The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205374

FR, MB91460, EDSU/MPU

This application note describes the functionality of the Embedded Debug Support Unit (EDSU) and Memory Protection Unit (MPU) and gives some examples. The EDSU is mostly used for debugging while the MPU is used to monitor if there is any memory protection violation.

Contents

1 Introduction 1
1.1 Key Features 1
2 Bit Search 2
2.1 Introduction 2
2.2 Block Diagram 2
2.3 Registers 3
2.4 Operation 8
3 DSU/MPU Examples 21
3.1 Setting the interrupt vector 21
3.2 Setting the Memory Protection 22
4 Additional Information 23
Document History 24

1 Introduction

This application note describes the functionality of the Embedded Debug Support Unit (EDSU) and Memory Protection Unit (MPU) and gives some examples. The EDSU is mostly used for debugging while the MPU is used to monitor if there is any memory protection violation.

1.1 Key Features

- Up to 8 number of Comparator Groups
- One Comparator group can configure maximum of 4 distinct breakpoints such as…
  - 4 Instruction Address Breaks
  - 4 Operand Address Breaks
  - 2 Operand Address Breaks and 2 Instruction Address Breaks
  - 2 Operand Address Breaks and 2 Data Value Breaks
- 2 Masks and 2 Range functions possible
- Separate interrupt vectors for Instruction Break, Operand Break and Memory Protection Exception
- All registers configurable only in Supervisor mode
- Tool NMI configurable by interrupt on specific UART or CAN channel
- CPU and DMA access can be differentiated
- Read/write//execute permissions can be set for user and supervisor mode

2 Bit Search

The Basic Functionality of EDSU AND MPU.

2.1 Introduction

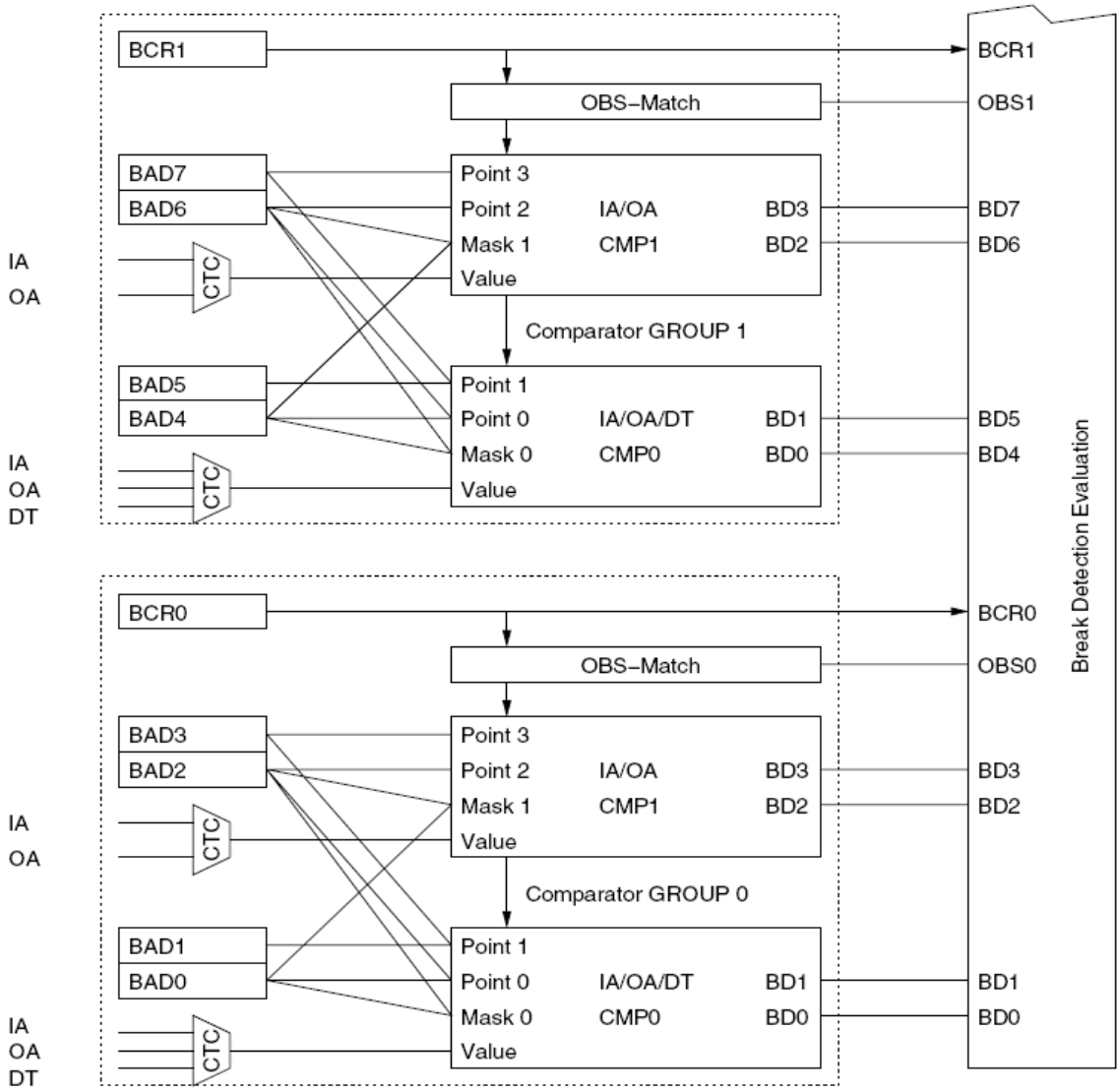
The EDSU allows the user to watch data exchange on the internal bus and detect address hits. Whereas the MPU can be used for supervising the memory in operating systems or for debugging user software on-chip.

