

A blue-tinted background image showing two people in a professional setting, one pointing at a computer screen displaying code or data.

Using the TOPPERS/ASP (RTOS) with the mbed/Arduino library.

on GR-PEACH (Cortex-A9)

With

Atollic TrueSTUDIO

INTRODUCTION

This document is an introduction on how to setup and get started with the “[TOPPERS/ASP kernel](#)” with the “[mbed](#)” and “[Arduino](#)” libraries on [GR-PEACH](#) with the Eclipse-based Atollic TrueSTUDIO on Windows. “[TOPPERS/ASP kernel](#)” is an RTOS developed in Japan (hereinafter referred to as ASP kernel).

The procedure of setting up a development environment necessary for building and debugging an application based on the ASP kernel has been complicated. The procedure requires the user to install the Cygwin environment, Cross-compiler and Debugger separately in order to build the ASP kernel on Windows. Using Atollic TrueSTUDIO, the user gets all the necessary tools out-of-the-box, that are required to build and debug an ASP kernel based application. Using Atollic TrueSTUDIO with the ASP kernel and mbed/Arduino libraries saves time and efforts previously necessary for environment construction.

Moreover, using the commercial version of Atollic TrueSTUDIO (Professional Edition), provides the user with ASP kernel awareness functions. These capabilities greatly simplify all RTOS debugging related efforts.

Introduction to GR-PEACH

GR-PEACH is an evaluation board from Renesas Electronics. It facilitates an ARM Cortex-A9 CPU called RZ/A1H.

Specification overview of Renesas RZ/A1H

- ARM Cortex-A9 (with NEON and Jazell) MAX 400MHz
- 32-Kbyte L1 I-Cache
- 32-Kbyte L1 D-Cache
- 128-Kbyte L2 Cache
- 10-Mbyte Large-capacity On-chip RAM
- More information about the CPU is available on the Renesas website:
<http://www.renesas.com/products/mpumcu/rz/rza/rza1h/index.jsp>

Specification overview of GR-PEACH

- 8MB FLASH
- 2x USB Host/Device Interface, 1xEthernet
- 3x SPI, 3x I2C, 8x UART, 7x 12-bits ADC, 2x CAN
- 2x Camera Input
- Arduino form-factor (Arduino UNO compatible)
- Built-in USB drag and drop FLASH programmer
- Please refer to the following mbed website for more information:
<https://developer.mbed.org/platforms/Renesas-GR-PEACH/>

GR-PEACH is a board provided by the *Gadget Renesas* project. The purpose of the *Gadget Renesas* is a project and a community which helps users to get started prototyping quickly and to turn ideas into reality. Read more about the *Gadget Renesas* community here:
<http://gadget.renesas.com/en/index.html>

Runtime Environment

You can download the software package from the following URL:

https://github.com/ncesnagoya/asp-gr_peach_gcc-mbed

This package includes the mbed and Arduino libraries that are integrated with the TOPPERS/ASP kernel. The runtime environment is illustrated as following.

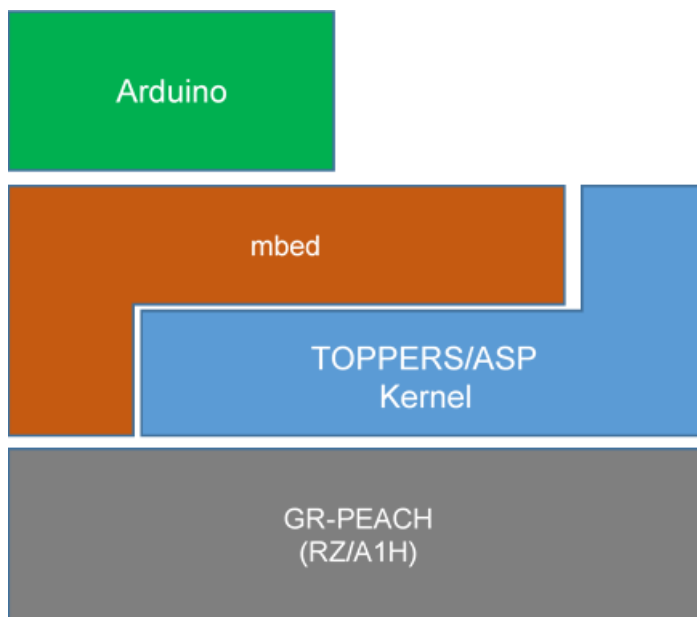


Figure 1 The runtime environment of ASP package

Boot method

Only SPI-NOR flash boot is available for executing an application on a GR-PEACH.

1. SPI-NOR flash boot Execution
 - Program is loaded into FLASH.
 - Flash operations are using the default USB MSD drag-and-drop without debugger.
 - When the device is reset, it will keep working until the application is stored in FLASH.
 - It is possible to debug the program that was loaded into the flash memory using the TrueSTUDIO debugger.

Preparations

The GR-PEACH comes delivered in cute pink and it is packed with computing power from a Cortex-A9 CPU.



Figure 2 The GR-PEACH box

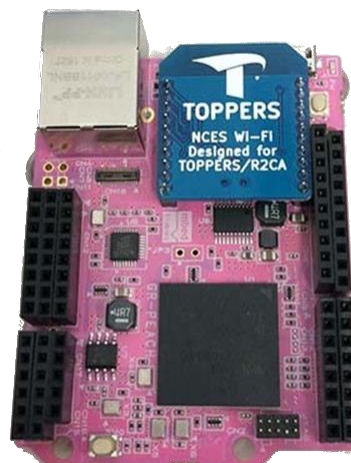


Figure 3 The GR-PEACH board

Preparing the hardware

- The [GR-PEACH](#)
 - The board can be purchased from Digi-Key. The price is ~120USD
- USB (mini-B) cable
 - Purchase a USB mini-B cable since it is not delivered with the board
- Windows PC
 - Atollic TrueSTUDIO runs on Windows Vista, 7, 8 and 10
 - Atollic TrueSTUDIO can use OpenOCD as GDB server to debug the GR-PEACH

