15 GPIO

15.1 Introduction

The GPIO module provides control for up to eight independent GPIO pins (the actual number present depend upon the GPIO port and part number). Each pin has the following capabilities:

- Can be configured as an input or an output. On reset, GPIOs default to being inputs.
- In input mode, can generate interrupts on high level, low level, rising edge, falling edge, or both edges.
- In output mode, can be configured for 2-mA, 4-mA, or 8-mA drive strength. The 8-mA drive strength configuration has optional slew rate control to limit the rise and fall times of the signal. On reset, GPIOs default to 2-mA drive strength.
- Optional weak pull-up or pull-down resistors. On reset, GPIOs default to no pull-up or pull-down resistors.
- Optional open-drain operation. On reset, GPIOs default to standard push/pull operation.
- Can be configured to be a GPIO or a peripheral pin. On reset, the default is GPIO. Note that not all pins on all parts have peripheral functions, in which case the pin is only useful as a GPIO.

Most of the GPIO functions can operate on more than one GPIO pin (within a single module) at a time. The *ucPins* parameter to these functions is used to specify the pins that are affected; only the GPIO pins corresponding to the bits in this parameter that are set are affected (where pin 0 is bit 0, pin 1 in bit 1, and so on). For example, if *ucPins* is 0x09, then pins 0 and 3 are affected by the function.

This protocol is most useful for the GPIOPinRead() and GPIOPinWrite() functions; a read returns only the values of the requested pins (with the other pin values masked out) and a write only affects the requested pins simultaneously (that is, the state of multiple GPIO pins can be changed at the same time). This data masking for the GPIO pin state occurs in the hardware; a single read or write is issued to the hardware, which interprets some of the address bits as an indication of the GPIO pins to operate on (and therefore the ones to not affect). See the part data sheet for details of the GPIO data register address-based bit masking.

For functions that have a *ucPin* (singular) parameter, only a single pin is affected by the function. In this case, the value specifies the pin number (that is, 0 through 7).

NOTE: A subset of GPIO pins on many Tiva devices are protected by a locking mechanism to prevent inadvertent reconfiguration. The actual pins vary by device but typically include any pin that is part of the JTAG or SWD interface, and any pin which may be configured as an NMI input. On a TM4C129XNCZAD part, for example, this affects pins PC[3:0], PD7 and PE7. Locked pins may not be reconfigured without first unlocking them using the mechanism described under “Commit Control” in the GPIO chapter of your device’s datasheet. This mechanism is also illustrated in the TivaWare “gpio_jtag” example application included for all target evaluation and development kits.

This driver is contained in `driverlib/gpio.c`, with `driverlib/gpio.h` containing the API declarations for use by applications.

15.2 API Functions

Functions

- void GPIOADCTriggerDisable (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOADCTriggerEnable (uint32_t ui32Port, uint8_t ui8Pins)
- uint32_t GPIODirModeGet (uint32_t ui32Port, uint8_t ui8Pin)
- void GPIODirModeSet (uint32_t ui32Port, uint8_t ui8Pins, uint32_t ui32PinIO)
- void GPIODMATriggerDisable (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIODMATriggerEnable (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOIntClear (uint32_t ui32Port, uint32_t ui32IntFlags)
- void GPIOIntDisable (uint32_t ui32Port, uint32_t ui32IntFlags)
- void GPIOIntEnable (uint32_t ui32Port, uint32_t ui32IntFlags)
- void GPIOIntRegister (uint32_t ui32Port, void (*pfnIntHandler)(void))
- uint32_t GPIOIntStatus (uint32_t ui32Port, bool bMasked)
- uint32_t GPIOIntTypeGet (uint32_t ui32Port, uint8_t ui8Pin)
- void GPIOIntTypeSet (uint32_t ui32Port, uint8_t ui8Pins, uint32_t ui32IntType)
- void GPIOIntUnregister (uint32_t ui32Port)
- void GPIOPadConfigGet (uint32_t ui32Port, uint8_t ui8Pin, uint32_t *pui32Strength, uint32_t *pui32PinType)
- void GPIOPadConfigSet (uint32_t ui32Port, uint8_t ui8Pins, uint32_t ui32Strength, uint32_t ui32PinType)
- void GPIOPinConfigure (uint32_t ui32PinConfig)
- int32_t GPIOPinRead (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeADC (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeCAN (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeCIR (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeComparator (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeComparatorOutput (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeEPI (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeEthernetLED (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeEthernetMII (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeGPIOInput (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeGPIOOutput (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeGPIOOutputOD (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeI2C (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeI2CSCL (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeKBColumn (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeKBRow (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeLCD (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeLEDSeq (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeLPC (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypePECIRx (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypePECITx (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypePWM (uint32_t ui32Port, uint8_t ui8Pins)

- void GPIOPinTypeQEI (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeSSI (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeTimer (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeUART (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeUSBAnalog (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeUSBDigital (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeWakeHigh (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeWakeLow (uint32_t ui32Port, uint8_t ui8Pins)
- uint32_t GPIOPinWakeStatus (uint32_t ui32Port)
- void GPIOPinWrite (uint32_t ui32Port, uint8_t ui8Pins, uint8_t ui8Val)

15.2.1 Detailed Description

The GPIO API is broken into three groups of functions: those that deal with configuring the GPIO pins, those that deal with interrupts, and those that access the pin value.

