SCIOPTA Kernel Configuration SCONF For Arm

v1.1

Abstract

This document describes the **SCIOPTA Kernel Configuration** for the SCIOPTA Kernel for Arm.

Copyright

Disclaimer

Trademark

Contact

Corporate Headquarters
SCIOPTA Systems GmbH
Hauptstrasse 293
79576 Weil am Rhein
Germany
Tel. +49 7621 940 919 0
Fax +49 7621 940 919 19
email: sales@sciopta.com
www.sciopta.com

Contents

1. Introduction

1.1. SCIOPTA Real-Time Operating System

SCIOPTA is a high performance fully pre-emptive real-time operating system for hard real-time application available for many target platforms.

Available modules:
• Pre-emptive Multitasking Real-Time Kernel
• Board Support Packages
• IPS - Internet Protocols v4/v6 (TCP/IP)
• IPS Applications - Internet Protocols Applications (Web Server, TFTP, FTP, DNS, DHCP, Telnet, SMTP etc.)
• FAT File System
• (fail) SAFE FAT File System
• Flash File System, NOR and NAND
• Universal Serial Bus, USB Device
• Universal Serial Bus, USB Host
• DRUID - System Level Debugger
• SCIOPTA PEG - Embedded GUI
• CONNECTOR - support for distributed multi-CPU systems
• SCIOPTA Memory Management System - Support for MMU
• SCAPI - SCIOPTA API for Windows or LINUX host
• SCSIM - SCIOPTA Simulator

SCIOPTA Real-Time Operating System contains design objects such as SCIOPTA modules, processes, messages and message pools. SCIOPTA is designed on a message based architecture allowing direct message passing between processes. Messages are mainly used for interprocess communication and synchronization.
SCIOPTA messages are stored and maintained in memory pools. The memory pool manager is designed for high performance and memory fragmentation is avoided. Processes can be grouped in SCIOPTA modules, which allows you to design a very modular system. Modules can be static or created and killed during runtime as a whole. SCIOPTA modules can be used to encapsulate whole system blocks (such as a communication stack) and protect them from other modules in the system.

The SCIOPTA Real-Time Kernel has a very high performance. The SCIOPTA architecture is specifically designed to provide excellent real-time performance and small size. Internal data structures, memory management, interprocess communication and time management are highly optimized. SCIOPTA Real-Time kernels will also run on small single-chip devices without MMU.

1.2. SCIOPTA Safety Kernels

SCIOPTA provides a Real-Time Operating System for some CPU families which is certified according to IEC 61508 up to SIL3, CENELEC EN 50128 up to SIL3/4 and ISO 26262 up to ASIL D:

`IEC INTERNATIONAL STANDARD 61508`
Edition 2.0 2010-04
Functional safety of electrical/electronic/programmable electronic safety-related systems
Part 1: General requirements
Part 2: Requirements for electrical/electronic/programmable electronic safetyrelated systems (in addition for INT Kernel)
Part 3: Software requirements
Part 4: Definitions and abbreviations

`CENELEC European Committee for Electrotechnical Standardization FprEN 50128:2011`
Railway applications
Communication, signalling and processing systems

Software for railway control and protection systems

ISO International Organization for Standardization 26262
First Edition 2018-12
Road vehicles – Functional safety
Part 2: Management of functional safety
Part 6: Product development at the software level
Part 8: Supporting processes

Please consult the SCIOPTA Kernel Manuals for more information (Ref. SCIOPTA Kernel Reference Manual and SCIOPTA Architecture Manual).

1.3. CPU Family

SCIOPTA is delivered for a specific CPU Family such as: ARM®7/9, ARM®11, ARM® Cortex-M™, ARM® Cortex™-R, ARM® Cortex™-A, Renesas RX, Freescale™ PowerPC, apm PowerPC, Freescale™ ColdFire and Marvell Xscale.
Please consult the latest version of the SCIOPTA Price List for the complete list.

1.4. About This Manual

The purpose of this SCIOPTA Kernel Configuration SCONF Manual is to give all needed information how to use and configure SCONF in an embedded project before you can generate the whole system.

2. SCONF Kernel Configuration

The kernel of a SCIOPTA system needs to be configured before you can generate the whole system. In the SCIOPTA configuration utility **SCONF (sconf.exe)** you will define the parameters for SCIOPTA systems such as name of systems, static modules, processes and pools etc.

The **SCONF** program is a graphical tool which will save all settings in an external XML file. If the setting are satisfactory for your system **SCONF** will generate three source files containing the configured part of the kernel. These files must be included when the SCIOPTA system is generated.

A SCIOPTA project can contain different SCIOPTA Systems which can also be in different CPUs. For each SCIOPTA System defined in **SCONF** a set of source files will be generated.

2.1. Starting SCONF

The SCIOPTA configuration utility **SCONF (sconf.exe)** can be launched from the SCIOPTA delivery (located at: <install_folder>\sciopta\<version>\bin\win32\). You can create short cuts to sconf.exe as described in the Windows documentation.

After starting the welcome screen will appear. The latest saved project will be loaded or an empty screen if the tool was launched for the first time.

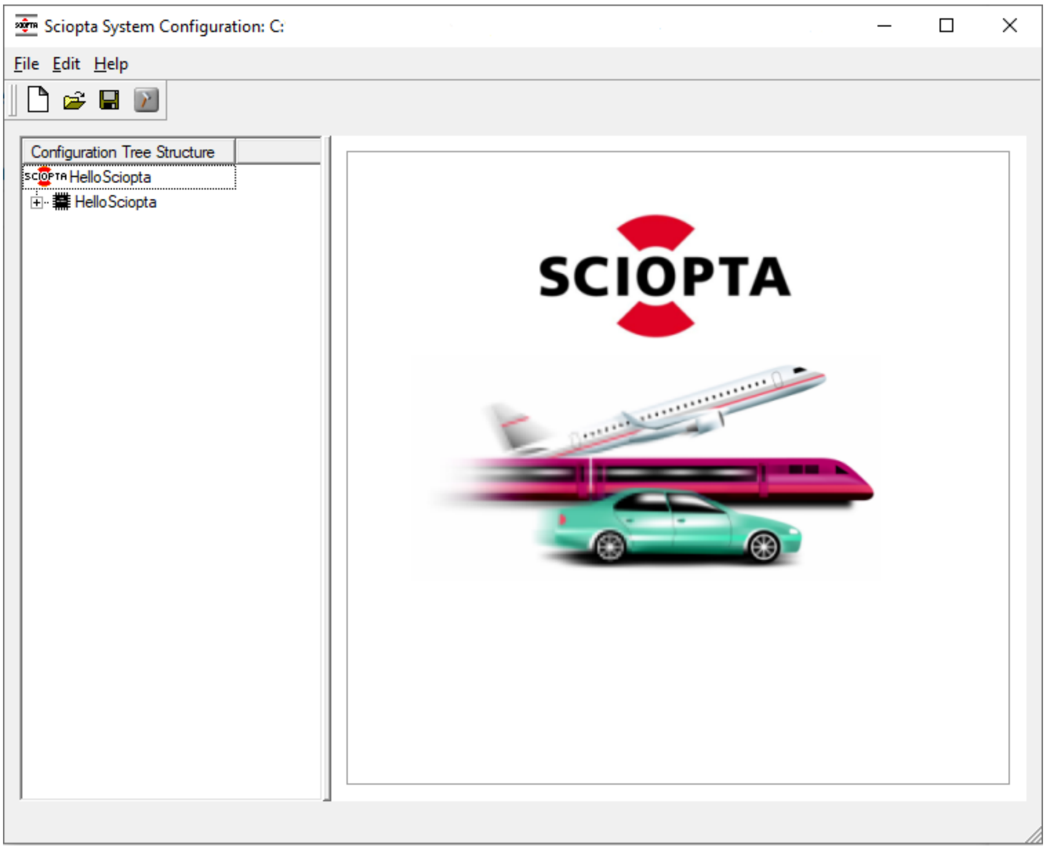

Figure 1. SCIOPTA Configuration Utility Start Screen

2.2. Preference File sc_config.cfg

The SCIOPTA Configuration Utility **SCONF** is storing some preference setting in the file **sc_config.cfg**.
Actually there are only three settings which are stored and maintained in this file:

1. Project name of the last saved project.
2. Location of the last saved project file.
3. Warning state (enabled/disabled).

The **sc_config.cfg** file is located in the home directory of the user. The location cannot be modified.
Every time **SCONF** is started the file **sc_config.cfg** is scanned and the latest saved project is entered.

At every project save the file **sc_config.cfg** is updated.

2.3. Project File

The project can be saved in an external XML file **<project_name>.xml**. All configuration settings of the project are stored in this file.

2.4. SCONF Windows

To configure a SCIOPTA system with **SCONF** you will work mainly in two windows.

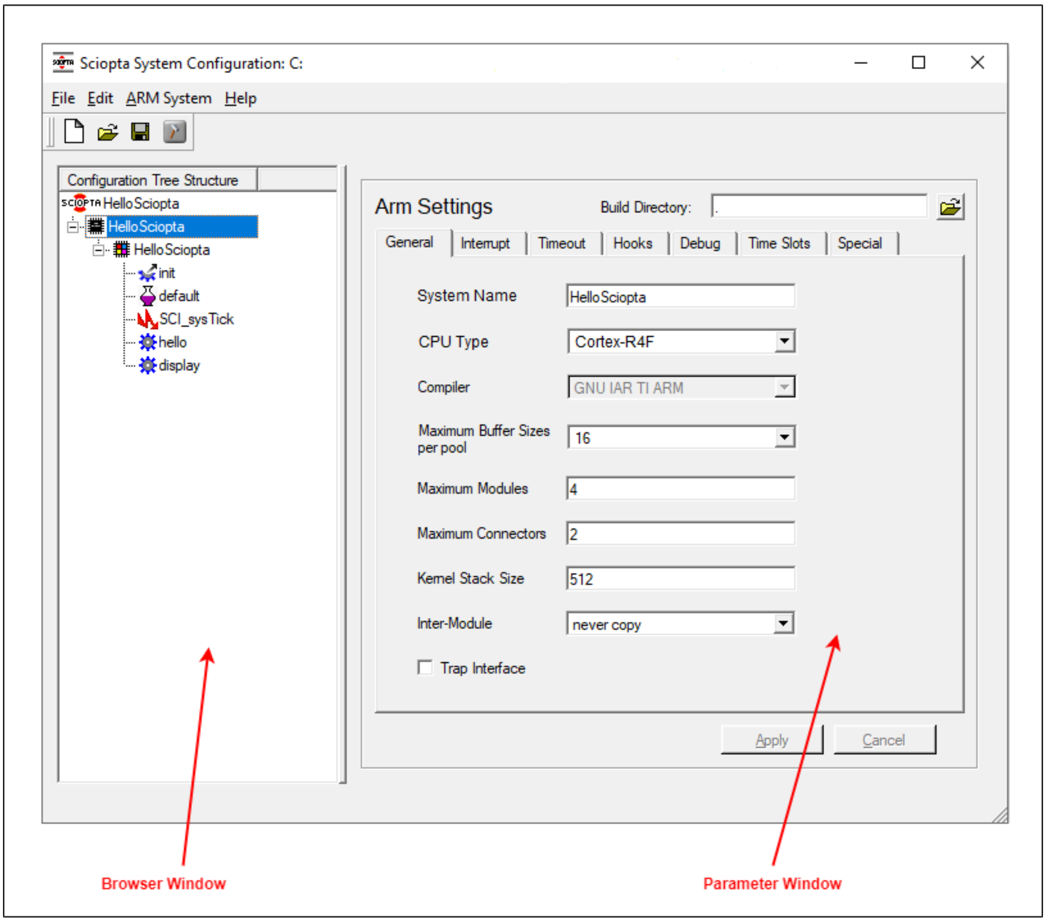

Figure 2. SCONF Windows

2.4.1. Parameter Window

For every level in the browser window (process level, module level, system level and project level) the layout of the parameter window change and you can enter the configuration parameter for the specific item of that level (e.g. parameters for a specific process). To open a specific parameter window just click on the item in the browser window.

2.4.2. Browser Window

The browser window allows you to browse through a whole SCIOPTA project and select specific items to configure.

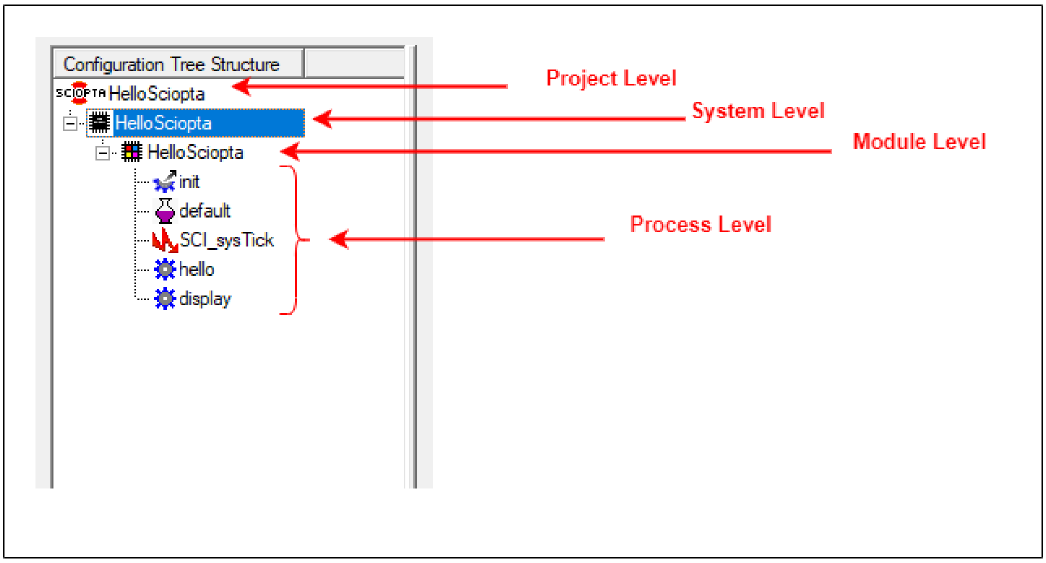

Figure 3. Browser Window

The browser shows four configuration levels and every level can expand into a next lower level. To activate a level you just need to point and click on it. On the right parameter window the configuration settings for this level can be viewed and modified.

1. The uppermost level is the Project Level where all project configurations will be done. The project name can be defined and you can create new systems for the project.
2. In the System Level you are configuring the system for one CPU. You can create the static modules for the system and configure system specific settings.
3. In SCIOPTA you can group processes into modules. On the Module Level you can configure the module parameters and create static processes and message pools.
4. The parameters of processes and message pools can be configured in the Process Level.

