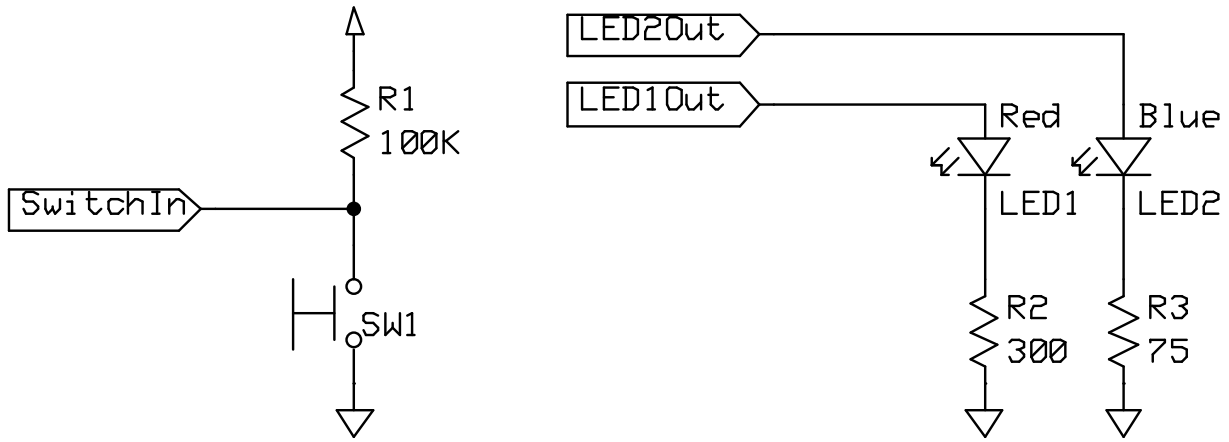
General Purpose I/O

Overview

- How do we make a program light up LEDs in response to a switch?
- GPIO
  - Basic Concepts
  - Port Circuitry
  - Alternate Functions
  - Peripheral Access In C
- Circuit Interfacing
  - Inputs
  - Outputs
- Additional Port Configuration

ARM

Basic Concepts

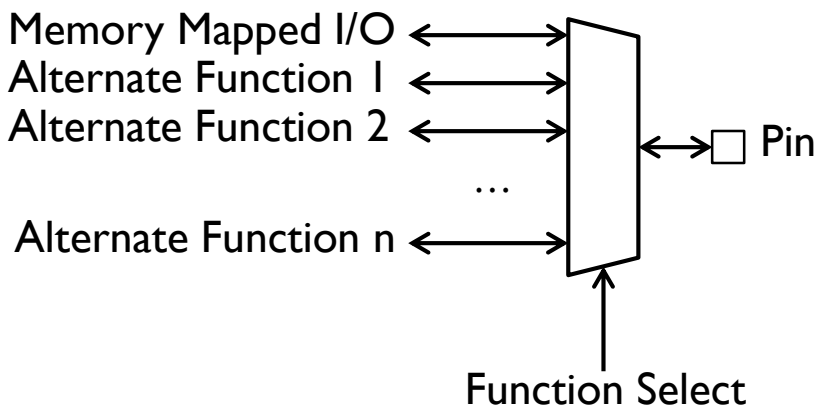


- Goal: light either LED1 or LED2 based on switch SW1 position
- GPIO = General-purpose input and output (digital)
  - Input: program can determine if input signal is a 1 or a 0
  - Output: program can set output to 1 or 0
- Can use this to interface with external devices
  - Input: switch
  - Output: LEDs

