Mod5282 GPIO Configuration

Application Note

Revision 2.1
March 2, 2009
Document Status: Second Revision

Table of Contents

Introduction

Most of the Motorola ColdFire 5282 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 26 – General Purpose I/O Module" of the MCF5282 user's manual.

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.

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.

To access the pin assignment and GPIO control registers, the following preprocessor directive must be included (the path may vary depending on where the sim5282.h header file is located, relative to the main program file):

```
#include “..\MOD5282\system\sim5282.h”
```

In the `sim5282.h` header file (`\Nburn\MOD5282\system\sim5282.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, or port clear output data 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**
`p`*`n`*`par`	Pin Assignment Register
`port`*`n`*	Port Output Data Register
`ddr`*`n`*	Port Data Direction Register
`port`*`n`*`p`	Port Pin Data/Set Data Register
`clr`*`n`*	Port Clear Output Data Register

The '*`n`*' value above represents the port being used. For example, if a pin associated with port TC is configured, then you would use `ptcpar` (port TC pin assignment register).

Rather than reading hard-coded bits, a special header file called gpio5282.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 types of registers. Each set is divided into subgroups, which indicate what GPIO port register(s) they 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 (PxPAR)

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.

Port Data Direction Register (DDRx)

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 DDRx registers are cleared. Setting any bit to ‘1’ in a DDRx register configures the corresponding GPIO pin as an output. Setting any bit to ‘0’ in a DDRx register configures the corresponding pin as an input.

Port Output Data Register (PORTx)

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 PORTx registers are set. Reserved bits always remain cleared. Reading a PORTx register returns the current values in the register. To set bits in a PORTx 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 PORTx 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 (PORTxP)

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 PORTxP registers are set to the current pin states. Reading a PORTxP 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 (CLRx)

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 CLRx register returns zeros.

Port E Pins

The port E pin assignment register (PEPAR) controls the functions of the R/*W, OE, TIP, and TA pins (MCF5282 user's manual, table 26-9). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTE, DDRE, PORTEP, and CLRE).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PEPAR]	GPIO Control Bit [xxxE(P)]
J1-4	R/*W	Read / not Write	8	4
J1-8	OE	Output Enable	14	7
J1-11	TIP	Transfer in Progress	1-0	0
J1-13	TA	Transfer Acknowledge	12	6

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port E Pin Assignment Register. Set all pins as GPIO.
sim.gpio.pepar &= ~( GPIO_PAR_RW | GPIO_PAR_OE | GPIO_PAR_TIP |
                     GPIO_PAR_TA );

// Port E Data Direction Register. Set OE as an output and TIP as an
// input.
sim.gpio.ddre |= GPIO_PIN_OE;
sim.gpio.ddre &= ~GPIO_PIN_TIP;

// Port E Output Data Register. Set OE low.
sim.gpio.porte &= ~GPIO_PIN_OE;

// Port E Pin Data/Set Data Register. Read the current state on the TIP
// pin.
BYTE value_e = sim.gpio.portep & GPIO_PIN_TIP;
```

Port J Pins

The port J pin assignment register (PJPAR) controls the functions of the Chip Select pins (MCF5282 user's manual, table 26-12). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTJ, DDRJ, PORTJP, and CLRJ).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PJPAR]	GPIO Control Bit [xxxJ(P)]
J1-5	CS1	Chip Select #1	1	1
J1-6	CS2	Chip Select #2	2	2
J1-7	CS3	Chip Select #3	3	3

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port J Pin Assignment Register. Set all pins as GPIO.
sim.gpio.pjpar &= ~( GPIO_PAR_CS1 | GPIO_PAR_CS2 | GPIO_PAR_CS3 );

// Port J Data Direction Register. Set CS1 and CS2 as outputs, and CS3
// as an input.
sim.gpio.ddrj |= ( GPIO_PIN_CS1 | GPIO_PIN_CS2 );
sim.gpio.ddrj &= ~GPIO_PIN_CS3;

// Port J Output Data Register. Set CS1 high and CS2 low.
sim.gpio.portj |= GPIO_PIN_CS1;
sim.gpio.portj &= ~GPIO_PIN_CS2;

// Port J Pin Data/Set Data Register. Read the current state on the CS3
// pin.
BYTE value_j = sim.gpio.portjp & GPIO_PIN_CS3;
```

