Lecture 4 – Storage and I/O

CSE 456: Embedded Systems

Storage

Memory Map

Example: Memory Map in MSP432

Available memory:

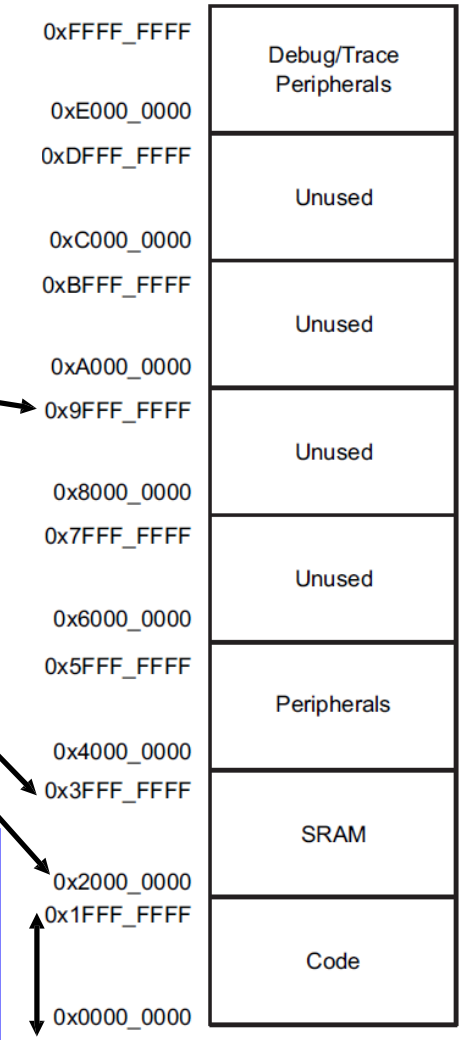
- The processor used in the lab (MSP432P401R) has built in 256kB flash memory, 64kB SRAM and 32kB ROM (Read Only Memory).

Address space:

- The processor uses 32 bit addresses. Therefore, the addressable memory space is 4 GByte (2^{32} Byte) as each memory location corresponds to 1 Byte.
- The address space is used to address the memories (reading and writing), to address the peripheral units, and to have access to debug and trace information (memory mapped microarchitecture).
- The address space is partitioned into zones, each one with a dedicated use. The following is a simplified description to introduce the basic concepts.

Example: Memory Map in MSP432

Memory map:



hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit

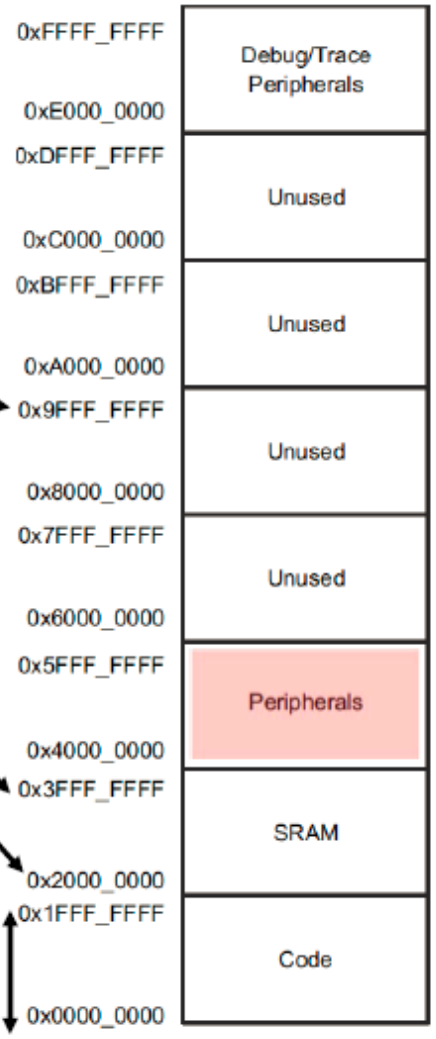
0011 1111 …. 1111
0010 0000 …. 0000

diff. = 0001 1111 …. 1111 ->
2^{29} different addresses
capacity = 2^{29} Byte =
512 MByte

Example: Memory Map in MSP432

Memory map:

hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit



0011 1111 1111
0010 0000 0000

diff. = 0001 1111 .... 1111 → 2^{29} different addresses
capacity = 2^{29} Byte = 512 MByte

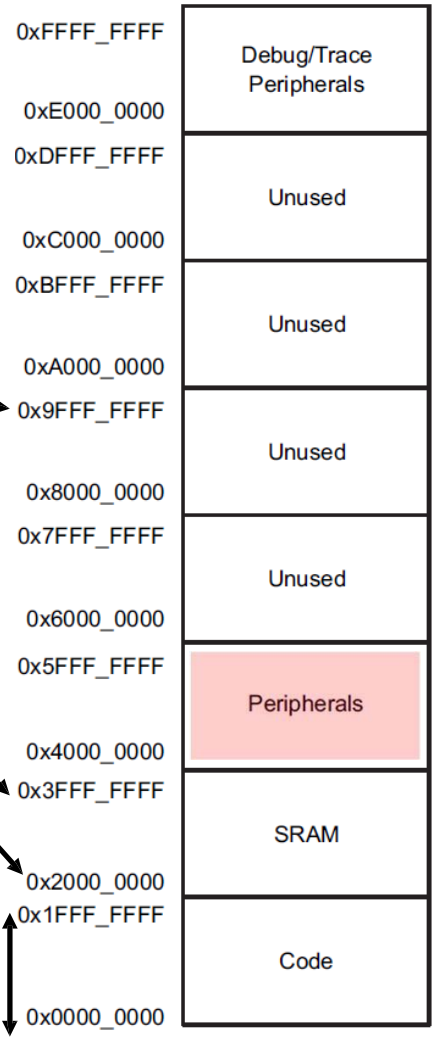
ADDRESS RANGE	PERIPHERAL
0x4000_0000 to 0x4000_03FF	Timer_A0
0x4000_0400 to 0x4000_07FF	Timer_A1
0x4000_0800 to 0x4000_0BFF	Timer_A2
0x4000_0C00 to 0x4000_0FFF	Timer_A3
0x4000_1000 to 0x4000_13FF	eUSCI_A0
0x4000_1400 to 0x4000_17FF	eUSCI_A1
0x4000_1800 to 0x4000_1BFF	eUSCI_A2
0x4000_1C00 to 0x4000_1FFF	eUSCI_A3
•••	
0x4000_4400 to 0x4000_47FF	RTC_C
0x4000_4800 to 0x4000_4BFF	WDT_A
0x4000_4C00 to 0x4000_4FFF	Port Module
•••	

Table 6-21. Port Registers (Base Address: 0x4000_4C00)

REGISTER NAME	ACRONYM	OFFSET from base address
Port 1 Input	P1IN	000h
Port 2 Input	P2IN	001h
Port 1 Output	P1OUT	002h
Port 2 Output	P2OUT	003h

