Mod5213 GPIO Configuration

Application Note

Revision 1.2
March 2, 2009
Document Status: Initial Release

Table of Contents

Introduction

Most of the Motorola ColdFire 5213 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 11 – General Purpose I/O Module" of the MCF5213 Reference Manual (Revision 1.2).

Electrical Specifications

The current drive capabilities of the GPIO pins are the same for all pins. The instantaneous maximum current for a single pin is 25 mA. The sustained current drive is 2 mA. Please see the document, "MCF5213 Microcontroller Family Hardware Specification" for more information.

Pin Assignment and GPIO Control Registers

Any pin that can be used for GPIO functionality is configurable by five types of 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.

To access the pin assignment and GPIO control registers, the following preprocessor directive must be included (the path may vary depending on where the sim5213.h header file is located):

```
#include "..\MOD5213\system\sim5213.h"
```

In the `sim5213.h` header file (`\Nburn\MOD5213\system\sim5213.h`), GPIO registers are grouped into a struct type definition called `gpiostruct`. Each GPIO register has a tag to identify whether it is a pin assignment, port output data, port data direction, port pin data/set data, or port clear output data register (additional information on 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. Note: "xx" represents the name of any port register.

Tag	Type of Register
PxxPAR	Pin Assignment Register
PORTxx	Port Output Data Register
DDRxx	Port Data Direction Register
SETxx	Port Pin Data/Set Data Register
CLRxx	Port Clear Output Data Register

Rather than reading hard-coded bits, a special header file called gpio5213.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 port data direction, port output data, port pin data/set data, and port clear output data registers. Each set is divided into subgroups that indicate what GPIO port they are designated for. Examples of how the definitions in the header file are 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 "01" 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.

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.

Port AS Pins

The port AS pin assignment register (PASPAR) controls the functions of the I^2C pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRAS, PORTAS, SETAS, and CLRAS).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PASPAR]	GPIO Control Bit [*AS]
4	SDA	I^2C Serial Data	1-0	0
5	SCL	I^2C Serial Clock	3-2	1

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port AS Pin Assignment Register. Set both pins as GPIO.
sim.gpio.paspar &= ~( GPIO_PAR_SDA | GPIO_PAR_SCL );

// Port AS Port Output Data Register. Set GPIO signal SDA low and
// SCL high.
sim.gpio.portas &= ~GPIO_PIN_SDA;
sim.gpio.portas |= GPIO_PIN_SCL;

// Port AS Port Data Direction Register. Set signals SDA and SCL
// as GPIO outputs.
sim.gpio.ddras |= GPIO_PIN_DATAL0;
sim.gpio.ddras |= GPIO_PIN_DATAH8;

// Port AS Port Pin Data/Set Data Register. Read the current pin values
// for GPIO pins corresponding to SDA and SCL.
BYTE value_as = sim.gpio.setas & ( GPIO_PIN_SDA | GPIO_PIN_SCL );
```

Port QS Pins

The port QS pin assignment register (PQSPAR) controls the functions of the QSPI pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRQS, PORTQS, SETQS, and CLRQS).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PQSPAR]	GPIO Control Bit [*QS]
33	QSPI_CS2	QSPI Chip Select 2	11-10	5
34	QSPI_CS1	QSPI Chip Select 1	9-8	4
35	QSPI_CS0	QSPI Chip Select 0	7-6	3
36	QSPI_DOUT	QSPI Data Out	1-0	0
37	QSPI_DIN	QSPI Data In	3-2	1
38	QSPI_CLK	QSPI Clock	5-4	2

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port QS Pin Assignment Register. Set all port QS pins as GPIO.
sim.gpio.pqspar &= ~( GPIO_PAR_QSPI_DOUT | GPIO_PAR_QSPI_DIN |
                      GPIO_PAR_QSPI_CLK | GPIO_PAR_QSPI_CS2 |
                      GPIO_PAR_QSPI_CS1 | GPIO_PAR_QSPI_CS0 );

// Port QS Port Output Data Register. Set the QSPI chip select GPIO
// signals low and the others high.
sim.gpio.portqs &= ~( GPIO_PIN_QSPI_CS2 | GPIO_PIN_QSPI_CS1 |
                      GPIO_PIN_QSPI_CS0 );
sim.gpio.portqs |= ( GPIO_PIN_QSPI_DOUT | GPIO_PIN_QSPI_DIN |
                     GPIO_PIN_QSPI_CLK );

// Port QS Port Data Direction Register. Set all QSPI GPIO signals as
// outputs.
sim.gpio.ddrqs |= ( GPIO_PAR_QSPI_DOUT | GPIO_PAR_QSPI_DIN |
                    GPIO_PAR_QSPI_CLK | GPIO_PAR_QSPI_CS2 |
                    GPIO_PAR_QSPI_CS1 | GPIO_PAR_QSPI_CS0 );

// Port QS Port Pin Data/Set Data Register. Read the current pin values
// for GPIO pins corresponding to the QSPI chip selects.
BYTE value_qs = sim.gpio.setqs & ( GPIO_PIN_QSPI_CS2 |
                                   GPIO_PIN_QSPI_CS1 |
                                   GPIO_PIN_QSPI_CS0 );
```

