CC2511 Week 3: Lecture 2

Software: More on the C language

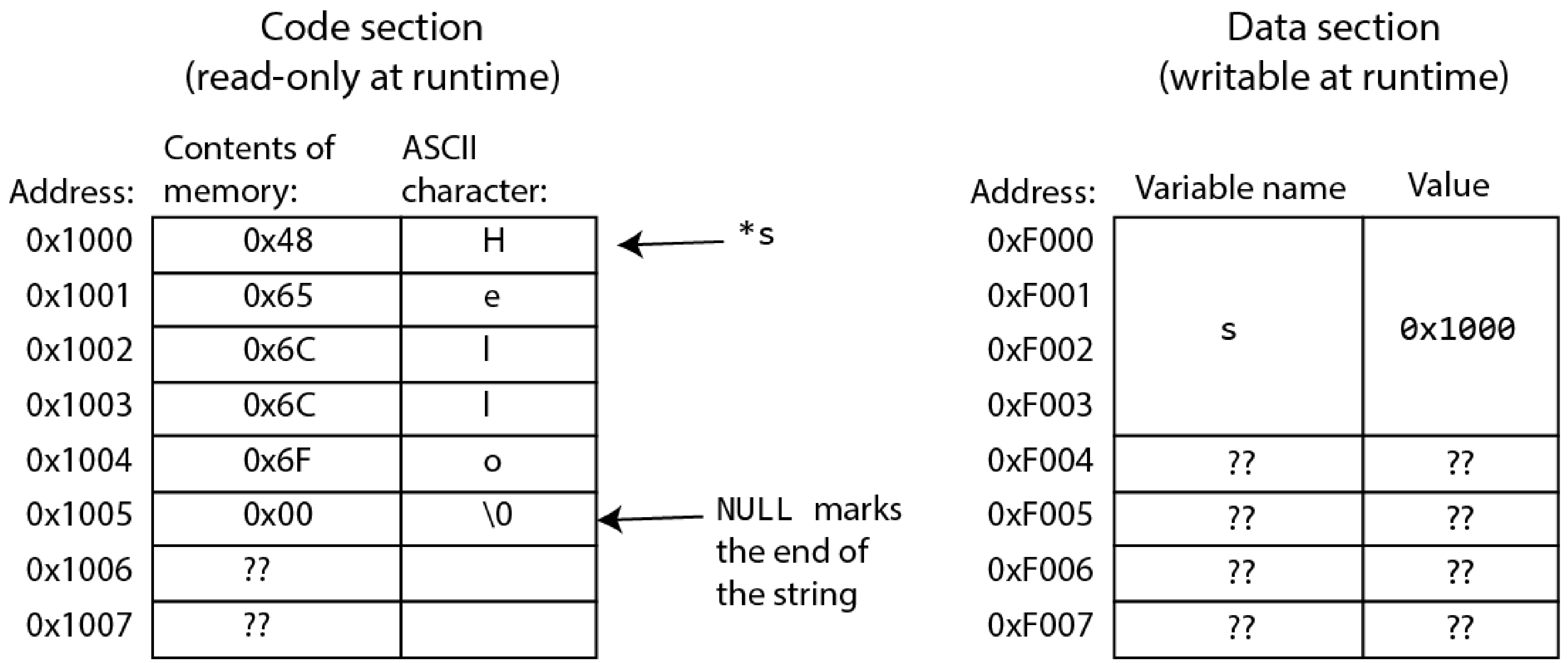
Hardware: Overview of the Kinetis K20/K22 clocks

Strings in C

- C strings are just arrays of chars.
- They have all the weaknesses of C’s arrays.

Strings

```c
char *s = "Hello";
```



Code section
(read-only at runtime)

Address:	Contents of memory:	ASCII character:
0x1000	0x48	H
0x1001	0x65	e
0x1002	0x6C	l
0x1003	0x6C	l
0x1004	0x6F	o
0x1005	0x00	\0
0x1006	??	
0x1007	??	

Data section
(writable at runtime)

Address:	Variable name	Value
0xF000	s	0x1000
0xF001		
0xF002		
0xF003		
0xF004	??	??
0xF005	??	??
0xF006	??	??
0xF007	??	??

Strings

- **Strings are arrays of characters**.

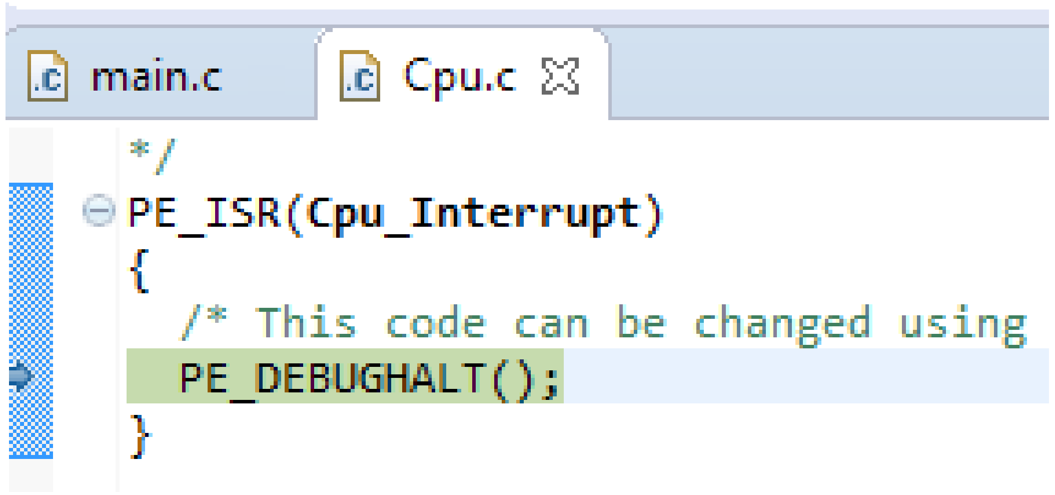
```
char *str = "Hello world";
```

- The string `"Hello world"` is placed in a fixed location in memory (as part of the program itself, i.e. alongside the code)
- A variable `str` is allocated to point to the first character in the string.
- The string continues until a NULL is reached (byte of 0, written as ‘\0’).