Preparing the software

- Atollic TrueSTUDIO
 - This instruction was developed and tested using Atollic TrueSTUDIO v6.0.0 Lite. Problems may occur using older versions than v.6.0.0.
 - Please download the TrueSTUDIO installer from the following URL and install it: <http://atollic.com/resources/downloads/>
- TOPPERS/ASP kernel and mbed/Arduino library
 - Please download “asp-gr_peach_gcc-mbed-master.zip” from the following URL and extract it: https://github.com/ncesnagoya/asp-gr_peach_gcc-mbed
- OpenOCD
 - Download and install 0.10.0-201601101000-dev. Since the earlier versions does not work correctly
 - Windows executable binary of OpenOCD can be downloaded from the following URL: <https://github.com/gnuarmclipse/openocd/releases/tag/gae-0.10.0-20160110>
 - Copy the configuration file supplied within the asp-gr_peach_gcc-mbed-master.zip from `${ASP_DIR}/examples/truestudio/renesas_rza1h_swd.cfg` to `${OpenOCD_DIR}/scripts/target/`
- Terminal software
 - Install General-purpose terminal software such as puTTY: <http://www.putty.org/>

Connecting the board to the PC

- Connect the “mbed debug interface” USB-port of the board to a PC USB-port. The board should enumerate as a USB MSD (mass storage device).
- Install the ARM mbed Windows serial port driver, in case the device is not recognized as a serial port. You can get the ARM mbed Windows serial port driver from the following URL:
<https://developer.mbed.org/handbook/Windows-serial-configuration>

When the driver is installed and a USB cable is connected, the board is recognized as a serial port and USB mass storage device.

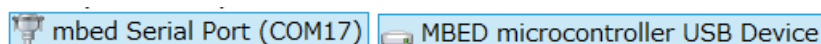


Figure 4 Serial port device + Mass storage device

The USB mass storage device is an interface to write the ROM memory of the board. More on this later.



Figure 5 Connect the GR-PEACH debugger using the USB-port in upper right corner

Building the example application

Start Atollic TrueSTUDIO

When Atollic TrueSTUDIO is started, the user will be prompted to specify a location for the workspace necessary for the tool to launch. The workspace is a container for projects. Please specify the path where the workspace should be located.

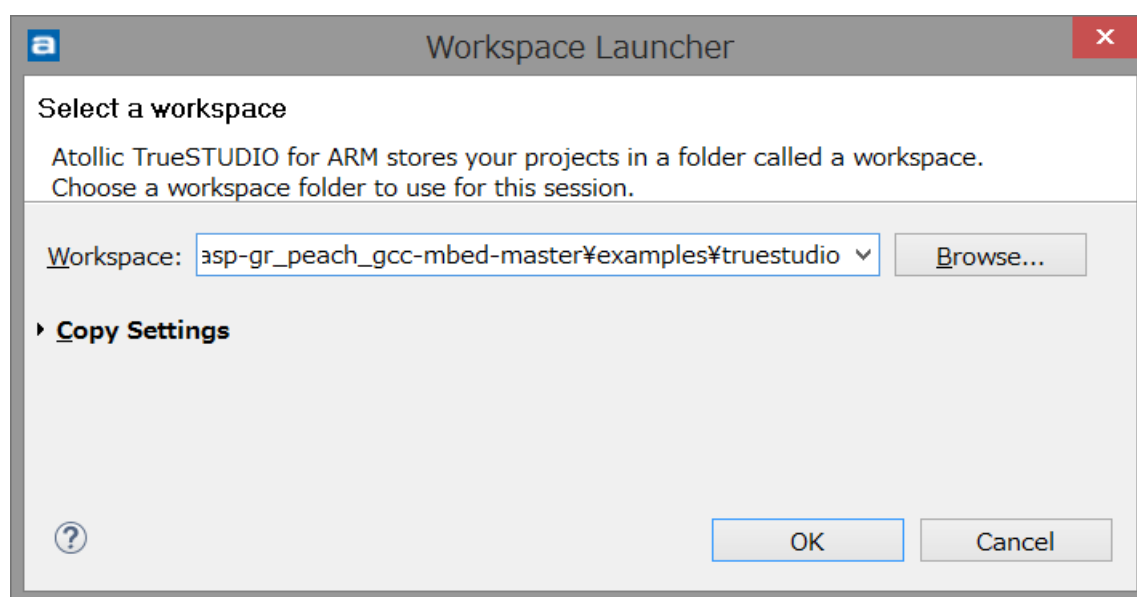


Figure 6 Workspace location

A recommendation is to select the workspace path to be one level above the extracted project:

`.asp-gr_peach_gcc-mbed-master\examples\truestudio`

Choosing this path as location for the workspace will make the project simpler to manage.

In the next step, you must import the C/C++ projects into the workspace.

The following directory under '.\truestudio' is a C/C++ project directory.

.\truestudio\blinky

.\truestudio\blinky_arduino

.\truestudio\httpsample

.\truestudio\multitask_arduino

.\truestudio\sample1

You must import above projects according to the procedure described below.

Select the 'File->Import...' and select the 'General->Existing Projects into Workspace'.