ARM

GPIO Alternative Functions

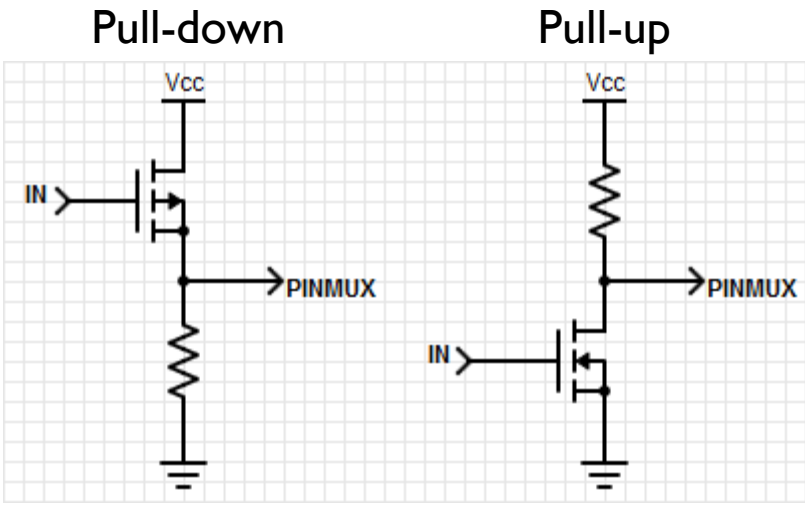
- Pins may have different features
- To enable an alternative function, set up the appropriate register
- May also have analogue paths for ADC / DAC etc.
- Advantages:
  - Saves space on the package
  - Improves flexibility



ARM

Pull-Up & Pull-Down Resistors

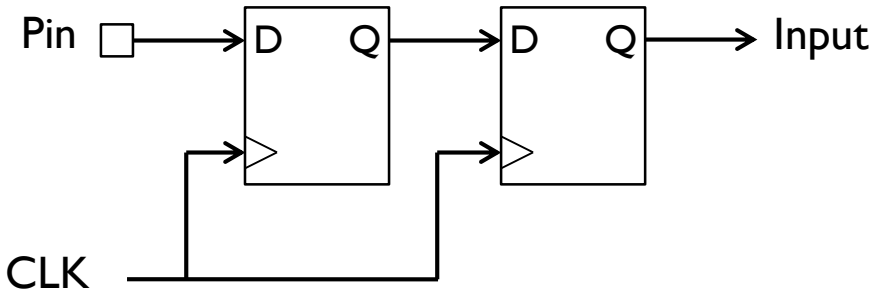
- Ensure a known value on the output if a pin is left floating
- In our example, we want the switch SW1 to pull the pin to ground, so we enable the pull-up
- The pin value is:
  - High when SW1 is not pressed
  - Low when SW1 is pressed



ARM

Input Synchronisation

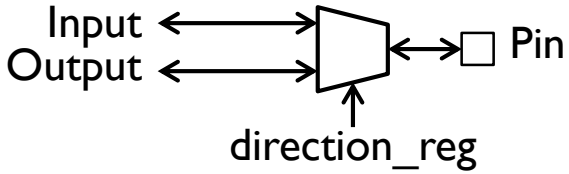
- External signals are asynchronous to internal clock
- If an external signal changes at the same time as the clock, a flip-flop can enter a metastable state indefinitely
- Solution – synchronise the input signals with the clock
- This is done for us by hardware, no need to worry!



