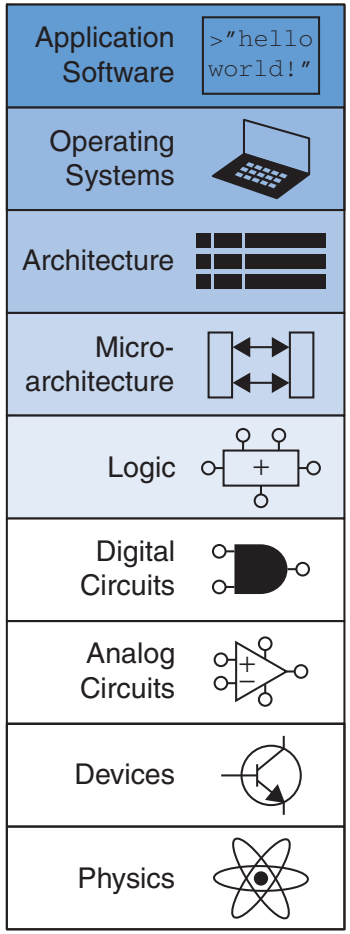
E85 Digital Design & Computer Engineering

Lecture 14:
Microcontrollers & Memory-Mapped I/O

HARVEY MUDD COLLEGE

Lecture 14

- Microcontrollers
- Memory-Mapped I/O
- General-Purpose I/O
- STM32
- Nucleo32 Board
- STM32 Memory-Mapped IO
- STM32 Clock Tree
- Delay Generation



Microcontrollers

- Microprocessors are computers on a chip
- Microcontrollers are microprocessors with flexible input/output (I/O) peripherals
- Peripheral examples
  - General-purpose I/O (GPIO): turn pins ON and OFF
  - Serial ports
  - Analog/Digital and Digital/Analog Converters (ADC/DAC)
  - Timers
  - Universal Serial Bus
  - Ethernet

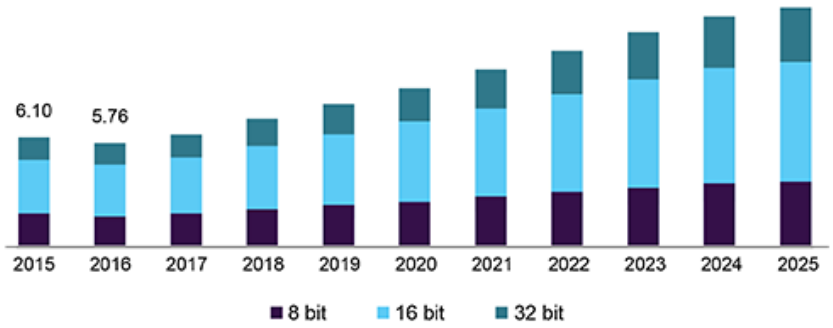
Embedded Systems

- Microcontrollers are commonly used in embedded systems
- An embedded system is a system whose user may be unaware there is a computer inside
  - Microwave oven
  - Clock/radio
  - Electronic fuel injection
  - Annoying toys with batteries for toddlers
  - Implantable glucose monitor

Microcontroller Market

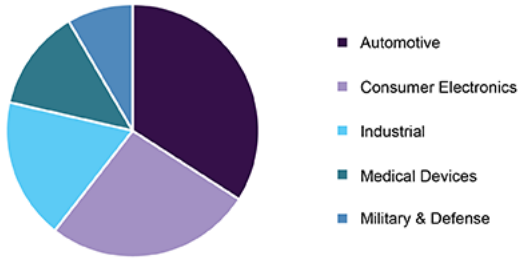
- Global Microcontroller market $18.6B in 2018
- Forecast to grow at 9%/year

North America microcontroller market size, by product, 2015 - 2025 (USD Million)



Source: www.grandviewresearch.com

Global microcontroller market share, by application, 2018 (%)



Source: www.grandviewresearch.com

Microcontroller Economics

- Highly cost-sensitive market
  - Often < $0.40; pennies matter
  - Cheaper than using a cable or a pushrod in a system
  - A microcontroller integrated on a larger chip can have a manufacturing cost of < $0.01
- Memory tends to dominate the cost
  - Select one with no more memory than necessary
- Classified as 8, 16, 32-bit based on internal bus
  - 1970s vintage 8-bit processors can still be adequate
  - 32-bit processors are faster and more flexible but use more code memory