String constants are read-only

```
char *str = "Hello world";
str[0] = ‘h’; // Access violation; will crash here.
```

- **String constants** are strings given directly in the source code.
- They are allocated in program memory, which is read-only on our machines.



```
*/
PE_ISR(Cpu_Interrupt)
{
  /* This code can be changed using
  PE_DEBUGHALT();
}
```

Mutable strings

```
char s1 [10];
s1[0] = ‘H’; // OK, s1 is a mutable array
s1[1] = ‘i’;
s1[2] = ‘\0’; // always end strings in a NULL

// Shorthand:
char s2 [10] = “Hi”;
// This syntax is only valid at initialisation!
```

Assigning strings

```
char s1 [10] = “Hi”; // OK. Setting an initial value
s1 = “Hello”; // Invalid! Compile error here.

// Correct syntax:
strncpy(s1, “Hello”, 10); // string copy function.
```

- The 10 is the number of bytes in the destination array.

Characters vs strings

```
char c = ‘Q’; // single quotes = a char
char *s = “Q”; // double quotes = string
```

- Double quotes have an implied NULL.
- Double quotes generate a pointer that points to the first character in the string.

Comparison with strings

Consider:

```
if (c == ‘Q’) { ... }
```

Vs

```
if (c == “Q”) { ... }
```

- The first checks if the char variable c contains Q.
- **The second checks if the char variable c is equal to the memory address where the compiler placed the “Q” string!**

Comparison with strings

- You cannot do string comparisons with “==“ or “!=“ in C!
  - Such an operator would compare the pointer addresses, not the contents of the string.
- To compare strings, use the function **strcmp** (“string compare”)

```
if (0 == strcmp(s1, s2)) {
    // s1 and s2 are equal
}
```

How do you know the length of a string?

```
char *str = "Hello world";
```

- Access characters as `str[0]`, `str[1]`, …
- What happens if you try to read `str[100]`?
- Remember that `"Hello world"` is part of the program code.
- **If you read off the end of the array you'll get whatever other data the compiler happened to place there.**

Calculating the length of a string

- The strlen function (“string length”) counts the number of characters in the string **not including the trailing NULL**.

Summary: String handling

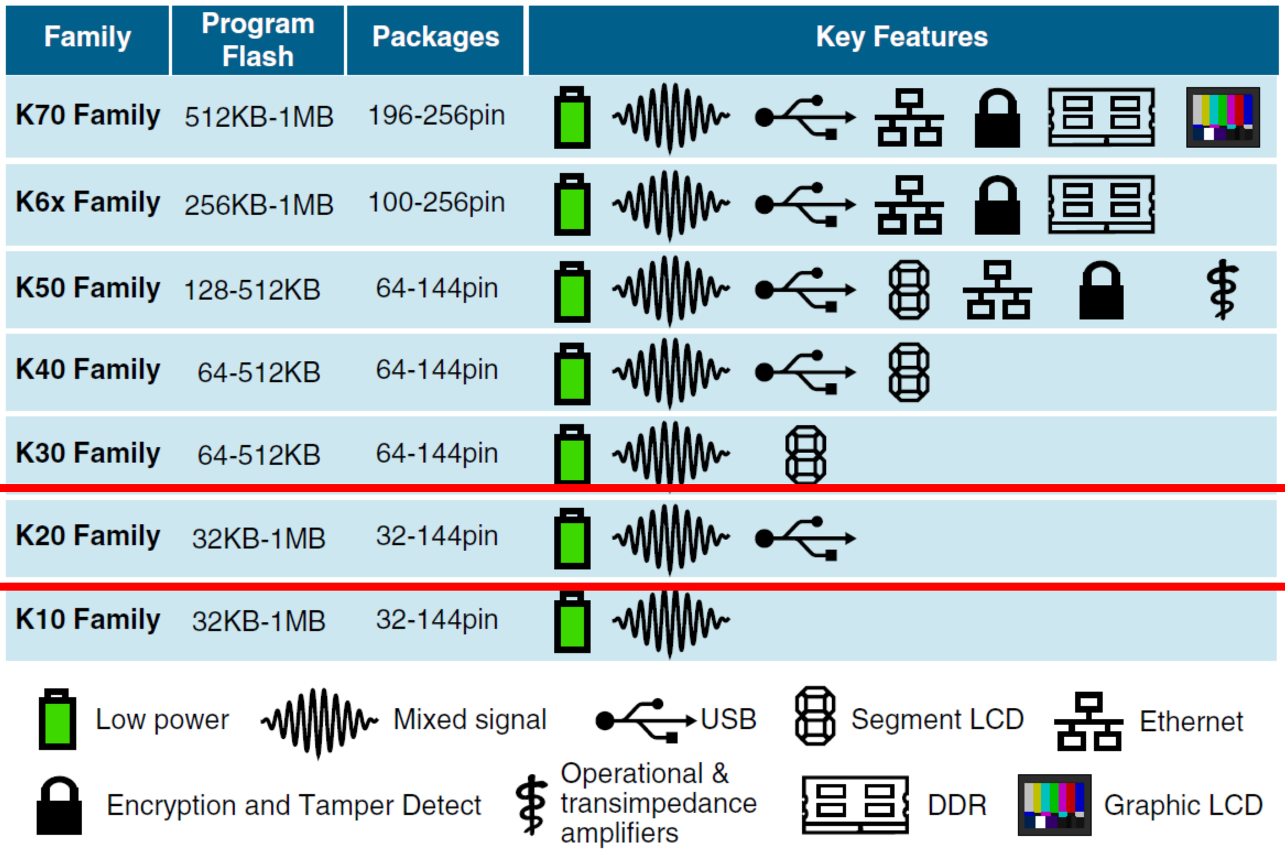
When manipulating strings:

- Make sure that there is always a trailing NULL at the end of the string.
- Make sure that you are aware of the size of the array so that you do not read/write into other parts of the program.
- Use C library functions like **strlen**, **strcmp**, **strncpy**.

C language syntax

- Continue on the C language reference.

Part 2: K20 family clock overview



Family	Program Flash	Packages	Key Features
K70 Family	512KB-1MB	196-256pin	🔋 〰 ⭠ 🖧 🔒 ▤ 🖵
K6x Family	256KB-1MB	100-256pin	🔋 〰 ⭠ 🖧 🔒 ▤
K50 Family	128-512KB	64-144pin	🔋 〰 ⭠ 8 🖧 🔒 ⚕
K40 Family	64-512KB	64-144pin	🔋 〰 ⭠ 8
K30 Family	64-512KB	64-144pin	🔋 〰 8
K20 Family	32KB-1MB	32-144pin	🔋 〰 ⭠
K10 Family	32KB-1MB	32-144pin	🔋 〰

🔋	Low power
〰	Mixed signal
⭠	USB
8	Segment LCD
🖧	Ethernet
🔒	Encryption and Tamper Detect
⚕	Operational & transimpedance amplifiers
▤	DDR
🖵	Graphic LCD

Figure 2-1. Kinetis MCU portfolio

Core

- The CPU core is an ARM Cortex-M4.
- The ARM instruction set is widespread in smartphones, tablets, etc.
- High performance, low power consumption.

System clocks

- Microcontrollers typically contain sophisticated apparatus for generating and controlling the clock speed of each part of the system.
- A faster clock speed produces higher performance at the expense of increased power draw.

