General Purpose Input/Output (GPIO)

Sasang Balachandran
11/08/2009
ECE 480 Design team 3

Keywords

GPIO, MCU, CPU, ADC, 8962

Executive Summary

Applications that use microcontrollers are rapidly growing as cots of production lower and performance of embedded systems increase, the need to provide flexibility in terms of data input and output is a necessity to create adaptability in microprocessor communication. The use of General Purpose Input/Outputs is a realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements. The variable methods of data handling implemented in these pins, such as ADC conversion and interrupt handling, provide alternative uses that are ideal for multi-input applications.

Introduction

A General Purpose Input/output (GPIO) is an interface available on most modern microcontrollers (MCU) to provide an ease of access to the devices internal properties. Generally there are multiple GPIO pins on a single MCU for the use of multiple interaction so simultaneous application. The pins can be programmed as input, where data from some external source is being fed into the system to be manipulated at a desired time and location. Output can also be performed on GPIOs, where formatted date can be transmitted efficiently to outside devices, this provides a simple mechanism to program and retransmit data depending on user desires through a single port interface. The pins are usually arranged into groups of 8 pins where signals can be sent or received to and from other devices. In many applications, the GPIOs can be configured as interrupt lines for a CPU to signal immediate processing of input lines. In many newer designs, they also have the ability to control and use Direct Memory Access (DMA) to transfer blocks of data in a more efficient manner. Essentially all ports can be tailored to fit specific design goals and provide reusability within applications.

Objectives

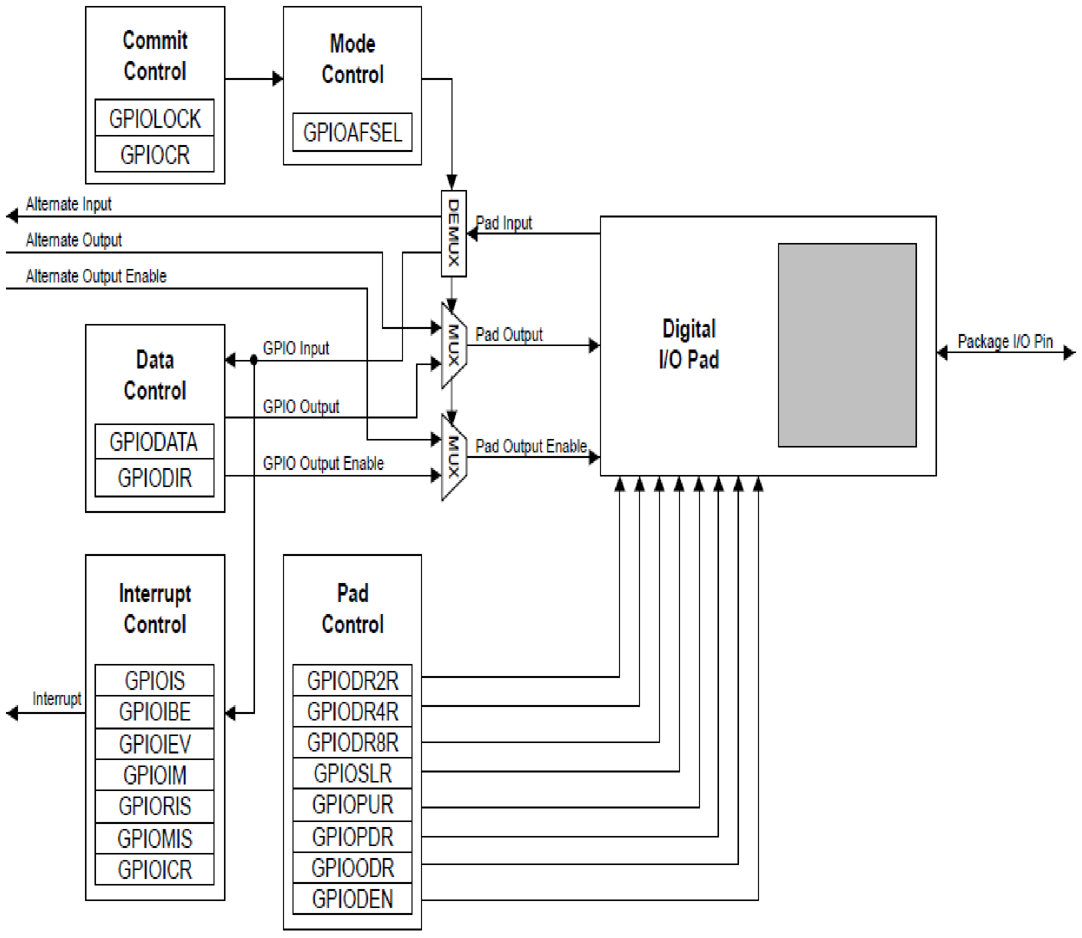
General Purpose I/O (GPIO) pins are single need to be provided to be versatile to digital and analog signals for ADC conversions. To provide efficiency the signals must be signals individually controllable on a particular chip board. Each GPIO should be able to define either an input mode or an output mode for individual pins on the chip. Finally the pins must be extendable for a wide array of applications and functional uses that define its generality in use.