It contains up to eight independent comparator groups, each group has two comparators, CMP0 and CMP1. Each comparator has two point registers which can be used to define two independent points or a range or mask. The two comparator masks can be interchangeably used while the range feature is used or while to independent points are used. The EDSU is also used by the Accemic Debugger.

2.2 Block Diagram

Figure 1 shows the internal block diagram of the EDSU.

Figure 1. EDSU Block Diagram



Also refer section 2.3.7 for the detailed understanding of the functionality of points and masks along with the comparators.

2.3 Registers

It should be noted that all the registers of the EDSU and MPU can be written ONLY in the supervisor mode. If they are attempted to be written in the user mode then the Memory Protection Exception would be generated. One can enter in the supervisor mode either by setting the `SV` bit of `CCR` (`ORCCR #0x40`) or by executing the `INT #5` instruction.

2.3.1 EDSU Control Register (BCTRL)

BCTRL is the default permission register. It defines the **lowest priority access permissions** for the whole memory and I/O address range of the MCU. Lowest priority means, that the default permission takes effect for all address regions, which are NOT covered by any dedicated channel configuration, operating in MPU mode. If the MPU access permissions are not used and configured by BCRn registers at all then BCTRL permissions would be used. Other than the default permissions it also configures filer options, emulation mode etc.

Table 1. BCTRL

Bit No.	Name	Explanation	Value	Operation
15	SR	Supervisor default Read permission	0	Supervisor is not permitted to read data
			1	Supervisor is permitted to read data
14	SW	Supervisor default Write permission	0	Supervisor is not permitted to write data
			1	Supervisor is permitted to write data
13	SX	Supervisor default Execute permission	0	Supervisor is not permitted to execute code
			1	Supervisor is permitted to execute code
12	UR	User default Read permission	0	User is not permitted to read data
			1	User is permitted to read data
11	UW	User default Write permission	0	User is not permitted to write data
			1	User is permitted to write data
10	UX	User default Execute permission	0	User is not permitted to execute code
			1	User is permitted to execute code
9	FCPU	Filter CPU access	0	Trigger on CPU access
			1	Do not trigger on CPU access
8	FDMA	Filter DMA access	0	Trigger on DMA access
			1	Do not trigger on DMA access
7	EEEM	Enable Emulation Mode	0	Disable Emulation Mode
			1	Enable Emulation Mode
6	PFD	Phantom Filter Disable	0	Instruction break detection uses phantom filter
			1	Phantom Filter disabled
5-4	SINT1-SINT0	Select Resource Interrupt source	0,0	Please refer the hardware manual for detailed description
			0,1	
			1,0	
			1,1	
3	EINT1	Enable Extended Interrupt 1	0	Disable extended interrupt source 1
			1	Enable extended interrupt source 1
2	EINT0	Enable Extended Interrupt 0	0	Disable extended interrupt source 0
			1	Enable extended interrupt source 0

1	EINTT	Enable Interrupt on Transmit	0	Disable transmit interrupt source channels 0 to 3
			1	Enable transmit interrupt source channels 0 to 3
0	EINTR	Enable Interrupt on Receive	0	Disable receive interrupt source channels 0 to 3
			1	Enable receive interrupt source channels 0 to 3

2.3.2 EDSU Status Register (BSTAT)

This register reflects the status of EDSU and MPU functionality.

Table 2. BSTAT

Bit No.	Name	Explanation	Value	Operation
15 -11	IDX4 – IDX0	Channel Index Indication of MPUPV Trigger	0-15	Points to the channel number of the last protection violation
			16	The last protection violation was caused by the violation of the default permissions set by BCTRL
10	CDMA	Capture DMA Indication	0	The operand access was executed by the CPU
			1	The operand access was executed by the DMAC
9-8	CSZ1 – CSZ0	Capture Operand Size	0,0	The operand has a bit size of 8
			0,1	The operand has a bit size of 16
			1,0	The operand has a bit size of 32
			1,1	Reserved
7-6	CRW1 – CRW0	Capture Operand Access Type	0,0	The operand has been read
			0,1	The operand has been read by read-modify-write
			1,0	The operand has been written
			1,1	No operand access
5	PV	Protection Violation Detection	0	No protection violation
			1	Protection violation
4	RST	Operation Initialization Reset (RST) Detection	0	Operation Reset was not triggered since last BSTAT read or clear
			1	Operation Reset was triggered since last BSTAT read or clear
3	INT1	Interrupt on extended source 1	0	No interrupt on extended source channel 1
			1	Interrupt on extended source channel 1
2	INT1	Interrupt on extended source 0	0	No interrupt on extended source channel 0
			1	Interrupt on extended source channel 0
1	INTT	Interrupt on Transmit source	0	No interrupt on transmit source
			1	Interrupt on transmit source
0	INTR	Interrupt on Receive source	0	No interrupt on receive source
			1	Interrupt on receive source

2.3.3 EDSU Instruction Address Capture Register (BIAC)

This register captures the address of the instruction (IA), which has caused the protection violation or the operand/ data value break. This register could be read only.