2.5. Creating a New Project

To create a new project select the New button in the tool bar:

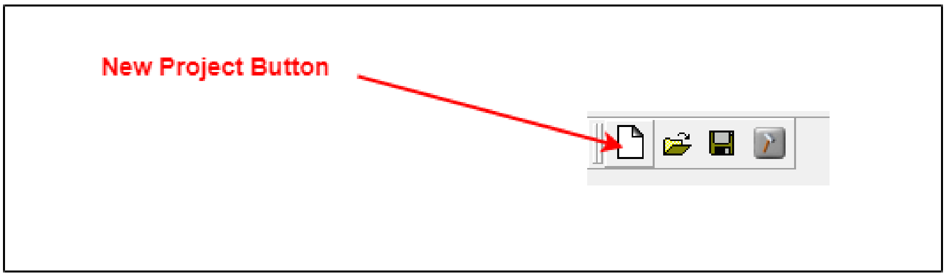

Figure 4. New Project Button

2.6. Configure the Project

You can define and modify the project name. Click on the project name on the right side of the SCIOPTA logo and enter the project name in the parameter window.

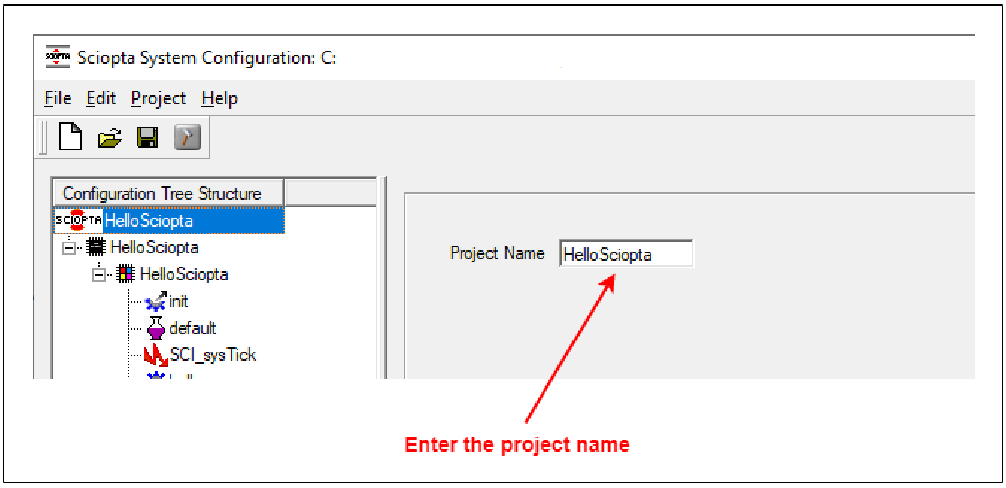

Figure 5. Defining the Project Name

Click on the **Apply** button to accept the name of the project.

2.7. Creating ARM Systems

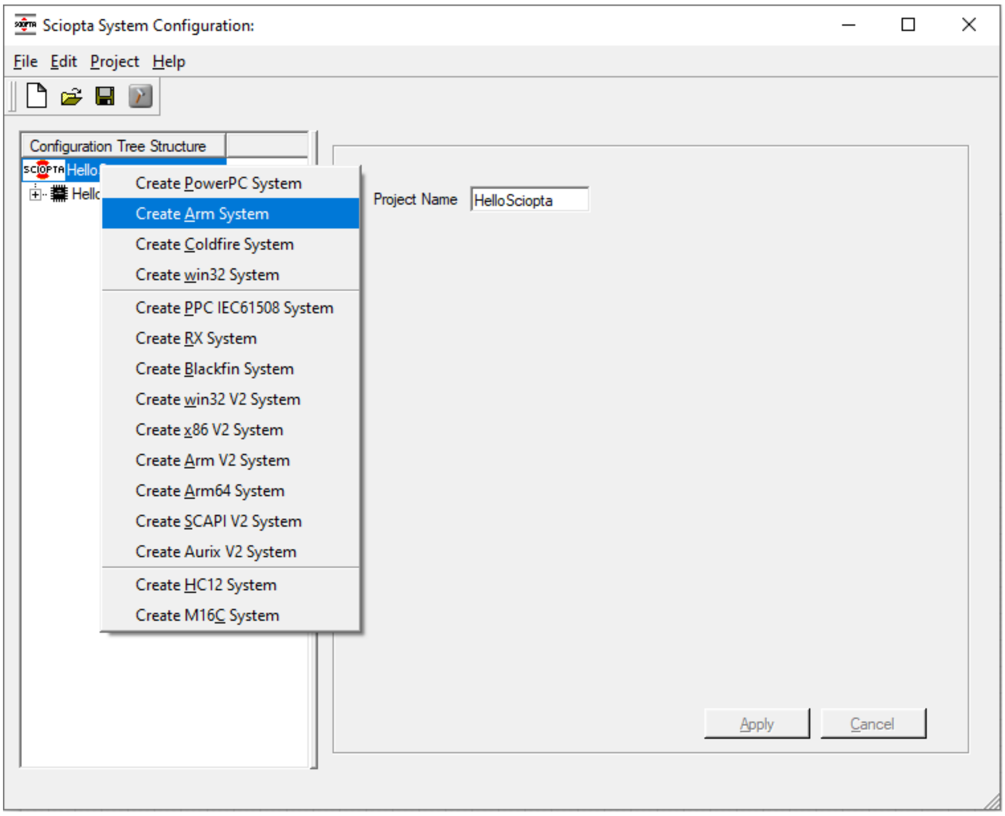

Figure 6. Create System Menu

A pop-up menu appears and allows you to select a system out of all SCIOPTA supported target CPUs. For Cortex-M and Cortex-R MCUs you need to select “Create Arm System”.

The same selection can be made by selecting the Project menu from the menu bar.
SCONF asks you to enter a directory where the generated files will be stored:

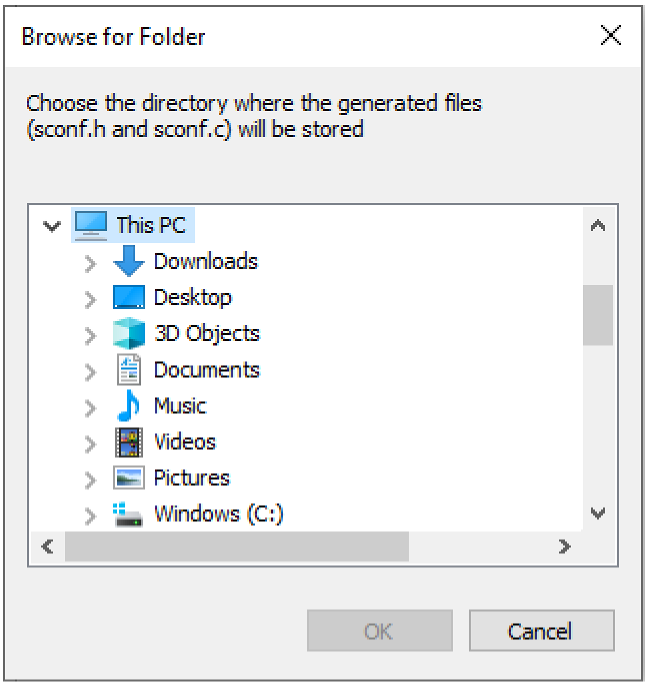

Figure 7. Select Destination Folder

A new SCIOPTA system for an ARM architecture with the default name ARM_system, the system module (module id 0) with the same name as the new target and a init process will be created.

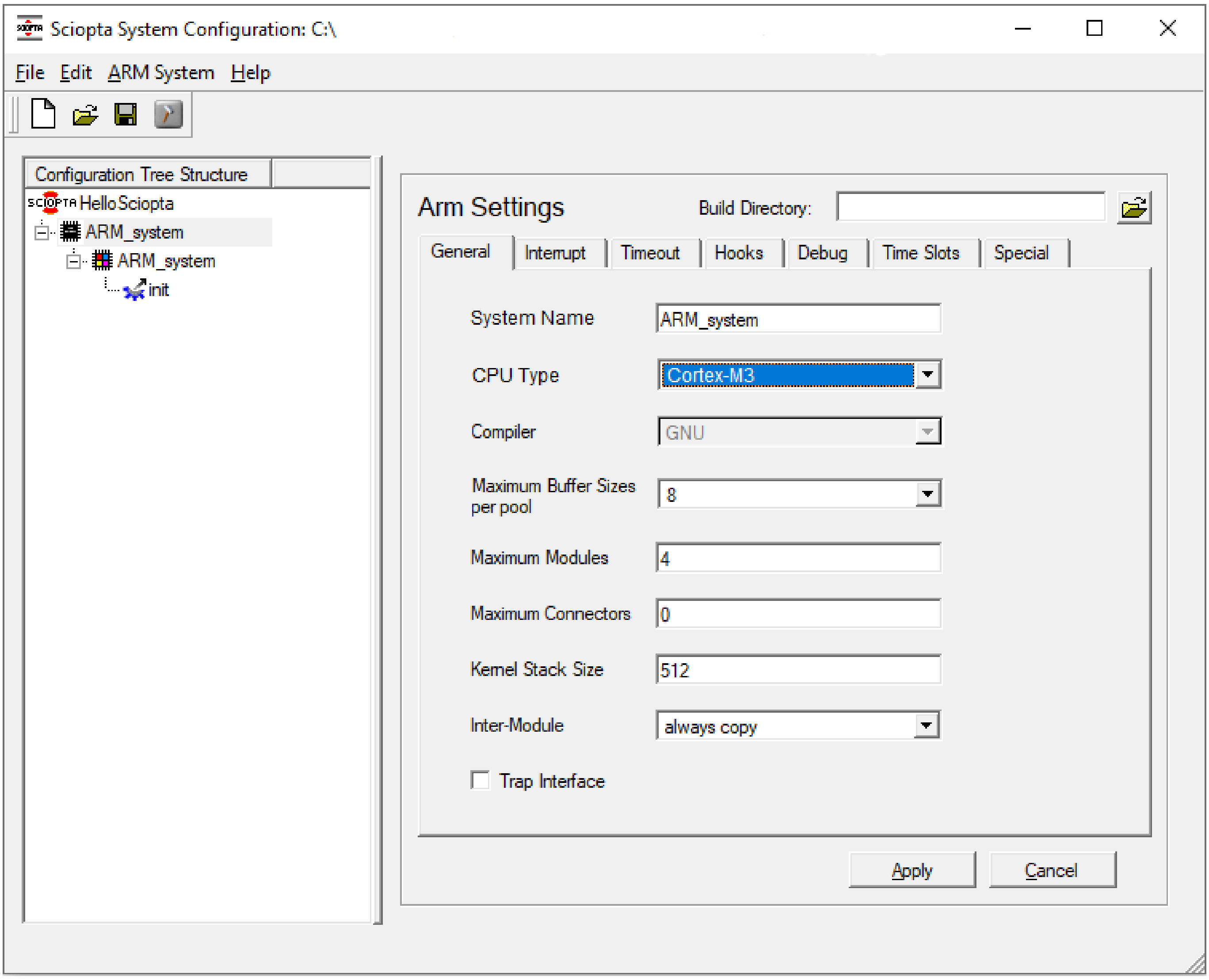

Figure 8. New System Window

You can create up to 128 systems inside a SCIOPTA project. The targets do not need to be of the same processor (CPU) type. You can mix any types or use the same types to configure a distributed system within the same SCIOPTA project.

You are now ready to configure the individual targets.

2.8. Configuring ARM Systems

After selecting a system with your mouse, the corresponding parameter window on the right side will show the parameters for an ARM architecture.

The system configuration for ARM architecture is divided into 7 tabs: General, Interrupt, Timeout, Hooks, Debug, Time Slots, and Special.

2.8.1. General System Configuration tab

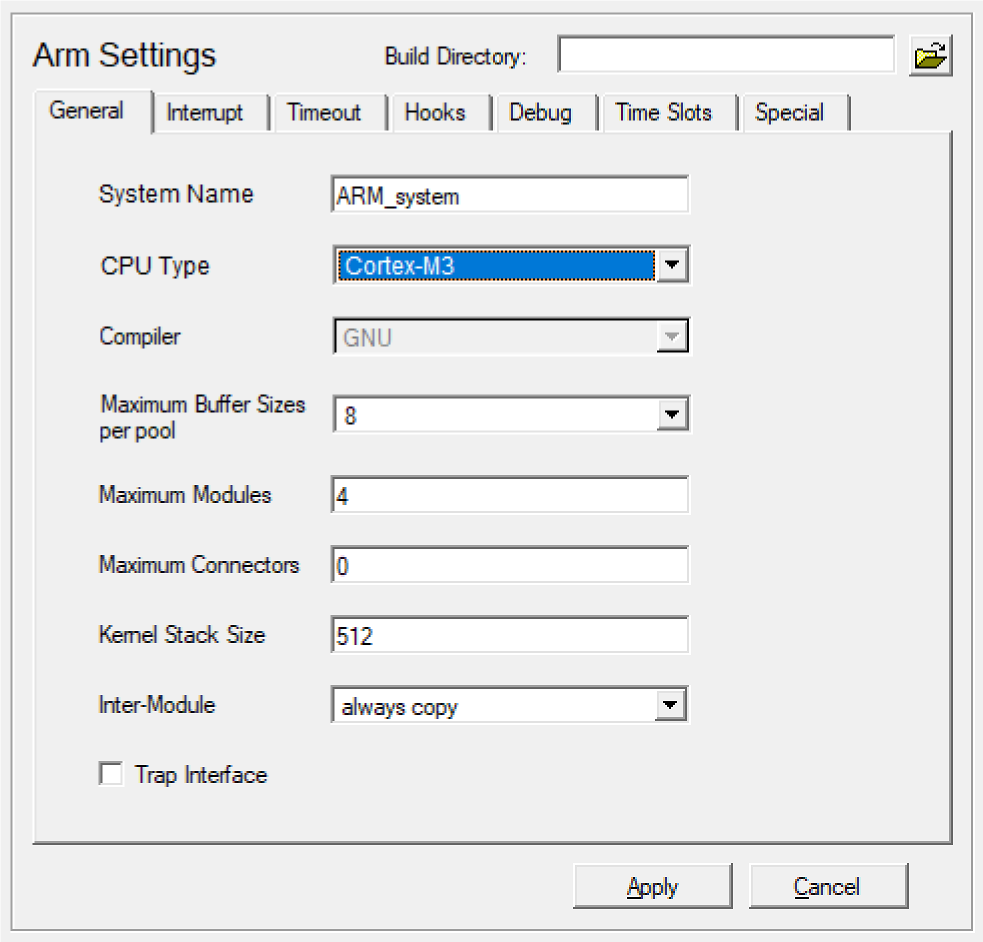

Figure 9. General Configuration Window

2.8.1.1. General tab Parameters

System Name	Name of the target system
	Enter the name of your system. Please note that the system module (module 0) in this system will get the same name.

