NetBurner

Mod5234 GPIO Configuration

Application Note

Revision 1.3
March 2, 2009
Document Status: Initial Release

Table of Contents

Introduction

Most of the Motorola ColdFire 5234 processor pins can be configured as general purpose input/output (GPIO) pins. These pins have multiple functions, and the specific pins that make sense to use as GPIO will depend on which peripherals are needed for the targeted application. For example, the QSPI pins can be used for their primary function, or they can be configured as GPIO. This document will examine each pin and show how it can be conifigured as GPIO. Additional information contained in this document can be found in "Chapter 12 – General Purpose I/O Module" of the MCF5235 reference manual.

Pin Assignment and GPIO Control Registers

Any pin that can be used for GPIO functionality is configurable by at least five registers: a port output data register, a port data direction register, a port pin data/set data register, a port clear output data register, and a pin assignment register. Some GPIO ports mentioned here have a sixth register called a drive strength control register which allows control of the current to be driven high or low on the pins by setting the corresponding bits.

To access the pin assignment and GPIO control registers, the following preprocessor directive must be included (the path may vary depending on where the sim5234.h header file is located, relative to the Nburn root directory):

```
#include <..\MOD5234\system\sim5234.h>
```

In the `sim5234.h` header file (`\Nburn\MOD5234\system\sim5234.h`), GPIO registers are grouped into a struct type definition called `gpiostruct`. Each GPIO register is prefixed with a tag to identify whether it is a pin assignment, port output data, port data direction, port pin data/set data, port clear output data, or drive strength control register (additional information of each register can be found in the following sections). The following table provides the name of the tags and the name of the register type associated with each tag.

Tag Prefix	Type of Register
`par`	Pin Assignment Register
`podr`	Port Output Data Register
`pddr`	Port Data Direction Register
`ppdsdr`	Port Pin Data/Set Data Register
`pclrr`	Port Clear Output Data Register
`dscr`	Drive Strength Control Register

Rather than reading hard-coded bits, a special header file called gpio5234.h is included with this application note to help make reading and understanding the examples in this document easier. The header file has two main sets of definitions: one set for the pin assignment register, and the second set for the rest of the other GPIO registers, with the exception of the drive strength control register. Each set is divided into subgroups, which indicate what port the GPIO register(s) are designated for. Examples of how this header file is used are shown in the code examples below for each set of port pins.

Pin Assignment Register (PAR)

The pin assignment register controls the functionality of the pins. Pins can have either one or two assigned bits for configuration. One-bit assignments allow pins to be configured between their primary function or GPIO, and two-bit assignments allow pins to be configured between their primary function, their alternate function, or GPIO (some may also have a fourth functionality). Setting a ‘0’ (1-bit) or “00” (2-bit) will configure any pin for GPIO, while a ‘1’ or “11” will configure any pin for their primary function.

In the charts below for each group of pins, the pin assignment bits column helps identify the bits in the pin assignment register that control the corresponding pins’ functionality. Pin assignment registers can be eight bits wide or sixteen bits wide. For pin assignment registers that are sixteen bits wide (i.e., the UART port pin assignment register), the associated GPIO control registers (port data direction, port pin data/set, port clear output, and port output data registers) are split into two eight-bit registers: a high and low register.

Port Data Direction Register (PDDR)

The port data direction registers control the direction of the pins when they are configured for GPIO. The registers are eight bits wide, but not all groups of pins mentioned here will use all eight bits.

The registers are read/write. At reset, all bits in the PDDR registers are cleared. Setting any bit to ‘1’ in a PDDR register configures the corresponding GPIO pin as an output. Setting any bit to ‘0’ in a PDDR register configures the corresponding pin as an input.

Port Output Data Register (PODR)

The port output data registers store the data to be driven on the corresponding port pins when the pins are configured for general purpose output. The registers are each eight bits wide, but not all of them use all eight bits.

The registers are read/write. At reset, all implemented bits in the PODR registers are set. Reserved bits always remain cleared. Reading a PODR register returns the current values in the register. To set bits in a PODR register, write ‘1’ to the bits, or write ‘1’ to the corresponding bits in the port pin data/set data register. To clear bits in a PODR register, write ‘0’ to the bits, or write ‘0’ to the corresponding bits in the port clear output data register.