GPIO on the LM3S8962

On the LM3S8962 the GPIO modules consist of seven separate blocks, each of these blocks will corresponding to individual ports on the GPIO interface, the ports in order are: (Port A, Port B, Port C, Port D, Port E, Port F, Port G). The GPIO module on this board supports up to 42 programmable input or output pins depending on the specific configuration being chosen.

Some of the features of the ports include: a standard logic tolerance of up to 5-V on both input and output, specific programmable control for GPIO interrupts which include interrupt generation masking, ADC sampling, programmable control for GPIO pad, digital input enables and open drain enables.

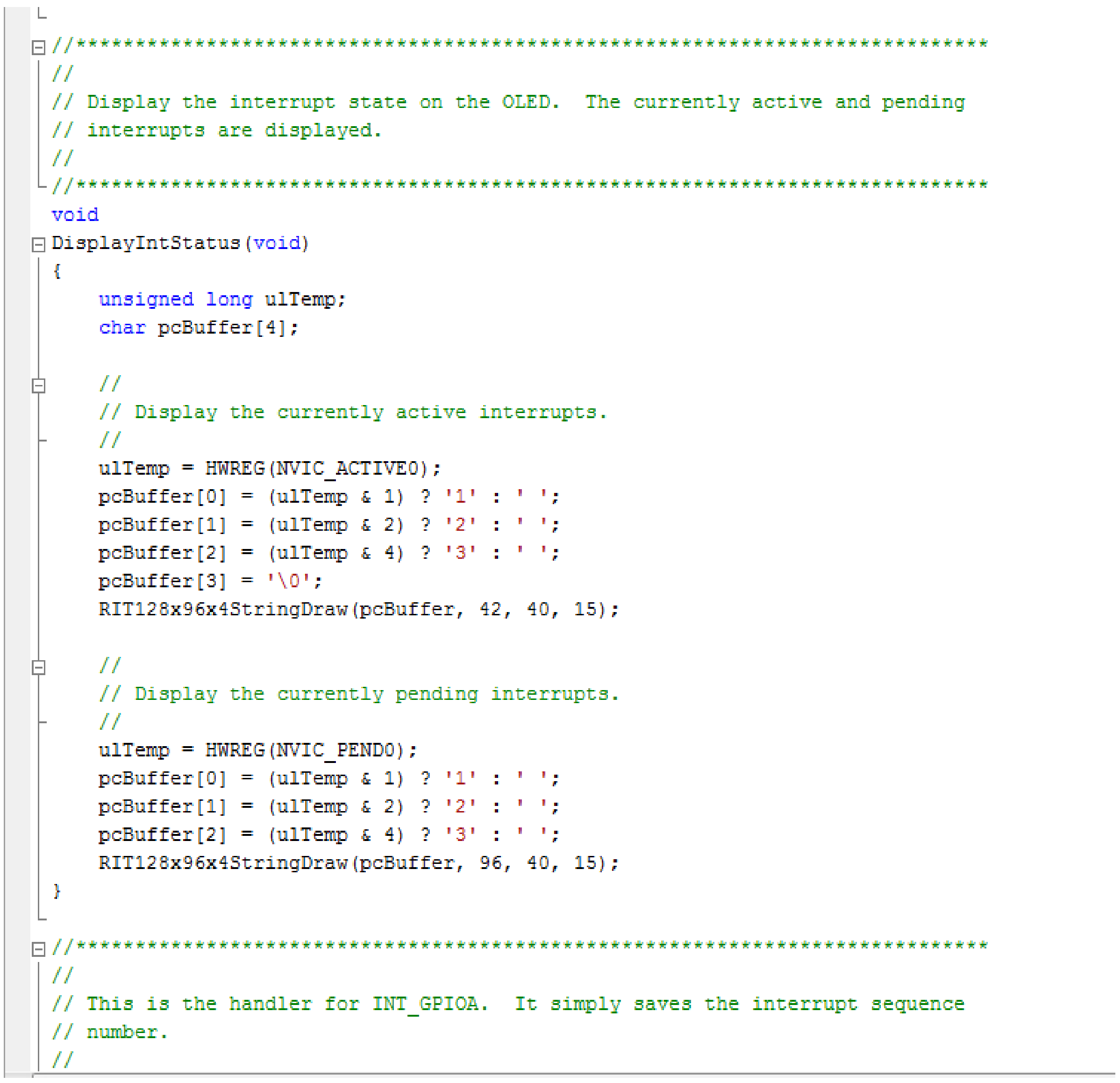
On the 8962 the data control registers allow software to configure the separate programmable modes on the GPIOs. This is done by configuring the data direction registers on the pins as either input or output for the lines. The data registers themselves will contain information to be driven out of the system or new data that’s entered the system. An example of the seven physical blocks of the GPIO is illustrated below:



