RH850/U2A-EVA

Using CRG, PBG and ECM

Introduction

The RH850/U2A-EVA series offers guard functions. These are implemented for functional safety to protect certain hardware macros from unauthorized access. This document provides a general introduction of the guard functions, focusing on Peripheral Bus Guard (PBG) and Cluster RAM Guard (CRG) in combination with the Error Control Module (ECM). It also contains guard configuration flows and the sample software.

Target Device

This Document is intended to describe the PEG, PBG and ECM sample software on RH850/U2A-EVA series.

In this document the RH850/U2A16 device R7F702300BFABA-C is employed to implement the example application. However, the concepts described in this document can be applied to all other members of the RH850/U2A-EVA series.

The RH850/U2A-EVA series has following variants:

RH850/U2A-EVA	FBGA-516	R7F702Z19AEDBG
		R7F702Z19BFDBG
RH850/U2A16	FBGA-516	R7F702300EBBG-C
		R7F702300AEBBC-C
		R7F702300BEBBC-C
	FBGA-373	R7F702300EBBB-C
		R7F702300AEBBB-C
		R7F702300BEBBB-C
	FBGA-292	R7F702300EABA-C
		R7F702300AFABA-C
		R7F702300BFABA-C
RH850/U2A8	FBGA-373	R7F702301EBBA-C
		R7F702301AEBBA-C
		R7F702301BEBBA-C
	FBGA-292	R7F702301EABG-C
		R7F702301AFABG-C
		R7F702301BFABG-C
RH850/U2A6	FBGA-292	R7F702302FABB-C
	HLQFP-176	R7F702302FAFK-C
	FBGA-156	R7F702302FABD-C
	HLQFP-144	R7F702302FAFM-C

Disclaimer

Renesas Electronics does not warrant the information included in this document. You are fully responsible for incorporation of these circuits, software, and information in the design of your equipment and system. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

Contents

Introduction 1
Target Device 1
Disclaimer 1
Contents 2
Table of Tables 4
Table of Figures 5
1. Background 6
2. Reference Documents 7
2.1 RH850/U2A-EVA Group User's Manual: Hardware 7
2.2 RH850/U2A16 Starter Kit 7
3. Definitions and Abbreviations 8
4. Software and Hardware Tools 9
4.1 Development Tools 9
4.2 Emulator E2 9
4.3 RH850/U2A-EVA Evaluation Boards 9
4.3.1 Starter Kit Board 9
5. General information on ECM, CRG and PBG 10
5.1 Memory Protection 10
5.2 CRG, PBG and ECM 12
5.2.1 SPID 12
5.2.2 CRG – Cluster RAM Guard 13
5.2.3 PBG – Peripheral Bus Guard 13
5.2.4 ECM and Error notifications 13
6. SW procedure and module setup for CRG and PBG applications 16
6.1 Cluster RAM Guard 16
6.1.1 CRG Setup 16
6.1.2 Detecting and Clearing errors 17
6.1.3 Inter-VM Communication (IVC) 19
6.1.4 Register description 22
6.2 Peripheral Bus Guard 24
6.2.1 PBG setup 24
6.2.2 Error identification & clearing 24
6.2.3 Register description 25

7. SW procedure and module setup for ECM applications 27
7.1 ECM Error Output reset and Error configuration 27
7.2 Error identification & clearing 29
7.3 Register description 29
7.4 Pin Assignment 31

8. Description of Sample Software 32
8.1 Application description and appearance 32
8.1.1 User LEDs 32
8.1.2 LED Ring 32
8.1.3 Test Selection 32
8.1.4 Test Description 33
8.2 Port and clock setup for TAUD, MSPI and INTP 33
8.2.1 Port Setup - External Input / Output 33
8.2.2 Port Write Protection 34
8.2.3 Clock Setup for TAUD and MSPI Module 34
8.3 Sample Software 35
8.3.1 main_pe0.c 35
8.3.2 main_pe1.c 35
8.3.3 r_mspi.c and r_mspi.h 35
8.3.4 r_led.c and r_led.h 36
8.3.5 r_tau.c and r_tau.h 36
8.3.6 r_intp.c and r_intp.h 36
8.3.7 r_pbg.c and r_pbg.h 37
8.3.8 r_crg.c and r_crg.h 37
8.3.9 r_ecm.c and r_ecm.h 37

Revision History 38

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products 1

Notice 1

Corporate Headquarters 1

Contact information 1

Trademarks 1

Table of Tables

Table 5-1 Common guard attributes for each slave guard 11
Table 5-2 Access attributes of each bus access 11
Table 5-3 Initial Value and Register Location of SPID 12
Table 5-4 Error notifications of the memory protection 15
Table 6-1 Register description for cluster RAM guards 22
Table 6-2 Register description for P-Bus guards 26
Table 7-1 ECM index list 29
Table 7-2 ECM Register Description 30
Table 8-1 Functions of User LED4 and LED5 32
Table 8-2 Functions of User LED Ring including LED10 to LED25 32
Table 8-3 External Input / Output and the Corresponding Pin Function of TAUD, MSPI and ECM 34
Table 8-4 Alternative Peripheral Function Selection 34

Table of Figures

Figure 5-1 Memory Protection Overview 10
Figure 6-1 Basic flow chart for CRG initialization 17
Figure 6-2 Basic flow chart for PBG initialization 24
Figure 7-1 Basic flow chart for ECM initialization 28

1. Background

Due to the increased complexity in future automotive applications and the corresponding functional safety requirements, the RH850/U2A-EVA devices provide guard and error control features, to control the access to memories, peripherals and other hardware components.

This application note introduces the guards and the error control feature ECM, describes the configuration as well as workflows using guards and ECM, and provides a software example.

The corresponding software example includes access to other peripherals such as the Timer Array Unit D (TAUD), the Multichannel Serial Peripheral Interface (MSPI) and the Interrupt Controller.

In this document, Section 2 describes the reference documents. For some detailed information or special use cases, please use this application note in combination with the references.

Section 3 lists of the in this document used abbreviations.

Section 4 provides a description about the hardware setups and software environment for the MCU applications with guards.

Section 5 contains general information regarding memory protection, focusing on ECM, CRG and PBG.

Sections 6 and 7 describe the SW procedures of ECM, CRG and PBG, which are used to configure and clear the error flags in the event of an error. These sections also contain associated register descriptions.

Section 8 explains the example software by describing its appearance, the peripherals used and the associated software.

2. Reference Documents

This Section contains information about the referenced documents.

2.1 RH850/U2A-EVA Group User’s Manual: Hardware

The Hardware User’s Manual provides information about the functional and electrical behavior of the device.

At the release time of this document the following manual version is available:

- RH850/U2A-EVA Group User’s Manual: Hardware (R01UH0864EJ0140, Rev.1.40)

It’s recommended to use the corresponding document or other later versions.

2.2 RH850/U2A16 Starter Kit

The Starter Kit User’s Manual provides information about the RH850/U2A16 Starter Kit and peripheral circuits.

At the release time of this document the following manual version is available.

- RH850/U2A16 Starter Kit User's Manual: Hardware (R20UT4861ED0300, Rev.3.00)

It’s recommended to use the corresponding document or other later versions.

3. Definitions and Abbreviations

For the purposes of this publication, the following abbreviations for common terms apply:

- **MSPI3:** Multichannel Serial Peripheral Interface 3
- **TAUD0:** Timer Array Unit D 0
  - **TAUD0C0:** Timer Array Unit D 0 Channel 0
  - **TAUD0C2:** Timer Array Unit D 0 Channel 2
- **CRG:** Cluster RAM Guard
  - **CRG0:** Cluster RAM Guard for CRAM0
  - **CRG1:** Cluster RAM Guard for CRAM1
  - **CRG2:** Cluster RAM Guard for CRAM2
  - **CRG3:** Cluster RAM Guard for CRAM3
- **PBG:** Peripheral Bus Guard
- **PE0:** Processor Element 0
- **PE1:** Processor Element 1
- **CRAM:** Cluster RAM
  - **CRAM0:** Cluster RAM in Cluster #0
  - **CRAM1:** Cluster RAM in Cluster #1
  - **CRAM2:** Cluster RAM in Cluster #2
  - **CRAM3:** Cluster/Retention RAM in Cluster #3
- **ECM:** Error Control Module
- **MPU:** Memory Protection Unit
- **SPID:** System Protection Identifier

4. Software and Hardware Tools

This section contains the information about software and hardware tools which are used to implement the Guard application.

4.1 Development Tools

The compiler used for the sample code is Green Hills Multi.

The required version of the GHS compiler for U2A-EVA devices is V7.1.6/2019.1.5 or later.

The EXEC, 850eserv2 and device files are also required before compiling the application code, for further information, please refer to the *Getting started software package - RH850U2A_GettingStarted_Vx.00.*

4.2 Emulator E2

The Renesas E2 Emulator is required for the debugging.

4.3 RH850/U2A-EVA Evaluation Boards

The following evaluation board is used to implement the Guard application.

4.3.1 Starter Kit Board

The Starter Kit Board Version 3, Y-ASK-RH850U2A16 carries the target device. The SW in this document is performed on the RH850/U2A16 device R7F702300BFABA-C within this this Starter Kit Board. The software uses the User LEDs and the encoder push button of the starter kit.