The GPIO pins are configured with GPIODirModeSet(), GPIOPadConfigSet(), and GPIOPinConfigure(). The configuration can be read back with GPIODirModeGet() and GPIOPadConfigGet().

The GPIO pin state is accessed with GPIOPinRead() and GPIOPinWrite().

The GPIO interrupts are handled with GPIOIntTypeSet(), GPIOIntTypeGet(), GPIOIntEnable(), GPIOIntDisable(), GPIOIntStatus(), GPIOIntClear(), GPIOIntRegister(), and GPIOIntUnregister().

15.2.2 GPIO Pin Configuration

Many of the GPIO pins on the TM4C123 and TM4C129 devices have other peripheral functions that can also use the GPIO pins for peripheral pins. The Peripheral Driver Library provides a set of convenience functions to configure the pins in the required or recommended input/output configuration for a particular peripheral; these are GPIOPinTypeADC(), GPIOPinTypeCAN(), GPIOPinTypeComparator(), GPIOPinTypeEPI(), GPIOPinTypeEthernetLED(), GPIOPinTypeEthernetMII(), GPIOPinTypeGPIOInput(), GPIOPinTypeGPIOOutput(), GPIOPinTypeGPIOOutputOD(), GPIOPinTypeI2C(), GPIOPinTypeI2CSCL(), GPIOPinTypeLCD() GPIOPinTypePWM(), GPIOPinTypeQEI(), GPIOPinTypeSSI(), GPIOPinTypeTimer(), GPIOPinTypeUART(), GPIOPinTypeUSBAnalog(), GPIOPinTypeUSBDigital(), GPIOPinTypeWakeHigh(), GPIOPinTypeWakeLow(), GPIOPinWakeStatus(), GPIODMATriggerEnable(), GPIODMATriggerDisable(), GPIOADCTriggerEnable(), and GPIOADCTriggerDisable(). In order to complete the pin configuration, the GPIOPinConfigure() function must also be called to enable the desired peripheral function on the given GPIO pin. The GPIOPinConfigure() function uses the pin definitions located in the `driverlib/pin_map.h` file. These definitions follow the **GPIO_P**<**port**><**pin**>_<**peripheral_function**> naming scheme. The available pin mappings are supplied on a per-device basis and are selected using the **PART_**<**partno**> defines to enable only the pin definitions that are valid for the given device. For example, on the TM4C129XNCZAD device the UART1 RX function can be enabled on one of two pins. The UART1 RX is found on GPIO port B pin 0(**GPIO_PB0_U1RX**) or it can also be found on GPIO port Q pin 4(**GPIO_PQ4_U1RX**). The application must define the **PART_TM4C129XNCZAD** in order to get the correct pin mappings for the TM4C129XNCZAD device.

Note:
The **PART_**<**partno**> macros also control the mapping of interrupt names to interrupt numbers. See the Interrupt Mapping section of this document for more details on how these defines

are used to determine interrupt mapping.

A locked GPIO pin must be unlocked prior to making calls to GPIODirModeSet(), GPIOPadConfigSet() or any of the GPIOPinType functions.

15.2.3 Function Documentation

15.2.3.1 GPIOADCTriggerDisable

Disable a GPIO pin as a trigger to start an ADC capture.

Prototype:

```
void
GPIOADCTriggerDisable(uint32_t ui32Port,
                      uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

This function disables a GPIO pin to be used as a trigger to start an ADC sequence. This function can be used to disable this feature if it was enabled via a call to GPIOADCTriggerEnable().

Returns:

None.

15.2.3.2 GPIOADCTriggerEnable

Enables a GPIO pin as a trigger to start an ADC capture.

Prototype:

```
void
GPIOADCTriggerEnable(uint32_t ui32Port,
                     uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

This function enables a GPIO pin to be used as a trigger to start an ADC sequence. Any GPIO pin can be configured to be an external trigger for an ADC sequence. The GPIO pin still generates interrupts if the interrupt is enabled for the selected pin. To enable the use of a GPIO pin to trigger the ADC module, the ADCSequenceConfigure() function must be called with the **ADC_TRIGGER_EXTERNAL** parameter.

