FRONTGRADE

APPLICATION NOTE

UT32M0R500

32-bit Arm™ Cortex® M0+
Microcontroller - Enable the GPIO Module

12/21/2017
Version #: 1.0.0

Product Name	Manufacturer Part Number	SMD #	Device Type	Internal Pic Number
Arm Cortex M0+	UT32M0R500	5962-17212	GPIO Module	QS30

Table 1: Cross Reference of Applicable Products

1.0 Overview

The UT32M0R500 provides three GPIO banks of 16-bit each. GPIO's can be configured as inputs or outputs. As inputs, they can be configured in pull-up or pull down mode. They include a Schmitt- trigger for noise immunity from external inputs. As outputs, they can be configured in push-pull or open drain configuration.

Figure 1 shows the GPIO input pull-up and pull-down mode.

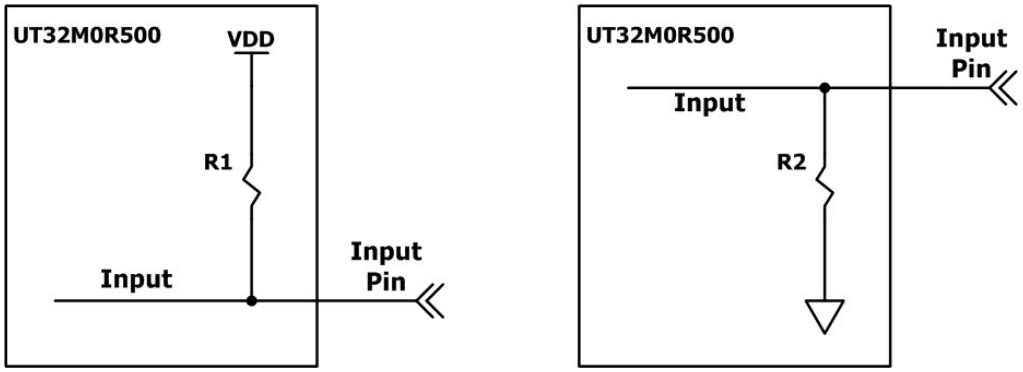

Figure 1: GPIO Input pull-up and pull-down mode equivalent circuits

Figure 2 shows the GPIO output push-pull mode.

