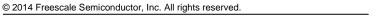
Document Number: MQXGSDS5 Rev. 06, 08/2014

Getting Started with ARM[®] Development Studio 5 (DS-5[™]) with Freescale MQX[™] RTOS

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	4.1.1
DESCRIPTION:	Using ARM [®] Development Studio 5 (DS-5™) with Freescale MQX™ RTOS
RELEASE DATE:	August, 2014





How to Reach Us:

Home Page: www.freescale.com

Web Support: http://www.freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Vybrid and Tower are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM, ARM Powered logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. © 2008-2014 Freescale Semiconductor, Inc.





Table of Contents

Getting Started with ARM [®] Development Studio 5 (DS-5™) with Freescale MQX™ RTOS	i
1 Read Me First	.2
2 Building the MQX RTOS Libraries	.3
2.1 Vybrid-kit special licenses	.5
3 Running and Debugging MQX RTOS application	.6
3.1 Debugging Primary Core - MQX RTOS Hello World program	
3.3 Multi-core debugging1	11
3.4 Debugging the Application loaded by MQX RTOS Boot Loader	14
4 MQX RTOS Task Aware Debugging in Development Studio 5 (DS-5) IDE1	16

1 Read Me First

This document describes how to build, debug, and run Freescale MQX[™] RTOS programs in the ARM[®] Development Studio 5 (DS-5[™]) development suite.

See *Getting Started with Freescale MQX[™] RTOS* and other user documentation included within the latest Freescale MQX RTOS distribution for further information on board specific build targets, jumper and HW settings, MQX RTOS API documentation, etc.

Getting Started with Freescale MQX™ RTOS and ARM DS-5 Rev. 06, 08/2014

2 Building the MQX RTOS Libraries

This chapter concentrates on steps specific to Development Studio 5 (DS-5) tool chain only. For details on generic build process and compile time configuration, see Chapter 2 of the *Getting Started with Freescale MQX*TM *RTOS*.

First, install the MQX RTOS Eclipse plugin using *Help\Install New Software\Add\Archive...* menu. Then, select the following archive: <max install dir>/tools/ds5/ds5 update site.zip

To rebuild the MQX RTOS libraries, import the

<mqx_install_dir>/build/<board>/ds5/build_libs.wsd working set description file using *File\Import\MQX\Import Working Sets* menu. The MQX RTOS library projects will be imported to DS-5 working space together with build configurations settings.

Import	
Select Choose import source.	Ľ
Select an import source:	
General C/C++ CVS Install Remote Systems Run/Debug Scatter File Editor Target Configuration Editor Target Configuration Editor Target Configuration Editor Target Configuration Editor Target Configuration Editor Target Configuration Editor	
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

Figure 1- MQX RTOS import working sets

• The following projects will be imported to your workspace

```
<mqx_install_dir>/mqx/build/ds5/bsp_<board>/.project
<mqx_install_dir>/mqx/build/ds5/psp_<board>/.project
<mqx_install_dir>/mfs/build/ds5/mfs_<board>/.project
<mqx_install_dir>/rtcs/build/ds5/rtcs_<board>/.project
<mqx_install_dir>/usb/host/build/ds5/usbh_<board>/.project
<mqx_install_dir>/usb/device/build/ds5/usbd_<board>/.project
<mqx_install_dir>/usb/device/build/ds5/usbd_<board>/.project
```

```
Freescale Semiconductor
```

```
<mqx_install_dir>/shell/build/ds5/shell_<board>/.project
```

Select the target and platform and build the libraries - hit the compile all button
 twnrf65gs10_m4 | Dt
 . All projects will be built in the selected configuration. The "Debug" configuration is dedicated for easy application debugging while the "Release" target has compiler and linker optimization set to maximum.

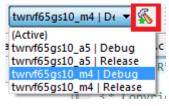


Figure 2- Debug/release

2.1 Vybrid-kit special licenses

In addition to the regular DS5 licenses there are two special Vybrid licenses available.