Port Pin Data/Set Data Register (PPDSDR)

The port pin data/set data register reflects the current pin states and control the setting of output pins when the pins are configured for GPIO. The registers are each eight bits wide, but not all groups of pins mentioned here will use all eight bits.

The registers are read/write. At reset, the bits in the PPDSDR registers are set to the current pin states. Reading a PPDSDR register returns the current state of the associated pins. Setting bits in this register sets the corresponding bits in the port output data register. Writing a ‘0’ has no effect.

Port Clear Output Data Register (PCLRR)

The port clear output data register clears the corresponding bits in the port output data register. The registers are each eight bits wide, but not all groups of pins mentioned here will use all eight bits.

The registers are read/write. Setting it has no effect. Writing ‘0’ to bits in this register clears the corresponding bits in the port output data register. Reading the PCLRR register returns zeros.

Drive Strength Control Register (DSCR)

The drive strength control register sets the output pin drive strengths of the current. All drive strength control registers are read/write. For more information on how to configure and use this register with the corresponding pins, please refer to the Motorola ColdFire 5235 reference manual, section 12.3.1.8.

External Bus Control Pins

The external bus control pin assignment register (PAR_BUSCTL) controls the functions of the external bus control pins (MCF5235 reference manual, table 12-10). The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_BUSCTL, PODR_ BUSCTL, PPDSDR_ BUSCTL, and PCLRR_ BUSCTL).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_BUSCTL]	GPIO Control Bit [xxx_BUSCTL]
J1-13	TA	Transfer Acknowledge	12	6

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// External Bus Control Pin Assignment Register. Set the BUSCTL output
// signal as GPIO.
sim.gpio.par_busctl &= ~PDDR_PODR_TA;

// External Bus Control Port Data Direction Register. Set the BUSCTL
// signal as GPIO output.
sim.gpio.pddr_busctl |= PDDR_PODR_TA;

// External Bus Control Port Output Data Register. Set GPIO signal
// TA high.
sim.gpio.podr_busctl |= PDDR_PODR_TA;

// External Bus Control Port Pin Data/Set Data Register. Read the
// current pin value for the GPIO pin TA.
BYTE value_busctl = sim.gpio.ppdsdr_busctl & PDDR_PODR_TA;

// External Bus Control Drive Strength Control Register. Set a high
// output drive strength for for the BUSCTL pin configured as GPIO
// (bit 0 - TA).
sim.gpio.dscr_eim |= 0x01;
```

Inter-Integrated Circuit (I^2C) Pins

The inter-integrated circuit pin assignment register (PAR_FECI2C) controls the functions of the I^2C pins (MCF5235 reference manual, table 12-14). The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_FECI2C, PODR_FECI2C, PPDSDR_FECI2C, and PCLRR_FECI2C).

WARNING: PAR_FECI2C[7:4] controls the ethernet management data clock and input/output signals for the fast ethernet controller.

MOD5234 Pin No.	**MOD5234 Signal Name**	**Signal Description**	**Pin Assign Bit(s) [PAR_FECI2C]**	**GPIO Control Bit [xxx_FECI2C]**
J2-39	SDA	I^2C Serial Data	1-0	0
J2-42	SCL	I^2C Serial Clock	3-2	1

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// I2C Pin Assignment Register. Set the SDA signal as GPIO.
sim.gpio.par_feci2c &= ~GPIO_PAR_SDA;

// I2C Port Data Direction Register. Set the SDA signal as a GPIO
// output.
sim.gpio.pddr_feci2c |= PDDR_PODR_SDA;

// I2C Port Output Data Register. Set the GPIO signal SDA high.
sim.gpio.podr_feci2c |= PDDR_PODR_SDA;

// I2C Port Pin Data/Set Data Register. Read the current pin values for
// the GPIO pin corresponding to SDA.
BYTE value_i2c = sim.gpio.ppdsdr_feci2c & PDDR_PODR_SDA;

// I2C Drive Strength Control Register. Set a low output drive strength
// for any I2C pin configured for GPIO (bit 0 – all I2C pins).
sim.gpio.dscr_feci2c &= ~( 0x01 );
```

