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.

AN205328

How to Use DBG Pin with F²MC-8FX Family MB95200H/210H Series

This application note describes how to use DBG pin. The DBG pin can be used as 1-line UART communication port in debug mode and can be used as P12/EC0 in free-run mode.

1 Introduction

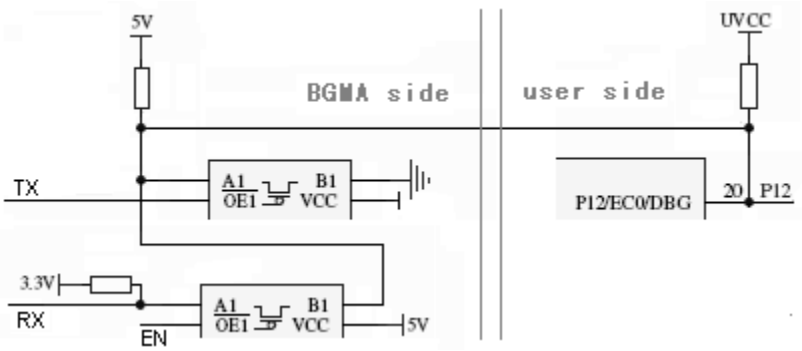
This application note describes how to use DBG pin. The DBG pin can be used as 1-line UART communication port in debug mode and can be used as P12/EC0 in free-run mode.

2 Function of DBG Pin

This chapter explains functions of DBG pin in different modes.

2.1 DBG Pin Hardware Connection (Debug Mode)

Figure 1. DBG Pin Hardware Connection in Debug Mode



2.2 Performance in Debug Mode

DBG pin is used as 1-line UART pin when target MCU enters into debug mode. To ensure real time communication with BGM Adapter, other functions of DBG pin are disabled.

The function of P12 is disabled. Any operation for P12 in user code is not effective.

EC0 is also disabled. Another pin can be used as EC0 pin. “1” is written to SYSC [EC0SL] to select P04/EC0 pin as external counter clock input. Do not set P04 pin as EC0 input and HCLK1 input at the same time.

2.3 Performance in Free-run Mode

1-line UART function is disabled when target MCU enters into free-run mode. P12/EC0 can be used in user code. But DBG pin cannot be set as 1-line UART pin in user code.

3 How to Use DBG Pin

This chapter gives some examples for DBG pin setting in different modes.

3.1 Used as 1-line UART Pin / P12 Pin

In Debug mode, DBG pin is used as 1-line UART pin automatically. DBG pin keeps high when UART is in idle status. In free-run mode, DBG pin is used as P12 by default. When the following user code are written,

```
DDR1_D12 = 1;  // set P12 as output pin
PDR1_P12 = 0;  // pull P12 to low.
```

DBG pin still keeps high level after running user codes above in debug mode. DBG pin is pulled to low level after running these codes in free-run mode.

3.2 Used as EC0 Pin

DBG pin cannot be used as EC0 pin in debug mode. To realize the external clock input for 8/16 compound timer, EC0 pin can be changed to P04 pin in debug mode.

```
SYSC = 0xEB;     // set P04 as EC0 pin, disable HCLK1
T00CR0 = 0x71;  // interval timer, external clock source.
TMCR0 = 0x43;  // 8 bit mode, output T00.
T00DR = 0x08;  // set counter.
T00CR1 = 0x81; // enable output, start T00.
```

P04 is set as EC0 pin in sample code above and T00 is set in continuous interval timer mode. Execute the following operations by order. Input a clock signal to P04 pin→run these codes in debug mode→check T00 pin; then output clock can be used as input clock divided by 8.

```
SYSC = 0xE3;   // set P12 as EC0 pin, disable HCLK1
T00CR0 = 0x71; // interval timer, external clock source.
TMCR0 = 0x43;  // 8bit mode, output T00.
T00DR = 0x08;  // set counter.
T00CR1 = 0x81; // enable output, start T00.
```

P12 is set as EC0 pin in sample code above and T00 is set in continuous interval timer mode. Execute the following operations by order. Input a clock signal to P12 pin→run these codes in free-run mode→check T00 pin, then output clock can be used as input clock divided by 8.

4 Precautions on Using DBG Pin

This chapter describes precautions on using DBG pin.