2.3.4 EDSU Operand Address Capture Register (BOAC)

This register captures the address of the operand access (OA), which has caused the protection violation or the operand/data value break. This register could be read only.

2.3.5 EDSU Break Detection Interrupt Request Register (BIRQ)

This register contains the interrupt request bits for the detected break conditions.

Table 3. BIRQ

Bit No.	Name	Explanation	Value	Operation
31-0	BD31–BD0	Break detection bit	0	Break condition not detected
			1	Break condition detected on channel corresponding to the bit position

Each bit is corresponding to the appropriate point register BAD31 to BAD0. The setting of these bits is very much dependent on whether the two neighboring registers in each comparator (CMP1 or CMP0) are configured as two independent points or a range or a point with a mask or two independent points with a mask or a range with a mask.

The following table explains the two independent points or range scenario.

Table 4. BD bit Eetting for Point or Range Match

Sr. No.	BD[n+1]	BD[n]	Compare value
1	0	0	No Match
2	0	1	Match on point Compare Value = BD[n]
3	1	0	Match on point Compare Value = BD[n+1]
4	1	1	Match on point BD[n+1] < Compare Value < BD[n]
n=0,2,4…30			

2.3.6 EDSU Channel Configuration Register (BCRn)

Each comparator group has one BCRn register (e.g. BCR0 configures BAD0 to BAD3). It provides setting for EDSU and MPU functionality.

Table 5. BCRn-I

Bit No.	Name	Explanation	Value	Operation
31 -24	–	-	–	-
23	SRX1	Supervisor Read/Execute permission for range 1	0	Supervisor has no execute/read permission on address range 1
			1	Supervisor has execute/read permission on address range 1
22	SW1	Supervisor Write permission for range 1	0	Supervisor has no write permission on address range 1
			1	Supervisor has write permission on address range 1
21	SRX0	Supervisor Read/Execute permission for range 0	0	Supervisor has no execute/read permission on address range 0
			1	Supervisor has execute/read permission on address range 0
20	SW0	Supervisor Write permission for range 0	0	Supervisor has no write permission on address range 0
			1	Supervisor has write permission on address range 0
19	URX1	User Read/Execute permission for range 1	0	User has no execute/read permission on address range 1
			1	User has execute/read permission on address range 1
18	UW1	User Write permission for range 1	0	User has no write permission on address range 1
			1	User has write permission on address range 1
17	URX0	User Read/Execute permission for range 0	0	User has no execute/read permission on address range 0
			1	User has execute/read permission on address range 0
16	UW0	User Write permission for range 0	0	User has no write permission on address range 0
			1	User has write permission on address range 0
15	MPE	Memory Protection Enable	0	The group of channels operates as debug interface and defines breakpoints
			1	The group of channels operates in memory protection mode
14	COMB	Channel Combination Enable	0	No combination between channels
			1	Combination between channels is effective

Table 6. BCRn-II

Bit No.	Name	Explanation	Value	Operation	
13 - 12	CTC1 – CTC0	Comparator Type Config		Break Function	MPU Function
			0,0	4 instruction break points	2 regions for code protection
			0,1	4 operand break points	2 regions for data protection
			1,0	2 instruction break points + 2 operand break points	1 region for code protection and 1 region for data protection or 1 region for combined code and data protection
			1,1	2 operand break points + 2 data value breaks	N. A.
11 - 10	OBS1 – OBS0	Operand Break Size	0,0	Byte	
			0,1	Half-word	
			1,0	Word	
			1,1	All of above	
9-8	OBS1– OBS0	Operand Break Size	0,0	Read	
			0,1	Read-modify-write	
			1,0	Write	
			1,1	All of above	
7	EP3	Enable break Point 3	0	Break point 3 register is disabled	
			1	Break point 3 register is enabled	
6	EP2	Enable break Point 2	0	Break point 2 register is disabled	
			1	Break point 2 register is enabled	
5	EP1	Enable break Point 1	0	Break point 1 register is disabled	
			1	Break point 1 register is enabled	
4	EP0	Enable break Point 0	0	Break point 0 register is disabled	
			1	Break point 0 register is enabled	
3	EM1	Enable Mask for CMP1	0	Mask function for CMP1 is disabled	
			1	Mask function for CMP1 is enabled	
2	EM0	Enable Mask for CMP0	0	Mask function for CMP0 is disabled	
			1	Mask function for CMP0 is disabled	
1	ER1	Enable Range for CMP1	0	Range detection for CMP1 (channels 2-3) is disabled	
			1	Range detection for CMP1 (channels 2-3) is enabled	
0	ER0	Enable Range for CMP0	0	Range detection for CMP0 (channels 0-1) is disabled	
			1	Range detection for CMP0 (channels 0-1) is enabled	

2.3.7 Break Address/Data Registers

2.3.7.1 BAD4n+0

As shown in the below block diagram, BAD0 register can be used as an independent point for CMP0 or the mask for CMP0 or CMP1 (depending mask and range settings) or as a lower limit of range for CMP0 (the upper limit would be defined by BAD1). In the special case of MPE=1 and COMB=1, BAD0 is not used for the point definition, rather it would be used as mask for CMP1 as well as CMP0 and CMP0 gets its point configuration then from BAD2.

