register

IW_SSR bit is cleared on any Reset. It is also cleared ***AFTER*** SSRAM register is read while executing in Secure Segment.

bit 1 **IR_SSR** (Read-Only Status bit)
0 = No invalid read of protected SSRAM has occurred since this register was last read
1 = At least one invalid read has occurred since last read of this register

IR_SSR bit cleared on any Reset. It is also cleared ***AFTER*** SSRAM register is read while executing in Secure Segment.

bit 0 **RL_SSR**
0 = SSRAM is held secure for Secure Segment only
1 = Secure Segment has released the secure RAM for general use. All but the highest 128 bytes are released

RL_SSR bit is cleared to zero on any Reset.

23.8 THE GENERAL SEGMENT

The General Segment has the lowest security privilege level. The General Segment is intended to contain the majority of the application code. Its size is essentially the on-chip memory minus the Boot and Secure Segments. If there are no Boot or Secure Segments, the General Segment uses all of the on-chip memory.

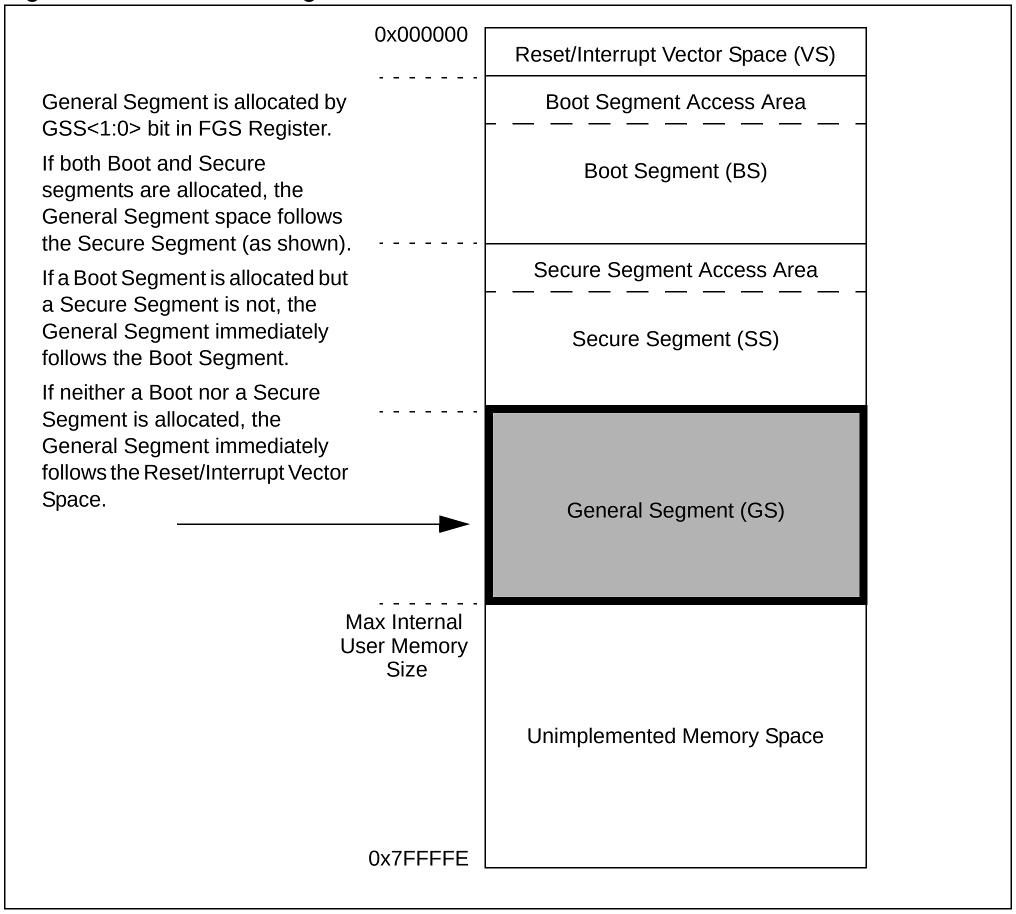
23.8.1 Allocating the General Segment

The General Segment always exists, regardless of whether Boot and Secure Segments are allocated. It is specified by the GSS<1:0> Configuration bit in the FGS register. The location of the General Segment depends on the existence of the Boot and Secure Segments.