- Vybrid edition license (ds.arm.com/vybrid/vybrid-edition/)
- Vybrid-tower-starter-kit license (ds.arm.com/vybrid/vybrid-tower-starter-kit/)

If you are using any of these licenses please be aware of the following limitations:

- The licenses are "node locked" and the tool cannot be accessed with the Windows Remote Desktop.
- Code size limitation (1 MB to Vybrid-edition, 256 KB for Vybrid-tower-starter-kit)
- Vectorization this feature is not supported in older DS5.14 with the special Vybrid license. Users have to turn off this feature manually (uncheck checkbox in IDE or use "--vectorize" option). This results in a less optimized code. The DS5.15 users are not affected.
- To use the command line build tools (with make), an additional compiler option is needed and the scatter file needs to be updated. Add the following options to the command line and update the scatter file by adding the same option to the end of the first line.
- Vybrid edition: "--tool_variant=vf6xx_tk"
- Vybrid-tower-starter-kit: "--tool_variant=vf6xx_sk"

3 Running and Debugging MQX RTOS application

The description bellow is provided for Vybrid microcontrollers BSPs - twrvf65gs10_a5 and twrvf65gs10_m4 and Hello World example application. The twrvf65gs10_a5 BSP runs on primary and twrvf65gs10_m4 runs auxiliary Vybrid core. The same procedure applies for all other BSPs and example applications distributed in the MQX RTOS release package.

3.1 Debugging Primary Core - MQX RTOS Hello World program

- Connect a serial cable to the TWR-SER or TWR-SER2 board DB9 connector. Set the communication speed to 115200.
- Select menu *File/Import/General/Existing Projects into Workspace* and import Hello World example application.

<mqx install dir>/mqx/examples/hello/build/ds5/hello twrvf65gs10 a5/.project

- Hit the compile button for to build application *Int. RAM Debug target*.
- Click the arrow next to the Debug button and select **Debug Configurations**.

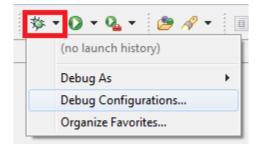
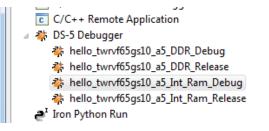


Figure 3- Debug configurations

 A dialog box will come up. Select the *hello_twrvf65gs10_a5_Int_Ram_Debug* configuration in the *Vybrid ARM Cortex[®]-A5 CMSIS-DAP* debug connection. Then hit the Debug button in the lower right corner.



4	Bare Metal Debug
	Debug Cortex-A5 via CMSIS-DAP
	Debug Cortex-A5 via DSTREAM/RVI
	Debug Cortex-M4 via CMSIS-DAP
	Debug Cortex-M4 via DSTREAM/RVI
\triangleright	Linux Application Debug
\triangleright	Linux Kernel and/or Device Driver Debug

Figure 4- Debug

• The Development Studio 5 (DS-5) will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.

Getting Started with Freescale MQX™ RTOS and ARM DS-5 Rev. 06, 08/2014

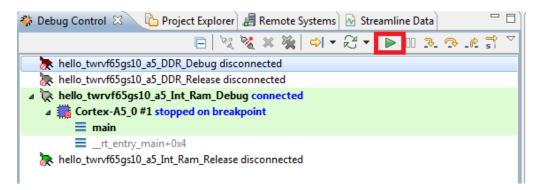


Figure 5- Debug control

- Press the **Run** button to continue in the Hello World program execution.
- The program prints Hello World on serial console terminal.



Figure 6- Hello World console

To debug the application again, push *Interrupt* button to stop program execution. Then
press *Disconnect from Target* and *Connect from Target* buttons in *Debug Control* Menu.

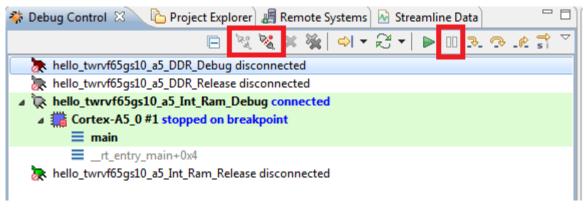


Figure 7- Debug again

Note: If subsequent connections to the target fail, it is recommended to reset the board by using the Reset button on TWR or by using PWR down/up sequence on the TWR elevator.