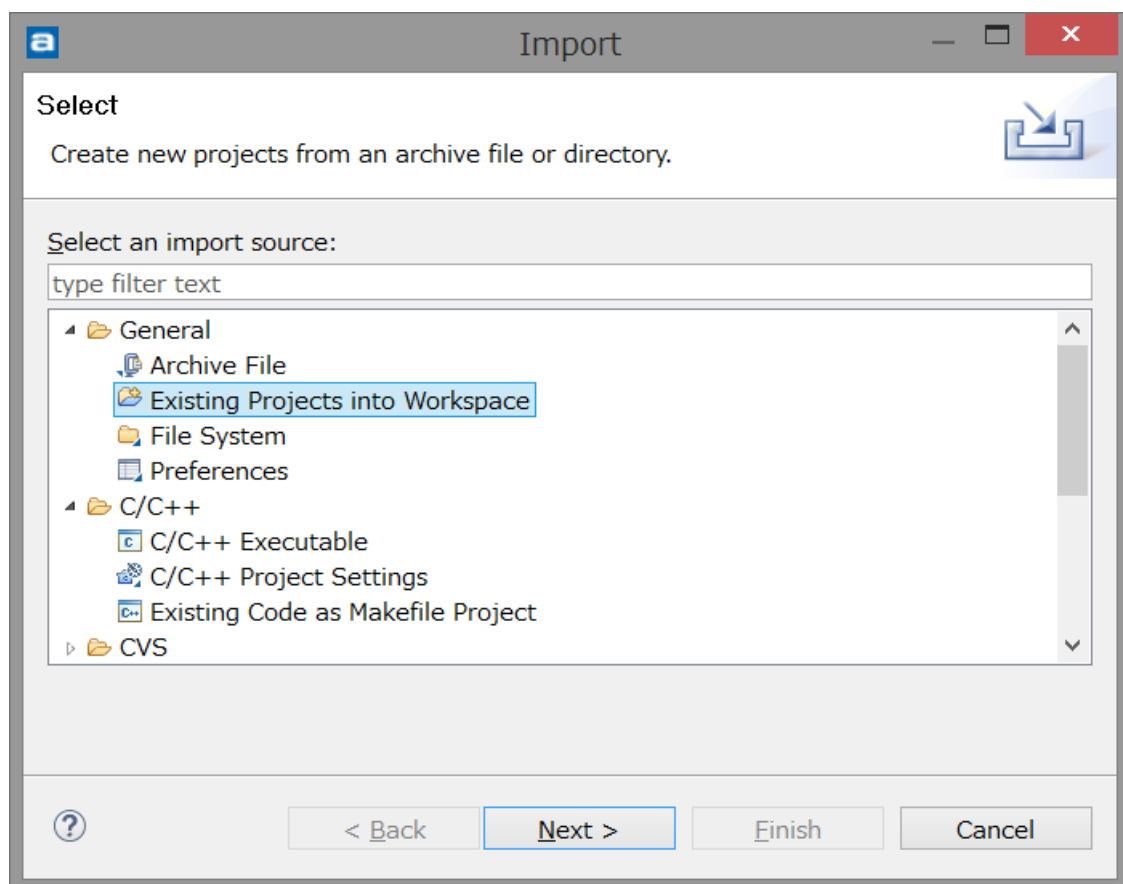


Figure 7 Import existing project

Select the C/C++ project directory under .\truestudio.

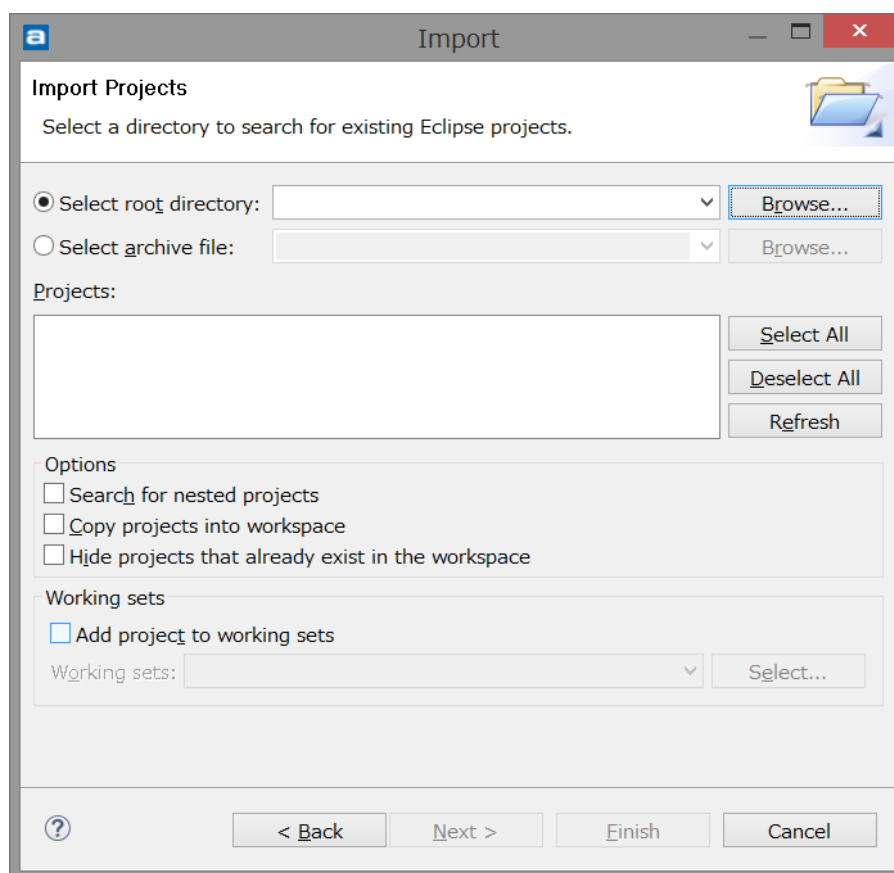


Figure 8 Import project dialogue

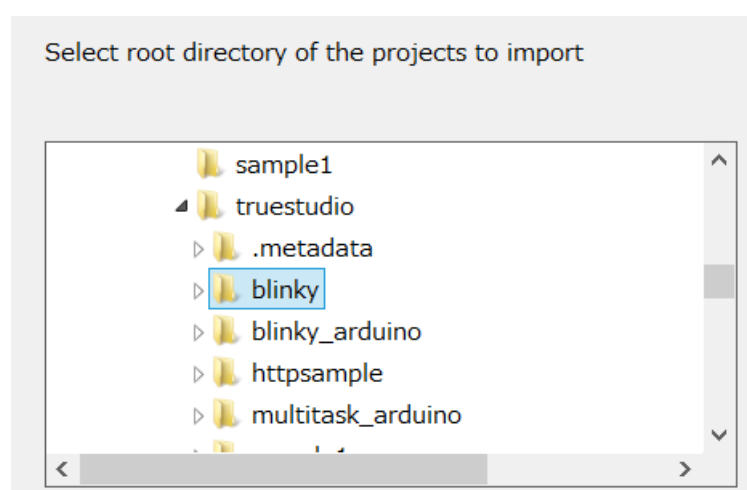


Figure 9 Choose project root directory

After importing the project, the project explorer will display as below.

