nRF52840 Introduction

Introduction to the nRF52840 System-on-chip(SoC) and the nRF52840 Development Kit

Andreas Haugland

NTNU

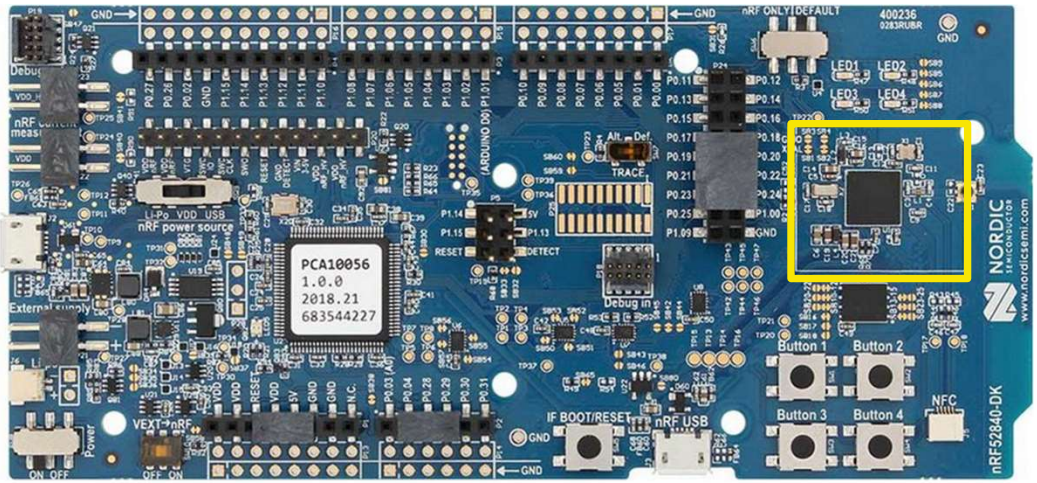
2024

nRF52840 Overview

Radio	BLE / ANT / 2.4 GHz / IEEE 802.15.4 2 Mbps / Long Range +8 dBm
Processor	▪ Cortex M4F ▪ 64 MHz
Memory	▪ 1024 kB Flash w/cache ▪ 256 kB RAM with EasyDMA
Power management	▪ 1.7 to 5.5V supply voltage ▪ LDO and Buck DC/DC

Digital Interfaces	4x SPI Master or Slave 2x TWI Master or Slave 2x UART 4x PWM QDEC PDM I2S NFC USB
Analog interfaces	▪ 8-channel 10/12-bit ADC ▪ 15-level LP Comparator ▪ 64-level GP Comparator
GPIO	▪ 48 pins
System peripherals	▪ 5 x 32-bit Timers (2 with 6 CC Regs) ▪ 3 x Real-Time Counters(RTC) ▪ Programmable Peripheral Interconnect(PPI)