Port TA Pins

The port TA pin assignment register (PTAPAR) controls the functions of the General Purpose Timer pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRTA, PORTTA, SETTA, and CLRTA).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PTAPAR]	GPIO Control Bit [*TA]
25	GPT3	General Purpose Timer 3	7-6	3
26	GPT2	General Purpose Timer 2	5-4	2
27	GPT1	General Purpose Timer 1	3-2	1
28	GPT0	General Purpose Timer 0	1-0	0

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port TA Pin Assignment Register. Set all port TA pins as GPIO.
sim.gpio.ptapar &= ~( GPIO_PAR_GPT0 | GPIO_PAR_GPT1 | GPIO_PAR_GPT2 |
                      GPIO_PAR_GPT3 );

// Port TA Port Output Data Register. Set GPT3 and GPT2 high.
sim.gpio.portta |= ( GPIO_PIN_GPT3 | GPIO_PIN_GPT2 );

// Port TA Port Data Direction Register. Set GPT3 and GPT2 as outputs,
// and set GPT1 and GPT0 as inputs.
sim.gpio.ddrta |= ( GPIO_PIN_GPT3 | GPIO_PIN_GPT2 );
sim.gpio.ddrta &= ~( GPIO_PIN_GPT1 | GPIO_PIN_GPT0 );

// Port TA Port Pin Data/Set Data Register. Read the current pin values
// for GPIO pins corresponding to GPT1 and GPT0.
BYTE value_ta = sim.gpio.setta & ( GPIO_PIN_GPT1 | GPIO_PIN_GPT0 );
```

Port TC Pins

The port TC pin assignment register (PTCPAR) controls the functions of the DMA Timer pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRTC, PORTTC, SETTC, and CLRTC).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PTCPAR]	GPIO Control Bit [*TC]
21	DTIN3	DMA Timer Input 3	7-6	3
22	DTIN2	DMA Timer Input 2	5-4	2
23	DTIN1	DMA Timer Input 1	3-2	1
24	DTIN0	DMA Timer Input 0	1-0	0

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port TC Pin Assignment Register. Set all port TC pins as GPIO.
sim.gpio.ptcpar &= ~( GPIO_PAR_DTIN3 | GPIO_PAR_DTIN2 |
                      GPIO_PAR_DTIN1 | GPIO_PAR_DTIN0 );

// Port TC Port Output Data Register. Set DTIN1 and DTIN0 pins high.
sim.gpio.porttc |= ( GPIO_PIN_DTIN1 | GPIO_PIN_DTIN0 );

// Port TC Port Data Direction Register. Set DTIN3 and DTIN2 as inputs,
// and set DTIN1 and DTIN0 as outputs.
sim.gpio.ddrtc |= ( GPIO_PIN_DTIN1 | GPIO_PIN_DTIN0 );
sim.gpio.ddrtc &= ~( GPIO_PIN_DTIN3 | GPIO_PIN_DTIN2 );

// Port TC Port Pin Data/Set Data Register. Read the current pin values
// for GPIO pins corresponding to the DTIN3 and DTIN2 pins.
BYTE value_tc = sim.gpio.settc & ( GPIO_PIN_DTIN3 | GPIO_PIN_DTIN2 );
```

Port UB Pins

The port UB pin assignment register (PUBPAR) controls the functions of the UART 1 pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRUB, PORTUB, SETUB, and CLRUB).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PUBPAR]	GPIO Control Bit [*UB]
29	URXD1	UART 1 – Receive	3-2	1
30	UTXD1	UART 1 – Transmit	1-0	0
31	UCTS1	UART 1 – Clear to Send	7-6	3
32	URTS1	UART 1 – Req. to Send	5-4	2

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port UB Pin Assignment Register. Set all port UB pins as GPIO.
sim.gpio.pubpar &= ~( GPIO_PAR_UCTS1 | GPIO_PAR_URTS1 |
                      GPIO_PAR_UTXD1 | GPIO_PAR_URXD1 );

// Port UB Port Output Data Register. Set the URXD1 pin low.
sim.gpio.portub &= ~( GPIO_PIN_URXD1 );

// Port UB Port Data Direction Register. Set the URXD1 pin as an
// output, and the other pins as inputs.
sim.gpio.ddrub |= ( GPIO_PIN_URXD1 );
sim.gpio.ddrub &= ~( GPIO_PIN_UTXD1 | GPIO_PIN_UCTS1 |
                     GPIO_PIN_URTS1 );