5. General information on ECM, CRG and PBG

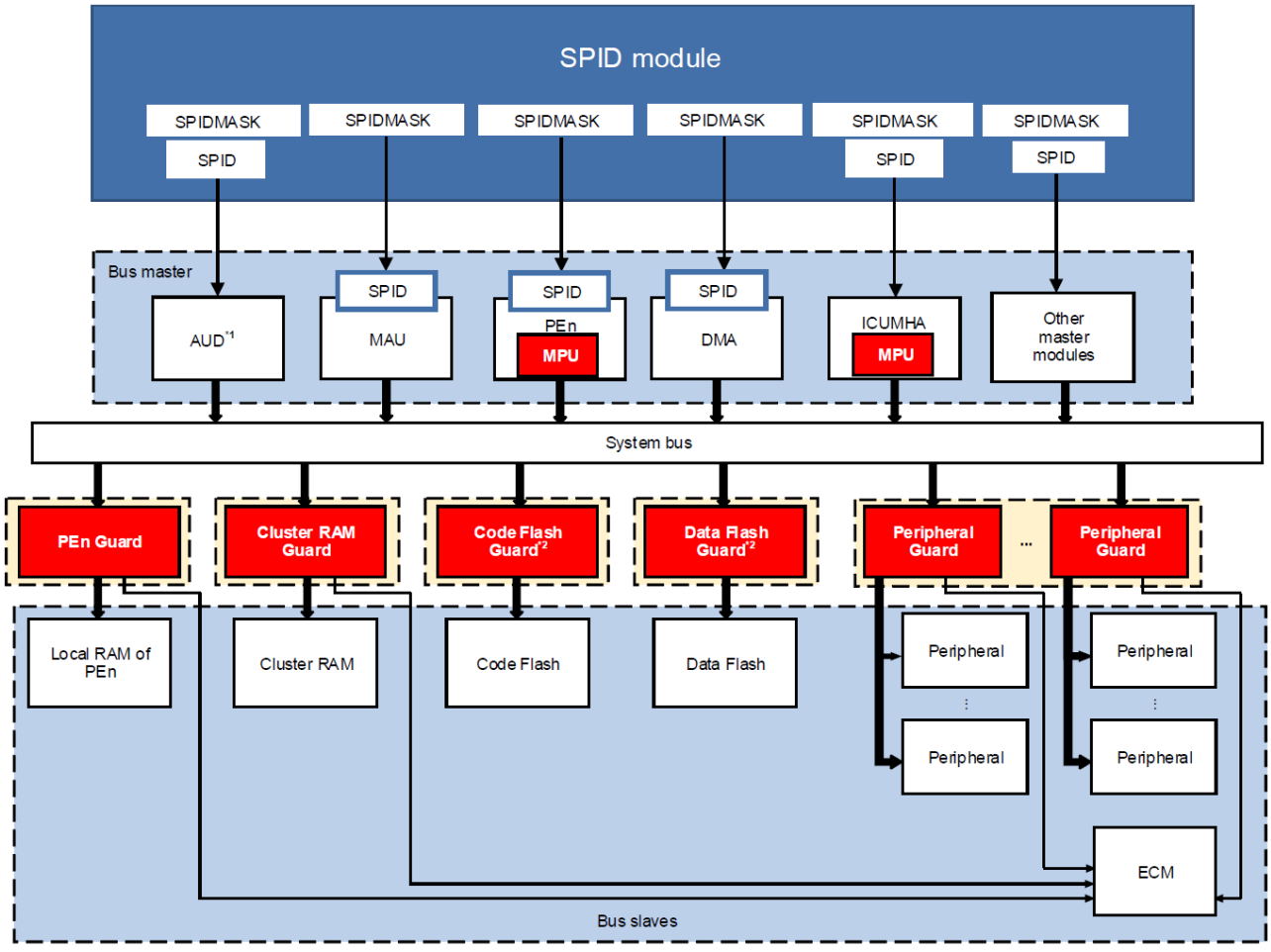
This section provides general information for the understanding of the ECM, CRG and PBG.

5.1 Memory Protection

The RH850/U2A-EVA group provides memory protection/guard functions to protect control- and memory-registers. *Figure 5-1* provides an overview of the memory protection architecture. The system consists of the SPID module, bus masters, system bus, guards, and bus slaves.

Access by a bus master via the system bus must be authorized by the respective guard. If this access is permitted according to the guard configuration, access to the bus slave takes place. If access is denied, the access is blocked and reported to the ECM in the case of the PEn Guard (PEG), CRG or PBG.

The Memory Protection Unit (MPU) of the CPU cores and ICUMHA blocks the requests of the CPU cores and ICUMHA if these address accesses are forbidden by the MPU. These accesses are then not executed. For details, refer to Hardware User's Manual R01UH0864EJxxxx *Sections 3 'CPU System' and 44.5 'Memory protection'* and *RH850/U2A-EVA Group Security User's Guide: Hardware*.



Notes: 1. AUD function is only supported in RH850/U2A-EVA.
2. For details of Code Flash Guard and Data Flash Guard, See the *RH850/U2A-EVA Group Security User's Manual: Hardware.*

Figure 5-1 Memory Protection Overview

Table 5-1 contains the common attributes of the Guard configuration registers, as well as their short description.

Table 5-1 Common guard attributes for each slave guard

Priority	Guard Attribute	Contents
High ↓ Low	SEC[*1]	Non-secure masters are blocked from access when enabled.
	DBG	All debug masters are allowed to access when enabled.
	UM	Masters in user mode are not allowed to access the target slave if this is disabled.
	SPID[m]	Masters, with the SPID m can access the target slave if this is enabled.
	WG	Any masters can write to the target slave if this attribute is enabled.
	RG	Any masters can read from the target slave if this attribute is enabled.

Notes: 1. This attribute is available only for PBG6L1.

Table 5-2 contains the common attributes of the respective bus accesses, which are also used in the error identifying registers.

Table 5-2 Access attributes of each bus access

Access attribute	Function
UM	Information about the operating mode of the bus master: 0: Supervisor mode 1: User mode Access from: CPU or ICUMHA[*1]: this access attribute shows the value stored in the system register of the CPU core. DTS, sDMAC or MAU[*2]: this access attribute displays the value set in each dedicated register. ICUM_AES0[*1]: always 1 Other masters: always 0
SEC	Specifies the access mode of ICUMHA or ICUM_AES0 when ICUMHA or ICUM_AES0 accesses an address. 0: Non-secure (access by other master) 1: Secure (access by ICUMHA/ICUM_AES0)
DBG	Indicates whether a debug master has accessed the address. 0: Normal mode (Normal access) 1: Debug mode (Debug access)
SPID	Specifies the SPID of the bus master that accessed the address.
WRITE	Indicates the access type of bus master. 0: Read access 1: Write access

Notes: 1. For details, see the RH850/U2A-EVA Group Security User's Manual: Hardware.
2. MAU is one of the debug functions to read and write the memory and registers.

5.2 CRG, PBG and ECM

This section contains information regarding memory protection. In addition to information on CRG and PBG, this information also includes information on the associated ECM error notifications and the SPID, which serve to understand CRG and PBG.

5.2.1 SPID

This section contains the basic configuration of the SPID that is required in relation to the use of the document.

The SPID serves as identifier for the bus masters of the memory protection. This means that certain SPIDs can be allowed or blocked for guards and errors can be identified.

Table 5-3 lists the default SPID configuration. This configuration can be changed. For further information, please refer to the Hardware User's Manual R01UH0864EJxxxx *Section 44.5.3 'SPID'*.

Table 5-3 Initial Value and Register Location of SPID

Initial value of SPID[*4]	Bus master	Register location
0	CPU0	CPU0
1	CPU1	CPU1
2	CPU2[*3]	CPU2
3	CPU3[*3]	CPU3
4 to 16	Reserved	-
17	Gb Ether[*6]	SPID module
18	Reserved	-
19	RHSIF0[*6]	SPID module
20	Reserved	-
21	Reserved	-
22	FlexRay1[*6]	SPID module
23	FlexRay0[*7]	
24	Fast Ether	
25	ICUM_AES0	
26	ICUMHA	
27	sDMAC1	sDMAC1
28	sDMAC0	sDMAC0
29	DTS[*1]	DTS
30	AUD[*5]	SPID module
31	MAU[*2]	MAU

Notes: 1. Initial value of SPID for DTS is stored in DTSRAM. Therefore, it is undefined after reset and it becomes 0 after RAM initialization. Hardware User's Manual R01UH0864EJxxxx *Section 9.5.6 'RAM Initialization'*.
2. MAU is one of the debug functions to read and write the memory and registers.
3. This function is not implemented in RH850/U2A8 (373/292 pins), RH850/U2A6.
4. The initial value shown in this table is the same as the 'Bus master index n' in the register description in this section.
5. This function is only implemented in RH850/U2A-EVA.
6. This function is not implemented in RH850/U2A6.
7. This function is not implemented in RH850/U2A6 (156-pins).

5.2.2 CRG – Cluster RAM Guard

This section contains information about the CRG.

5.2.2.1 General information

The CRG is part of the Functional Safety Division's memory protection and has two basic functions, inter-VM communication (IVC) and CRAM protection. The CRG can control read and write access and report violations to the ECM.

If the guard enable bits (GEN) are all at 0, access is allowed for the bus masters.

Each CRG has multiple channels consisting of eight GUARD_CRAMCRGn and four GUARD_CRAMCSGn channels. This results in the following channel relationship:

- Multiple GUARD_CRAMCRGn overlap: Access granted for bus master if one channel permits access.
- Multiple GUARD_CRAMCSGn overlap: Access for bus master not granted if one channel prohibits access.
- Overlap between GUARD_CRAMCRGn and GUARD_CRAMCSGn: Access permitted depending on GUARD_CRAMCSGn configuration

The CRG features a write access protection for the configuration relevant registers to protect the registers from unauthorized access.

5.2.2.2 IVC - Inter-VM Communication

IVC is used for communication between virtual machines (VMs) by means of shared memory blocks. This makes it possible to allocate shared memory blocks to a single owner without a Virtual Machine Monitor management. This function is then called Inter-VM Communication (IVC).

For this purpose, the IVC offers the following functions:

- CRG based access control
- Data exchange between VMs via the shared memory is limited to two VMs
- One VM alone may write to the memory, the other may only read. The VM with write access is called "owner". It has so-called "ownership" rights with the write access.
- The VMs can exchange the access rights with each other.
- The "ownership" rights can be requested and transferred or denied after confirmation of the VM that owns them.

5.2.3 PBG – Peripheral Bus Guard

The PBG is used for the MPU, for memory protection of the individual peripheries and is therefore also part of the functional safety. The PBG is connected to the system bus and the ECM.

The PBG is divided into 20 groups, which have a maximum of 16 channels. Each channel can control access to a single peripheral circuit. Details about blocked accesses of PBG channels are stored in the PBG group.

The PBG features a write access protection for the configuration relevant registers to protect the registers from unauthorized access.

5.2.4 ECM and Error notifications

This section contains the ECM error notifications that are of interest when using the MPU. It also includes a short description of the ECM.