Example: Memory Map in MSP432

Memory map:



hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit

0011 1111 …. 1111
0010 0000 …. 0000

diff. = 0001 1111 …. 1111 □

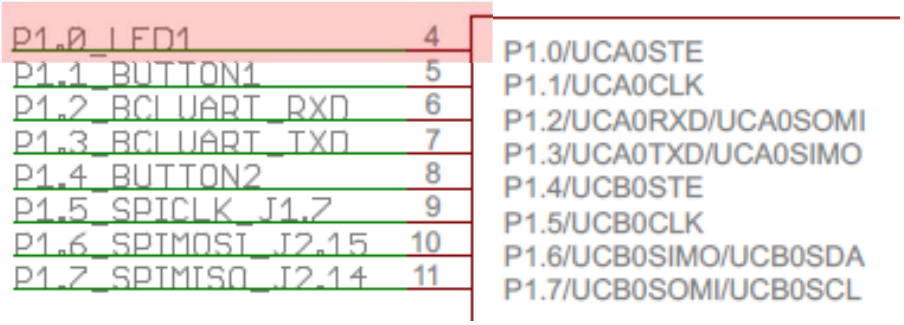
2^{29} different addresses

capacity = 2^{29} Byte = 512 MByte

Table 6-21. Port Registers (Base Address: 0x4000_4C00)

REGISTER NAME	ACRONYM	OFFSET
Port 1 Input	P1IN	000h
Port 2 Input	P2IN	001h
Port 1 Output	P1OUT	002h
Port 2 Output	P2OUT	003h

Schematic of LaunchPad as used in the Lab:

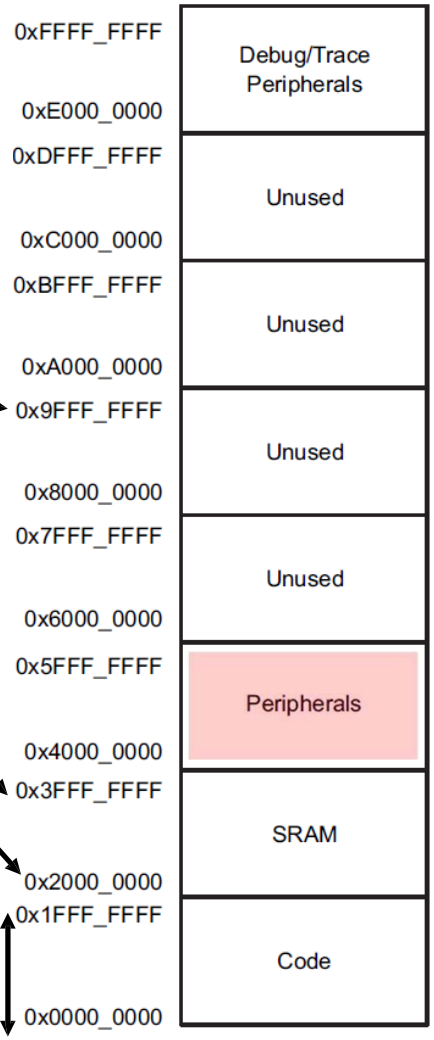


LED1 is connected to Port 1, Pin 0

How do we toggle LED1 in a C program?

Example: Memory Map in MSP432

Memory map:



hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit

0011 1111 …. 1111
0010 0000 …. 0000

diff. = 0001 1111 …. 1111 □

2^{29} different addresses

capacity = 2^{29} Byte = 512 MByte

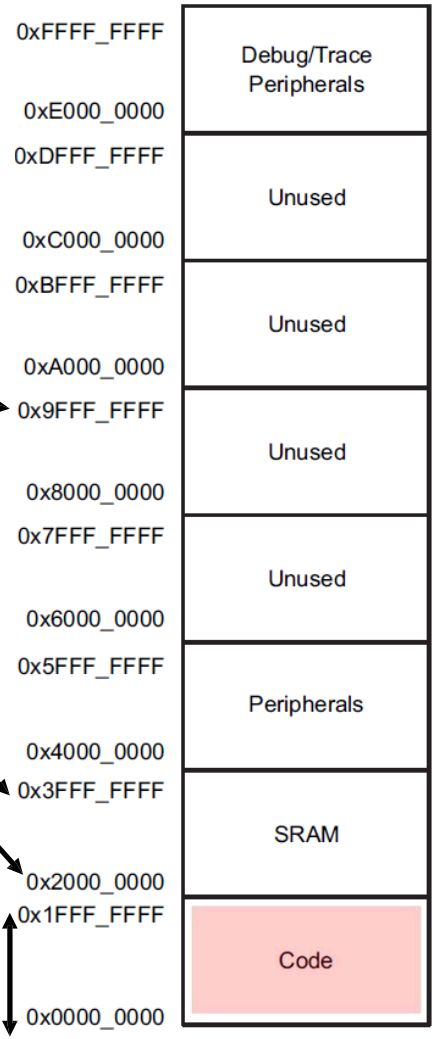
Many necessary elements are missing in the sketch below, in particular the configuration of the port (input or output, pull up or pull down resistors for input, drive strength for output). See lab session.

```
…
//declare p1out as a pointer to an 8Bit integer
volatile uint8_t*  p1out;

//P1OUT should point to Port 1 where LED1 is connected
p1out = (uint8_t*) 0x40004C02;

//Toggle Bit 0 (Signal to which LED1 is connected)
*p1out = *p1out ^ 0x01;
```

^ : XOR

Example: Memory Map in MSP432

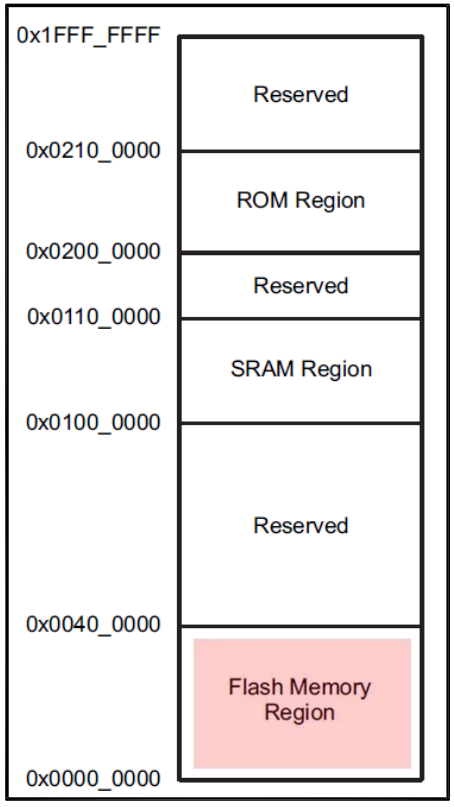
Memory map:

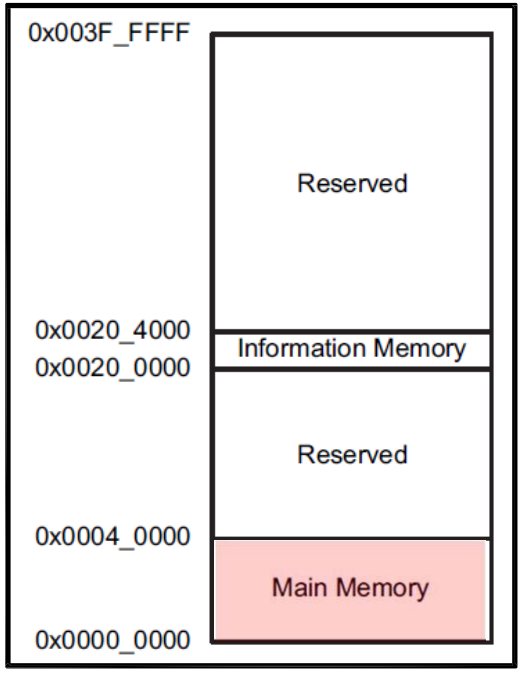


hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit

0011 1111 …. 1111
0010 0000 …. 0000

diff. = 0001 1111 …. 1111 □
2^{29} different addresses
capacity = 2^{29} Byte = 512 MByte

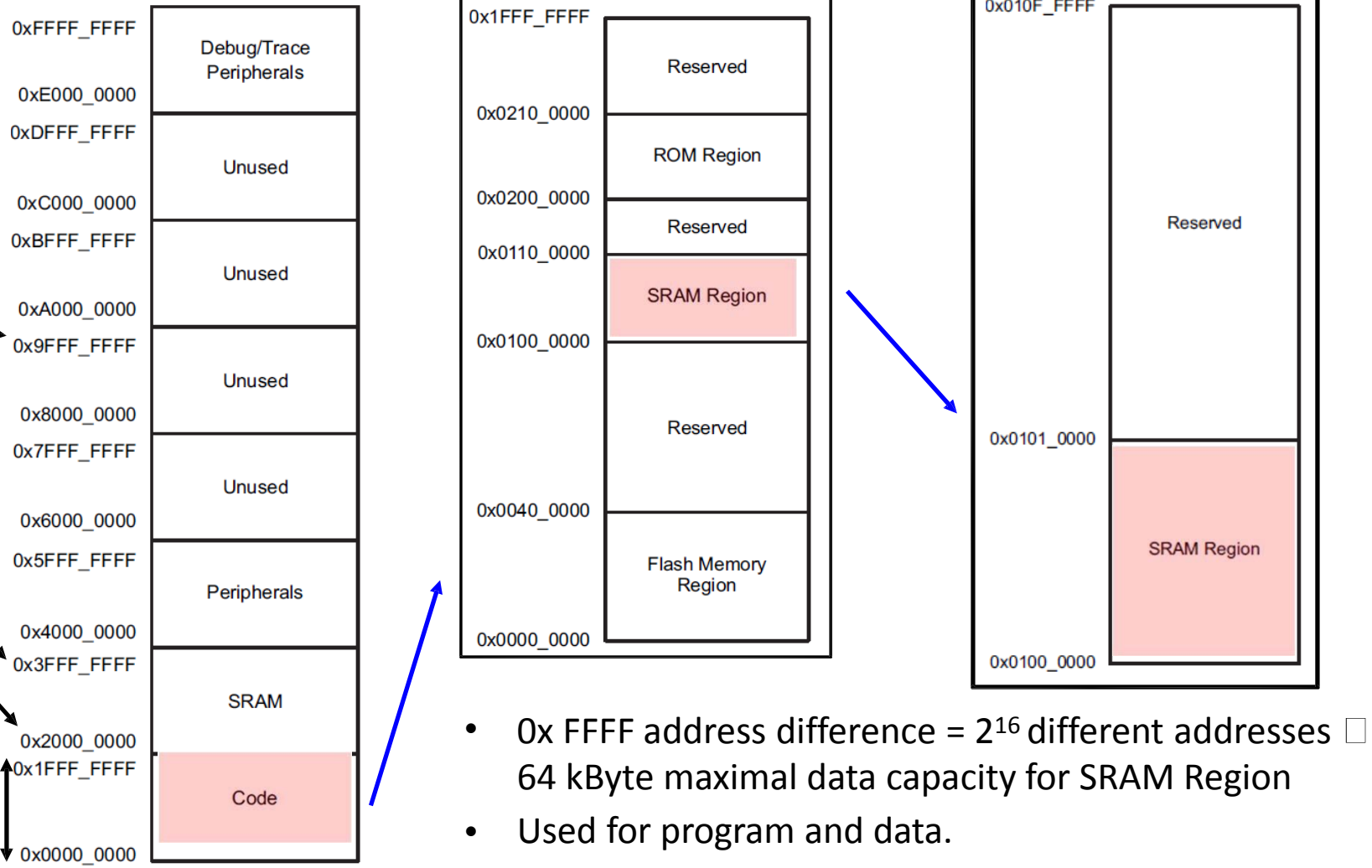




- 0x3FFFF address difference = 4 * 2^{16} different addresses □ 256 kByte maximal data capacity for Flash Main Memory
- Used for program, data and non-volatile configuration.

Example: Memory Map in MSP432

Memory map:



hexadecimal representation of a 32 bit binary number; each digit corresponds to 4 bit

0011 1111 …. 1111
0010 0000 …. 0000

diff. = 0001 1111 …. 1111 □
2^{29} different addresses
capacity = 2^{29} Byte = 512 MByte

- 0x FFFF address difference = 2^{16} different addresses □
  64 kByte maximal data capacity for SRAM Region
- Used for program and data.

Example: Memory Map in MSP432

1. How many addresses are available for peripherals

2. Suppose a switch (input) is connected to Port 1, Pin 1, determine the memory address of Port 1, Pin 1.

Example: Memory Map in MSP432

1. How many addresses are available for peripherals

Solution: Peripherals addresses have the following range [0x4000_0000, 0x5FFF_FFFF].

Total addressable locations = 0x5FFF_FFFF - 0x4000_0000 + 1
= 0x2000_0000 or 2^29

2. Suppose a switch (input) is connected to Port 1, Pin 1, determine the memory address of Port 1, Pin 1.

Solution: Port 1 Pin 1 address = 0x4000_4C00 + 0x0000 = 0x4000_4C00

Input and Output

Device Communication

Very often, a processor needs to *exchange information with other processors* or devices. To satisfy various needs, there exists many different *communication protocols*, such as

- ***UART*** (Universal Asynchronous Receiver-Transmitter)
- ***SPI*** (Serial Peripheral Interface Bus)
- ***I2C*** (Inter-Integrated Circuit)
- ***USB*** (Universal Serial Bus)

- As the principles are similar, we will just explain a representative of an asynchronous protocol (*UART*, no shared clock signal between sender and receiver) and one of a synchronous protocol (*SPI* , shared clock signal).