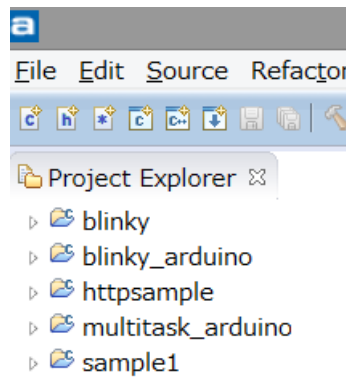


Figure 10 the workbench populated with the example project

Comments described in the source file of TOPPERS/ASP v1.9.2 are written in Japanese Kanji (UTF-8 encoding). We are currently preparing the English comments.

How to build the example project.

Below is a description on how to build the example application.

1. First select the *example* project from in the *Project explorer* view
2. Start a build process by clicking the build button

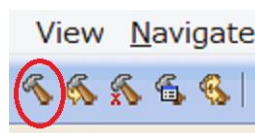
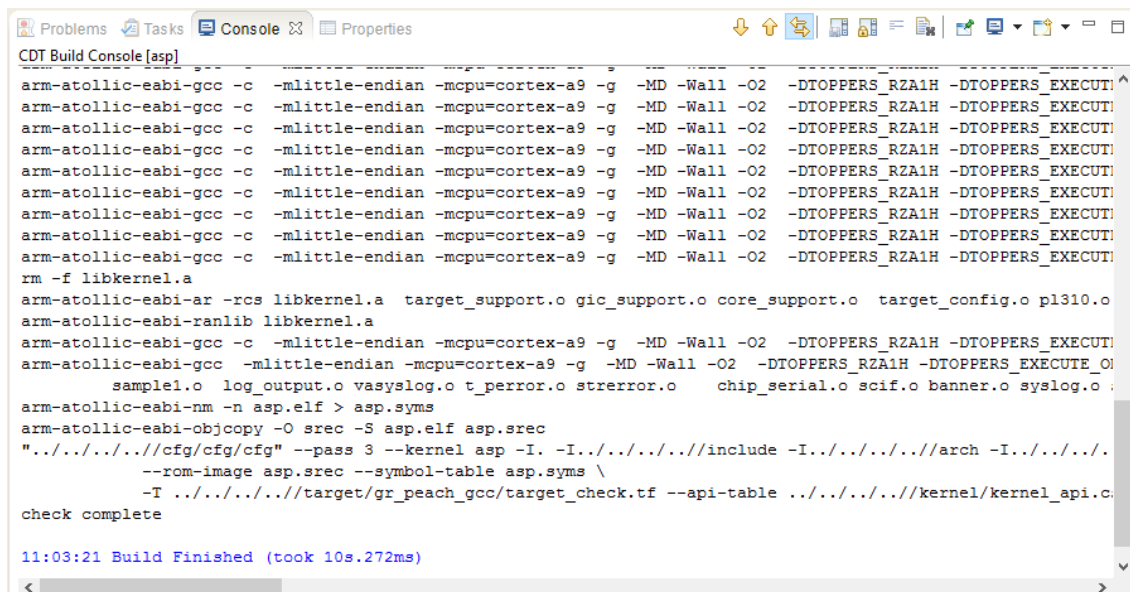


Figure 11 Buttons for: build, rebuild, clean, build settings, manage build configurations

When the build is successful, the executable application (asp.elf) and load module (asp.bin) are generated. If you already had built the project before, then use the *clean + build*, or the *rebuild* button in the toolbar above to build again.

The default build output format in Atollic TrueSTUDIO is *.elf*. The flash mechanism only supports *.bin* files. Therefore, this project has been configured to also convert elf → bin file to enable flash debugging.

After a successful build, the build console output should look similar to this:



```

CDT Build Console [asp]
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
rm -f libkernel.a
arm-atollic-eabi-ar -rcs libkernel.a target_support.o gic_support.o core_support.o target_config.o pl310.o
arm-atollic-eabi-ranlib libkernel.a
arm-atollic-eabi-gcc -c -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUT
arm-atollic-eabi-gcc -mlittle-endian -mcpu=cortex-a9 -g -MD -Wall -O2 -DTOPPERS_RZA1H -DTOPPERS_EXECUTE_O
sample1.o log_output.o vasyslog.o t_perror.o strerror.o chip_serial.o scif.o banner.o syslog.o :
arm-atollic-eabi-nm -n asp.elf > asp.syms
arm-atollic-eabi-objcopy -O srec -S asp.elf asp.srec
"../../../../cfg/cfg/cfg" --pass 3 --kernel asp -I. -I../../../../include -I../../../../arch -I../../../../
--rom-image asp.srec --symbol-table asp.syms \
-T ../../../../../../target/gr_peach/gcc/target_check.tf --api-table ../../../../../../kernel/kernel_api.c
check complete

11:03:21 Build Finished (took 10s.272ms)

```

Figure 12 Typical build output after successful build of libraries + executable

Debugging

In order to be able to debug the GR-PEACH, the user must first setup OpenOCD, then launch the debug configuration corresponding to either RAM or ROM execution.

Setup debugger - integrate OpenOCD

Atollic TrueSTUDIO needs to be configured to use the OpenOCD debugger.