Microprocessor Architectures

- The architecture is the native machine language a microprocessor understands.
- Commercial Examples
  - ARM
  - PIC
  - Intel x86
  - MIPS
  - PowerPC
  - RISC-V
  - 8051
  - Z80

Microcontroller Suppliers

Leading MCU Suppliers ($M)

2016 Rank	Company	2015	2016	% Change	% Marketshare
1	**NXP***	1,350	2,914	116%	19%
2	**Renesas**	2,560	2,458	-4%	16%
3	**Microchip****	1,355	2,027	50%	14%
4	**Samsung**	2,170	1,866	-14%	12%
5	**ST**	1,514	1,573	4%	10%
6	**Infineon**	1,060	1,106	4%	7%
7	**Texas Instruments**	820	835	2%	6%
8	**Cypress*****	540	622	15%	4%

*Acquired Freescale in December 2015.

**Purchased Atmel in April 2016.

***Includes full year of sales from Spansion acquisition in March 2015.

Source: IC Insights, company reports

Cortex M-Series Microcontrollers

- ARM Cortex M-Series microcontrollers are a market leader
  - Over 90B ARM processors have shipped
  - Over 75% of humans on Earth use a product with an ARM processor
- 32-bit performance with code density approaching 16-bit
- Robust ecosystem of chips, boards, compilers, libraries
- Inexpensive
  - Many competing suppliers

Memory-Mapped I/O

- Control peripherals by reading or writing memory locations called *registers*
  - Not really the same as a bank of flip-flops
- Just like accessing any other variable, but these registers cause physical things to happen.
- Portion of the address space is reserved for I/O registers rather than program or data.
- In C, use pointers to specify addresses to read or write.

General-Purpose I/O (GPIO)

- GPIOs can be driven to 0/1 or read.
- Examples: LEDs, switches, connect to other digital logic.
- In general:
  - Look up how many pins your microcontroller has, how they are named.
  - Each pin needs to be configured as an input, output, or special function.
  - Then read or write the pin.

STM32F042K6 Microcontroller

- Popular ARM Cortex-M0 microcontroller
- Used in labs for this class
- Manufactured by ST Microelectronics
- Available on an $10 Nucleo32 dev board
  - Unit cost of $1.14 in quantity of 10,000
- Many handy peripherals
- Good development environment

STM32F042K6

- STM32: ST Microelectronics family of 32-bit ARM Cortex microcontrollers
- F0: General Purpose Cortex-M0 Core
- All devices in the STM32F0 family have the same processor and general set of peripherals and share a datasheet
- 42: Sub-family
- K: 32 pins (LQFP32 surface mount package)
- 6: 32 KB Code memory