2.3.7.2 BAD4n+1

BAD1 register can be used as an independent point or as an upper limit of range for CMP0 (the lower limit would be defined by BAD0). In the special case of MPE=1 and COMB=1, BAD1 is not used for the point definition AT ALL, CMP0 gets its point configuration then from BAD3.

2.3.7.3 BAD4n+2

BAD2 register can be used as an independent point for CMP1 or the mask for CMP0 or CMP1 (depending mask and range settings) or as a lower limit of range for CMP1 (the upper limit would be defined by BAD3). In the special case of MPE=1 and COMB=1, CMP0 gets its point configuration then from BAD2.

2.3.7.4 BAD4n+3

BAD3 register can be used as an independent point or as an upper limit of range for CMP1 (the lower limit would be defined by BAD0). In the special case of MPE=1 and COMB=1, CMP0 gets its point configuration then from BAD3.

2.4 Operation

The configurations discussed in the subsequent sections are to give a reasonable idea to the programmer that in what different ways the EDSU/MPU can be configured. However, it may be possible to have more ways to configure EDSU/MPU than those discussed here.

2.4.1 Instruction Address Break

Instruction address break occurs when an instruction is fetched at the address specified by BADn registers.

To precisely determine the instruction address where a break occurs, use the PC value saved on the stack during entry to the instruction break interrupt service routine.

Table 7. Instruction Address Break-I

1.	**BCR0**												
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	0	0	0	x	x	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			0xFFFFAAAA			0x0000AAAA			0x00005555			
	This configuration would set 4 instruction address breakpoints at the 4 addresses described above. If the instruction at the address 0x12345678 is fetched then, bit BD0 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0xFFFFAAAA is fetched then, bit BD1 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0x0000AAAA is fetched then, bit BD2 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0x00005555 is fetched then, bit BD3 of BIRQ register would be set and Instruction Break exception ISR would be executed.												
2.	**BCR0**												
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	0	0	0	x	x	0	1	0	1	0	0	1	1
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x000000FF			0x12345678			0x0000FF00			0x0000AAAA			
	This configuration would set 2 instruction address breakpoints with mask If any instruction between the address range 0x12345600 - 0x123456FF is fetched then, bit BD1 of BIRQ register would be set and Instruction Break exception ISR would be executed. If any instruction between the address range 0x000000AA - 0x0000FFAA is fetched then, bit BD3 of BIRQ register would be set and Instruction Break exception ISR would be executed.												

Table 8. Instruction Address Break-II

3.	**BCR0**												
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	0	0	0	x	x	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set 2 instruction address breakpoints with range If the instruction at the address 0x12340000 is fetched then, bit BD0 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0x12348000 is fetched then, bit BD1 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0x80000000 is fetched then, bit BD2 of BIRQ register would be set and Instruction Break exception ISR would be executed.												

If the instruction at the address 0x81000000 is fetched then, bit BD3 of BIRQ register would be set and Instruction Break exception ISR would be executed.

If any instruction between the address range 0x12340000 - 0x12348000 is fetched then, bits BD0 and BD1 of BIRQ register would be set and Instruction Break exception ISR would be executed.

If any instruction between the address range 0x80000000 - 0x81000000 is fetched then, bits BD2 and BD3 of BIRQ register would be set and Instruction Break exception ISR would be executed.

4.

BCR0												
CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
0	0	0	x	x	1	1	0	1	1	0	1	0

BAD0	**BAD1**	**BAD2**	**BAD3**
0x12340000	0x12348000	0x000F0000	0x81000000

This configuration would set 1 instruction address breakpoint with range and mask and 1 instruction address breakpoint.

If the instruction at the address 0x12340000 is fetched then, bit BD0 of BIRQ register would be set and Instruction Break exception ISR would be executed.

If the instruction at the address 0x12348000 is fetched then, bit BD1 of BIRQ register would be set and Instruction Break exception ISR would be executed.

If the instruction at the address 0x81000000 is fetched then, bit BD3 of BIRQ register would be set and Instruction Break exception ISR would be executed.

If any instruction between the below address ranges is fetched

0x12300000 - 0x12308000,
0x12310000 - 0x12318000,
0x12320000 - 0x12328000,
0x12330000 - 0x12338000,
0x12340000 - 0x12348000,
0x12350000 - 0x12358000,
0x12360000 - 0x12368000,
0x12370000 - 0x12378000,
0x12380000 - 0x12388000,
0x12390000 - 0x12398000,
0x123A0000 - 0x123A8000,
0x123B0000 - 0x123B8000,
0x123C0000 - 0x123C8000,
0x123D0000 - 0x123D8000,
0x123E0000 - 0x123E8000,
0x123F0000 - 0x123F8000,

then, bits BD1 and BD0 of BIRQ register would be set and Instruction Break exception ISR would be executed.

2.4.2 Operand Address Break

Operand address break occurs when an operand is accessed at the address specified by BADn registers.