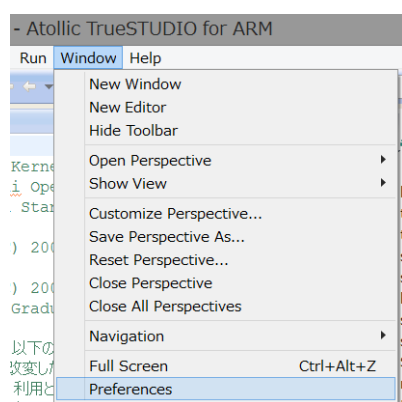


Figure 13 Help → Windows → Preferences

1. Help --> Windows --> Preferences --> Run/Debug --> Embedded C/C++ Application --> Debug Hardware --> OpenOCD
2. Setup the path to the server executable in the “Server” field
3. Setup the path to the server working directory (same directory as above)
4. Wait (s) before connecting to server: 2
5. Click OK

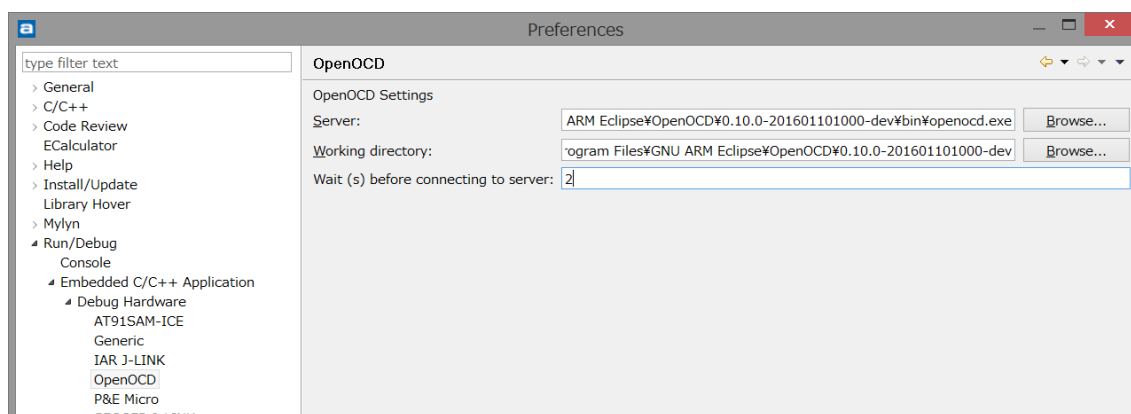


Figure 14 Configuration of path to OpenOCD GDB-server

How to load and execute the example programs.

Make sure that the GR-PEACH is connected to the Windows PC with a USB mini-B cable.

1. After connecting the board, the user must copy the “*asp.bin*” from the *Project explorer* view to the MBED drive which enumerates as a USB mass storage device in *Windows file explorer*.

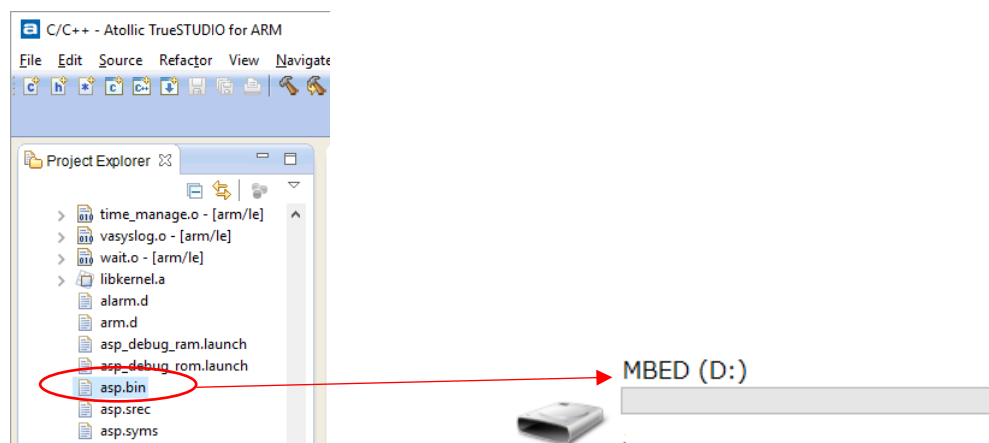


Figure 1 Drag-and-drop the bin file to the MBED USB mass storage device

2. The program is written to the Flash ROM of the board. Wait until flash operation is completed.
3. Push the reset button on the GR-PEACH board, the program is started and execution logs are displayed on the serial terminal.

How to debug the loaded example programs.

1. Select the *example* project that has been loaded into board flash in the *Project explorer* view .
2. Toolbar menu: *Run* → *Debug Configurations...*
3. Select debug configuration which corresponds to the loaded example.