STM32F042K6 Pinout & Package

- 26 GPIO Pins
  - PA15:0, BP7:0, PF1:0
- 3 power, 2 GND, 1 reset

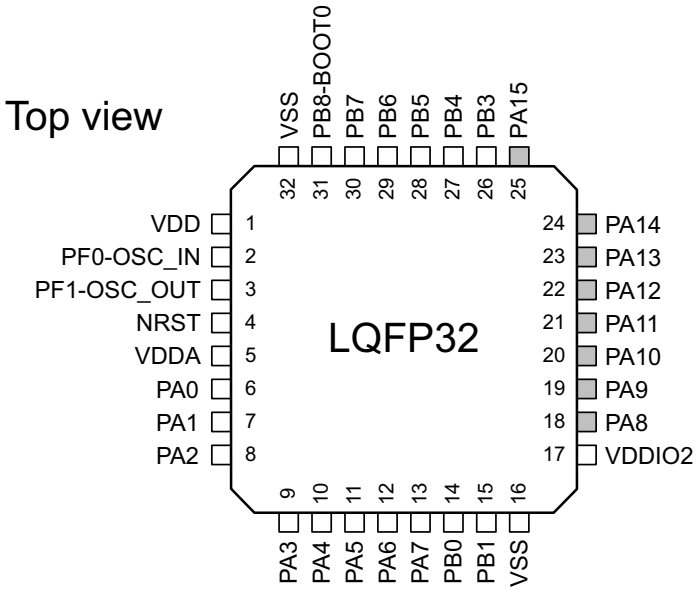
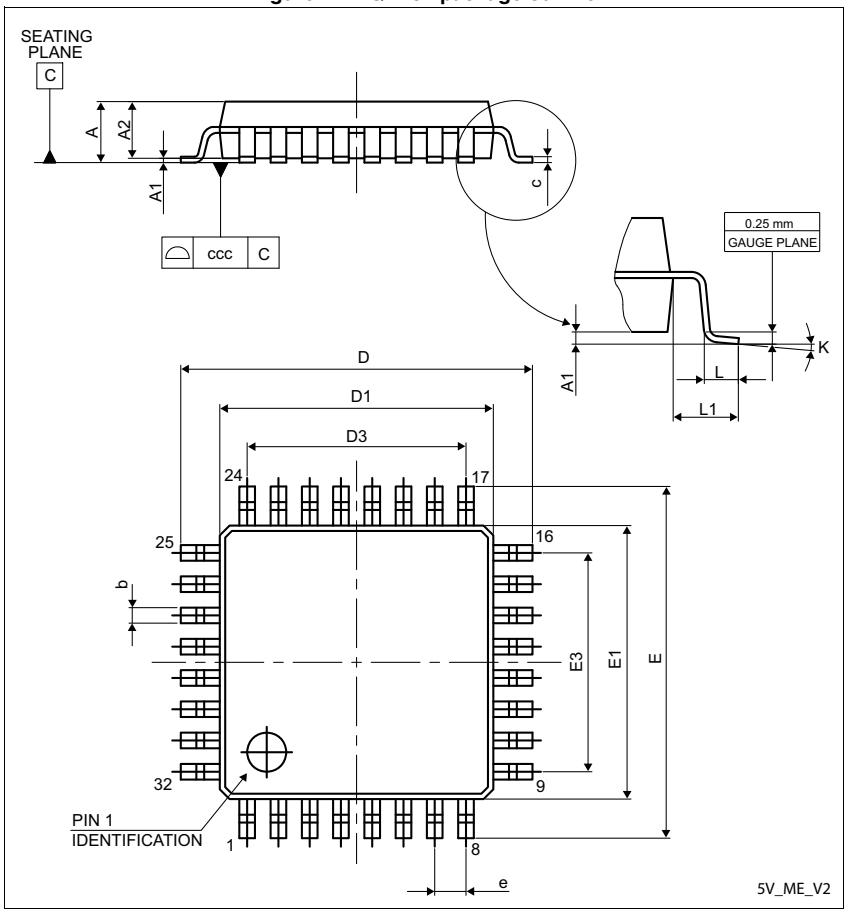


Figure 42. LQFP32 package outline



STM32F042x6 Datasheet

Nucleo32-F042K6 Development Board

- $10 Development Board
  - STM32F042K6 Microcontroller
  - 3.3V power supply (regulated from USB or external)
    - Use 3.3V output to power external circuitry
    - Turns off external circuits when Nucleo powers off, avoid damage
    - 5V inputs could damage the STM32
  - ST-Link Programmer/Debugger
    - Allows you to step through your code line by line and watch variables
  - 30 pins, suitable for breadboarding
    - Arduino Nano form factor and pin names
  - User LED
  - Reset button