Table 9. Operand Address Break-I

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	1	0	0	0	3	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			0xFFFFAAAA			0x0000AAAA			0x00005554			
	This configuration would set 4 operand address breakpoints at the 4 addresses described above. If the operand at the address 0x12345678 is byte accessed then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0xFFFFAAAA is byte accessed then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x0000AAAA is byte accessed then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x00005554 is byte accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												
	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	1	0	0	2/3*	0	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			**0xFFFFAAAA**			**0x0000AAAA**			**0x00005555**			
	This configuration would set 4 operand address breakpoints at the 4 addresses described above. If the operand at the address range 0x12345678 to 0x1234567A is read then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address range 0xFFFFAAAA to 0xFFFFAAAD is read then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address range 0x0000AAAA to 0x0000AAAD is read then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address range 0x00005558 is read then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												
	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	1	0	0	3	3	0	1	0	1	0	0	1	1
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x000000FF			0x12345678			0x0000FF00			0x0000AAAA			
	This configuration would set 2 operand address breakpoints with mask. If any operand between the address range 0x12345600 - 0x123456FF is accessed then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If any operand between the address range 0x000000AA - 0x0000FFAA is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

Table 10. Operand Address Break-II

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	1	0	0	0	3	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set 2 operand address breakpoints with range If the operand at the address 0x12340000 is byte accessed then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x12348000 is byte accessed then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x80000000 is byte accessed then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is byte accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If any operand between the address range 0x12340000 - 0x12348000 is byte accessed then, bits BD0 and BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If any operand between the address range 0x80000000 - 0x81000000 is byte accessed then, bits BD2 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												
	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	1	0	0	3	3	1	1	0	1	1	0	1	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x000F0000			0x81000000			
	This configuration would set 1 operand address breakpoint with range and mask and 1 operand address breakpoint. If the operand at the address 0x12340000 is accessed then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x12348000 is accessed then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is fetched then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If any Operand between the below address ranges is fetched 0x12300000 - 0x12308000, 0x12310000 - 0x12318000, 0x12320000 - 0x12328000, 0x12330000 - 0x12338000, 0x12340000 - 0x12348000, 0x12350000 - 0x12358000, 0x12360000 - 0x12368000, 0x12370000 - 0x12378000, 0x12380000 - 0x12388000, 0x12390000 - 0x12398000, 0x123A0000 - 0x123A8000, 0x123B0000 - 0x123B8000, 0x123C0000 - 0x123C8000, 0x123D0000 - 0x123D8000,												

	0x123E0000 - 0x123E8000, 0x123F0000 - 0x123F8000, then, bits BD1 and BD0 of BIRQ register would be set and Operand Break exception ISR would be executed.

Table 11. Operand Address Break-III

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	2	0	0	0	3	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			0xFFFFAAAA			0x0000AAAA			0x00005555			
	This configuration would set 2 operand address breakpoints with 2 instruction address breakpoints. If the instruction at the address 0x12345678 is fetched then, bit BD0 of BIRQ register would be set and Instruction Break exception ISR would be executed. If the instruction at the address 0xFFFFAAAA is fetched then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x0000AAAA is accessed then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x00005555 is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

2.4.3 Data Value Break

The data value break occurs when the specified data is read or written at an address specified by BADn registers. This data access can be by CPU or DMA. In order to determine what exactly has caused the break FCPU and FDMA of BCTRL register can be used.

2.4.3.1 Operand Address Break and Data Value Break (COMB = 0)

Table 12. Operand Address Break and Data Value Break - I

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	0	0	2	3	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			0xFFFFAAAA			0x0000AAAA			0x00005554			
	This configuration would set 2 operand address breakpoints with 2 data value breakpoints. If the data 0x12345678 is appeared on the bus then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. If the data 0xFFFFAAAA is appeared on the bus then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x0000AAAA is accessed then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x00005554 is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	0	0	3	3	0	1	0	1	0	0	1	1
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x0000FFFF			0x12345678			0x0000FF00			0x0000AAAA			
	This configuration would set 1 operand address breakpoint with mask and 1 data value breakpoint with mask. If the half-word data 0x1234 is appeared on the bus (at half-word aligned address) then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. If any operand between the address range 0x000000AA - 0x0000FFAA is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

Table 13. Operand Address Break and Data Value Break - II

1.	**BCR0**											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	0	0	2	3	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	– This configuration would set 1 operand address breakpoint with range and 1 data value breakpoint with range. – If the data 0x12340000 is appeared on the bus then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. – If the data 0x12348000 is appeared on the bus then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. – If the operand at the address 0x80000000 is accessed then, bit BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. – If the operand at the address 0x81000000 is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. – If any data between the range 0x12340000 - 0x12348000 is appeared on the bus then, bits BD0 and BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. – If any operand between the address range 0x80000000 - 0x81000000 is accessed then, bits BD2 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												
2.	**BCR0**											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	0	0	3	3	1	1	0	1	1	0	1	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x000F0000			0x81000000			
	– This configuration would set 1 operand address breakpoint and 1 data value breakpoint with range and mask. – If the data 0x12340000 is appeared on the bus then, bit BD0 of BIRQ register would be set and Operand Break exception ISR would be executed. – If the data 0x12348000 is appeared on the bus then, bit BD1 of BIRQ register would be set and Operand Break exception ISR would be executed. – If the operand at the address 0x81000000 is accessed then, bit BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. – If any data between the below address ranges is appeared on the bus – 0x12300000 - 0x12308000, – 0x12310000 - 0x12318000, – 0x12320000 - 0x12328000, – 0x12330000 - 0x12338000, – 0x12340000 - 0x12348000, – 0x12350000 - 0x12358000,												

	– 0x12360000 - 0x12368000, – 0x12370000 - 0x12378000, – 0x12380000 - 0x12388000, – 0x12390000 - 0x12398000, – 0x123A0000 - 0x123A8000, – 0x123B0000 - 0x123B8000, – 0x123C0000 - 0x123C8000, – 0x123D0000 - 0x123D8000, – 0x123E0000 - 0x123E8000, – 0x123F0000 - 0x123F8000, then, bits BD1 and BD0 of BIRQ register would be set and Operand Break exception ISR would be executed.