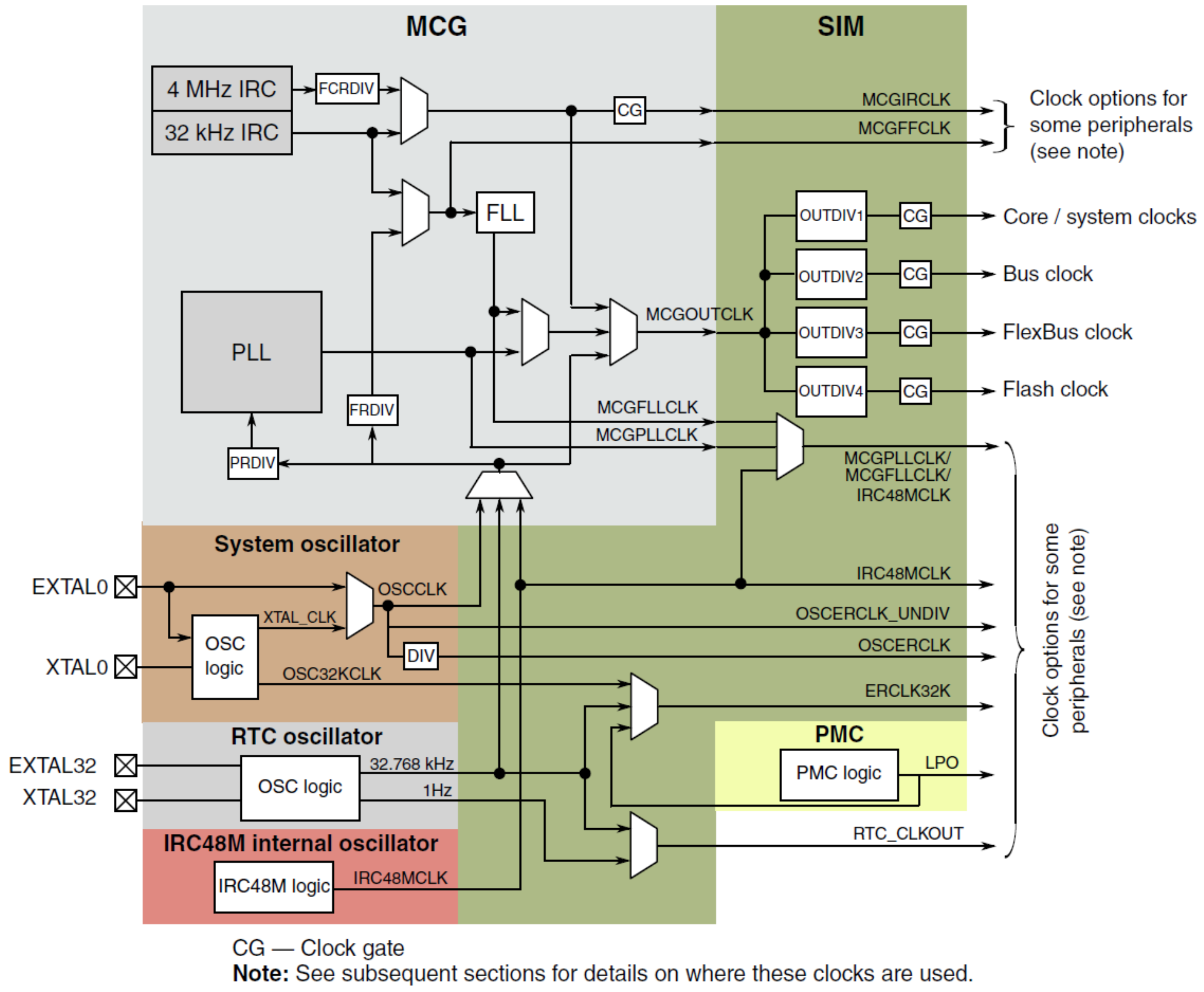


Figure 5-1. Clocking diagram

Image from the K22 reference manual. The K20 processor has the same architecture except it lacks the IRC48M internal oscillator.

Clock sources

- Two **internal reference clocks**.
- Simple. No external components required.
- Used until the software configures a different clock source.

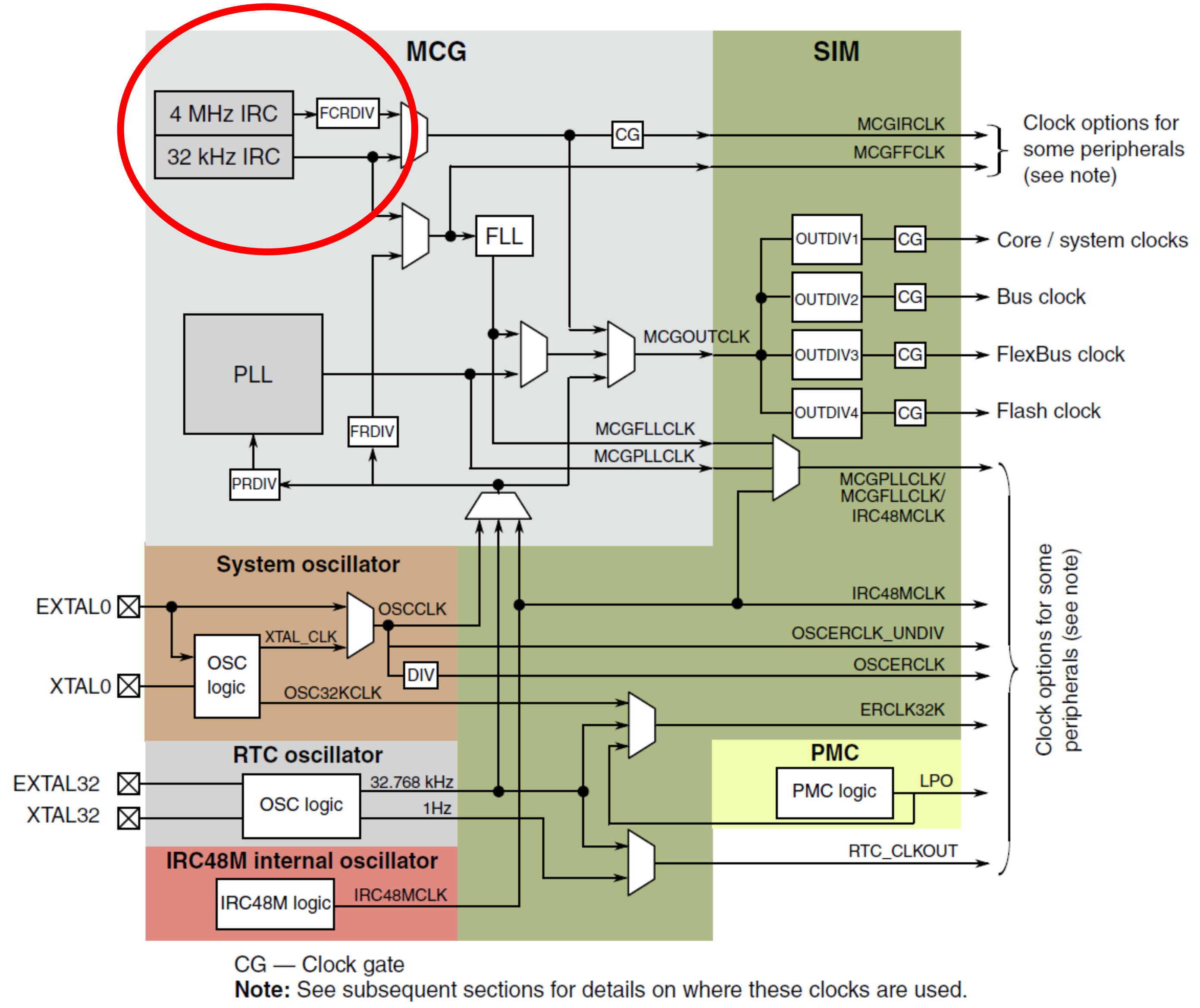


Figure 5-1. Clocking diagram

Clock sources

- **System oscillator** which uses a precise external crystal
- Generates an accurate high frequency clock
- Needed to get the system clock to full speed.

