PV198 - GPIO

One-chip Controllers

Daniel Dlhopolček, Marek Vrbka, Jan Koniarik, Oldřich Pecák, Tomáš Rohlínek, Ján Labuda, Jan Horáček, Matúš Škvarla, Ondřej Bleha, Martin Klimeš, Adam Valt

Faculty of Informatics, Masaryk University

2/2024

Content

GPIO Overview

Buttons

Debouncing

Application

LED using SDK example

Button using config tools

Homework

- Have you checked the preliminaries in study materials?
- Do not forget to setup a new branch for this week (*Week_02*)!

What is GPIO

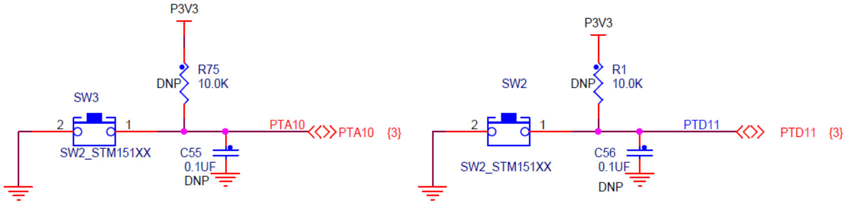
- **GPIO** – **G**eneral **P**urpose **I**nput **O**utput
- Direct control of pins of the MCU
- Basic interaction with external world
- Can be programmed as *input* or *output*
- Has only 2 states (logic 0, logic 1)

What is it used for

- Anything that works with 2 states – on/off
- LED
- Buttons
- Sensors
- And used by more sophisticated peripherals

How buttons on board work

Connects pit to ground (logic 0) or to voltage (logic 1)



Button debouncing

- Bouncing
  - Looks like button is pressed multiple times
  - Cause by mechanical contact of the switch
- Solution
  1. HW debounce (add capacitor)
  2. SW debounce (wait few milliseconds)

Steps required to create an application

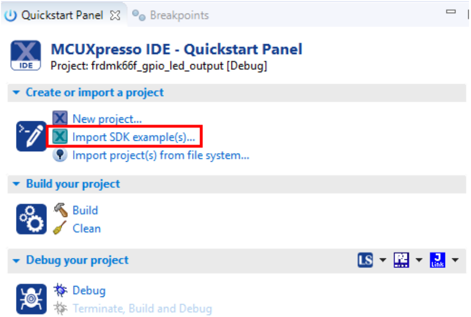
- Initialize (*MCUXpresso Configuration Tools* help here)
  1. Pin
  2. Clocks
  3. Peripherals
- Write application code

Peripheral configuration options

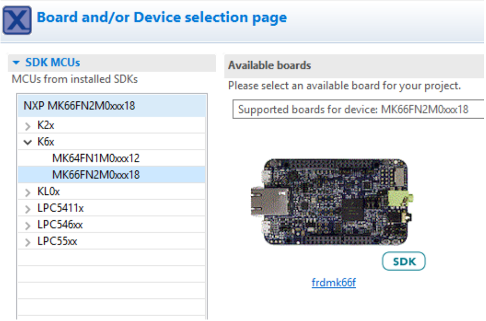
1. Write everything from scratch
   - Error prone, time demanding, tedious
2. Use SDK example
   - Works out of box
   - Difficult to modify
3. Use config tools
   - Easy to use and modify

LED using the SDK example

- Select Import SDK example(s)...


Quickstart Panel
Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: frdmk66f_gpio_led_output [Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

1. Open the K6x
2. Select the MK66FN2M0xxx18
3. Click the board image



Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs

NXP MK66FN2M0xxx18

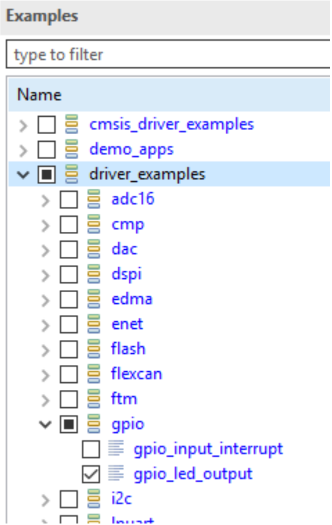
- K2x
- K6x
  - MK64FN1M0xxx12
  - MK66FN2M0xxx18
- KL0x
- LPC5411x
- LPC546xx
- LPC55xx

Available boards

Please select an available board for your project.