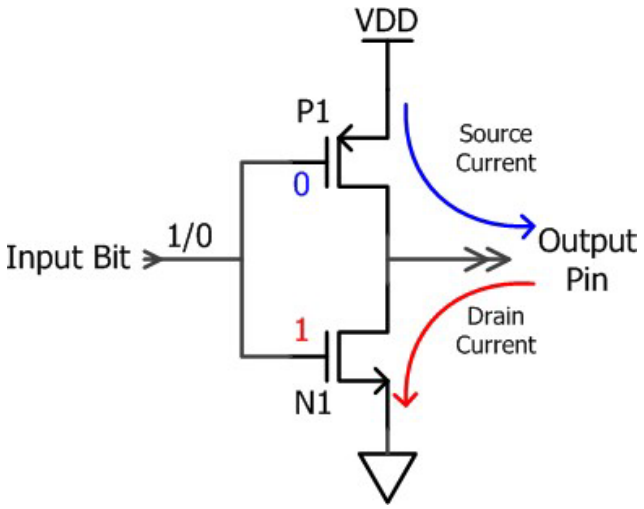

Figure 2: GPIO Output Push-Pull mode

Figure 3 shows the GPIO output open-drain mode

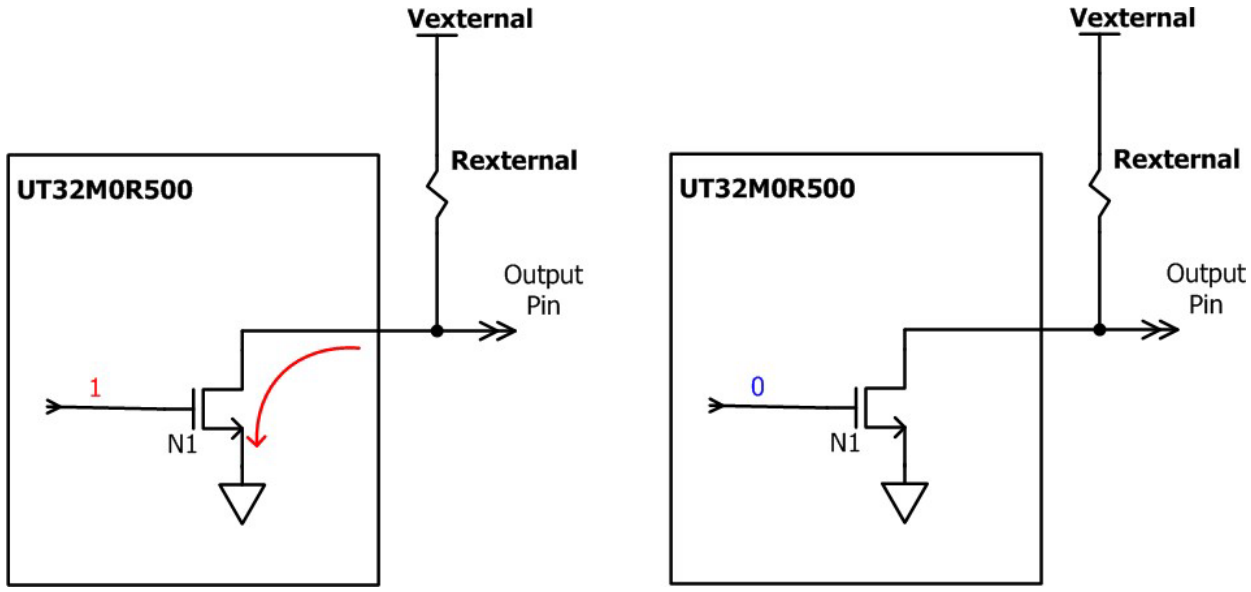

Figure 3: GPIO Open-drain mode

2.0 Application Note Layout

This application note (AN) provides a brief description of the GPIO unit's memory map, configuration and programming.

3.0 GPIO Module Hardware

The GPIO Unit is mapped to the memory region from 0x40020000 – 0x40020FFF, 0x40021000 – 0x40021FFF, 0x40022000 – 0x40022FFF for Bank 0-2 respectively. It has 18 registers plus 8 peripheral ID and 4 component ID registers. For more information on each register, refer to Chapter 8 of the UT32R500 Functional Manual.

3.1 GPIO Alternate Function Set

The Alternate Function Set register **(ALTFUNCSET)** sets the GPIO alternate function bit (ALTFUNCSET), bits [15:0], to either 1 for alternate function or 0 for input/output. The Alternate Function Clear register **(ALTFUNCCLR)** clears the bit.

3.2 GPIO Interrupt Enable

The Interrupt Enable register (INTENSET) sets the GPIO interrupt enable bit (ALTFUNCSET), bits [15:0], to either 1 for interrupt enable or 0 for interrupt disable. The Interrupt Clear register (INTENCLR) clears the bit.

3.3 GPIO Interrupt Type Set

The Interrupt Enable register **(INTTYPESET)** sets the GPIO interrupt type bit (ALTFUNCSET), bits [15:0], to either 1 for interrupt type or 0 for not active. The Interrupt Type Clear register **(INTTYPECLR)** clears the bit.

3.4 GPIO Interrupt Polarity Set

The Interrupt Enable register **(INTTYPESET)** sets the GPIO interrupt polarity bit (ALTFUNCSET), bits [15:0], to either 1 for interrupt type or 0 for not active. The Interrupt Type Clear register **(INTTYPECLR)** clears the bit.

3.5 GPIO Programmable Features

The GPIO has four different features:

- Interrupt generation
- Mask access
- Soft reset
- Alternate function

3.5.1 Interrupt Generation

Interrupt generation provides programmable interrupt features. It consists of three registers for enabling, setting the type and polarity to each particular GPIO interrupt. Table 2 shows the different interrupt register settings.

Interrupt Enable	Interrupt polarity	Interrupt type	Interrupt features
0	-NA-	-NA-	Disable
1	0	0	Low-level
1	0	1	Falling edge
1	1	0	High-level
1	1	1	Rising edge

Table 2: Interrupt Configuration Settings

3.5.2 Mask Access

Mask access permits individual bits or multiple bits to be read from or written to in a single transfer. This avoids software-based read-modify-write operations that are not thread safe. With the masked access operations, the 16bit I/O is divided into two halves, lower byte and upper byte. The bit mask address spaces are defined as two arrays, each containing 256 words. For example, to set bits[1:0] to 1 and clear bits[7:6] in a single operation, you can carry out the write to the lower byte mask access address space. The required bit mask is 0xC3, and you can write the operation as MASKLOWBYTE[0xC3] = 0x03.