Universal Asynchronous Receive and Transmit (UART) Pins

The UART pin assignment register (PAR_UART) controls the functions of the UART pins (MCF5235 reference manual, table 12-15). The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_UARTL / PDDR_UARTH and PODR_UARTL / PODR_UARTH, PPDSDR_UARTL / PPDSDR_UARTH, and PCLRR_UARTL / PCLRR_UARTH).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_UART]	GPIO Control Bit [xxx_UARTL]
J2-38	U0RTS	UART 0 Request to Send	0	2
J2-29	U0CTS	UART 0 Clear to Send	1	3
J2-04	U0TXD	UART 0 Transmit	2	1
J2-03	U0RXD	UART 0 Receive	3	0
J2-23	U1RTS	UART 1 Request to Send	5-4	6
J2-24	U1CTS	UART 1 Clear to Send	7-6	7
J2-22	U1TXD	UART 1 Transmit	9-8	5
J2-21	U1RXD	UART 1 Receive	11-10	4

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_UART]	GPIO Control Bit [xxx_UARTH]
J2-44	U2TXD	UART 2 Transmit	12	1
J2-41	U2RXD	UART 2 Receive	13	0

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// UART Pin Assignment Register. Set signals U0RTS and U2TXD as GPIO.
sim.gpio.par_uart &= ~( GPIO_PAR_U0RTS | GPIO_PAR_U2TXD );

// UART Port Data Direction Register. Set signals U0RTS and U2TXD as
// GPIO outputs.
sim.gpio.pddr_uarth |= PDDR_PODR_U2TXD;
sim.gpio.pddr_uartl |= PDDR_PODR_U0RTS;

// UART Port Output Data Register. Set GPIO signal U0RTS high and U2TXD
// low.
sim.gpio.podr_uarth &= ~PDDR_PODR_U2TXD;
sim.gpio.podr_uartl |= PDDR_PODR_U0RTS;

// UART Port Pin Data/Set Data Register. Read the current pin values
// for the GPIO pins corresponding to U0RTS and U2TXD.
BYTE value_uarth = sim.gpio.ppdsdr_uarth & PDDR_PODR_U2TXD;
BYTE value_uartl = sim.gpio.ppdsdr_uartl & PDDR_PODR_U0RTS;

// UART Drive Strength Control Register. Set a high output drive
// strength for any UART 0 pins configured for GPIO and a low output
// drive strength for any UART 2 pins configured for GPIO (bit 0 – UART
// 0 pins, bit 2 – UART 1 pins, bit 4 – UART 2 pins).
sim.gpio.dscr_uart |= ( 0x01 );
sim.gpio.dscr_uart &= ~( 0x10 );
```

Queued Serial Peripheral Interface (QSPI) Pins

The QSPI pin assignment register (PAR_QSPI) controls the functions of the QSPI pins (MCF5235 reference manual, table 12-16). The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_QSPI, PODR_QSPI, PPDSDR_QSPI, and PCLRR_QSPI).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_QSPI]	GPIO Control Bit [xxx_QSPI]
J2-25	SPI_CLK	SPI Clock	1-0	2
J2-28	SPI_DOUT	SPI Data Out	2	0
J2-27	SPI_DIN	SPI Data In	4-3	1
J2-30	SPI_CS0	SPI Chip Select 0	5	3
J2-40	SPI_CS1	SPI Chip Select 1	7-6	4

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// QSPI Pin Assignment Register. Set signals SPI_CLK and SPI_CS0 as
// GPIO.
sim.gpio.par_qspi = ~( GPIO_PAR_SPICLK | GPIO_PAR_SPICS0 );

// QSPI Port Data Direction Register. Set signals SPI_CLK and SPI_CS0
// as GPIO outputs.
sim.gpio.pddr_qspi |= ( PDDR_PODR_SPICLK | PDDR_PODR_SPICS0 );

// QSPI Port Output Data Register. Set both GPIO signals low.
sim.gpio.podr_qspi &= ~( PDDR_PODR_SPICLK | PDDR_PODR_SPICS0 );

// QSPI Port Pin Data/Set Data Register. Read the current pin values
// for the GPIO pins corresponding to SPI_CLK and SPI_CS0.
BYTE value_qspi = sim.gpio.ppdsdr_qspi & ( PDDR_PODR_SPICLK |
                                           PDDR_PODR_SPICS0 );

// QSPI Drive Strength Control Register. Set a high output drive
// strength for any QSPI pin configured for GPIO (bit 0 – all QSPI
// pins).
sim.gpio.dscr_qspi |= 0x01;
```