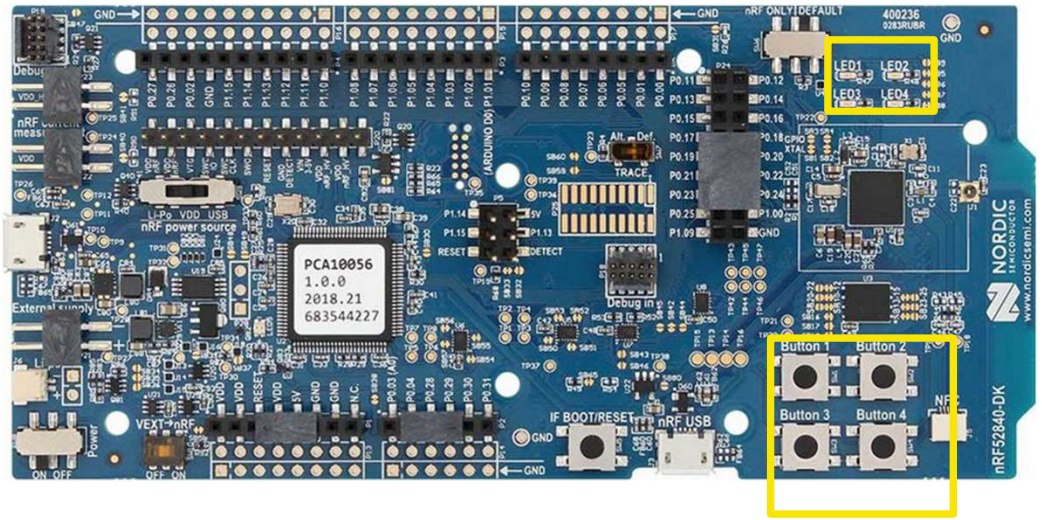
nRF52840 Development Kit



Key Features

- nRF52840 SoC

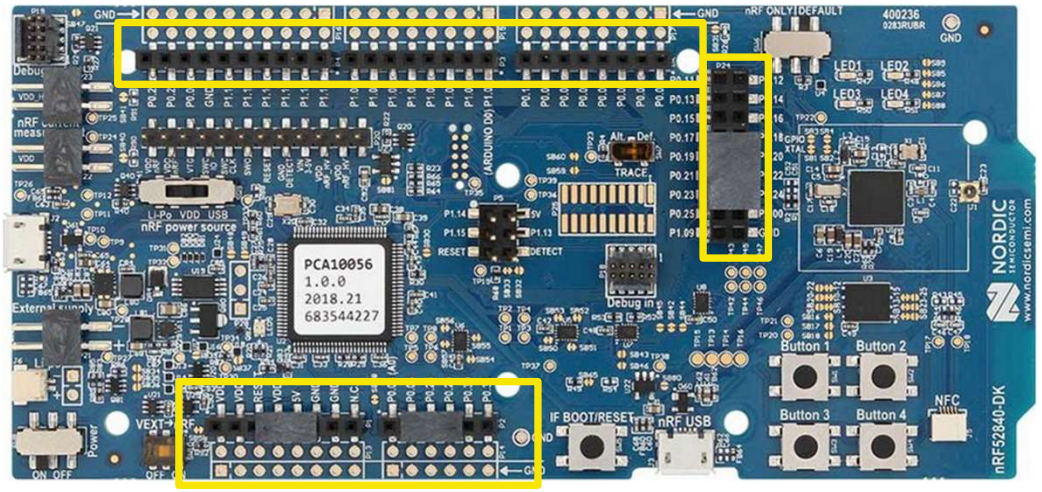
nRF52840 Development Kit



Key Features

- nRF52840 SoC
- Buttons and LEDs for user interaction

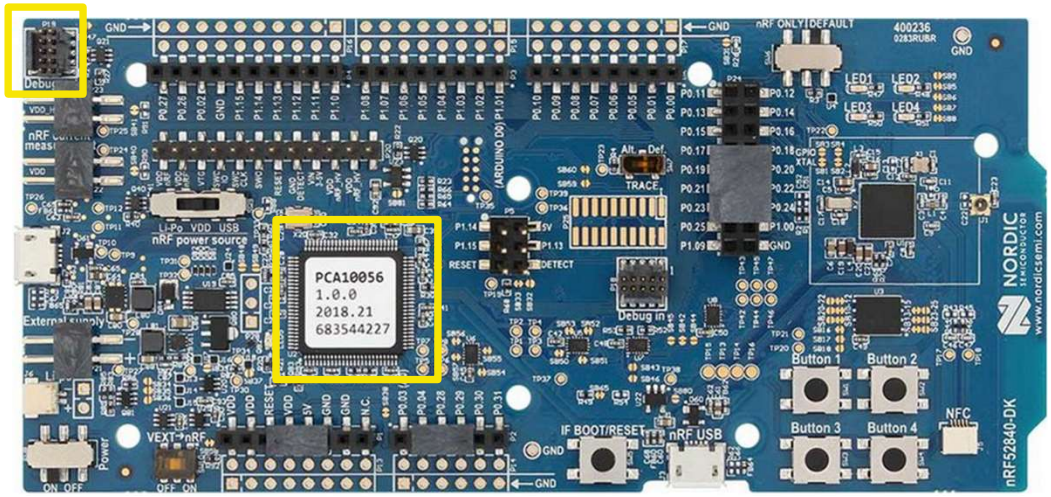
nRF52840 Development Kit



Key Features

- nRF52840 SoC
- Buttons and LEDs for user interaction
- I/O interface for Arduino form factor plug-in modules

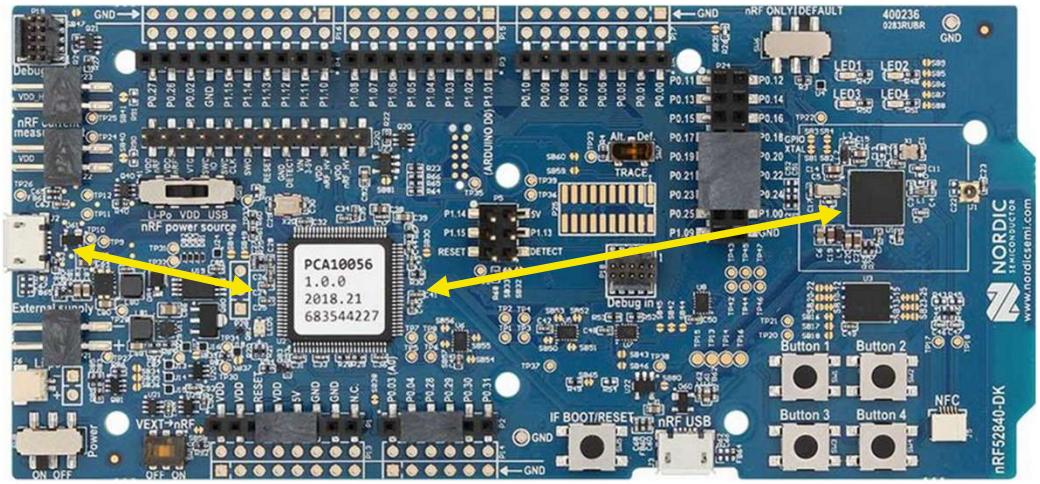
nRF52840 Development Kit



Key Features

- nRF52840 SoC
- Buttons and LEDs for user interaction
- I/O interface for Arduino form factor plug-in modules
- Segger J-link OB Debugger with debug out functionality