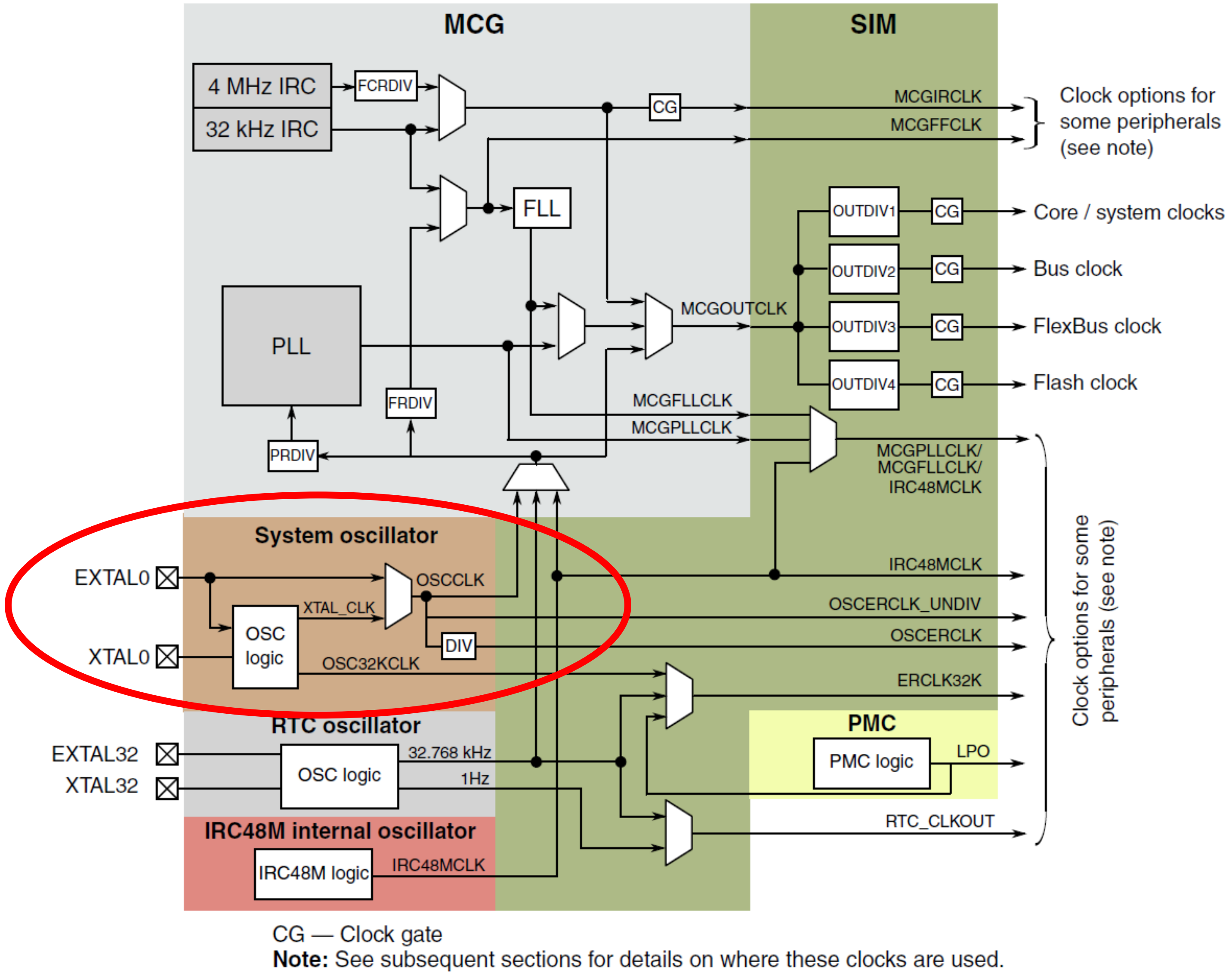


Figure 5-1. Clocking diagram

Clock sources

- The **Real Time Clock (RTC) oscillator** is a low-power clock intended to keep time even when the device is powered off.
- Requires an external battery. (Not populated on the FRDM board)