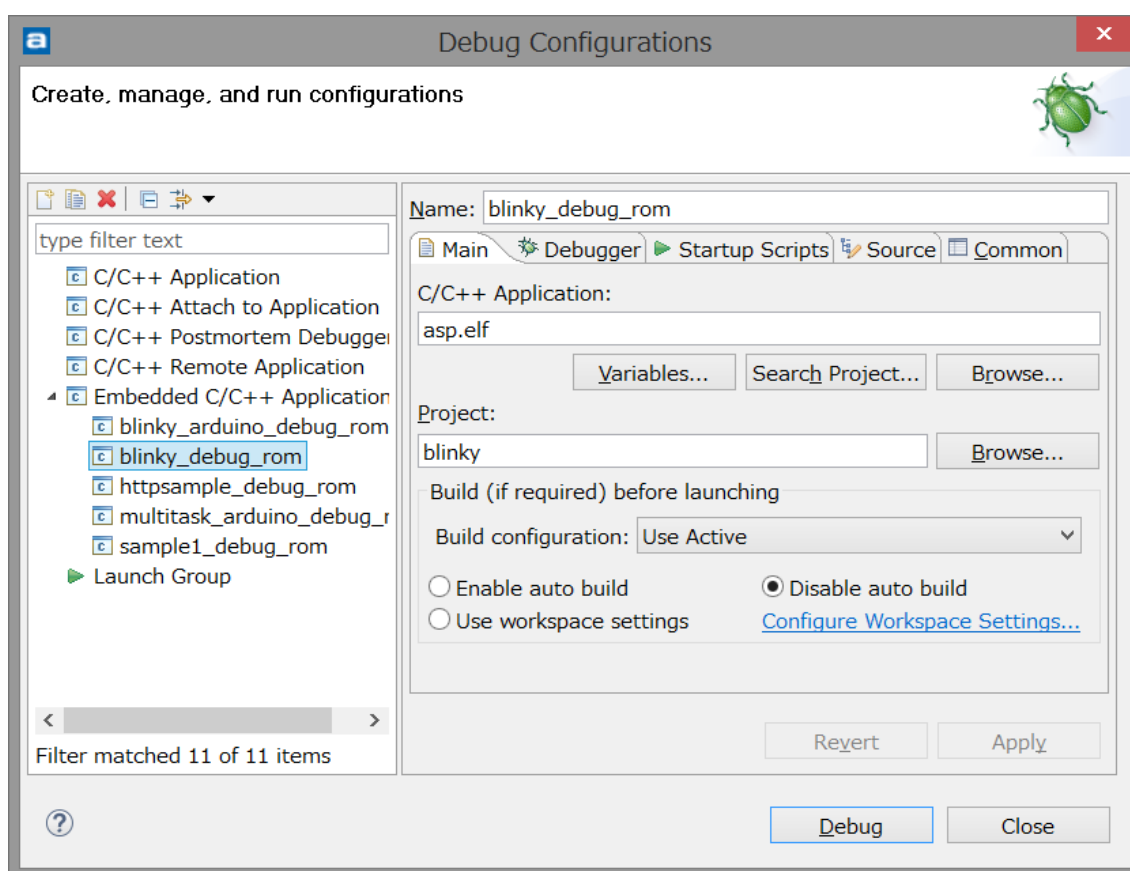


Figure 16 Debug configurations for example projects.

4. Start the debugger with the selected debug configuration by clicking *Debug*

The debugger is now started. The IDE automatically switch from *C/C++ Editing* perspective to *Debug* perspective. This is seen in the picture below.

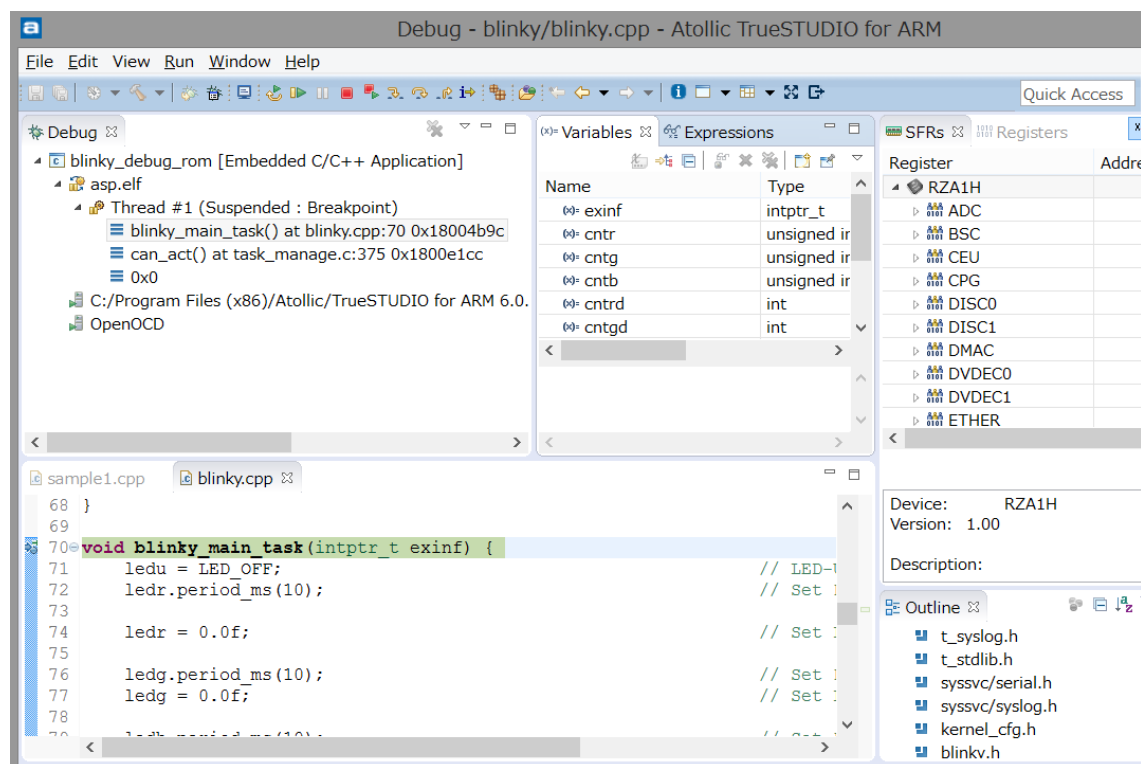


Figure 17 Target is waiting for "Resume" button to be pressed

The debugger automatically stops at the temporary breakpoint configured in the startup script. The user can click Execution Resume from this break point.