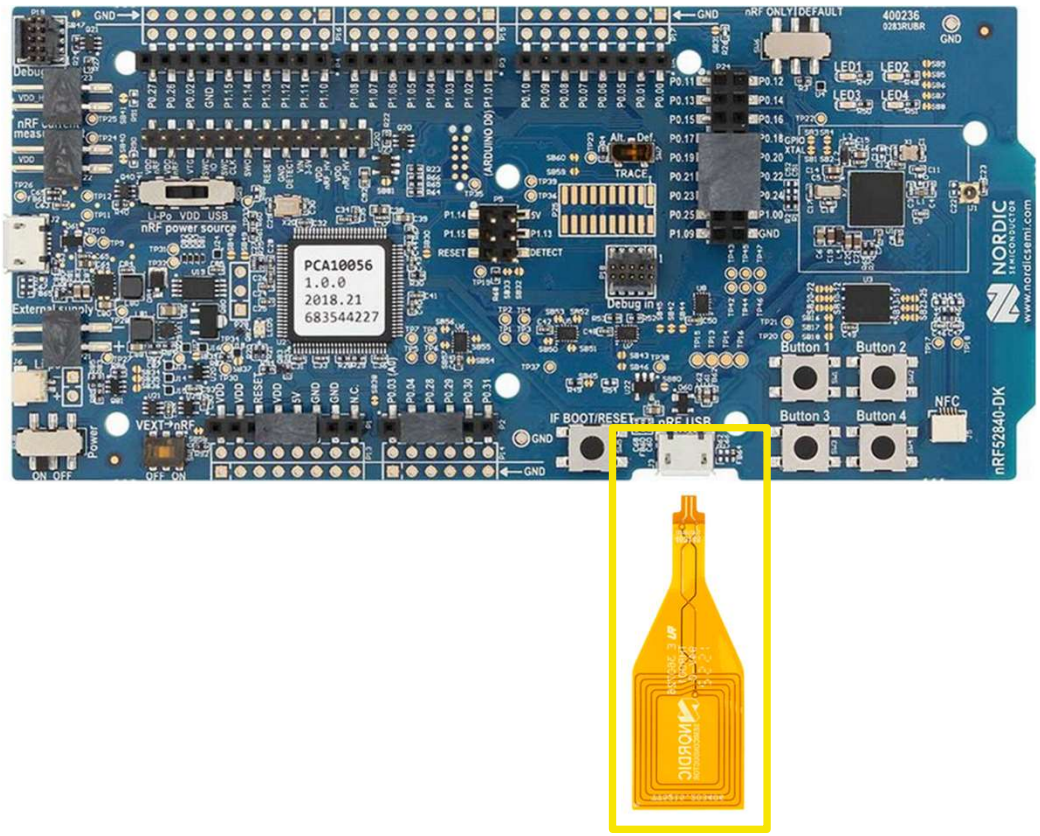
nRF52840 Development Kit



Key Features

- nRF52840 SoC
- Buttons and LEDs for user interaction
- I/O interface for Arduino form factor plug-in modules
- Segger J-link OB Debugger with debug out functionality
- Virtual COM Port interface via UART
- Drag-and-drop Mass Storage Device programming

nRF52840 Development Kit



Key Features

- nRF52840 SoC
- Buttons and LEDs for user interaction
- I/O interface for Arduino form factor plug-in modules
- Segger J-link OB Debugger with debug out functionality
- Virtual COM Port interface via UART
- Drag-and-drop Mass Storage Device programming
- NFC-A Support

