Subsystem Design

Emulating a Digital MUX

1 Description

The *Emulating a Digital MUX software* example demonstrates how to use GPIO interrupts to emulate a digital MUX. Similar to a logic based MUX, the MCU uses select signals (S0 and S1) to determine which input channel (C0, C1, C2, and C3) is output at a given time. Doing this through the MCU not only eliminates the need for an external MUX, but also allows flexible pin assignments that can help aid PCB routing. This specific example emulates a 4-input channel, 2-select-signal digital MUX.

Figure 1-1 displays the functional block diagram for this subsystem.

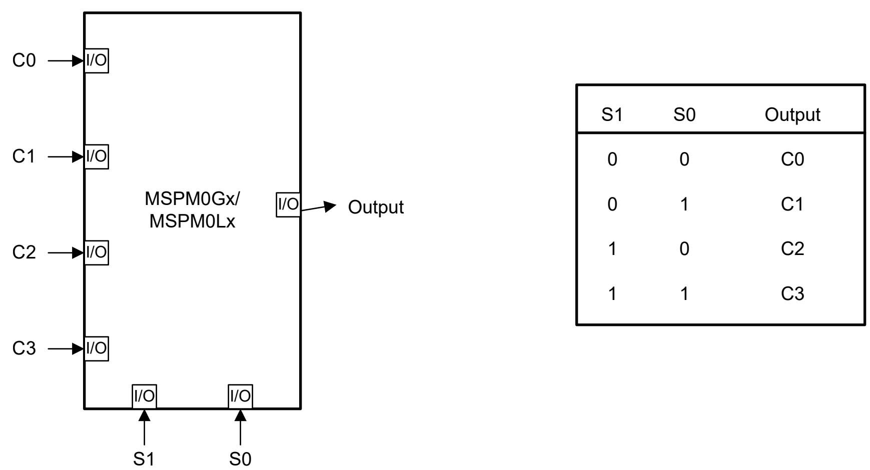

Figure 1-1. Subsystem Functional Block Diagram

2 Required Peripherals

This application requires seven GPIO pins and GPIO interrupts.

Table 2-1. Required Peripherals

Subblock Functionality	Notes
GPIO	Pin groups are referred to as INPUT, OUTPUT, and SELECT in code

3 Compatible Devices

Based on the requirements in Table 2-1, the compatible devices are listed in Table 3-1. The corresponding EVM can be used for quick evaluation.

Table 3-1. Compatible Devices

Compatible Devices	EVM
MSPM0C	LP-MSPM0C1104
MSPM0Lx	LP-MSPM0L1306
MSPM0Gx	LP-MSPM0G3507

4 Design Steps

1. Determine the amount of GPIOs needed by the application. In this case, there are 4 input channel pins, two select pins, and one output pin.
2. Configure the GPIO output pin in SysConfig as an output.
3. Configure the GPIO input channel pins and select pins in SysConfig as inputs with interrupts.
4. Write application code for the interrupts to change the output based on the Channel and SELECT digital signals.

5 Design Considerations

1. Number of input channel and select pins: A 4-input MUX requires two select pins. However, an 8-input MUX requires three select pins.
2. The logic table: What select pin configuration determines which input channel is selected as the output.
3. Interrupts: Interrupts must be placed on all input channel and select pins as the output signal is generated by setting or clearing the output signal based on the selected input channel.
4. Propagation delay: There is a possibility for a propagation delay due to interrupts. The propagation delay is based on the clock speed.

6 Software Flow Chart

Figure 6-1 shows the software flow chart for this subsystem example and explains the GPIO interrupt routine used to emulate a digital MUX.