Serial terminal communication

The example application uses a USB virtual serial port to handle I/O (input/output) communication to the debugger. Connect using your favorite serial terminal application (i.e. PuTTY). Use the following serial communication settings:

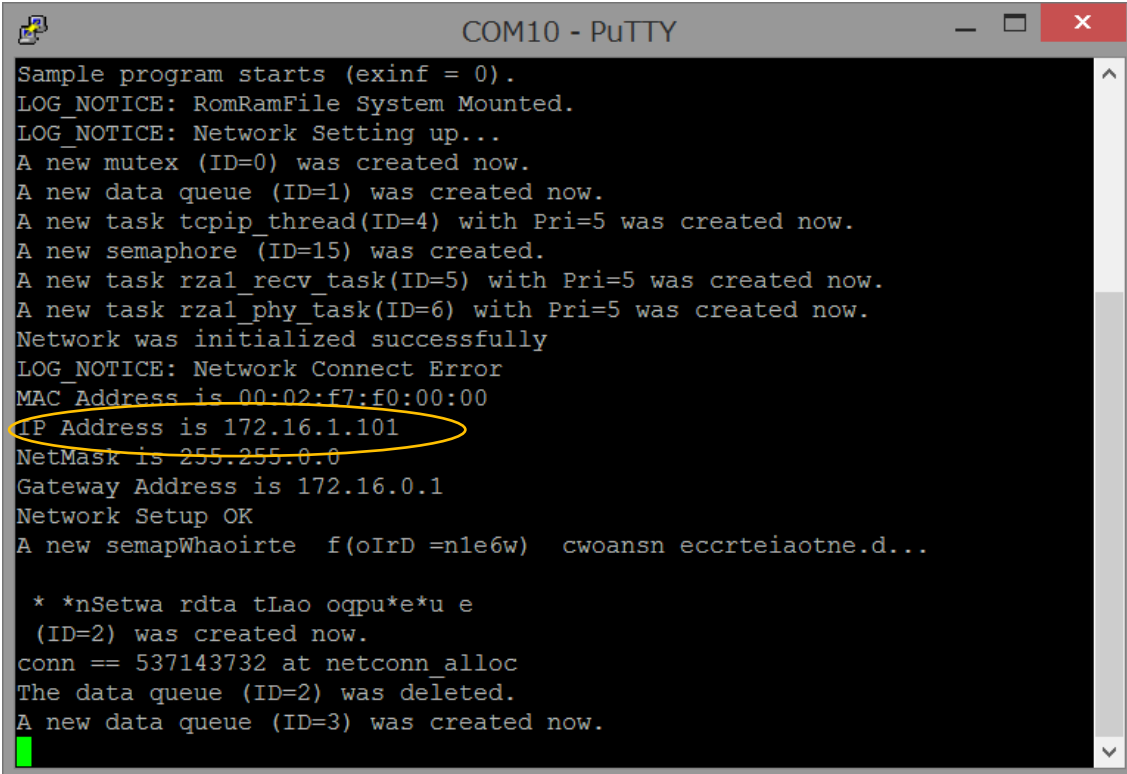
- connection type: "Serial"
- Baud Rate: 115200bps
- Data Bits: 8bit
- Parity: None
- Stop Bits: 1
- Flow Control: None

Running the HTTP server example

Introduce the HTTP server example in the sequence below.

After building the 'httpsamle' project and load asp.bin into on-board flash, connect LAN cable into GR-PEACH and push the reset button.

Run-time information from the application is output to the serial terminal log. See screenshot below.



```

Sample program starts (exinf = 0).
LOG_NOTICE: RomRamFile System Mounted.
LOG_NOTICE: Network Setting up...
A new mutex (ID=0) was created now.
A new data queue (ID=1) was created now.
A new task tcpip_thread(ID=4) with Pri=5 was created now.
A new semaphore (ID=15) was created.
A new task rza1_recv_task(ID=5) with Pri=5 was created now.
A new task rza1_phy_task(ID=6) with Pri=5 was created now.
Network was initialized successfully
LOG_NOTICE: Network Connect Error
MAC Address is 00:02:f7:f0:00:00
IP Address is 172.16.1.101
NetMask is 255.255.0.0
Gateway Address is 172.16.0.1
Network Setup OK
A new semapWhaoirte f(oIrD =nle6w) cwoansn eccrteiaotne.d...

* *nSetwa rdta tLao ogpu*e*u e
(ID=2) was created now.
conn == 537143732 at netconn_alloc
The data queue (ID=2) was deleted.
A new data queue (ID=3) was created now.

```

Figure 18 General purpose serial terminal can be used to monitor the application message.

The HTTP server example gets its own IP address from the DHCP server and displays it on the serial terminal. You can browse the homepage in GR-PEACH by using this IP address on your WEB browser.

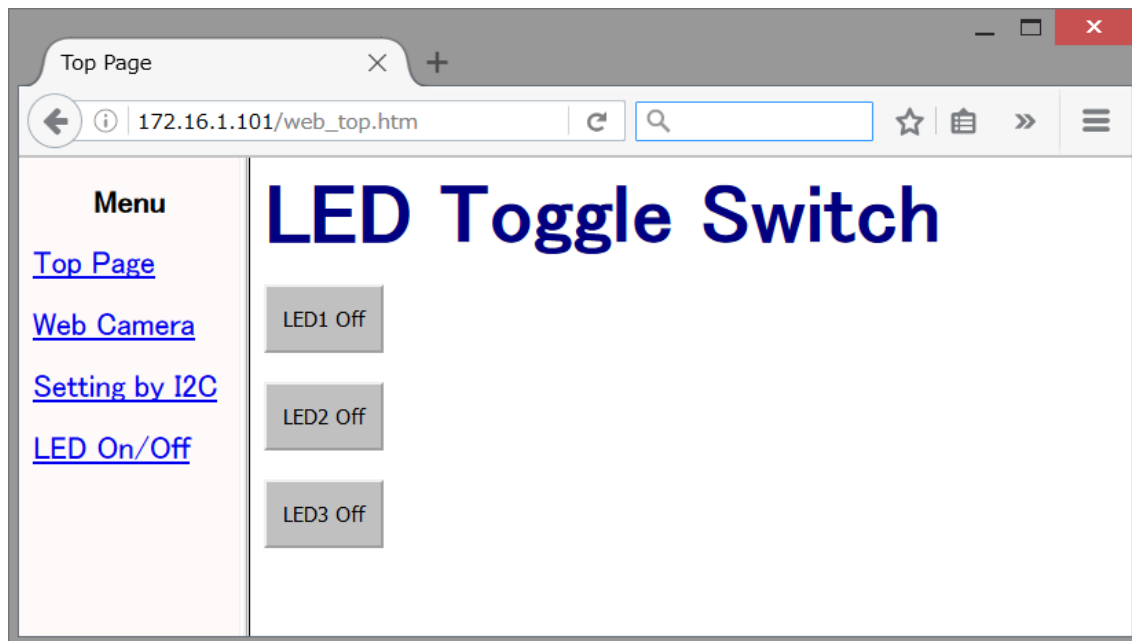


Figure 19 CGI controls on HTTP Server.