2.4.3.2 Operand Address Break with Data Value Break (COMB = 1)

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	1	0	2	2	1	1	1	1	0	0	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12345678			0xFFFFAAAA			0x0000AAAA			0x00005554			
	This configuration would set 2 operand address breakpoints with 2 data value breakpoints, combined. If the operand at the address 0x0000AAAA is written with the data 0x12345678 then, bit BD0 and BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x00005554 is written with the data 0x0000AAAA then, bit BD1 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												
	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	1	0	3	3	0	1	0	1	0	0	1	1
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x0000FFFF			0x12345678			0x0000FF00			0x0000AAAA			
	This configuration would set 1 operand address breakpoint with mask and 1 data value breakpoint with mask, combined. If any operand between the address range 0x000000AA - 0x0000FFAA is accessed with the half-word data 0x1234 then, bits BD1 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	8	1	0	2	0	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set 1 operand address breakpoint with range and 1 data value breakpoint with range, combined. If the operand at the address 0x80000000 is read with the data 0x12340000 then, bit BD0 and BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is read with the data 0x12348000 then, bit BD1 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x80000000 is read with the data 0x12348000 then, bit BD1 and BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is read with the data 0x12340000 then, bit BD0 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x80000000 is read with any data from 0x12340001 to 0x12347FFF then, bit BD0, BD1 and BD2 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is read with any data from 0x12340001 to 0x12347FFF then, bit BD0, BD1 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If any address from 0x80000004 to 0x80FFFFFC is read with the data 0x12340000 then, bit BD0, BD2 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If any address from 0x80000004 to 0x80FFFFFC is read with the data 0x12348000 then, bit BD1, BD2 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed.												

Table 14. Operand Address Break with Data Value Break - II

	BCR0											
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	3	0	0	3	2	1	1	0	1	1	0	1	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x000F0000			0x81000000			
	This configuration would set 1 operand address breakpoint and 1 data value breakpoint with range and mask, combined. If the operand at the address 0x81000000 is written with the data 0x12348000 then, bit BD1 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If the operand at the address 0x81000000 is written with the data 0x12340000 then, bit BD0 and BD3 of BIRQ register would be set and Operand Break exception ISR would be executed. If any data between the below address ranges 0x12300000 - 0x12308000, 0x12310000 - 0x12318000, 0x12320000 - 0x12328000, 0x12330000 - 0x12338000, 0x12340000 - 0x12348000, 0x12350000 - 0x12358000, 0x12360000 - 0x12368000, 0x12370000 - 0x12378000, 0x12380000 - 0x12388000,												

	0x12390000 - 0x12398000, 0x123A0000 - 0x123A8000, 0x123B0000 - 0x123B8000, 0x123C0000 - 0x123C8000, 0x123D0000 - 0x123D8000, 0x123E0000 - 0x123E8000, 0x123F0000 - 0x123F8000, is written at the address then, bits BD3, BD1 and BD0 of BIRQ register would be set and Operand Break exception ISR would be executed.

2.4.4 Memory Protection

The comparators as discussed in sections 2.1 and 2.2 are also used for the memory protection unit. With the usage of MPU it is possible to inhibit the CPU execute the code or access the data in supervisor as well as user mode.

In case of attempting to execute the code form the protected memory area, the Memory Protection exception ISR gets executed well BEFORE the particular instruction execution whereas in the case of attempting to access the data from the protected memory area, the Memory Protection exception ISR gets executed AFTER the particular data access (read/write/read-modify-write).

The BADn registers in this case are used to define the range or point with mask where the access needs to be inhibited.

It should be noted that in the Memory Protection exception ISR the contents of the BIAC and BOAC register should be read BEFORE clearing the PV bit of BSTAT register. Once the PV bit of BSTAT register is cleared the contents of these registers would become irrelevant.

2.4.4.1 Code Protection

Table 15. Code Protection – I

	BCR0											
	SRX1	**SW1**	**SRX0**	**SW0**	**URX1**	**UW1**	**URX0**	**UW0**				
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**				
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	0	0	1	3	3	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set code protection on two address ranges by supervisor and user If, in any of the supervisor or the user mode, the instruction at the address 0x12340000 is fetched then, bit BD0 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the instruction at the address 0x12380000 is fetched then, bit BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the instruction at the address 0x80000000 is fetched then, bit BD2 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the instruction at the address 0x81000000 is fetched then, bit BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, any instruction between the address range 0x12340000 - 0x12348000 is fetched then, bits BD0 and BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, any instruction between the address range 0x80000000 - 0x81000000 is fetched then, bits BD2 and BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed.											