Returns:

None.

15.2.3.3 GPIODirModeGet

Gets the direction and mode of a pin.

Prototype:

```
uint32_t
GPIODirModeGet(uint32_t ui32Port,
               uint8_t ui8Pin)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pin is the pin number.

Description:

This function gets the direction and control mode for a specified pin on the selected GPIO port. The pin can be configured as either an input or output under software control, or it can be under hardware control. The type of control and direction are returned as an enumerated data type.

Returns:

Returns one of the enumerated data types described for GPIODirModeSet().

15.2.3.4 GPIODirModeSet

Sets the direction and mode of the specified pin(s).

Prototype:

```
void
GPIODirModeSet(uint32_t ui32Port,
               uint8_t ui8Pins,
               uint32_t ui32PinIO)
```

Parameters:

ui32Port is the base address of the GPIO port

ui8Pins is the bit-packed representation of the pin(s).

ui32PinIO is the pin direction and/or mode.

Description:

This function configures the specified pin(s) on the selected GPIO port as either input or output under software control, or it configures the pin to be under hardware control.

The parameter *ui32PinIO* is an enumerated data type that can be one of the following values:

- **GPIO_DIR_MODE_IN**
- **GPIO_DIR_MODE_OUT**
- **GPIO_DIR_MODE_HW**

where **GPIO_DIR_MODE_IN** specifies that the pin is programmed as a software controlled input, **GPIO_DIR_MODE_OUT** specifies that the pin is programmed as a software controlled output, and **GPIO_DIR_MODE_HW** specifies that the pin is placed under hardware control.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
GPIOPadConfigSet() must also be used to configure the corresponding pad(s) in order for them to propagate the signal to/from the GPIO.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.5 GPIODMATriggerDisable

Disables a GPIO pin as a trigger to start a DMA transaction.

Prototype:
```
void
GPIODMATriggerDisable(uint32_t ui32Port,
                      uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
This function disables a GPIO pin from being used as a trigger to start a uDMA transaction. This function can be used to disable this feature if it was enabled via a call to GPIODMATriggerEnable().

Returns:
None.

15.2.3.6 GPIODMATriggerEnable

Enables a GPIO pin as a trigger to start a DMA transaction.

Prototype:
```
void
GPIODMATriggerEnable(uint32_t ui32Port,
                     uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
This function enables a GPIO pin to be used as a trigger to start a uDMA transaction. Any GPIO pin can be configured to be an external trigger for the uDMA. The GPIO pin still generates interrupts if the interrupt is enabled for the selected pin.

Returns:
None.

15.2.3.7 GPIOIntClear

Clears the specified interrupt sources.

Prototype:
```
void
GPIOIntClear(uint32_t ui32Port,
             uint32_t ui32IntFlags)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui32IntFlags is the bit mask of the interrupt sources to disable.

Description:
Clears the interrupt for the specified interrupt source(s).

The *ui32IntFlags* parameter is the logical OR of the **GPIO_INT_*** values.

Note:
Because there is a write buffer in the Cortex-M processor, it may take several clock cycles before the interrupt source is actually cleared. Therefore, it is recommended that the interrupt source be cleared early in the interrupt handler (as opposed to the very last action) to avoid returning from the interrupt handler before the interrupt source is actually cleared. Failure to do so may result in the interrupt handler being immediately reentered (because the interrupt controller still sees the interrupt source asserted).

Returns:
None.

15.2.3.8 GPIOIntDisable

Disables the specified GPIO interrupts.

Prototype:
```
void
GPIOIntDisable(uint32_t ui32Port,
               uint32_t ui32IntFlags)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui32IntFlags is the bit mask of the interrupt sources to disable.

Description:
This function disables the indicated GPIO interrupt sources. Only the sources that are enabled can be reflected to the processor interrupt; disabled sources have no effect on the processor.

The *ui32IntFlags* parameter is the logical OR of any of the following:

- **GPIO_INT_PIN_0** - interrupt due to activity on Pin 0.

- **GPIO_INT_PIN_1** - interrupt due to activity on Pin 1.
- **GPIO_INT_PIN_2** - interrupt due to activity on Pin 2.
- **GPIO_INT_PIN_3** - interrupt due to activity on Pin 3.
- **GPIO_INT_PIN_4** - interrupt due to activity on Pin 4.
- **GPIO_INT_PIN_5** - interrupt due to activity on Pin 5.
- **GPIO_INT_PIN_6** - interrupt due to activity on Pin 6.
- **GPIO_INT_PIN_7** - interrupt due to activity on Pin 7.
- **GPIO_INT_DMA** - interrupt due to DMA activity on this GPIO module.

Returns:
None.

15.2.3.9 GPIOIntEnable

Enables the specified GPIO interrupts.

Prototype:
```
void
GPIOIntEnable(uint32_t ui32Port,
              uint32_t ui32IntFlags)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui32IntFlags is the bit mask of the interrupt sources to enable.