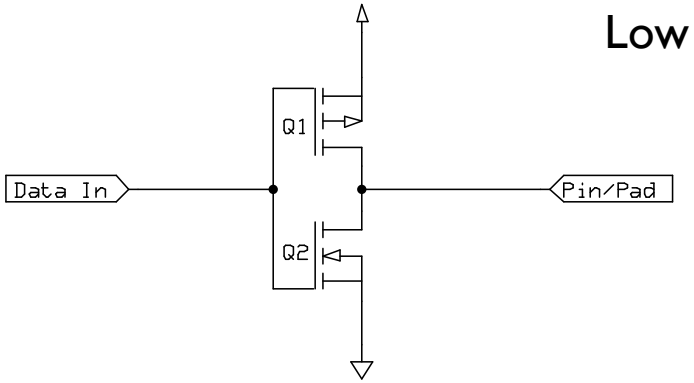
ARM

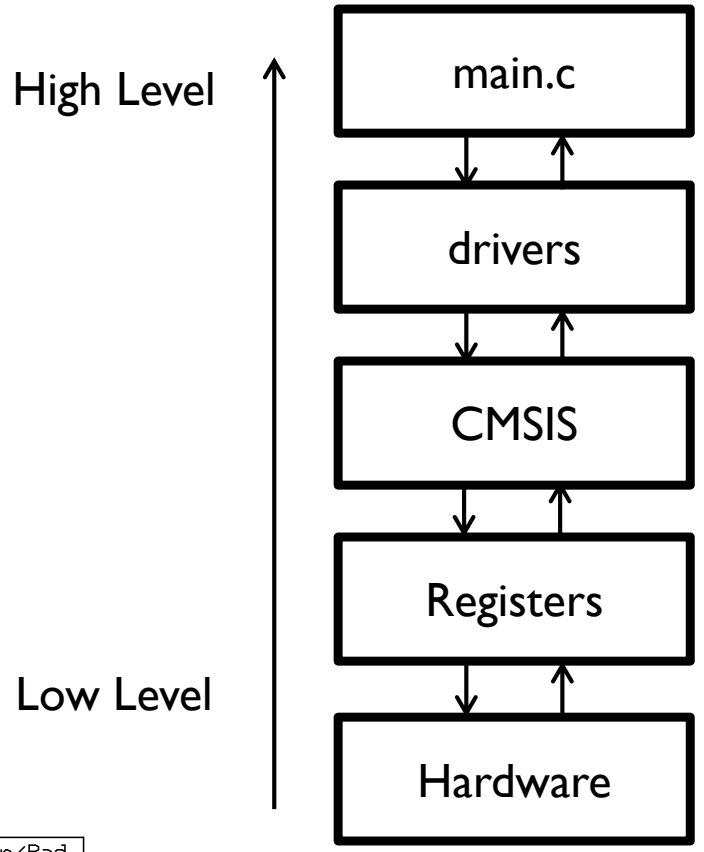
Code Structure

- Main code talks to the drivers, producing easy to read and understand code
  - gpio_set_mode(P2_5, Output)
- Drivers utilise CMSIS library and group relevant actions
  - port_struct->direction_reg = output
- CMSIS transforms memory mapped registers into C structs
  - #define PORT0 ((struct PORT*)0x2000030)
- Registers directly control hardware



- Hardware drives IO pins physically





ARM

Drivers Layer: How It Works

```
void gpio_set(Pin pin, int value)
1)    mask = 1 << pin index
2)    tmp = port_struct->data_reg & ~mask
3)    tmp |= value << pin index
4)    port_struct->data_reg = tmp
```

- e.g. gpio_set(P2_5, 1) with PORT_DATA_REGISTER = 0b01010101
  1. Create a mask for the bit we want to set (0b00100000)
  2. Invert the mask (0b11011111) to select all the other bits in the port data register, and save the status of the other bits (tmp = 0b01010101)
  3. Move the new value of the bit into position, and or it with the new register value (tmp = 0b01110101)
  4. Write the new data register value out to the port (PORT_DATA_REGISTER = 0b01110101)

ARM

Drivers Layer: How It Works

```
int gpio_get(Pin pin)
1)    mask = 1 << pin index
2)    tmp = port_struct->data_reg & mask
3)    tmp >>= pin index
4)    return tmp
```

- e.g. gpio_get(P2_5) with PORT_DATA_REGISTER = 0b01110101
  1. Create a mask for the bit we want to get (0b00100000)
  2. Select the bit in the port data register based on the mask (tmp = 0b00100000)
  3. Bitshift the value to produce a one or zero (tmp = 0b00000001)
  4. Return the value of the pin back to the user