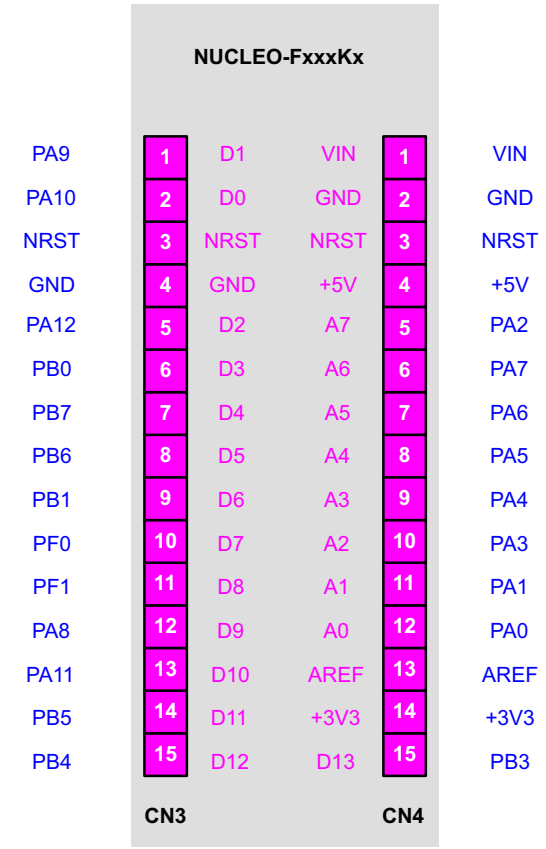
Nucleo Board Pin-Out

- Not all STM32 pins are tapped out to Nucleo board for lack of space
  - PA12:0
  - PB7:0 except PB2
  - PF1:0
- Weird Nucleo naming (Arduino-based)
  - D0-D13, A0-A7
- Rather random mapping of STM32 names to Nucleo names
- Some pins have special purposes
  - PB3 (D13) is connected to green on-board LED
  - PA12 (D2) is jumpered to ground for self-test. Don't use.
  - PB6 and PB7 (D4 and D5) are bridged to other pins and cannot be used reliably.
  - If you try to use D2, D4, or D5 in the lab, you will have a bad day

Nucleo Pin Numbering

Table 9. Arduino Nano connectors on NUCLEO-F042K6

Connector	Pin number	Pin name	STM32 pin	Function
Left connector				
CN3	1	D1	PA9	USART1_TX
	2	D0	PA10	USART1_RX
	3	RESET	NRST	RESET
	4	GND	-	Ground
	5	D2	PA12	-
	6	D3	PB0	TIM3_CH3
	7	D4(1)	PB7	-
	8	D5(1)	PB6	TIM16_CH1N(2)
	9	D6	PB1	TIM14_CH1
	10	D7(3)	PF0	-
	11	D8(3)	PF1	-
	12	D9	PA8	TIM1_CH1
	13	D10	PA11	SPI_CS(4) || TIM1_CH4
	14	D11	PB5	SPI1_MOSI || TIM3_CH2
	15	D12	PB4	SPI1_MISO
Right connector				
CN4	1	VIN	-	Power input
	2	GND	-	Ground
	3	RESET	NRST	RESET
	4	+5V	-	5V input/output
	5	A7	PA2	ADC_IN2(5)
	6	A6	PA7	ADC_IN7
	7	A5(1)	PA6	ADC_IN6 || I2C1_SCL
	8	A4(1)	PA5	ADC_IN5 || I2C1_SDA
	9	A3	PA4	ADC_IN4
	10	A2	PA3	ADC_IN3
	11	A1	PA1	ADC_IN1
	12	A0	PA0	ADC_IN0
	13	AREF	-	AVDD
	14	+3V3	-	3.3V input/output
	15	D13	PB3	SPI1_SCK



Nucleo32 Datasheet

STM Datasheets

- Information is spread across three datasheets
  - Learn to find the appropriate one for the task
- STM32F0xx Reference Manual
  - Information common to all STM32F0 microcontrollers
  - Memory maps, I/O Peripheral Operation
- STM32F042x6 Reference Manual
  - Information specific to STM32F042x6
  - Package pinouts
- NUCLEO32 User Manual
  - Pinouts and features of Nucleo32 board

STM GPIO Registers

- MODE: 2 bits for each pin on a port (reset to all 0s except a few upper bits of A)
  00 = input, 01 = output, 10 = alternate, 11 = reserved
- ODR, IDR: 1 bit for each pin on a port (only 16 bits used per port)
- GPIOA, GPIOB, GPIOF
- 0x4800 0000 – 0x4800 03FF GPIOA (p. 46)
- 0x4800 0400 – 0x4800 07ff GPIOB (p. 46)
- ModeR: offset 0 (Table 24 on p. 163)
- IDR: offset 10 (Table 24 on p. 163)
- ODR: offset 14 (Table 24 on p. 163)

	0xE000 0000 - 0xE00F FFFF	1MB	Cortex®-M0 internal peripherals	
	0x4800 1800 - 0x5FFF FFFF	~384 MB	Reserved	
AHB2	0x4800 1400 - 0x4800 17FF	1KB	GPIOF	*Section 8.4.12 on page 163*
	0x4800 1000 - 0x4800 13FF	1KB	GPIOE	*Section 8.4.12 on page 163*
	0x4800 0C00 - 0x4800 0FFF	1KB	GPIOD	*Section 8.4.12 on page 163*
	0x4800 0800 - 0x4800 0BFF	1KB	GPIOC	*Section 8.4.12 on page 163*
	0x4800 0400 - 0x4800 07FF	1KB	GPIOB	*Section 8.4.12 on page 163*
	0x4800 0000 - 0x4800 03FF	1KB	GPIOA	*Section 8.4.12 on page 163*