Description:
This function enables the indicated GPIO interrupt sources. Only the sources that are enabled can be reflected to the processor interrupt; disabled sources have no effect on the processor.

The *ui32IntFlags* parameter is the logical OR of any of the following:

- **GPIO_INT_PIN_0** - interrupt due to activity on Pin 0.
- **GPIO_INT_PIN_1** - interrupt due to activity on Pin 1.
- **GPIO_INT_PIN_2** - interrupt due to activity on Pin 2.
- **GPIO_INT_PIN_3** - interrupt due to activity on Pin 3.
- **GPIO_INT_PIN_4** - interrupt due to activity on Pin 4.
- **GPIO_INT_PIN_5** - interrupt due to activity on Pin 5.
- **GPIO_INT_PIN_6** - interrupt due to activity on Pin 6.
- **GPIO_INT_PIN_7** - interrupt due to activity on Pin 7.
- **GPIO_INT_DMA** - interrupt due to DMA activity on this GPIO module.

Note:
If this call is being used to enable summary interrupts on GPIO port P or Q (GPIOIntTypeSet() with GPIO_DISCRETE_INT not enabled), then all individual interrupts for these ports must be enabled in the GPIO module using GPIOIntEnable() and all but the interrupt for pin 0 must be disabled in the NVIC using the IntDisable() function. The summary interrupts for the ports are routed to the INT_GPIOP0 or INT_GPIOQ0 which must be enabled to handle the interrupt. If this is not done then any individual GPIO pin interrupts that are left enabled also trigger the individual interrupts.

Returns:
None.

15.2.3.10 GPIOIntRegister

Registers an interrupt handler for a GPIO port.

Prototype:

```
void
GPIOIntRegister(uint32_t ui32Port,
                void (*pfnIntHandler)(void))
```

Parameters:

ui32Port is the base address of the GPIO port.

pfnIntHandler is a pointer to the GPIO port interrupt handling function.

Description:

This function ensures that the interrupt handler specified by *pfnIntHandler* is called when an interrupt is detected from the selected GPIO port. This function also enables the corresponding GPIO interrupt in the interrupt controller; individual pin interrupts and interrupt sources must be enabled with GPIOIntEnable().

See also:

IntRegister() for important information about registering interrupt handlers.

Returns:

None.

15.2.3.11 GPIOIntStatus

Gets interrupt status for the specified GPIO port.

Prototype:

```
uint32_t
GPIOIntStatus(uint32_t ui32Port,
              bool bMasked)
```

Parameters:

ui32Port is the base address of the GPIO port.

bMasked specifies whether masked or raw interrupt status is returned.

Description:

If *bMasked* is set as **true**, then the masked interrupt status is returned; otherwise, the raw interrupt status is returned.

Returns:

Returns the current interrupt status for the specified GPIO module. The value returned is the logical OR of the **GPIO_INT_*** values that are currently active.

15.2.3.12 GPIOIntTypeGet

Gets the interrupt type for a pin.

Prototype:

```
uint32_t
GPIOIntTypeGet(uint32_t ui32Port,
               uint8_t ui8Pin)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pin is the pin number.

Description:

This function gets the interrupt type for a specified pin on the selected GPIO port. The pin can be configured as a falling-edge, rising-edge, or both-edges detected interrupt, or it can be configured as a low-level or high-level detected interrupt. The type of interrupt detection mechanism is returned and can include the **GPIO_DISCRETE_INT** flag.

Returns:

Returns one of the flags described for GPIOIntTypeSet().

15.2.3.13 GPIOIntTypeSet

Sets the interrupt type for the specified pin(s).

Prototype:

```
void
GPIOIntTypeSet(uint32_t ui32Port,
               uint8_t ui8Pins,
               uint32_t ui32IntType)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

ui32IntType specifies the type of interrupt trigger mechanism.

Description:

This function sets up the various interrupt trigger mechanisms for the specified pin(s) on the selected GPIO port.

One of the following flags can be used to define the *ui32IntType* parameter:

- **GPIO_FALLING_EDGE** sets detection to edge and trigger to falling
- **GPIO_RISING_EDGE** sets detection to edge and trigger to rising
- **GPIO_BOTH_EDGES** sets detection to both edges
- **GPIO_LOW_LEVEL** sets detection to low level
- **GPIO_HIGH_LEVEL** sets detection to high level

In addition to the above flags, the following flag can be OR'd in to the *ui32IntType* parameter:

- **GPIO_DISCRETE_INT** sets discrete interrupts for each pin on a GPIO port.