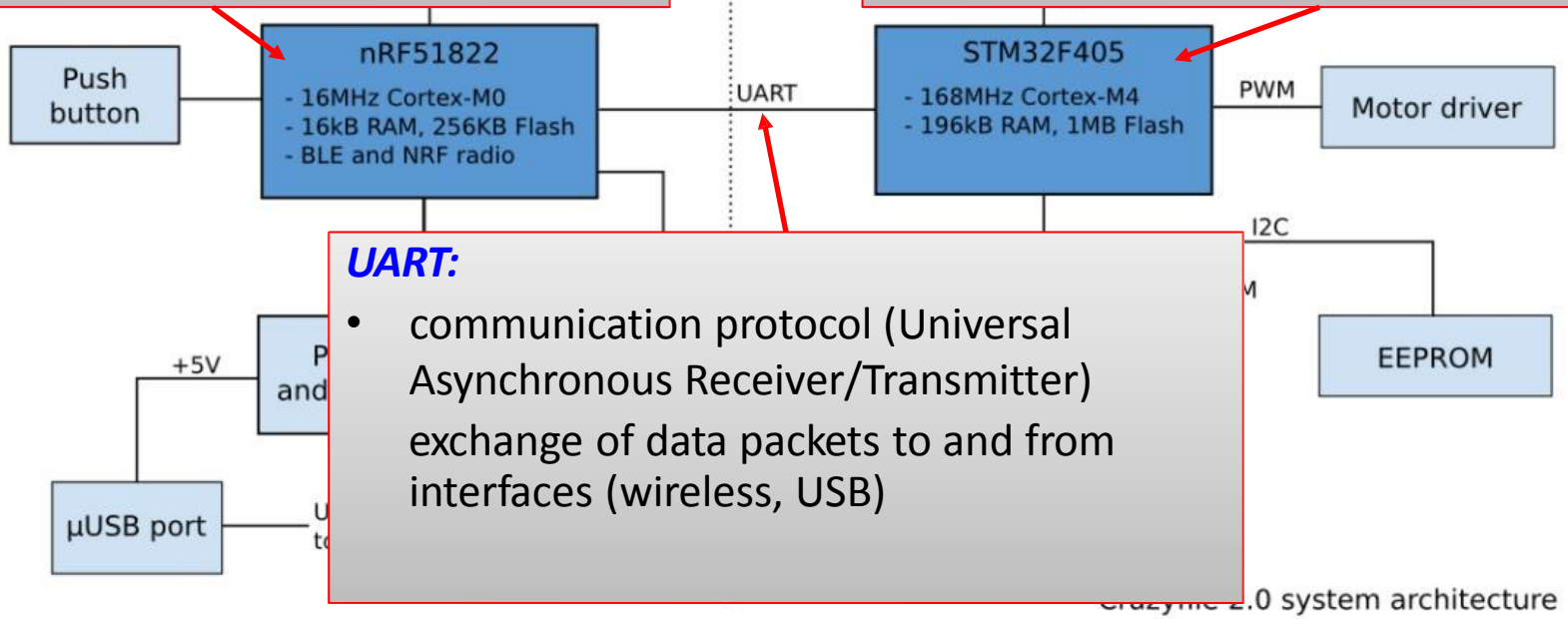
Remember?

low power CPU

- enabling power to the rest of the system
- battery charging and voltage measurement
- wireless radio (boot and operate)
- detect and check expansion boards

higher performance CPU

- sensor reading and motor control
- flight control
- telemetry (including the battery voltage)
- additional user development
- USB connection



UART:

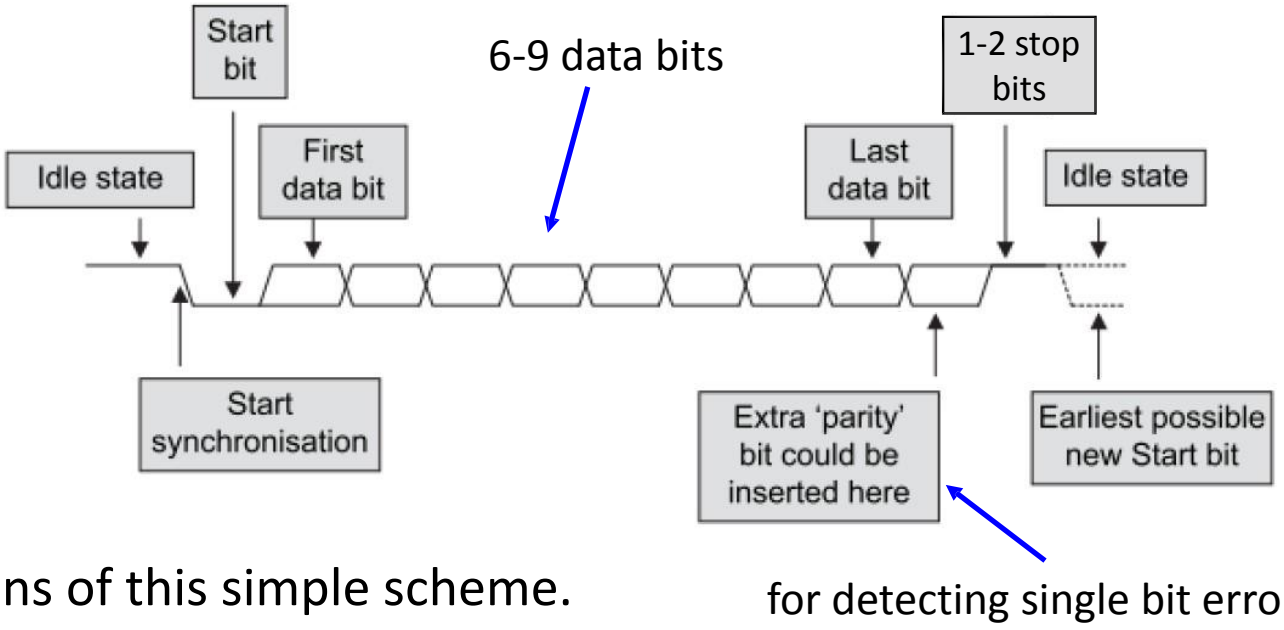
- communication protocol (Universal Asynchronous Receiver/Transmitter)

  exchange of data packets to and from interfaces (wireless, USB)

Crazyflie 2.0 system architecture

UART

- *Serial communication* of bits via a single signal, i.e. UART provides parallel-to-serial and serial-to-parallel conversion.
- Sender and receiver need to *agree on the transmission rate*.
- Transmission of a serial packet starts with a start bit, followed by data bits and finalized using a stop bit:

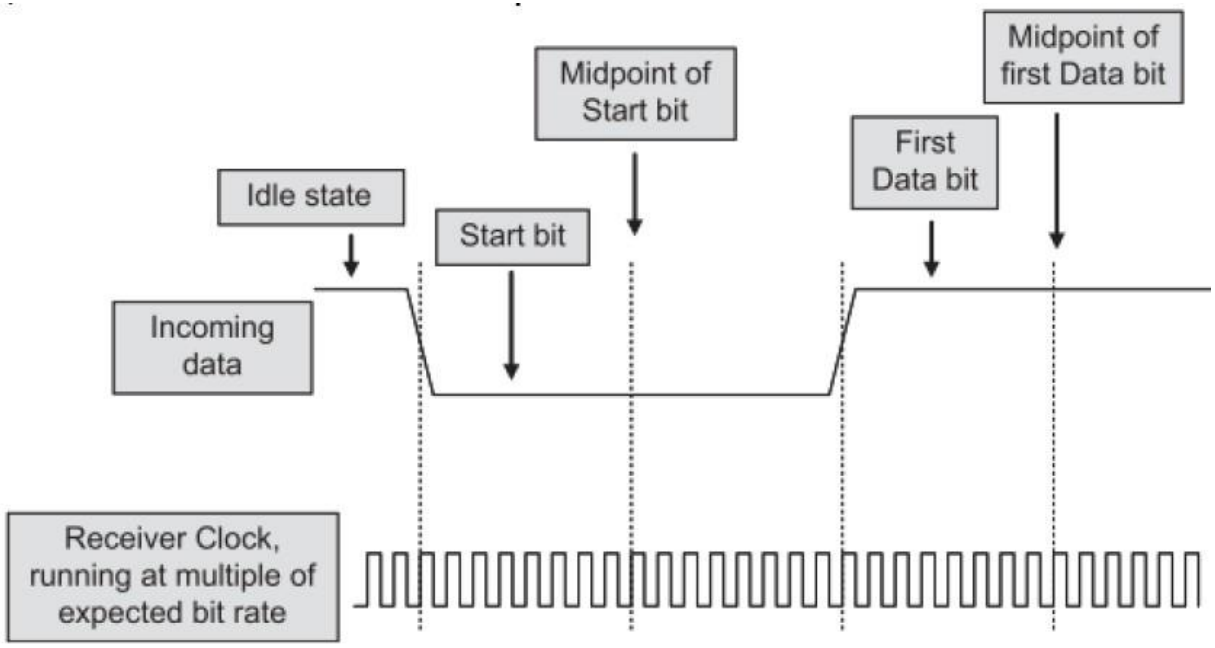


- There exist many variations of this simple scheme.

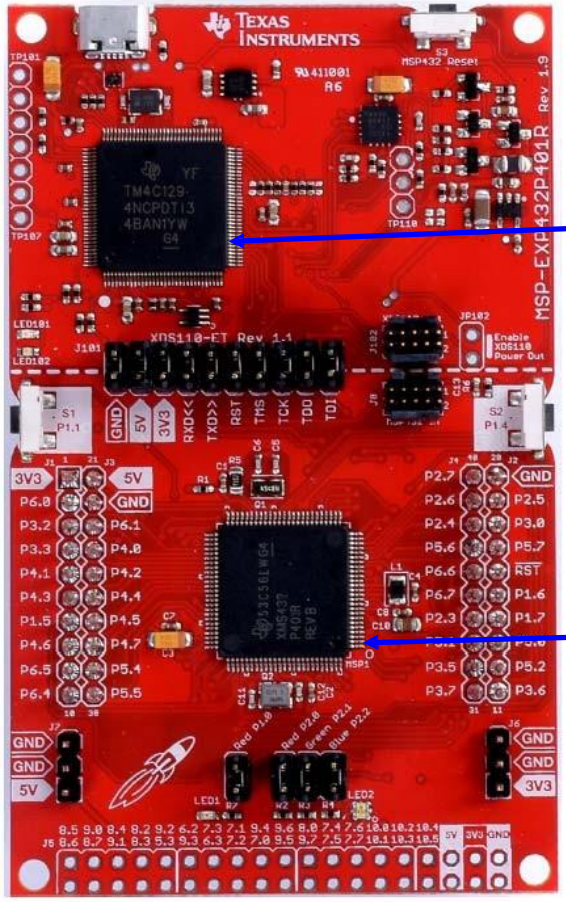
for detecting single bit errors

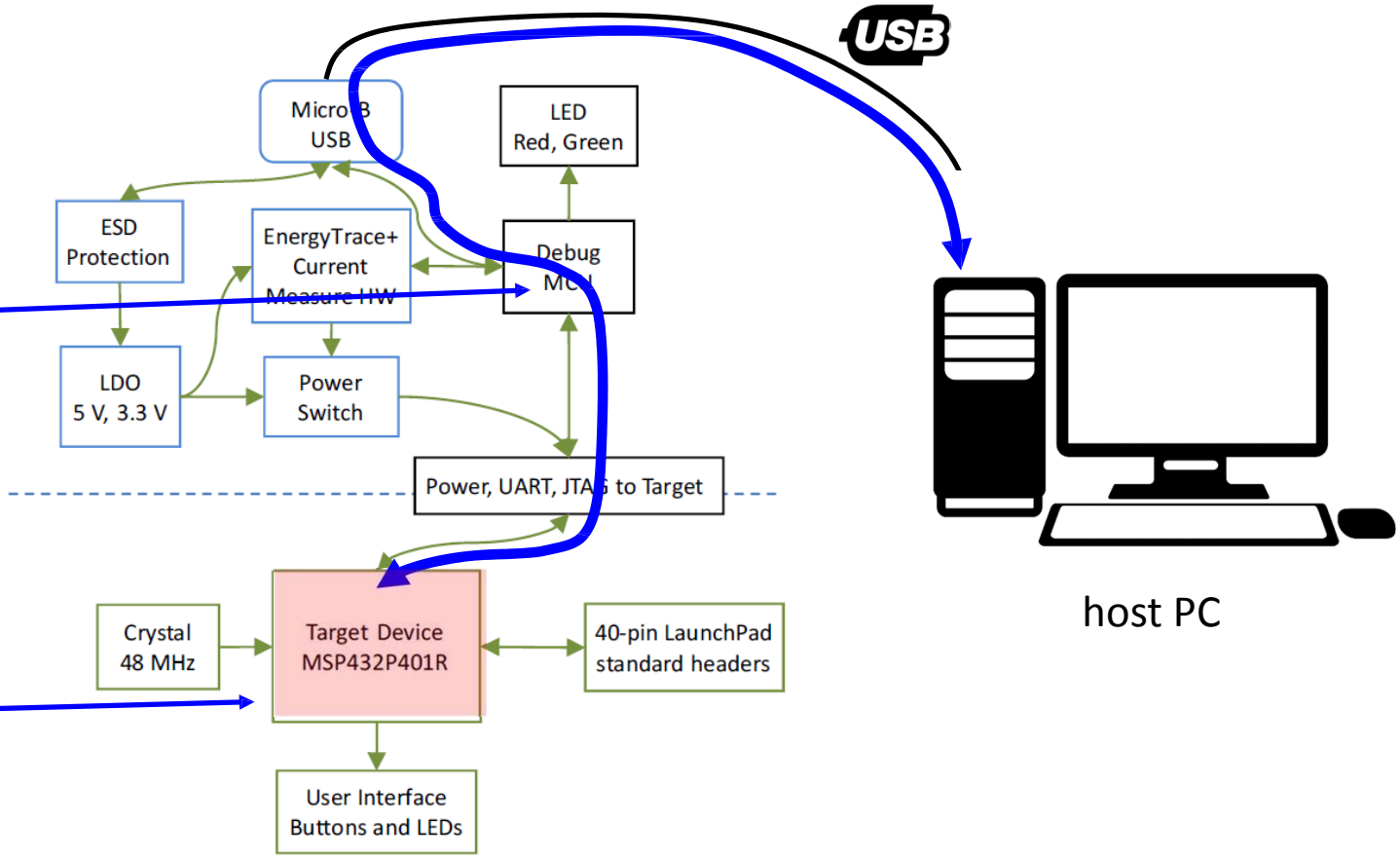
UART

- The receiver runs an *internal clock* whose frequency is an exact multiple of the expected bit rate.
- When a *Start bit* is detected, a counter begins to count clock cycles e.g. 8 cycles until the midpoint of the anticipated Start bit is reached.
- The clock counter counts a further 16 cycles, to the middle of the first *Data bit*, and so on until the *Stop bit*.