CPU Type	CPU family for the selected architecture	
	ARMv4T+MMU	For ARM7TDMI and ARM9TDMI CPU-Family including MMU support.
	ARMv4T	For ARM7TDMI and ARM9TDMI CPU-Family.
	ARMv5T	For ARM946E-S CPU-Family.
	ARMv5TE	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family.
	ARMv5TE+FPU	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family including FPU support.
	ARMv5TE+MMU	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family including MMU support.
	ARMv6	For ARM1136J(F)-S CPU-Family.
	ARMv6+FPU	For ARM1136J(F)-S CPU-Family including FPU support.
	ARMv6M	For Cortex-M0 and Cortex-M1 CPU-Family.
	Cortex-M0	For Cortex-M0 CPU-Family.
	Cortex-M3	For Cortex-M3 CPU-Family.
	Cortex-M4	For Cortex-M4 CPU-Family.
	Cortex-M4F	For Cortex-M4F CPU-Family including FPU support.
	Cortex-M7	For Cortex-M7 CPU-Family.
	Cortex-M7F	For Cortex-M7F CPU-Family including FPU support.
	Cortex-R4	For Cortex-R4 CPU-Family.
	Cortex-R4F	For Cortex-R4F CPU-Family including FPU support.
	Cortex-R5	For Cortex-R5 CPU-Family.
	Cortex-R5F	For Cortex-R5F including FPU support.
	ARMv7R	For Cortex-R4, Cortex-R5, Cortex-R7 and Cortex-R8 CPU-Family.
	ARMv7R+FPU	For Cortex-R4, Cortex-R5, Cortex-R7 and Cortex-R8 CPU-Family including FPU support.
	ARMv7A	For Cortex-A5, Cortex-A7, Cortex- A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family.
	ARMv7A+FPU	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU support.
	ARMv7A+FPU+D32	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU and VFPv4-D32 support.
	ARMv7A+FPU+NEON	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU with NEON support.
	ARMv7A+FPU+NEON+D32	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU and VFPv4-D32 with NEON support.
	Cortex-A7	For Cortex-A7 CPU-Family.
	Cortex-A15	For Cortex-A15 CPU-Family.

	ZYNQ_MC	For ARMv7-A CPU-Family.
	Xscale	For ARMv5TE CPU-Family.
Compiler	C/C++ Compiler	
	GNU	GCC `Valid for all compilers`
Maximum Buffer Sizes	Maximum number of message buffer sizes	
	4, 8 or 16	If a process allocates a message there is also the size to be given. The user just gives the number of bytes needed. SCIOPTA is not returning the exact amount of bytes requested but will select one of a list of buffer sizes which is large enough to contain the requested number. This list can contain 4, 8 or 16 sizes which is configured here in the maximum buffer sizes entry. See also Sciopta Architecture Manual chapter Message Sizes.
Maximum Modules	Maximum number of SCIOPTA modules in the system	
	1 … 127	Here you can define a maximum number of modules which can be created in this system. The maximum value is 127 modules. It is important that you give here a realistic value of maximum number of modules for your system as SCIOPTA is initializing some memory statically at system start for the number of modules given here. See also Sciopta Architecture Manual chapter Modules.
Maximum Connectors	Maximum number of CONNECTOR processes in the system	
	0 … 127	CONNECTORS are specific SCIOPTA processes and responsible for linking a number of SCIOPTA systems. The maximum number of connectors in a system may not exceed 127 which correspond to the maximum number of systems. See also Sciopta Architecture Manual chapter Distributed Systems.
Kernel Stack Size	Size of the global kernel stack	
	512	Currently not used. Entered values are not considered.
Inter-Module	Defines if messages between modules are copied or not	
	never copy	Messages between modules are never copied.
	always copy	Messages between modules are always copied.
	friends	The message copy behaviour is defined by the friendship setting between the modules.
	See also Sciopta Architecture Manual chapter Messages Modules and Module Friend Concept	
Trap Interface	Enables the trap interface	

In a typical monolithic SCIOPTA systems the kernel functions are directly called. In more complex dynamic systems using load modules or MMU/MPU protected modular systems the kernel functions cannot be accessed any more by direct calls. SCIOPTA offers a trap interface. In such systems you need to enable the Trap Interface and assemble the file syscall.S.
See also SCIOPTA Getting Start Manual chapter Trap Interface

2.8.2. Interrupt Configuration tab

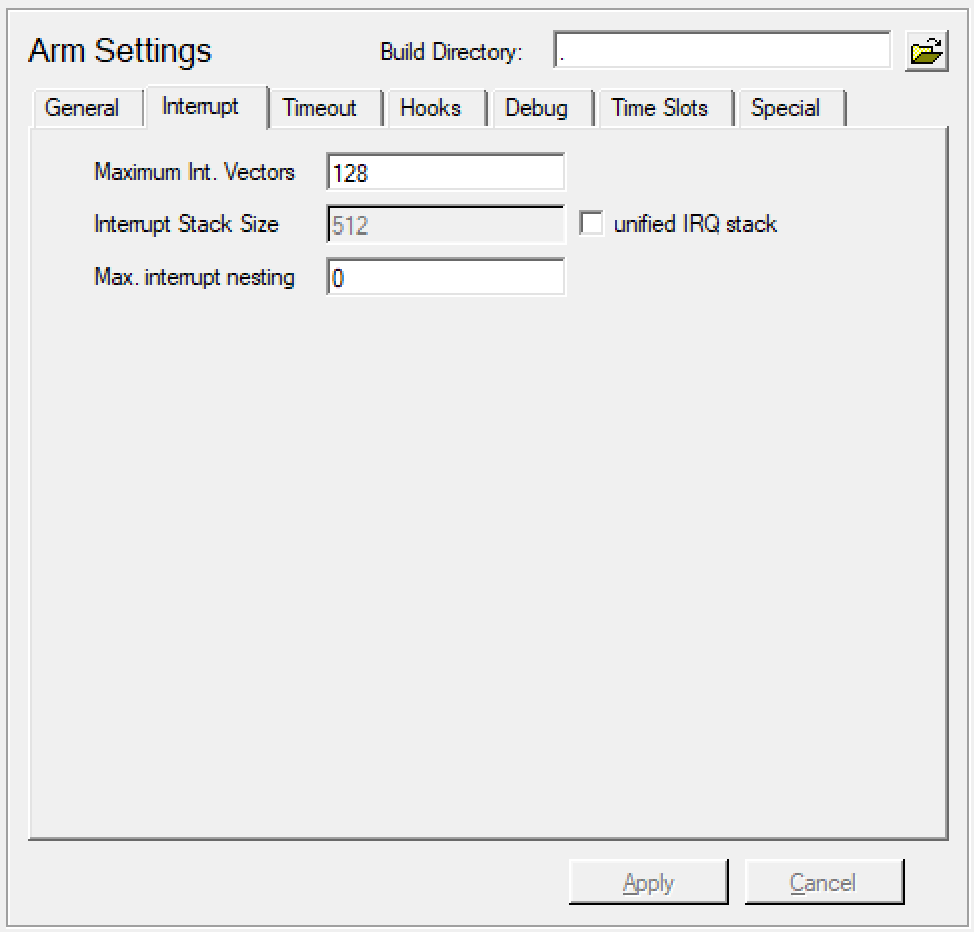

Figure 10. Interrupt Configuration Window

2.8.2.1. Interrupt tab Parameters

Maximum Int. Vectors	Maximum number of interrupt vectors	
	0 … 255	Select maximum interrupt vectors used in your Cortex-M and Cortex-R application system. See also SCIOPTA Getting Start Manual chapter Interrupt Vector Table.

Interrupt Stack Size	Size of the global interrupt stack	
	<size>	Only used if **“unified IRQ stack”** checkbox is selected. The stack size given must be big enough to hold the stacks of the interrupt processes with the biggest stack needs taken in account the interrupt nesting.

Max interrupt nesting	Maximum interrupt nesting level	
	0	No nesting
	<level>	Maximum nesting level of interrupt processes in the system.

2.8.3. Timeout Configuration tab

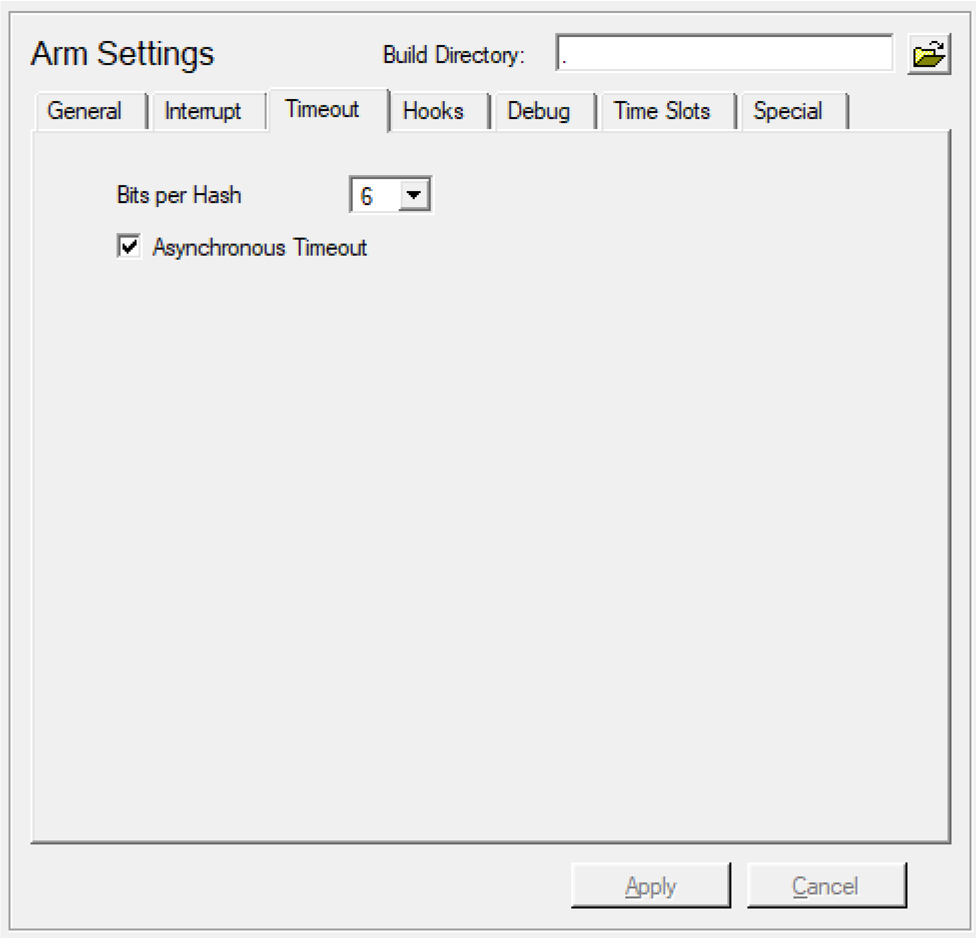

Figure 11. Timeout Configuration Window

2.8.3.1. Timeout tab Parameters

Bits per Hash	Some targets allow setting of hash size	
	2..6	The smaller the hash tables the more often a process has to be re-added into another table until the timeout has expired. The total size of a hash table is 4*(1<<SC_BITS_PER_HASH)*sizeof(__ptrsize_t)*2. Thus for a 32Bit CPU ranges from 128 bytes to 2048 bytes.

Asynchronous Timeout	Enables the time-out server
	The SCIOPTA time-out server can be enabled and disabled. Disabling the timeout server will reduce the kernel size. See also Sciopta Architecture Manual chapter Time-Out Server.

2.8.4. Hooks Configuration tab

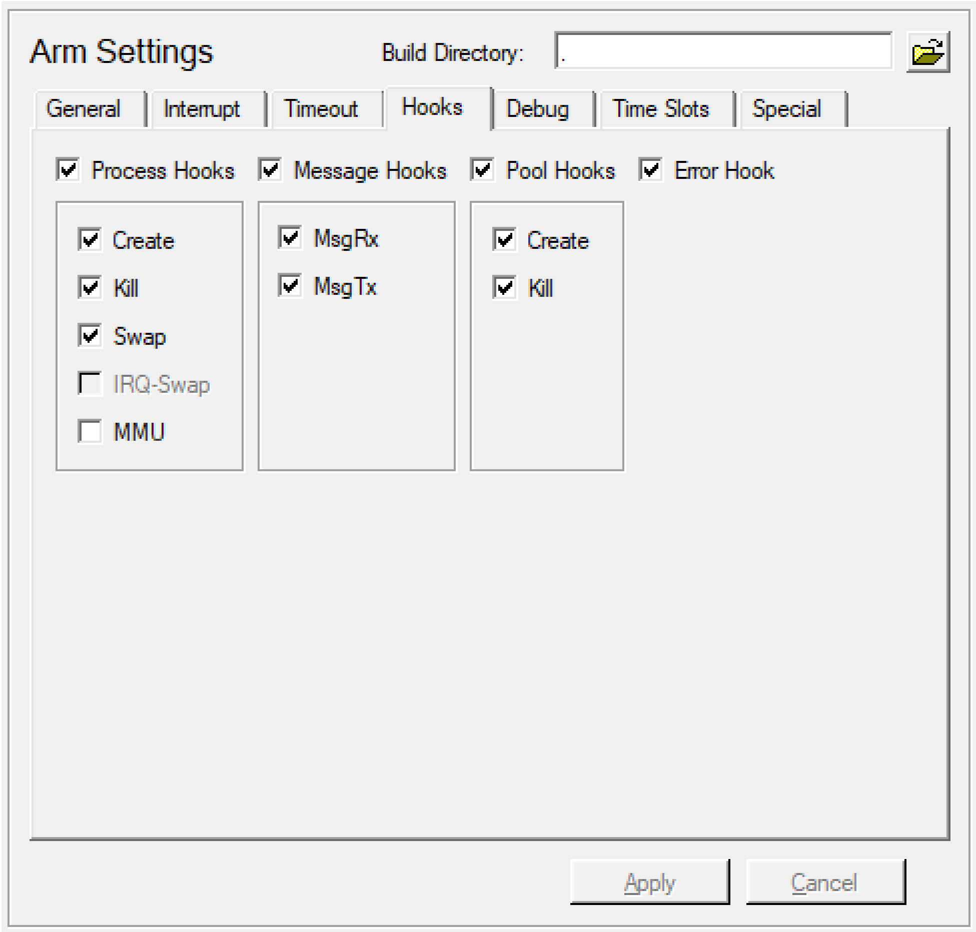

Figure 12. Hooks Configuration Window

Hooks are user written functions which are called by the kernel at different locations. They are only called if the user defined them at configuration. User hooks are used for a number of different purposes and are system dependent.

You can enable the hooks separately by selecting the corresponding check box or all hooks belonging to a group all together.

Please consult Sciopta Architecture Manual chapter Hooks for more information about SCIOPTA hooks.

You must select the MMU checkbox if you want to use the Cortex-M and Cortex-R MPU. This will enable the MPU functions in the kernel.