1. The peripheral circuit connected to DBG pin shouldn't affect 1-line UART communication when using BGM Adapter for debugging. Please refer to Figure 1 when designing user board.
2. It is recommended that P12 is set as input pin or output low level in user code instead of output high level when debugging. DBG pin will be set as output high when target MCU runs in free-run mode, if P12 is set as output high level. A large current will pass through DBG pin when BGM adapter transfer "0" with 1-line UART pin if target board is connected to BGM adapter on this condition.

5 Sample Code

Start.asm

```
//Sample code for initialization
//-------------------------------------
             .PROGRAM     start
             .TITLE       start

// Settings
             #set         OFF       0
             #set         ON        1
             # set        STACKSIZE   64      // Stack size
//Clock Mode Selection
             # set        SUB         0       //Sub clock mode
             # set        MAIN        2       // Main clock mode
//DIV Clock Selection (Machine clock division ratio)
             # set        CLK_0       0       // Original oscillator div 1
             # set        CLK_4       1       // Original oscillator div 4
             # set        CLK_8       2       //Original oscillator div 8
             # set        CLK_16      3       //Original oscillator div 16
//external declaration of symbols
             .EXPORT      __start
             .IMPORT      _main
//definition to stack area
          .SECTION        STACK, STACK, ALIGN=1
          .RES.B          STACKSIZE
STACK_TOP:
          .RES.B          2
//========= ================================================
// Start-Up Code
//=========================================================
//    ___  _____   __    ___  _____
//   /       |    /  \   |   \    |
//   \___    |   |    | |___/    |
//       \   |   |----| |  \     |
//    ___/   |   |    | |   \    |
//Begin of actual code section
//=========================================================
//The Mode Byte is defined at the beginning of the start.asm
          .SECTION        RESVECT, CONST, LOCATE=H'FFFD
          .DATA.B         0
          .DATA.W         __start
; code area
          .SECTION  CODE, CODE, ALIGN=1
__start:
; set stack pointer
            MOVW          A , #STACK_TOP
            MOVW          SP , A
// Set Register bank Pointer 0 / set Direct bank Pointer 0 (0x80..0xFF)
            MOVW          A  ,  PS
            MOVW          A  ,  #0x00BF // RP=0, DP=0, I=0
            ANDW          A
            MOVW          PS ,  A
// Set ILM to the lowest level(3)
            MOVW          A  ,  PS
            MOVW          A  ,  #0x0030
            ORW           A
            MOVW          PS ,  A
            MOV           A  ,  0X0FE4  // READ CRTH
            AND           A  ,  #0X9F   //CLEAR CRTH[6:5]
```

```
            OR              A   ,   #0X60   // 10MHz,    0X00: 1MHz
                                            //0X20: 12.5MHz
                                            // 0X40: 10MHz     0X60: 8MHz
            MOV             0X0FE4 , A      // WRITE SYSC


// call main routine
            CALL            _main
end:        JMP             end

            .END            __start
```