Table 24. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = A..F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	GPIOx_ODR (where x = A..F)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LED GPIO Example

- To turn on LED3 (PB3, Pin 3 of bank B of the GPIO)

1. Find location of GPIO bank B register (Table 1 on page 46)

	0xE000 0000 - 0xE00F FFFF	1MB	Cortex®-M0 internal peripherals	
	0x4800 1800 - 0x5FFF FFFF	~384 MB	Reserved	
AHB2	0x4800 1400 - 0x4800 17FF	1KB	GPIOF	*Section 8.4.12 on page 163*
	0x4800 1000 - 0x4800 13FF	1KB	GPIOE	*Section 8.4.12 on page 163*
	0x4800 0C00 - 0x4800 0FFF	1KB	GPIOD	*Section 8.4.12 on page 163*
	0x4800 0800 - 0x4800 0BFF	1KB	GPIOC	*Section 8.4.12 on page 163*
	0x4800 0400 - 0x4800 07FF	1KB	GPIOB	*Section 8.4.12 on page 163*
	0x4800 0000 - 0x4800 03FF	1KB	GPIOA	*Section 8.4.12 on page 163*

Nucleo Pin Numbering

- To turn on LED3 (PB3, Pin 3 of bank B of the GPIO)
  - ➢ Find Offset to MODE register (Table 24 on page 163)
  - ➢ Set MODER3 (Pin 3's mode) of GPIOB to 01

00 = input, 01 = output, 10 = alternate, 11 = reserved

Table 24. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nucleo Pin Numbering

- To turn on LED3 (PB3, Pin 3 of bank B of the GPIO)
  - ➢ Find Offset to MODE register (Table 24 on page 163)
  - ➢ Set MODER3 (Pin 3's mode) of GPIOB to 01

    00 = input, 01 = output, 10 = alternate, 11 = reserved

Table 24. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**GPIOx_IDR** (where x = **A..F**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = **A..F**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nucleo Pin Numbering

- To turn on LED3 (PB3)

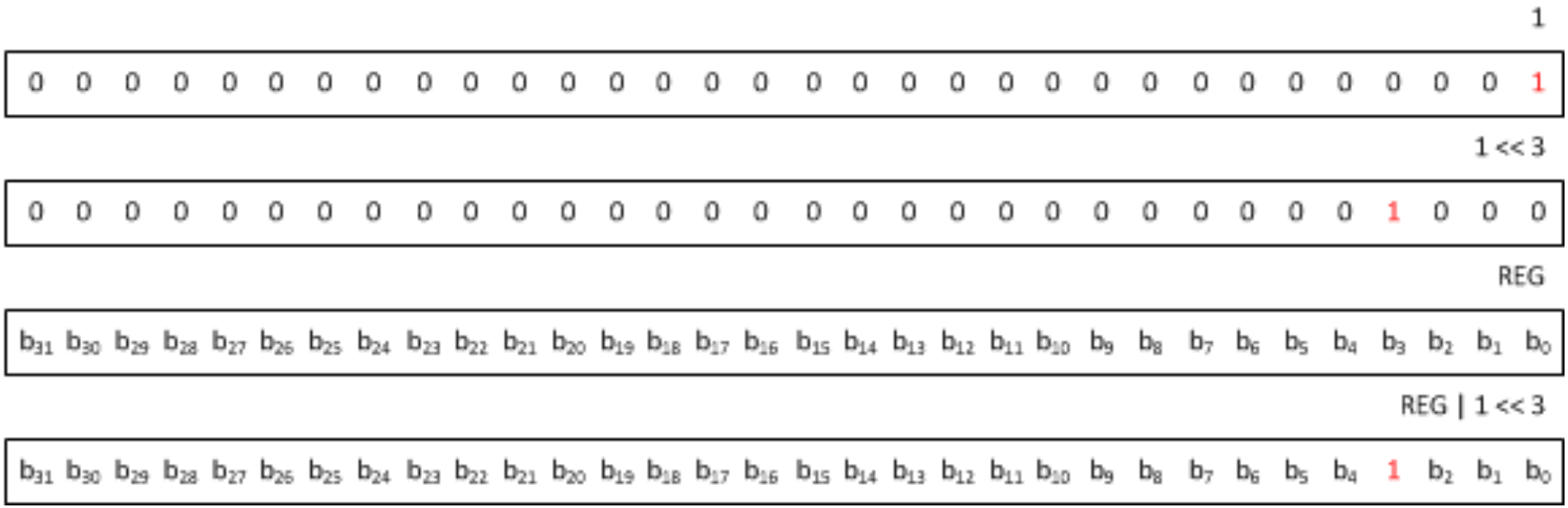
```
volatile unsigned long *GPIOB_MODER = (unsigned long*)0x48000400;
volatile unsigned long *GPIOB_ODR = (unsigned long*)0x48000414;
```