Interrupt Handling

In general, the interrupt capabilities of each of the ports are maintained by seven of the available registers. These registers define the source of the interrupts, the type of interrupt signal and even the edge property of the signal. In the case where one of more of the input pins triggers an interrupt, the signals are optimized where only a single interrupt output is sent for the entire block.

For the 8962, specific input/output pins can be assigned to be watched for interrupts where the interrupt handling function can trigger a function or certain actions to take place on the microcontroller. An example of an interrupt handling function is reading active and pending interrupts and displaying them to an OLED using output GPIO ports. This is an example of such a code written in C++:



```
//*****************************************************************************
//
// Display the interrupt state on the OLED.  The currently active and pending
// interrupts are displayed.
//
//*****************************************************************************
void
DisplayIntStatus(void)
{
    unsigned long ulTemp;
    char pcBuffer[4];

    //
    // Display the currently active interrupts.
    //
    ulTemp = HWREG(NVIC_ACTIVE0);
    pcBuffer[0] = (ulTemp & 1) ? '1' : ' ';
    pcBuffer[1] = (ulTemp & 2) ? '2' : ' ';
    pcBuffer[2] = (ulTemp & 4) ? '3' : ' ';
    pcBuffer[3] = '\0';
    RIT128x96x4StringDraw(pcBuffer, 42, 40, 15);

    //
    // Display the currently pending interrupts.
    //
    ulTemp = HWREG(NVIC_PEND0);
    pcBuffer[0] = (ulTemp & 1) ? '1' : ' ';
    pcBuffer[1] = (ulTemp & 2) ? '2' : ' ';
    pcBuffer[2] = (ulTemp & 4) ? '3' : ' ';
    RIT128x96x4StringDraw(pcBuffer, 96, 40, 15);
}

//*****************************************************************************
//
// This is the handler for INT_GPIOA.  It simply saves the interrupt sequence
// number.
//
```

Debugging

When writing code to perform desired operation on a microcontroller, the need to debug on a systematic basis in essential to efficient code design. The GPIO pins can be an effective way to monitor and display information about the system in a real time basis.