The ECM is a central aspect in the field of functional safety. It offers many functions for error detection and interpretation to simplify error handling. For this purpose, errors can be signaled at the error outputs, interrupts can be generated, and ECM reset signals can be issued. Table 45.10 in the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 45.2.1 'Specification Overview'* contains further information on this topic.

For functional safety, the ECM has two instances: the ECM master and the ECM checker. The ECM checker is connected to the error inputs in the same way as the ECM master and should be treated in the same way as the ECM master.

The ECM is supplied by the CLK_HSB as PCLK and as count clock for the timer-based functions the CLK_ECMCNT as cntclk is applied.

The ECM generates the following interrupts:

- INTECMME as ECM maskable interrupt (EI Level) with interrupt number EIINT8
- INTECMDCLSMI as DCLS error interrupt (EL level) with interrupt number EIINT9
- INTECMNMI as ECM maskable interrupt (FE level) with interrupt number FEINT

The ECM has the following inputs/outputs:

- $\overline{\text{ERROROUT_M}}$ as error output master signal
- $\overline{\text{ERROROUT_C}}$ as error output checker signal
- $\overline{\text{ERRORINk}}$ as external error input signal k

It also has the following internal output signals:

- ERROROUTZ as error trigger signal for Hi-Z control of the PIC
- ERROROUTnZ as error trigger signal n for the port safe state
- ECMRESZ for the internal reset generation

The ECM offers the possibility to manage and simulate error sources. For this purpose, error sources can be masked via the associated bits or the respective pseudo errors can be initiated. These pseudo errors behave like real errors.

More information regarding the ECM, like the clock supply, the default error output states or more features, can be found in the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 45 'Error Control Module (ECM)'*.

Table 5-4 lists the memory protection relevant error notifications of the ECM.

Table 5-4 Error notifications of the memory protection

Error Notification	Description	ECM Error Number
CRAM Guard error	CRG error	168
I-Bus Guard error	IBG error	169
P-Bus Guard error	DTS guard error sDMAC guard error INTC2 guard error PBG error	170
H-Bus Guard error	HBG error	171
PE Guard error (PE0)	PEG error by access to PE0 slave	240
PE Guard error by PE0 read access (PE0)	PEG error detected in a read request from PE0 to the other LRAM	241
PE Guard error (PE1)	PEG error by access to PE0 slave	272
PE Guard error by PE1 read access (PE1)	PEG error detected in a read request from PE0 to the other LRAM	273
PE Guard error (PE2)	PEG error by access to PE0 slave	304
PE Guard error by PE2 read access (PE2)	PEG error detected in a read request from PE0 to the other LRAM	305
PE Guard error (PE3)	PEG error by access to PE0 slave	336
PE Guard error by PE3 read access (PE3)	PEG error detected in a read request from PE0 to the other LRAM	337

Note: For details, see Hardware User's Manual R01UH0864EJxxxx *Sections 3 'CPU System'* and *Section 44.5 'Memory Protection'* and the *RH850/U2A-EVA Group Security User's Manual: Hardware*.

6. SW procedure and module setup for CRG and PBG applications

This section provides a description of the software procedure and related configurations for guard applications containing CRG or PBG.

For detailed information of the registers which are mentioned in this section refer to the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx*.

6.1 Cluster RAM Guard

This section contains information about the software procedures used to initialize the CRG and to identify and clear errors. It also contains a register description of the mentioned registers.

6.1.1 CRG Setup

The setup procedure of the CRG is shown in *Figure 6-1*.

To edit the register values, the register protection must first be disabled. To do this, the value $A5A5A501_H$ is written to the CRGKCPROT.

Then the configuration registers can be edited. For this the valid SPIDs are set in the CRGSPIDm register, whose rights are to be configured.

The base address from which the guard channel is to protect the cluster RAM can also be set. This is set in the CRGBADm register and refers to the CRAM base address. The range from the address configured in the BAD bits is limited by the ADV bits in the CRGADVm register. For more information see the example below.
Example:

- Guard area configuration using CRGBADm & CRGADVm for CRAM0:

  CRAM0 base address: FE00 0000_H → 1111 1110 0000 0000 0000 0000 0000 0000_B

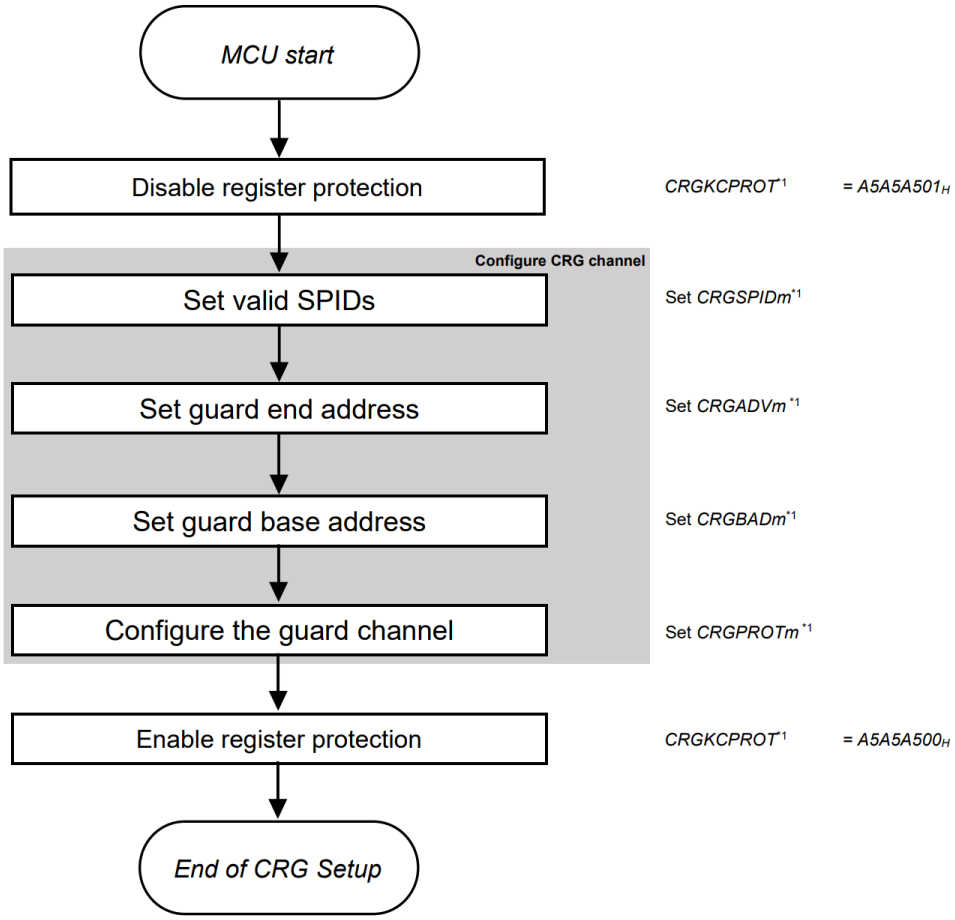
  CRGBADm.BAD[18:8] = 000 0000 0000_B → 1111 1110 0000 0[000 0000 0000] 0000 0000_B

  CRGADVm.ADV[18:8] = 111 1110 1110_B → 1111 1110 0000 0[000 000X 000X] XXXX $XXXX_B$

  The areas protected by the Guard are FE00 0000_H to FE00 $00FF_H$, FE00 0100_H to FE00 $01FF_H$, FE00 1000_H to FE00 $10FF_H$ and FE00 1100_H to FE00 $11FF_H$.

To finalize the configuration the CRGKCPROTm register must be set. It activates the guard configuration and configures the read and write access globally or to certain peripherals defined in CRGSPIDm. It can also take over the protection for the user mode, the debug master and the Inter-VM communication. For details, please refer to RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 3.8 'Inter-VM Communication (IVC)'*.

To ensure functional safety and to avoid unforeseen changes, register protection should be re-enabled by writing A5A5A500H to the CRGKCPROT register.



Notes: 1. For details of the ECM registers, please refer to RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 44.5.5 'CRG'*.

Figure 6-1 Basic flow chart for CRG initialization

6.1.2 Detecting and Clearing errors

If errors occur during the runtime of the MCU Cluster RAM Guard, this section describes how to detect and identify them and how to clear the associated error flags.

For larger applications, the use of ECM is recommended. This can initially indicate during runtime that an error has occurred and in connection with which peripheries this has occurred. Then the errors can be analyzed in more detail. In case of CRG errors those can be identified using the following registers:

- **CRGERRADDR**
- **CRGERRTYPE**
- **CRGOVFSTAT**
- **CRGSPIDERRSTAT**

The CRGERRADDR register contains information about which specific register address was accessed when the violation occurred and the CRG blocked the access.

The CRGERRTYPE register provides information about the type of access. It contains information about whether it was a write or read access, which SPID caused the error, whether the user or debug mode was active, or the SEC was active.

The CRGSPIDERRSTAT register indicates which SPID caused the error.

The CRGOVFSTAT register shows whether several accesses have already been blocked.

These registers are used to clear the error flags:

- **CRGSPIDERRCLR**
- **CRGOVFCLR**

To clear the error flags mentioned in the previous section, the corresponding bits in the registers must be set. To reset the overflow flag (OVF) in the CRGOCFSTAT register, the CLRO bit in the CRGOVFCLR register must be set. To reset the SPID error flag (SPIDERR[31:0]) of the SPID that caused the error, the corresponding SPIDCLR[31:0] bit in the CRGSPIDERRCLR register must be set.

Note for the use of CRGs in connection with the ECM: For the ECM error to occur again, the SPID flags should be cleared, otherwise no bit change from 0 to 1 can take place in the event of a new error.

More information on these registers can be found in *section 6.1.4 Register description*.

6.1.3 Inter-VM Communication (IVC)

This section contains information about the Inter-VM communication (IVC) and its configuration. The IVC function is based on the CRG function.

6.1.3.1 Configuration

The IVC is configured by the Virtual Machine Monitor management. Many of the configuration steps are similar to section *6.1.1 CRG Setup*.

Figure 6-2 represents the IVC configuration sequence.

To edit the register values, the register protection must first be disabled. To do this, the value $A5A5A501_H$ is written to the CRGKCPROT.

The message box size must be set for the IVC via the CRGBADm and CRGADVm register. For more information, see Section *6.1.1 CRG Setup*.