2.8.5. Debug Configuration tab

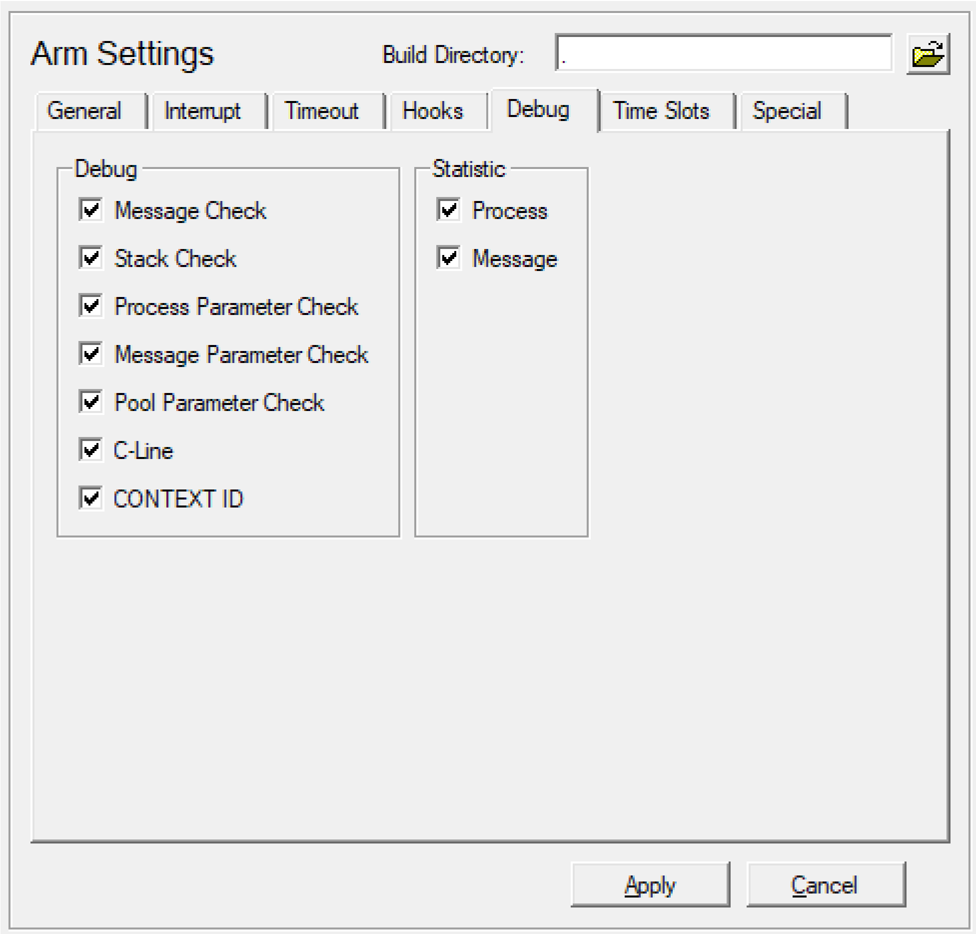

Figure 13. Debug Configuration Window

2.8.5.1. Debug tab Parameters

Message Check	Enables the message check functions in the kernel All internal message test functions will be included in the kernel.
Stack Check	Enables the stack check functions.
Process Parameter Check	Enables process parameter checks Parameter check of the process system calls will be included in the kernel.
Message Parameter Check	Enables message parameter checks. Parameter check of the message system calls will be included in the kernel.
Pool Parameter Check	Enables pool parameter checks. Parameter check of the pool system calls will be included in the kernel.

C-Line	Enables C-line informations. If you are selecting this check box the kernel will include line number information which can be used by the SCIOPTA DRUID Debug System or an error hook. Line number and file of the last system call is recorded in the per process data. Note: If configured, you may add a compiler command line macro SC_CDEBUG=1.
CONTEXT ID	Enables CONTEXT ID informations.
Process Statistics	Includes process statistics. The kernel will maintain a process statistics data field where information such as number of process swaps can be read.
Message Statistics	Includes message statistics. The kernel will maintain a message statistics data field in the pool control block where information such as number of message allocation can be read.

2.8.6. Time Slots Configuration tab

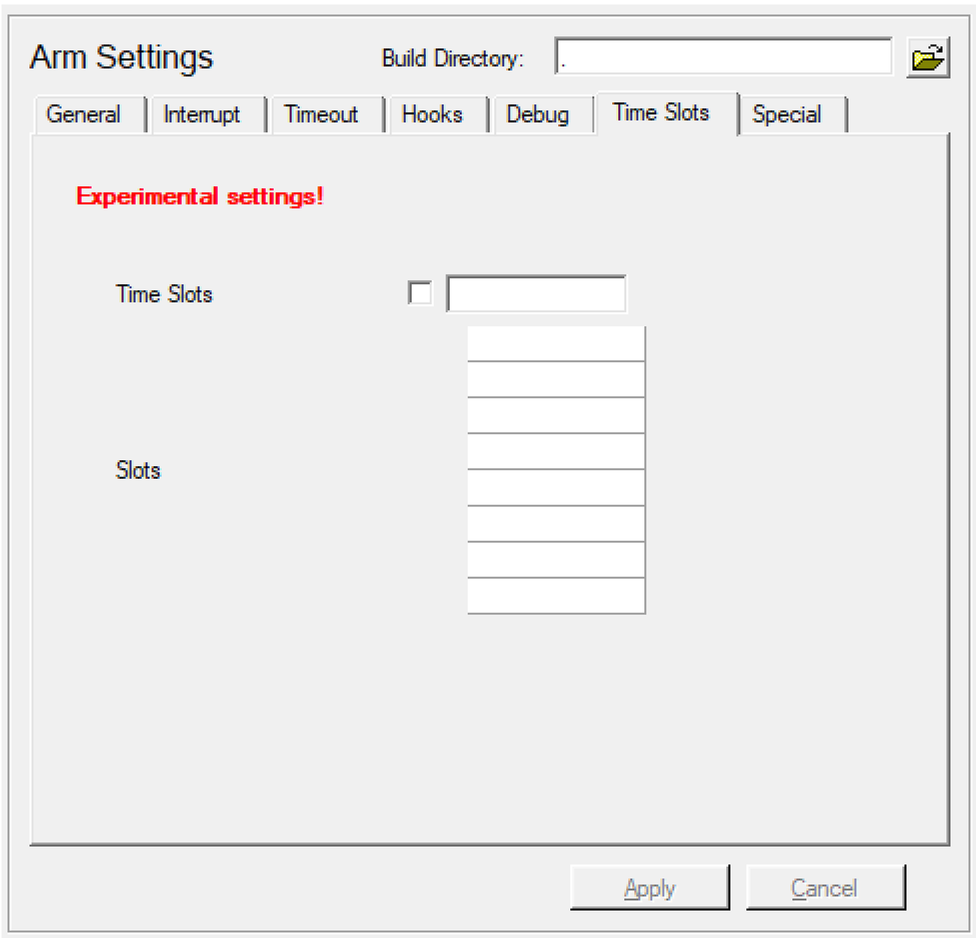

Figure 14. Time Slots Configuration Window

Internal use only, do not touch! If you have question please ask support service

2.8.7. Special Configuration tab

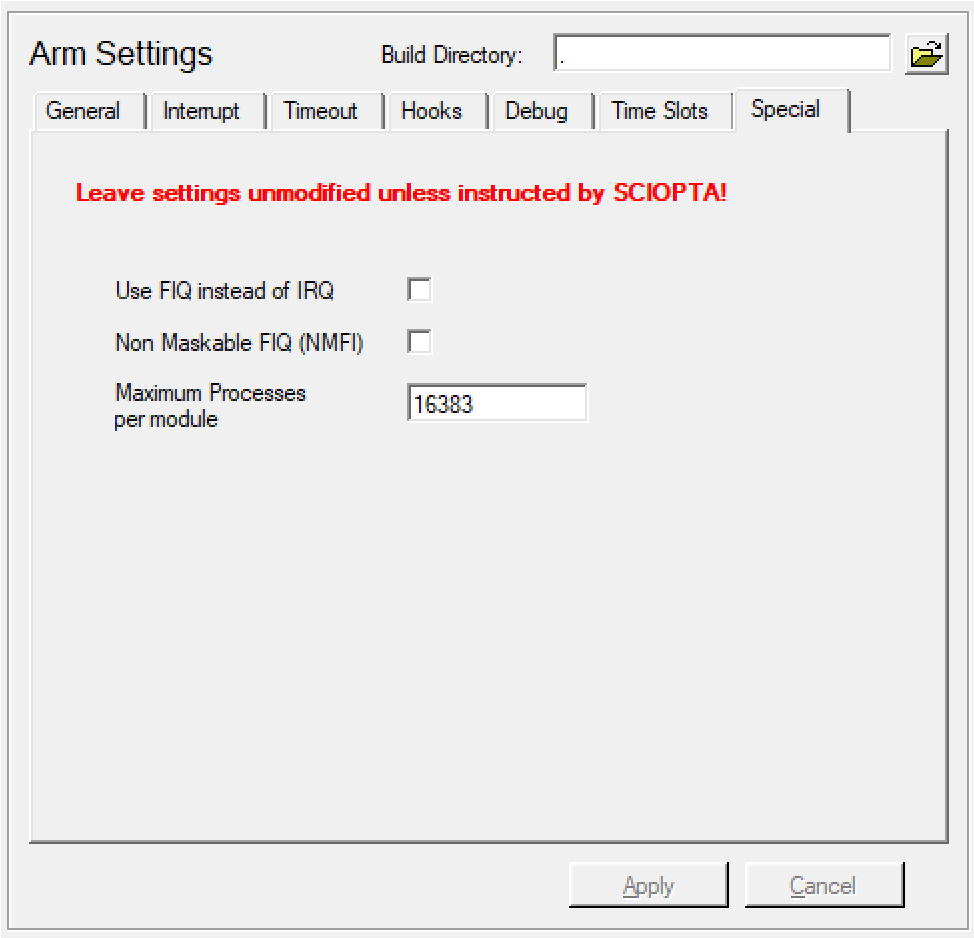

Figure 15. Special Configuration Window

`Internal use only, do not touch! If you have question please ask support service`

2.8.8. Applying Target System Configuration

If your SCIOPTA system settings are satisfactory click on the `Apply` button to accept and store it.

2.9. Configuring ARMv2 Systems

After selecting a system with your mouse, the corresponding parameter window on the right side will show the parameters for an ARM V2 System.

The system configuration for ARM V2 architecture is divided into 7 tabs: General, Interrupt, Timeout, Hooks, Time Slots, Debug, and Special.

2.9.1. General System Configuration tab

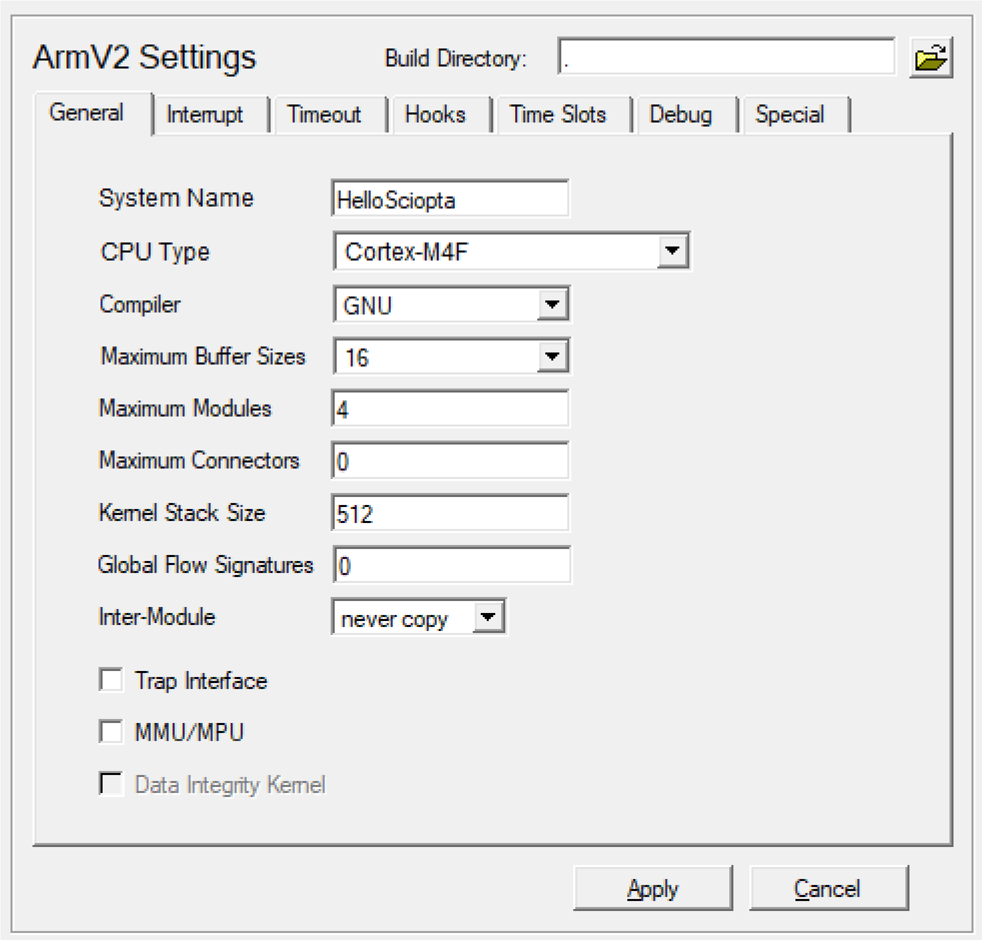

Figure 16. General Configuration Window

2.9.1.1. General tab Parameters

System Name	Name of the target system
	Enter the name of your system. Please note that the system module (module 0) in this system will get the same name.

CPU Type	CPU family for the selected architecture	
	ARMv4T+MMU	For ARM7TDMI and ARM9TDMI CPU-Family including MMU support.
	ARMv4T	For ARM7TDMI and ARM9TDMI CPU-Family.
	ARMv5TE	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family.
	ARMv5TE+FPU	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family including FPU support.
	ARMv5TE+MMU	For ARM946E-S, ARM966E-S, ARM996HS, and ARM1020E CPU-Family including MMU support.
	ARMv6	For ARM1136J(F)-S CPU-Family.
	ARMv6+FPU	For ARM1136J(F)-S CPU-Family including FPU support.
	ARMv6M	For Cortex-M0 and Cortex-M1 CPU-Family.
	Cortex-M3	For Cortex-M3 CPU-Family.
	Cortex-M4	For Cortex-M4 CPU-Family.
	Cortex-M4F	For Cortex-M4F CPU-Family including FPU support.
	ARMv7R	For Cortex-R4, Cortex-R5, Cortex-R7 and Cortex-R8 CPU-Family.
	ARMv7R+FPU	For Cortex-R4, Cortex-R5, Cortex-R7 and Cortex-R8 CPU-Family including FPU support.
	ARMv7A	For Cortex-A5, Cortex-A7, Cortex- A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family.
	ARMv7A+FPU	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU support.
	ARMv7A+FPU+D32	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU and VFPv4-D32 support.
	ARMv7A+FPU+NEON	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU with NEON support.
	ARMv7A+FPU+NEON+D32	For Cortex-A5, Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, Cortex-A15 and Cortex-A17 CPU-Family including FPU and VFPv4-D32 with NEON support.
	Cortex-A7	For Cortex-A7 CPU-Family.
	Cortex-A15	For Cortex-A15 CPU-Family.
	Cortex-A53-32	For Cortex-A53 CPU-Family.
	Cortex-A57-32	For Cortex-A57 CPU-Family.
	Xscale	For ARMv5TE CPU-Family.