Port UA Pins

The port UA pin assignment register (PUAPAR) controls the functions of the UART pins (MCF5282 user's manual, table 26-19). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTUA, DDRUA, PORTUAP, and CLRUA).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PUAPAR]	GPIO Control Bit [xxxUA(P)]
J2-3	URXD0	UART 0 – Receive	1	1
J2-4	UTXD0	UART 0 – Transmit	0	0
J2-21	URXD1	UART 1 – Receive	3	3
J2-22	UTXD1	UART 1 – Transmit	2	2

Code examples (Note: `sim5282.h` and `gpio5282.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 URXD1 and UTXD1 as GPIO.
sim.gpio.puapar &= ~( GPIO_PAR_URXD1 | GPIO_PAR_UTXD1 );

// Port UA Data Direction Register. Set URXD1 and UTXD1 as outputs.
sim.gpio.ddrua |= ( GPIO_PIN_URXD1 | GPIO_PIN_UTXD1 );

// Port UA Output Data Register. Set URXD1 low and UTXD1 high.
sim.gpio.portua &= ~GPIO_PIN_URXD1;
sim.gpio.portua |= GPIO_PIN_UTXD1;

// Port UA Pin Data/Set Data Register. Read the current pin state on
// URXD1.
BYTE value_ua = sim.gpio.portuap & GPIO_PIN_URXD1;
```

Port QS Pins

The port QS pin assignment register (PQSPAR) controls the functions of the QSPI pins (MCF5282 user's manual, table 26-16). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTQS, DDRQS, PORTQSP, and CLRQS).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PQSPAR]	GPIO Control Bit [xxxQS(P)]
J2-25	SPI_CLK	SPI Clock	2	2
J2-26	SPI_CS3	SPI Chip Select 3	6	6
J2-27	SPI_DIN	SPI Data In	1	1
J2-28	SPI_DOUT	SPI Data Out	0	0
J2-30	SPI_CS0	SPI Chip Select 0	3	3
J2-35	SPI_CS2	SPI Chip Select 2	5	5
J2-40	SPI_CS1	SPI Chip Select 1	4	4

Code examples (Note: `sim5282.h` and `gpio5282.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 pins as GPIO.
sim.gpio.pqspar &= ( GPIO_PAR_SPI_CLK | GPIO_PAR_SPI_CS3 |
                     GPIO_PAR_SPI_DIN | GPIO_PAR_SPI_DOUT |
                     GPIO_PAR_SPI_CS0 | GPIO_PAR_SPI_CS2 |
                     GPIO_PAR_SPI_CS1 );

// Port QS Data Direction Register. Set the CS pins as inputs and rest
// as outputs.
sim.gpio.ddrqs &= ~( GPIO_PIN_SPI_CS3 | GPIO_PIN_SPI_CS0 |
                     GPIO_PIN_SPI_CS2 | GPIO_PIN_SPI_CS1 );
sim.gpio.ddrqs |= ( GPIO_PIN_SPI_CLK | GPIO_PIN_SPI_DIN |
                    GPIO_PIN_SPI_DOUT );

// Port QS Output Data Register. Set all the configured output pins
// high.
sim.gpio.portqs |= ( GPIO_PIN_SPI_CLK | GPIO_PIN_SPI_DIN |
                     GPIO_PIN_SPI_DOUT );

// Port QS Pin Data/Set Data Register. Read the current pin state of
// the CS pins.
BYTE value_qs = sim.gpio.portqsp & ( GPIO_PIN_SPI_CS3 |
                                     GPIO_PIN_SPI_CS0 |
                                     GPIO_PIN_SPI_CS2 |
                                     GPIO_PIN_SPI_CS1 );
```