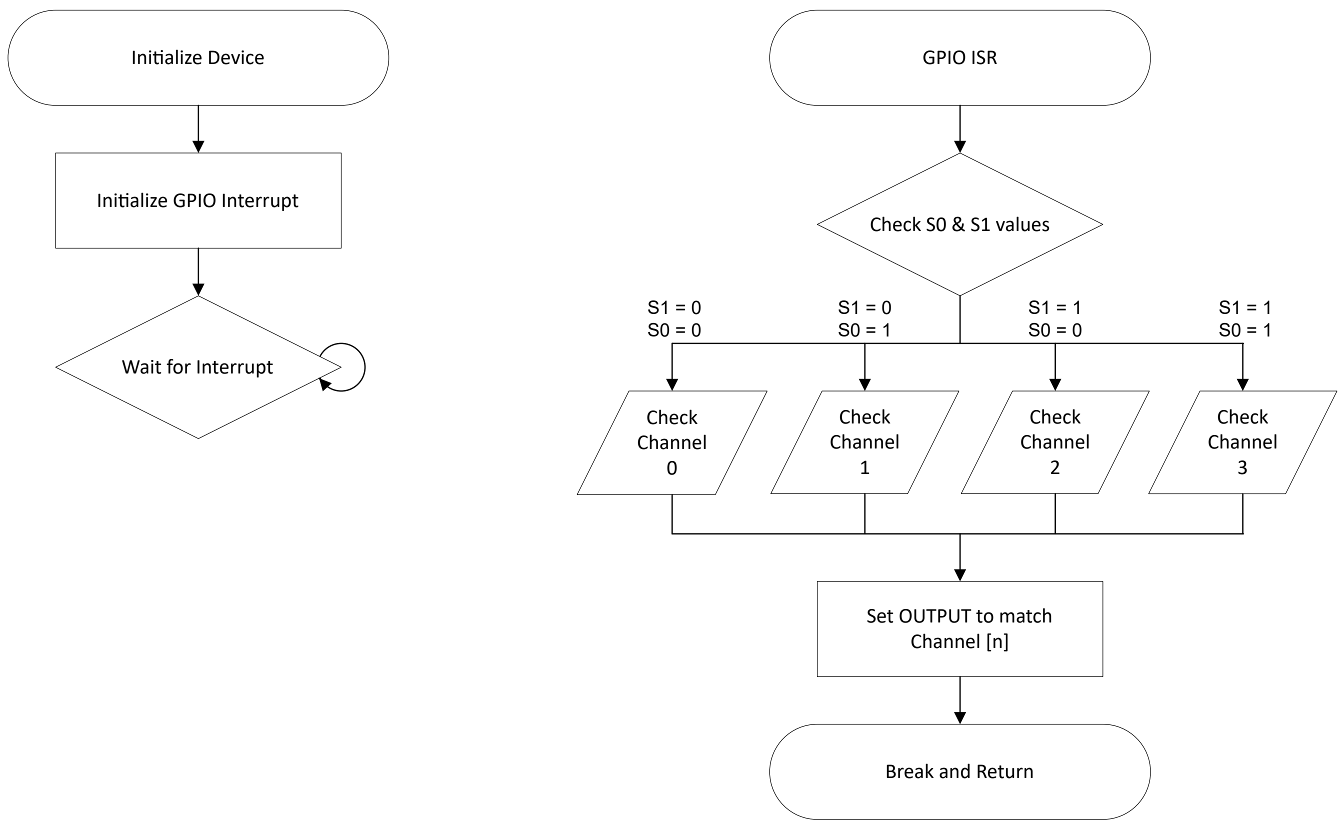

Figure 6-1. Application Software Flow Chart

7 Application Code

This application uses the *TI System Configuration tool (SysConfig)* graphical interface to generate the configuration code for the device peripherals. Using a graphical interface to configure the device peripherals streamlines the application prototyping process.

In addition, this application uses GPIO interrupts on all input pins configured and enabled in the GPIO peripheral in SysConfig. Based on the GPIO pins configured in SysConfig, the respective GPIO interrupts must also be manually enabled in the `main()` portion of the code using the `NVIC_EnableIRQ();` function. After enabling the interrupts, the `main()` code waits for an interrupt. This means that any time one of the input signals changes state, the GPIO interrupt service routine starts. The `main()` portion of this code is as follows:

```
int main(void)
{
    SYSCFG_DL_init();
    /* Enable GPIO Port A Interrupts */
    NVIC_EnableIRQ(GPIO_MULTIPLE_GPIOA_INT_IRQN);

    while (1) {
        __WFI();
    }
}
```

The following code snippet showcases the GPIO interrupt service routine. There are two switch cases: one for the interrupt types, and one to determine which input channel is selected to be output. The second switch case first checks the select pins to determine the respective states. Depending on those states, the input channel is selected based on the logic truth table (see Figure 1-1). For each individual case, the selected input channel pin is checked, and the output pin is set to match. The code then breaks out of the interrupt service routine, and then returns to wait for another interrupt. In addition, this example code uses pin PA0 on the LP-MSPM0L1306 as the output pin, which turns a red LED on and off based on the output signal.

```
void GROUP1_IRQHandler(void){
    switch (DL_Interrupt_getPendingGroup(DL_INTERRUPT_GROUP_1)){
        case GPIO_MULTIPLE_GPIOA_INT_IIDX:
            switch (DL_GPIO_readPins(SELECT_PORT, SELECT_S1_PIN | SELECT_S0_PIN)){
                case 0: /* S1 = 0, S0 = 0 */
                /* Check Channel 0 and set output to match */
                    if (DL_GPIO_readPins(INPUT_PORT, INPUT_CHANNEL_0_PIN)){
                        DL_GPIO_setPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    } else {
                        DL_GPIO_clearPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    }
                    break;
                case SELECT_S0_PIN: /* S1 = 0, S0 = 1 */
                /* Check Channel 1 and set output to match */
                    if (DL_GPIO_readPins(INPUT_PORT, INPUT_CHANNEL_1_PIN)){
                        DL_GPIO_setPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    } else {
                        DL_GPIO_clearPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    }
                    break;
                case SELECT_S1_PIN: /* S1 = 1, S0 = 0 */
                /* Check Channel 2 and set output to match */
                    if (DL_GPIO_readPins(INPUT_PORT, INPUT_CHANNEL_2_PIN)){
                        DL_GPIO_setPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    } else {
                        DL_GPIO_clearPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    }
                    break;
                case SELECT_S1_PIN | SELECT_S0_PIN: /* S1 = 1, S0 = 1 */
                /* Check Channel 3 and set output to match */
                    if (DL_GPIO_readPins(INPUT_PORT, INPUT_CHANNEL_3_PIN)){
                        DL_GPIO_setPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    } else {
                        DL_GPIO_clearPins(OUTPUT_PORT, OUTPUT_LED_PIN);
                    }
                    break;
            }
            break;
```

```
    }
}
```