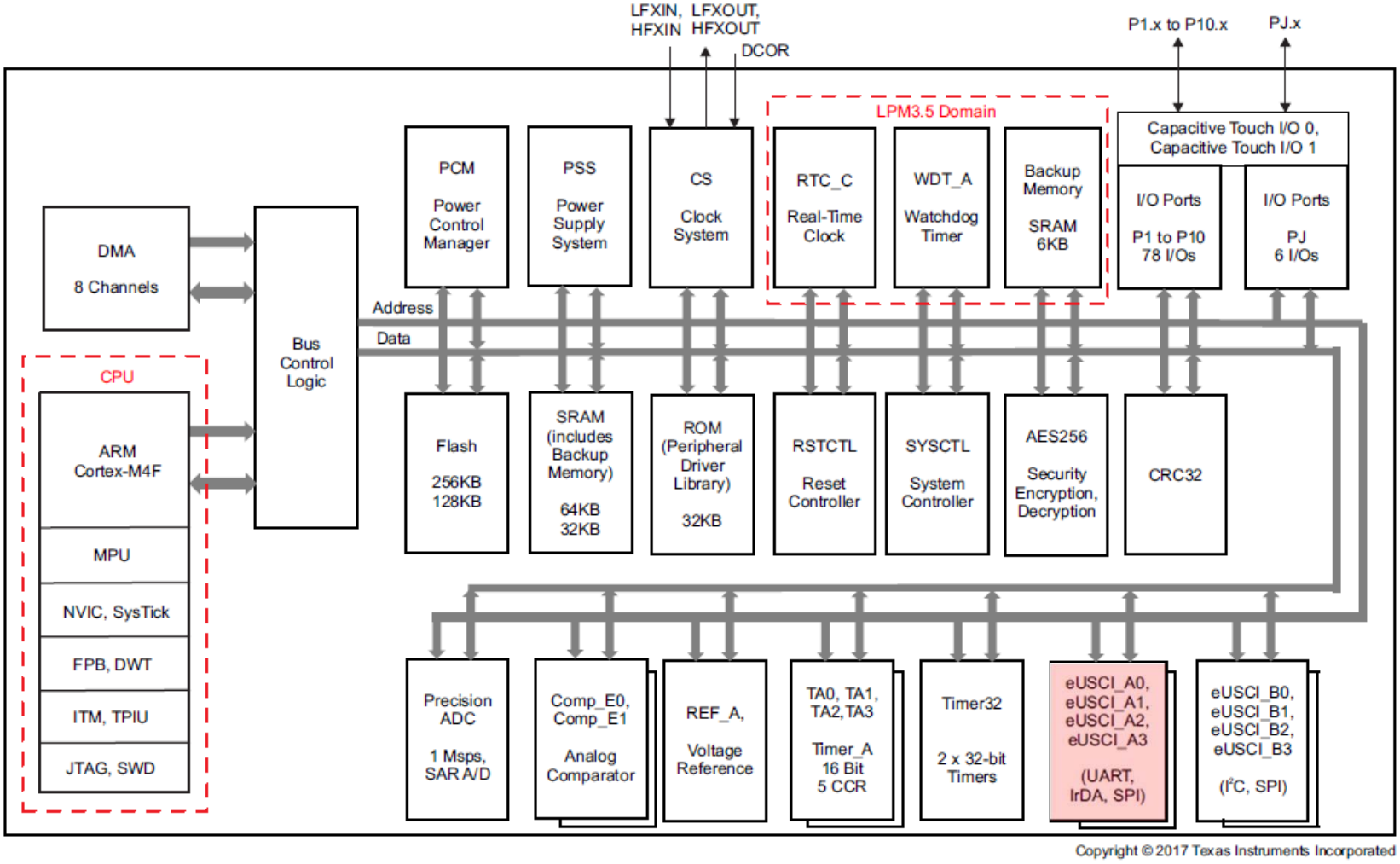


UART with MSP432





UART with MSP432



Memory-Mapped Device Access

eUSCI_A0 Registers (Base Address: 0x4000_1000)