Supported boards for device: MK66FN2M0xxx18

1. Open `driver_examples` → `gpio`
2. Select the `gpio_led_output` example
3. Click *Finish*



Examples

type to filter

Name

- › □ cmsis_driver_examples
- › □ demo_apps
- ⌄ ■ driver_examples
  - › □ adc16
  - › □ cmp
  - › □ dac
  - › □ dspi
  - › □ edma
  - › □ enet
  - › □ flash
  - › □ flexcan
  - › □ ftm
  - ⌄ ■ gpio
    - □ gpio_input_interrupt
    - ☑ gpio_led_output
  - › □ i2c
  - › □ lpuart

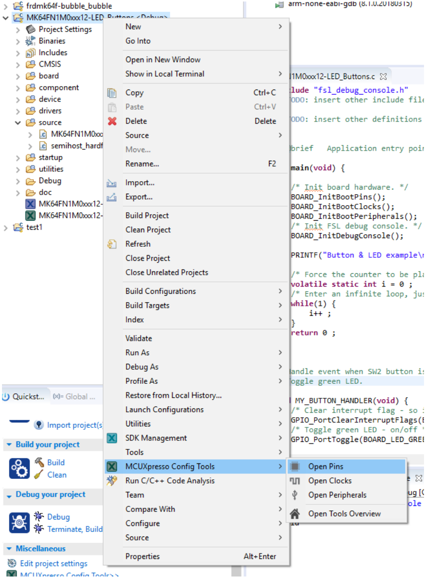
Opened example project

- Pins and clocks are already configured
- `GPIO_PinInit`
- `GPIO_PortToggle`

Button control program

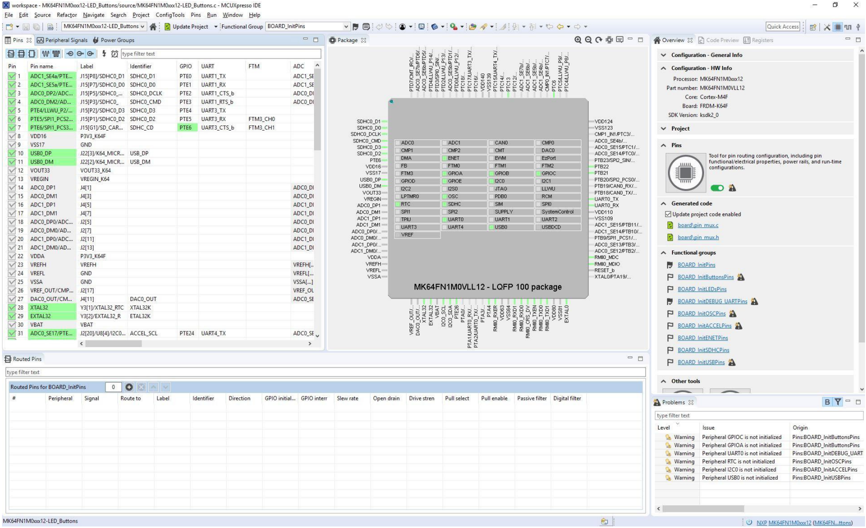
- We will show you how to check for button presses
- The end goal is to write program which will print text to console when SW2 is pressed

Application Button using config tools



```
lude "fsl_debug_console.h"
ODO: insert other include file

ODO: insert other definitions

brief   Application entry poi

main(void) {

/* Init board hardware. */
BOARD_InitBootPins();
BOARD_InitBootClocks();
BOARD_InitBootPeripherals();
/* Init FSL debug console. */
BOARD_InitDebugConsole();

PRINTF("Button & LED example\r

/* Force the counter to be pla
volatile static int i = 0 ;
/* Enter an infinite loop, jus
while(1) {
    i++ ;
}
return 0 ;


andle event when SW2 button i
oggle green LED.

 MY_BUTTON_HANDLER(void) {
/* Clear interrupt flag - so
GPIO_PortClearInterruptFlags(
/* Toggle green LED - on/off
GPIO_PortToggle(BOARD_LED_GRE
```

You should see the Pin tool now



workspace - MK64FN1M0xxx12-LED_Buttons/source/MK64FN1M0xxx12-LED_Buttons.c - MCUXpresso IDE

File Edit Source Refactor Navigate Search Project ConfigTools Pins Run Window Help
MK64FN1M0xxx12-LED_Buttons
Update Project
Functional Group BOARD_InitPins
Quick Access
Pins
Peripheral Signals
Power Groups
type filter text
Package
MK64FN1M0VLL12 - LQFP 100 package
Overview
Code Preview
Registers

