RZ/A2M Group

OctaBus Driver for Accessing OctaRAM

Introduction

This application note describes the operation of the OctaBus Driver for Accessing OctaRAM.

It provides a comprehensive overview of the driver. For further details please refer to the software driver itself.

The user is assumed to have knowledge of e2 studio and to be equipped with GR-MANGO Board.

Target Device

RZ/A2M Group

Driver Dependencies

This driver has no other driver dependencies.

Contents

1. Outline of Software Driver

The OctaBus Driver for Accessing OctaRAM provides an initialization procedure for accessing OctaRAM via memory-mapped area.

Note: This driver must not use in loader project.

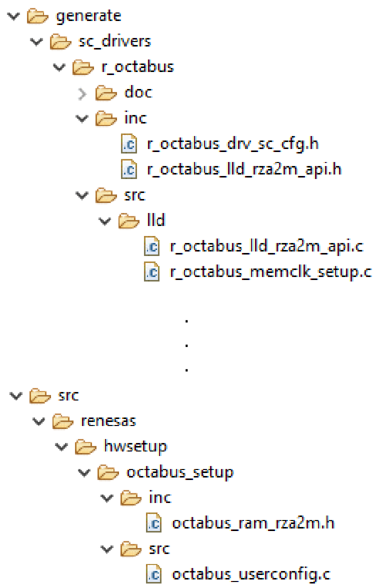
2. Description of the Software Driver

The key features of the driver include:

— Initialise of controller
— Initialise of memory device (Provides command)

2.1 Structure

Unlike many of the other drivers, the OctaBus driver currently consists of a single layer: Low Layer Driver (LLD). This includes all the hardware specific functions and provides the API to the application.



2.2 Description of each file

Each file's description can be seen in the following table.

Table 2.1 Description of each file

Filename	Usage	Description
Low Layer API		
r_octabus_lld_xxxx.h	API header file	Low Layer Driver (LLD) header file (where "xxxx" is a device and board-specific identification). This is the header file to include in application code.
Low Layer Driver (LLD) Source		
r_octabus_lld_xxxx.c	Private (LLD only)	(Where "xxxx" is a device and board specific identification). Provides the function definitions for the Low Layer Driver interface.
r_octabus_memclk_setup.c	Private (LLD only)	Provide the function selects the OctaBus clock, invoked by start-up procedure.
Middleware Source		
octabus_userconfig.c	User modifiable	Provide the default function initializes the controller and the device.

2.3 Low Layer Driver

The Low Layer Driver provides the functions to configure the hardware.

Table 2.2 Low Layer Driver Functions

Return Type	Function	Description	Arguments	Return
void	R_OCTABUS_Setup(void)	Perform a set of procedure initialise the controller and the OctaRAM device. Invoked by start-up procedure.	None	None
int_t	R_OCTABUS_Init (const st_octabus_cfg_t * p_cfg)	Initialise the controller for accessing OctaRAM device	p_cfg: [in] configuration data described the parameters	DRV_SUCCESS or DRV_ERROR
int_t	R_OCTABUS_ManualCalib (const st_octabus_cfg_t * p_cfg)	Calibrate data sampling points to improve reading accuracy	p_cfg: [in] configuration data described the parameters	DRV_SUCCESS or DRV_ERROR
int_t	R_OCTABUS_AutoCalib (const st_octabus_cfg_t * p_cfg)	Start automatic calibration capability to keep reading accuracy	p_cfg: [in] configuration data described the parameters	DRV_SUCCESS or DRV_ERROR
uint32_t	R_OCTABUS_ReadConfigMode (st_octabus_configmode_t * p_config)	Read configuration register from OctaRAM	p_config: [in] configuration data described the command	Value read from OctaRAM
void	R_OCTABUS_WriteConfigMode (st_octabus_configmode_t * p_config, uint32_t write_value)	Write the specified value to the configuration register of OctaRAM	p_config: [in] configuration data described the command write_value: [in] value to write to configuration register	None
int_t	R_OCTABUS_GetVersion (st_ver_info_t *p_ver_info)	Get Low Layer Driver version information	p_ver_info: [out] pointer to version information structure	DRV_SUCCESS
int_t	OctaBus_UserConfig (const st_octabus_cfg_t * p_cfg)	User modifiable device startup routine	p_cfg: [in] configuration data described the parameters	DRV_SUCCESS or DRV_ERROR

3. How to Import the Driver

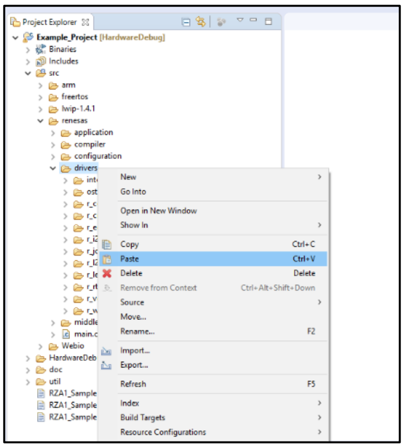
This section describes how to import the driver into your project. Generally, there are two steps in any IDE:

1) Copy the software driver to the location in the source tree that you require for your project.
2) Add the include path of the driver to the compiler.

3.1 e2 studio

To import the driver into your project please follow the instructions below.

1) In Windows Explorer, right-click on the r_octabus and the octabus_setup folder, then click Copy.
2) In e2 studio Project Explorer view, select the folder where you wish the driver project to be located; right-click and click Paste.
3) Right-click on the parent project folder (in this case 'Example_Project') and click Properties ...
4) In 'C/C++ Build → Settings → Cross ARM Compiler → Includes', add the include folder of the newly added driver, e.g.
   '${ProjDirPath}\src\renesas\drivers\r_cache\inc'
   '${ProjDirPath}\src\renesas\drivers\octabus_setup\inc'



4. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware

The latest version can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
Rev.1.00	Sep.24.20	–	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.