In the CRGSPIDm register the sender VM is defined, whereas in the CRGIVCSPIDm register the two communication partners of the IVC are defined (sender and receiver VM). This means that in CRGSPIDm one bit must be set, whereas in CRGIVCSPIDm two bits must be set.

Then the CRGPROTm register must be set. For this it is necessary to set the GEN and the OW bit to 1. (00010100_H)

To ensure functional safety and to avoid unforeseen changes, register protection should be re-enabled by writing A5A5A500H to the CRGKCPROT register.

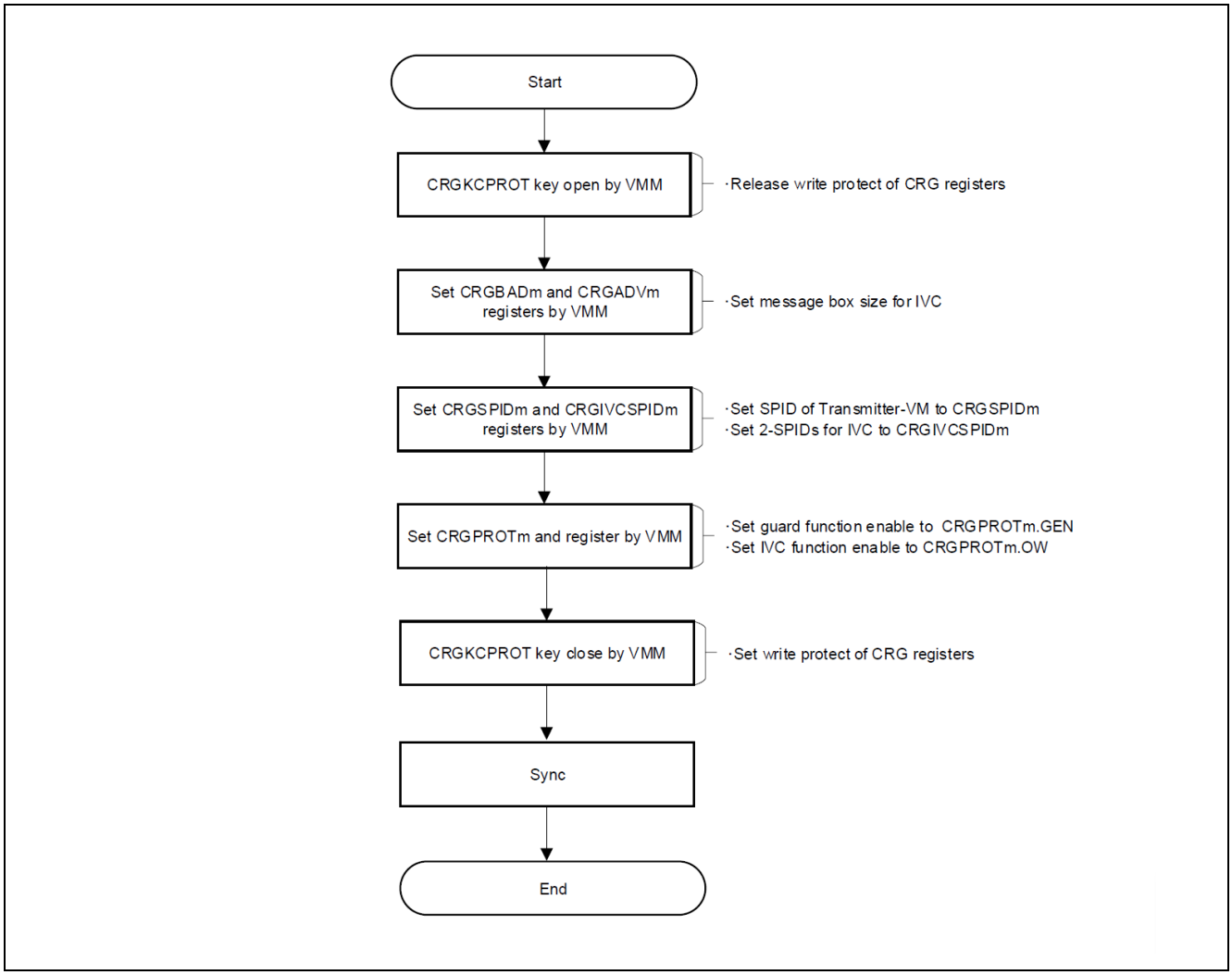


Figure 6-2 - IVC Configuration

6.1.3.2 Data Transmission

Figure 6-3 represents the IVC configuration sequence.

At the beginning it should be checked if the Owner VM is the same as the Transmitter VM. For this purpose the CRGIVCOWNRm register must be read.

If this does not match, it must be set in the CRGIVCREQm register by the appropriate SPID bit.

Otherwise the configured message box can be written with data from the transmitter VM.

Then the owner VM must be reconfigured to the receiver VM by setting the corresponding bit of the SPID in the CRGIVCREQm register.

After a register synchronization an interrupt is sent to the Reciever-VM.

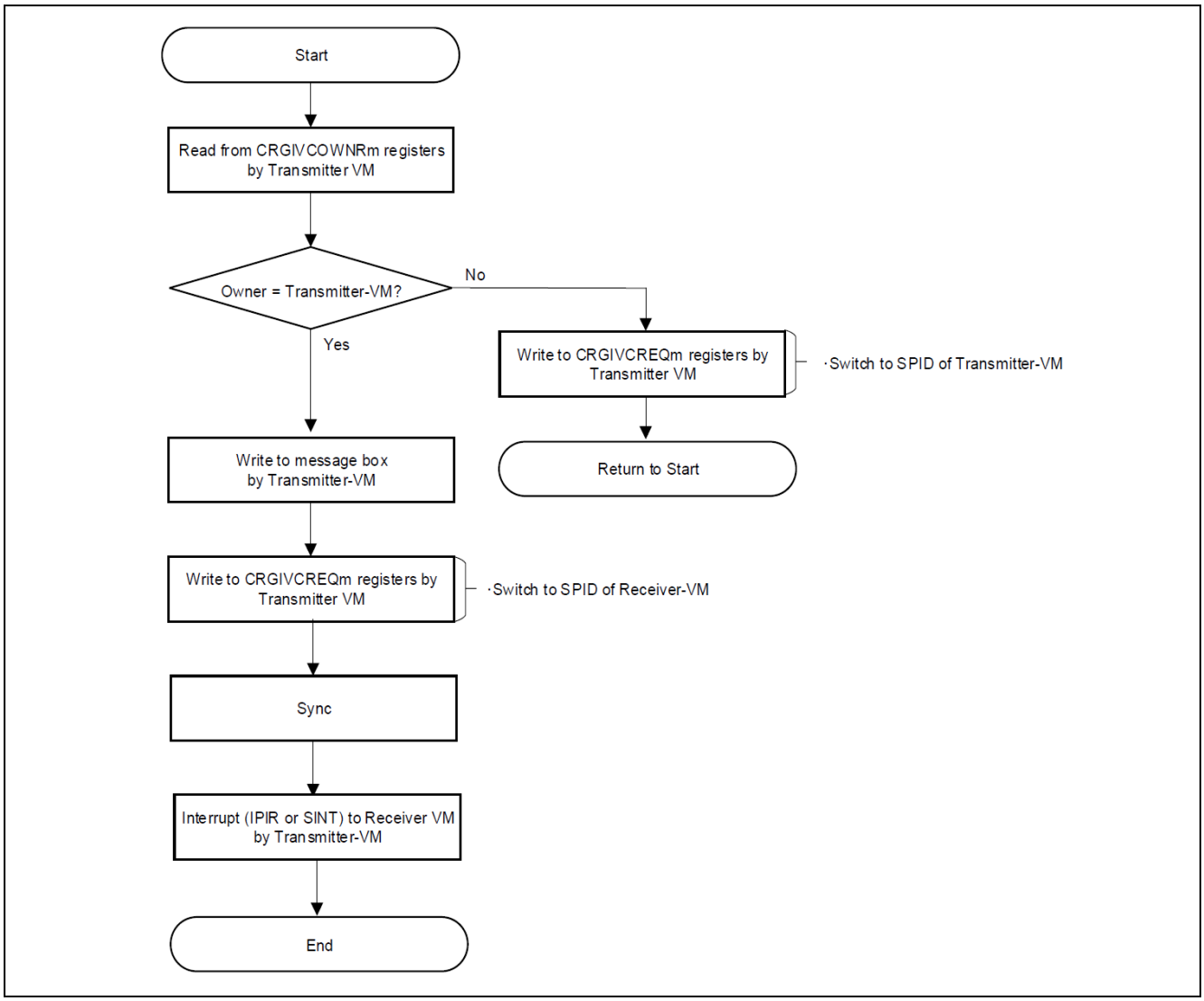


Figure 6-3 - IVC Data Transmission

6.1.3.3 Data Reception

Figure 6-4 represents the IVC configuration sequence.

To read data from the IVC, ownership requests from sender VMs must be blocked. To do this, the LK bit of the CRGIVCLOCK register must be set.

Then the CRGIVCOWNRm register must be read to check if the owner VM matches the receiver VM.

If this is not the case, an error has occurred (Lock fail). Then wait for the interrupt from the sender VM and restart the process.

If the VMs match, the message box can be read.

Then the CRGIVCREQm register should be configured to the transmitter VM. This will automatically clear the CRGIVCLOCK register and unblock ownership requests.

After a register synchronization an interrupt is sent to the transmitter VM.