Compiler	C/C++ Compiler	
	GNU	GCC `Valid for all compilers`
	IAR	IAR Embedded Workbench for Arm
	Metrowerks	C/C++ Compiler for Embedded PowerPC
	ADS/RVDS	RealView Compilation Tools (RVCT) ARM C/C++ Compiler (armcc) ARM C compiler for ARM Developer Suite (ADS)
	CCS	CCS C Compiler

Maximum Buffer Sizes	Maximum number of message buffer sizes	
	4, 8 or 16	If a process allocates a message there is also the size to be given. The user just gives the number of bytes needed. SCIOPTA is not returning the exact amount of bytes requested but will select one of a list of buffer sizes which is large enough to contain the requested number. This list can contain 4, 8 or 16 sizes which is configured here in the maximum buffer sizes entry. See also Sciopta Architecture Manual chapter Message Sizes.

Maximum Modules	Maximum number of SCIOPTA modules in the system	
	1 … 127	Here you can define a maximum number of modules which can be created in this system. The maximum value is 127 modules. It is important that you give here a realistic value of maximum number of modules for your system as SCIOPTA is initializing some memory statically at system start for the number of modules given here. See also Sciopta Architecture Manual chapter Modules.

Maximum Connectors	Maximum number of CONNECTOR processes in the system	
	0 … 127	CONNECTORS are specific SCIOPTA processes and responsible for linking a number of SCIOPTA systems. The maximum number of connectors in a system may not exceed 127 which correspond to the maximum number of systems. See also Sciopta Architecture Manual chapter Distributed Systems.

Kernel Stack Size	Size of the global kernel stack	
	512	Currently not used. Entered values are not considered.

Global Flow Signatures	Signature value. Initialize global flow signature	
	512	Currently not used. Entered values are not considered.

Inter-Module	Defines if messages between modules are copied or not	
	never copy	Messages between modules are never copied.
	always copy	Messages between modules are always copied.
	friends	The message copy behaviour is defined by the friendship setting between the modules.
	See also Sciopta Architecture Manual chapter Messages Modules and Module Friend Concept	

Trap Interface	Enables the trap interface
	In a typical monolithic SCIOPTA systems the kernel functions are directly called. In more complex dynamic systems using load modules or MMU/MPU protected modular systems the kernel functions cannot be accessed any more by direct calls. SCIOPTA offers a trap interface. In such systems you need to enable the Trap Interface and assemble the file syscall.S. See also SCIOPTA Getting Start Manual chapter Trap Interface

MMU/MPU	Enables the MMU/MPU
	You must select the MMU checkbox if you want to use the Cortex-M and Cortex-R MPU. This will enable the MPU functions in the kernel.

2.9.2. Interrupt Configuration tab

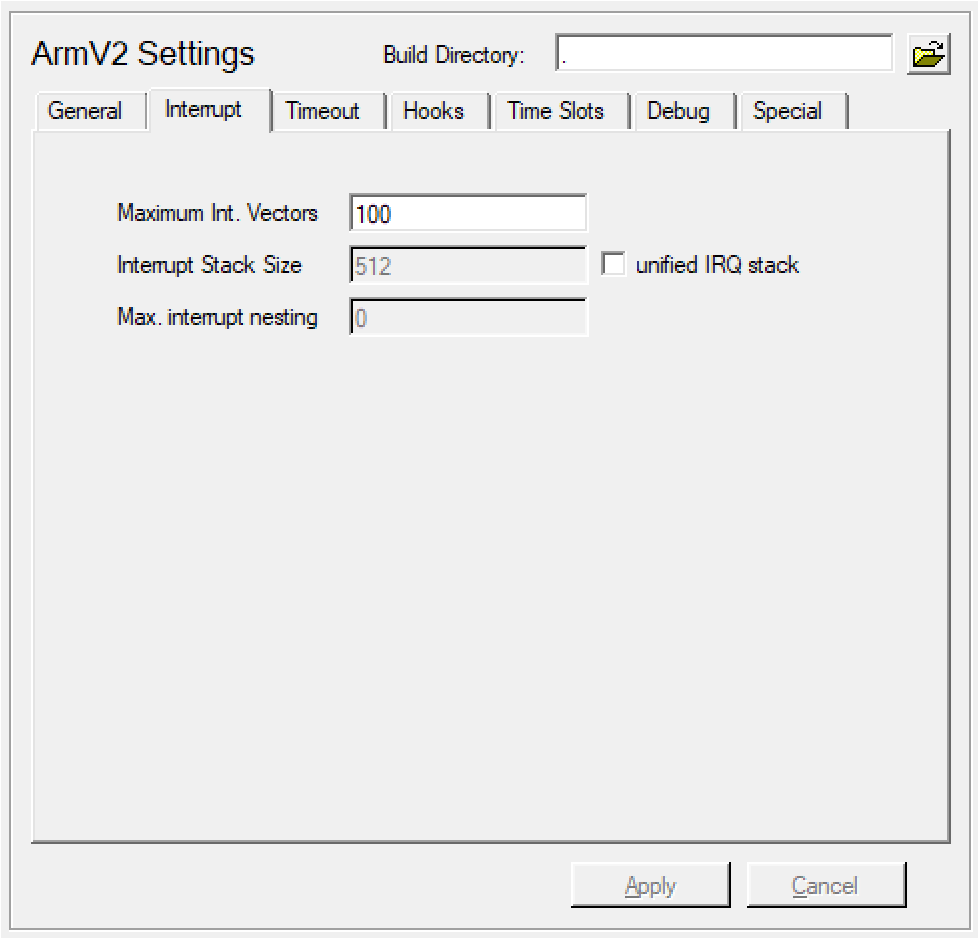

Figure 17. Interrupt Configuration Window

2.9.2.1. Interrupt tab Parameters

Maximum Int. Vectors	Maximum number of interrupt vectors	
	0 … 255	Select maximum interrupt vectors used in your Cortex-M and Cortex-R application system. See also SCIOPTA Getting Start Manual chapter Interrupt Vector Table.
Interrupt Stack Size	Size of the global interrupt stack	
	<size>	Only used if **“unified IRQ stack”** checkbox is selected. The stack size given must be big enough to hold the stacks of the interrupt processes with the biggest stack needs taken in account the interrupt nesting.
Max interrupt nesting	Maximum interrupt nesting level	
	0	No nesting
	<level>	Maximum nesting level of interrupt processes in the system.

2.9.3. Timeout Configuration tab

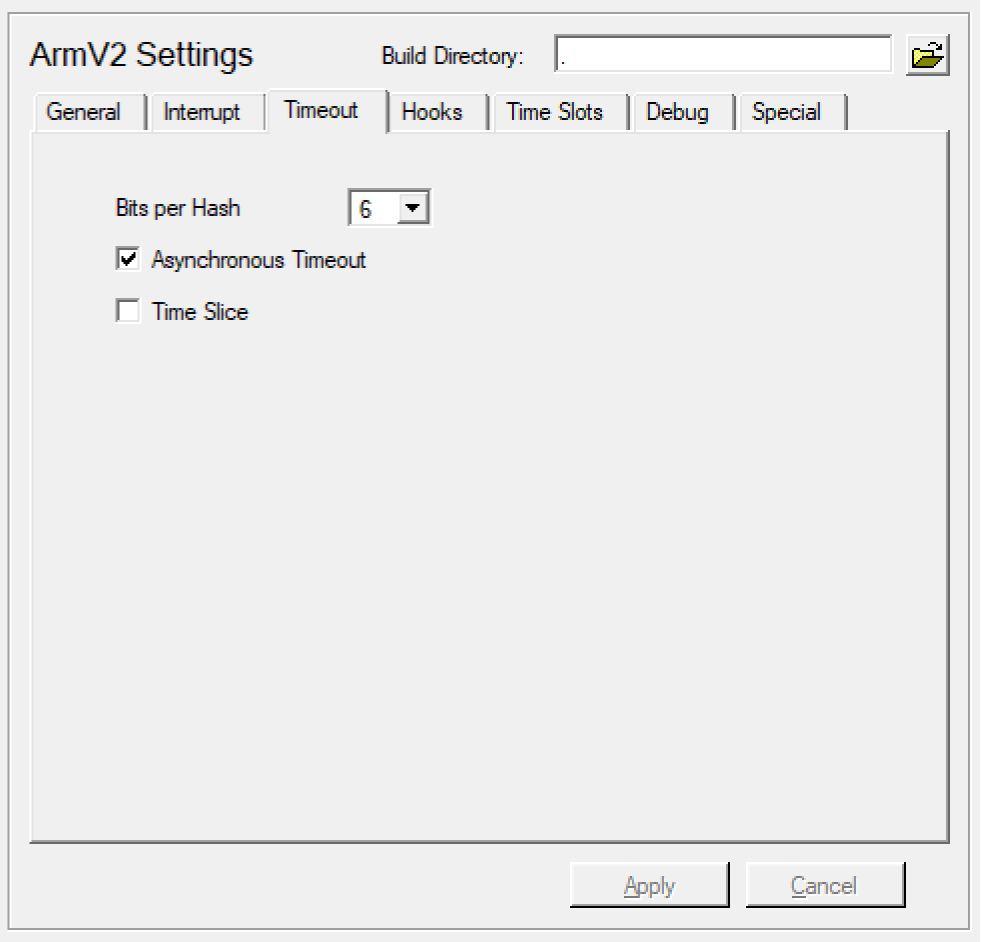

Figure 18. Timeout Configuration Window

2.9.3.1. Timeout tab Parameters

Bits per Hash	Some targets allow setting of hash size	
	2..6	The smaller the hash tables the more often a process has to be re-added into another table until the timeout has expired. The total size of a hash table is 4*(1<<SC_BITS_PER_HASH)*sizeof(__ptrsize_t)*2. Thus for a 32Bit CPU ranges from 128 bytes to 2048 bytes.
Asynchronous Timeout	Enables the time-out server	
	The SCIOPTA time-out server can be enabled and disabled. Disabling the timeout server will reduce the kernel size. See also Sciopta Architecture Manual chapter Time-Out Server.	
Time Slice	Enables the Time Slice	
	The Time slice of a prioritized or timer process.	

2.9.4. Hooks Configuration tab

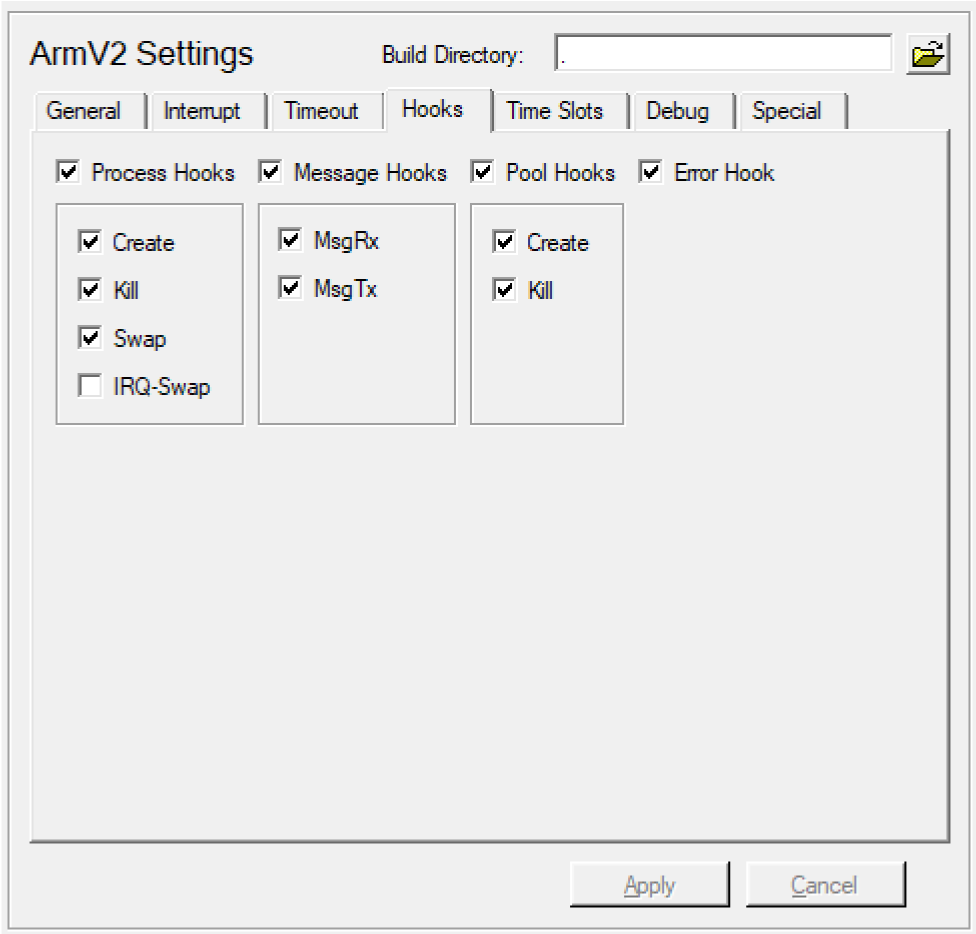

Figure 19. Hooks Configuration Window

Hooks are user written functions which are called by the kernel at different locations. They are only called if the user defined them at configuration. User hooks are used for a number of different purposes and are system dependent.

You can enable the hooks separately by selecting the corresponding check box or all hooks belonging to a group all together.

Please consult Sciopta Architecture Manual chapter Hooks for more information about SCIOPTA hooks.