vector.c

```
//  VECTORS.C Interrupt level (priority) setting
 - Interrupt vector definition
#include "mb95200.h"
void InitIrqLevels(void)
{   ILR0 = 0xEF;  // IRQ0:  external interrupt  ch4        --> 01
                  // IRQ1:  external interrupt  ch5        --> 01
                  // IRQ2:  external interrupt ch2 | ch6   --> 01
                  // IRQ3:  external interrupt ch3 | ch7   --> 01
    ILR1 = 0xFF;  // IRQ4:  UART/SIO ch0
                  // IRQ5:  8/16-bit timer ch0 (lower)
                  // IRQ6:  8/16-bit timer ch0 (upper)
                  // IRQ7:  LIN-UART (reception)
    ILR2 = 0xFF;  // IRQ8: LIN-UART (transmission)
                  // IRQ9:  8/16-bit PPG ch1 (lower) | UART/SIO ch1
                  // IRQ10: 8/16-bit PPG ch1 (upper) | I2C ch1
                  // IRQ11: 16-bit reload timer ch0
    ILR3 = 0xFF;  // IRQ12: 8/16-bit PPG ch0 (upper)
                  // IRQ13: 8/16-bit PPG ch0 (lower)
                  // IRQ14: 8/16-bit timer ch1 (upper)
                  // IRQ15: 16-bit PPG ch0 + ch2
    ILR4 = 0xFF;  // IRQ16: 16-bit reload timer ch1 | I2C ch0
                  // IRQ17: 16-bit PPG ch1
                  // IRQ18: 10-biat A/D-converter    --> 01
                  // IRQ19: Timebase timer
    ILR5 = 0xFF;  // IRQ20: Watch timer / counter
                  // IRQ21: external interrupt ch 8-11
                  // IRQ22: 8/16-bit timer ch1 (lower) | external interrupt ch 12-15
                  //  IRQ23: Flash | Custom ch1
}

//   Prototypes
//=====================================================================
//Add your prototypes, each vector definition needs a prototype. Either do it here or
include a header file containing them.
//=====================================================================
//extern unsigned int delay_timer;
__interrupt void DefaultIRQHandler (void);

// Vector definition
// Use following statements to define vectors.
// All resource related vectors are predefined.
// Remaining software interrupts can be added hereas well.
#pragma intvect DefaultIRQHandler 0   //IRQ0:  external interrupt ch4
#pragma intvect DefaultIRQHandler 1   //IRQ1:  external interrupt ch5
#pragma intvect DefaultIRQHandler 2   //IRQ2:  external interrupt ch2 | ch6
#pragma intvect DefaultIRQHandler 3   //IRQ3:  external interrupt ch3 | ch7
#pragma intvect DefaultIRQHandler 5   //IRQ5:  8/16-bit timer ch0 (lower)
```

```
#pragma intvect DefaultIRQHandler 6   //IRQ6:  8/16-bit timer ch0 (upper)
#pragma intvect DefaultIRQHandler 7   //IRQ7:  LIN-UART (reception)
#pragma intvect DefaultIRQHandler 8   //IRQ8:  LIN-UART (transmission)
#pragma intvect DefaultIRQHandler 14  //IRQ14: 8/16-bit timer ch1 (upper)
#pragma intvect DefaultIRQHandler 18  //IRQ18: 10-biat A/D-converter
#pragma intvect DefaultIRQHandler 19  //IRQ19: Timebase timer
#pragma intvect DefaultIRQHandler 20  //IRQ20: Watch timer / counter
#pragma intvect DefaultIRQHandler 22  //IRQ22: 8/16-bit timer ch1 (lower) |
                                      //external interrupt ch 12-15
#pragma intvect DefaultIRQHandler 23  //IRQ23: Flash | Custom ch1


__interrupt
void DefaultIRQHandler (void)
{
    __DI();              // disable interrupts
    While(1)
        __wait_nop();    // halt system
}
```

Main.c

```
//=====================================================================
//following code created for DBG pin usage test
#define DEBUGMODE


//set DBG pin as P12 output
void gpio_setting(void)
{
   DDR1_P12 = 1;   /* set P12 as output pin */
   PDR1_P12 = 0;   /* pull P12 to low. */

   AIDRL_P04 = 1;  /* port input enable */

}

//set DBG pin as EC0 pin
void p12_as_ec0(void)
{
   SYSC = 0xE3;    // set P12 as EC0 pin, disable HCLK1
   T00CR0 = 0x71;  // interval timer, external clock source.
   TMCR0 = 0x43;   // 8bit mode, output timer00.
   T00DR = 0x08;   // set counter.
   T00CR1 = 0x81;  // enable output, start timer00.
}
//change EC0 pin to P04
void p04_as_ec0(void)
{
   SYSC = 0xEB;    // set P04 as EC0 pin, disable HCLK1
   T00CR0 = 0x71;  // interval timer, external clock source.
   TMCR0 = 0x43;   // 8bit mode, output timer00.
   T00DR = 0x08;   // set counter.
   T00CR1 = 0x81;  // enable output, start timer00.
}

//main function
void main(void)
{
gpio_setting();    //pull low P12 pin
...
#ifdef DEBUGMODE
```

```
P04_as_ec0();       //set P04 as EC0 pin in debug mode
#else
P12_as_ec0();       //set P12 as EC0 pin in free run mode
#endif
}
```

Document History

Document Title: AN205328 - How to Use DBG Pin with F²MC-8FX Family MB95200H/210H Series

Document Number: 002-05328

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	HUAL	03/18/2008	Initial release.
*A	5260187	HUAL	05/09/2016	Migrated Spansion Application note from MCU-AN- 500009-E-10 to Cypress format.

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
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com