Port AS Pins

The port AS pin assignment register (PASPAR) controls the functions of the I^2C and CAN pins (MCF5282 user's manual, table 26-16). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTAS, DDRAS, PORTASP, and CLRAS).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PASPAR]	GPIO Control Bit [xxxAS(P)]
J2-39	SDA	I^2C Serial Data	3-2	1
J2-41	CANRX	CAN – Receive	7-6	3
J2-42	SCL	I^2C Serial Clock	1-0	0
J2-44	CANTX	CAN – Transmit	5-4	2

Code examples (Note: `sim5282.h` and `gpio5282.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 SDA as GPIO.
sim.gpio.paspar &= ~GPIO_PAR_SDA;

// Port AS Data Direction Register. Set SDA as an output.
sim.gpio.ddras |= GPIO_PIN_SDA;

// Port AS Output Data Register. Set SDA low.
sim.gpio.portas &= ~GPIO_PIN_SDA;

// Port AS Pin Data/Set Data Register. Read the current pin state on
// SDA.
BYTE value_as = sim.gpio.portasp & GPIO_PIN_SDA;
```

Port TC Pins

The port TC pin assignment register (PTCPAR) controls the functions of the DMA Timer 2 and 3 pins (MCF5282 user's manual, table 26-17). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTTC, DDRTC, PORTTCP, and CLRTC).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PTCPAR]	GPIO Control Bit [xxxTC(P)]
J2-29	TIN2	DMA Timer Input 2	3-2	1
J2-32	DTOUT3	DMA Timer Output 3	5-4	2
J2-33	DTOUT2	DMA Timer Output 2	1-0	0
J2-38	TIN3	DMA Timer Input 3	7-6	3

Code examples (Note: `sim5282.h` and `gpio5282.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 the DMA Timer 3 pins as GPIO.
sim.gpio.ptcpar &= ~( GPIO_PAR_DTOUT3 | GPIO_PAR_TIN3 );

// Port TC Data Direction Register. Set DTOUT3 as an output and TIN3 as
// an input.
sim.gpio.ddrtc |= GPIO_PIN_DTOUT3;
sim.gpio.ddrtc &= ~GPIO_PIN_TIN3;

// Port TC Output Data Register. Set DTOUT3 high.
sim.gpio.porttc |= GPIO_PIN_DTOUT3;

// Port TC Pin Data/Set Data Register. Read the current pin state on
// TIN3.
BYTE value_tc = sim.gpio.porttcp & GPIO_PIN_TIN3;
```

Port TD Pins

The port TD pin assignment register (PTDPAR) controls the functions of the DMA Timer 0 and 1 pins (MCF5282 user's manual, table 26-18). The GPIO control bits correspond to bit positions in the GPIO control registers (PORTTD, DDRTD, PORTTDP, and CLRTD).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PTDPAR]	GPIO Control Bit [xxxTD(P)]
J2-31	TIN0	DMA Timer Input 0	3-2	1
J2-34	DTOUT1	DMA Timer Output 1	5-4	2
J2-36	DTOUT0	DMA Timer Output 0	1-0	0
J2-37	TIN1	DMA Timer Input 1	7-6	3

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port TD Pin Assignment Register. Set the DMA Timer 0 pins as GPIO.
sim.gpio.ptdpar &= ~( GPIO_PAR_DTOUT0 | GPIO_PAR_TIN0 );

// Port TD Data Direction Register. Set DTOUT0 as an output and TIN0 as
// an input.
sim.gpio.ddrtd |= GPIO_PIN_DTOUT0;
sim.gpio.ddrtd &= ~GPIO_PIN_TIN0;

// Port TD Output Data Register. Set DTOUT0 high.
sim.gpio.porttd |= GPIO_PIN_DTOUT0;