// Port UB Port Pin Data/Set Data Register. Read the current pin value
// for GPIO pin corresponding to the UTXD1 pin.
BYTE value_ub = sim.gpio.setub & ( GPIO_PIN_UTXD1 );
```

Port UA Pins

The port UA pin assignment register (PUAPAR) controls the functions of the UART 0 pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRUA, PORTUA, SETUA, and CLRUA).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PUAPAR]	GPIO Control Bit [*UA]
2	URXD0	UART 0 – Receive	3-2	1
3	UTXD0	UART 0 – Transmit	1-0	0

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port UA Pin Assignment Register. Set all port UA pins as GPIO.
sim.gpio.puapar &= ~( GPIO_PAR_URXD0 | GPIO_PAR_UTXD0 );

// Port UA Port Output Data Register. Set the UTXD0 pin low.
sim.gpio.portua &= ~( GPIO_PIN_UTXD0 );

// Port UA Port Data Direction Register. Set UTXD0 as output, and URXD0
// as input.
sim.gpio.ddrua |= ( GPIO_PIN_UTXD0 );
sim.gpio.ddrua &= ~( GPIO_PIN_URXD0 );

// Port UA Port Pin Data/Set Data Register. Read the current pin value
// for GPIO pin corresponding to the UTXD1 pin.
BYTE value_ua = sim.gpio.setua & ( GPIO_PIN_URXD0 );
```

Port AN Pins

The port AN pin assignment register (PANPAR) controls the functions of the Analog-to-Digital Converter pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRAN, PORTAN, SETAN, and CLRAN).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PANPAR]	GPIO Control Bit [*AN]
11	AN2	Analog Input 2	2	2
12	AN1	Analog Input 1	1	1
13	AN0	Analog Input 0	0	0
14	AN3	Analog Input 3	3	3
15	AN7	Analog Input 7	7	7
16	AN6	Analog Input 6	6	6
17	AN5	Analog Input 5	5	5
18	AN4	Analog Input 4	4	4

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port AN Pin Assignment Register. Set the AN7 pin as GPIO.
sim.gpio.panpar &= ~( GPIO_PAR_AN7 );

// Port AN Port Output Data Register. Set the AN7 pin high.
sim.gpio.portan |= ( GPIO_PIN_AN7 );

// Port AN Port Data Direction Register. Set the AN7 pin as output.
sim.gpio.ddran |= ( GPIO_PIN_AN7 );

// Port AN Port Pin Data/Set Data Register. Read the current pin value
// for GPIO pin corresponding to the AN7 pin.
BYTE value_an = sim.gpio.setan & ( GPIO_PIN_AN7 );
```

Port NQ Pins

The port NQ pin assignment register (PNQPAR) controls the functions of the Interrupt pins. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRNQ, PORTNQ, SETNQ, and CLRNQ).

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PNQPAR]	GPIO Control Bit [*NQ]
6	IRQ1	External Interrupt #1	1-0	1
7	IRQ4	External Interrupt #4	4	4
8	IRQ7	External Interrupt #7	7	7

Code examples (Note: `sim5213.h` and `gpio5213.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port NQ Pin Assignment Register. Set all port NQ pins as GPIO.
sim.gpio.pnqpar &= ~( GPIO_PAR_IRQ1 | GPIO_PAR_IRQ4 | GPIO_PAR_IRQ7 );

// Port NQ Port Output Data Register. Set all the port NQ pins high.
sim.gpio.portnq |= ( GPIO_PIN_IRQ1 | GPIO_PIN_IRQ4 | GPIO_PIN_IRQ7 );

// Port NQ Port Data Direction Register. Set all the port NQ pins as
// outputs.
sim.gpio.ddrnq |= ( GPIO_PIN_IRQ1 | GPIO_PIN_IRQ4 | GPIO_PIN_IRQ7 );

// Port NQ Port Pin Data/Set Data Register. Read the current pin values
// for GPIO pins corresponding to the port NQ pins.
BYTE value_nq = sim.gpio.setnq & ( GPIO_PIN_IRQ1 | GPIO_PIN_IRQ4 |
                                   GPIO_PIN_IRQ7 );
```

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 four pins associated with Port TC at once, but you cannot program a pin from Port TC and a pin from the Port UB at once. The following example shows how this is done with Port TC (use the Port TC chart to see what pins are affected by the bit configurations) :

MOD5213 Pin No.	MOD5213 Signal Name	Signal Description	Pin Assign Bit(s) [PTCPAR]	GPIO Control Bit [*TC]
21	DTIN3	DMA Timer Input 3	7-6	3
22	DTIN2	DMA Timer Input 2	5-4	2
23	DTIN1	DMA Timer Input 1	3-2	1
24	DTIN0	DMA Timer Input 0	1-0	0

```
// Port TC Pin Assignment Register. Set all four of the Port TC
// signals at GPIO.
sim.gpio.ptcpar &= 0x00;   // 0000 0000

// Port TC Data Direction Register. Set the all the DMA Timer Inputs as
// GPIO outputs.
sim.gpio.ddrtc = 0x0F;     // 0000 1111
```

In the above example, all four pins belonging to Port TC were programmed for use as GPIO pins, followed by configuring all of them as GPIO outputs.