Pin	Pin name	Label	Identifier	GPIO	UART	FTM	ADC
☑ 1	ADC1_SE4a/PTE...	J15[P8]/SDHC0_D1	SDHC0_D1	PTE0	UART1_TX		ADC1_SE
☑ 2	ADC1_SE5a/PTE...	J15[P7]/SDHC0_D0	SDHC0_D0	PTE1	UART1_RX		ADC1_SE
☑ 3	ADC0_DP2/ADC...	J15[P5]/SDHC0_...	SDHC0_DCLK	PTE2	UART1_CTS_b		ADC0_DI
☑ 4	ADC0_DM2/AD...	J15[P3]/SDHC0_	SDHC0_CMD	PTE3	UART1_RTS_b		ADC0_DI
☑ 5	PTE4/LLWU_P2/...	J15[P2]/SDHC0_D3	SDHC0_D3	PTE4	UART3_TX		
☑ 6	PTE5/SPI1_PCS2...	J15[P1]/SDHC0_D2	SDHC0_D2	PTE5	UART3_RX	FTM3_CH0	
☑ 7	PTE6/SPI1_PCS3...	J15[G1]/SD_CAR...	SDHC_CD	PTE6	UART3_CTS_b	FTM3_CH1	
☑ 8	VDD16	P3V3_K64F					
☑ 9	VSS17	GND					
☑ 10	USB0_DP	J22[3]/K64_MICR...	USB_DP				
☑ 11	USB0_DM	J22[2]/K64_MICR...	USB_DM				
☑ 12	VOUT33	VOUT33_K64					
☑ 13	VREGIN	VREGIN_K64					
☑ 14	ADC0_DP1	J4[1]					ADC0_DI
☑ 15	ADC0_DM1	J4[3]					ADC0_DI
☑ 16	ADC1_DP1	J4[5]					ADC1_DI
☑ 17	ADC1_DM1	J4[7]					ADC1_DI
☑ 18	ADC0_DP0/ADC...	J2[5]					ADC0_DI
☑ 19	ADC0_DM0/AD...	J2[7]					ADC0_DI
☑ 20	ADC1_DP0/ADC...	J2[11]					ADC1_DI
☑ 21	ADC1_DM0/AD...	J2[13]					ADC1_DI
☑ 22	VDDA	P3V3_K64F					
☑ 23	VREFH	VREFH					VREFH[...
☑ 24	VREFL	GND					VREFL[...
☑ 25	VSSA	GND					VSSA[...]
☑ 26	VREF_OUT/CMP...	J2[17]					VREF_OU
☑ 27	DAC0_OUT/CM...	J4[11]	DAC0_OUT				ADC0_SE
☑ 28	XTAL32	Y3[1]/XTAL32_RTC	XTAL32K				
☑ 29	EXTAL32	Y3[2]/EXTAL32_R	ETAL32K				
☑ 30	VBAT	VBAT					
☑ 31	ADC0_SE17/PTE...	J2[20]/U8[4]/I2C0...	ACCEL_SCL	PTE24	UART4_TX		ADC0_SE

Routed Pins

type filter text

Routed Pins for BOARD_InitPins 0

#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial...	GPIO interr	Slew rate	Open drain	Drive stren	Pull select	Pull enable	Passive filter	Digital filter

Configuration - General Info

Configuration - HW Info

Processor: MK64FN1M0xxx12
Part number: MK64FN1M0VLL12
Core: Cortex-M4F
Board: FRDM-K64F
SDK Version: ksdk2_0

Project

Pins

Tool for pin routing configuration, including pin functional/electrical properties, power rails, and run-time configurations.

Generated code

☑ Update project code enabled

board\pin_mux.c

board\pin_mux.h

Functional groups

- BOARD_InitPins
- BOARD_InitButtonsPins
- BOARD_InitLEDsPins
- BOARD_InitDEBUG_UARTPins
- BOARD_InitOSCPins
- BOARD_InitACCELPins
- BOARD_InitENETPins
- BOARD_InitSDHCPins
- BOARD_InitUSBPins

Other tools

Problems

type filter text

Level	Issue	Origin
Warning	Peripheral GPIOC is not initialized	Pins:BOARD_InitButtonsPins
Warning	Peripheral GPIOA is not initialized	Pins:BOARD_InitButtonsPins
Warning	Peripheral UART0 is not initialized	Pins:BOARD_InitDEBUG_UART
Warning	Peripheral RTC is not initialized	Pins:BOARD_InitOSCPins
Warning	Peripheral I2C0 is not initialized	Pins:BOARD_InitACCELPins
Warning	Peripheral USB0 is not initialized	Pins:BOARD_InitUSBPins

MK64FN1M0xxx12-LED_Buttons

NXP MK64FN1M0xxx12 (MK64FN...ttons)