// Port TD Pin Data/Set Data Register. Read the current pin state on
// TIN0.
BYTE value_td = sim.gpio.porttdp & GPIO_PIN_TIN0;
```

Port QA Pins

The port QA pins (QADC analog I/O channels 52, 53, 55, and 56) do not have a pin assignment register (MCF5282 user's manual, chapter 28). There is no need to configure them for GPIO since all pins default to GPIO inputs at reset. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRQA and PORTQA). Note: PORTQA has the same write functionality as a GPIO PORTn register, but also has the read functionality of a PORTnP register (reads port pin values, not the values in the register).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [N/A]	GPIO Control Bit [xxxQA]
J2-9	AN56	Analog I/O Channel 56	-	4
J2-11	AN53	Analog I/O Channel 53	-	1
J2-12	AN52	Analog I/O Channel 52	-	0
J2-13	AN55	Analog I/O Channel 55	-	3

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port QA Data Direction Register. Set AN56 to be an output.
// to be an input.
sim.qadc.ddrqa |= GPIO_PIN_AN56;

// Port QA Data Register. Set AN56 high.
sim.qadc.portqa |= GPIO_PIN_AN56;

// Port QA Data Register. Read the current pin state of AN53. Since it
// is GPIO input by default, there is no need to configure the pin.
BYTE value_qa = sim.qadc.portqa & GPIO_PIN_AN53;
```

Port QB Pins

The port QB pins (QADC analog I/O channels 0, 1, 2, and 3) do not have a pin assignment register (MCF5282 user's manual, chapter 28). There is no need to configure them for GPIO since all pins default to GPIO inputs at reset. The GPIO control bits correspond to bit positions in the GPIO control registers (DDRQB and PORTQB). Note: PORTQB has the same write functionality as a GPIO PORTn register, but also has the read functionality of a PORTnP register (reads port pin values, not the values in the register).

MOD5282 Pin No.	**MOD5282 Signal Name**	**Signal Description**	**Pin Assign Bit(s) [N/A]**	**GPIO Control Bit [xxxQB]**
J2-6	AN3	Analog I/O Channel 3	-	3
J2-7	AN1	Analog I/O Channel 1	-	1
J2-8	AN2	Analog I/O Channel 2	-	2
J2-10	AN0	Analog I/O Channel 0	-	0

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See "Pin Assignment and GPIO Control Registers" above for more information):

```
// Port QB Data Direction Register. Set AN1 to be an output.
// to be an input.
sim.qadc.ddrqb |= GPIO_PIN_AN1;

// Port QB Data Register. Set AN1 low.
sim.qadc.portqb &= ~GPIO_PIN_AN1;

// Port QB Data Register. Read the current pin state of AN0. Since it
// is GPIO input by default, there is no need to configure the pin.
BYTE value_qb = sim.qadc.portqb & GPIO_PIN_AN0;
```

Edge Port Pins

The edge port pin assignment register (EPPAR) controls the functions of the external interrupt pins (MCF5282 user's manual, chapter 11). 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)].

MOD5282 Pin No.	**MOD5282 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: `sim5282.h` and `gpio5282.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 |= ( GPIO_PIN_IRQ1 | GPIO_PIN_IRQ5 );