Figure 23-7 shows that if both Boot and Secure Segments are allocated, the General Segment immediately follows the Secure Segment. If a Boot Segment is allocated, but a Secure Segment is not, the General Segment immediately follows the Boot Segment. If neither a Boot Segment nor a Secure Segment is allocated, the General Segment immediately follows the Reset/Interrupt Vector Space (VS).

The device default is to exclude Boot and Secure Segments. By default, the entire program memory is allocated as General Segment.

Figure 23-7: General Segment Allocation


0x000000
Reset/Interrupt Vector Space (VS)
Boot Segment Access Area
Boot Segment (BS)
Secure Segment Access Area
Secure Segment (SS)
General Segment (GS)
Max Internal User Memory Size
Unimplemented Memory Space
0x7FFFFE
General Segment is allocated by GSS<1:0> bit in FGS Register.
If both Boot and Secure segments are allocated, the General Segment space follows the Secure Segment (as shown).
If a Boot Segment is allocated but a Secure Segment is not, the General Segment immediately follows the Boot Segment.
If neither a Boot nor a Secure Segment is allocated, the General Segment immediately follows the Reset/Interrupt Vector Space.

Register 23-5: FGS: General Segment Configuration Register

U	U	U	U	U	R/P	R/P	R/P
R	R	R	R	R	GSS1	GSS0	GWRP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7-3	**Reserved:** Used for silicon validation purposes
bit 2-1	**GSS<1:0>:** General Segment Program Flash Code Protection bits 11 = General Segment not protected 10 = Standard security; general program Flash segment starts at end of SS and ends at EOM 0X = High security; general program Flash segment starts at end of SS and ends at EOM
bit 0	**GWRP:** General Segment Program Flash Write Protection 1 = General segment can be written 0 = General segment is write-protected

23.8.2 Selecting the Security Level of the General Segment

Depending on the device, there are up to three levels of security to choose for the General Segment. Refer to the specific device data sheet to determine how many options are available.

Configuration bits GSS<1:0> (FGS<2:1>) determine the level of protection for this segment:

`11` = No protection

`10` = Standard security

`0X` = High security

23.8.3 Write Protection of the General Segment

The General Segment can be write-protected by programming the GWRP (FGS<0>) Configuration bit, similarly to write protecting the Boot Segment:

`1` = General code segment can be written

`0` = General code segment is write-protected

23.9 THE RESET, TRAP AND INTERRUPT SERVICE ROUTINE VECTOR SPACE

The first 256 instruction words are reserved for the RESET instruction, trap, and interrupt vectors.

Protection of this segment depends on the state of the BSS<2:0> (FBS<3:1>) and GSS<1:0> (FGS<2:1>) or GCP (FGS<1>) code protection bits. If a Boot Segment is allocated, the Vector Space protection is the same as the Boot Segment. In other words, if a Boot Segment is defined, erase and programming operations of the Vector Space can only be performed via Boot Segment code. If a Boot Segment is not allocated, the Vector Space protection is the same as the General Segment and erase and programming operations of the Vector Space can be performed via General Segment code.

A write to this segment is enabled or disabled by the BWRP bit if a Boot Segment is allocated, or by the GWRP bit if a Boot Segment is excluded.

23.10 DEFINITION OF SECURITY PRIVILEGES

It is important to understand the relative privilege levels of the three code protection segments. Operations can be described as being relative to higher or lower privilege segments. The Boot Segment has the highest privilege level and can directly access code in the lower segments. The Secure Segment can directly access code in the General Segment, but can only issue calls to code in the Boot Segment. The General Segment can only access code from either of the higher segments by issuing calls.

Rules governing access privileges are discussed in sections **23.11 "Rules Concerning Program Flow"** through **23.14.1 "Rules for Programming Devices in RTSP"**. Table 23-16 presents a summary overview of these rules during normal run-time operation.

