Embedded Systems Essentials with Arm: Getting Started

Module 3

KV4 (3): Controlling Peripherals with Mbed

GPIOs can be controlled by using the Mbed API. It contains several drivers that provide access to general purpose microcontroller hardware.

The key APIs for digital input/output are:

- DigitalIn
- DigitalOut
- BusIn
- and BusOut.

The DigitalIn interface can be used to read the value of a digital input pin, where the logic level is either 0 or 1.

The DigitalOut interface can be used to configure and control a digital output pin by setting the pin to a logic level of 0 or 1.

Any number of Arm Mbed pins can be used as DigitalIn or DigitalOut.

This simple example shows how a digital input pin called "mybutton" and a digital output pin called "Led_out" are set to turn on an LED if a button is pressed.

There is also a DigitalInOut interface, which is a bidirectional digital pin. It can be used to read the value of a digital pin when set as an input, as well as write the value when set as an output.

The DigitalIn class of commands provides several functions, as shown in this table.

The first two functions, DigitalIn, are constructors for creating a DigitalIn object.

“DigitalIn(PinName pin)” creates a DigitalIn object that is connected to the specified pin.

“DigitalIn(PinName pin, PinMode mode)” creates a DigitalIn object that is connected to a specified pin, and specifies the initial mode of the pin.

The difference between the two is the parameter used to initialize the object.

The “read()” function reads the input and is represented as either 0 or 1.

The "mode" function is used to set the input mode.

The function "is_connected" returns the output setting, represented as 0 or 1.

Finally, the "operator int" function is an operator shorthand for the "read" function.

The DigitalOut class of commands also provides a variety of functions, as shown in this table.

The first two functions, “DigitalOut”, are constructors for creating a DigitalOut object.

“DigitalOut(PinName pin)” creates a DigitalOut object that is connected to the specified pin.

“DigitalOut(PinName pin, int value)” creates a DigitalOut object that is connected to a specified pin, and specifies the initial pin value.

The difference between the two is the parameters used to initialize the object.

The "write()" command is used to set the output, specified as either 0 or 1. The parameter is an integer specifying the pin output value, where 0 is logical 0 and 1 is logical 1.

The "read" function is used to return the output setting of the pin, which will be represented as a 0 for logical 0 or 1 for logical 1.

The "is_connected" function is also used to return the output setting.

The remaining functions are operator shorthands for the "write" and "read" functions.

The BusIn API allows you to combine several DigitalIn pins, enabling you to read all of them at the same time. This is useful for checking multiple inputs together as a single interface instead of individual pins.

The BusOut API allows you to combine several DigitalOut pins, enabling you to write to all of them at the same time. This is useful for writing to multiple pins together as a single interface instead of individual pins.

Any number of Arm Mbed pins can be used as a BusIn or BusOut. This example shows how a BusOut object is created, containing 4 led pins. This can then be used to control the LEDs in a variety of ways.

The BusIn class of commands provides several functions, as shown in this table.

The first two functions, “BusIn (PinName p0)” and “BusIn (PinName pins[16])” are constructors for creating a BusIn object. The functions take multiple pins as its arguments and will connect to the specified pins.

The "read" function is used to read the value of the input bus. This will return an integer with each bit corresponding to the value read from the associated DigitalIn pin.

The "mode" function is used to set the input pin mode.

The "mask" function is a binary mask of bus pins connected to actual pins.

The "operator int()" function is shorthand for the "read" function.

Finally, the "DigitalIn& operator" function provides access to a particular bit in random-iterator fashion.

The BusOut class also has several functions, as shown in this table.

The first two functions, “BusOut (PinName p0)” and “BusOut (PinName pins[16])”, are constructors for creating BusOut objects. The parameters used are DigitalOut pins to connect to a bus bit.

The “write” function is used to write the value to the output bus. The parameter "int value" is an integer that specifies a bit to write for every corresponding DigitalOut pin.

The "read" function is used to read the value being outputted on the bus. This returns an integer with each bit corresponding to the associated DigitalOut pin setting.

The "mask" function is a binary mask of bus pins connected to actual pins.

The two “operator” functions provide shorthand alternatives for the "write" and "read" functions.

Finally, “DigitalIn& operator” provides access to a particular bit in random-iterator fashion.