

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

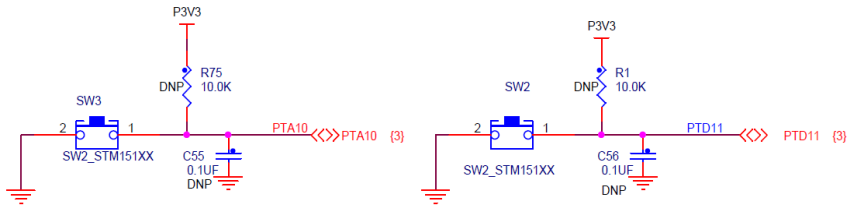
- **GPIO** – **G**eneral **P**urpose **I**nput **O**utput
- Direct control of pins of the MCU
- Basic interaction with external world
- Can be programmed as *input* or *output*
- Has only 2 states (logic 0, logic 1)

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 pin 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

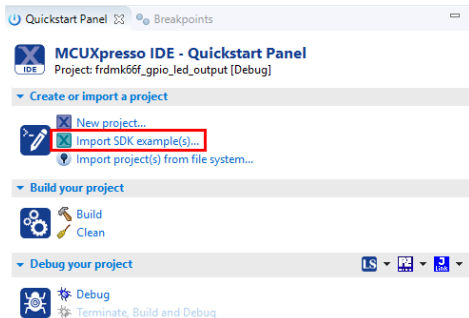
- Initialize (*MCUXpresso Configuration Tools* help here)
 1. Pin
 2. Clocks
 3. Peripherals
- Write application code

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)...



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

Board and/or Device selection page

SDK MCUs

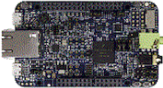
MCUs from installed SDKs

- NXP MK66FN2M0xxx18
 - > K2x
 - ▼ K6x
 - MK64FN1M0xxx12
 - MK66FN2M0xxx18**
 - > KL0x
 - > LPC5411x
 - > LPC546xx
 - > LPC55xx

Available boards

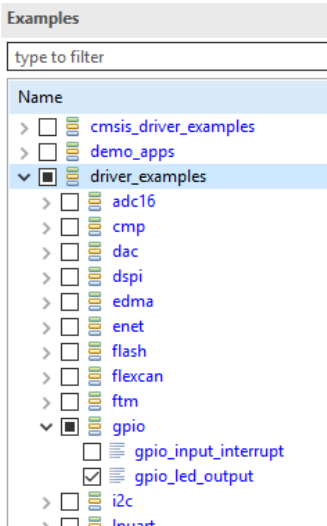
Please select an available board for your project.