Table 23-16: Privileged Operations Rules Summary

Target Segment		**General Segment**						**Secure Segment**				**Boot Segment**				**IVT and AIVT**					
Protection Level		None		Standard		High		Standard		High		Standard		High		None		Standard		High	
Write-Protected		No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Requested Operation (Yes/No)																					
PC Rollover into Target Segment		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A	N/A	n/a **(Note 3)**					
PFC from reset vector instruction to Target Segment **(Note 5)**		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Note 2**	**Note 2**	Yes	Yes	**Note 2**	**Note 2**	**Note 4**					
VFC (Vector Flow Change) to Target Segment **(Note 5)**		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Note 2**	**Note 2**	Yes	Yes	**Note 2**	**Note 2**	**Note 4**					
PFC from BS to Target Segment **(Note 1)**		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Note 4**					
PFC from SS to Target Segment **(Note 1)**		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Note 2**	**Note 2**	**Note 4**					
PFC from GS to Target Segment **(Note 1)**		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Note 2**	**Note 2**	Yes	Yes	**Note 2**	**Note 2**	**Note 4**					
R/W of Target Segment RAM while executing from: Note: Stack assumed to be in GS RAM space, access needed	BS	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	n/a					
	SS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No						
	GS	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No						
Table Read/PSV of Target Segment Program Flash while executing from: **(Note 7)**	BS	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	SS	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
	GS	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Table Write of Target Segment (load write latches)		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Program/Erase row of Target Segment program Flash while executing from:	BS	Yes	No	Yes	No	No	No	Yes	No	No	No	Yes	No	Yes	No	Yes	No	Yes	No	No	No
	SS	Yes	No	Yes	No	No	No	Yes	No	Yes	No	No	No	No	No	**Note 6**	No	**Note 6**	No	**Note 6**	No
	GS	Yes	No	Yes	No	Yes	No	No	No	No	No	No	No	No	No	**Note 6**	No	**Note 6**	No	**Note 6**	No
Erase Target Segment data flash while executing from:	BS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	n/a					
	SS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
	GS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
Erase All		Command not valid in RTSP mode																			
Erase BS Segment/code-protect		Erase GS/SS/BS/VS segments and GS/SS/BS code protection fuses																			
Erase SS Segment/code-protect		Erase GS/SS segments and GS/SS code protection fuses Erase VS if no Boot Segment defined																			
Erase GS Segment/code-protect		Erase GS segment and GS code protection fuses Erase VS if no Boot Segment defined																			
Erase GS Segment		Erase GS Segment Only																			
Program configuration register		Yes																			

Note 1: PFC (Program Flow Change) is defined as when the PC is loaded with a new value instead of the normal automatic increment. It includes `JUMP`, `CALL`, `RETURN`, `RETFIE`, Computed Jump etc.
2: PFC is allowed only to the first 32 instruction locations of the segment.
3: Since execution is not permitted in the VS segment, this condition is not possible.
4: A PFC operation (i.e., branch, call etc.) into the IVT and AIVT segment is possible. But as soon as execution is attempted out of this segment an illegal address trap will result (unless pointed to Reset vector at address 0x000000).
5: VFC (Vector Flow Change) is defined as when the PC is loaded with a Interrupt or trap vector address.
6: Operation allowed if there is no higher security privilege-segment defined.
7: `TBLRD` or `DS` read will execute but return all `0`s if not allowed.

23.11 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. When code protection is implemented, program flow conforms to privilege level. That is, a program executing from code-protected memory can flow from a higher security segment to a lower segment, but not vice versa. For example, a program executing from the Secure Segment can flow into the General Segment but not into the Boot Segment.

Program Flow Change (PFC) occurs when the Program Counter is reloaded as a result of Call, Jump, Computed Jump, Return, Return from Subroutine, or other form of branch instruction. A PFC allows the program flow to follow an alternate path. A normal PFC only allows the program to branch within the same segment. A Restricted PFC allows the program to branch to a special Segment Access Area of a higher security segment.

Vector Flow Change (VFC) occurs when the Program Counter is reloaded with an Interrupt or Trap vector.

Jumping into secure code at unintended locations can expose code to algorithm detection. Therefore, PFC and VFC operations are restricted if they violate the privilege hierarchy.