The **GPIO_DISCRETE_INT** is not available on all devices or all GPIO ports, consult the data sheet to ensure that the device and the GPIO port supports discrete interrupts.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
In order to avoid any spurious interrupts, the user must ensure that the GPIO inputs remain stable for the duration of this function.

Returns:
None.

15.2.3.14 GPIOIntUnregister

Removes an interrupt handler for a GPIO port.

Prototype:
```
void
GPIOIntUnregister(uint32_t ui32Port)
```

Parameters:
ui32Port is the base address of the GPIO port.

Description:
This function unregisters the interrupt handler for the specified GPIO port. This function also disables the corresponding GPIO port interrupt in the interrupt controller; individual GPIO interrupts and interrupt sources must be disabled with GPIOIntDisable().

See also:
IntRegister() for important information about registering interrupt handlers.

Returns:
None.

15.2.3.15 GPIOPadConfigGet

Gets the pad configuration for a pin.

Prototype:
```
void
GPIOPadConfigGet(uint32_t ui32Port,
                 uint8_t ui8Pin,
                 uint32_t *pui32Strength,
                 uint32_t *pui32PinType)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pin is the pin number.
pui32Strength is a pointer to storage for the output drive strength.
pui32PinType is a pointer to storage for the output drive type.

Description:
This function gets the pad configuration for a specified pin on the selected GPIO port. The values returned in *pui32Strength* and *pui32PinType* correspond to the values used in GPIOPadConfigSet(). This function also works for pin(s) configured as input pin(s); however, the only meaningful data returned is whether the pin is terminated with a pull-up or down resistor.

Returns:
None

15.2.3.16 GPIOPadConfigSet

Sets the pad configuration for the specified pin(s).

Prototype:

```
void
GPIOPadConfigSet(uint32_t ui32Port,
                 uint8_t ui8Pins,
                 uint32_t ui32Strength,
                 uint32_t ui32PinType)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

ui32Strength specifies the output drive strength.

ui32PinType specifies the pin type.

Description:

This function sets the drive strength and type for the specified pin(s) on the selected GPIO port. For pin(s) configured as input ports, the pad is configured as requested, but the only real effect on the input is the configuration of the pull-up or pull-down termination.

The parameter *ui32Strength* can be one of the following values:

- **GPIO_STRENGTH_2MA**
- **GPIO_STRENGTH_4MA**
- **GPIO_STRENGTH_8MA**
- **GPIO_STRENGTH_8MA_SC**
- **GPIO_STRENGTH_6MA**
- **GPIO_STRENGTH_10MA**
- **GPIO_STRENGTH_12MA**

where **GPIO_STRENGTH_xMA** specifies either 2, 4, or 8 mA output drive strength, and **GPIO_OUT_STRENGTH_8MA_SC** specifies 8 mA output drive with slew control.

Some Tiva devices also support output drive strengths of 6, 10, and 12 mA.

The parameter *ui32PinType* can be one of the following values:

- **GPIO_PIN_TYPE_STD**
- **GPIO_PIN_TYPE_STD_WPU**
- **GPIO_PIN_TYPE_STD_WPD**
- **GPIO_PIN_TYPE_OD**
- **GPIO_PIN_TYPE_ANALOG**
- **GPIO_PIN_TYPE_WAKE_HIGH**
- **GPIO_PIN_TYPE_WAKE_LOW**

where **GPIO_PIN_TYPE_STD**$*$ specifies a push-pull pin, **GPIO_PIN_TYPE_OD**$*$ specifies an open-drain pin, $*$**_WPU** specifies a weak pull-up, $*$**_WPD** specifies a weak pull-down, and **GPIO_PIN_TYPE_ANALOG** specifies an analog input.

The **GPIO_PIN_TYPE_WAKE_*** settings specify the pin to be used as a hibernation wake source. The pin sense level can be high or low. These settings are only available on some Tiva devices.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.17 GPIOPinConfigure

Configures the alternate function of a GPIO pin.

Prototype:
```
void
GPIOPinConfigure(uint32_t ui32PinConfig)
```

Parameters:
ui32PinConfig is the pin configuration value, specified as only one of the **GPIO_P**??_??? values.

Description:
This function configures the pin mux that selects the peripheral function associated with a particular GPIO pin. Only one peripheral function at a time can be associated with a GPIO pin, and each peripheral function should only be associated with a single GPIO pin at a time (despite the fact that many of them can be associated with more than one GPIO pin). To fully configure a pin, a GPIOPinType*() function should also be called.

The available mappings are supplied on a per-device basis in `pin_map.h`. The **PART_**<**partno**> defines controls which set of defines are included so that they match the device that is being used. For example, **PART_TM4C129XNCZAD** must be defined in order to get the correct pin mappings for the TM4C129XNCZAD device.