3.2 Run MQX RTOS Hello World program on auxiliary core on dual core system (Vybrid ARM Cortex[®]-M core)

 Before loading the application to the primary core as described in the previous chapter, it is important to check the Enable Cortex-M4 clock in the DTSL Options menu (accessible from the Debug Configuration dialogue).

lt X	Name of configuration: default	
	Enable Cortex-A5 core trace	-
	☐ Data Trace ✓ Addresses ✓ Values ☐ Enable data-only trace mode ☐ Trace capture range	
	Start address Ox0 End address OxFFFFFFF]
	Enable ITM trace	

Figure 8- DTSL configuration

- Run the primary core application and stop the program execution using the *Interrupt* button
- Switch to C/C++ Perspective using the button in right top corner
- Select menu *File/Import/General/Existing Projects into Workspace* and import Hello World example application for Cortex-M4 auxiliary core to your workspace:
 - <mqx_install_dir>/mqx/examples/hello/ds5/hello_twrvf65gs10_m4/.project
 - Hit the compile button and build application in the selected target. By default, the project is compiled in the *Int Ram Debug* target.

 - Highlight the *hello_twrvf65gs10_m4_Int_Ram_Debug* target.

🗱 Debug Control 🙁 🔥 Project Explorer 🔏 Remote System	IS 🛛	🐼 Streamline Data 🛛 🗖 🗖	
E 백 월 # 🎉 이 ·	₹ R	3 • ▶ 0 2 0 .0 \$ 7	
hello_twrvf65gs10_a5_DDR_Debug disconnected			
hello_twrvf65gs10_a5_DDR_Release disconnected			
A 🗮 hello_twrvf65gs10_a5_Int_Ram_Debug connected			
Cortex-A5_0 #1 stopped on breakpoint			
🗖 main			
rt_entry_main+0x4			
hello_twrvf65gs10_a5_Int_Ram_Release disconnected			
hello_twrvf65gs10_m4_Int_Ram_Debug disconnected	120	Constant Transf	
🔭 hello_twrvf65gs10_m4_Int_Ram_Release disconnected	12.0		
	2	Disconnect from Target	
Debug Configurations			
Launch in background Step Out to This Frame			
hello_twrvf65gs10_m4_Int_Ram_Debug_disconnected	\$	Reset DS-5 Views to 'Linked'	
mqx_main.c	×	Remove Connection	

Figure 9- Debug configurations

• Select **Debug Configurations** and verify that the **Enable Cortex-M4 clock** option in the **DTSL Options** menu is still checked.

