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.

AN205373

FR, MB91460, Bit Search

This application note describes the functionality of the Bit Search and gives some examples.

Contents

1 Introduction

This application note describes the functionality of the Bit Search and gives some examples.

1.1 Key Features

- Detects the first ‘0’ position
- Detects the first ‘1’ position
- Detects the first position where data changes from ‘0’ to ‘1’ or vice versa
- Previous bit search result can be restored

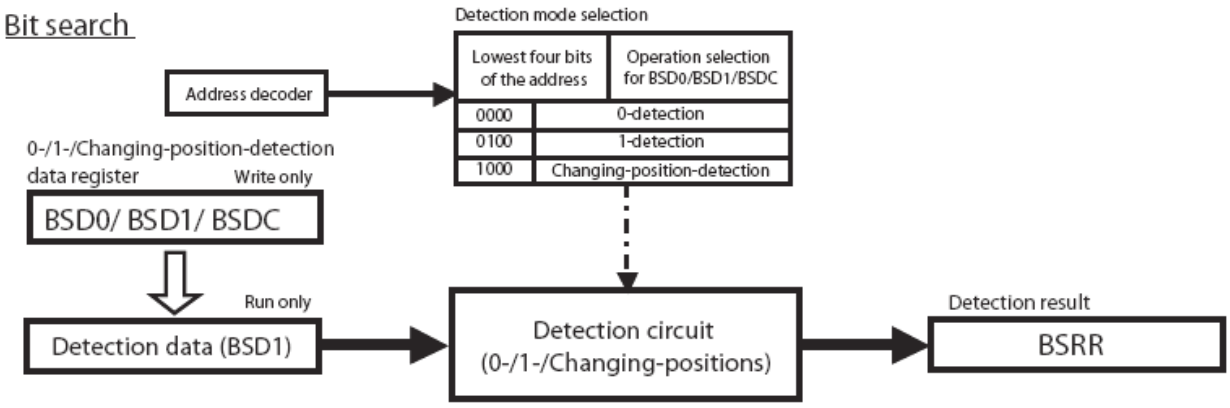
2 Bit Search

The basic functionality of Bit Search

2.1 Block Diagram

Figure 1 shows the internal block diagram of the Bit Search unit.

Figure 1. Bit Search Block Diagram



2.2 Registers

2.2.1 0 Detection Register (BSD0)

If the first '0' position needs to be detected in a word then it should be loaded in to `BSD0` register.

2.2.2 1 Detection Register (BSD1)

If the first '1' position needs to be detected in a word then it should be loaded in to `BSD1` register.

It should be noted that this register can be read as well as written. The reading of this register can be used restore previous bit search result.

2.2.3 Changing Position Detection Data Register (BSDC)

If the first position where data changes from '0' to '1' or vice versa needs to be detected in a word then it should be loaded in to the BSDC register.

2.2.4 Detection Result Register (BSRR)

The BSRR register contains the latest detection result for the data written in either BSD0 or BSD1 or BSDC. The type of detection result cannot be determined i.e. the information related to whether the result is related to 0 detection or 1 detection or changing position detection cannot be determined from BSRR.

2.3 Operation

The detection result in BSRR register is basically with reference to the MSB (bit-31) of the data written in either BSD0 or BSD1 or BSDR. That means in case of 0 position detection if the data written to BSD0 is 0xFFFFFFFE then the BSSR register would have the result as 0x0000001F. This is because the first occurrence of 0 is at the 31st bit with reference to MSB. When there is no 0 or 1 or changing position detected respectively in the case of BSD0 or BSD1 or BSDR then the BSRR register would have the result as 0x00000020.

The result is available immediately after writing either BSD0 or BSD1 or BSDR.

Table 1. Bit Search Examples

Sr. No.	Type of Detection	Register Used	Data	Result in BSRR
1	Zero	`BSD0`	`0x00000000`	`0x00000000`
2	Zero	`BSD0`	`0x800055AA`	`0x00000001`
3	Zero	`BSD0`	`0xFFFF0000`	`0x00000010`
4	Zero	`BSD0`	`0xFFFFFFFF`	`0x00000020`
5	One	`BSD1`	`0x00000000`	`0x00000020`
6	One	`BSD1`	`0x800055AA`	`0x00000000`
7	One	`BSD1`	`0x0000FFFF`	`0x00000010`

Sr. No.	Type of Detection	Register Used	Data	Result in BSRR
8	One	BSD1	0xFFFFFFFF	0x00000000
9	Changing Position	BSDC	0x00000000	0x00000020
10	Changing Position	BSDC	0xFFFFFFFF	0x00000020
11	Changing Position	BSDC	0xFF0000FF	0x00000008
12	Changing Position	BSDC	0x000005AA	0x00000015
13	Changing Position	BSDC	0x01234567	0x00000007

3 Bit Search Examples

Examples for Bit Search

3.1 Restoring Previous Bit Search State

The following example demonstrates to restore the previous bit search results when the bit search unit is simultaneously used in the application and the interrupt service routine. Here the bit search unit is used the main routine “normally” whereas in order to use it in the ISR first the contents of the BSD1 register is saved in a temporary variable, the required bit search is performed and then the original contents of the BSD1 register is copied back to it. This way previous bit search state and corresponding results are restored no matter the previous bit search was 0-detection, 1-detection or changing position detection.

```
void main(void)
{
       unsigned int Data[3] = {0x12340000, 0xFFAA5500, 0xFFF3230F};
       unsigned int Result[3];
       __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
       . . .
   BSD0 = Data[0];   // Detect the first 0-position in Data[0]
       Result[0] = BSRR; // Detection result copied to Result[0]
       . . .
   BSD1 = Data[1];   // Detect the first 1-position in Data[1]
       Result[1] = BSRR; // Detection result copied to Result[1]
       . . .
   BSD2 = Data[2];   // Detect the changing position in Data[2]
       Result[2] = BSRR; // Detection result copied to Result[2]
              . . .
   while(1)           // endless loop
   {
             HWWD_CL = 0;
          }
   }
   __interrupt void Resource1_IRQ(void)
   {
       unsigned int temp,
       unsigned int I_Data = 0xFEDCBA98;
       unsigned int I_Result;
       temp = BSD1;       // Evacuate BSD1

       BSD2 = I_Data;     // Perform the intended bit search
   I_Result = BSRR;

   BSD1 = temp;       // Restore BSD1
   }
```

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: AN205373 - FR, MB91460, Bit Search

Document Number: 002-05373

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	06/17/2008	First Version; MPi
*A	5132492	NOFL	02/10/2016	Converted Spansion Application Note “MCU-AN-300079-E-V10” to Cypress format
*B	5862900	AESATMP9	08/24/2017	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

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