Note:
If the same signal is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter and the assignment to the higher letter port is ignored.

Returns:
None.

15.2.3.18 GPIOPinRead

Reads the values present of the specified pin(s).

Prototype:

```
int32_t
GPIOPinRead(uint32_t ui32Port,
            uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The values at the specified pin(s) are read, as specified by *ui8Pins*. Values are returned for both input and output pin(s), and the value for pin(s) that are not specified by *ui8Pins* are set to 0.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Returns:

Returns a bit-packed byte providing the state of the specified pin, where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on. Any bit that is not specified by *ui8Pins* is returned as a 0. Bits 31:8 should be ignored.

15.2.3.19 GPIOPinTypeADC

Configures pin(s) for use as analog-to-digital converter inputs.

Prototype:

```
void
GPIOPinTypeADC(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The analog-to-digital converter input pins must be properly configured for the analog-to-digital peripheral to function correctly. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an ADC input; it only configures an ADC input pin for proper operation.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration.

These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.20 GPIOPinTypeCAN

Configures pin(s) for use as a CAN device.

Prototype:
```
void
GPIOPinTypeCAN(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The CAN pins must be properly configured for the CAN peripherals to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into a CAN pin; it only configures a CAN pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the CAN function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.21 GPIOPinTypeCIR

Configures pin(s) for use as Consumer Infrared inputs or outputs.

Prototype:
```
void
GPIOPinTypeCIR(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as Consumer Infrared pins. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a CIR pin; it only configures a CIR pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the Consumer Infrared function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.22 GPIOPinTypeComparator

Configures pin(s) for use as an analog comparator input.

Prototype:

```
void
GPIOPinTypeComparator(uint32_t ui32Port,
                      uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The analog comparator input pins must be properly configured for the analog comparator to function correctly. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an analog comparator input; it only configures an analog comparator pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the analog comparator function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration.

These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.23 GPIOPinTypeComparatorOutput

Configures pin(s) for use as an analog comparator output.

Prototype:
```
void
GPIOPinTypeComparatorOutput(uint32_t ui32Port,
                            uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The analog comparator output pins must be properly configured for the analog comparator to function correctly. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Returns:
None.

15.2.3.24 GPIOPinTypeEPI

Configures pin(s) for use by the external peripheral interface.

Prototype:
```
void
GPIOPinTypeEPI(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The external peripheral interface pins must be properly configured for the external peripheral interface to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an external peripheral interface pin; it only configures an external peripheral interface pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the external peripheral interface function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.25 GPIOPinTypeEthernetLED

Configures pin(s) for use by the Ethernet peripheral as LED signals.

Prototype:

```
void
GPIOPinTypeEthernetLED(uint32_t ui32Port,
                       uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The Ethernet peripheral provides four signals that can be used to drive an LED (for example, for link status/activity). This function provides a typical configuration for the pins.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an Ethernet LED pin; it only configures an Ethernet LED pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure the pin for the Ethernet LED function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.26 GPIOPinTypeEthernetMII

Configures pin(s) for use by the Ethernet peripheral as MII signals.

Prototype:
```
void
GPIOPinTypeEthernetMII(uint32_t ui32Port,
                       uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The Ethernet peripheral on some parts provides a set of MII signals that are used to connect to an external PHY. This function provides a typical configuration for the pins.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into an Ethernet MII pin; it only configures an Ethernet MII pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure the pin for the Ethernet MII function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.27 GPIOPinTypeGPIOInput

Configures pin(s) for use as GPIO inputs.

Prototype:
```
void
GPIOPinTypeGPIOInput(uint32_t ui32Port,
                     uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as GPIO inputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.28 GPIOPinTypeGPIOOutput

Configures pin(s) for use as GPIO outputs.

Prototype:

```
void
GPIOPinTypeGPIOOutput(uint32_t ui32Port,
                      uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as GPIO outputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.29 GPIOPinTypeGPIOOutputOD

Configures pin(s) for use as GPIO open drain outputs.

Prototype:

```
void
GPIOPinTypeGPIOOutputOD(uint32_t ui32Port,
                        uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as GPIO outputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.30 GPIOPinTypeI2C

Configures pin for use as SDA by the I2C peripheral.

Prototype:

```
void
GPIOPinTypeI2C(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin.

Description:

The I2C pins must be properly configured for the I2C peripheral to function correctly. This function provides the proper configuration for the SDA pin.

The pin is specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into an I2C SDA pin; it only configures an I2C SDA pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the I2C SDA function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.31 GPIOPinTypeI2CSCL

Configures pin for use as SCL by the I2C peripheral.

Prototype:
```
void
GPIOPinTypeI2CSCL(uint32_t ui32Port,
                  uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin.

