



*New Eagle*TM

MECHATRONIC CONTROL SOLUTIONS

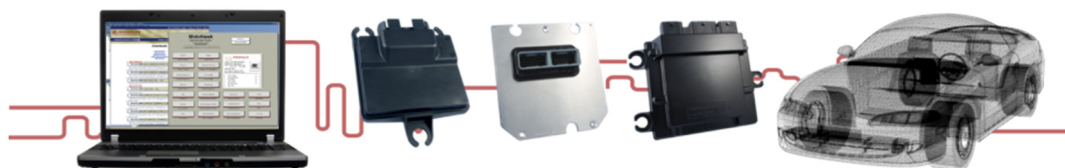


Working with S12 MotoHawk

Author(s):

New Eagle Consulting
3588 Plymouth Road, #274
Ann Arbor, MI 48105-2603
Phone: +1 (734) 929-4557

Ben Hoffman
Product Development Lead /
Senior Software Engineer
bhoffman@neweagle.net
August 11, 2011



ToolChain

The toolchain for the S12-based MotoHawk is different from all the other modules. For the S12, you will require FreeScale's CodeWarrior compiler. This for-purchase compiler can be obtained online and generally you can get started right away using the 30-day trial feature. During the 30-day trial you will have what appears to be a full-functioning license. This trial allows you time to secure a full license or merely experiment. There are different versions of this compiler, please refer to the Freescale documentation to find the most suitable and compatible version of Codewarrior for your machine. While the other modules have the free compiler option of GCC, this does not apply to the S12 MotoHawk modules.



Note: After the 30-day trial is complete, there is a 32 KB compiler limit. This is too small for most applications and will result in a link error which manifests itself in an error during build that will look similar to below:

```
>>ERROR L1102: Out of allocation space in segment RAM_BLOCK0_SEG at address 0x3BFC  
>>Error using ==> motocoder_codegen at 84
```



Memory Resources

One of the first issues you will discover upon attempting to build your MotoHawk model targeted to the S12 is that the base project, generated by the 'motohawk_project' script, does not compile! This is evidence of the memory constraints of the lower-cost module family. As a comparison, in the table below is a comparison of an S12 module to the 55xx 112-pin module.

	S12 24-pin Module	55xx 112-pin Module
		
	GCM-0S12-024-402-F	ECM-5554-112-0904-F
FLASH (Program Space)	128K	2048K (2MB)
RAM (Memory Space)	8K	64K
EEPROM (Data Storage Space)	2K	32K

Compared to the 55xx module, the S12 module has:

- 6.25% of the FLASH
- 12.25% of the RAM
- 6.25% of the EEPROM

As you can imagine this can have significant bearing on your software design. Looking at the default memory allocations in the target definition block, we see:

Foreground Stack	3K
Background Stack	2K
Idle Stack	1K
Interrupt Stack	1.5K
Application Interrupt Stack	1K
Shutdown Stack	1K
Heap Size	4K

Each of the stack sizes must be a multiple of 8 bytes.

The DLL Name and SRZ Name are optional. If unspecified, the model name is used for both fields. These fields must not contain the file extension (i.e. .dll or .srz)

The Build Directory is optional. If empty, the current directory will be used. If specified, it may be an absolute path, or be relative to the current model location by starting with ".".

Copyright 2010 Woodward. All Rights Reserved.

Parameters

Target:

Memory Layout:

Floating Point Type:

Foreground Stack Size (Time-Based):

Foreground Stack Size (Angle-Based):

Background Stack Size:

Idle Stack Size:

Interrupt Stack Size:

Application Interrupt Stack Size:

Shutdown Stack Size:

Heap Size:

Since these add up to 13.5K, which is larger than the 8K of RAM that is available on the S12 module, we know it won't compile this way. These settings will vary based on your application's profile, but recommended starting values might be:

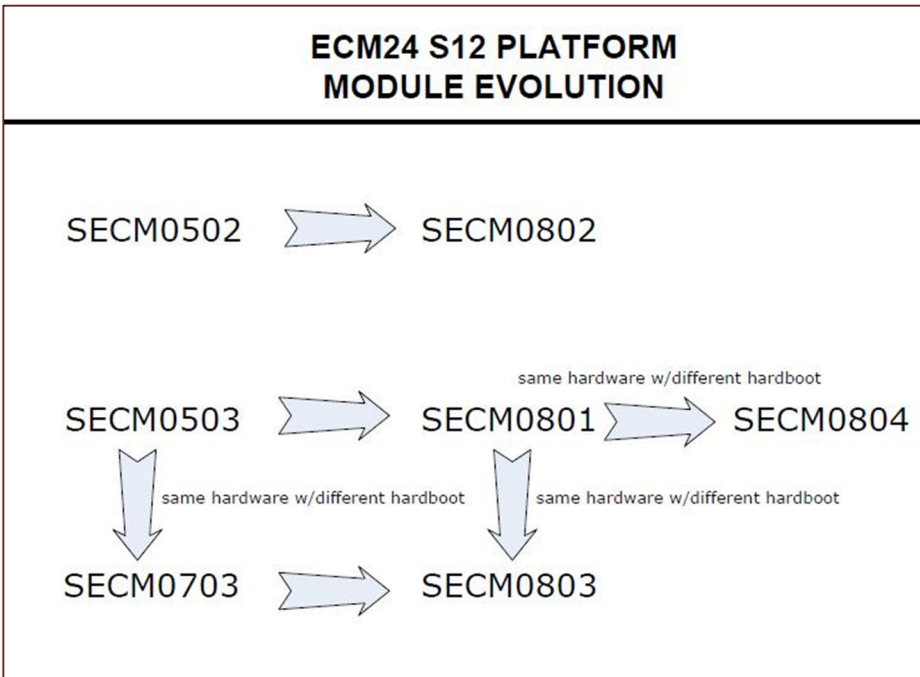
Foreground Stack	1024 bytes
Background Stack	512 bytes
Idle Stack	512 bytes
Interrupt Stack	768 bytes
Application Interrupt Stack	512 bytes
Shutdown Stack	512 bytes
Heap Size	1024 bytes

These values add up to 4.75K which is just over half the total RAM for the module. These should be adjusted as needed by your application.


Foreground Stack Size (Time-Based)	Numeric	Used for temporary variables in foreground periodic events. On 5xx targets must be a multiple of 8 bytes with a minimum of 512 bytes. On S12 or 55xx targets any positive integer will work. Run-time stack usage can be monitored in MotoTune (System Memory).
Foreground Stack Size (Angle-Based)	Numeric	Used for temporary variables in foreground angle events (Hires, _30 ...). On 5xx targets must be a multiple of 8 bytes and have a minimum of 512 bytes or be 0 bytes. On S12 or 55xx targets any positive integer will work. Note: if this is 0 all angle based variables go into the periodic foreground stack. Run-time stack usage can be monitored in MotoTune (System Memory).
Background Stack Size	Numeric	Used for temporary variables in background periodic events. On 5xx targets must be a multiple of 8 bytes with a minimum of 1024 bytes. On S12 or 55xx targets any positive integer will work. Run-time stack usage can be monitored in MotoTune (System Memory).
Idle Stack Size	Numeric	Used for temporary variables in idle events. On 5xx targets must be a multiple of 8 bytes with a minimum of 512 bytes. On S12 or 55xx targets any positive integer will work. Run-time stack usage can be monitored in MotoTune (System Memory).
Interrupt Stack Size	Numeric	Used for temporary variables during an interrupt (CAN message, crank tooth). On 5xx targets must be a multiple of 8 bytes with a minimum of 512 bytes. On S12 or 55xx targets any positive integer will work. Run-time stack usage can be monitored in MotoTune (System Memory).
Application Interrupt Stack Size	Numeric	Sets the stack size for application driven interrupts. Only available on S12 and 55xx targets. Run-time stack usage can be monitored in MotoTune (System Memory).
Shutdown Stack Size	Numeric	Sets the stack for the shutdown tasks. Only available on S12 and 55xx targets. Run-time stack usage can be monitored in MotoTune (System Memory).




Target Selection

Not all S12-based modules are listed in the MotoHawk Target Definition block. A part of the reason for that is the evolution of these modules over time which resulted in part number and name changes. The following chart provides information about the evolution of SECMs.



Further, for certain targets you will have to select a target other than that listed on your module. The following cross reference (from the MotoHawk Help documentation) is helpful in selecting the proper option in the MotoHawk Target Definition block. In addition, something to double-check would be the ‘Memory Layout’ option just below the Target Selection. That option defaults to DEV so needs to be set properly according to your target.