DMA Timer Pins

The DMA timer pin assignment register (PAR_TIMER) controls the functions of the DMA timer pins (MCF5235 reference manual, table 12-17). The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_TIMER, PODR_TIMER, PPDSDR_TIMER, and PCLRR_TIMER).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_TIMER]	GPIO Control Bit [xxx_TIMER]
J2-36	DT0OUT	DMA Timer Output 0	1-0	0
J2-34	DT1OUT	DMA Timer Output 1	3-2	2
J2-33	DT2OUT	DMA Timer Output 2	5-4	4
J2-31	DT0IN	DMA Timer Input 0	9-8	1
J2-37	DT1IN	DMA Timer Input 1	11-10	3

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// DMA Timer Pin Assignment Register. Set the three DMA timer output
// signals as GPIO.
sim.gpio.par_timer &= ~( GPIO_PAR_DT0OUT | GPIO_PAR_DT1OUT |
                         GPIO_PAR_DT2OUT );

// DMA Timer Port Data Direction Register. Set the three DMA timer
// output signals as GPIO outputs.
sim.gpio.pddr_timer |= ( PDDR_PODR_DT0OUT | PDDR_PODR_DT1OUT |
                         PDDR_PODR_DT2OUT );

// DMA Timer Port Output Data Register. Set GPIO signals DT0OUT and
// DT1OUT high and DT2OUT low.
sim.gpio.podr_timer |= ( PDDR_PODR_DT0OUT | PDDR_PODR_DT1OUT );
sim.gpio.podr_timer &= ~PDDR_PODR_DT2OUT;

// DMA Timer Port Pin Data/Set Data Register. Read the current pin
// values for the GPIO pins corresponding to DT0OUT, DT1OUT, and
// DT2OUT.
BYTE value_timer = sim.gpio.ppdsdr_timer & ( PDDR_PODR_DT0OUT |
                                             PDDR_PODR_DT1OUT |
                                             PDDR_PODR_DT2OUT );

// DMA Timer Drive Stength Control Register. Set a high output drive
// strength for any DMA timer pins configured for GPIO (bit 0 – all
// DMA timer pins).
sim.gpio.dscr_timer |= 0x01;
```

ETPU Pins

The ETPU pin assignment register (PAR_ETPU) controls the functions of the ETPU pins (MCF5235 reference manual, table 12-18), not the ETPU channel pins. The GPIO control bits correspond to bit positions in the GPIO control registers (PDDR_ETPU, PODR_ETPU, PPDSDR_ETPU, and PCLRR_ETPU).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_ETPU]	GPIO Control Bit [xxx_ETPU]
J2-35	LTPUODIS	Lower TPU Channel Output Disable	0	0
J2-32	UTPUODIS	Upper TPU Channel Output Disable	1	1
J2-26	TCRCLK	TPU Time Base Clock	2	2

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// ETPU Pin Assignment Register. Set the three ETPU signals as GPIO.
sim.gpio.par_etpu &= ~( GPIO_PAR_LTPUODIS | GPIO_PAR_UTPUODIS |
                        GPIO_PAR_TCRCLK );

// ETPU Port Data Direction Register. Set the ETPU signals as GPIO
// outputs.
sim.gpio.pddr_etpu |= ( PDDR_PODR_LTPUODIS | PDDR_PODR_UTPUODIS |
                        PDDR_PODR_TCRCLK );

// ETPU Port Output Data Register. Set GPIO signals LTPUODIS and
// UTPUODIS high and TCRCLK low.
sim.gpio.podr_etpu |= ( PDDR_PODR_LTPUODIS | PDDR_PODR_UTPUODIS );
sim.gpio.podr_etpu &= ~PDDR_PODR_TCRCLK;

// ETPU Port Pin Data/Set Data Register. Read the current pin values
// for the GPIO pins corresponding to LTPUODIS, UTPUODIS, and TCRCLK.
BYTE value_etpu = sim.gpio.ppdsdr_etpu & ( PDDR_PODR_LTPUODIS |
                                           PDDR_PODR_UTPUODIS |
                                           PDDR_PODR_TCRCLK );
```

