Step 7 to Build a Smart Thermostat Using an MCU: How to Pick the Right MCU

TEXAS INSTRUMENTS

Britta Ruelander

You made it to the last post in this series. Congratulations! You should have almost everything you need to get your smart thermostat application up and running, and you've learned how to tailor it to your application needs. In this last installment, I will summarize what you've learned so far and give you some tips and tricks on how to best choose a microcontroller (MCU) that fits your application needs so that you can put it all into practice.

The First Six Installments

Let's quickly recapture the first six steps that led you here:

1. Understanding the basic building blocks of a smart thermostat.
2. Setting up the right signal chain for temperature measurement.
3. Processing and transferring the captured data.
4. Adding the application interface.
5. Setting up your connected application.
6. Optimizing your application's current consumption.

Now, let's take a look at the heart of a smart thermostat: the MCU. During the course of this series you already got some hints on important MCU characteristics; this installment will guide you through the decision process.

Don't Choose What You Can Get – Take What You Need

To choose the right MCU for your application, it's crucial to look at the system-level needs. Don't get distracted by MCU features you could pick, but rather identify what you need for your application, making it easy to set up, cost-efficient and technically sophisticated.

System-level requirements can include the need for signal amplification (through operational amplifiers or programmable gain amplifiers), integrated analog-to-digital converters (ADCs), high processing power or memory footprint. You might also consider the needs for a small package and/or integrated communication modules. Your application will tell you what it is that you need (most). For many applications, there are trade-offs between two or more different requirements – for example, between high memory-footprint needs and form factor restrictions calling for a small package size.

Your decision points may include:

- **Memory footprint.** Implementing multiple application features can lead to higher memory (code and data) requirements. For example, HMI features (like graphics display, capacitive touch interface), sensor(s) data processing, and communication stacks.
- **Temperature range.** Some environments can have extended temperature ranges at either the low or high ends (or both); keep this in mind for your MCU choice.
- **Clock speed.** Depending on the performance needs of your application, you might want to go for a higher or lower clock speed.
- **Core architecture.** The core architecture of the MCU will support a certain instruction set and bus access; for portability across product families, it can be beneficial to work with one core architecture.
- **Integrated modules.** Some MCUs offer integrated modules for your signal chain like operational amplifiers, ADCs and digital-to-analog converters (DACs), which can bring down the overall system size and cost.
- **Current consumption (peak and average).** The battery lifetime requirements and architecture in battery-powered devices dictate limited current-consumption allowances for average and peak consumption.
- **Package size and package type.** You may encounter space constraints on your printed circuit board (PCB) that set form factor restrictions for the MCU. On the other hand, PCB design (e.g. number of layers) and assembly can impact the package selection decision. In other words, very small packages can save you space but may require more PCB layers and sophisticated assembly; thus, driving overall PCBs cost higher.
- **PCB form factor constraints.** The application size itself can limit the overall PCB size; a smart thermostat certainly shouldn't occupy a lot of space in a living room. This again is a reason that could lead you to look at higher system integration. Remember that there are more parts on the PCB than just the MCU.
- **Price.** Last but not least, you will have a target price – either the amount you can charge your customers or simply the limit that you wanted to spend., Consider system integration and the quality that the MCU brings to the application for overall system cost considerations.

There may be more requirements, including device scalability, the software ecosystem and overall supplier support. Of course, this is also applicable to you if you are building your own application: You might consider any former experience with tools or devices that can help you to get started even faster. To make a decision, consider your requirements on the chip but also on the system level.

Get Going Now with the MSP430FR4133

Throughout this series, my co-author and I included device suggestions, like the MSP430FR2633 capacitive touch solution or MSP432™ wireless host MCU. Here is another: for a low cost single chip thermostat application, check out the MSP430FR4133, which comes with an integrated 10 Bit ADC, 16 kB of ferroelectric random access memory (FRAM), different serial communication channels and integrated liquid crystal display (LCD) drivers for your simple human machine interface (HMI) setup.

I hope that you learned a lot in this series and can't wait to get your hands dirty. Check out other Texas Instruments blogs to learn about new ideas, tips, tricks and trends.

Additional Resources
- Start working with the MSP430FR4133 in the Thermostat Implementation with FRAM Microcontroller Reference Design.
- Check out other recent blog posts in the TI E2E™ Community to learn about new ideas, tips, tricks and trends.

IMPORTANT NOTICE AND DISCLAIMER