## Agenda

* [Understanding Circuit Diagrams](#_Understanding_Circuit_Diagrams)
* [Importance of Datasheets](#_Datasheets)
* [Software Development Environment](#_Software_Development_Environment)
* [Hands-on Task: Blinking an LED](#_Practical_Task_–)

## Understanding Circuit Diagrams

Why It Matters:

* Firmware interacts directly with hardware.
* Circuit diagrams show the physical connections of components:
* MCU (microcontroller)
* Peripherals (LCD, buttons, LEDs, sensors)
* Power lines, pull-up/down resistors
* Communication buses (UART, SPI, I2C)

Key Takeaways:

* Know which pins are used, and what their electrical behavior is.
* Understand how components are powered and how to avoid hardware damage.
* Identify signal directions: input/output or bidirectional.

## Datasheets

A datasheet is a document provided by the manufacturer of an electronic component that contains detailed technical information about that component.

It tells you everything you need to know to use the component correctly in your hardware or firmware design.

### MCU (ATSAM4S16B)

Why Read MCU Datasheets:

* Pinout Information – what each pin does.
* Clocks and PLLs – needed to configure the system clock.
* Peripheral Descriptions – UART, SPI, TIMERS, etc.
* Interrupt Vectors – essential for real-time behavior.
* Register Maps – for direct control over hardware.

ATSAM4S16B Specifics:

* ARM Cortex-M4 based MCU from Microchip.
* Rich peripheral set: multiple UARTs, SPI, I2C, Timers, PWM, ADCs.
* Clock configuration: multiple internal/external clock sources with PLL.
* Flash and SRAM sizes important for memory planning.

### LCD Module

Importance:

* Know the communication interface: SPI/I2C/Parallel.
* Understand the command set:
  + Initialization sequences
  + Commands for refreshing order (rotation), drawing, etc.

Tips:

* Check for timing constraints (e.g., minimum delays between commands).
* Understand reset and initialization behavior.

### Bluetooth Module (FSC-BT986)

What to Look For:

* Communication type: UART (usually).
* Baud rate configuration
* Supported AT commands (for classic modules)
* Power-on sequence and required delays
* Status pins (e.g., connection state indicators)

Example Use Case:

* To send/receive data via Bluetooth, understand:
* UART configuration on both ends
* Module-specific initialization
* Flow control (RTS/CTS)

## Software Development Environment

Key Concepts:

Regardless of vendor, IDEs typically offer:

* Editor
* Compiler (GCC/ARMCC/etc.)
* Debugger (J-Link, ST-Link, etc.)
* Flashing tools
* SDKs or HALs (Hardware Abstraction Layer)
* RTOS integration (optional)

Microchip Studio with ASF:

* IDE used for ATSAM MCUs
* Based on Visual Studio Shell
* Uses Atmel Software Framework (ASF):
* Pre-written drivers for peripherals
* System clock and peripheral setup templates
* Event system support
* ASF helps avoid writing from scratch:  
  gpio\_set\_pin\_high(), usart\_write\_buffer\_wait(), etc.
* Debugging via embedded debugger or J-Link

## Practical Task – Blink an LED

Objective:

* Blink an LED (200ms ON / 1000ms OFF)

What You Need:

* Use GPIO for LED control
* Use Timer or SysTick interrupt for timekeeping

Steps Overview:

1. Initialize GPIO pin as Output
2. Configure SysTick Timer (1ms tick)
3. Create a simple state machine:

* Track elapsed time in ISR
* Toggle LED state and reset timer when threshold is reached

How to Proceed

1. Development Steps:
2. Create a new ASF project in Microchip Studio
3. Select ATSAM4S16B
4. Enable GPIO and SysTick modules
5. Set system clock if needed (use 12MHz crystal + PLL to get desired speed)
6. Write main.c:

* Init system
* Configure SysTick for 1ms interrupts
* Configure LED GPIO
* Implement LED toggle logic in ISR

## Annex A

### What is a Circuit Diagram?

A circuit diagram is like a map that shows how all the parts of a hardware board are connected together. It includes components like the microcontroller, LEDs, buttons, sensors, power supply, and communication modules.

Each line on the diagram shows how electrical signals flow between these parts.

### Why Do Firmware Developers Need to Understand It?

When you write firmware, you are telling the microcontroller (MCU) how to interact with the hardware. If you don’t know how things are connected, your code may not work—or worse, you might damage the board.

For example:

You need to know which pin is connected to an LED before turning it on.

If a button is pulled up with a resistor, you need to know whether to detect a HIGH or LOW signal when it's pressed.

If an LCD screen is connected via SPI, you need to find the SPI pins and control lines (like CS, DC, or RESET).

### What to look for in a Circuit Diagram:

Pin assignments: Which MCU pins are connected to what.

Power supply lines: Make sure devices have the correct voltage.

Pull-up/pull-down resistors: These affect how you read buttons or inputs.

Interfaces: UART, SPI, I2C connections and their direction (input/output).

Shared buses: Some devices might share the same SPI or I2C lines—you need to control them properly.

### What is a Datasheet?

A datasheet is a big technical document provided by the manufacturer that tells you everything about a device, like a microcontroller (MCU). It may look scary at first, but it's full of important details you’ll need.

#### Why is it Important?

Before you can use any part of the MCU—like a timer, UART, or GPIO—you need to know:

* What registers to configure
* Which clock sources are available
* Which pins are connected to which internal peripherals
* The datasheet is your guidebook to understand how to control the chip.

#### Key Features of the ATSAM4S16B:

* Manufacturer: Microchip (formerly Atmel)
* Core: ARM Cortex-M4, 120 MHz
* Flash Memory: 1 MB (to store your program)
* RAM: 160 KB
* Operating Voltage: 1.62V to 3.6V
* Peripherals:
* 4 UARTs (for serial communication)
* SPI and I2C (for connecting to sensors or display)
* 16-bit Timers and PWM
* ADC (Analog to Digital Converter)

#### How to Use the Datasheet in Practice:

1. Clocks

The MCU can use a crystal oscillator, an internal oscillator, and PLL (Phase-Locked Loop) to generate system clocks.

You can choose different clock speeds depending on your needs (e.g., higher speed for display, lower for low power).

2. GPIO (General Purpose I/O)

Check the pinout table to see which pin is PA0, PA1, etc.

Look at the PIO (Parallel I/O Controller) section to understand how to:

Set a pin as output or input

Write HIGH or LOW

Enable pull-ups or interrupts

3. Timers

Use timers to generate delays, measure time, or trigger periodic tasks like blinking an LED.

The datasheet explains:

* Timer registers
* Prescaler values
* How to enable timer interrupts

4. UART and SPI

Find out:

* Which pins are used for TX/RX (for UART) or MOSI/MISO/SCK (for SPI)
* Baud rate settings (e.g., 9600, 115200)
* How to send/receive bytes using registers