- volatile means access memory / peripherals directly each time

```
*GPIOB_MODER |= (1 << 6);  // set PB3 mode to output
*GPIOB_ODR |= (1 << 3);    // turn on PB3
```

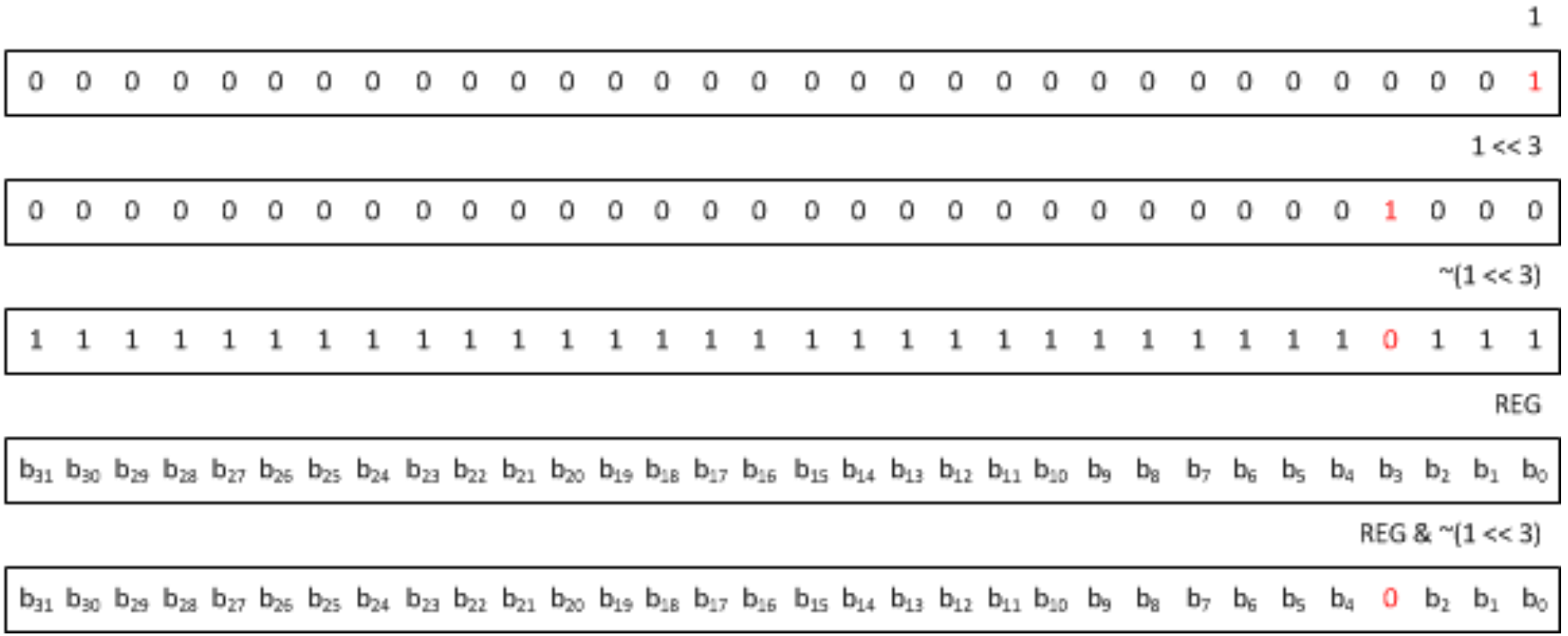
C Idioms: Write Bit to 1

```
*REG |= (1 << 3);     // turn on bit 3
```