ARM

C Interface: GPIO Configuration

```
/*! This enum describes the directional setup of a GPIO pin. */
typedef enum {
        Reset,   //!< Resets the pin-mode to the default value.
        Input,   //!< Sets the pin as an input with no pull-up or pull-down.
        Output,  //!< Sets the pin as a low impedance output.
        PullUp,  //!< Enables the internal pull-up resistor and sets as input.
        PullDown //!< Enables the internal pull-down resistor and sets as input.
} PinMode;

/*! \brief Configures the output mode of a GPIO pin.
 *  Used to set the GPIO as an input, output, and configure the
 *  possible pull-up or pull-down resistors.
 *  \param pin   Pin to set.
 *  \param mode  New output mode of the pin.
 */
void gpio_set_mode(Pin pin, PinMode mode);
```


ARM

C Interface: Reading and Writing

```
/*! \brief Sets a pin to the specified logic level.
 *  \param pin   Pin to set.
 *  \param value New logic level of the pin (0 is low, otherwise high).
 */
void gpio_set(Pin pin, int value);

/*! \brief Get the current logic level of a GPIO pin.
 *  \param  pin   Pin to read.
 *  \return The logic level of the GPIO pin (0 if low, 1 if high).
 */
int gpio_get(Pin pin);
```


ARM

Pseudocode for Program

```
Make LED1 and LED2 outputs
Make switch an input with a pull-up resistor
do forever {
      if switch is not pressed {
            Turn off LED1
            Turn on LED2
      } else {
            Turn off LED2
            Turn on LED1
      }
}
```


ARM

C Code

```
gpio_set_mode(P_LED1, Output); // Set LED pins to outputs
gpio_set_mode(P_LED2, Output);
gpio_set_mode(P_SW, Pullup); // Switch pin to resistive pull-up

while (1) {
        if (gpio_get(P_SW)) {
                // Switch is not pressed (active low), turn LED1 off and LED2 on.
                gpio_set(P_LED1, 0);
                gpio_set(P_LED2, 1);
        } else {
                // Switch is pressed, turn LED2 off and LED1 on.
                gpio_set(P_LED2, 0);
                gpio_set(P_LED1, 1);
        }
}
```


ARM

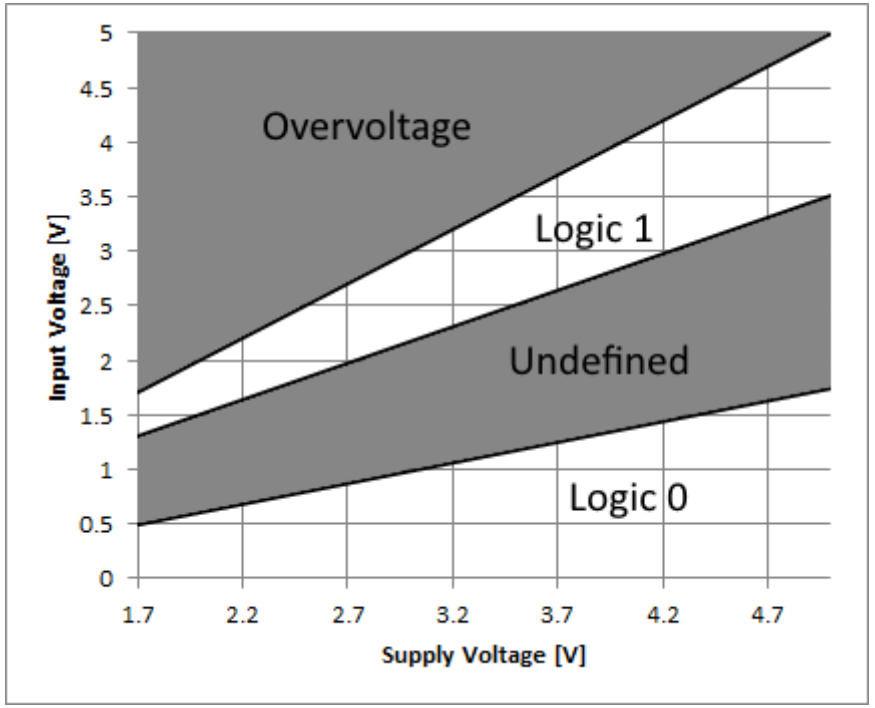
Inputs and Outputs, Ones and Zeros, Voltages and Currents

INTERFACING

ARM

Inputs: What's a One? A Zero?