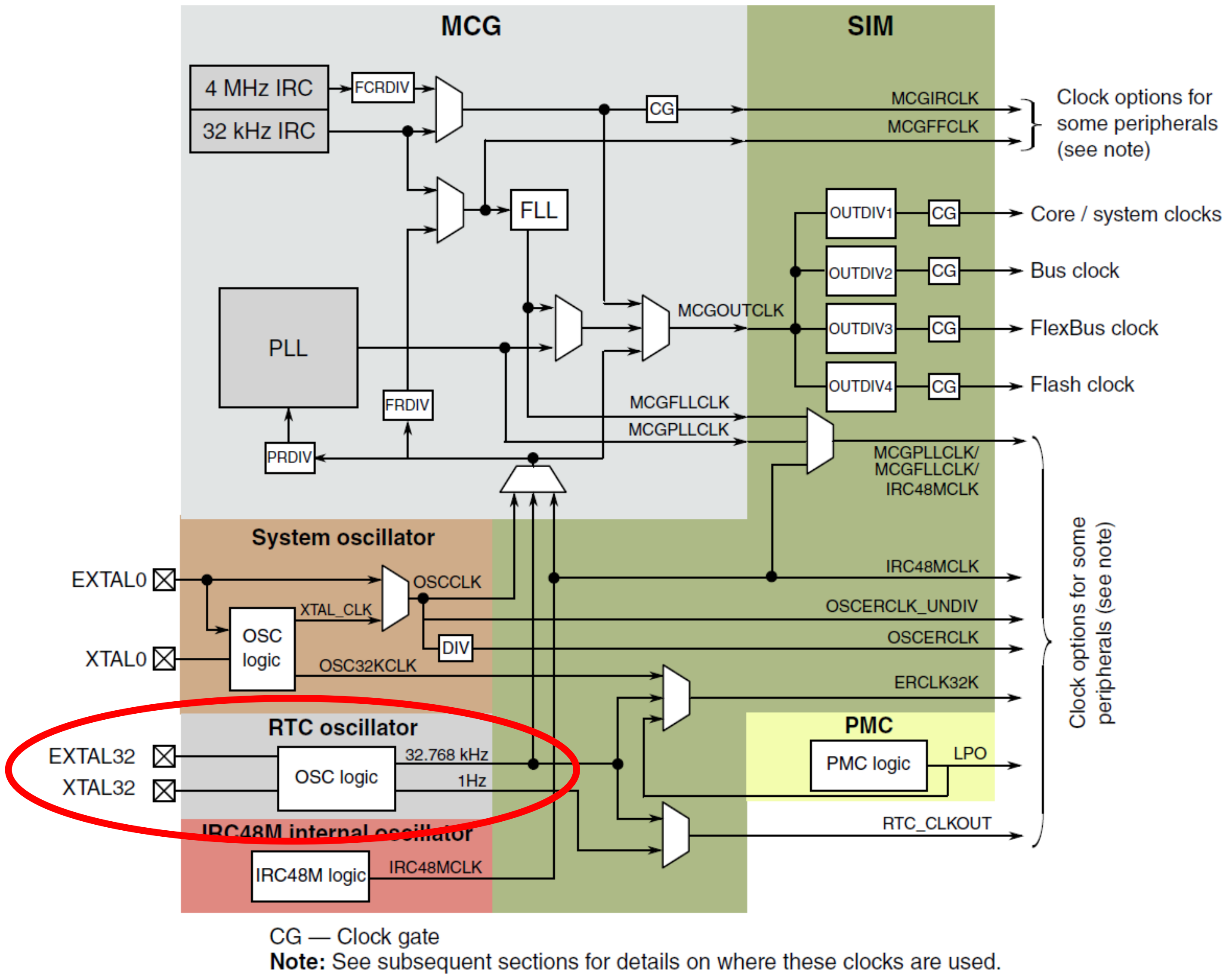
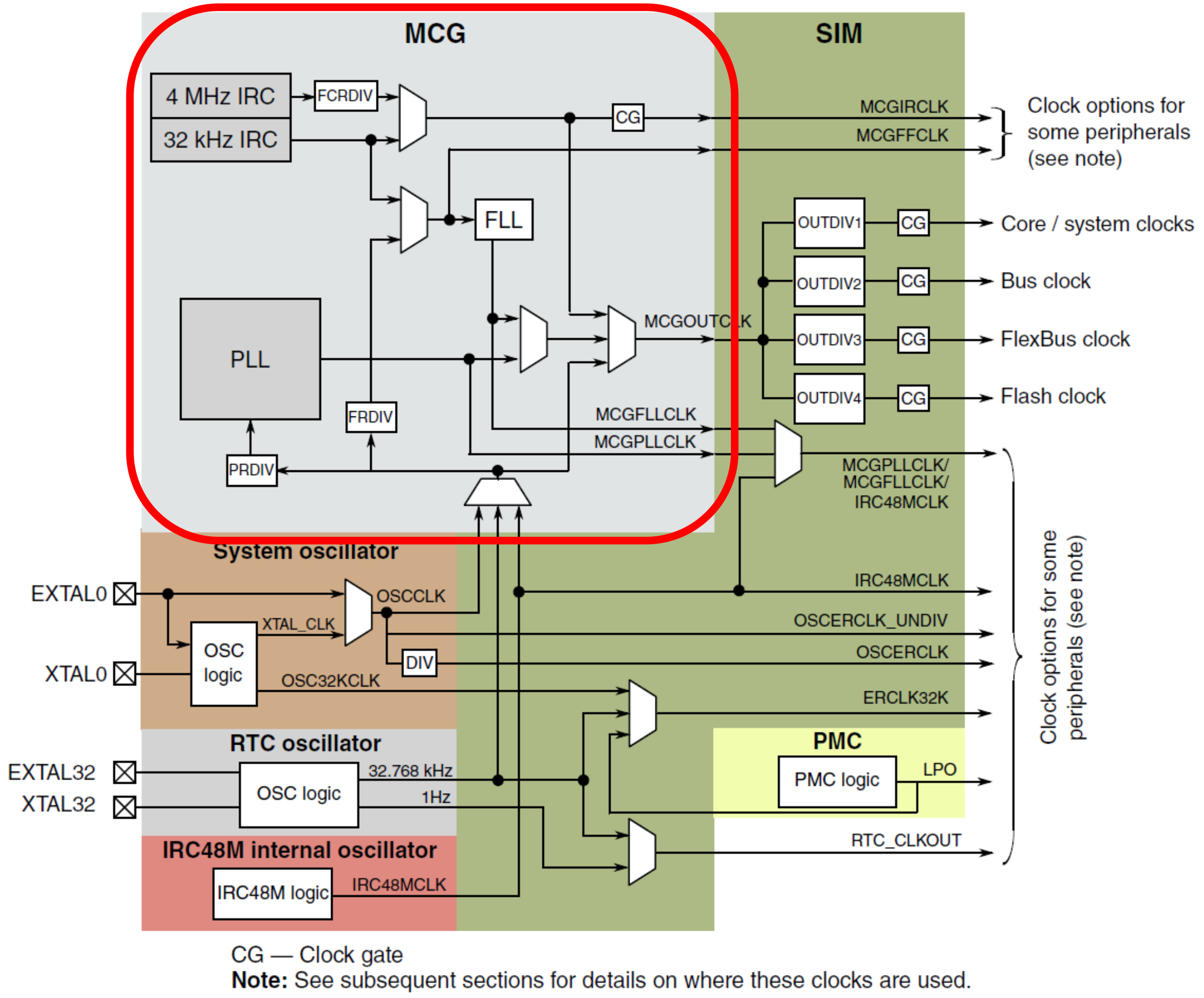


Figure 5-1. Clocking diagram

Multi-clock generator (MCG)

- Reference source (IRC or system oscillator) selectable in software
- Phase-locked loop (PLL) multiplies the frequency of the reference.