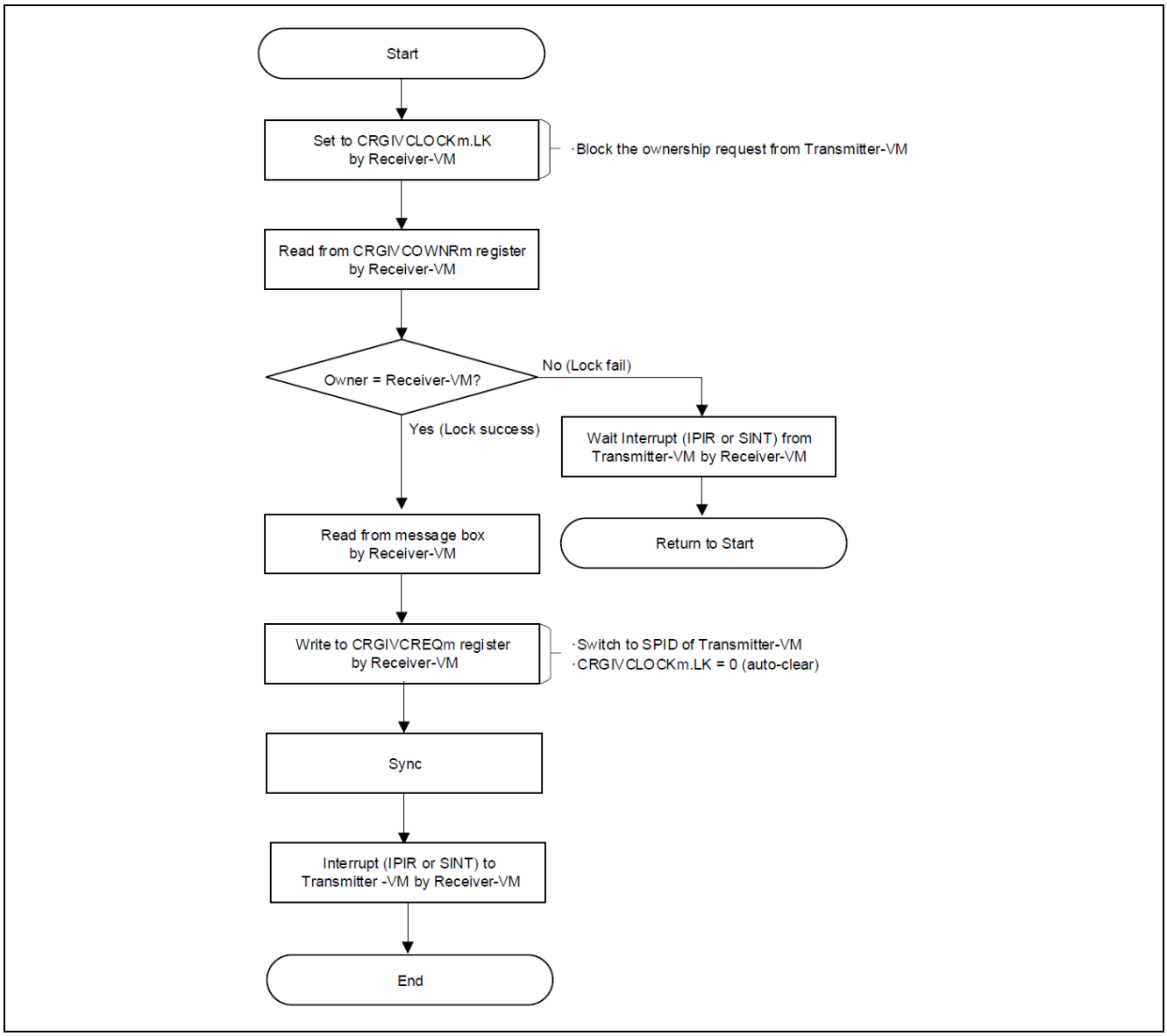


Figure 6-4 - IVC Data Recieve

6.1.4 Register description

Table 6-1 contains information on the main registers of the CRG.

Table 6-1 Register description for cluster RAM guards

Register	Function	Function (Bit-specific)	Bit Name	Bit Position
CRGKCPROT	Enable/Disable modification of KCE Bit.		KCPROT[31:1]	31 to 1
	Lock/Unlock guard register protection.	1: Unlock protection 0: Lock protection	KCE	0
CRGPROTm	Configure the Guard	Inter-VM Communication: 1: Enable, 0: Disable	OW	16
		Guard Setting: 1: Enable, 0: Disable	GEN	8
		For details see *Table 5-1 Common guard attributes for each slave guard*	DBG	6
			UM	4
			WG	1
			RG	0
CRGSPIDm	Set SPIDs for CRGPROTm	Bit number corresponds to SPID number	SPID[31:0]	31 to 0
CRGBADm	Set CRG base address to specify a range for this guard channel.	Base address	BAD[20:19][*1]	20,19
		Base address	BAD[18][*2]	18
		Base address	BAD[17][*2]	17
		Base address	BAD[16:8]	15 to 8
CRGADVm	Configure CRG range for the guard channel.	Valid bits of CRGBADm	ADV[20:19][*1]	20,19
		Valid bits of CRGBADm	ADV[18][*2]	18
		Valid bits of CRGBADm	ADV[17][*2]	17
		Valid bits of CRGBADm	ADV[16:8]	15 to 8
CRGERRADDR	Memory address to which unauthorized access has been blocked.		ADDR[31:0]	31 to 0
CRGERRTYPE	Non-secure master access	For details see *Table 5-2 Access attributes of each bus access*	SEC	13
	Debug Master access		DBG	12
	User mode access		UM	11
	Blocked SPID		SPID[4:0]	10 to 6
	Read/write access		WRITE	0
CRGOVFSTAT	Overflow flag is set if the same SPID causes an error again		OVF	1
CRGSPIDERRSTAT	Information about which SPID caused an access error		SPIDERR[31:0]	31 to 0
CRGSPIDERRCLR	Clear associated SPID flag in CRGSPIDERRSTAT		SPIDCLR[31:0]	31 to 0
CRGOVFCLR	Clear overflow flag in CRGOVFSTAT		CLRO	1
CRGIVCSPIDm	Defines used VMs for IVC	Bit number corresponds to SPID number	IVCSPID[31:0][*3]	31 to 0
CRGIVCREQm	Sets an ownership request for IVC	Bit number corresponds to SPID number	IVCREQ[31:0]	31 to 0

CRGIVCLOCKm	Lock ownership requests from non-owner VMs	1: Ownership requests blocked 0: Ownership requests from non-owner VMs are accepted	LK	0
CRGIVCOWNRm	Indicates SPID of IVC owner	Bit number corresponds to SPID number	IVCOWN[31:0]	31 to 0

Notes: 1. Implemented for CRG2. Reserved bit and read only for CRG0, CRG1 and CRG3 in RH850/U2A-EVA, RH850/U2A16 and RH850/U2A8.
Reserved bit in RH850/U2A6.
2. RH850/U2A-EVA, RH850/U2A16 and RH850/U2A8: Implemented only in CRG0, CRG1 and CRG2. Reserved bit and read only CRG3
RH850/U2A6: Implemented only in CRG0. Reserved bit and read only for CRG3.
3. It is mandatory to set only two bits to control IVC.

For the detailed information of the registers which are mentioned in this section refer to the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 44.5.5 'CRG'*.

6.2 Peripheral Bus Guard

This section contains information about the software procedures used to initialize the PBG and to identify and clear errors. It also contains a register description of the mentioned registers.

6.2.1 PBG setup

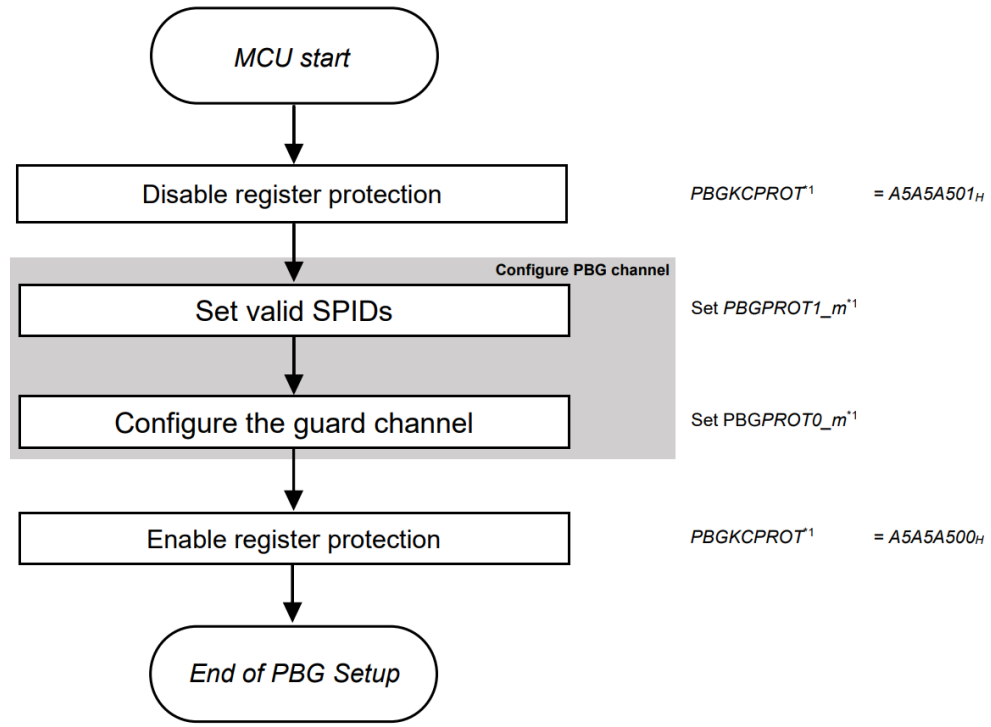
The setup procedure of the PBG is shown in *Figure 6-6-5*.

To edit the register values, the register protection must first be disabled. To do this, the value A5A5A501$_H$ is written to the PBGKCPROT.

The PBGPROT1_m register specifies, if required, the valid SPIDs whose permissions are to be configured.

The PBGPROT0_m register configures the most important settings for the PBG channel. Global or SPID-dependent read and write permissions can be assigned, the access rights for the user mode and the non-secure master can be restricted, and access rights to the debug master can be assigned. It is also possible to activate the guard settings and to lock the guard configuration until the next reset, so that they can no longer be modified.

To ensure functional safety, to avoid unforeseen changes, register protection should be re-enabled by writing A5A5A500$_H$ to the PBGKCPROT register.



Notes: 1. For details of the ECM registers, please refer to RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 44.5.10 'PBG'*.

Figure 6-6-5 Basic flow chart for PBG initialization

6.2.2 Error identification & clearing

If errors occur during the runtime of the MCU PBG, this section describes how to detect and identify them and how to clear the associated error flags.