MotoHawk Target	Manufacturer	Aliases (x = deprecated)		
		MCSName	PartNumber	Labelled
<div style="display: flex; align-items: center;">  <div style="margin-left: 10px;"> GCM-0S12-024-0401-F </div> </div>	Continental	CTRLMSIM00100	1751-6289	MSIM0401
		CTRLSMM00100	1751-6379	SMUX0401
		GCM0S120240401F00	1751-6338	MCHI0401 CTRLCHI00202
		GCM0S120240402F00	1751-6340 x	GCM-0S12-024 MSIM0401 CTRLMSIM00100
		GCM0S120240402F30	1751-6342	MSIM0401
		GCM0S120240402F31	1751-6344	MSIM0401
		GCM0S120240403F00	1751-6346	GCM-0S12-024 CTRLSMM00100
		GCM0S120240403F10	1751-6348	SMUX CMD STATION

MotoHawk Target	Manufacturer	Aliases (x = deprecated)		
		MCSName	PartNumber	Labelled
ECM-0S12-024-0502 	Continental	ECM0S120240502F00	1751-6370	SECM0502 CTRLSECM00202
		ECM0S120240502CP0	1751-6367	SECMDEV0502
		ECM0S120240802F00	1751-6428	SECM0802
		ECM0S120240802CP0	1751-6507	SECMDEV0802
ECM-0S12-024-0503 	Continental	ECM0S120240503Fxx	1751-6375 x	CTRLSECM00601 SECM0503
		ECM0S120240503Fxx	1751-6377 x	CTRLSECM008D0 SECM0503
		ECM0S120240503CPx	1751-6373 x	CTRLSECM005AD1 SECMDEV0503
		ECM0S120240503CPx	1751-6376 x	CTRLSECM007AD0 W/RTV
		ECM0S120240503CPx	1751-6422 x	SECMDEV0503 CTRLSECM00601
		ECM0S120240703CP0	1751-6423	SECMDEV0703
		ECM0S120240703F00	1751-6424	SECM0703 W/V2.2 HB
		ECM0S120240801CP0	1751-6536	SECMDEV0801
		ECM0S120240801F00	1751-6426	SECM0801
		ECM0S120240803CP0	1751-6535	SECMDEV0803
ECM-0S12-024-0804 	Continental	ECM0S120240804CP0	1751-6534	SECMDEV0804
		ECM0S120240804F00	1751-6432	SECM0804

Programming the module

If you would program these modules under conditions like the following, you would have to be aware that the module will have non-standard City IDs by default. These conditions are:

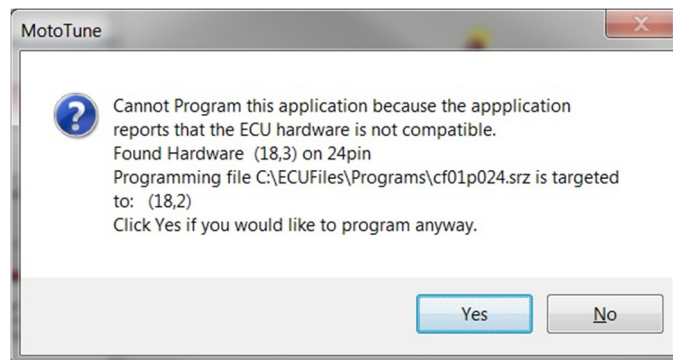
1. Programming the module for the very first time after purchasing it.
2. Using a boot harness to program the module.
3. Always until the City ID change has been made through a change in CAN settings in the model.

Module	Default City ID
GCMS12-24 (uCHI)	0x91
ECMS12-24 (0801/0802/0803)	0x81
ECMS12-24 (0804)	0x0B

All this calls for setting up another MotoTune port to reflect the right City ID. (The information on procedure for setting up the MotoTune port can be found on New Eagle Learning Center, our wiki site.)

If you are using a Boot harness, after flashing the module, you would have to switch over to a normal development or programming harness in order to access the application over MotoTune.

Another issue that will result from selecting a target which is different from the label on the module is a MotoTune error message that looks something like this:



If you have selected the proper target from the list above, it is safe to click on YES to program anyway. This will not affect the normal running of the program.