Table 16. Code Protection – II

1.	BCR0												
	SRX1	SW1	SRX0	SW0	URX1	UW1	URX0	UW0					
	1	0	1	0	0	0	0	0					
	CTC	COMB	MPE	OBS	OBT	EP0	EP1	EP2	EP3	ER0	ER1	EM0	EM1
	0	0	1	3	3	1	1	1	1	1	1	0	0
	BAD0			BAD1			BAD2			BAD3			
	0x12340000			0x12348000			0x80000000			0x81000000			
	– This configuration would set code protection on two address ranges by supervisor. – If, the user tries to execute the instruction within the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR would be executed. – If, the supervisor tries to execute the instruction outside the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR may get executed depending upon the setting of BCTRL register.												
2.	BCR0												
	SRX1	SW1	SRX0	SW0	URX1	UW1	URX0	UW0					
	0	0	0	0	1	0	1	0					
	CTC	COMB	MPE	OBS	OBT	EP0	EP1	EP2	EP3	ER0	ER1	EM0	EM1
	0	0	1	3	3	1	1	1	1	1	1	0	0
	BAD0			BAD1			BAD2			BAD3			
	0x12340000			0x12348000			0x80000000			0x81000000			
	– This configuration would set code protection on two address ranges by user. – If, the supervisor tries to execute the instruction within the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR would be executed. – If, the user tries to execute the instruction outside the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR may get executed depending upon the setting of BCTRL register.												

2.4.4.2 Data Protection

Table 17. Code Protection - I

	BCR0											
	SRX1	SW1	SRX0	SW0	URX1	UW1	URX0	UW0				
	0	0	0	0	0	0	0	0				
	CTC	COMB	MPE	OBS	OBT	EP0	EP1	EP2	EP3	ER0	ER1	EM0	EM1
	1	0	1	3	3	1	1	1	1	1	1	0	0
	BAD0			BAD1			BAD2			BAD3			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set data protection on two address ranges by supervisor and user If, in any of the supervisor or the user mode, the operand at the address 0x12340000 is accessed then, bit BD0 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the operand at the address 0x12380000 is accessed then, bit BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the operand at the address 0x80000000 is accessed then, bit BD2 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the operand at the address 0x81000000 is accessed then, bit BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed.											

If, in any of the supervisor or the user mode, any operand between the address range 0x12340000 - 0x12348000 is accessed then, bits BD0 and BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed.

If, in any of the supervisor or the user mode, any operand between the address range 0x80000000 - 0x81000000 is accessed then, bits BD2 and BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed.

BCR0												
SRX1	**SW1**	**SRX0**	**SW0**	**URX1**	**UW1**	**URX0**	**UW0**					
1	1	1	1	0	0	0	0					
CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
1	0	1	3	3	1	1	1	1	1	1	0	0

BAD0	**BAD1**	**BAD2**	**BAD3**
0x12340000	0x12348000	0x80000000	0x81000000

This configuration would set data protection on two address ranges by supervisor.

If, the supervisor tries to access the operand outside the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR may get executed depending upon the setting of BCTRL register.

If, the user tries to access the operand within the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR may get executed depending upon the setting of BCTRL register.

Table 18. Code Protection - II

BCR0												
SRX1	**SW1**	**SRX0**	**SW0**	**URX1**	**UW1**	**URX0**	**UW0**					
0	0	0	0	1	1	1	1					
CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
1	0	1	3	3	1	1	1	1	1	1	0	0

BAD0	**BAD1**	**BAD2**	**BAD3**
0x12340000	0x12348000	0x80000000	0x81000000

This configuration would set data protection on two address ranges by user.

If, the supervisor tries to access the operand within the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR would be executed.

If, the user tries to access the operand outside the address range 0x12340000 - 0x12348000 or the address range 0x80000000 - 0x81000000, then Memory Protection exception ISR may get executed depending upon the setting of BCTRL register.

2.4.4.3 Code and Data Protection (COMB = 0)

Table 19. Code and Data Protection

	BCR0											
	SRX1	**SW1**	**SRX0**	**SW0**	**URX1**	**UW1**	**URX0**	**UW0**				
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**				
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	2	0	1	3	3	1	1	1	1	1	1	0	0
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x12340000			0x12348000			0x80000000			0x81000000			
	This configuration would set code protection on one address range and data protection on another address range by supervisor and user If, in any of the supervisor or the user mode, the instruction at the address 0x12340000 is fetched then, bit BD0 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the instruction at the address 0x12380000 is fetched then, bit BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the operand at the address 0x80000000 is accessed then, bit BD2 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, the operand at the address 0x81000000 is accessed then, bit BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, any instruction between the address range 0x12340000 - 0x12348000 is accessed then, bits BD0 and BD1 of BIRQ register would be set and Memory Protection exception ISR would be executed. If, in any of the supervisor or the user mode, any operand between the address range 0x80000000 - 0x81000000 is accessed then, bits BD2 and BD3 of BIRQ register would be set and Memory Protection exception ISR would be executed.												