nRF52840 Introduction

About some of the peripherals on the nRF52840

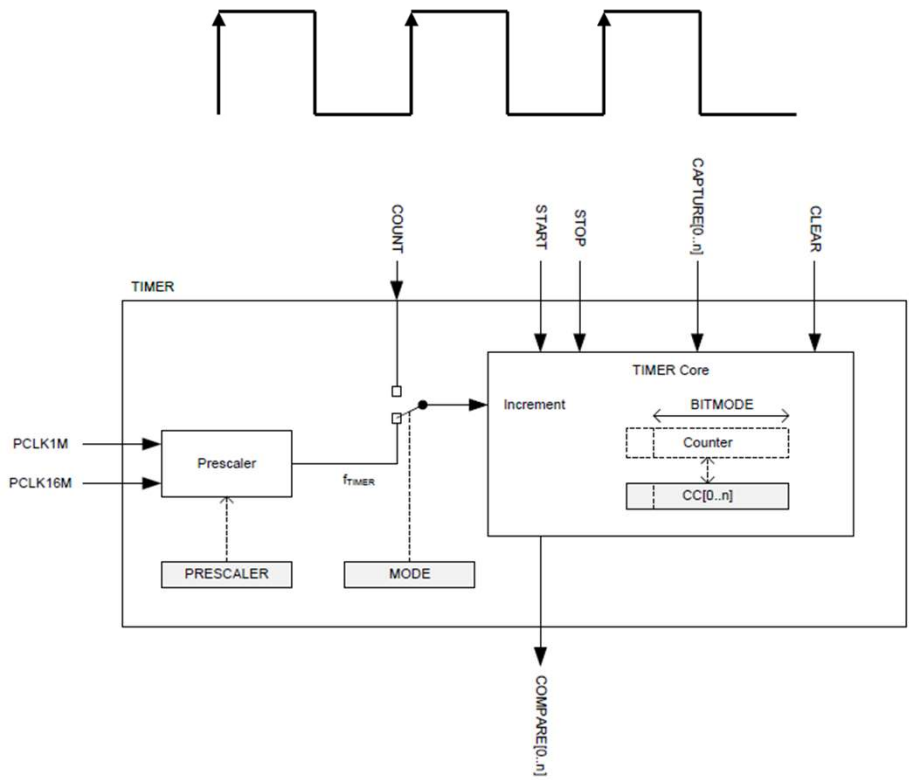
Andreas Haugland

NTNU

2024

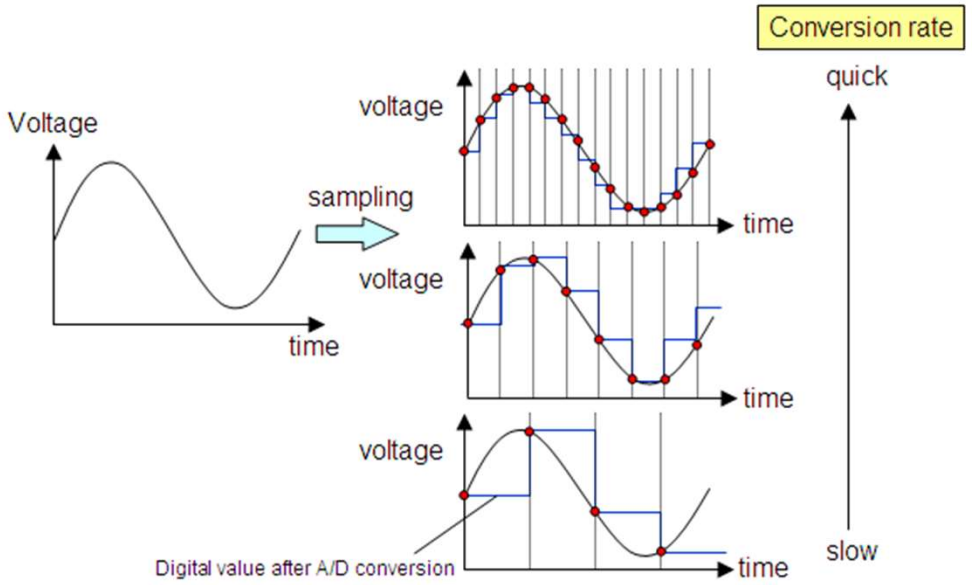
Timers/Counters

- Counts the clock cycles of the system clock
- Comes in 8-bit, 16-bit, 24-bit or 32-bit configurations(usually configurable)
- COUNTER register keeps track of the number of clock cycles
- Compares Counter with Capture/Compare(CC) registers.
- Prescaler determine how often the Counter is updated, e.g prescaler of 8 will result in the counter incrementing every 8th clock cycle



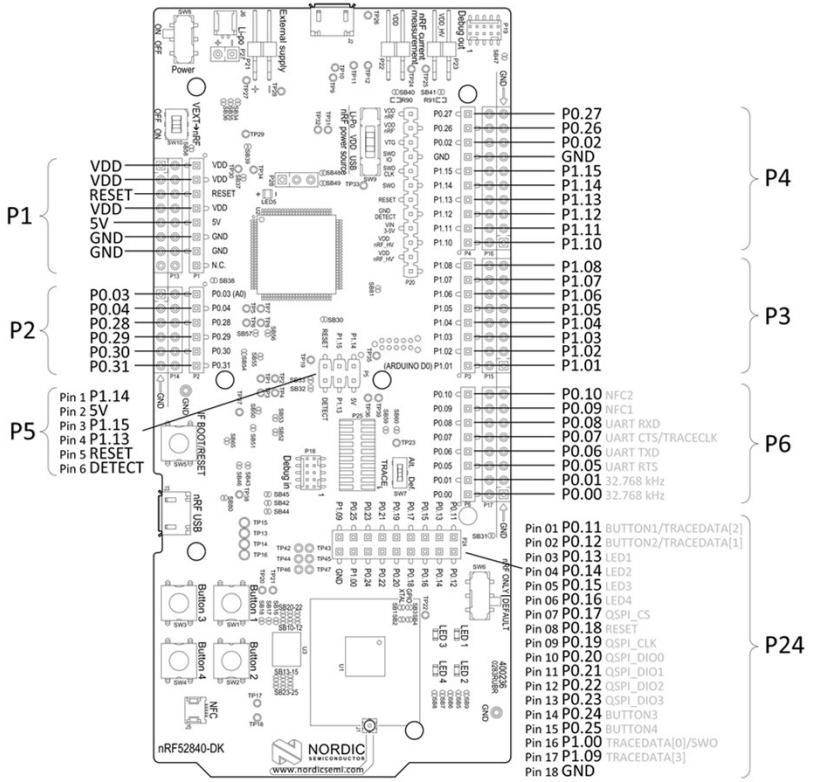
Analog-Digital Converter(ADC)

- Converts an analog voltage on a pin to a digital number.
- Ratiometric value
- $$\frac{ADC\ Resolution}{Reference\ Voltage \times ADC\ value} = V_{measured}$$
- Resolution:
  - 8-bit: 256(2^8) discrete levels
  - 10-bit: 1,024 (2^{10}) discrete levels
  - 16-bit: 65,536 (2^{16}) discrete levels
- ADCs are fairly complex!



General Purpose Input/Output(GPIO) pins

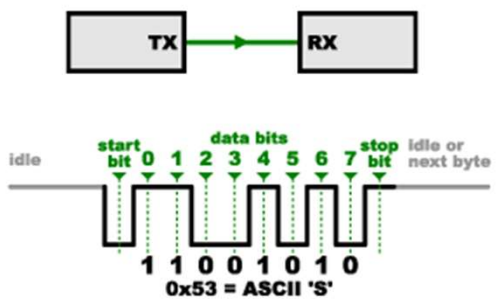
- Generic pin controllable by the user at run time.
- Arranged in ports, pins per port is determined by architechture, e.g. 32-bit -> 32 pins/port
- Configurable parameters:
  - Input/Output
  - Digital/Analog
  - Internal pull-up and/or pull-down resistor
  - Drive strength

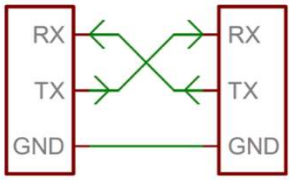


Serial Communication

Universal asynchrounous receiver/transmitter(UART)

- Asynchronous = data is transfer without no clock signal.
- Can be **simplex(1 line)**, **half-duplex(1/2 lines) or full-duplex(2 lines)**
- Data needs to be framed:
  - Data bits: Data that you want to sent
  - Synchronization bits: Start and stop bits
  - Parity bits: Used for error checking
- Both sides must agree Baudrate: Transfer speed