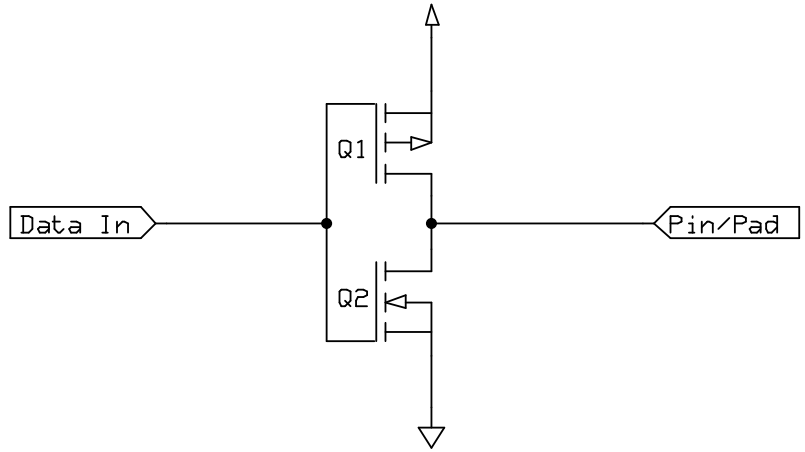
- Input signal's value is determined by voltage
- Input threshold voltages depend on supply voltage VDD
- Exceeding VDD or GND may damage chip

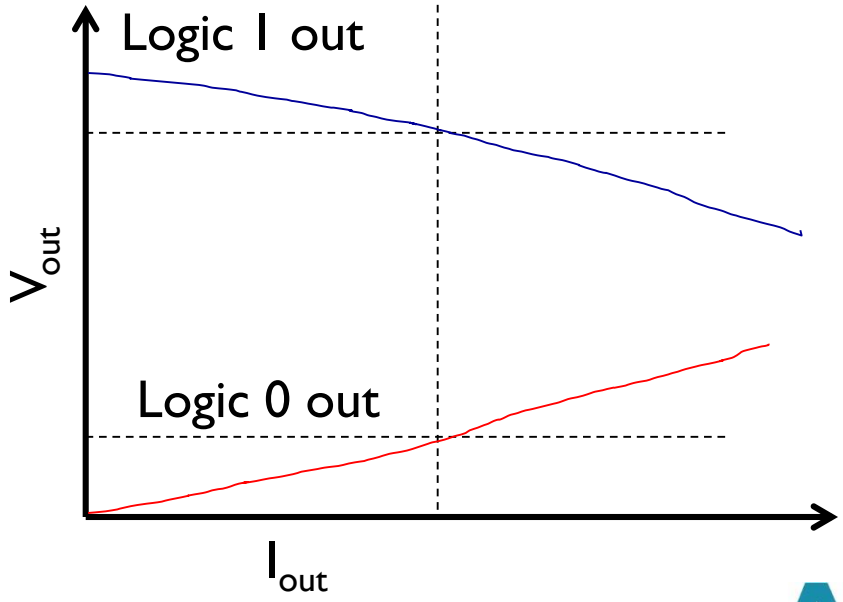


ARM

Outputs: What's a One? A Zero?

- Nominal output voltages
  - 1: VDD-0.5 V to VDD
  - 0: 0 to 0.5 V
- Note: Output voltage depends on current drawn by load on pin
  - Need to consider source-to-drain resistance in the transistor
  - Above values only specified when current < 5 mA (18 mA for high-drive pads) and VDD > 2.7 V