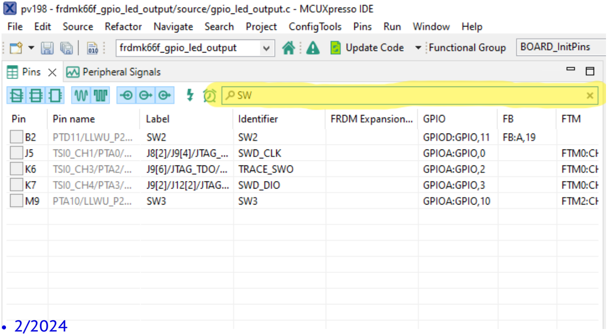
Initialization

How configuration tools can help us:

- Modify settings easily
- Visual representation of configuration
- Great for custom boards (generates defines for custom boards that simplify management)

Configuration

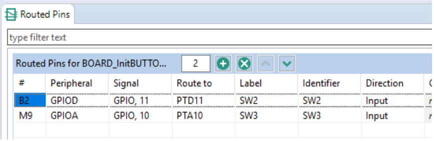
- Pins tool contains predefined configurations
- We should already see the red LED configured
- Add the configuration for SW2 and SW3 buttons
  - Search for SW2 and SW3 on the “pins” window
  - Click on the checkboxes for SW2 and SW3 and add the GPIO option
  - This will call initialization code for the button pins on program startup


pv198 - frdmk66f_gpio_led_output/source/gpio_led_output.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Pins Run Window Help
frdmk66f_gpio_led_output
Update Code
Functional Group
BOARD_InitPins
Pins
Peripheral Signals

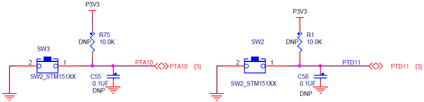
SW

Pin	Pin name	Label	Identifier	FRDM Expansion...	GPIO	FB	FTM
□ B2	PTD11/LLWU_P2...	SW2	SW2		GPIOD:GPIO,11	FB:A,19	
□ J5	TSI0_CH1/PTA0/...	J8[2]/J9[4]/JTAG_...	SWD_CLK		GPIOA:GPIO,0		FTM0:CH
□ K6	TSI0_CH3/PTA2/...	J9[6]/JTAG_TDO/...	TRACE_SWO		GPIOA:GPIO,2		FTM0:CH
□ K7	TSI0_CH4/PTA3/...	J9[2]/J12[2]/JTAG...	SWD_DIO		GPIOA:GPIO,3		FTM0:CH
□ M9	PTA10/LLWU_P2...	SW3	SW3		GPIOA:GPIO,10		FTM2:CH

- *Code preview* was updated
- If you check the *Code Preview* tab, you should see that the `pin_mux.c` file now has extra SW2 and SW3 configuration
- You should see in the *Routed Pins* tab (lower-left corner) that button pins are routed to PTD11 (SW2) and PTA10 (SW3)



#	Peripheral	Signal	Route to	Label	Identifier	Direction
B2	GPIOD	GPIO, 11	PTD11	SW2	SW2	Input
M9	GPIOA	GPIO, 10	PTA10	SW3	SW3	Input



Updating code

- Click the *Update Code* button
  - It opens the *Update Files* dialog
  - You can check which changes will be made
  - For now, just click OK

Writing actual code

Task - Reading a button and printing to console

- Read the current state of the GPIO Button (SW2 and/or SW3)
- If button is pressed, print text to console
- Otherwise, do nothing

Issues

- When you press the button, text is printed several times
  - Why?
  - What are the ways to resolve it?

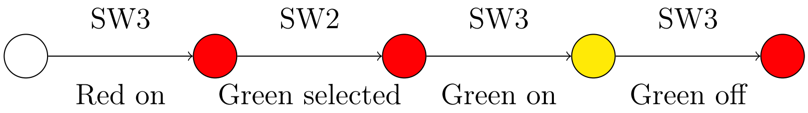
Work Progress

- Write an application that toggles green LED when SW3 is pressed
- Fix the issue with the button press being registered more than once
- Make the LED change color every time it is turned on
  - There are three controllable LEDs on the board

Homework

Write an application which reacts to both buttons

- SW2 selects color
- SW3 toggles the color on and off
- All colors start turned off
- Selected color starts on red
- Colors switch in the following order: Red → Green → Blue → Red...
- Application must be immune to the effects of bouncing



Submission

- Git tag - “Submission_02_x”
- One project per branch!

MUNI
FACULTY
OF INFORMATICS