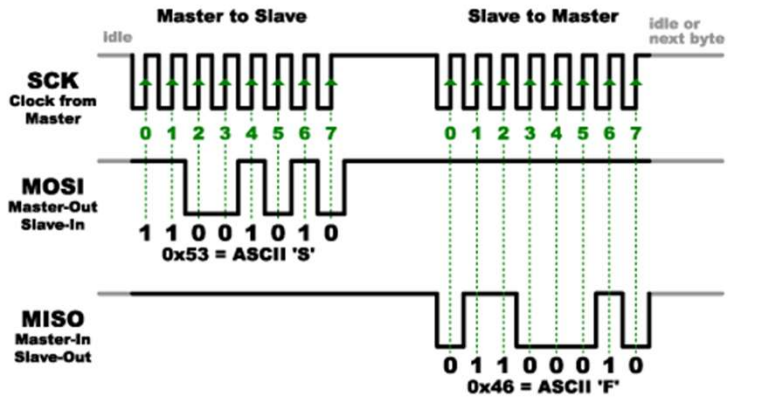


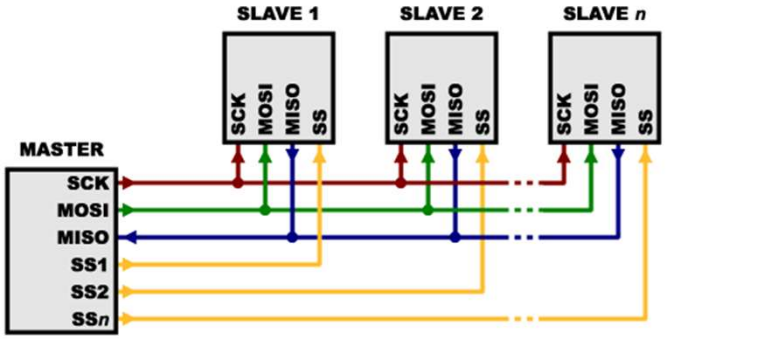


Serial Communication

Serial Peripheral Interface(SPI)

- Synchronous data transfer = clock signal
- Uses four lines
  - SCK – Clock signal
  - MOSI(Master-Output, Slave-Input)
  - MISO(Master-Input, Slave-Output)
  - SS – Slave Select
- Advantages
  - Full-duplex
  - Support clock rates upwards of 10MHz
  - Supports multiple slaves
- Disadvantages
  - Uses minimum 4 lines.
  - Each slave needs a separate SS line

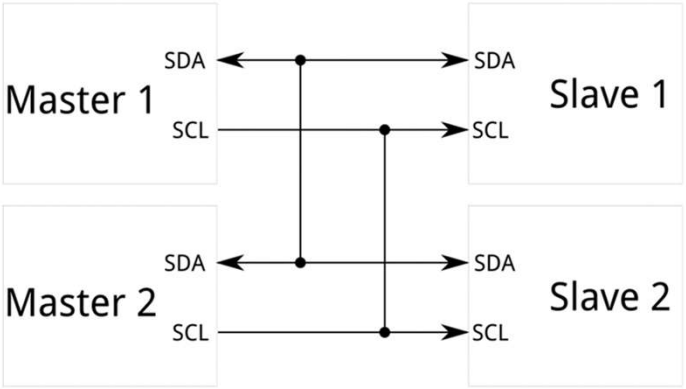


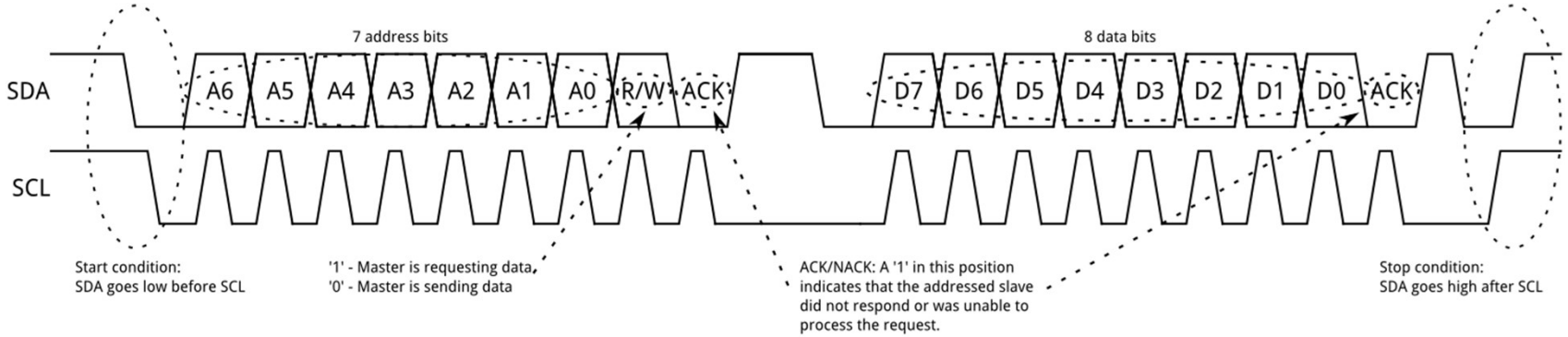


Serial Communication

Inter-integrated Circuit(I^2C)