Controlled Shutdown - NonVolatile RAM

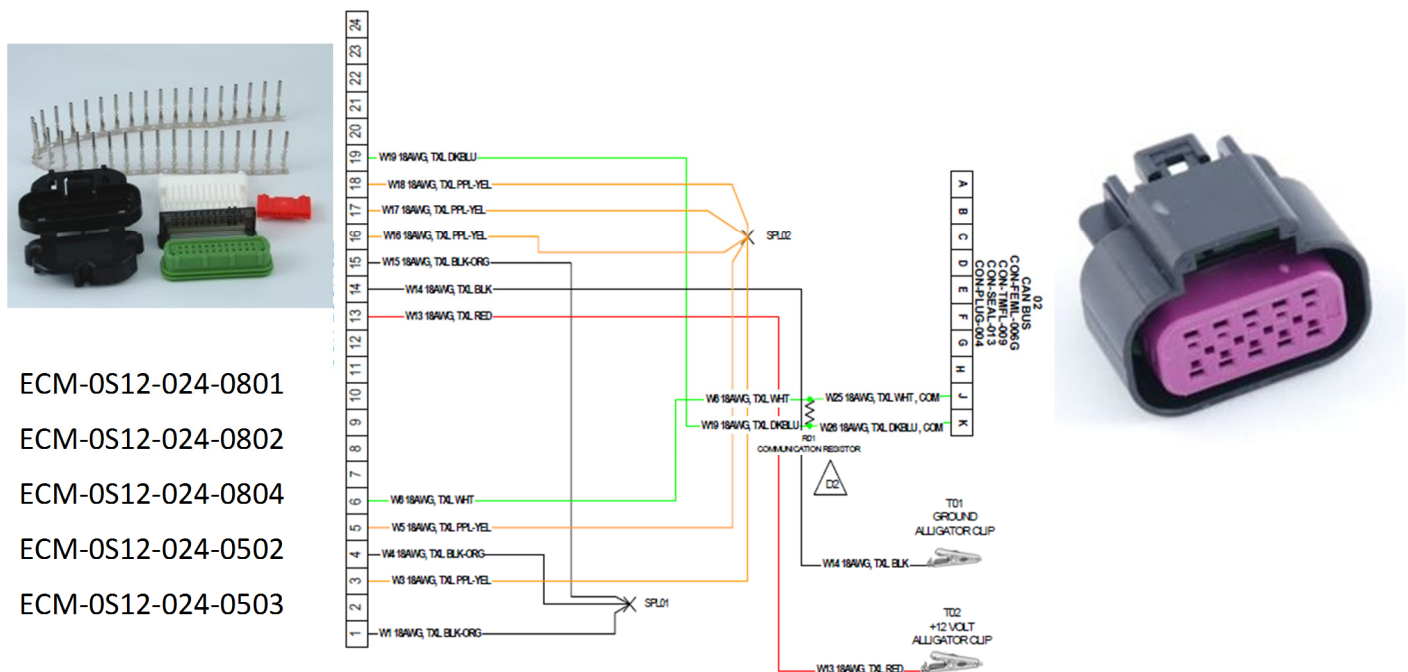
The Small Engine Control (SECM) S12 modules (ECM-0S12-024-0801, ECM-0S12-024-0802, ECM-0S12-024-0804, ECM-0S12-024-0502, ECM-0S12-024-0503) do not have a KEYSWITCH input. As a result, these modules do not operate with a 'controlled shutdown'. Normally, during a controlled shutdown software found within the 'Main Power Relay Block' will cause datastore values set as 'NonVolatile' to be written to the module's EEPROM memory. Without a controlled shutdown, the application will have to store Nonvolatile data manually using the 'MotoHawk NonVolatile Store' block. Since the EEPROM memory has a limited write-cycle lifetime (approx 100,000 times), the application must issue the store command judiciously. As an example, issuing the command once per second would burn out the EEPROM in a day or so.



Boot Procedure

If you happen to program an invalid application into a module, or are otherwise unable to communicate with the module you may need to 'boot' the module to recover. The 'boot' procedure prevents the module from beginning execution of the user application on startup for some period of time - the processor is kept in the bootloader awaiting reprogramming commands. The S12 family of MotoHawk modules don't all use the standard boot key that other modules do. Certain modules require a special 'boot harness' which provides a combination of I/O states that the module recognizes as a signal to stay in 'boot' mode.