Image: Section in the image: Sectio		ge, and run configurations choose a configuration to launch a DS-5 debug	jing session.
C/C++ Application C/C++ Attach to Application C/C++ Attach to Application C/C++ Remote Application Debug Contex-A5 via CMSIS-DAP Debug Contex-A5 via CMSIS-DAP Debug Contex-A5 via DSTREAM/RVI Debug Contex-A5 via DSTREAM/RVI Unix Application Debug Toon Python Run Toon Python Run Toon Python Run Toon Python Run Dista Configuration Editor Ju Unix Just Debug and Trace Services Layer (DTSL) Configuration Add Add edit or choose a DTSL configuration for file: dtsL configuration: default PyDet PyDet PyDet PyDet PyDet PyDet PyDet PyDet PyDet	🖹 🗎 🗶 🗌	□ ≱ ▼	Name: hwtimer_twrvf65gs10_m4_Int_Ram_Debug
C/C++ Attach to Application C/C++ Postmortem Debugger C/C++ Remote Application Select the manufacturer, board, project type and debug or C/C++ Remote Application Boby S5-Debugger thello_twnr65gs10_s5_DDR, Debug thello_twnr65gs10_s5_DR, Release thello_twnr65gs10_s5_DR, Release thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Debug thello_twnr65gs10_s5_Int_Ram_Release I/Ion Python unitest Java Java Jur Unit Debug and Trace Services Layer (DTSL) Configuration Add Add Pytho Imache Pytho Imache Pytho Imache Jur Unit Debug and Trace Services Layer (DTSL) Configuration for file: dtl_config.script.pv, class: DtlScript_CMSIS Imache Imache Pytho	type filter text		In Connection Files The Debugger 3
Ju JUnit Debug and Trace Services Layer (DTSL) Configuration Add, edit or choose a DTSL configuration for file : dtsl_config_script.py, class : DtslScript_CMSIS Lunit PyDe PyDe Image: Service Configuration for file : dtsl_config_script.py, class : DtslScript_CMSIS Image: Service Configuration for file : dtsl_config_script.py, class : DtslScript_CMSIS Image: Start address Image: Start address	E C/C++ E C/C++ E C/C++ 蒂 D5-5 D S → # * * * * * * * Iron P) Ø Java ₽	Attach to Application Postmortem Debugger Remote Application lebugger hello_twr/655g10_a5_DDR_Debug hello_twr/655g10_a5_Int_Ram_Debug hello_twr/655g10_a5_Int_Ram_Release hello_twr/655g10_a5_Int_Ram_Release hello_twr/655g10_a5_Int_Ram_Release thon Run thon unittest mate	Select the manufacturer, board, project type and debug ope Filter platforms Debug Cortex-A5 via CMSIS-DAP Debug Cortex-A4 via DSTREAM/RVI Debug Cortex-M4 via DSTREAM/RVI Linux Application Debug Linux Kernel and/or Device Driver Debug
Pyths Filer matche Filer matche Cycle Accurate Data Trace Values Enable data-only trace mode Trace capture range Start address Out	 ✓ Jytho ✓ Jytho ► Laun ✓ PyDe ▲ PyDe ✓ Pytho 	Add, edit or choose a DTSL configuration for	file : dtsl_config_script.py, class : DtslScript_CMSIS
Enable ITM trace Enable Cortex-M4 clock Enable Cortex-M4 core trace	Eilter matcher		Cycle Accurate Data Trace ✓ Addresses ✓ Values Enable data-only trace mode Trace capture range Start address 0x0 End address 0xFFFFFFF Enable ITM trace abble Cortex-M4 clock

Figure 10- Create, manage, and run configurations

- Confirm the selection and then hit the **Debug** button in the lower right corner.
- Project will be loaded to device and execution. CorexM4 core execution will stop in the main() function
- Press the run button and the program running on the auxiliary Cortex-M4 core will print "Hello World" on the serial console.



Figure 11- Hello World console

Note: It is always necessary to execute MQX RTOS application on primary core first. The MQX RTOS primary core application startup sequence contains settings required by Cortex-M4 core (clock setup etc).

3.3 Multi-core debugging

This chapter describes the basics of multi-core debugging with MQX RTOS. Description is provided for Vybrid microcontroller BSPs - twrvf65gs10_a5/twrvf65gs10_m4 and Multicore Communication (MCC) "pingpong" example application. The Cortex-A5 is primary core in this setup while the Cortex-M4 is set to auxiliary core.

- Select menu *File/Import/General/Existing Projects into Workspace* and import Cortex-A5 and Cortex-M4 MCC library projects as follows:
 - <mqx_install_dir>/mcc/build/ds5/mcc_twrvf65gs10_a5/.project
 - <mqx_install_dir>/mcc/build/ds5/mcc_twrvf65gs10_m4/.project
- Select the mcc_twrvf65gs10_a5 project in the Project Explorer View and then hit the compile button so build the **Debug target**.
- Select the mcc_twrvf65gs10_m4 project in the Project Explorer View and then hit the compile button so build the Debug target.
- Select menu *File/Import/General/Existing Projects into Workspace* and import MCC Pingpong example applications for both Cortex-A5 and Cortex-M4 core as follows:

<mqx_install_dir>/mcc/examples/pingpong/ds5/pingpong_example_twrvf65gs10_a5/.project
<mqx install dir>/mcc/examples/pingpong/ds5/pingpong example twrvf65gs10 m4/.project