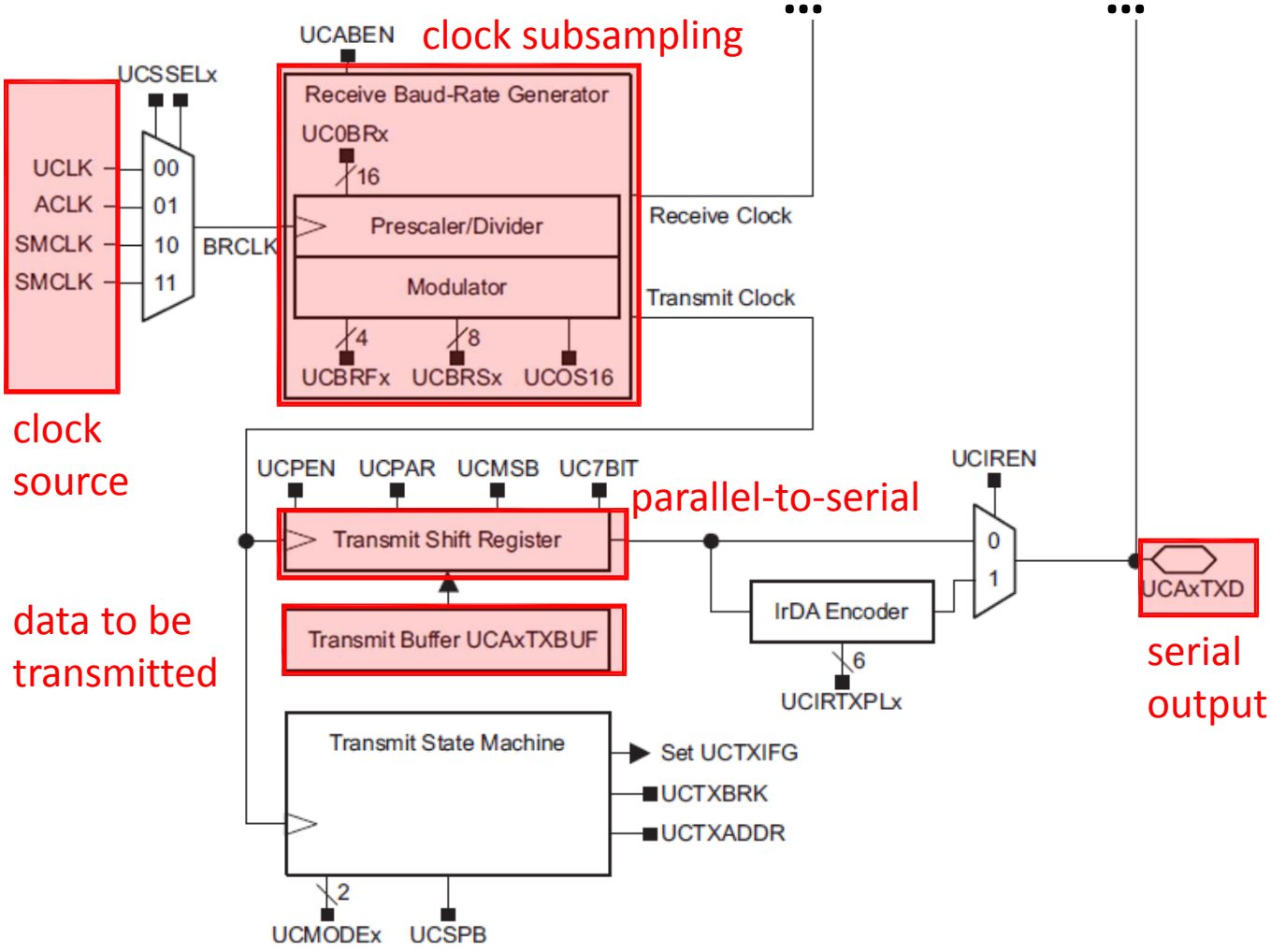
REGISTER NAME	OFFSET
eUSCI_A0 Control Word 0	00h
eUSCI_A0 Control Word 1	02h
eUSCI_A0 Baud Rate Control	06h
eUSCI_A0 Modulation Control	08h
eUSCI_A0 Status	0Ah
eUSCI_A0 Receive Buffer	0Ch
eUSCI_A0 Transmit Buffer	0Eh
eUSCI_A0 Auto Baud Rate Control	10h
eUSCI_A0 IrDA Control	12h
eUSCI_A0 Interrupt Enable	1Ah
eUSCI_A0 Interrupt Flag	1Ch
eUSCI_A0 Interrupt Vector	1Eh

- *Configuration of Transmitter and Receiver must match*; otherwise, they can not communicate.
- Examples of configuration parameters:
  - transmission rate (baud rate, i.e., symbols/s)
  - LSB or MSB first
  - number of bits per packet
  - parity bit
  - number of stop bits
  - interrupt-based communication
  - clock source

in our case: bit/s

buffer for received bits and bits that should be transmitted

Transmission Rate



Clock subsampling:

- The clock subsampling block is complex, as one tries to match a large set of transmission rates with a fixed input frequency.

Clock Source:

- SMCLK in the lab setup = 3MHz
- Quartz frequency = 48 MHz, is divided by 16 before connected to SMCLK

Example:

- Transmission rate 4800 bit/s
- 16 clock periods per bit (see 3-26)
- Subsampling factor = 3*10^6 / (4.8*10^3 * 16) = 39.0625

Software Interface

Part of C program that *prints a character to a UART* terminal on the host PC:

```
...
static const eUSCI_UART_Config uartConfig =
{
  EUSCI_A_UART_CLOCKSOURCE_SMCLK,                 // SMCLK Clock Source
  39,                                             // BRDIV  = 39 , integral part
  1,                                              // UCxBRF =  1 , fractional part * 16
  0, EUSCI_A_UART_NO_PARITY,                      // UCxBRS =  0
  EUSCI_A_UART_LSB_FIRST,                         // No Parity
  EUSCI_A_UART_ONE_STOP_BIT,                      // LSB First
  EUSCI_A_UART_MODE,                              // One stop bit
                                                  // UART mode
  EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION}; // Oversampling Mode
GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P1,
          GPIO_PIN2 | GPIO_PIN3, GPIO_PRIMARY_MODULE_FUNCTION );      //Configure CPU signals
UART_initModule(EUSCI_A0_BASE, &uartConfig);              // Configuring UART Module A0
UART_enableModule(EUSCI_A0_BASE);                         // Enable UART module A0

UART_transmitData(EUSCI_A0_BASE,'a');                     // Write character 'a' to UART

...
```

data structure uartConfig contains the configuration of the UART

use uartConfig to write to eUSCI_A0 configuration registers

start UART

base address of A0 (0x40001000), where A0 is the instance of the UART peripheral

Software Interface

Replacing UART_transmitData(EUSCI_A0_BASE,'a') by a *direct access to registers*:

```
...
volatile uint16_t*  uca0ifg = (uint16_t*) 0x4000101C;
volatile uint16_t*  uca0txbuf = (uint16_t*) 0x4000100E;
...
// Initialization of UART as before
...
while (!((*uca0ifg >> 1) & 0x0001));
*uca0txbuf = (char) 'g'; // Write to transmit buffer
...
```

declare pointers to UART configuration registers

wait until transmit buffer is empty

write character ‘g’ to the transmit buffer

Table 22-18. UCAxIFG Register Description

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCAxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending

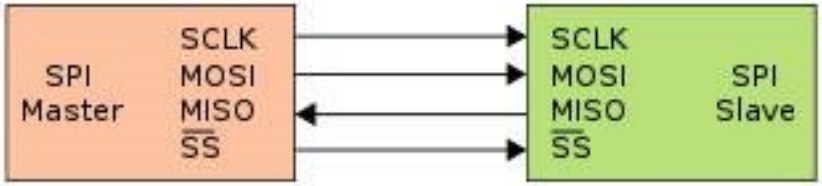
shift 1 bit to the right

```
!((*uca0ifg >> 1) & 0x0001)
```

expression is ‘1’ if bit UCTXIFG = 0 (buffer not empty).