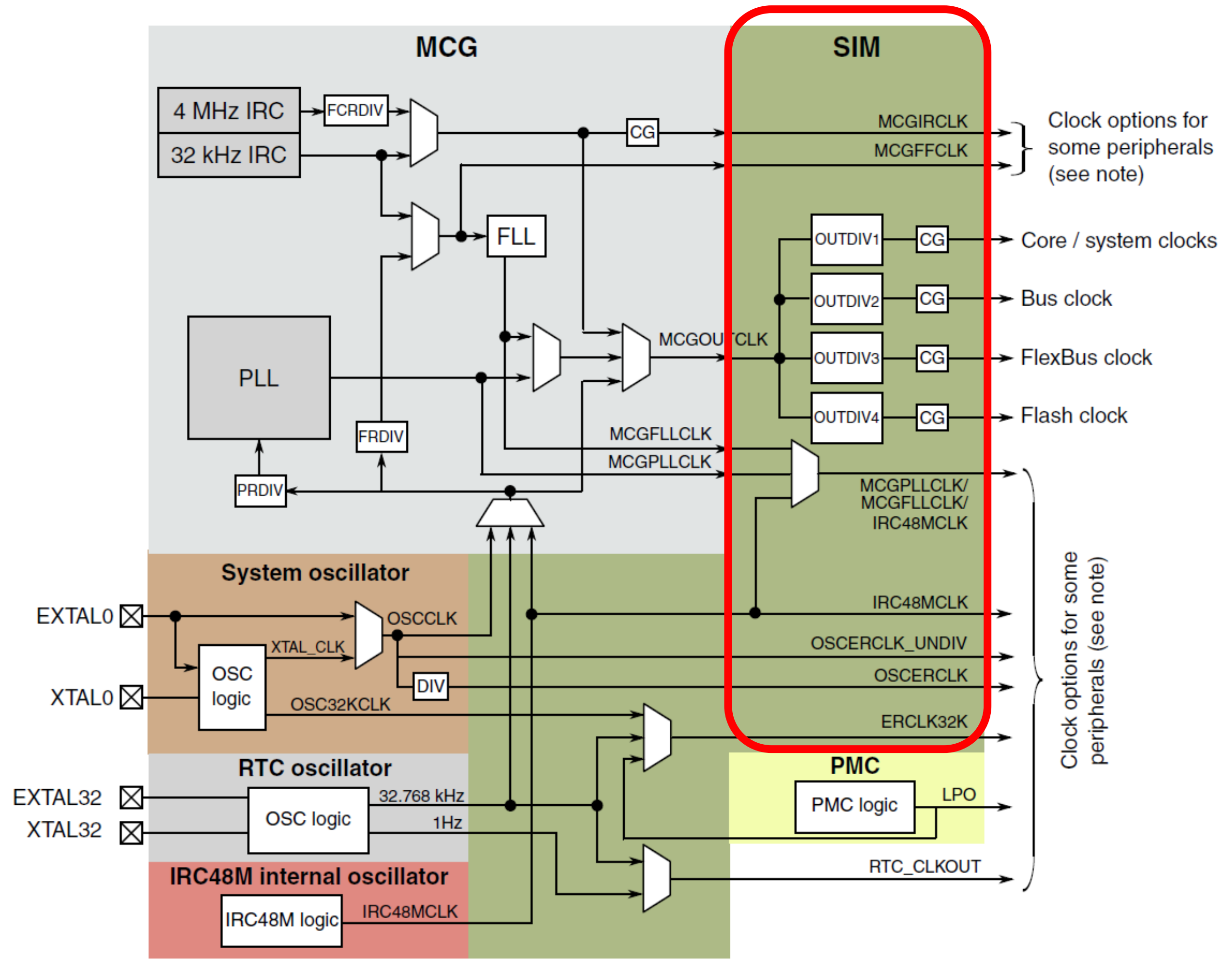


CG — Clock gate
Note: See subsequent sections for details on where these clocks are used.

Figure 5-1. Clocking diagram

System Integration Module (SIM)

- Divides the MCG to generate the system clock, bus clock, and flash clock.
- Selects the clocks to various peripheral components



CG — Clock gate
Note: See subsequent sections for details on where these clocks are used.

Figure 5-1. Clocking diagram

Maximum clock speed settings

- The K20DX128M5 has a maximum clock speed of 50 MHz.
- The K22FN512M12 has two modes:
  - Run mode, with maximum clock speed of 80 MHz.
  - High speed run mode, with a maximum clock speed of 120 MHz.

Maximum speed clock settings

Example of how to achieve a 50 MHz system clock:

- The FRDM board has an 8 MHz crystal. The system oscillator uses this to generate an 8 MHz clock with high precision.
- However, 50 MHz is not an integer multiple of 8 MHz. Use these settings:
- Configure the phase locked loop (PLL) module:
  - Reference clock source = system oscillator (@ 8 MHz)
  - Reference clock divider = 2 (to get a reference clock of 4 MHz)
  - PLL multiplication factor = 25 (4 MHz * 25 = 100 MHz)
- Configure the System Integration Module (SIM) with a system clock divider of 2 to reach 50 MHz.