Supported boards for device: MK66FN2M0xxx18



[frdmk66f](#) SDK

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



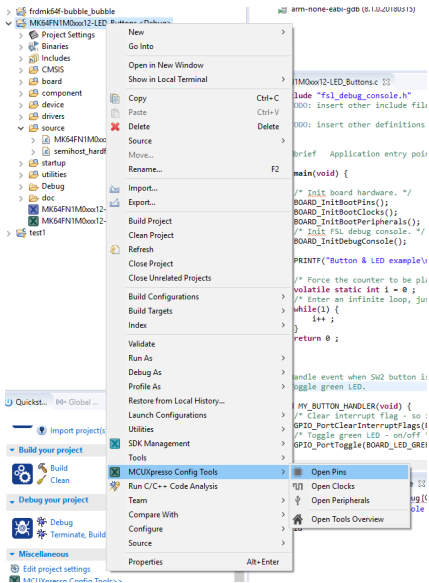
Opened example project

- 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



The screenshot shows an IDE interface with a project tree on the left and a code editor on the right. The project tree is expanded to show the 'source' folder. A context menu is open over the 'source' folder, with 'MCUXpresso Config Tools' selected. A sub-menu is also open, showing options like 'Open Pins', 'Open Clocks', 'Open Peripherals', and 'Open Tools Overview'.

```
arm-none-eabi-gdb (6.11.0.20180513)
MK64FN1M0xx12-LED
  Project Settings
  Binaries
  Includes
  CMSIS
  board
  component
  device
  drivers
  source
    MK64FN1M0xx12-LED
    semihost_hardf
  startup
  utilities
  Debug
  doc
  MK64FN1M0xx12-LED
  MK64FN1M0xx12-LED
  test1

MK64FN1M0xx12-LED
  Project Settings
  Binaries
  Includes
  CMSIS
  board
  component
  device
  drivers
  source
    MK64FN1M0xx12-LED
    semihost_hardf
  startup
  utilities
  Debug
  doc
  MK64FN1M0xx12-LED
  MK64FN1M0xx12-LED
  test1

MK64FN1M0xx12-LED_Buttons.c
#include "fsl_debug_console.h"
#define INSERT_OTHER_INCLUDE_FILES_HERE
#define INSERT_OTHER_DEFINITIONS_HERE
brief Application entry point
main(void) {
/* Init board hardware. */
BOARD_InitBootPins();
BOARD_InitBootClocks();
BOARD_InitBootPeripherals();
/* Init FSL debug console. */
BOARD_InitDebugConsole();

PRINTF("Button & LED example\n");

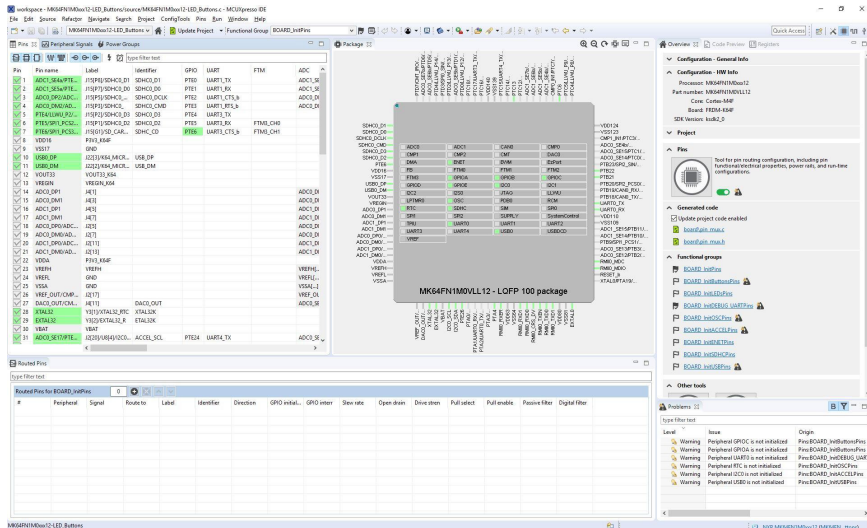
/* Force the counter to be global. */
volatile static int i = 0;
/* Enter an infinite loop, just in case. */
while(1) {
i++;
}
return 0;
}

handle event when SW2 button is pressed.
Toggle green LED.

MY_BUTTON_HANDLER(void) {
/* Clear interrupt flag - so we can get another interrupt. */
GPIO_PortClearInterruptFlags(GPIO_PORTC_DR00R, GPIO_PIN_0);
/* Toggle green LED - on/off */
GPIO_PortToggle(GPIO_LED_GREEN, GPIO_PIN_0);
}

Open Pins
Open Clocks
Open Peripherals
Open Tools Overview
```

You should see the Pin tool now



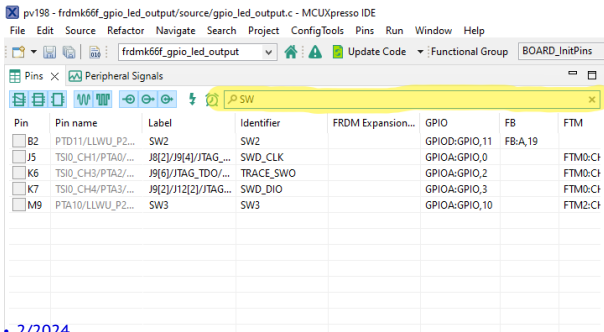
Initialization

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



The screenshot shows the MCUXpresso IDE interface with the Pins tool open. The table below is a representation of the data shown in the tool's 'Peripheral Signals' window.

Pin	Pin name	Label	Identifier	FRDM Expansion...	GPIO	FB	FTM
<input type="checkbox"/>	B2	PTD11/LLWU_P2...	SW2		GPIOD:GPIO,11	FB:A,19	
<input type="checkbox"/>	J5	TSIO_CH1/PTA0/...	J8[2]/J9[4]/JTAG...	SW2	GPIOA:GPIO,0		FTM0:CT
<input type="checkbox"/>	K6	TSIO_CH3/PTA2/...	J9[6]/JTAG_TDO/...	TRACE_SWO	GPIOA:GPIO,2		FTM0:CT
<input type="checkbox"/>	K7	TSIO_CH4/PTA3/...	J9[2]/J12[2]/JTAG...	SWD_DIO	GPIOA:GPIO,3		FTM0:CT
<input type="checkbox"/>	M9	PTA10/LLWU_P2...	SW3		GPIOA:GPIO,10		FTM2:CT

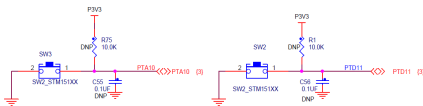
- *Code preview* was updated
- If you check the *Code Preview* tab, you should see that the `pin_mux.c` file now has extra SW2 and SW3 configuration
- You should see in the *Routed Pins* tab (lower-left corner) that button pins are routed to PTD11 (SW2) and PTA10 (SW3)

Routed Pins

type filter text

Routed Pins for BOARD_InitBUTTO... 2

#	Peripheral	Signal	Route to	Label	Identifier	Direction
B2	GPIOB	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?

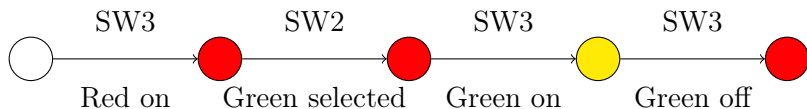
Work Progress

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



Submission

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

MUNI

FACULTY

OF INFORMATICS