Edge Port Module

The edge port pin assignment register (EPPAR) controls the functions of the external interrupt pins (MCF5235 reference manual, chapter 15). Since all pins default to general purpose input pins at reset, there is no need to configure them for GPIO, nor are they configurable for GPIO via EPPAR. The GPIO control bits correspond to bit positions in the GPIO control registers (EPORT data direction register [EPDDR], EPORT data register [EPDR], and EPORT pin data register [EPPDR]).

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [EPPAR]	GPIO Control Bit [EPxxx]
J2-43	IRQ1	External Interrupt 1	3-2	1
J2-45	IRQ3	External Interrupt 3	7-6	3
J2-47	IRQ5	External Interrupt 5	11-10	5
J2-48	IRQ7	External Interrupt 7	15-14	7

Code examples (Note: `sim5234.h` and `gpio5234.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// Edge Port Data Direction Register. Set signals IRQ1 and IRQ5 as
// GPIO outputs.
sim.eport.epddr |= ( PDDR_PDR_IRQ1 | PDDR_PDR_IRQ5 );

// Edge Port Data Register. Set GPIO signals IRQ1 and IRQ5 low.
sim.eport.epdr &= ~( PDDR_PDR_IRQ1 | PDDR_PDR_IRQ5 );

// Edge Port Pin Data Register. Read the current pin values for the
// GPIO pins corresponding to IRQ1 and IRQ5.
BYTE value_ep = sim.eport.eppdr & ( PDDR_PDR_IRQ1 | PDDR_PDR_IRQ5 );

// UART Drive Stength Control Register. Set a high output drive
// strength for any IRQ pins configured for GPIO (bit 6 – all
// IRQ pins).
sim.gpio.dscr_uart |= 0x40;
```

Programming Multiple Pins on the Same Port

Programming multiple pins in one line of code is possible as long as they belong to the same port. For example, you can program all five pins associated with the DMA Timer port at once, but you cannot program a pin from the DMA Timer port and a pin from the QSPI port at once. The following example shows how this is done with the DMA Timer port (use the DMA Timer port chart to see what pins are affected by the bit configurations) :

MOD5234 Pin No.	MOD5234 Signal Name	Signal Description	Pin Assign Bit(s) [PAR_TIMER]	GPIO Control Bit [xxx_TIMER]
J2-36	DT0OUT	DMA Timer Output 0	1-0	0
J2-34	DT1OUT	DMA Timer Output 1	3-2	2
J2-33	DT2OUT	DMA Timer Output 2	5-4	4
J2-31	DT0IN	DMA Timer Input 0	9-8	1
J2-37	DT1IN	DMA Timer Input 1	11-10	3

```
// DMA Timer Pin Assignment Register. Set all six of the DMA timer
// signals at GPIO.
sim.gpio.par_timer &= 0xF0C0;   // 1111 0000 1100 0000

// DMA Timer Port Data Direction Register. Set the DMA Timer Inputs as
// GPIO outputs and the DMA Timer Outputs as GPIO inputs.
sim.gpio.pddr_timer = 0x0A;     // 0000 1010
```

In the above example, all five pins belonging to the DMA Timer port were programmed for use as GPIO pins, followed by configuring the DMA Timer Output signal pins as GPIO inputs, and the DMA Timer Input signal pins as GPIO outputs.