3.5.3 Soft Reset

Soft reset controls whether to perform or ignore a reset to the GPIO.

3.5.4 Alternate Function

Alternate function allows sharing the I/O among different interfaces. Alternate functions are controlled by ALTFUNCSET and ALTFUNCLR registers. See PWM example code and PWM app note for more information.

4.0 GPIO Unit Initialization

Code 1 initializes the GPIO output enable set register, and for specifics on the API's, refer to StdPeriphLib at www.frontgrade.com/HiRel.

```
// Init GPIO2
GPIO_StructInit(&GPIO_InitStruct);

// Enable pull-up for GPIO 46
GPIO_InitStruct.PinPUllEnable = PIN_14;
GPIO_InitStruct.PinPullDirection = PIN_14;
GPIO_Init(GPIO2, &GPIO_InitStruct);

// OUTENABLESET = 0x2000; enable GPIO 45 as output and GPIO 46 as input
GPIO_SetPinDirectionsRaw(GPIO2_PIN_GPIO45_OUTPUT);
```

Code 1: GPIO Initialization

5.0 GPIO Unit Programming

Section 3.0 presented some of the basic configurations for the GPIO core. The following sections show programming examples by making use of Frontgrade API's for the UT32RM0R500.

5.1 GPIO Set Output High

The API provides a function for setting the particular out high. The function in code 2 references the GPIO structure and for output high, sets the particular GPIO bit high.

```
// Set GPIO 45 high
GPIO_WriteOutputDataBit(GPIO2, GPIO2_PIN_GPIO45_OUTPUT, SET);
```

Code 2: Set GPIO output high

5.2 GPIO Output Low

The API provides a function for setting the particular out high. The function in code 2 references the GPIO structure and for output low, sets the particular GPIO bit low.

```
// Set GPIO 45 low
GPIO_WriteOutputDataBit(GPIO2, GPIO2_PIN_GPIO45_OUTPUT, RESET);
```

Code 3: Set GPIO output low

Figure 4 shows the Oscilloscope timing diagram for outputting high and low on GPIO 45.

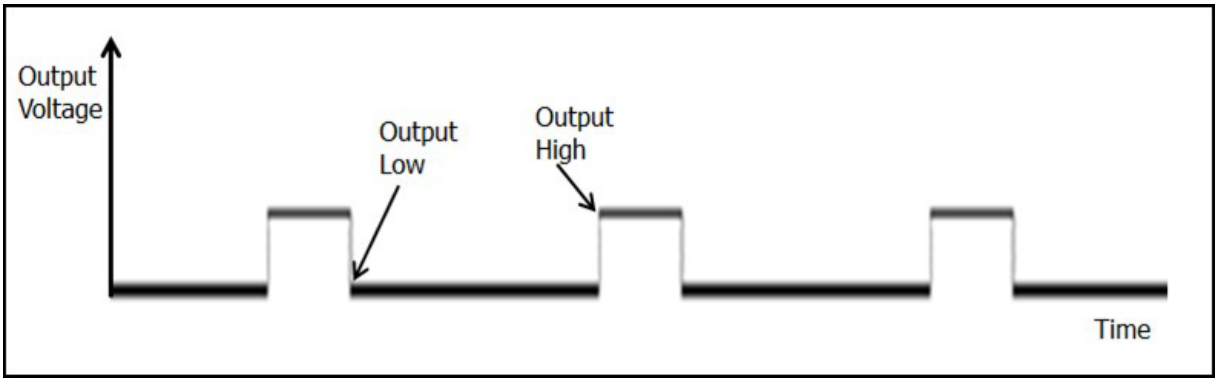

Figure 4: GPIO 45 high and low output

5.3 GPIO Interrupt

All GPIO interrupts are shared to one interrupt (IRQ), which is mapped to number 24 in the Interrupt Vector Table. The address of interrupt 24 in the Interrupt Vector Table is mapped to the GPIO2_ALL_IRQHandler, which is the interrupt service routine (ISR) for all GPIO interrupts. In the ISR, software must check for which interrupt happened.

```
// Enable GPIO 46 bit 14 as input interrupt
GPIO_SetPinInterrupt(SPI, PIN_14, ENABLE, SET, FALLING_EDGE);

// Enables a device specific interrupt in the NVIC interrupt controller
NVIC_EnableIRQ(GPIO2_ALL_IRQn);
```

Code 4: GPIO Interrupt

Figure 5 shows GPIO 46 connected to a switch for input interrupt.