Clock Summary

- Internal Reference Clock (IRC) – simple, available immediately on power on, but relies on internal components and therefore less accurate.
- System Oscillator – requires some set up, but much more precise. Needed to reach the maximum system clock.
- Real-Time Clock (RTC) – keeps the time of day even when the microprocessor is inactive.
    - Requires a battery source that our boards don’t have. You’d need to solder a coin cell holder onto the back side of the board to use the RTC.

The system clock as a power-performance tradeoff

- When the system clock is lower, the microprocessor will consume less power at the expense of reduced computational speed.
- Raising the system clock increases the speed of the processor at the expense of consuming more power.
- Highly configurable clocks (as in our device) allow precise control of the tradeoff between computing capability and power consumption.
- It’s possible to choose different clocks for different parts of the chip for detailed optimisation of the power-performance tradeoff.
- Power consumption is very important for battery-powered devices.

Signal Multiplexing

- The Port Control module connects internal signals to external pins.
- External pins generally can be used for many different signals.
- The Port Control module selects which signal goes to which pin.

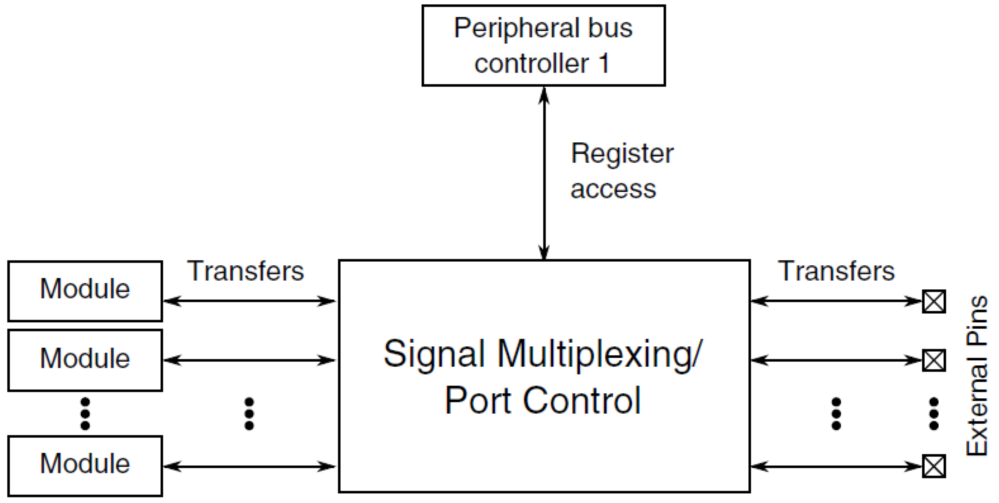
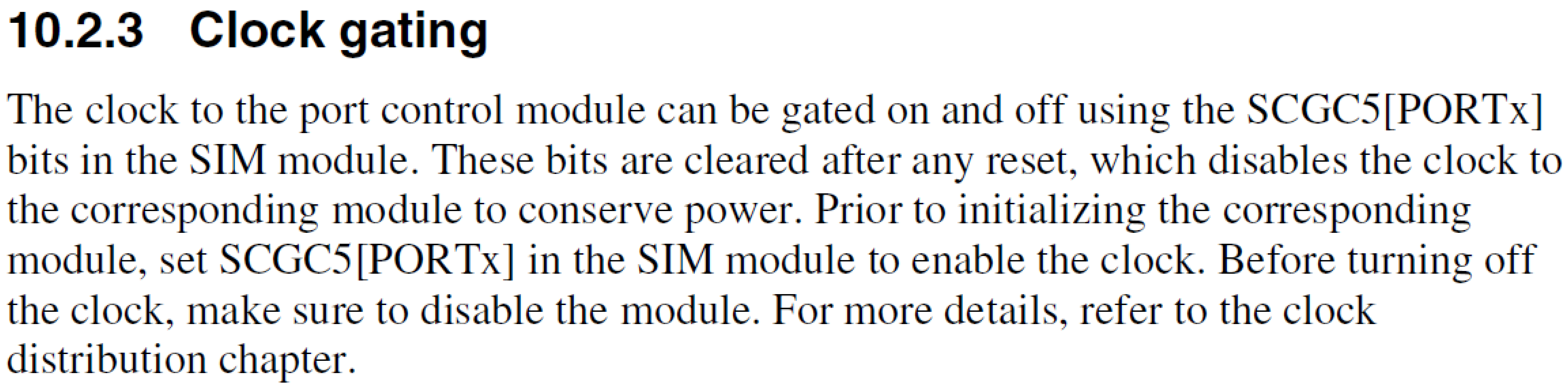


Figure 10-1. Signal multiplexing integration

Read the datasheet carefully!

- The Port Control module does not receive a digital clock by default.
- In the lab, you will need to clock Port Control before you can use it.



10.2.3 Clock gating

The clock to the port control module can be gated on and off using the SCGC5[PORTx] bits in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing the corresponding module, set SCGC5[PORTx] in the SIM module to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution chapter.

Summary

- Use the C code reference to help your programming in the upcoming labs.

This week’s lab

- You’ll write your first microprocessor program.
- Task: control the LEDs on the board.

- Bring your FRDM boards to the lab!