Figure 23-8: Program Flow Rules

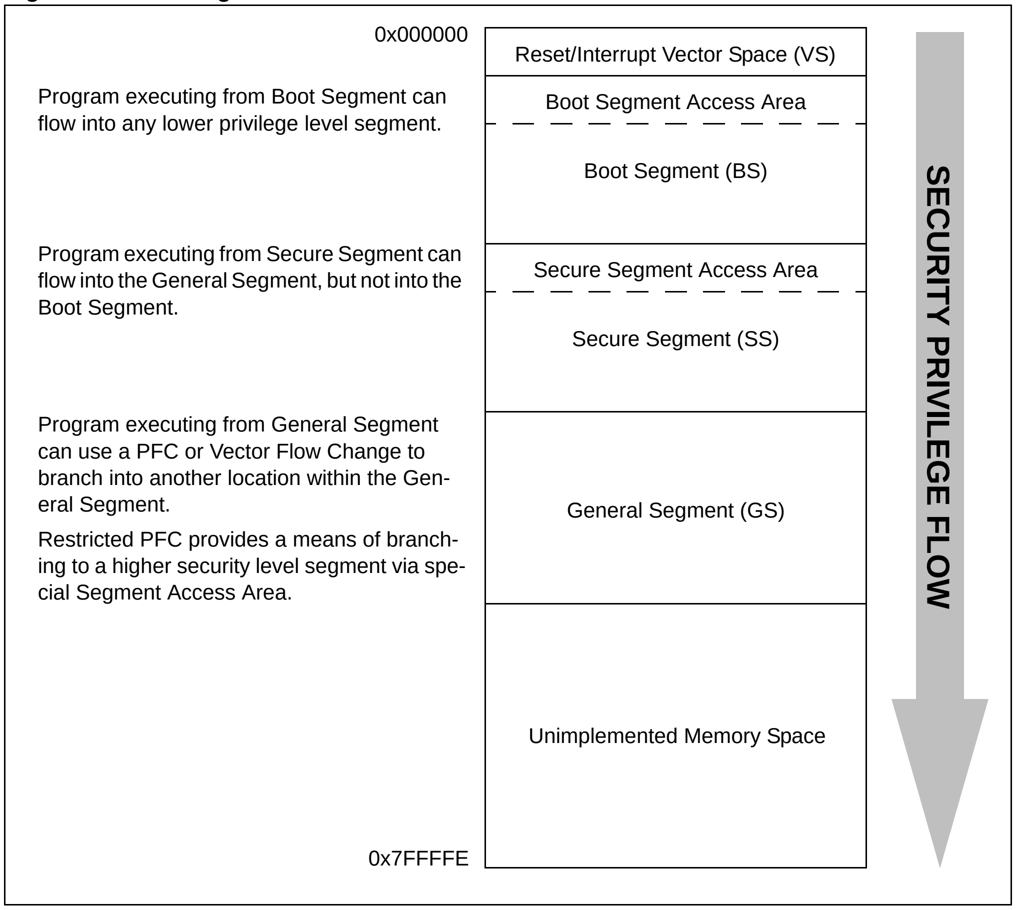

0x000000
Program executing from Boot Segment can flow into any lower privilege level segment.
Program executing from Secure Segment can flow into the General Segment, but not into the Boot Segment.
Program executing from General Segment can use a PFC or Vector Flow Change to branch into another location within the General Segment.
Restricted PFC provides a means of branching to a higher security level segment via special Segment Access Area.
0x7FFFFE
Reset/Interrupt Vector Space (VS)
Boot Segment Access Area
Boot Segment (BS)
Secure Segment Access Area
Secure Segment (SS)
General Segment (GS)
Unimplemented Memory Space
SECURITY PRIVILEGE FLOW

Table 23-17 presents an overview matrix of possible operations between program memory segments. Blank cells indicate conditions where read/write/erase and PFC operations can not be performed. When the Boot Segment or Secure Segment is implemented with high security level, a PFC from a lower privilege level must be restricted to the segment access area. Any PFC attempt outside the segment access area results in a security Reset (refer to **Section 23.11.3 “Program Flow Errors”**).

Table 23-17: Possible Operations Between Program Memory Segments

Code Executed From:		**Operation To:**						
		Boot Segment Security Level		**Secure Segment Security Level**		**General Segment Security Level**		
		Standard	**High**	**Standard**	**High**	**Standard**	**High**	**None**
BS	Standard	R, P, PFC		R, P, PFC	PFC*	R, P, PFC	PFC	R, P, PFC
	High		R, P, PFC	R, P, PFC	PFC*	R, P, PFC	PFC	R, P, PFC
SS	Standard	PFC	PFC*	R, P, PFC		R, P, PFC	PFC	R, P, PFC
	High	PFC	PFC*		R, P, PFC	R, P, PFC	PFC	R, P, PFC
GS	Standard	PFC	PFC*	PFC	PFC*	R, P, PFC		
	High	PFC	PFC*	PFC	PFC*		R, P, PFC	
	None	PFC	PFC*	PFC	PFC*			R, P, PFC

Legend: R – Read
P – Program (write)/Erase
PFC – Program Flow Change allowed to anywhere in the segment
PFC* – Restricted Program Flow changes (can branch to Segment Access areas only)

23.11.1 Flow Changes

PFCs within a segment are unrestricted. Generally, PFC and VFC changes from one segment to another segment are not restricted, except as follows.