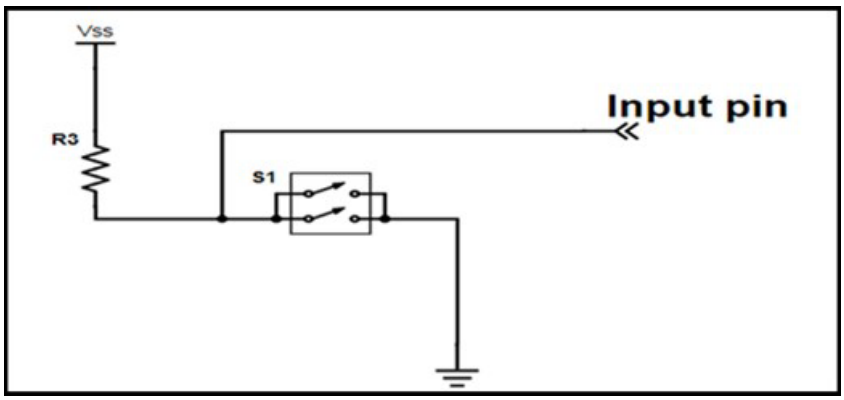

Figure 5: GPIO 46 Input connected to a push button

Figure 6 shows the input GPIO 46 timing diagram for servicing the interrupt service routine after pushing the button.

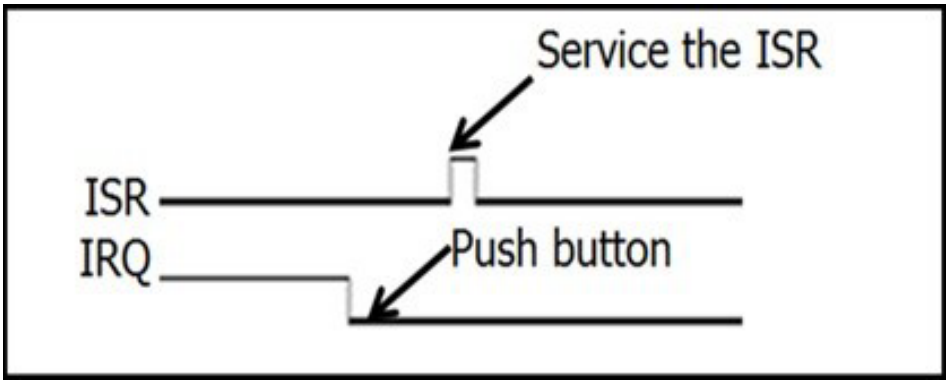

Figure 6: Push button and ISR

Putting it all together, code 4 shows the main subroutine in an endless loop. Code 5 shows The GPIO2_ALL_IRQHandler, which is the interrupt service routine for handling the particular GPIO interrupt. It uses linked list to check which GPIO 2 pin the interrupt happened.

```
int main (void){
 // Initialization and setting from previous sections go here.
 for(;;){
      // do something useful here.
      _ASM volatile("nop");
 }
}
```

Code 5: Sample GPIO program

```
void GPIO2_ALL_IRQHandler(void)
{
 NodePtr Ptr;
 Ptr = &GPIOTasks[0]; // Points to first task in linked list
 while (Ptr){ // Handles all GPIO2 requests    if(GPIO2->INSTATUS & (Ptr->Mask)){
   (*Ptr->GPIOHandler)(); // Execute GPIO Handler
  }
 Ptr= Ptr->Next; // Poll next device
 }
}
```

Code 6: Sample Program for GPIO Interrupt.

6.0 Summary and Conclusion

GPIO can be set as input or output. Inputs can be in pull-up or pull-down mode. Outputs can be in push-pull or open-drain mode. Aside from input/output mode, the GPIO has four different features: interrupt, mask access, soft reset and alternate function.

For more information about our UT32M0R500 microcontroller and other products, please visit our website: www.frontgrade.com/HiRel.

Revision History

Date	Revision #	Author	Change Description	Page #
Dec 2017	1.0.0	JA	Initial Release	