- Select the *pingpong_example_twrvf65gs10_a5* project in the Project Explorer View and then hit the compile button for to build the *DDR Debug target*.
- Select the *pingpong_example_twrvf65gs10_m4* project in the Project Explorer View and then hit the compile button for to build the *Int Ram Debug target*.
- Click the arrow next to the Debug button and select **Debug Configurations**:

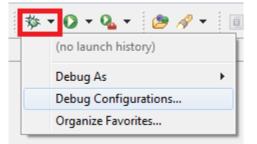


Figure 12- Arrow button

A dialog box will come up. Select the *pingpong_example_twrvf65gs10_a5_DDR_Debug* configuration and the *Vybrid Cortex-A5 CMSIS-DAP* debug connection. Before hitting the Debug button in the lower right corner, check the Enable Cortex-M4 clock in the DTSL Options menu.





DTSL Configuration Editor Debug and Trace Services Layer Add, edit or choose a DTSL configu	(DTSL) Configuration ration for file : dtsl_config_script.py, class : DtslScript_CMSIS	ß
default	Name of configuration: default Image: Start address Image: Start address Start address 0x0 Enable ITM trace Image: Start address	
	Enable Cortex-M4 clock Enable Cortex-M4 core trace	-
?	ОК Са	ancel

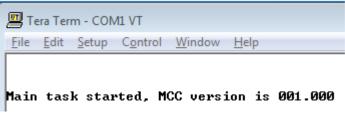
Figure 14- DTSL configuration

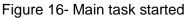
- The Development Studio 5 (DS-5) will switch to Debug Perspective automatically. Then, the project will be loaded to the device and execution will stop in the main() function.
- Press the **Run** button to continue in the *pingpong_example_twrvf65gs10_a5* program execution.

🏘 Debug Control 🛛 💫 Project Explorer 🚚 Remote Systems 🗟 Stream	line Data		ſ
	🔲 🗏 💐 💥 🎇 🔿 🗸 🎜 🕶 🕨 💷	7 (? (?)	
pingpong_example_twrvf65gs10_a5_DDR_Debug connected			1
Cortex-A5_0 #1 stopped on breakpoint			
🚍 main			
<pre>rt_entry_main+0x4</pre>			

Figure 15- Debug control

• The program will print "Main task started, MCC version is xxx.xxx" on the serial console terminal.





- Once the Cortex-A5 code application is running, start the execution of the auxiliary core (Cortex-M4).
- Select menu Run/Debug Configurations.
- When the Debug Configuration dialogue occurs, select the pingpong_example_twrvf65gs10_m4_Int_Ram_Debug configuration and the Vybrid Cortex-M4 CMSIS-DAP debug connection.
- Then hit the *Debug* button in the lower right corner.



Figure 17- Debug button

The Cortex-M4 project will be loaded to the device and execution will stop in the main () function. Press the *Run* button to continue in the *pingpong_example_twrvf65gs10_m4* program execution.

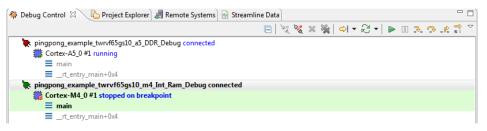


Figure 18- Run button

• The responder will be started and message "pingpong" between the cores will be initialized. See the console log.

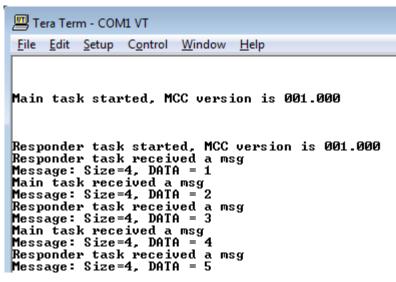


Figure 19- Pingpong message

3.4 Debugging the Application loaded by MQX RTOS Boot Loader

This chapter describes debugging the application which was loaded to the processor memory by MQX RTOS Boot Loader. The similar approach can be used for debugging an application loaded by a different boot loader e.g. U-Boot. This chapter also briefly describes steps required for preparing bootable SD Card image and application images in DS-5 tool set. For details about the Vybrid Boot Loader usage, see Readme.txt located in the MQX RTOS Boot Loader application folder (<mqx_install_dir>/mqx/examples/bootloader_vybrid/Readme.txt)