SECM Boot Harness



ECM-0S12-024-0801

ECM-0S12-024-0802

ECM-0S12-024-0804

ECM-0S12-024-0502

ECM-0S12-024-0503

GCM 24 Boot Harness



GCM-0S12-024-0401

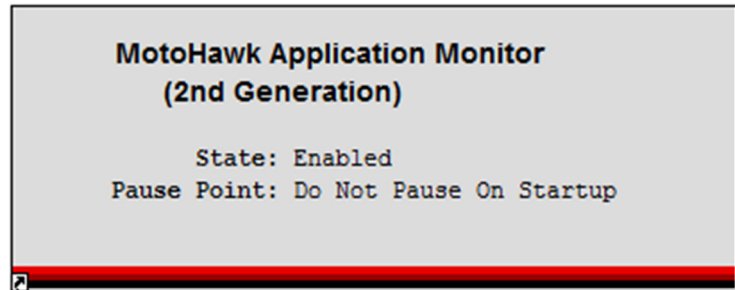
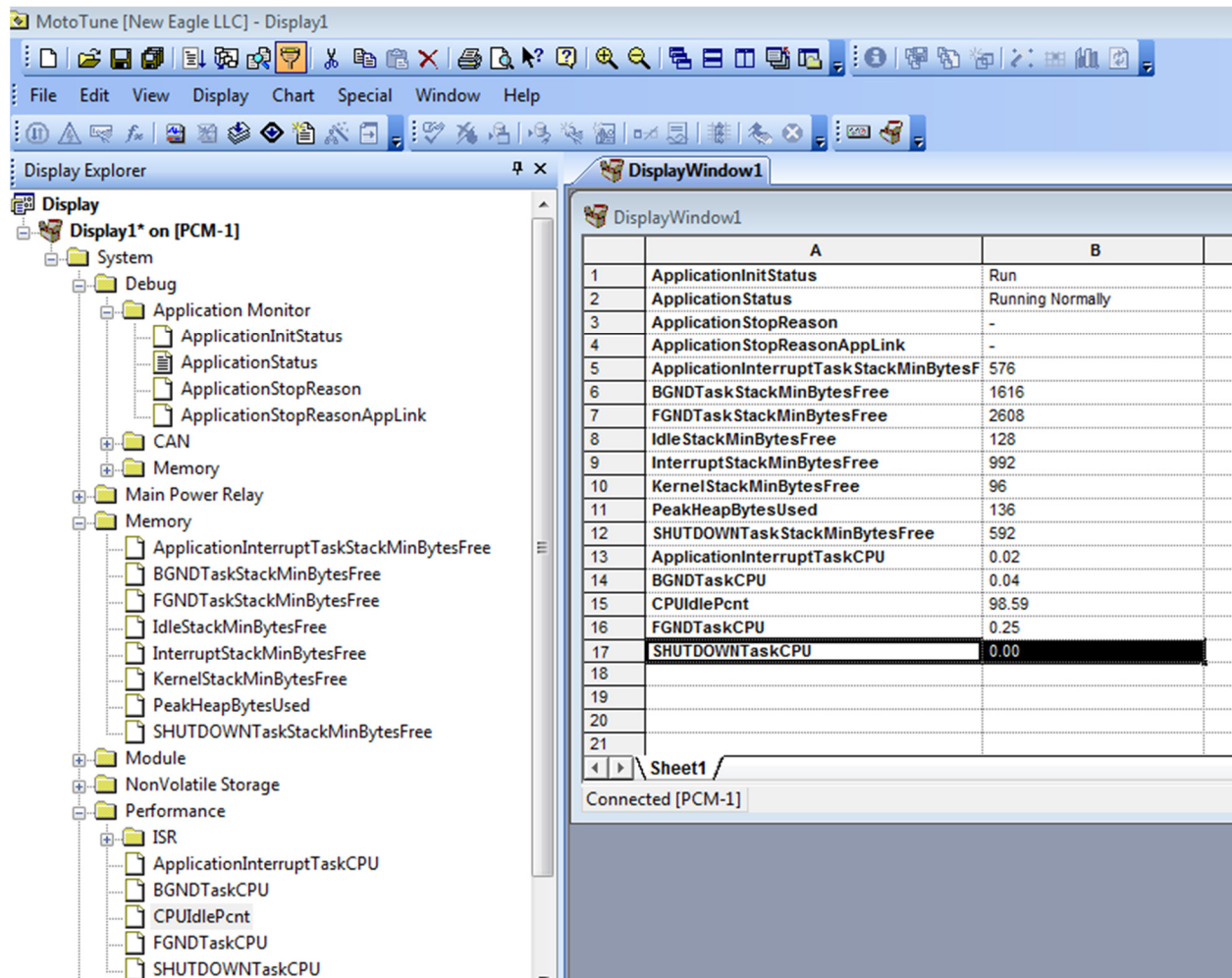
GCM-0S12-024-0402

