Target setup for RZA1H

This document describes the setup required to build and run CGI Studio applications with RZA1H device package on Renesas RZA1H hardware.

It is explained how to run the RZA1H target demo application part of the release package, how to setup the build environment and how to build own applications for RZA1H target (Renesas RZA1H).

- RZA1H BSP
  - Prerequisites
  - Running the CGI CourierSampleAppLight Demo Application on RZA1H
  - Building Own Application for the RZA1H
- Various Device Specific Content
  - RZA1HBitmapConverter

RZA1H BSP

Prerequisites

Hardware: Renesas RZA1H

CGI Studio has been tested on Renesas RZA1H custom board.

CMake

In order to build Candera, the Meta build system CMake needs to be installed with at least version mentioned here.

CMake is able to create the cross compilation make files that can be used with various compilers.

Ninja

To allow cross compilation with makefiles, MinGW 4.2.1 (Ninja 1.9.0) was used for this release.

GCC Toolchain

The build tool chain is not part of the delivery. gcc-arm-none-eabi-9-2020-q2-update-win32.exe and the according Renesas BSP and RGL driver have to be obtained from customer and installed according to their installation guide.

CGI Studio was tested with gcc-arm-none-eabi-9-2020-q2-update-win32.exe

Debugging was done over J-LINK 10.1 in Renesas e² studio.

Renesas RGA Library

The graphics driver is not part of this release. CGI Studio was integrated and tested with the libraries received from customer

Running the CGI CourierSampleAppLight Demo Application on RZA1H

Download the Application

To download and debug the application you need Renesas e^2 Studio, a sample project and a J-Link probe.

Include "cgi_studio_courier_apps\src\CourierSampleAppLight\AppPlatform\Target_FreeRTOS_ARM\SampleAppConnector.h" in the main.c og the sample application. Declare extern function SwapRequiredBuffers "extern void SwapRequiredBuffers();"
Initialize the App with SampleAppInit();
In a loop call SampleAppRun(); and SwapRequiredBuffers();

Start SEGGER J-Flash lite and download the asset file to the address defined in linker_settings.ld as __graphics_data_start.

Compile the project and use Renesas e^2 Studio to download and debug the binary to the target.

Building Own Application for the RZA1H

Generate Makefiles with CMake

For general information about CGI Studio Build System and CMake properties available to configure Candera applications refer to CGI Studio Build System.

To configure and generate makefiles for compiling the CGI CourierSampleAppLight, follow below steps:

1. Point CMake to source directory
   - Folder containing top-level "CMakeLists.txt", e.g.

     ```
     /cmake/Apps/Courier/CourierSampleAppLight
     ```

2. Point CMake to build directory
   - May not exist, create folder with CMake if necessary
   - Root directory for generated makefiles / solutions
   - Root directory for build artifacts
3. Configure
   - Specify MinGW Makefiles (Ninja) as generator for the project
   - Specify tool-chain file for cross-compiling
4. Select the toolchain-file from the delivery (adapt for your toolchain installation and application specific needs)

   ```
   /cgi_studio_devices/src/RZA1H/ToolchainFiles/GCC-toolchain-RZA1H_Native2D.txt
   ```

5. Adapt the CMake settings if necessary, e.g.:
   - CMAKE_BUILD_TYPE-> Debug or Release
   - Press Configure
6. Press Generate

Building Application

After generating your cmake files, in Command Prompt simply navigate to the specified build directory and type "mingw32-make" ("ninja") to start the build.

The output executable file can be found in your build directory.

Open this file with the Green Hills MULTI IDE debugger and load it to RAM as described in the previous chapter.

Various Device Specific Content

RZA1HBitmapConverter

Description

This document provides a complete list of bitmap formats supported by the RZA1H Target.

These formats define how pixel color and alpha data are stored in memory. The RZA1H extended bitmap formats support **alpha-only**, **RGB**, **ARGB**, and **JPEG compressed images** formats to address different memory, performance, and visual quality requirements.

These bitmap converters are supported only on the target. Windows simulation uses the default bitmap converter.

Alpha Only Formats

Alpha formats (A1/A4/A8) are primarily used for masking and blending operations.

Serial No.	Format	Description
1	RZA1HExtendedBitmapFormatA1	1-bit alpha mask.
2	RZA1HExtendedBitmapFormatA4	4-bit alpha channel.
3	RZA1HExtendedBitmapFormatA8	8-bit alpha channel.

RGB/ARGB Formats

RGB/ARGB formats balance memory usage and visual quality depending on color depth and transparency needs.

Serial No.	Format	Description
1	RZA1HExtendedBitmapFormatRGB565	16-bit RGB (5-6-5), no alpha.
2	RZA1HExtendedBitmapFormatARGB8888	32-bit ARGB, 8 bits per channel.
3	RZA1HExtendedBitmapFormatARGB1555	16-bit ARGB with 1-bit alpha.
4	RZA1HExtendedBitmapFormatARGB4444	16-bit ARGB, 4 bits per channel.

5	RZA1HExtendedBitmapFormatXRGB8888	32-bit RGB, unused alpha byte.
6	RZA1HExtendedBitmapFormatR8G8B8A8	32-bit RGBA, 8 bits per channel.

JPEG Compressed Image Format

JPEG format enables efficient storage of large images with reduced memory requirements.

Serial No.	Format	Description
1	RZA1HExtendedBitmapFormatJpg	JPEG compressed image format.