SPI (Serial Peripheral Interface Bus)

- Typically *communicate across short distances*
- *Characteristics:*
  - 4-wire synchronized (clocked) communications bus
  - supports single master and multiple slaves
  - always full-duplex: Communicates in both directions simultaneously
  - multiple Mbps transmission speeds can be achieved
  - transfer data in 4 to 16 bit serial packets

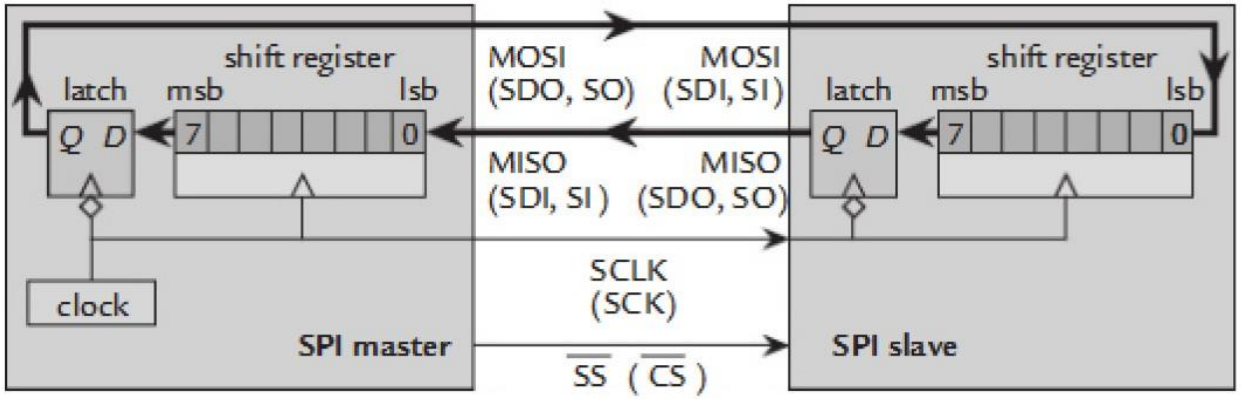


- *Bus wiring:*
  - MOSI (Master Out Slave In) – carries data out of master to slave MISO
  - (Master In Slave Out) – carries data out of slave to master Both MOSI
  - and MISO are active during every transmission
  - $\overline{SS}$ (or CS) – signal to select each slave chip
  - System clock SCLK – produced by master to synchronize transfers

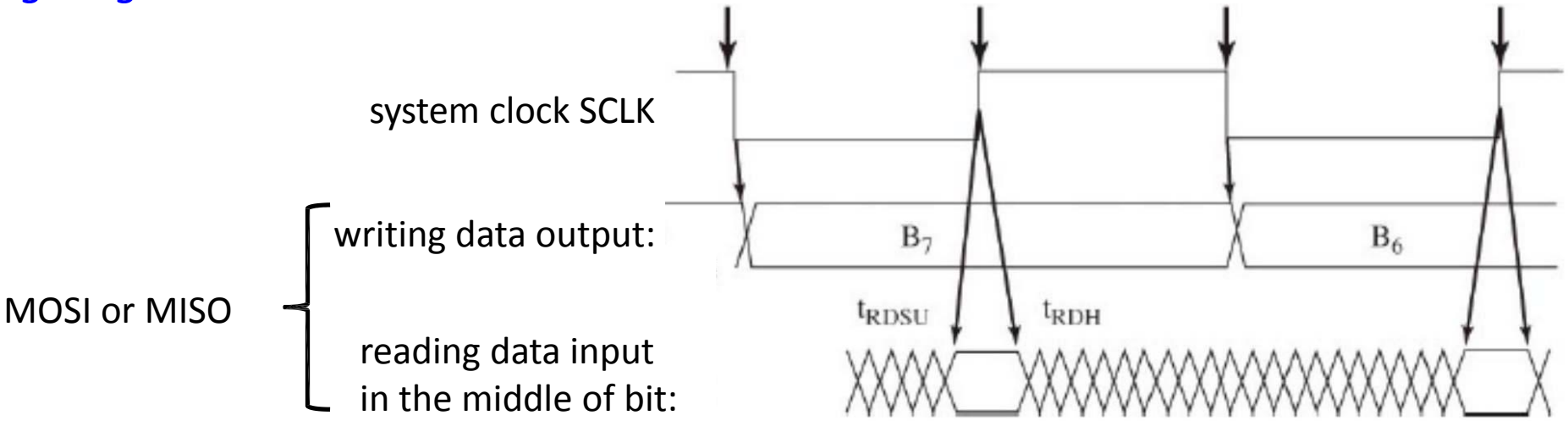
SPI (Serial Peripheral Interface Bus)

More detailed circuit diagram:

- details vary between different vendors and implementations

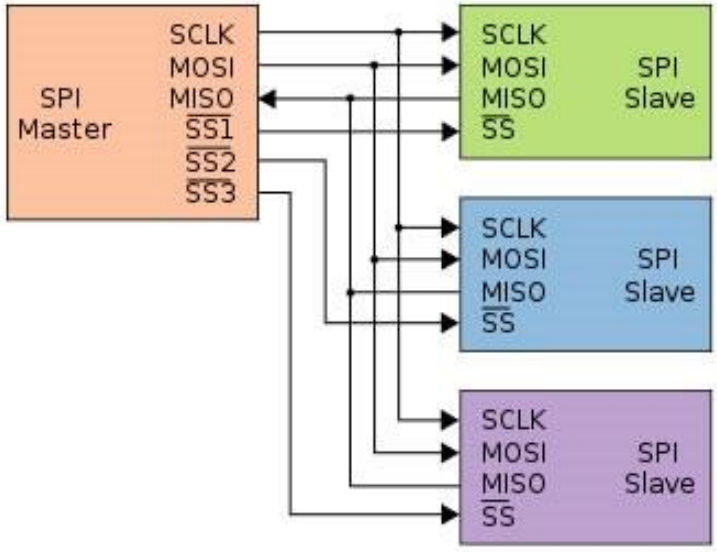


Timing diagram:

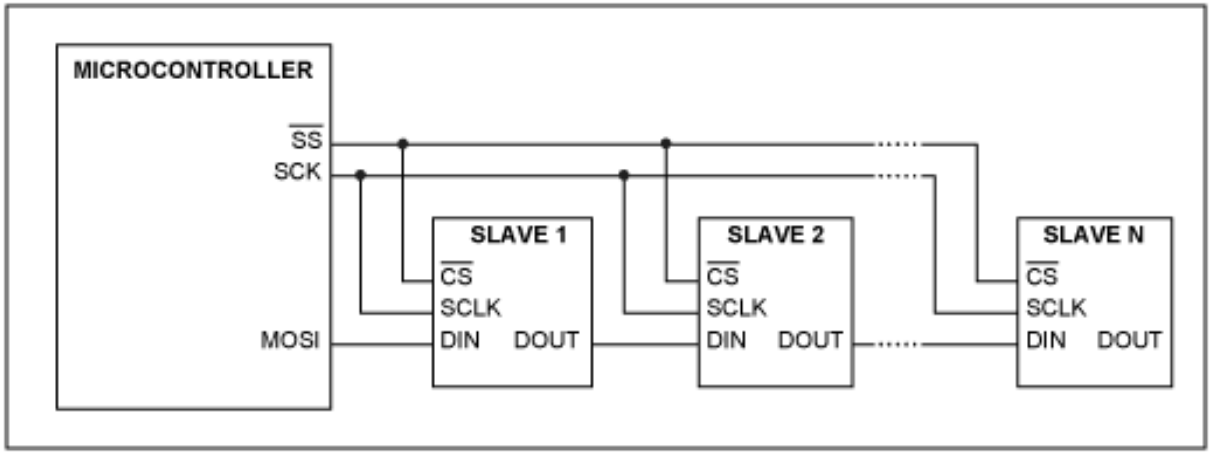


SPI (Serial Peripheral Interface Bus)

Two examples of bus configurations:



Master and multiple independent slaves



Master and multiple daisy-chained slaves