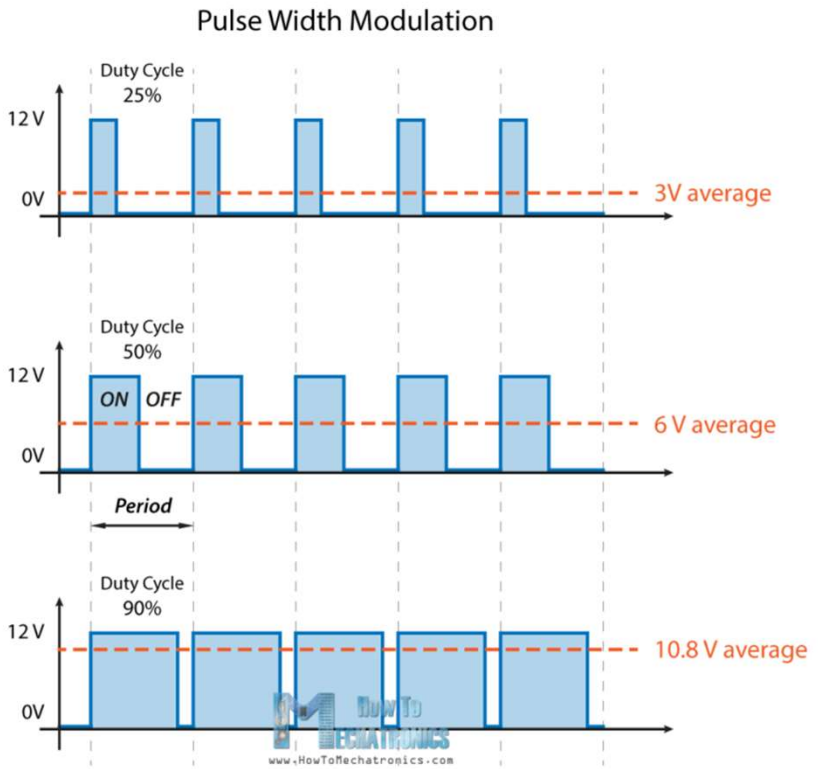
- I2C bus consists of two signals:
  - SCL is the clock signal
  - SDA is the data signal
- Half-duplex
- Clock speed at 100kHz or 400kHz





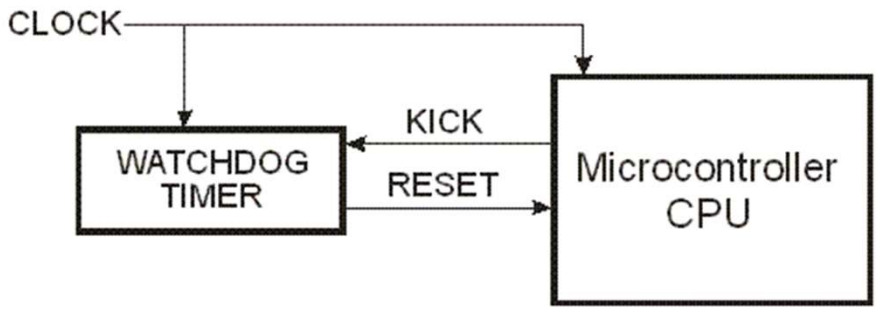
Pulse-Width Modulation(PWM)

- A method for generating an analog signal using a digital source.
- A digital signal is either **High** or **Low**, cannot be something in-between.
- A PWM signal is defined by its:
  - Frequency(Hz)
  - Duty cycle(percentage)



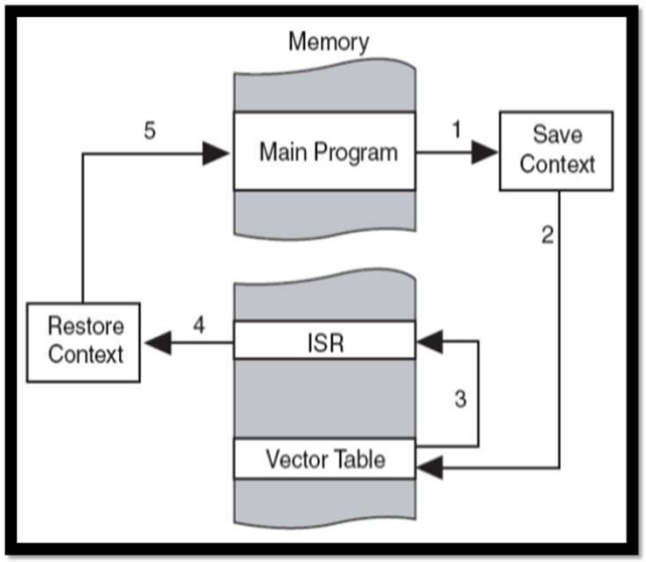
Watchdog Timer(WDT)

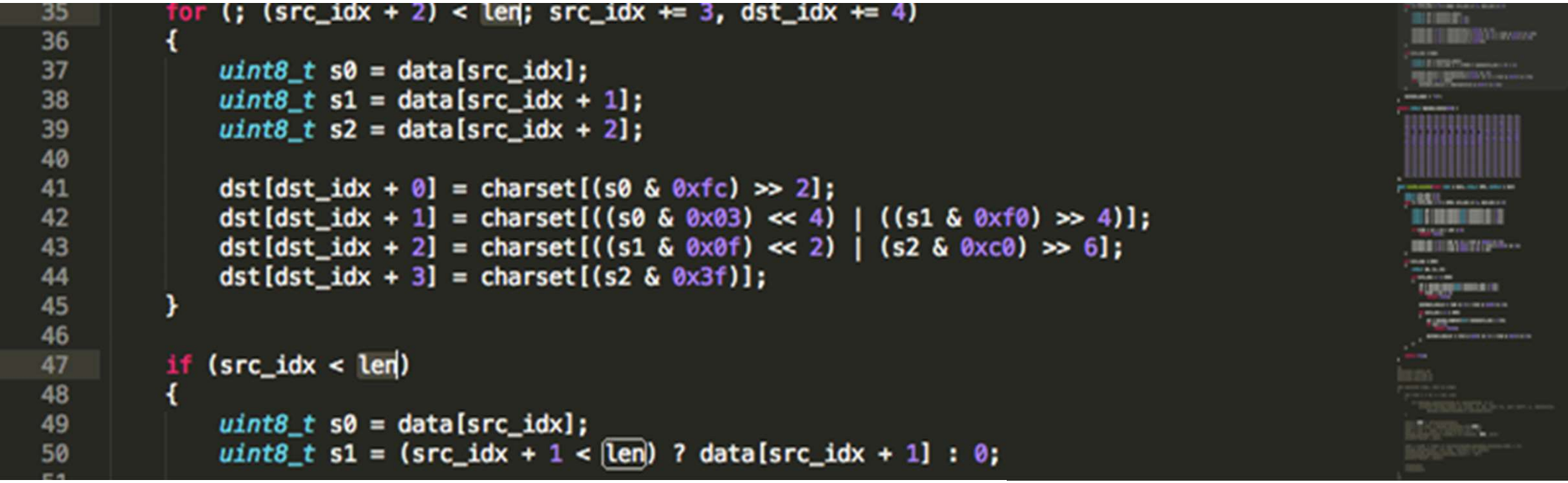
- Failsafe mechanism
- Counter starting at a predetermined value and counts down to zero.
- System will be reset if the watchdog counter reach zero.
- CPU must «feed» or «kick» the Watchdog to reset the counter.