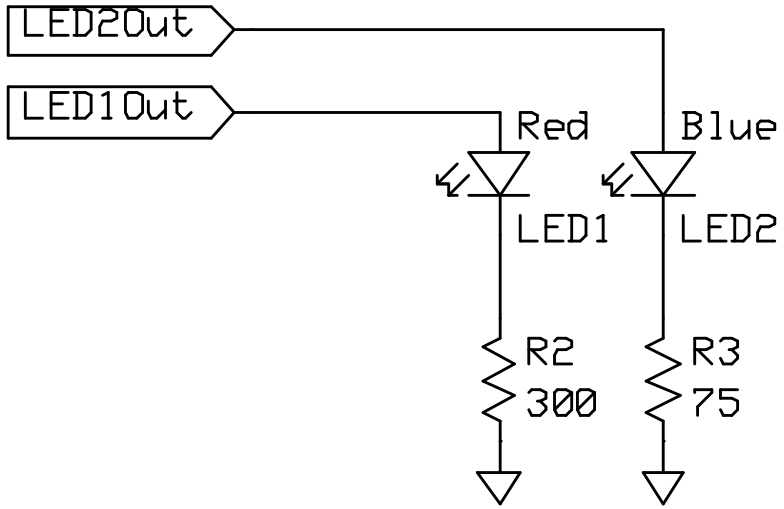




ARM

Output Example: Driving LEDs

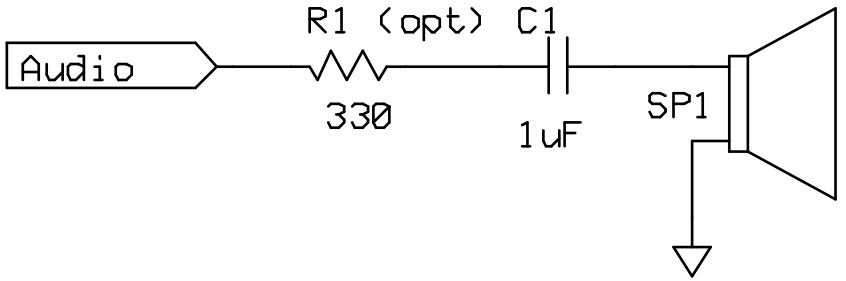
- Need to limit current to a value which is safe for both LED and MCU port driver
- Use current-limiting resistor
  - R = (VDD – VLED)/ILED
- Set ILED = 4 mA
- VLED depends on type of LED (mainly color)
  - Red: ~1.8V
  - Blue: ~2.7V
- Solve for R given VDD = ~3.0V
  - Red: 300 Ω
  - Blue: 75 Ω
- We will see the code next time



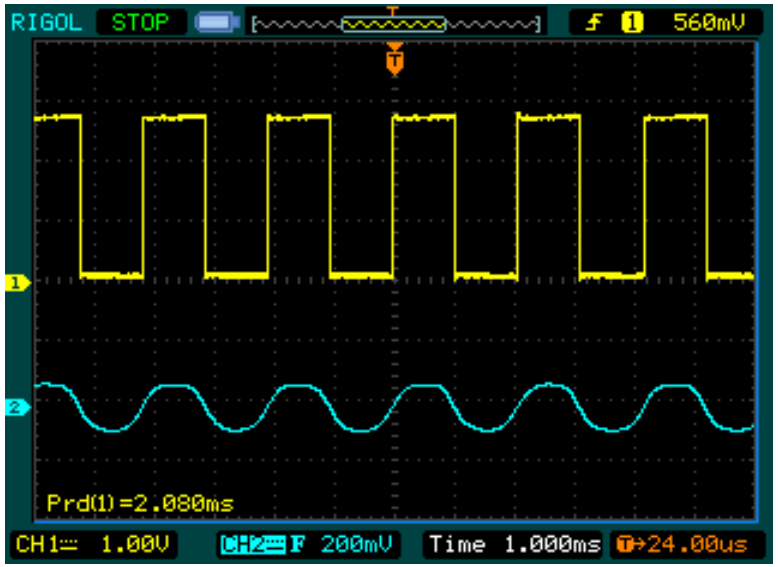
ARM

Output Example: Driving a Speaker

- Create a square wave with a GPIO output
- Use capacitor to block DC value
- Use resistor to reduce volume if needed




```
void beep(void) {
      unsigned int period = 20;
      while (1) {
            gpio_toggle(P_SPEAKER);
            delay_ms(period/2);
      }
}
```



ARM