8 Results

Figure 8-1 shows a logic capture of the different input-to-output signals. The input channels C0 through C3 are colored white, brown, red, and orange, respectively. S0 is yellow and S1 is green. Finally, the output signal is blue. The capture is marked to showcase how the different inputs change the output signal.

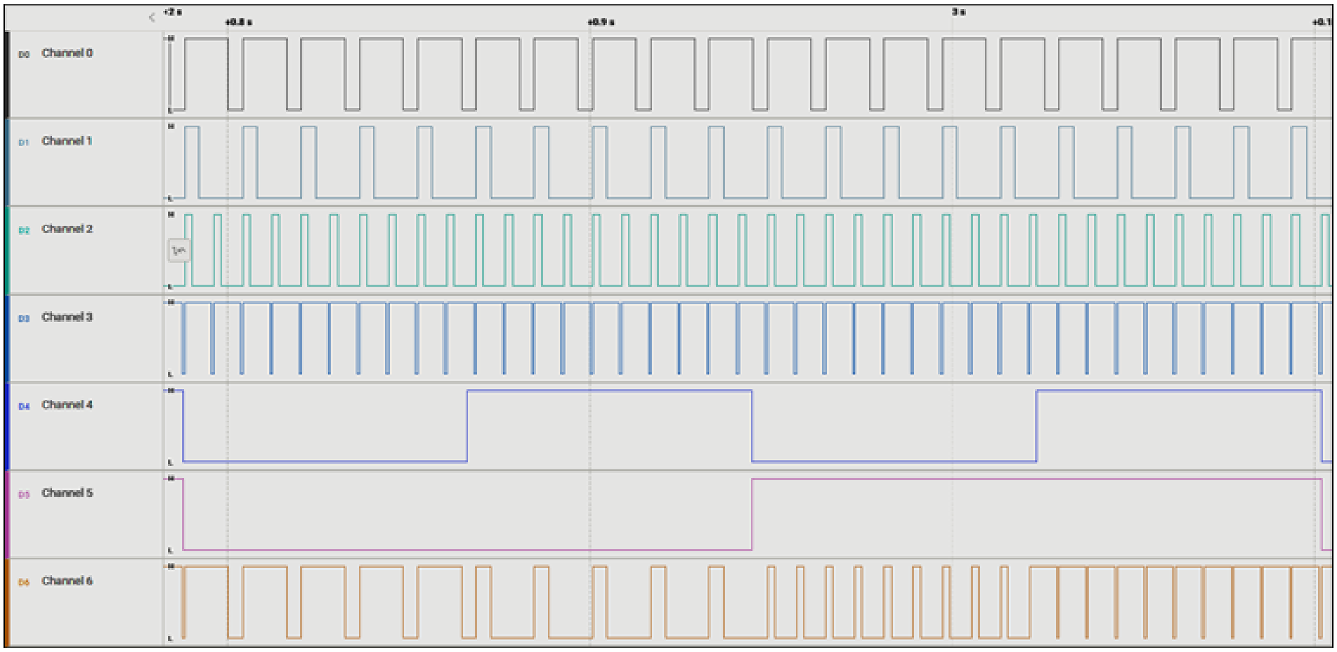

Figure 8-1. Results

9 Additional Resources

- Texas Instruments, Download the MSPM0 SDK
- Texas Instruments, Learn more about SysConfig
- Texas Instruments, MSPM0L LaunchPad™
- Texas Instruments, MSPM0G LaunchPad™
- Texas Instruments, MSPM0C LaunchPad™
- Texas Instruments, MSPM0 Academy

10 E2E

See TI's E2E™ support forums to view discussions and post new threads to get technical support for utilizing MSPM0 devices in designs.

11 Trademarks

LaunchPad™ and E2E™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265