Polling and Interrupts

- **Polling:** CPU continously monitors a variable/register to see if a specific task should be performed.
- **Interrupt:** CPU is free to execute its main code until an interrupt occurs, it will then branch to the Interrupt Service Routine or Interrupt Handler which contains code to do the task.
- Interrupts are more efficient than polling. (Mailbox anology)





Embedded C programming

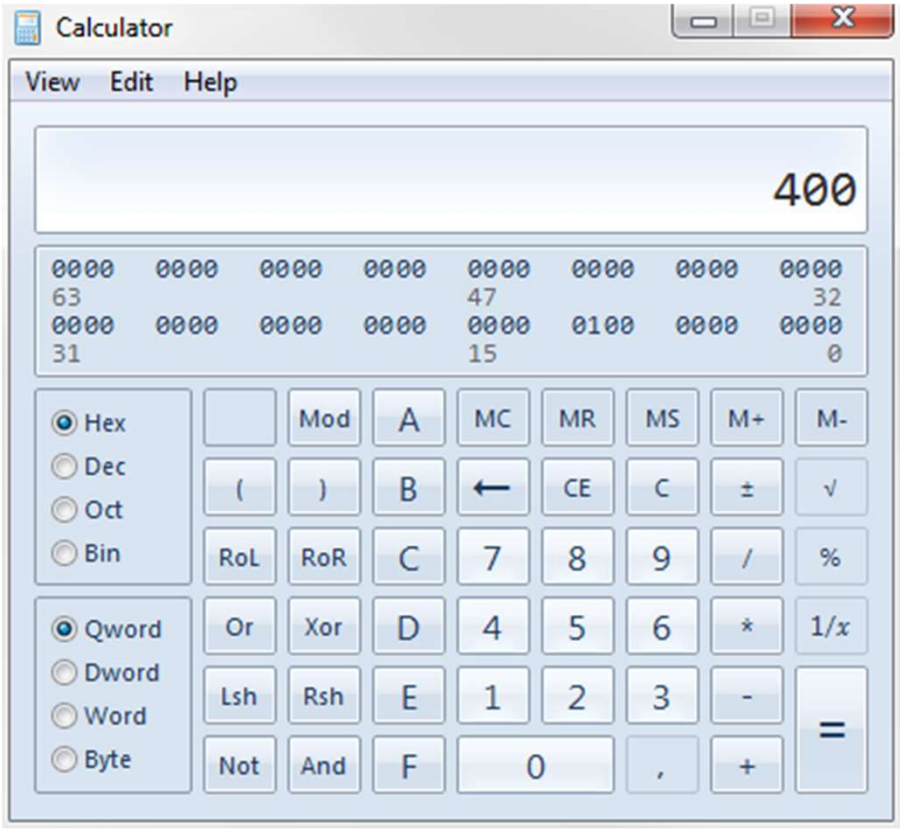
Short introduction to C programming on embedded devices and FA-«How-did-this-work-again»-Q

Data types

Data Type	Size (bytes)	Size (bits)	Value Range
unsigned char	1	8	0 to 255
signed char	1	8	-128 to 127
char	1	8	either
unsigned short	2	16	0 to 65,535
short	2	16	-32,768 to 32,767
unsigned int	4	32	0 to 4,294,967,295
int	4	32	-2,147,483,648 to 2,147,483,647
unsigned long	8	64	0 to 18,446,744,073,709,551,616
long	8	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	8	64	0 to 18,446,744,073,709,551,616
long long	8	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4	32	3.4E +/- 38 (7 digits)
double	8	64	1.7E +/- 308 (15 digits)
long double	8	64	1.7E +/- 308 (15 digits)
bool	1	8	false or true

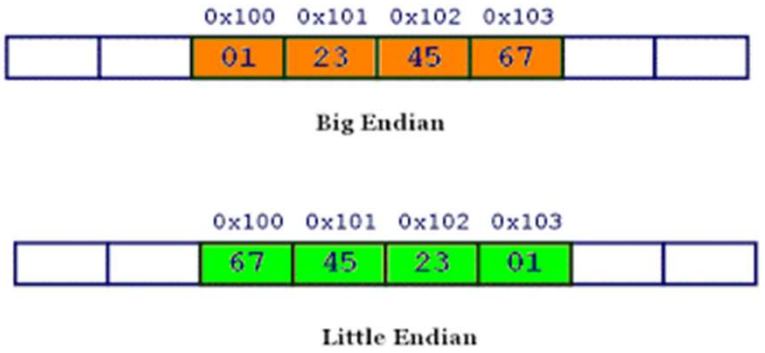
Binary & Hexadecimal

- Binary is base 2
  - Uses only 0 and 1
- Hexadecimal is base 16
  - Uses digits 0-9 and letters A-F
- 4 binary digits can be represented by 1 hexadeximal digit. ($16 = 2^4$)
- Numbers take up less space and are easier to write.
  Ex. 10 = 0b1010 = 0xA