Building Boot Loader and creating bootable SD card

- First, import the MQX RTOS Boot Loader project to your workspace by using the File/Import/General/Existing Projects into Workspace menu.
- Select the bootloader_vybrid from your MQX RTOS installation directory:

<mqx_install_dir>/mqx/examples/bootloader_vybrid/ds5/bootloader_vybrid_twrvf65gs1 0_a5

- Select Int Ram Debug target and hit the compile button
- Use DS5 "C:\Program Files\DS-5\bin\fromelf" utility to create the binary image:

fromelf.exe --bin --output=bootloader_vybrid_twrvf65gs10_a5.bin
bootloader vybrid twrvf65gs10 a5.axf

• Follow <mqx_install_dir>\mqx\examples\bootloader_vybrid\Readme.txt description and use prepare binary image to prepare the bootable SD Card.

Building and Debugging the Application images

- Build the applications you want to run on A5 and M4 cores and convert them to binary format (.bin) by using *fromelf* DS-5 utility.
- Store the binary images on the root directory on bootable SD card.
- Copy setup.ini to the SD Card and modify according to Readme.txt description.
- Remove the SD Card from the PC and plug it into Micro SD Card slot on your Vybrid board.

Getting Started with Freescale MQX™ RTOS and ARM DS-5 Rev. 06, 08/2014

• Power up the Vybrid board. MQX RTOS Boot Loader will print out the following message on the default console (RS232 TWR-SER) and start execution of M4 and A5 applications.

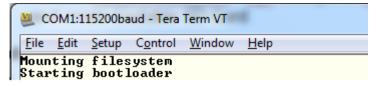


Figure 20- Mounting filesystem message

• To debug the running application, click the arrow next to the Debug button and select **Debug Configurations**.

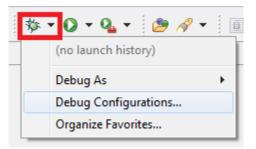


Figure 21- Debug configurations

- Then, select the application and target you want to debug and select Connect only.
- Finally, hit the **Debug** button in the lower right corner.

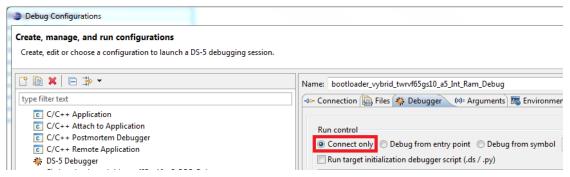


Figure 22- Create, manage, and run configurations

• The debugger will connect to the selected application. You can stop the selected core and debug the booted image.

Getting Started with Freescale MQX™ RTOS and ARM[®] Development Studio 5 (DS-5™), Rev. 06, 08/2014

4 MQX RTOS Task Aware Debugging in Development Studio 5 (DS-5) IDE

MQX RTOS Task Aware Debugging plug-in (TAD) is an optional extension to a debugger tool which helps to visualize internal MQX RTOS data structures, task-specific information, I/O device drivers, and other MQX RTOS context data.

The TAD plug-in is distributed separately from MQX RTOS release and directly from ARM[®] Ltd. For detailed documentation, contact your ARM distributor.

TCS MQX Window Help	MQX Window Help
RTCS Config	Kernel Data
Socket Summary	Check for Errors
Socket Config	Task Summary
TCP Config	Ready Queues
UDP Config	Stack Usage
IP Config	Memory Pools
NAT Config	Memory Blocks
ENET Config	Memory Extension Blocks
TCP Stats	Lightweight Memory Blocks
UDP Stats	Lightweight Memory Pools
IP Stats	Partition Summary
ICMP Stats	
IGMP Stats	Message Pools
ARP Stats	Message Queues
IPIF Stats	Lightweight Message Queues
NAT Stats	Lightweight Events
ENET Stats	Lightweight Semaphores
	Events
	Semaphores
	Mutexes
	Task Queues
	Initialization
	Interrupts
	IO Devices
	Lightweight Timer Queues
	Kernel Log
	Logs
	Names

Example of available DS-5 TAD menu:

Figure 23- Example menu