2.4.4.4 Code with Data Protection (COMB = 1)

Table 20. Code along with Data Protection Combined

	BCR0												
	SRX1	**SW1**	**SRX0**	**SW0**	**URX1**	**UW1**	**URX0**	**UW0**					
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**					
	CTC	**COMB**	**MPE**	**OBS**	**OBT**	**EP0**	**EP1**	**EP2**	**EP3**	**ER0**	**ER1**	**EM0**	**EM1**
	2	1	1	3	3	1	0	1	1	0	1	0	1
	BAD0			**BAD1**			**BAD2**			**BAD3**			
	0x000F0000			-			0x12340000			0x123480000			
	This configuration would set code protection with data protection combined on one address range with mask by supervisor and user If, in any of the supervisor or the user mode, the instruction or the data at the below mentioned address ranges is fetched or accessed, 0x12300000 - 0x12308000, 0x12310000 - 0x12318000, 0x12320000 - 0x12328000, 0x12330000 - 0x12338000, 0x12340000 - 0x12348000,												

	0x12350000 - 0x12358000, 0x12360000 - 0x12368000, 0x12370000 - 0x12378000, 0x12380000 - 0x12388000, 0x12390000 - 0x12398000, 0x123A0000 - 0x123A8000, 0x123B0000 - 0x123B8000, 0x123C0000 - 0x123C8000, 0x123D0000 - 0x123D8000, 0x123E0000 - 0x123E8000, 0x123F0000 - 0x123F8000, then Memory Protection exception ISR would be executed.

3 DSU/MPU Examples

Examples for EDSU/MPU

3.1 Setting the interrupt vector

The following example demonstrates to setup the *vectors.c* file in order to use the EDSU and MPU interrupts.

```
#pragma intvect DefaultIRQHandler       2      /* System reserved  */
#pragma intvect DefaultIRQHandler       3      /* System reserved  */
#pragma intvect DefaultIRQHandler       4      /* System reserved  */
#pragma intvect EDSU_Interrupt5_ISR     5      /* INT #5 Instruction*/
#pragma intvect EDSU_MPUPV_ISR          6      /* Memory Protection exception */
#pragma intvect DefaultIRQHandler       7      /* Co-Processor fault trap */
#pragma intvect DefaultIRQHandler       8      /* Co-Processor error trap */
#pragma intvect DefaultIRQHandler       9      /* INTE Instruction       */
#pragma intvect EDSU_InstructionBreak_ISR      10     /* Instruction Break
          Exception */
#pragma intvect EDSU_OperandBreak_ISR          11     /* Operand Break trap */
#pragma intvect DefaultIRQHandler              12     /* Step Trace trap  */
#pragma intvect EDSU_ToolNMI_ISR               13     /* NMI Interrupt (Tool) */
#pragma intvect DefaultIRQHandler              14     /* Undefined Instruction*/
```

3.2 Setting the Memory Protection

The below software example demonstrates how to configure the MPU for the data protection. Here the memory protection exception would occur when the main function would try to access the elements from 25 onwards of the `data_hit[]` array. In the ISR the bit 0 of port 16 is toggled.

```
#define LENGTH 50
unsigned int data_hit[LENGTH];

void InitEDSU (void)
{
    CSCFG_EDSUEN = 1;       // Enable EDSU in the Clock Source Configuration
// Register
    while(!CSCFG_EDSUEN) HWWD_CL = 0;    // Wait until the bit is set
BCR0_MPE = 1;    // Enable memory protection
BCR0_COMB = 0;   // No Combination
BCR0_CTC = 1;    // 2 regions for data protection
BCR0_OBS = 3;    // Break on all operand sizes
BCR0_OBT = 3;    // Break on all operand access types
BCR0_EP1 = 1;    // Enable BAD0 as point
BCR0_EP0 = 1;    // Enable BAD1 as point
BCR0_ER0 = 1;    // Enable range BAD0-BAD1
BAD0 = &data_hit[((LENGTH/2)-1)];];     // Start of the data region where
       // access is allowed
BAD1 = &data_hit[LENGTH-1];       // End of the data region where access is
       // allowed
}

__interrupt void EDSU_MPUPV_ISR(void)
{
    if( BIRQ_BD0 || BIRQ_BD1 )
    {
        BIRQ_BD0 = 0;             // Clear Pending Interrupt
        BIRQ_BD1 = 0;             // Clear Pending Interrupt
        PDR16_D0 = ~1;      // Toggle the pin
       BSTAT_PV = 0;        // Clear MPU pending interrupt
    }
}
void InitLeds(void)
{
        DDR16 = 0xff;             // port as output
        PFR16 = 0x00;             // use GPIO function
        PDR16 = 0x00;             // all LEDs off
 }
```

```
void main(void)
{
       __EI();           // Enable interrupts

       __set_il(31);     // Allow all levels
       InitIrqLevels();  // Init interrupts

       PORTEN = 0x3;     // Enable I/O Ports
                         // This feature is not supported by MB91V460A
                         // For all other devices the I/O Ports must be enabled
       InitLeds();       // Initialize led port

       #pragma asm
#0x40   // Switch to Supervisor mode to configure EDSU
       // registers
       #pragma endasm

       InitEDSU ();        // Initialize EDSU

       #pragma asm
        ANDCCR #0xBF
       #pragma endasm

       while(1)            // endless loop
       {
        HWWD_CL = 0;
        for (i=0; i<LENGTH; i++)
       {
       data_hit = i;
       }
        }
}
```

4 Additional Information

Information about CYPRESS Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

Document History

Document Title: AN205374 - FR, MB91460, EDSU/MPU

Document Number:002-05374

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	06/26/2008	First Version; MPi
*A	5132691	NOFL	02/10/2016	Converted Spansion Application Note “MCU-AN-300081-E-V10” to Cypress format
*B	5843073	AESATP12	08/03/2017	Updated logo and copyright.
*C	6070279	NOFL	02/13/2018	Sunset Review Migrated to new template Updated links

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2008-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.