// Edge Port Data Register. Set GPIO signals IRQ1 and IRQ5 low.
sim.eport.epdr &= ~( GPIO_PIN_IRQ1 | GPIO_PIN_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 & ( GPIO_PIN_IRQ1 | GPIO_PIN_IRQ5 );
```

General Purpose Timer A Pins

The general purpose timer A pins do not have a pin assignment register. The GPTA pins become configured for GPIO when the GPT functionality is disabled via bit 7 in the GPT System Control Register 1 (MCF5282, section 20.5.6). All GPTA pins become GPIO when the bit is cleared; they cannot be configured for GPIO or their primary function individually. The GPIO control bits correspond to bit positions in the GPIO control registers (DDR and PORT). Note: PORT has the same write functionality as a GPIO PORTn register, but also has the read functionality of a PORTnP register (reads port pin values, not the values in the register).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Sys Control Bit {GPT[0].SCR1}	GPIO Control Bit [DDR / PORT]
J2-15	GPTA3	General Purpose Timer A3	7	3
J2-17	GPTA2	General Purpose Timer A2	7	2
J2-19	GPTA1	General Purpose Timer A1	7	1
J2-23	GPTA0	General Purpose Timer A0	7	0

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// GPTA System Control Register 1. Set all pins as GPIO.
sim.gpt[0].scr1 &= ~GPIO_SCR_GPTA;

// GPTA Data Direction Register. Set GPTA3 and GPTA2 as outputs, and
// GPTA1 and GPTA0 as inputs.
sim.gpt[0].ddr |= ( GPIO_PIN_GPTA3 | GPIO_PIN_GPTA2 );
sim.gpt[0].ddr &= ~( GPIO_PIN_GPTA1 | GPIO_PIN_GPTA0 );

// GPTA Port Data Register. Set GPTA3 high and GPTA2 low.
sim.gpt[0].port |= GPIO_PIN_GPTA3;
sim.gpt[0].port &= ~GPIO_PIN_GPTA2;

// GPTA Port Data Register. Read the current pin states of GPTA1 and
// GPTA0.
BYTE value_gpta = sim.gpt[0].port & ( GPIO_PIN_GPTA1 |
                                      GPIO_PIN_GPTA0 );
```

General Purpose Timer B Pins

The general purpose timer B pins do not have a pin assignment register. The GPTB pins become configured for GPIO when the GPT functionality is disabled via bit 7 in the GPT System Control Register 1 (MCF5282, section 20.5.6). All GPTA pins become GPIO when the bit is cleared; they cannot be configured for GPIO or their primary function individually. The GPIO control bits correspond to bit positions in the GPIO control registers (DDR and PORT). Note: PORT has the same write functionality as a GPIO PORTn register, but also has the read functionality of a PORTnP register (reads port pin values, not the values in the register).

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Sys Control Bit {GPT[1].SCR1}	GPIO Control Bit [DDR / PORT]
J2-16	GPTB3	General Purpose Timer B3	7	3
J2-18	GPTB2	General Purpose Timer B2	7	2
J2-20	GPTB1	General Purpose Timer B1	7	1
J2-24	GPTB0	General Purpose Timer B0	7	0

Code examples (Note: `sim5282.h` and `gpio5282.h` header files must be included for the following examples to work. See “Pin Assignment and GPIO Control Registers” above for more information):

```
// GPTB System Control Register 1. Set all pins as GPIO.
sim.gpt[1].scr1 &= ~GPIO_SCR_GPTB;

// GPTB Data Direction Register. Set GPTB3 and GPTB2 as outputs, and
// GPTB1 and GPTB0 as inputs.
sim.gpt[1].ddr |= ( GPIO_PIN_GPTB3 | GPIO_PIN_GPTB2 );
sim.gpt[1].ddr &= ~( GPIO_PIN_GPTB1 | GPIO_PIN_GPTB0 );

// GPTB Port Data Register. Set GPTB3 high and GPTB2 low.
sim.gpt[1].port |= GPIO_PIN_GPTB3;
sim.gpt[1].port &= ~GPIO_PIN_GPTB2;

// GPTB Port Data Register. Read the current pin states of GPTB1 and
// GPTB0.
BYTE value_gptb = sim.gpt[1].port & ( GPIO_PIN_GPTB1 |
                                      GPIO_PIN_GPTB0 );
```

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 TD 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) :

MOD5282 Pin No.	MOD5282 Signal Name	Signal Description	Pin Assign Bit(s) [PTCPAR]	GPIO Control Bit [xxxTC(P)]
J2-29	TIN2	DMA Timer Input 2	3-2	1
J2-32	DTOUT3	DMA Timer Output 3	5-4	2
J2-33	DTOUT2	DMA Timer Output 2	1-0	0
J2-38	TIN3	DMA Timer Input 3	7-6	3

```
// 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 DMA Timer Inputs as
// GPIO outputs and the DMA Timer Inputs as GPIO inputs.
sim.gpio.ddrtc = 0x0A;     // 0000 1010
```

In the above example, all four pins belonging to Port TC 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.