Description:
The I2C pins must be properly configured for the I2C peripheral to function correctly. This function provides the proper configuration for the SCL pin.

The pin is specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into an I2C SCL pin; it only configures an I2C SCL pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the I2C SCL function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.32 GPIOPinTypeKBColumn

Configures pin(s) for use as scan matrix keyboard columns (inputs).

Prototype:

```
void
GPIOPinTypeKBColumn(uint32_t ui32Port,
                    uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as scan matrix keyboard inputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a scan matrix keyboard column pin; it only configures a scan matrix keyboard column pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the scan matrix keyboard function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.33 GPIOPinTypeKBRow

Configures pin(s) for use as scan matrix keyboard rows (outputs).

Prototype:

```
void
GPIOPinTypeKBRow(uint32_t ui32Port,
                 uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as scan matrix keyboard outputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a scan matrix keyboard row pin; it only configures a scan matrix keyboard row pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the scan matrix keyboard function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.34 GPIOPinTypeLCD

Configures pin(s) for use by the LCD Controller.

Prototype:

```
void
GPIOPinTypeLCD(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The LCD controller pins must be properly configured for the LCD controller to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an LCD pin; it only configures an LCD pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the LCD controller function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.35 GPIOPinTypeLEDSeq

Configures pin(s) for use as an LED sequencer output.

Prototype:
```
void
GPIOPinTypeLEDSeq(uint32_t ui32Port,
                  uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The GPIO pins must be properly configured in order to function correctly as LED sequencers. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into an LED sequencer output pin; it only configures an LED sequencer output pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the LED sequencer function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.36 GPIOPinTypeLPC

Configures pin(s) for use by the LPC module.

Prototype:
```
void
GPIOPinTypeLPC(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:

The LPC pins must be properly configured for the LPC module to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into an LPC pin; it only configures an LPC pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the LPC function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.37 GPIOPinTypePECIRx

Configures a pin for receive use by the PECI module.

Prototype:

```
void
GPIOPinTypePECIRx(uint32_t ui32Port,
                  uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The PECI receive pin must be properly configured for the PECI module to function correctly. This function provides a typical configuration for that pin.

The pin is specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a PECI receive pin; it only configures a PECI receive pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the PECI receive function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example

application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.38 GPIOPinTypePECITx

Configures a pin for transmit use by the PECI module.

Prototype:
```
void
GPIOPinTypePECITx(uint32_t ui32Port,
                  uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The PECI transmit pin must be properly configured for the PECI module to function correctly. This function provides a typical configuration for that pin.

The pin is specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into a PECI transmit pin; it only configures a PECI transmit pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure the pin for the PECI transmit function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.39 GPIOPinTypePWM

Configures pin(s) for use by the PWM peripheral.

Prototype:
```
void
GPIOPinTypePWM(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The PWM pins must be properly configured for the PWM peripheral to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a PWM pin; it only configures a PWM pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the PWM function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.40 GPIOPinTypeQEI

Configures pin(s) for use by the QEI peripheral.

Prototype:

```
void
GPIOPinTypeQEI(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The QEI pins must be properly configured for the QEI peripheral to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, not using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a QEI pin; it only configures a QEI pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the QEI function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.41 GPIOPinTypeSSI

Configures pin(s) for use by the SSI peripheral.

Prototype:
```
void
GPIOPinTypeSSI(uint32_t ui32Port,
               uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
The SSI pins must be properly configured for the SSI peripheral to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into a SSI pin; it only configures a SSI pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the SSI function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.42 GPIOPinTypeTimer

Configures pin(s) for use by the Timer peripheral.

Prototype:

```
void
GPIOPinTypeTimer(uint32_t ui32Port,
                 uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The CCP pins must be properly configured for the timer peripheral to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a timer pin; it only configures a timer pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the CCP function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.43 GPIOPinTypeUART

Configures pin(s) for use by the UART peripheral.

Prototype:

```
void
GPIOPinTypeUART(uint32_t ui32Port,
                uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The UART pins must be properly configured for the UART peripheral to function correctly. This function provides a typical configuration for those pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into a UART pin; it only configures a UART pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the UART function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.44 GPIOPinTypeUSBAnalog

Configures pin(s) for use by the USB peripheral.

Prototype:
```
void
GPIOPinTypeUSBAnalog(uint32_t ui32Port,
                     uint8_t ui8Pins)
```

Parameters:
ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).

Description:
USB analog pins must be properly configured for the USB peripheral to function correctly. This function provides the proper configuration for any USB analog pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:
This function cannot be used to turn any pin into a USB pin; it only configures a USB pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the USB function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the "gpio_jtag" example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:
None.

15.2.3.45 GPIOPinTypeUSBDigital

Configures pin(s) for use by the USB peripheral.