To ensure the integrity of the operations of code within the Boot and Secure Segments, the user must restrict program flow options to those segments. Program flow can be limited to only allow the segment access areas to be a branch target. The segment access areas are the first 32 instruction locations of the Boot Segment or Secure Segment code space.

If the security level of the Boot Segment or Secure Segment is high, and a PFC originates from a lower priority segment, the target of the PFC must be within the access area.

If the security level of Boot Segment or Secure Segment is high, and a VFC occurs, the target of the VFC must be within the segment access area.

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

23.11.2 Reset Instruction

A typical device Reset will reset the PC to 0x000000, then begin execution at that location. There must be a branch instruction at locations 0x000000-0x000002 that branch to beginning of code.

The instruction at the Reset vector can branch to any location unless the Boot Segment or Secure Segment is in high security, then the target must be to the specific segment access area. Typically, the target of the RESET instruction is to the General Segment code space.

23.11.3 Program Flow Errors

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 program flow or vector flow change 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 attempted, it results in an address error trap.

23.11.4 Re-Targeting Reset After Boot

The Reset operation is independent of which segment the device is operating in when a reset occurs. To prevent code probing, the Reset vector is protected when a Boot Segment is allocated. If a Boot Segment exists, the Vector Space, including the Reset vector, is part of the Boot Segment and restricted by Boot Segment rules. If a Boot Segment does not exist, the Vector Space, including the Reset vector, is part of the General Segment and can be modified by the General Segment.

For example, assume that a part has a boot sector that contains a boot loader. At Reset, initially, the device Resets to a location within the boot loader. The boot loader will run and load user code into the General Segment. When this operation is complete, the boot loader can rewrite the Reset vector instruction to point to the user code. At the next Reset, the Reset will go to the user code; however, the user code cannot then rewrite the Reset vector instruction.

23.12 RULES CONCERNING INTERRUPTS

23.12.1 Interrupts and Traps In Secure Modes

Interrupt handling is restricted for the following reasons:

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

23.12.1.1 BS AND SS INTERRUPT VECTORS

If an interrupt occurs while the program is running in the Boot or Secure Segment, the processor obtains the interrupt vector from the special Boot Segment interrupt vector location at (BS + 0x20), or the special Secure Segment interrupt vector at (SS + 0x20).

Note: There are two special interrupt vectors: one within the Boot Segment and the other within Secure Segment. Interrupts occurring when code executes in one of these segments causes the processor to vector to the special interrupt vector for that segment. Users may employ a special Interrupt Service Routine (ISR) within the protected segment to hide critical data, and then manually vector to the real ISR by reading the INTTREG SFR.

23.12.1.2 INTERRUPT AND TRAP HANDLING SEQUENCE

The sequence for handling interrupts and traps is as follows:

1. Interrupt or trap occurs while code is executing in a Secure Segment, for example in the Boot Segment
2. Return address is pushed on the Stack
3. The contents of location (BS + 0x20) are loaded into the Program Counter instead of the usual interrupt vector
4. Special ISR is executed at address pointed to by (BS + 0x20)
5. Sensitive information from W registers is stored to the secure RAM area, whether this is the Boot Segment or Secure Segment RAM space
6. Actual return address is retrieved from Stack and saved in secure RAM
7. Actual return address is replaced with new return address. For example, BS + 0x30. BS + 0x30 is located in the BS to BS + 0x3E address range. Keep in mind that this range is where Program Flow Change is allowed from outside Boot Segment
8. INTTREG SFR is read to determine which interrupt vector to jump to
9. Interrupt vector is read from the vector table and executes an indirect jump
10. Users ISR begins execution
11. User code executes
12. Return from interrupt (back to location BS + 0x30)
13. Read actual return address from secure RAM area
14. Restore W registers
15. Execute indirect jump to go back to Boot Segment