2.9.5. Debug Configuration tab

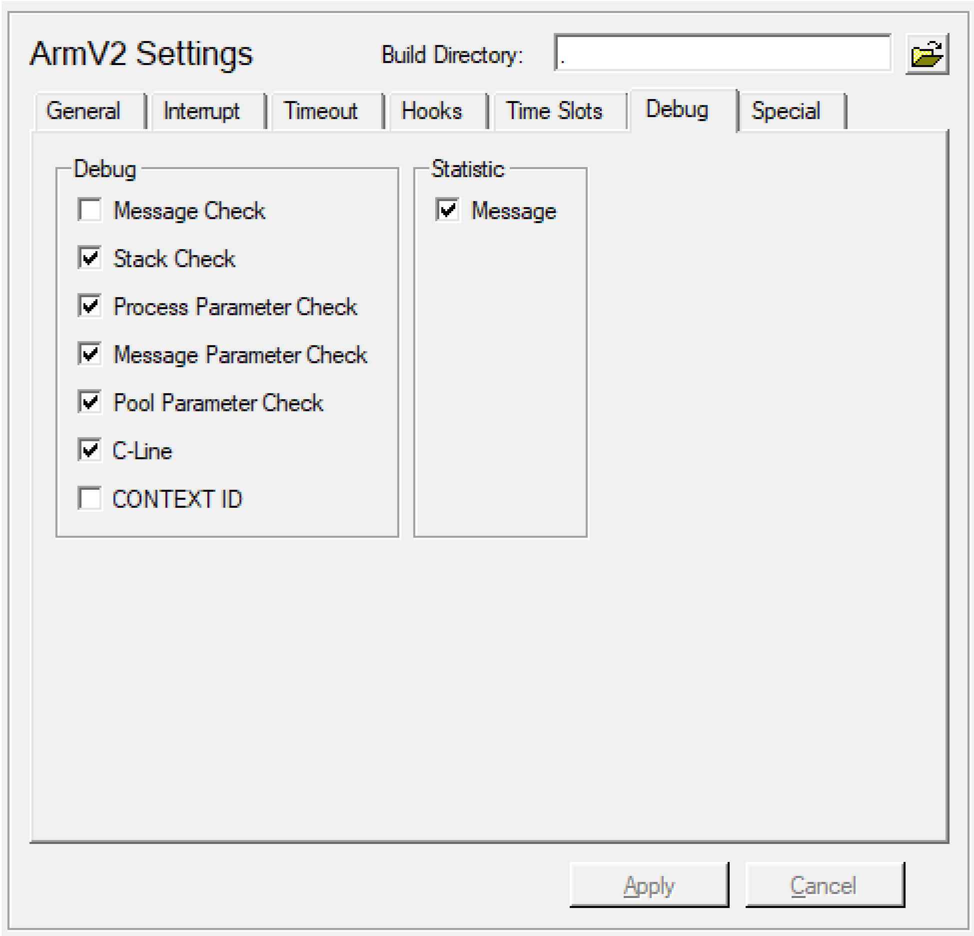

Figure 20. Debug Configuration Window

2.9.5.1. Debug tab Parameters

Message Check	Enables the message check functions in the kernel All internal message test functions will be included in the kernel.
Stack Check	Enables the stack check functions.
Process Parameter Check	Enables process parameter checks Parameter check of the process system calls will be included in the kernel.
Message Parameter Check	Enables message parameter checks. Parameter check of the message system calls will be included in the kernel.
Pool Parameter Check	Enables pool parameter checks. Parameter check of the pool system calls will be included in the kernel.

C-Line	Enables C-line informations. If you are selecting this check box the kernel will include line number information which can be used by the SCIOPTA DRUID Debug System or an error hook. Line number and file of the last system call is recorded in the per process data. Note: If configured, you may add a compiler command line macro SC_CDEBUG=1.

CONTEXT ID	Enables CONTEXT ID informations.

Message Statistics	Includes message statistics. The kernel will maintain a message statistics data field in the pool control block where information such as number of message allocation can be read.

2.9.6. Time Slots Configuration tab

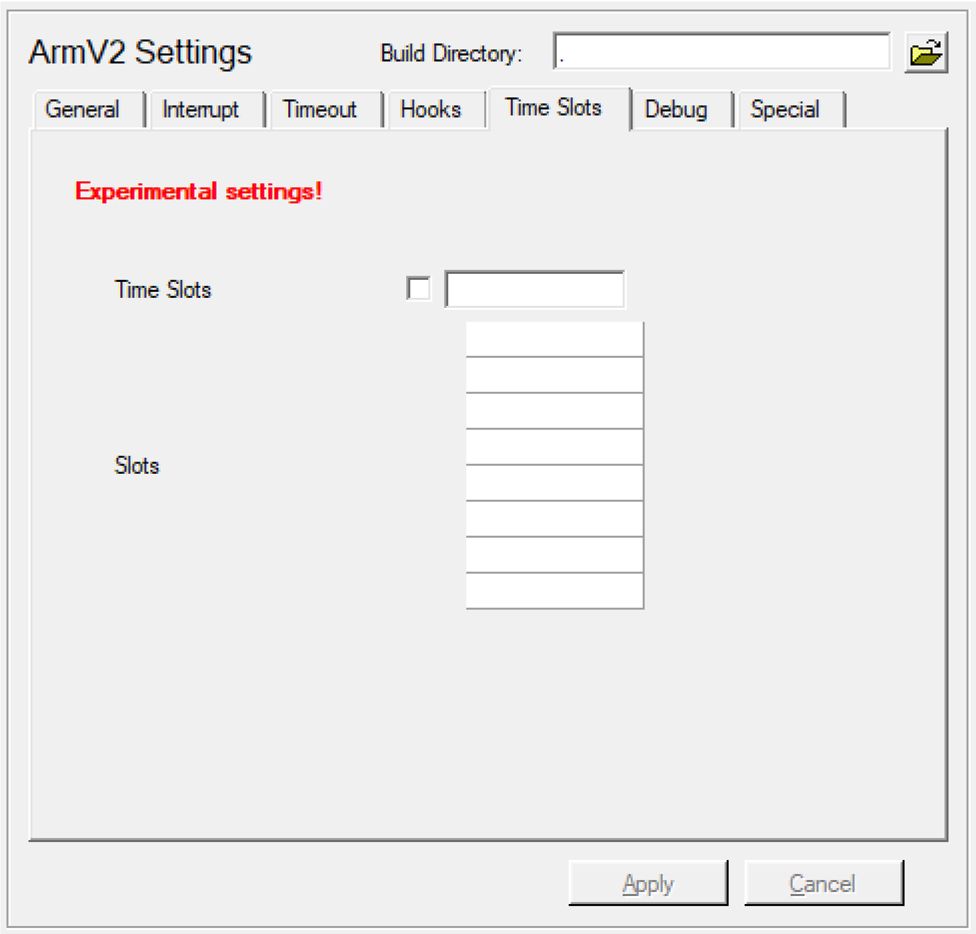

Figure 21. Time Slots Configuration Window

`Internal use only, do not touch! If you have question please ask support service`

2.9.7. Special Configuration tab

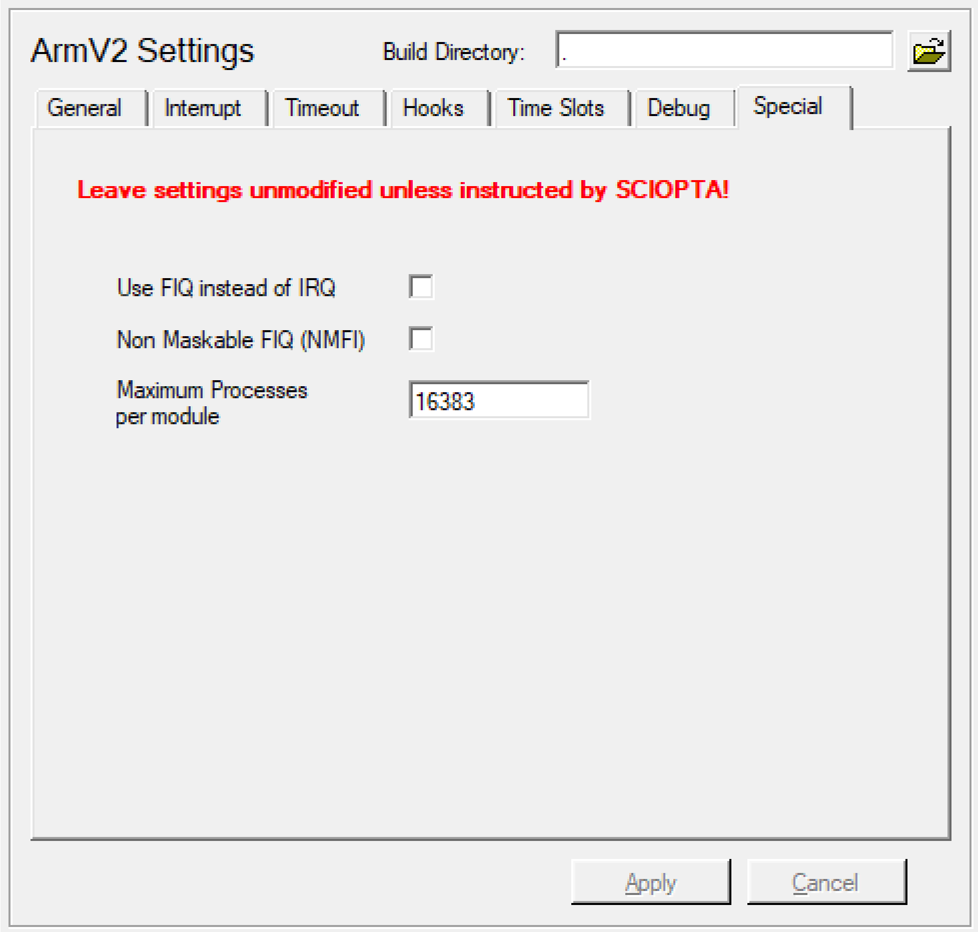

Figure 22. Special Configuration Window

`Internal use only, do not touch! If you have question please ask support service`

2.9.8. Applying Target System Configuration

If your SCIOPTA system settings are satisfactory click on the `Apply` button to accept and store it.

2.10. Creating Modules

From the system level you can create new modules. Move the mouse pointer over the system and right-click the mouse.
Remember that the system module has been created automatically after defining a new target system.

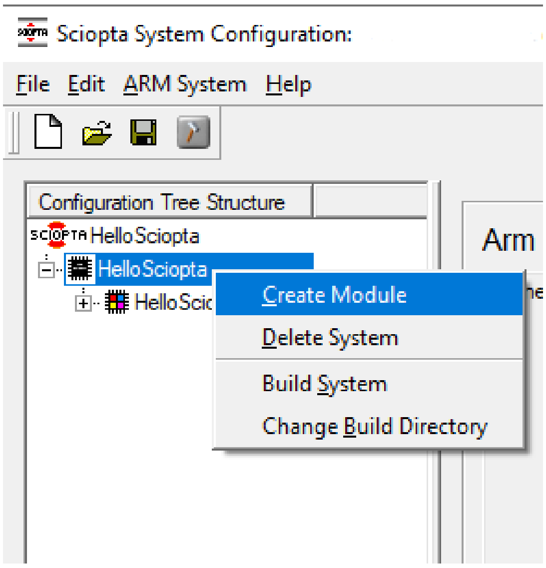

Figure 23. Create Module Menu

A pop-up menu appears and allows you to create a new module.
The same selection can be made by selecting the Target System from the menu bar.
A new module for your selected target with a default name and an init process in the module will be created.

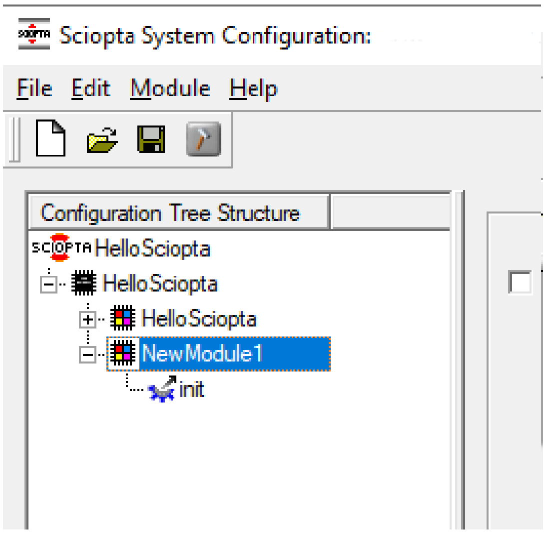

Figure 24. Module Name

You can create up to 127 modules.

You are now ready to configure the individual modules.

2.11. Configuring Modules

After selecting a module with your mouse, the corresponding parameter window on the right side will show the module parameters.

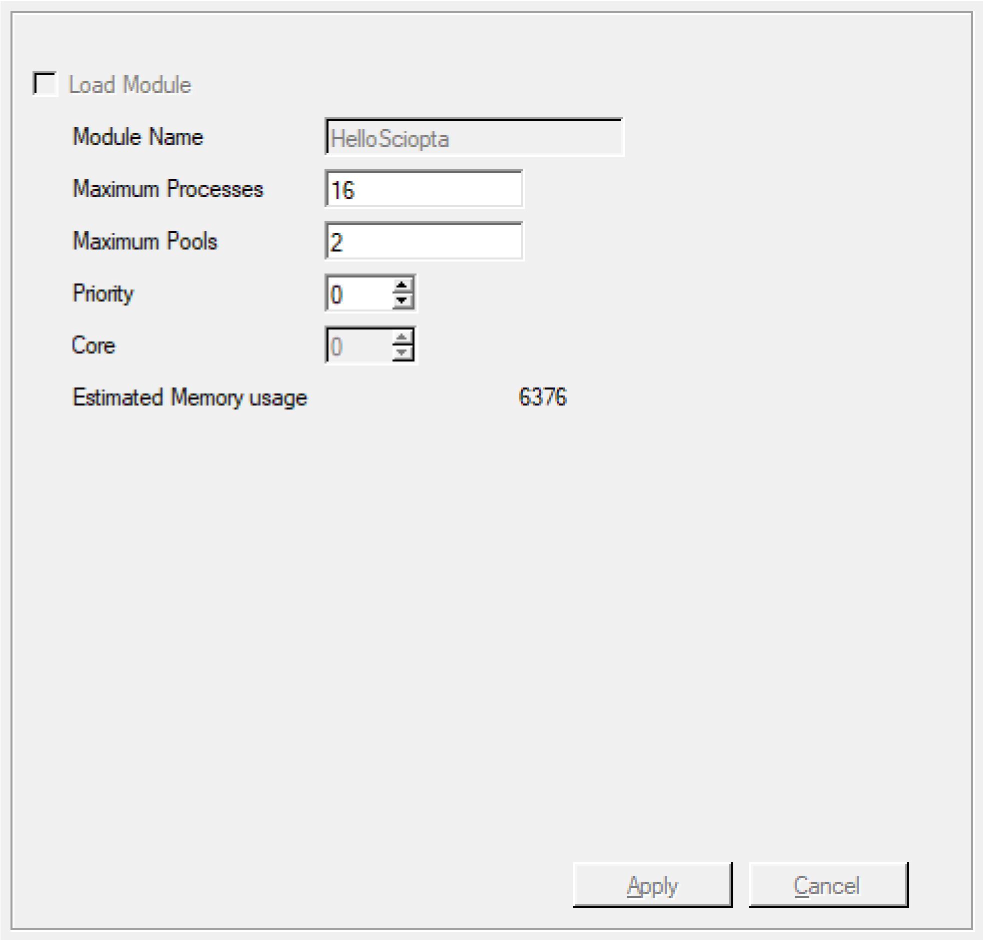

Figure 25. Module Parameters

2.11.1. Module Parameters

Load Module	Module is a load module. Check this box if the module will be loaded at run-time into the system. This check box is not available for the system module.	
Module Name	Name of the module. Enter the name of the module. If you have selected the system module (the first module or the module with the id 0) you cannot give or modify the name as it will have the same name as the target system.	
Maximum Processes	Maximum number of processes in the module.	
	1 … 16383	The kernel will not allow to create more processes inside the module than stated here.
Maximum Pools	Maximum number of message pools.	
	1 … 128	Enter the maximum number of pools in the module. The kernel will not allow to create more pools inside the module than stated here.
Priority	Module priority.	
	0 … 31	Enter the priority of the module. 0 is the highest and 31 is the lowest module priority. See also Sciopta Architecture Manual chapter Module Priority.