Prototype:

```
void
GPIOPinTypeUSBDigital(uint32_t ui32Port,
                      uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

USB digital pins must be properly configured for the USB peripheral to function correctly. This function provides a typical configuration for the digital USB pin(s); other configurations may work as well depending upon the board setup (for example, using the on-chip pull-ups).

This function should only be used with EPEN and PFAULT pins as all other USB pins are analog in nature or are not used in devices without OTG functionality.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

This function cannot be used to turn any pin into a USB pin; it only configures a USB pin for proper operation. Note that a GPIOPinConfigure() function call is also required to properly configure a pin for the USB function.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.46 GPIOPinTypeWakeHigh

Configures pin(s) for use as a hibernate wake-on-high source.

Prototype:

```
void
GPIOPinTypeWakeHigh(uint32_t ui32Port,
                    uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as hibernate wake-high inputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.47 GPIOPinTypeWakeLow

Configures pin(s) for use as a hibernate wake-on-low source.

Prototype:

```
void
GPIOPinTypeWakeLow(uint32_t ui32Port,
                   uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The GPIO pins must be properly configured in order to function correctly as hibernate wake-low inputs. This function provides the proper configuration for those pin(s).

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

None.

15.2.3.48 GPIOPinWakeStatus

Retrieves the wake pins status.

Prototype:

```
uint32_t
GPIOPinWakeStatus(uint32_t ui32Port)
```

Parameters:

ui32Port is the base address of the GPIO port.

Description:

This function returns the GPIO wake pin status values. The returned bitfield shows low or high pin state via a value of 0 or 1.

Note:

This function is not available on all devices, consult the data sheet to ensure that the device you are using supports GPIO wake pins.

A subset of GPIO pins on Tiva devices, notably those used by the JTAG/SWD interface and any pin capable of acting as an NMI input, are locked against inadvertent reconfiguration. These pins must be unlocked using direct register writes to the relevant GPIO_O_LOCK and GPIO_O_CR registers before this function can be called. Please see the “gpio_jtag” example application for the mechanism required and consult your part datasheet for information on affected pins.

Returns:

Returns the wake pin status.

15.2.3.49 GPIOPinWrite

Writes a value to the specified pin(s).

Prototype:

```
void
GPIOPinWrite(uint32_t ui32Port,
             uint8_t ui8Pins,
             uint8_t ui8Val)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

ui8Val is the value to write to the pin(s).

Description:

Writes the corresponding bit values to the output pin(s) specified by *ui8Pins*. Writing to a pin configured as an input pin has no effect.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Returns:

None.

15.3 Programming Example

The following example shows how to use the GPIO API to initialize the GPIO, enable interrupts, read data from pins, and write data to pins.

Example: Configure pins for use as input, interrupt, and output GPIOs.

```
int32_t i32Val;

//
// Register the port-level interrupt handler.  This handler is the first
// level interrupt handler for all the pin interrupts.
//
GPIOIntRegister(GPIO_PORTA_BASE, PortAIntHandler);

//
// Initialize the GPIO pin configuration.
//
// Set pins 2, 4, and 5 as input, SW controlled.
//
GPIOPinTypeGPIOInput(GPIO_PORTA_BASE,
                     GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_5);

//
// Set pins 0 and 3 as output, SW controlled.
//
GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_3);

//
// Make pins 2 and 4 rising edge triggered interrupts.
//
GPIOIntTypeSet(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_4, GPIO_RISING_EDGE);

//
// Make pin 5 high level triggered interrupts.
//
GPIOIntTypeSet(GPIO_PORTA_BASE, GPIO_PIN_5, GPIO_HIGH_LEVEL);

//
// Read some pins.
//
i32Val = GPIOPinRead(GPIO_PORTA_BASE,
                     (GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_3 |
                      GPIO_PIN_4 | GPIO_PIN_5));

//
// Write some pins.  Even though pins 2, 4, and 5 are specified, those pins
// are unaffected by this write because they are configured as inputs.  At
// the end of this write, pin 0 is low, and pin 3 is high.
//
GPIOPinWrite(GPIO_PORTA_BASE,
             (GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_3 |
              GPIO_PIN_4 | GPIO_PIN_5),
             (GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
              GPIO_PIN_7));

//
// Enable the pin interrupts.
//
GPIOIntEnable(GPIO_PORTA_BASE, GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_5);
```

Example: Configure Port B Pins for use as a UART 1.

```
//
```

```
// Configure GPIO Port B pins 0 and 1 to be used as UART1.
//
GPIOPinTypeUART(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1);

//
// Enable UART1 functionality on GPIO Port B pins 0 and 1.
//
GPIOPinConfigure(GPIO_PB0_U1RX);
GPIOPinConfigure(GPIO_PB1_U1TX);
```