Table 23-18: Vector Operations In Normal User Mode

Vector Operation	Result in Normal User Mode
Hardware Interrupt while in BS	Obtain vector from special BS ISR vector location at BS + 0x20
Hardware Interrupt while in SS	Obtain vector from special SS ISR vector location at SS + 0x20
Hardware Interrupt while in GS	Obtain vector from normal ISR vector location
Software Interrupt and Trap while in BS	Obtain vector from special BS ISR vector location at BS + 0x20
Software Interrupt and Trap while in SS	Obtain vector from special SS ISR vector location at SS + 0x20
Software Interrupt and Trap while in GS	Obtain vector from normal ISR vector location

23.13 RULES FOR ACCESSING RAM DATA

23.13.1 Using Segment RAM

If the Boot Segment or the Secure Segment has allocated protected RAM space, that RAM is not accessible by code running outside of that segment. For example, code running in the Secure or General Segment cannot access RAM protected by the Boot Segment.

If an instruction does an unauthorized read of a protected RAM location, the read operation and the instruction occur; however, the resulting write of the instruction is disabled. For example, "`mov ssram, w0`" will read from the SSRAM location, however the write to `w0` will not occur. The results from the ALU are zeroed out and the write does not occur.

An unauthorized read of a protected RAM location, within the boot RAM segment, sets the IR_BSR (BSRAM<0>) bit. This bit remains set until a device Reset or until the BSRAM register is read by code executing from within the Boot Segment.

Similarly, a unauthorized read from the secure RAM segment sets the IR_SSR (SSRAM<0>) bit. This bit remains set until a device Reset or until the SSRAM register is read by code executing from within the Secure Segment.

An unauthorized write to a protected RAM location causes a write of '`0`' to the protected location.

An unauthorized write of a protected RAM location will set either the IW_BSR (BSRAM<2>) bit or IW_SSR (SSRAM<1>) bit to the unauthorized read status bits. They are cleared by a device Reset or until the BSRAM or SSRAM register is read by code executing from within the Boot or Secure Segment respectively.

23.13.2 Stack Allocation

The user can allocate the stack space anywhere in the RAM; however, the General Segment RAM is the only area that both the Boot and Secure Segment can access. Therefore, the stack should be allocated in the General Segment RAM area.

If the stack accidentally intrudes into the "BSRAM" or "SSRAM" area, then writes due to push operations and reads due to pop operations will act as unauthorized RAM accesses described in the previous section.

This can and should be prevented by using the Stack Pointer Limit register (SPLIM) and setting the contents of the SPLIM to an appropriate address.

23.13.3 Register Dumping Protection

The device initializes W registers on all resets. Data RAM is not initialized and resets can leave valid data in data RAM. RAM contents security must be maintained.

23.14 SECURITY FEATURES AND DEVICE OPERATIONAL MODE

Security functions are dependant on the operational mode of the device. Each device can operate in one of following modes:

- In Run-Time Self-Programming (RTSP) mode (normal device operation), the application code is running and the application code can invoke self programming.
- In In-Circuit Serial Programming™ (ICSP™) mode, the programming mode provides native, low-level programming capability to erase, program and verify the chip. The device is under the command of a device programmer such as a PRO MATE® 3 or MPLAB® ICD 2.

23.14.1 Rules for Programming Devices in RTSP

The device programs itself by using erase commands to first clear a portion of the code. It then writes the new code or data into the write latches, and finally uses a programming command to program the write latch contents into the Flash array. Erase or programming commands are specified by the device specific NVMCON Special Function Register. The NVMOP bit field selects the particular function and the ERASE bit selects between programming and erase functions. The WR bit within the NVMCON register invokes programming operations. Consequently, to protect code integrity, the device restricts the operations that occur on setting the WR bit.

23.14.1.1 ERASING AND PROGRAMMING CODE ROWS OR PAGES

Depending on the implementation of the Flash array, the NVMOP specifies erasing or programming a page of the program Flash array.

- If segment write protection is enabled, then no erase or programming operations occurs within that segment.
- Code running within a segment can erase or program part of its own segment.
- Code running within higher priority segments can erase or program part of a lower priority segment, unless the lower priority segment has selected high security.
- If the Vector Space has inherited high security associated with the Boot or General Segment, then no segment can erase or program part of the Vector Space. If a Boot Segment is defined, only the Boot Segment can erase or program part of the Vector Space. If no Boot Segment is defined, any segment can erase or program part of the Vector Space.