For larger applications the use of ECM is recommended. This can indicate at runtime that an error has occurred and in connection with which peripherals this occurred. Then the errors can be analyzed in more detail. In the case of PBG errors with the help of the following registers:

- **PBGERRADDR**

- **PBGERRTYPE**
- **PBGOVFSTAT**
- **PBGSPIDERRSTAT**

The PBGERRADDR register contains information about which specific register address was accessed when the violation occurred and the PBG blocked the access.

The PBGERRTYPE register provides information about the type of access. It contains information about whether it was a write or read access, which SPID caused the error, whether the user or debug mode was active, or the SEC was active.

The PBGSPIDERRSTAT register indicates which SPID caused the error.

The PBGOVFSTAT register shows whether several accesses have already been blocked.

These registers are used to clear error flags:

- **PBGSPIDERRCLR**
- **PBGOVFCLR**

To clear the error flags mentioned in the previous section, the corresponding bits in the registers must be set. To reset the overflow flag (OVF) in the PBGOCFSTAT register, the CLRO bit in the PBGOVFCLR register must be set. To reset the SPID error flag (SPIDERR[31:0]) of the SPID that caused the error, the corresponding SPIDCLR[31:0] bit in the PBGSPIDERRCLR register must be set.

Note for the use of PBGs in connection with the ECM: For the ECM error to occur again, the SPID flags should be cleared, otherwise no bit change from 0 to 1 can take place in the event of a new error.

More information on the ECM cand be found in section *7* SW procedure and module setup for ECM applications

More information on these registers can be found in section *6.2.3 Register description*.

6.2.3 Register description

Table 6-2 contains information on the main registers of the PBG.

Table 6-2 Register description for P-Bus guards

Register	Function	Function (Bit-specific)	Bit Name	Bit Position
PBGKCPROT	Enable/Disable modification of KCE Bit.		KCPROT[31:1]	31 to 1
	Un-/Lock guard register protection.	1: Unlock register protection 0: Lock register protection	KCE	0
PBGPROT0_m	Configure the Guard	1: Lock guard configuration 0: Unlock guard configuration	LOCK	31
		1: Enable guard setting 0: Disable guard setting	GEN	8
		For details see *Table 5-1 Common guard attributes for each slave guard*	SEC	7
			DBG	6
			UM	4
			WG	1
			RG	0
PBGPROT1_m	Set SPIDs for PBGPROT0_m	Bit number corresponds to SPID number	SPID[31:0]	31 to 0
PBGERRADDR	Address to which unauthorized access has been blocked.		ADDR[31:0]	31 to 0
PBGERRTYPE	Non-secure master access	For details see *Table 5-2 Access attributes of each bus access*	SEC	13
	Debug Master access		DBG	12
	User mode access		UM	11
	Blocked SPID		SPID[4:0]	10 to 6
	Read/write access		WRITE	0
PBGOVFSTAT	Indicator whether multiple errors have occurred	Overflow flag: 1: Multiple errors occurred 0: One or less error occured	OVF	1
PBGSPIDERRSTAT	Information about which SPID caused an access error	Bit number corresponds to SPID number	SPIDERR[31:0]	31 to 0
PBGSPIDERRCLR	Clear associated SPIDERR flag in PBGSPIDERRSTAT	1: Clear SPIDERR flag	SPIDCLR[31:0]	31 to 0
PBGOVFCLR	Clear overflow flag in PBGOVFSTAT	1: Clear OVF flag	CLRO	1

For the detailed information of the registers which are mentioned in this section refer to the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 44.5.10 'PBG'*.

7. SW procedure and module setup for ECM applications

This section provides a description of the software procedure and related configurations.

For the detailed information of the registers which are mentioned in this section refer to the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx*.

7.1 ECM Error Output reset and Error configuration

Figure 7-1 shows the setup workflow of the ECM. This workflow is intended to reset the error output pins (return to (inactive) high level) and to pass certain error sources so that they can be processed via an interrupt and to signal an error that has occurred at the error output pins.

To edit the register values, the register protection must first be disabled. To do this, the value $A5A5A501_H$ is written to the ECMKCPROT.

Since the $\overline{\text{ERROROUT_M}}$ assumes the active low level in most cases after reset or boot up, as described in RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 45 'Error Control Module (ECM)'*, this must first be reset.
For this purpose, depending on the application, it should be checked whether errors exist, unless all error sources are masked by means of ECMEMKj registers. If no errors exist, the pseudo errors of the "ECM compare error" for ECM checker and master are masked by setting the ECMPEM register to 00000003_H. Then the ECMmECLR registers set an error clear trigger for both checker and master by writing 00000001_H, which resets the error outputs to the inactive high level. Finally, the pseudo errors can be reactivated (by writing 000000_H to the ECMPEM register).

To process specific errors at runtime, they must first be enabled. This is done by enabling the interrupt generation for the desired errors on the desired PE. For this purpose, the corresponding bits of the registers ECMINCFGi_j are set to 1 for PEi.

To handle the errors in the ECM, the error source in the ECMEMKj register must be unmasked by setting the corresponding bits to 0.

Example:

- Enable Error source 168 for error output and interrupt on PE1:

  ECMINCFG1_5 = 00000100_H → Interrupt generation for ECM error source 168 on PE1 enabled → EIINT8 will occur on PE1 in case of an error

  ECMEMK5 = $FFFFFEFF_H$ → Unmasks ECM error source 168 for error output

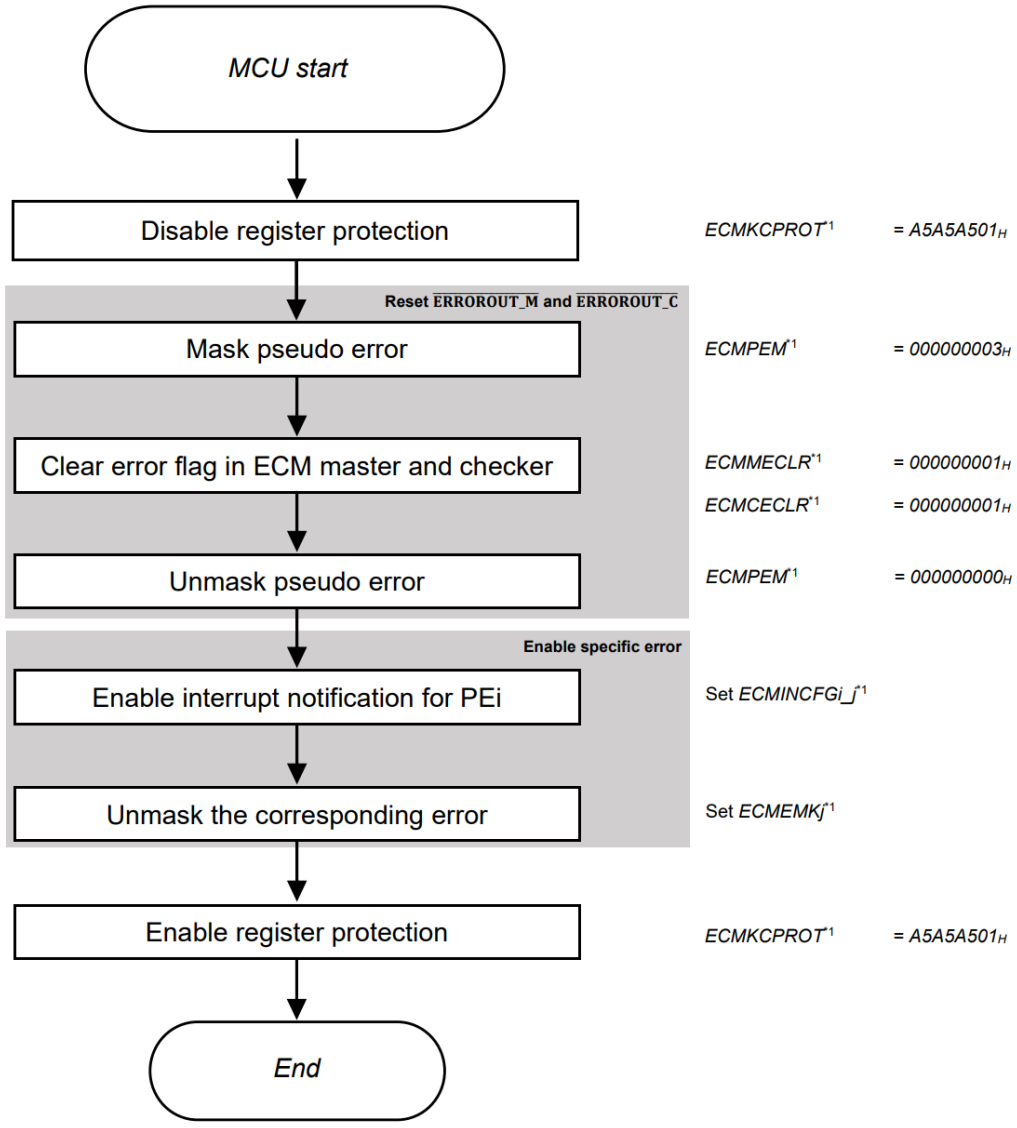
To ensure functional safety, to avoid unforeseen changes, register protection should be re-enabled by writing $A5A5A500_H$ to the ECMKCPROT register.

There are also other configuration options available, which are briefly outlined here:

- Configuring a dynamic error output, by resetting the error output, configuring an OSTM0 and by enabling the dynamic mode in the ECMEPCFG (with the value 00000001_H). After that the error outputs toggle with the OSTM0 speed if there is no error.
- Configure the ECM error interrupt as EI level maskable interrupt or FE level maskable interrupt. For this the bits in ECMISCFGj must be set.
- Configuring an internal reset when a certain error occurs, which forces an ECM internal reset at the reset controller. For this the corresponding bit in the ECMIRCFGj register must be set to 1.
- Configuration of an error trigger, which can control the port output ERROROUTnZ. For this the respective bit must be set in the desired ECMETMKn_j register.
- Configuring an ECM error out delay timer is also possible. For this the ECMDTMR register must be set with a counter value, the compare register must be set to the same value and one or more

channels must be set via the ECMDTMCFGj register which starts the delay timer. The timer can also be started and stopped manually via the ECMDTMCTL register.