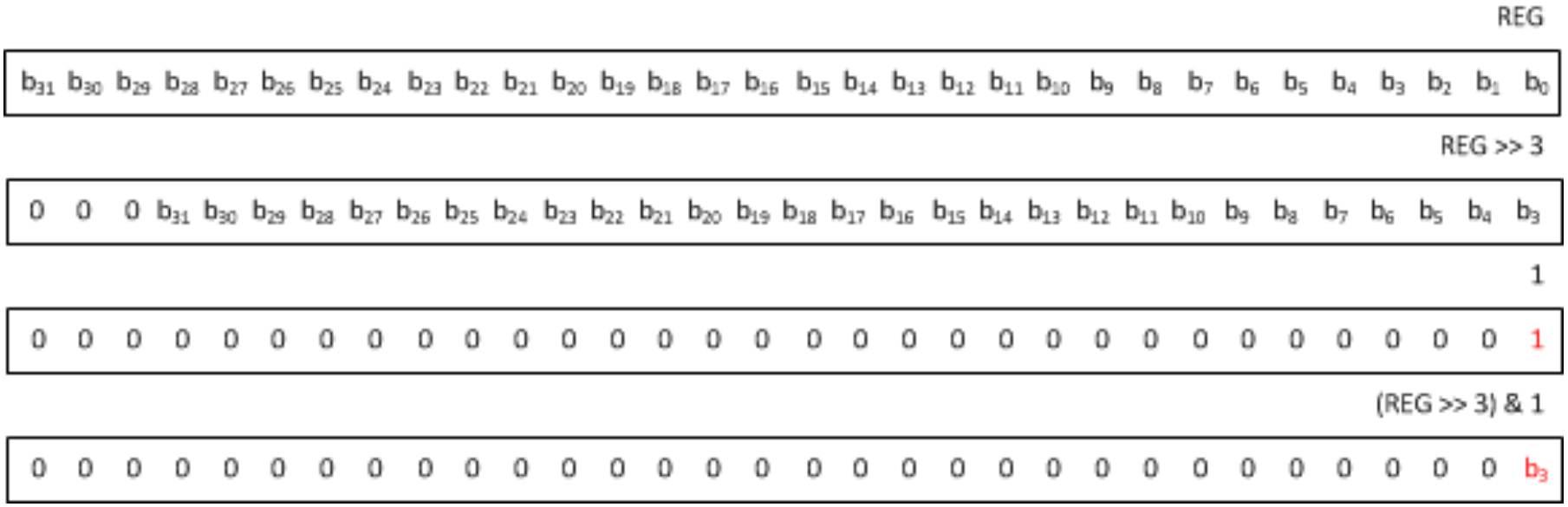
C Idioms: Write Bit to 0

```
*REG &= ~(1 << 3);     // turn off bit 3
```



C Idioms: Read value of bit

```
int bit = (*REG >> 3) & 1; // get value of bit 3
```



C Idioms: Wait for a bit

- Wait for bit 3 to become 1

```
while (!((*REG >> 3) & 1)); // wait for bit 3
```

- Wait for bit 3 to become 0

```
while (((*REG >> 3) & 1)); // wait for bit 3
```

Clock Tree Configuration

- STM processors have many options for clock sources
- Simplest is the HSI clock
- Comes from ~8 MHz on-chip RC oscillator
- Speed will vary ~1% from chip to chip at 25°C, and more across temperature

Peripheral Clocks

- All clocks to peripherals default to OFF to save power
- You will need to turn them on
- The RCC AHBENR register enables clocks (see ref manual page 120)
  - RCC addresses 0x40021000 (p. 46)
  - AHBENR offset is 0x14
  - So RCC_AHBENR is at 0x40021014
  - Bit 18 enables port B clock

```
volatile unsigned long *RCC_AHBENR = (unsigned long*)0x40021014;
*RCC_AHBENR |= (1 << 18);
```

- needed to turn on LED on PB3

Generating Delays

```
// Delay constant
#define COUNTSPERMS 898 //Counts Per millisecond

void delayLoop(int ms) {
  // declare loop counter volatile so it isn't optimized away
  // countsperms empirically determined such that
  // delayLoop(100) waits 100 ms

  volatile int i = COUNTSPERMS * ms;

  while (i--); // count down time
}
```