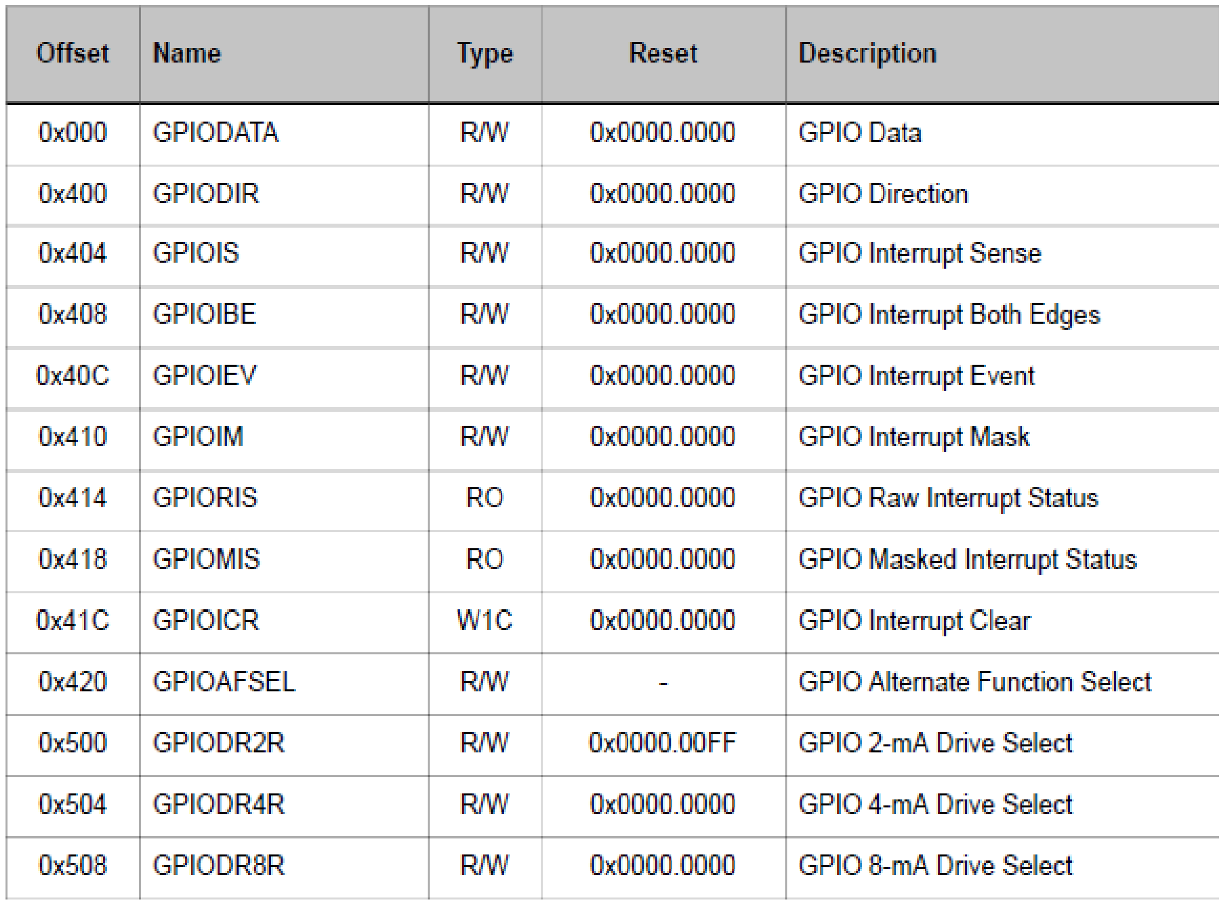
GPIO pins can be used at any time in test execution to display any pertinent information about the program. By providing different messages at different stages to the

GPIO ports, a program or sequence of operations can be easily followed on the hardware level.

Register Controls

To demonstrate the function and use of the registers on a microcontroller, the implementation on the 8962 will be used as an example. The two registers that will be discussed are the data registers and the data direction registers.

The data register on the LM3S8962 is referred to as the GPIODATA register. When writing code to manipulate the values in them, values should be written to GPIODATA. These values will be transferred through the GPIO ports depending on the directionality of the port, which will be discussed next. In order to write to GPIODATA, the corresponding bits in the mask must be set to High. If this is not the case, the bit values will not change by a write sequence. This is inversely true for values read from a register. Bits that are 1 in the address mask cause the corresponding bits in GPIODATA to be read, and bits that are 0 in the address mask cause the corresponding bits in GPIODATA to be read as 0.Finally all bits in the registers are cleared in the event of a reset. A simple mapping of the registers can be seen from the details on the LM3S8962 datasheet:



Offset	Name	Type	Reset	Description
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select

Conclusion

The reason we need pins that provide general purpose use is to provide an interface that can be controlled by various devices and similarly be used to control the behavior of other devices. As an example, a USB or Serial I/ O interface can be programmed to control lines on the GPIO though register setup, these in turn can be programmed to control LEDs or switches on external devices through limited rearrangement of the pins on the board. The ability to program directionality for individual applications and the functionality to handle interrupts and analog signals make application code developed for one purpose easily extendable to other applications through limited editing. Time and efficiency for repetitive modes are the highest priority when using GPIOs and variability of modern processors certainly provide these required responsibilities.

References

[1] Luminary Micro, *EK-LM3S811 Firmware Development Package,* Available at: http://www.luminarymicro.com/products/software_updates.html

[2] Luminary Micro Technical Staff, *LM3S8962 Evaluation Board, User's Manual*, Texas Instruments, 2009. Available at: http://www.luminarymicro.com/index.php?option=com_remository&func=download&id=523&chk=222579d07d3e13fde74ae411749ae30e&Itemid=591