2.11.2. Applying Module Configuration

If your module settings are satisfactory click on the `Apply` button to accept and store it.

2.12. Creating Processes and Pools

From the module level you can create new processes and pools. Move the mouse pointer over the module and right-click the mouse.

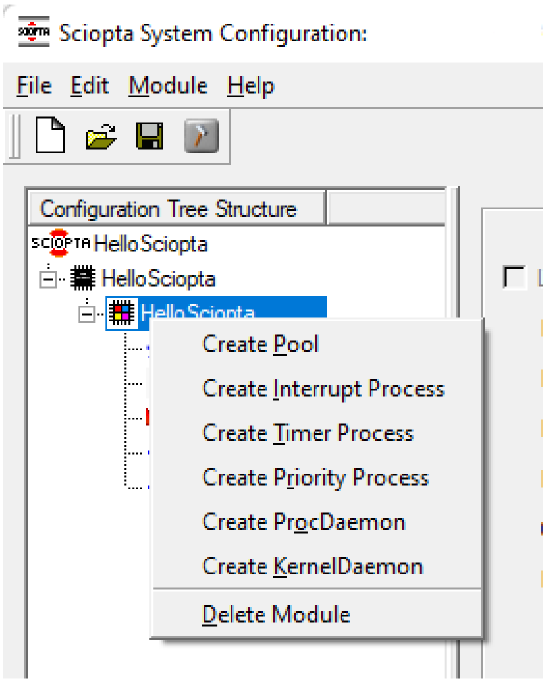

Figure 26. Creating Processes and Pools

A pop-up menu appears and allows you to create pools, interrupt processes, timer processes, prioritized processes and if it is the system module also the process daemon and the kernel daemon.

The Process Daemon (ProcDaemon) and Kernel Daemon (KernelDaemon) can only be created in the system module.
The same selection can be made by selecting the Module menu from the menu bar.

2.12.1. Configuring the Init Process

After selecting the init process with your mouse the parameter window on the right side will show the configuration parameters for the init process. There is always one init process per module and this process has the highest priority. Only the stack size of the init process can be configured.

Please consult Sciopta Architecture Manual chapter Init Processes for more information about init processes.

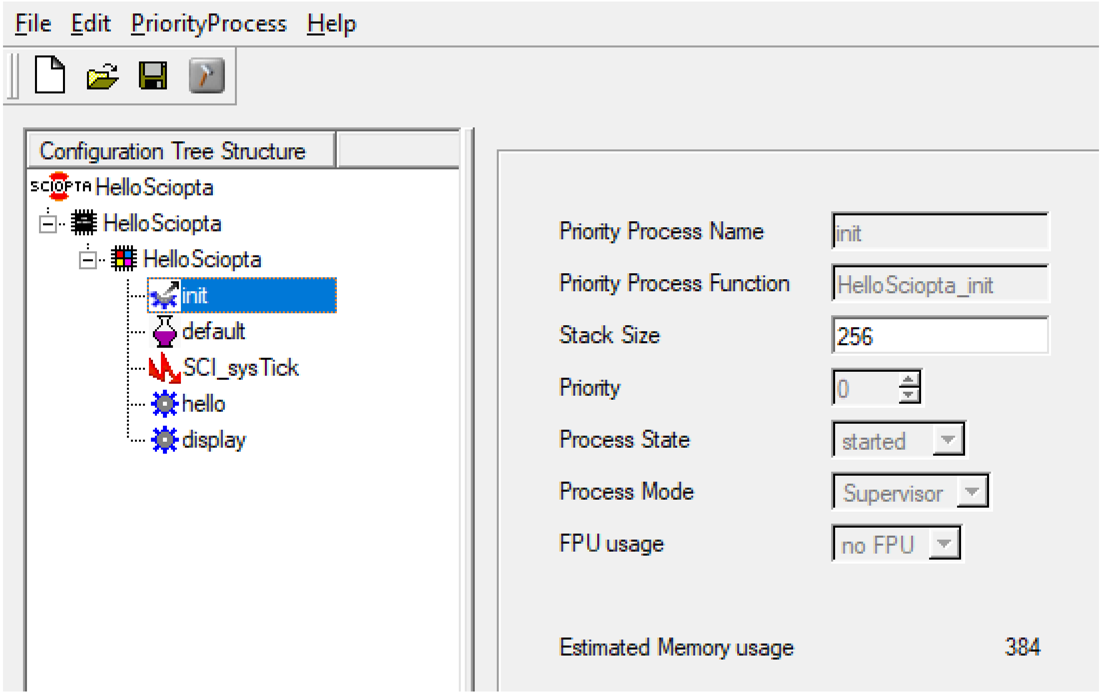

Figure 27. Init Process Parameters

2.12.2. Parameter

Priority	Init process stack size. Enter a big enough stack size.

2.12.3. Applying Init Process Configuration

If your init process settings are satisfactory click on the `Apply` button to accept and store it.

2.13. Interrupt Process Configuration

After selecting an interrupt process with your mouse the parameter window on the right side will show the configuration parameters for the interrupt process.

Please consult Sciopta Architecture Manual chapter Interrupt Processes for more information about interrupt processes.

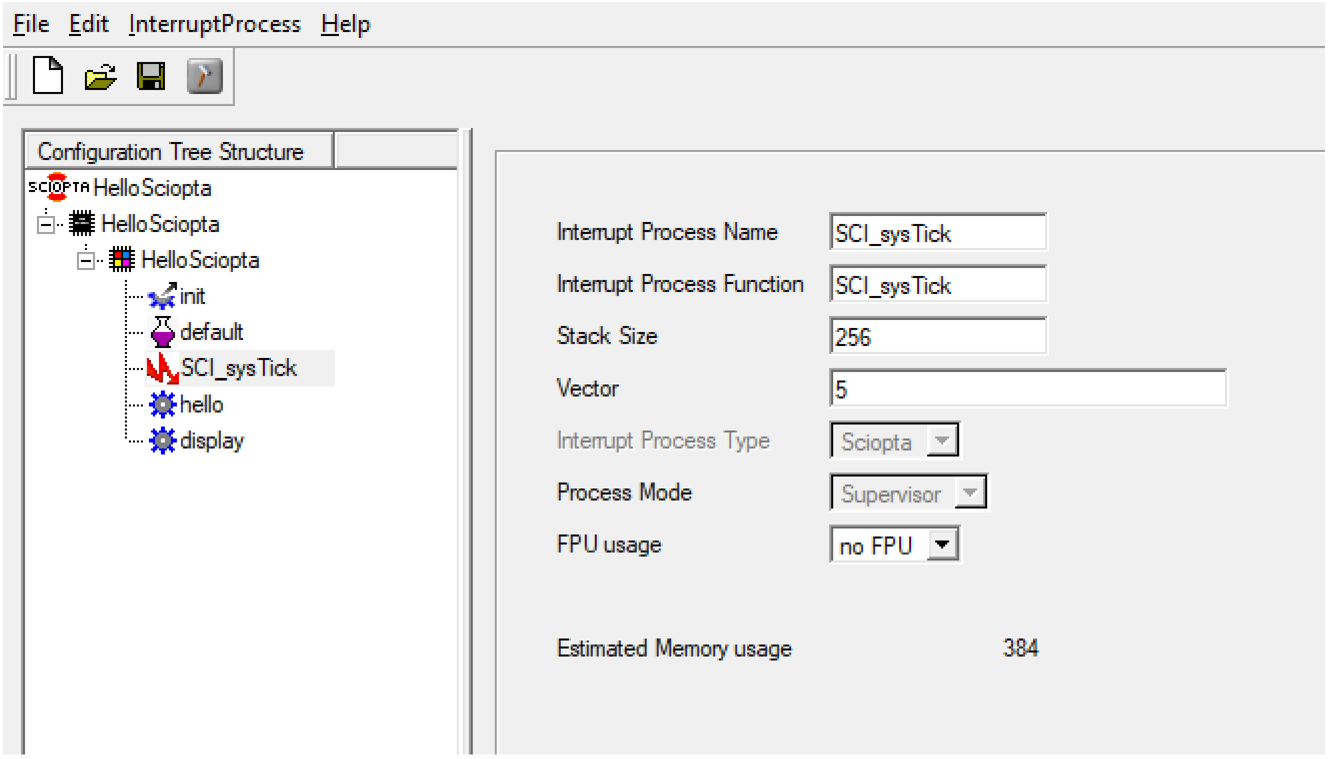

Figure 28. Interrupt Process Parameters

2.13.1. Parameters

Interrupt Process Name	Interrupt process name
Interrupt Process Function	Interrupt process function entry Function name of the interrupt process function. This is the address where the created process will start execution. More than one interrupt processes (names) can have the same interrupt process function.
Stack Size	Interrupt process stack size Enter a big enough stack size of the created interrupt process in byte. A value can only be entered when the **“unified IRQ stack”** checkbox in the target system configuration window is not selected. See chapter General System Configuration tab.
Vector	Interrupt vector Enter the interrupt vector connected to the interrupt process. Refere to the SoCs hardware manual for the correct vector number.

FPU usage	Selects if a Floating Point Unit exists and will be used.	
	no FPU	Process does not use the FPU/is not allowed to use the FPU.
	FPU	Process may use the FPU Note: Depending on the architecture, the FPU may be active no matter.

2.13.2. Applying Interrupt Process Configuration

If your interrupt process settings are satisfactory click on the `Apply` button to accept and store it.

2.14. Timer Process Configuration

After selecting a timer process with your mouse the parameter window on the right side will show the configuration parameters for the timer process.

Please consult Sciopta Architecture Manual chapter Timer Processes for more information about timer processes.

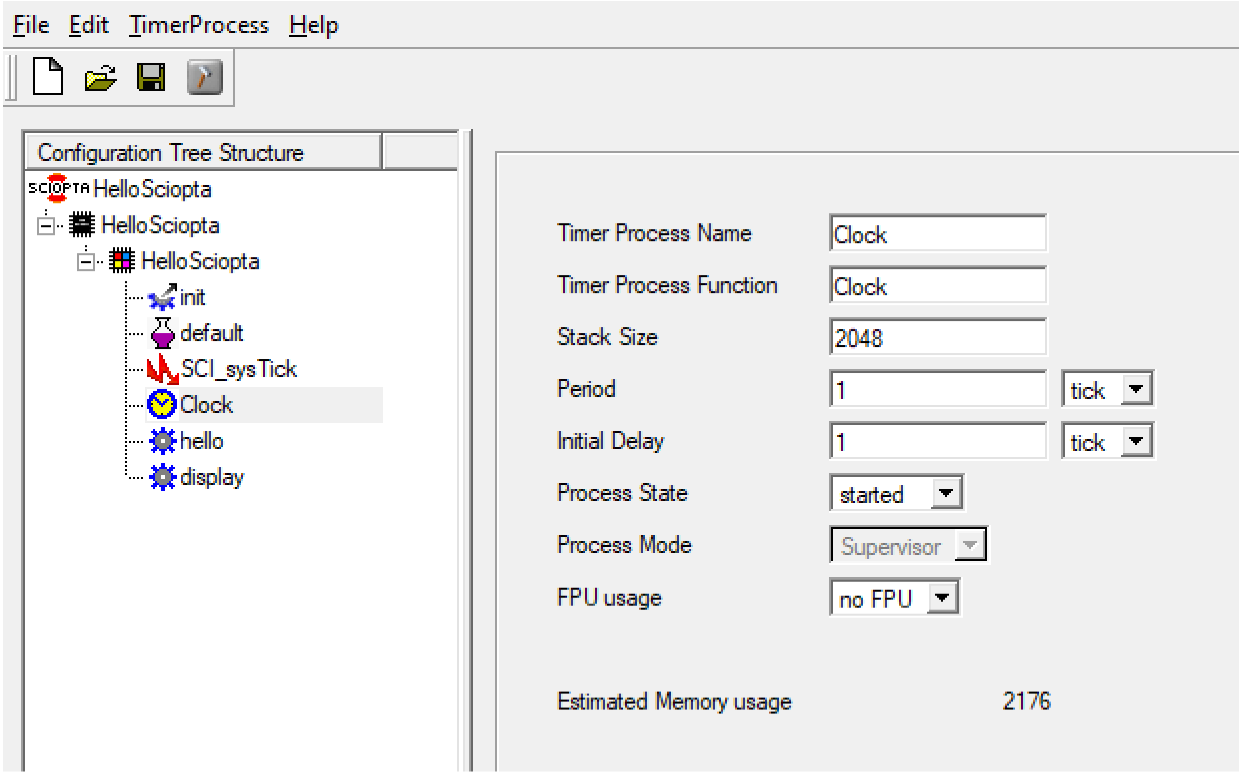

Figure 29. Timer Process Parameters

2.14.1. Parameters

Timer Process Name	Timer process name
Timer Process Function	Timer process function entry. Function name of the timer process function. This is the address where the created process will start execution.
Stack Size	Timer process stack size. Enter a big enough stack size of the created timer process in bytes. A value can only be entered when the **"unified IRQ stack"** checkbox in the target system configuration window is not selected. See chapter General System Configuration tab .
Period	Timer process interval time Period of time between calls to the timer process in ticks or in milliseconds.

Initial Delay	Initial delay time Initial delay before the first time call to the timer process in ticks or milliseconds. To avoid tick overload due to timer processes having the same period.	
Process State	Starting state of the timer process	
	started	The timer process will be started after creation.
	stopped	The process is stopped after creation. Use the sc_procStart system call to start the process.
FPU usage	Selects if a Floating Point Unit exists and will be used.	
	no FPU	Process does not use the FPU/is not allowed to use the FPU.
	FPU	Process may use the FPU Note: Depending on the architecture, the FPU may be active no matter.

2.14.2. Applying Timer Process Configuration

If your timer process settings are satisfactory click on the `Apply` button to accept and store it.

2.15. Prioritized Process Configuration

After selecting a prioritized process with your mouse the parameter window on the right side will show the configuration parameters for the prioritized process.

Please consult Sciopta Architecture Manual chapter Prioritized Processes for more information about prioritized processes.