2	F
3	A
4	B
5	A
13	A
14	B
15	B
16	A
17	A



Application Monitor

The 'MotoHawk Application Monitor' sets up a monitor within your application that checks critical system parameters as your application executes. While monitoring these parameters it will halt the application if usage exceeds certain thresholds you configure within the mask for this block. Although the application is halted when a threshold is exceeded, the MotoTune protocol handler should continue to execute meaning that you can inspect the state of the controller using MotoTune to determine what has occurred. The following MotoTune Display variables are useful in working with the App Monitor:

Display Explorer

- Display
 - Display1* on [PCM-1]
 - System
 - Debug
 - Application Monitor
 - ApplicationInitStatus
 - ApplicationStatus
 - ApplicationStopReason
 - ApplicationStopReasonAppLink
 - CAN
 - Memory
 - Main Power Relay
 - Memory
 - ApplicationInterruptTaskStackMinBytesFree
 - BGNDTaskStackMinBytesFree
 - FGNDTaskStackMinBytesFree
 - IdleStackMinBytesFree
 - InterruptStackMinBytesFree
 - KernelStackMinBytesFree
 - PeakHeapBytesUsed
 - SHUTDOWNTaskStackMinBytesFree
 - Module
 - NonVolatile Storage
 - Performance
 - ISR
 - ApplicationInterruptTaskCPU
 - BGNDTaskCPU
 - CPUIdlePcnt
 - FGNDTaskCPU
 - SHUTDOWNTaskCPU

DisplayWindow1

	A	B
1	ApplicationInitStatus	Run
2	ApplicationStatus	Running Normally
3	ApplicationStopReason	-
4	ApplicationStopReasonAppLink	-
5	ApplicationInterruptTaskStackMinBytesFree	576
6	BGNDTaskStackMinBytesFree	1616
7	FGNDTaskStackMinBytesFree	2608
8	IdleStackMinBytesFree	128
9	InterruptStackMinBytesFree	992
10	KernelStackMinBytesFree	96
11	PeakHeapBytesUsed	136
12	SHUTDOWNTaskStackMinBytesFree	592
13	ApplicationInterruptTaskCPU	0.02
14	BGNDTaskCPU	0.04
15	CPUIdlePcnt	98.59
16	FGNDTaskCPU	0.25
17	SHUTDOWNTaskCPU	0.00
18		
19		
20		
21		

Sheet1 /
Connected [PCM-1]

Configuring the Application Monitor for the S12 is done via parameters in the target definition block for most parameters. There is a separate block for the starvation timer. The parameters available are shown in the table below:

S12 and 55xx Target Specific (when Target selection has a S12 or 55xx processor)		
Foreground Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the periodic foreground stack.
Foreground Angle Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the angle based foreground stack.
Background Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the background stack.
Idle Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the idle stack.
Interrupt Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the interrupt stack.
Application Interrupt Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the application interrupt stack.
Shutdown Stack Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the shutdown stack.
Heap Margin [bytes]	Numeric	This triggers the Application Monitor Definition 2nd Generation when less than this number of bytes are free in the heap.

The App Monitor can stop your application for the following reasons:

- App Monitor Notification Stop
- Application Stopped (User Command)
- Starvation timer margin violation
- Heap margin violation
- Idle stack margin violation
- Interrupt stack margin violation
- ApplicationInterruptTask stack margin violation
- BGNDDTask stack margin violation
- SHUTDOWNTask stack margin violation
- FGNDTask stack margin violation

When the application monitor stops your application for one of these reasons, it is because a condition has been detected that needs to be resolved in your software. Often times, all that is required is adjusting the stack sizes to better match your application, but for applications that are approaching the resource constraints of the module, this can involve tradeoffs and optimizations.