23.14.1.2 ERASING A SEGMENT AND CLEARING CODE PROTECTION

There are several variants of `NVMOP` commands that will erase an entire segment of program Flash, erase all lower priority segments of program Flash and clear the code protection Configuration bits in one operation. This is the only way to release code protection on a segment. These commands can be executed when running from any segment. These commands will not erase any contents from the segment RAM.

23.14.2 Rules for Programming Devices Using ICSP

When the device is connected to a device programmer, the allowable operations are limited to erasing, programming and verifying the device code memory.

- The device programmer uses segment erase commands to erase the device and clear the code protection.
- Programming commands are ignored if any level of code protection is selected. To program, there must be no Boot Segment or Secure Segment specified, and the General Segment must have no code protection.
- Devices with any level of code protection cannot be verified. Attempts to verify code-protected devices results in reading '`0`'s.

Once the device is programmed with the desired code, 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 erasing and clearing the code protection once more.

23.15 TYPICAL PROCEDURES FOR BOOT LOADING A DEVICE

A typical scenario for boot loading a device using code protection of these devices, is a system upgraded in the field. Here, the device uses two segments, the Boot Segment and the General Segment. The General Segment contains the application. The Boot Segment contains a secure boot loader. Both segments have high security enabled.

At system Reset, the device vectors to the application in the General Segment.

As the system is operating in the field, a technician connects a reprogramming tool to the system. The application recognizes this connection and branches to a location within the Boot Segment access area. This branch is highly secure and the attempt to modify this branch likely results in a device Reset.

The Boot Segment contains code that allows encrypted communication with the tool. The encryption keys are safe within the contents of the Boot Segment code because only the Boot Segment can access them. If serialized programming is used when the boot loader was initially programmed into the system, the encryption key could be specific to a particular system, further enhancing the strength of the encrypted communication.

Once the boot loader verifies valid communication with the external programming tool, it can then erase the code within the General Segment and clear the General Segment code protection.

The boot loader then receives the encrypted code update from the tool, decrypts it and programs it into the general space.

As the boot loader is running, it is immune from disruption from interrupts or traps as it can vector those to a secure location within the boot loader itself.

As the boot loader finishes, it can program the Configuration bits to reprotect the general space, make any necessary updates to the vectors and then return to the general application.

23.16 TYPICAL INSTALLATION OF THIRD PARTY PROTECTED ALGORITHM

In this scenario, the system integrator is purchasing an algorithm from a third party vendor. Here, the system integrator wants to protect his system code from problems with the third party algorithm, and the third party algorithm vendor wants to ensure that no one in the system integrators company compromises his code.

Normally, this is a matter of trust for the third party vendor, since normally the third party vendor must provide the integrator native code for the integrator to link into his system code and program into the device.

If the third party vendor can provide his code in an encrypted fashion, and if the device can isolate the third party code, the third party vendor can ensure his code remains proprietary.

In this scenario, the device allocates Boot, Secure and General segments. The system integrator's code resides in the General Segment and a boot loader similar to the one described in the previous section is contained in the Boot Segment.

As the system integrator builds his system, his boot loader and application code are programmed into the device.

A special loader, provided by the third party vendor, is programmed into the Secure Segment. This loader decrypts and programs the third party's algorithm using a key provided by the third party vendor to the system integrator. Once the Secure Segment is protected, the algorithm inside is not accessible by the system integrators code in the Boot or General Segments. The third party code can only be accessed through calls to the Secure Segment access area.

The protected algorithm should maintain critical data parameters within protected RAM area. When the algorithm is about to finish and return to code within another segment, it should "cleanse" its RAM area.

23.17 DESIGN TIPS

Question 1: ***Can I boot load 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 boot loader that might be resident within the General Segment.

This limits the options for booting but does not prevent it. The boot loader 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 boot loader 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.

23.18 RELATED APPLICATION NOTES

This section lists application notes related to this section of the manual. These application notes may not be written specifically for the PIC24H device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Flash Programming module are:

Title	**Application Note #**
CodeGuard™ Security: Protecting Intellectual Property in Collaborative System Designs	DS70179

Note: For additional Application Notes and code examples for the PIC24H device family, visit the Microchip web site (www.microchip.com).

23.19 REVISION HISTORY

Revision A (May 2007)

This is the initial release of this document.