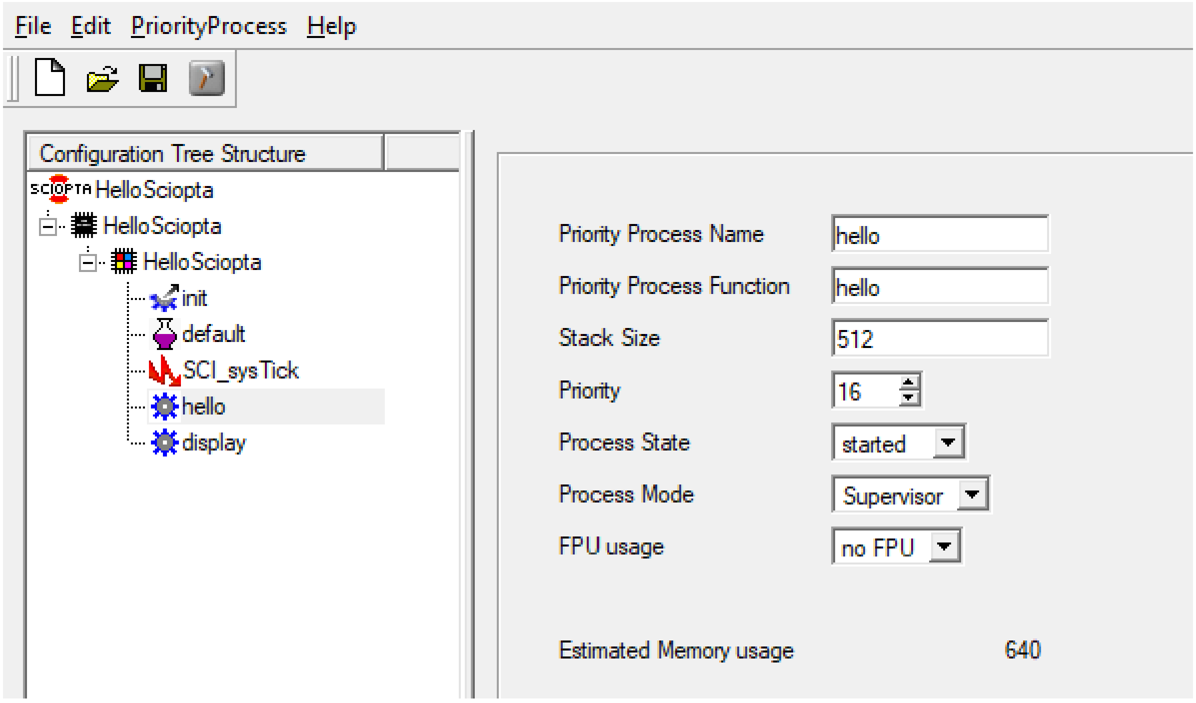

Figure 30. Prioritized Process Parameters

2.15.1. Parameters

Priority Process Name	Process name	
Priority Process Function	Process function entry Function name of the prioritized process function. This is the address where the created process will start execution. More than one prioritized process (names) can have the same prioritized process function.	
Stack Size	Process stack size. Enter a big enough stack size of the created prioritized process in bytes.	
Priority	Priority of the process.	
	0 … 31	0 is the highest and 31 is the lowest module priority. An error will be generated if the priority is higher than the module priority. See also Sciopta Architecture Manual chapter Module Priority

Process State	Starting state of the prioritized process	
	started	The process will be on READY state. It is ready to run and will be swapped-in if it has the highest priority of all READY processes.
	stopped	The process is stopped after creation. Use the sc_procStart system call to start the process.

Processor Mode	Selects interrupt processor mode.	
	Supervisor	The process runs in CPU supervisor mode.
	User	The process runs in CPU user mode.

FPU usage	Selects if a Floating Point Unit exists and will be used.	
	no FPU	Process does not use the FPU/is not allowed to use the FPU.
	FPU	Process may use the FPU Note: Depending on the architecture, the FPU may be active no matter.

2.15.2. Applying Prioritized Process Configuration

If your prioritized process settings are satisfactory click on the `Apply` button to accept and store it.

2.16. Pool Configuration

After selecting a pool with your mouse the parameter window on the right side will show the configuration parameters for the pool.

Please consult Sciopta Architecture Manual chapter Message Pools for more information about Pool Configuration.

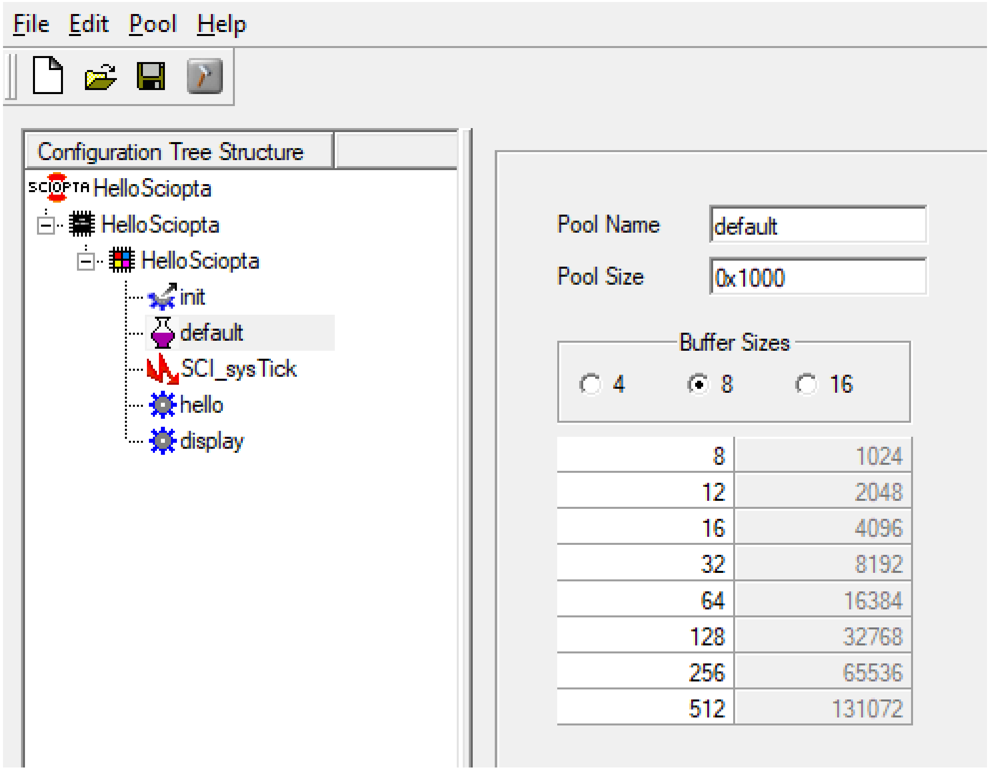

Figure 31. Pool Parameters

2.16.1. Parameters

Pool Name	Name of the message pool.	
Pool Size	Size of the message pool. See Sciopta Architecture Manual chapter Pool Size.	
Buffer Sizes	Number of buffer sizes.	
	4, 8 or 16	Define the different buffer sizes for your selection. See Sciopta Architecture Manual chapter Message Sizes

2.16.2. Applying Pool Configuration

If your pool settings are satisfactory click on the `Apply` button to accept and store it.

2.17. Build

2.17.1. Configuration Files

The **SCONF** will generate the following files which need to be included into your SCIOPTA project.

sconf.h This is a header file which contains the kernel configuration. This file will be included by the kernel during assembling and sconf.c while compiling. You need to include this in all your files which need configuration information.

sconf.c This is a C source file which contains the system initialization code. You need to compile this file in the system building process.

2.17.2. Build The System

To build the two files right-click on the system. Select the menu Build System.

The files **sconf.h** and **sconf.c** will be generated into your defined build directory.

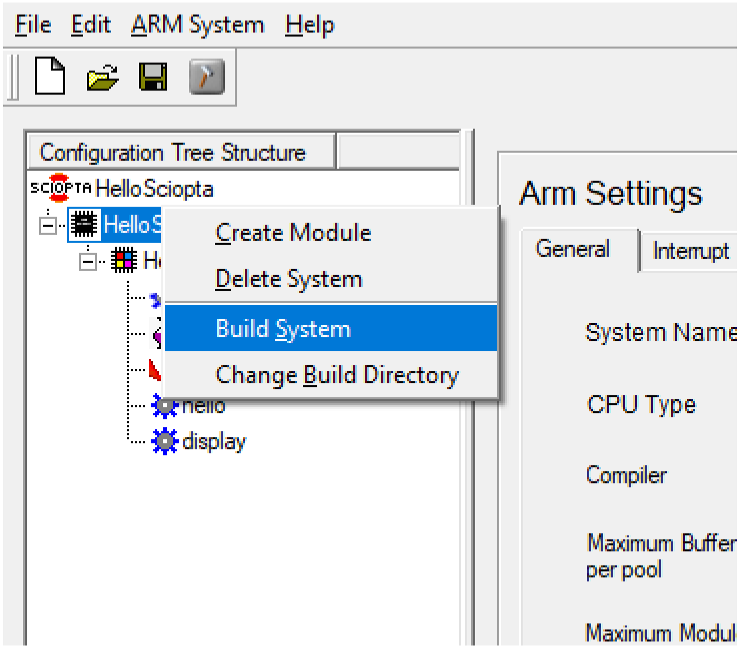

Figure 32. Building

2.17.3. Changing the Build Directory

When you are creating a new system, **SCONF** ask you to give the directory where the two generated files will be stored.
You can modify this build directory for each system individually by clicking to the system which you want to build and right click the mouse.

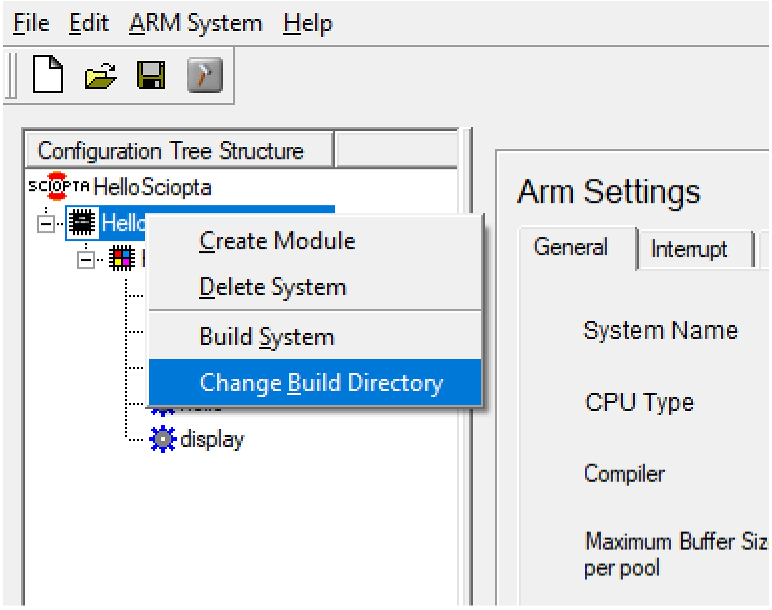

Figure 33. Change Build Directory

Select the last item in the menu for changing the build directory.
The actual Build Directory is shown in the System Settings Window:

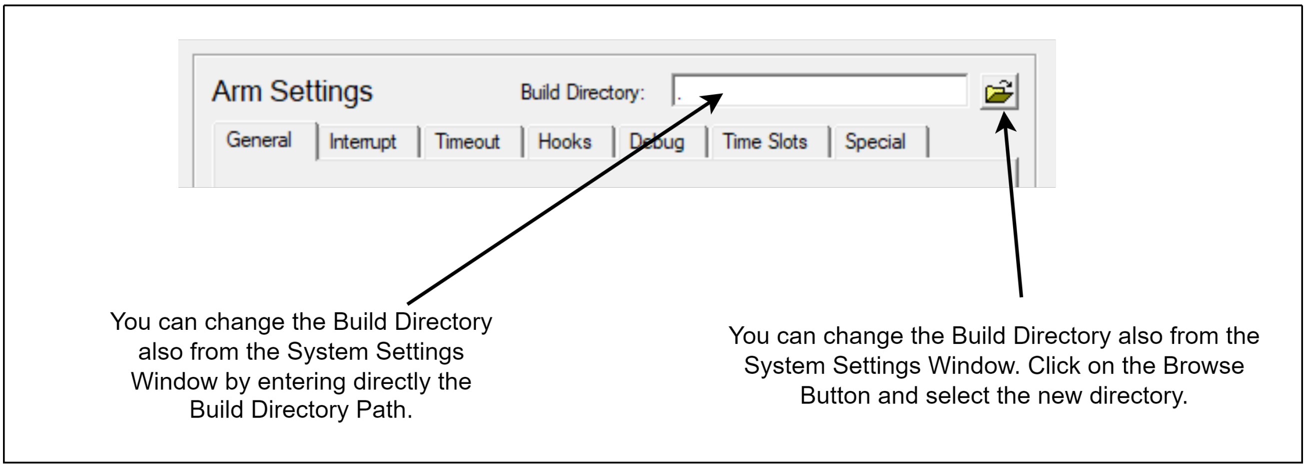

Figure 34. Build Directory

2.17.4. Build All

If you have more than one system in your project, you can build all systems at once by clicking on the **Build All** button.
Select the Build All button from the button bar to generate the set of three files for each system.

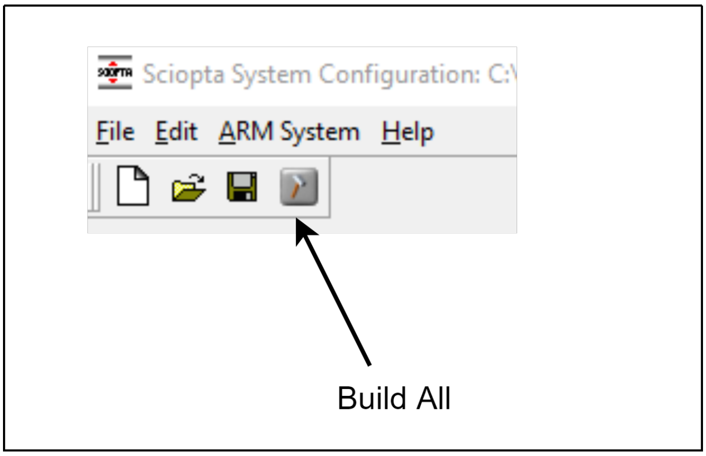

Figure 35. Build All

The files **sconf.h** and **sconf.c** will be generated for every target into the defined build directories of each target which exists in the project.
SCONF will prompt for generating the files for each system.

Please note:

`You need to have different build directories for each system as the names of the three generated files are the same for each system.`

2.18. Command Line Version

2.18.1. Introduction

The **SCONF** configuration utility can also be used from a command line.

This is useful if you want to modify or create the XML configuration file manually or if the XML configuration file will be generated by a tool automatically and you want to integrate the configuration process in a makefile. The best way to become familiar with the structure of the XML file is to use the graphic **SCONF** tool once and save the XML file.

2.18.2. Syntax

By calling the **SCONF** utility with a -c switch, the command line version will be used automatically.

```
<install_dir>\bin\win32\sconf.exe -c <XML File>
```

You need to give also the extension of the XML file.

2.19. Interrupt Vector Symbols

It’s possible to set or modify an SCONF option by defining symbols in the XML file. This is particularly useful for filling in entries for the interrupt vector table.

The default handler will be a weak symbol and just dead loops. For exemple If **"SCONF_USE_SCONF_PRELOAD_H"** is defined for compilation, then **sconf_preload.h** will be included.

All symbols used can be defined in the sconf_preload.h or command line (like -D SERIAL_IRQ=53).

2.19.1. Syntax

The header file should be included in **sconf.c**

```
#ifdef USE_SCONF_PRELOAD_H
#include "sconf_preload.h"
#endif
```

3. Manual Versions

3.1. Manual Version 1.0

- Initial
  - Initial version.

3.2. Manual Version 1.1

- Chapter folding
  - Initial chapters are folded.
  - Some clarifcations.
  - Layout fixes.