- It is also possible to configure an error output clear invalidation function, which defines after what time after the occurrence of the error the error output of the ECM may be reset. For this the function must be enabled in the ECMEOCCFG register and a counter value must be defined, which determines the timer duration. The error trigger n can also be provided with such a function. For this the ECMETCCFGn register must be set. Also, there the function must be enabled, and a counter value must be set, which determines the duration. In both cases it is to be noted that the CMPW bit must be on 0, so that the register values can be edited.



Notes: 1. For details of the ECM registers, please refer to RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 45.3 'Register Specification'*.

Figure 7-1 Basic flow chart for ECM initialization

7.2 Error identification & clearing

If errors occur during the runtime of the MCU ECM, this section describes how to detect and identify them and how to clear the associated error flags.

To identify an error the ECMMESSTRj registers are read. From this the error source can be derived which has caused the error.

For example: If ECMMESSTR5 takes the value 00000100_H and the ECMMESSTR7 takes the value 80000000_H, it means that the error sources 168 and 255 have occurred. From this it is likely that PE0 was blocked by a CRG when accessing one of the CRAMs.

The value of the ECMMESSTRj registers can be verified by reading the ECMCESSTRj registers.

These errors can be cleared by setting the corresponding bits to 1 in the ECMESSTCj registers. To prevent the errors from recurring, the error flags should be cleared, especially when using guards.

If error output is enabled on the error output pins, these must be reset as in the configuration flow after the errors have been cleared.

More information on these registers can be found in section 7*.3 Register description*.

7.3 Register description

This section contains information about the main registers of the ECM module.

Table 7-1 ECM index list

Index	RH850/U2A-EVA U2A16 mode (516 pins), RH850/U2A16 (516/373/292 pins)	RH850/U2A-EVA U2A8 mode (516 pins), RH850/U2A8 (373/292 pins), RH850/U2A6 (292/176/156/144 pins)	Meaning
i	i = 0 to 3	i = 0 to 1	Throughout section 7, the core number is identified by the index “i”.
j	j = 0 to 10	j = 0 to 8	Throughout section 7, the error number divided by 32 is identified by the index “j”.
k	k = 0 to 3	k = 0 to 1	Throughout section 7, the external error input number is identified by the index “k”.
m	m = M, C		Throughout section 7, the individual ECM Master and ECM Checker are identified by the index “m”.
n	n = 0 to 3	n = 0 to 1	Throughout section 7, the Error trigger number is identified by the index “n”.

Table 7-2 ECM Register Description

Register	Function	Function (Bit-specific)	Bit Name	Bit Position
ECMKCPROT	Enable/Disable modification of KCE Bit.		KCPROT[31:1]	31 to 1
	Un-/Lock guard register protection.	1: Unlock register protection 0: Lock register protection	KCE	0
ECMPEM	Mask Pseudo error of “ECM compare error”	1: Masked for ECM master 0: Unmasked for ECM master	MSKM	1
		1: Masked for ECM checker 0: Unmasked for ECM checker	MSKC	0
ECMmECLR	Master/checker Error clear trigger	1: Reset pin output level to inactive (high) level	ECMmECT	0
ECMINCFGi_j[*1]	Interrupt Notification Configuration	1: Interrupt generation enabled 0: Interrupt generation disabled	ECMIEi_j_31 to ECMIEi_j_0	31 to 0
ECMEMKj[*1]	Masks error sources for error pin output	1: Error output masked 0: Error output not masked	ECMEMKj_31 to ECMEMKj_0	1
ECMESSTCj[*1]	Clears error flags for master and checker in ECMmESSTRj	Associated error flag: 1: Cleared 0: Unchanged	ECMCLSSEj_31 to ECMCLSSEj_0	31 to 0
ECMmESSTRj[*1]	Error source status register	1: Error occurred 0: Error not occured	ECMmSSEj_31 to ECMmSSEj_0	31 to 0
ECMEPCFG	Error Pulse Configuration	1: Dynamic mode 0: Non-dynamic mode	ECMSL0	0
ECMISCFGj[*1]	Interrupt type Selection Configuration	1: Maskable FE level interrupt 0: Maskable EI level interrupt	ECMISj_31 to ECMISj_0	31 to 0
ECMIRCFGj[*1]	Internal Reset Configuration	ECM internal reset generation: 1: Enabled, 0: Disabled	ECMIREj_31 to ECMIREj_0	31 to 0
ECMETMKn_j[*1]	Mask error trigger n for error sources	Error trigger output: 1: Masked, 0: Not masked	ECMETMKn_j_31 to ECMETMKn_j_0	31 to 0
ECMDTMR	Current delay timer counter value	ECMDTMR[23:0]	23 to 0	
ECMDTMCMP	Delay timer compare register	Writing progress to ECMDTMCMP ongoing: 1: In progress, writing to ECMDTMCMP prohibited 0: Not in progress	CMPW	24
		Delay timer compare value	ECMDTMCMP[23:0]	23 to 0
ECMDTMCFGj[*1]	Delay timer configuration	Delay timer start for error source: 1: Enabled, 0: Disabled	ECMDTMEj_31 to ECMDTMEj_0	31 to 0

ECMDTMCTL	Delay timer control	Delay timer start status: 1: Timer does not start 0: Timer starts	DTMSTACNTCLK	4
		Delay Timer stop bit: 1: Stop requested 0: Timer completed or stop is not requested	DTMSTP	1
		Delay Timer start bit: 1: Delay timer starts 0: Delay timer not started	DTMSTA	0
ECMEOCCFG	Error Output Clear Invalidation Configuration	Invalidation function: 1: Enabled, 0: Disabled	EOCIEN	31
		Delay Counter: 1: Counter in progress 0: Not in progress, ECMEOUTCLRT can be edited	CMPW	24
		Clock cycle Number of time to wait to reset the error output by SW	ECMEOUTCLRT [23:0]	23 to 0
ECMETCCFGn	Error Trigger n Clear Invalidation Configuration	Invalidation function: 1: Enabled, 0: Disabled	ETCIEN	31
		Delay Counter: 1: Counter in progress 0: Not in progress, ECMETRGCLRT can be edited	CMPW	24
		Clock cycle Number of time to wait to reset the error trigger by SW	ECMETRGCLRT [23:0]	23 to 0

Notes: 1. The source of the error can be determined as follows: Error source No [j x 32 + bit number].

7.4 Pin Assignment

For the external input/output and pin assignment of ECM, please refer to the attachment file: *ECM_External_Input_Output_and_Pin_Assignment.xlsx*.

8. Description of Sample Software

This section contains information about the behavior and appearance of the sample software application. It also contains information about the port and clock setup of the TAUD and MSPI used. At the end of this section the used C-files and header files are described.

8.1 Application description and appearance

This section contains information about the behavior and appearance of the application.

8.1.1 User LEDs

The user LEDs indicate whether the application is running and whether the peripheral guard is working.

Table 8-1 Functions of User LED4 and LED5

LED	Function
LED4	Lights up continuously as long, as the application is running.
LED5	Controlled by the TAUD0 channel 2 output and flashes every second during the application is running.

8.1.2 LED Ring

The user LED ring is used to indicate test selection, time remaining to selection and as an error indicator.

Table 8-2 Functions of User LED Ring including LED10 to LED25

<table>
<tr><th rowspan="2">LED</th><th colspan="2">Function</th></tr>
<tr><th>Description</th><th>Remaining Time</th></tr>
<tr><td>LED10</td><td rowspan="6">Countdown for test selection:
Each LED stands for 0.5 seconds remaining. When the test selection is started, all LEDs turn on. Every 0.5 seconds, another LED goes off, starting from the outermost one.</td><td>3s</td></tr>
<tr><td>LED11</td><td>2.5s</td></tr>
<tr><td>LED12</td><td>2s</td></tr>
<tr><td>LED13</td><td>1.5s</td></tr>
<tr><td>LED14</td><td>1s</td></tr>
<tr><td>LED15</td><td>0.5s</td></tr>
<tr><td>LED16</td><td colspan="2">-</td></tr>
<tr><td colspan="2"></td><td>Related Test</td></tr>
<tr><td>LED17</td><td rowspan="6">Guard error indication:
LEDs on: guard error occurred
LEDs off: guard error cleared</td><td rowspan="2">Test 3</td></tr>
<tr><td>LED18</td></tr>
<tr><td>LED19</td><td rowspan="2">Test 2</td></tr>
<tr><td>LED20</td></tr>
<tr><td>LED21</td><td rowspan="2">Test 1</td></tr>
<tr><td>LED22</td></tr>
<tr><td>LED23</td><td rowspan="3">Test Selection:
LED on: test selected
LED off: test not selected</td><td>Test 3</td></tr>
<tr><td>LED24</td><td>Test 2</td></tr>
<tr><td>LED25</td><td>Test 1</td></tr>
</table>

8.1.3 Test Selection

To test more than one guard, three different tests are implemented. These tests can be selected by pressing the Encoder push button.

The tests can be selected using the encoder push button, which is connected to the external interrupt 5. The test can be selected by pressing the button continuously expected times regarding to the test number, i.e., to select Test 3, the button should be pressed 3 times continuously. The button can be pressed up to 3 times in a test selection phase to select the tests, if the button is pressed more than 3 times, test 3 remains selected.

When the button is pressed for the first time, a countdown of 3.5 seconds starts, which limits the test selection time. The countdown is visible on LED10 to LED15. After this selection time, the selected test is

performed automatically. The selected test is shown on LED23 to LED25 (see Table 5-2 for more information on the LED ring). LED17 to LED22 show which guard error has occurred. If all LEDs light up simultaneously, an unexpected ECM error occurs.

8.1.4 Test Description

The application contains the following tests.

8.1.4.1 Test 1 – Cluster RAM Guard – Processor Element 1

Test 1 is executed by PE1. For this test, the CRGs and the ECM for CRAM2 and CRAM3 are initialized. The guards are configured so that only PE1 and the debugger can access CRAM2 (read and write access). CRAM3 can only be accessed by PE0 and the debugger (read and write access). The test fills CRAM2 and attempts to write to CRAM3. This latter access is blocked by the CRG and leads to an ECM error. When the ECM error occurs, the ERROROUT LED34 lights up red for less than one second.