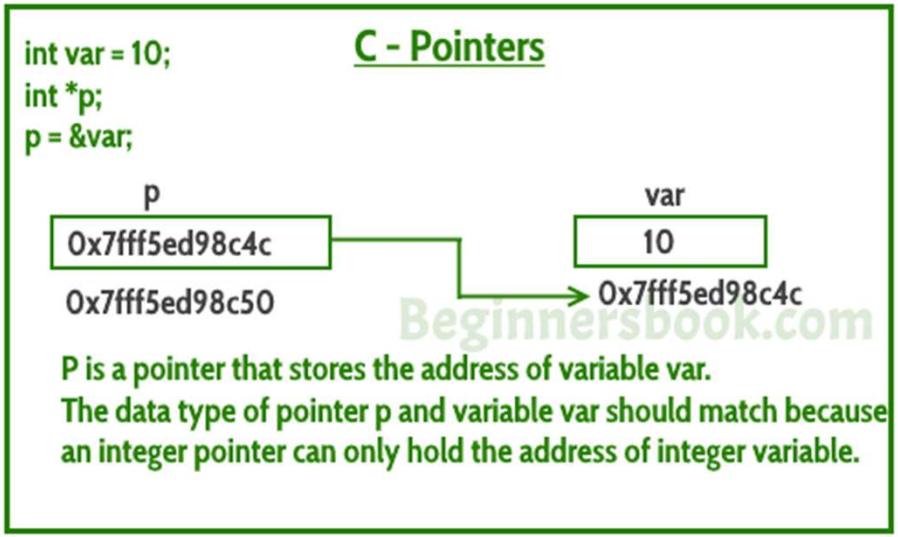
Endianess

- Byte/bit order.
- **Big endian:** most significant bits (MSBs)occupy the lower address.
- **Little endian:** least significant bits (LSBs) occupy the lower address.
- nRF52840 uses **Little Endian.**



Pointers

- A pointer points to the memory address of a variable.
- The pointer itself has an memory address.
- The pointer can be dereferenced to access the value at the memory address.
- * is called the dereference operator
- & is the reference operator.



Keywords

- Keywords are predefined, reserved words that have special meanings to the C compiler.
- 40 keywords in C, we'll concentrate on:
  - typedef
  - const
  - static
  - struct
  - void
- Assume that **if**, **else**, **for** and **while** are known.

Keywords in C Language

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Keywords - typedef

- The **typedef** keyword is used to explicitly associate a type with an identifier.
- More convenient to write *int8_t* than *signed char*
- Make the code more readable and understandable.

```
typedef signed    char      int8_t;
typedef unsigned  char      uint8_t;

typedef signed    short     int16_t;
typedef unsigned  short     uint16_t;

typedef signed    int       int32_t;
typedef unsigned  int       uint32_t;

typedef signed    long long int64_t;
typedef unsigned  long long uint64_t;
```

Keywords - struct

- The struct keyword is used for declaring a structure(often used in combination with typedef).
- A structure can hold member variables of different types under a single name.
- The . (dot) operator is used to access member variables
- The -> operator is used to access member variables from a structure pointer.

```
1
2   typedef struct person_t {
3       int age;
4       double weight;
5   }
6
7   person_t person_1;      // structure of type person_t
8   person_1.age = 28;
9   person_1.weight = 75;
10
11  printf("Age:",person_1.age);
12  printf("Weigth:", person_1.weight);
13
14  person_t * p_person;  // pointer to a structure of person_t
15  p_person = &person_1;
16
17  printf("Age:", p_person->age);
18  printf("Weigth:", p_person->weight);
19
20  // p_person->weight is the same as (*p_person).weight
21
```

Keywords – const and static

- A constant is a value or an identifier whose value cannot be altered in a program.
- static has two different meanings:
  1. A static global variable/function or a function is "seen" only in the file it's declared in.
  2. A static variable inside a function keeps its value between function calls.

```
const double pi = 3.14;
static int value = 5;

void counter()
{
    static int32_t count = 0;
    count = count + 1;
    printf("Count", count);
}

int main()
{
    counter();
    counter();
    counter();
}
```

Keywords - void

- The **void** keyword indicates that a function doesn't return any value.
- The void keyword can also be passed as an argument to a function, indicating that this function has no parameters

```
/**@brief Function for putting the chip into sleep mode.
 *
 * @note This function will not return.
 */
static void sleep_mode_enter(void)
{
    ret_code_t err_code;

    /* Go to system-off mode (this function will not return;
    wakeup will cause a reset). */
    err_code = sd_power_system_off();
    APP_ERROR_CHECK(err_code);
}
```

Macros and Conditional directives

- A macro is a fragment of code which has been given a name. The name will be replaced by the code fragment.
- Two types of macros
  - Object-like macros
  - Function-like macros
- Conditional directives are used to decide which code chunks that should be compiled.

```
#define DEVICE_NAME         "Nordic_Template"

#define APP_ADV_INTERVAL    300

/**@brief Macro for converting milliseconds to ticks.
 *
 * @param[in] TIME          Number of milliseconds to convert.
 * @param[in] RESOLUTION    Unit to be converted to in [us/ticks].
 */
#define MSEC_TO_UNITS(TIME, RESOLUTION) (((TIME) * 1000) / (RESOLUTION))

#define MIN_CONN_INTERVAL    MSEC_TO_UNITS(100, UNIT_1_25_MS)

#ifdef SOFTDEVICE_PRESENT
#include "nrf_sdh.h"
#include "nrf_sdh_soc.h"
#endif
```