Also, you can control the LED On/Off by using the home page button.

RTOS debugging

Using Atollic TrueSTUDIO Pro (commercial license), the user is provided with RTOS views for TOPPERS/ASP.

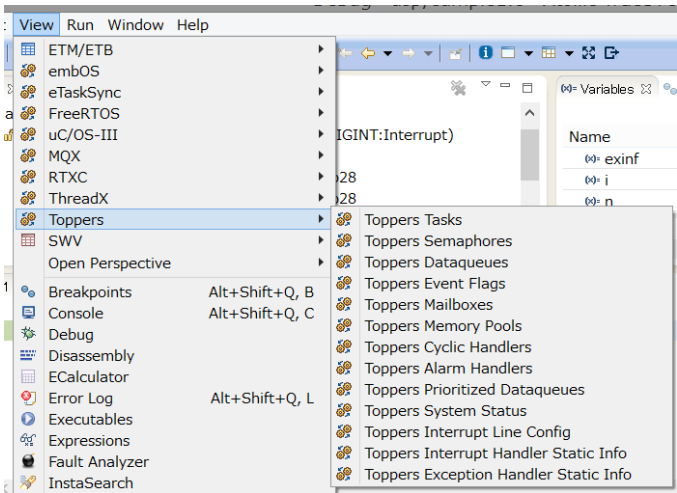


Figure 2 Available Kernel aware debugging view for TOPPERS / ASP

The debugger provides the user with TOPPERS/ASP kernel object information for tasks, semaphores, etc. - that the user has defined. Providing this information in Toppers views improves the efficiency of RTOS debugging.

Below is a screenshot that displays static information about the parameters used when creating different tasks.

Static information								
* Id	Auto start	Initial Prio	Entry	Tex routine	Entry Arg	Stack Area	Stack Size	
1	Yes	3	logtask_main()	0x0	0x000002	0x20001898	1024	
2	No	10	task()	tex_routine()	0x000001	0x20000c98	1024	
3	No	10	task()	tex_routine()	0x000002	0x20000898	1024	
4	No	10	task()	tex_routine()	0x000003	0x20001498	1024	
5	Yes	5	main_task()	0x0	0x000000	0x20001098	1024	

Figure 3 Static information about TOPPERS tasks

The next screenshot display a view displaying the “current status” of all tasks.

Static information Current status										
* Id	Current Prio	Status	Waiting object	Remaining time	Pend...	Pend ...	Enable Tex	Tex Pattern	Sp	Remaining Stack
1	3	Waiting	Delay	11 ms			Disable	0x000000	0x20001c18	896
→ 2	10	Running					Enable	0x000000	0x20000ff4	860
3	10	Ready					Disable	0x000000	0x20000c98	1024
4	10	Ready					Disable	0x000000	0x20001898	1024
5	5	Waiting	Semaphore	Forever			Disable	0x000000	0x200013d8	832

Figure 4 Dynamic information about TOPPERS task. Updated on debugger suspend.

Final words

Atollic TrueSTUDIO and GR-PEACH enables you to get a free of charge high quality development environment for Cortex-A9. Purchasing the optional Atollic TrueSTUDIO Pro license will allow for kernel aware debugging of the TOPPERS / ASP RTOS, significantly simplifying development and debugging of TOPPERS based applications. Try for yourself and let us know what you think!

About the TOPPERS project



The TOPPERS (Toyohashi Open Platform for Embedded Real-time Systems) Project develops a variety of software which is the basis of embedded system construction starting from the Technology development achievements of the ITRON specification and releases them as high quality open source software.

In this document, TOPPERS / ASP is configured and used with a single-core microcontroller. The TOPPERS project release various open source RTOS and middleware packages for different types of CPU cores and configurations requirements:

- Multi-cores (FMP)
- Protection enhancements RTOS (HRP2)
- RTOS for automotive (ATK2)
- DualOS monitor (SafeG)
- Etc.

More information on TOPPERS

If you are interested in the TOPPERS Project, please refer to the TOPPERS Project website (English):

<https://www.toppers.jp/en/index.html>

Document:

TOPPERS with mbed/Arduino on Renesas GR-PEACH

Website:

<http://www.aicp.co.jp/en/index.html>

E-mail:

atollic@aicp.co.jp

Related website information:

The uITRON4.0 specification (English) is available from the following URL:

<http://www.ertl.jp/ITRON/SPEC/FILE/mitron-400e.pdf>

This document is a TOPPERS/ASP Commentary blog (English), which is a part of the Education Program (NEP) operated by Nagoya University Center for Embedded Computing Systems (NCES):

http://www.nces.is.nagoya-u.ac.jp/NEXCESS/blog_en/

Author : A.I. Corporation (AIC)

AIC is the Distributor in Japan of Atollic:

<http://www.aicp.co.jp/en/corporate/index.shtml>

AIC has also been a member of the TOPPERS Project and are involved in the promotional activities since it was established.

When Atollic implemented the TOPPERS RTOS kernel aware debugger view function in Atollic TrueSTUDIO, AIC was involved as an advisor.