8.1.4.2 Test 2 – Peripheral Bus Guard – Processor Element 1

Test 2 is also executed by PE1. For this test, PBG and ECM are initialized to protect TAUD0 from being accessed by PE1 and to intercept the associated ECM error. The test attempts to change the TAUD0 comparison value, which would accelerate the flashing frequency of LED32. The protection mechanisms are configured so that only PE0 and the debugger can access the TAUD0 registers (read and write access). This access is blocked by the CRG and leads to an ECM error. When the ECM error occurs, the ERROROUT LED34 lights up red for less than one second.

8.1.4.3 Test 3 – Cluster RAM Guard – Processor Element 0

Test 3 is executed by PE0. The guards and the ECM are configured so that only PE1 and the debugger can access CRAM2 (read and write access) and the error can be caught by the ECM. CRAM3 can only be accessed by PE0 and the debugger (read and write access). In this test, the processor element fills CRAM3 and tries to access CRAM2. This latter access is blocked by the CRG and results in an ECM error. When the ECM error occurs, the ERROROUT LED34 lights up red for less than one second.

8.2 Port and clock setup for TAUD, MSPI and INTP

This section describes the connection settings for the TAUD, the INTP and the MSPI used in the Guard application.

8.2.1 Port Setup - External Input / Output

Table 8-3 lists the external input / output pins of the TAUD, MSPI, ECM and INTP modules on the RH850/U2A16 device, which are used for this Guard application.

Table 8-3 External Input / Output and the Corresponding Pin Function of TAUD, MSPI and ECM

Signal Name	Description	Corresponding Pin	Port Number	Port Function	I/O Direction
TAUD0TTOUT2	Channel 2 output	P2	2	Alternative function 6*1	Output
INTP05	External pin INTP05		7	Alternative function 2*1	Input
MSPI3SO	Output serial data signal	P22	0	Alternative function 4*1	Output
MSPI3SI	Input serial data signal		1		Input
MSPI3SC	Serial clock signal		4		Input / Output
P22_2	Latch Enable for LED module		2	Port Mode	Output
$\overline{\text{ERROROUT_M}}$	Error output master signal	$\overline{\text{ERROROUT_M}}$	-	-	Output
P10_10	Output Enable for LED module	P10	10	Port Mode	Output

Notes: 1. The corresponding pins operate in alternative mode software I/O control, the pins operate as alternative functions, and the input / output direction is selected by setting PMn.PMn_m bit to 1 / 0. To configure the operation mode, the PMCn, PFCn, PFCEn and PFACEn registers should be configured. Table 8-4 provides an overview of the register settings.
For detailed information, please refer to the HW User's Manual *R01UH0864EJxxxx Section 2 'Pin Functions'*.

Table 8-4 Alternative Peripheral Function Selection

Alternative-Function	Bit Name			
	PMCn_m	PFCAEn_m	PFCEn_m	PFCn_m
Alternative function 1	1	0	0	0
Alternative function 2				1
Alternative function 3			1	0
Alternative function 4				1
Alternative function 5		1	0	0
Alternative function 6				1
Alternative function 7			1	0
Alternative function 8				1

8.2.2 Port Write Protection

Before the port registers described above can be written to, appropriate write processes must be enabled by the port key code protection register PKCPROT and by setting the port specific bit in the port write enable register PWE.

For details, please refer to the RH850/U2A-EVA Hardware User's Manual *R01UH0864EJxxxx Section 2 'Pin Functions'*.

8.2.3 Clock Setup for TAUD and MSPI Module

The supply clock for the MSPI and TAUD module is CLK_HSB with a maximum frequency of 80 MHz.

For detailed information on the clock supply and the corresponding settings, refer to the RH850/U2A-EVA Hardware User's Manual R01UH0864EJxxxx or the RH850/U2A-EVA Application Note of Clock Supply R01AN4593EDxxxx_RH850_U2A.

To enable the clock connections to the peripherals, the MSR_TAUD.BIT.MS_TAUD_0 and MSR_MSPI.BIT.MS_MSPI_3 bits should be set to 0.

Before writing to this register, write protection must be disabled by setting A5A5A501H in the key code protection register MSRKCPROT. If the write protection is required in further accesses, please write A5A5A500H to the MSRKCPROT register after the clock setup.

8.3 Sample Software

The corresponding software example of this document can be found in the attachment file: *r20an6094ed0100_rh850_u2a_sw.7z*.

It is based on the software tools and hardware setups described in section 4.

The target device of the software is the RH850/U2A16 device R7F702300BFABA-C, the device files in the package DF-RH850U2A-EE_E130b.zip are used for the sample SW. Please notice that the latest device files are always recommended for the SW development.

The example SW is executed by core 0 and core 1. The SW contains the configuration of the ECM, CRG2 & CRG3 and PBG, the three guard tests, the error handling, port configuration, clock configuration, TAUD configuration, MSPI configuration, external interrupt configuration, as well as the main functions of core 0 and core 1.

8.3.1 main_pe0.c

The location of this file is in folder:
… \core0\src

- Main function: *int main(int argc, char *argv[])*
  - Enable required interrupts
  - Print core startup message
  - Initialize required CRAM0 pointer
  - Supply module clocks
  - Start flashing user LED

8.3.2 main_pe1.c

The location of this file is in folder:
… \core1\src

- Main function *int main(int argc, char *argv[])*
  - Enable required interrupts
  - Print core startup message
  - Port setup and supply module clocks
  - Initialize required CRAM0 pointer
  - Module initializations and setups

8.3.3 r_mspi.c and r_mspi.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\mspi

r_mspi.c contains the configuration for the basic operations for the LED driver operations of the Guard application:

- MSPI initialization & associated port setup: *R_MSPI_Init()*
- Enable MSPI, MSPI interrupts, Configure P10_10: *R_MSPI30_Start()*
- Send bitmask to LED driver and accept it as LED mask: *R_MSPI30_Send(uint16_t* data)*

r_mspi.h is the header file of r_mspi.c, in which the variables and functions are defined.

8.3.4 r_led.c and r_led.h

The location of this file is in folder:
… \shared\additional_sources\modules\led

r_led.c contains the configuration for the basic operations for the LED driver operation of the Guard application based on the MSPI functions:

- Initialize ports for LED driver latch and output enable: *R_LED_RingGpioInit()*
- Initialize the LED driver and output: *R_LED_RingInit()*
- Turn off all LEDs: *R_LED_RingOutputDisable()*
- Turn on last LED Bitmask: *R_LED_RingOutputEnable()*
- Update LED Bitmask *R_LED_RingUpdate(uint16_t* LedBitMask)*

r_led.h is the header file of r_led.c, in which the LED functions are defined.

8.3.5 r_tau.c and r_tau.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\tau

- Configuration and initialization of TAUD0C0, TAUD0C2: *R_TAUD_Init()*
  - TAUD0C0: ~0.5 sec interval timer
  - TAUD0C2: ~1 sec interval timer with real time output
- Start trigger TAUD0C0: *R_TAUDC0_Start()*
- Start trigger TAUD0C2: *R_TAUDC2_Start()*
- Stop trigger TAUD0C0: *R_TAUDC0_Stop()*
- Stop trigger TAUD0C2: *R_TAUDC2_Stop()*
- TAUD0C0 interrupt routine: *taud0c0_interrupt()*
  - Update LED ring – countdown
  - Execute PE1 tests

r_tau.h is the header file of r_tau.c, where the functions for configuration and control of the TAUD are defined.

8.3.6 r_intp.c and r_intp.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\intp

r_intp.c contains the configuration of the external interrupts that serve the test selection process:

- Configuration of external interrupt 4 and required port (SW17): *R_INTP4_Init()*
- Configuration of external interrupt 5 and required port (Encoder push button): *R_INTP5_Init()*
- External Interrupt 7 - interrupt routine: *intp5_routine()*
  - LED ring – countdown start
  - LED ring – Test selection
  - Execute test 3, if selected

r_intp.h is the header file of r_intp.c where the functions for the configuration are defined.

8.3.7 r_pbg.c and r_pbg.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\pbg

r_pbg.c contains the configuration of the PBG that protects the TAUD0 from PE1 accesses (used for test 2):

- Configuration of PBG for TAUD0 protection (channel 51, 1) *R_PE0_PBG_Init()*
- Stop PBG protection, Disable Guard setting (channel 51, 1) *R_PE0_PBG_Stop()*

r_pbg.h is the header file of r_pbg.c, where the required functions are defined.

8.3.8 r_crg.c and r_crg.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\crg

r_crg.c contains the configuration of the CRG2 and CRG3 as well as test functions for test 1 and test 3:

- Configuration of CRG2 only for MAU and PE1 access: *R_PE1_CRG_Init()*
- Configuration of CRG3 only for MAU and PE0 access: *R_PE0_CRG_Init()*
- PE1-Test: Fill CRAM2 (allowed), write twice to CRAM3 (denied): *R_PE1_Test()*
- PE0-Test: Fill CRAM3 (allowed), write twice to CRAM2 (denied): *R_PE0_Test()*

r_crg.h is the header file of r_crg.c, in which the required functions are defined.

8.3.9 r_ecm.c and r_ecm.h

The location of this file is in folder:
… \shared\additional_sources\peripherals\ecm

r_ecm.c contains functions for the configuration of the ECM, error detection and error recovery as well as the application related LED ring error output:

- Configuration of the ECM: *R_ECM_Init()*
  - Masking of possible error sources
  - Reset of ERROROUT_M/C
  - Release of PE0 / PE1 / CRG / PBG errors
  - Error routines handled by PE1
- Test 2 Error clearing: *R_ECM_Clear_PBGError()*
  - Clear PBG & PE1 errors
  - Reset ERROROUT_M/C:
- Test 1 & Test 3 Error clearing: *R_ECM_Clear_CRGError()*
  - Clear CRG & PE0 & PE1 error
  - Reset ERROROUT_M/C
- Error identification by test number: *uint8_t R_ECM_GetErrorFlag()*
  - Returns test number or 4 if none of the known errors occured
- ECM Error Interrupt: *ecm_interrupt()*
  - Error detection
  - Update LED Bitmask
  - Reset error output

r_ecm.h is the header file of r_ecm.c, where the functions for configuration and error handling of the ECM are defined.

Revision History

Rev.	Date	Description	
		Section	Summary
1.00	Jan.02.2024		First release

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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/.

© 2021 Renesas Electronics Corporation. All rights reserved.