Developer's Guide To Pacemaker Development

Tutorial 1.1: Pacemaker Project Overview

SFWRENG/MECHTRON 3K04
McMaster University

September 2, 2025

Getting Started With The Pace-maker Project

The purpose of this tutorial is to outline the vision, scope and goal of the pacemaker project. Since a modest understanding of embedded systems is required for the project, this document begins by providing an overview of embedded systems. By the end of this tutorial, you will have gained a stronger understanding of the system you will be developing for the course project. You will have also begun to decompose the project requirements.

Topics Covered

- The pacemaker as an embedded system
- What am I building? Vision, goal and scope of the pacemaker project

1 Background

1.1 Understanding the Pacemaker System

Embedded systems are all around us. If you use a thermostat to keep your room warm so you can attend your online lectures comfortably, if you use a laundry machine to do your laundry, if you use a dishwasher to wash your dishes, or a microwave to heat up your food — you have interacted with a machine with an embedded system. From the modules in a digital camera to the engine control unit in an industry airplane, embedded systems are an integral part of life in our modern world.

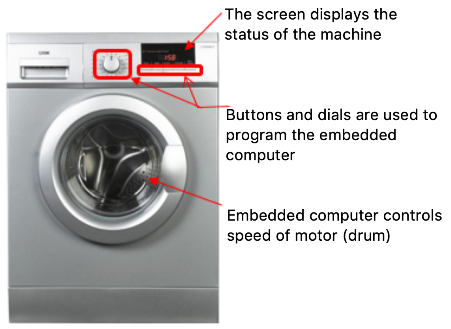

The screen displays the status of the machine
Buttons and dials are used to program the embedded computer
Embedded computer controls speed of motor (drum)

Figure 1: Embedded System Examples

The pacemaker is an example of an embedded system. An **embedded system** can be thought of as a computing system that is made up of hardware that is controlled by software. Embedded systems, in contrast to general computing systems, are designed and optimized to perform a specific, dedicated function within a larger electrical or mechanical system. They are "embedded" into the system in which they control.

The following diagram illustrates the Input-Output (I/O) and Communication relationships of an embedded system. The blue arrows represent data or communication channels; the red arrows represent control channels.

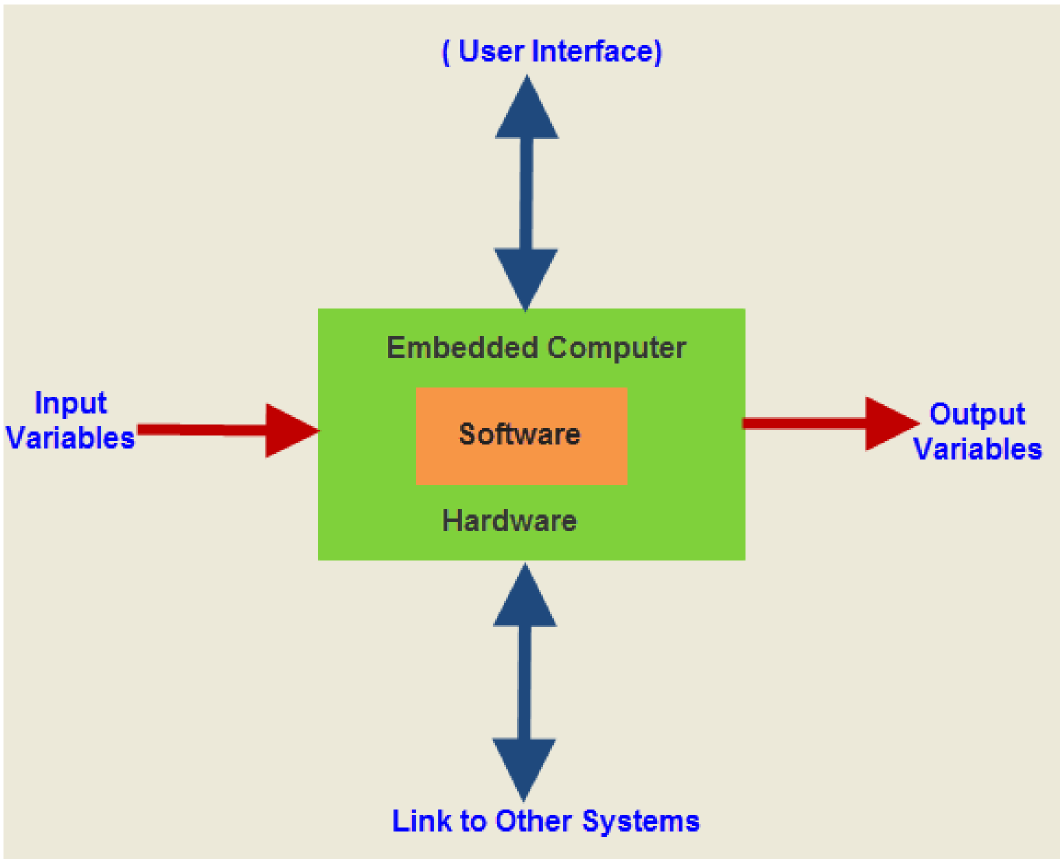


Figure 2: External Block Diagram of an Embedded System

The Embedded Computer: the core component of the embedded system. In the context of the pacemaker system, the embedded computer is the pulse generator.

Software: embedded software resides in the memory of the embedded computer. The software governs the relationship between the input variables, the output variables and timing. Pacemaker software implements bradycardia operating modes and a communication protocol between the embedded computer and the user interface. Bradycardia operating modes will be elaborated on in **Tutorial 2 — Pacing, Sensing and Timing Cycles**.

User Interface: the user interface enables a person to interact with the embedded computer via a set of sensory and controls channels. A user interface may be local (i.e. an LCD screen, push buttons or dials mounted directly on the embedded computer) or remote (i.e. a connection to a screen that is separate from the unit and displays the status of the embedded computer). In the context of the pacemaker system, the Device Controller-Monitor (DCM) includes a remote graphical user interface (GUI) that allows the user to receive data from the pacemaker device and, given that the pacemaker is **programmable**, transmit a new set of instructions to alter the behaviour of the pacemaker device.

Input variables: represent the physical connection between the pacemaker leads and the Cardiac Conduction System (CCS) that allows the pulse generator to sense the electrical activity of the the heart.

Output variables: represent the physical connection between the pacemaker leads and the Cardiac Conduction System (CCS) that allows the pulse generator to send electric stimuli to the heart.

Link to Other Systems: the pacemaker system does not interact with any other system(s).

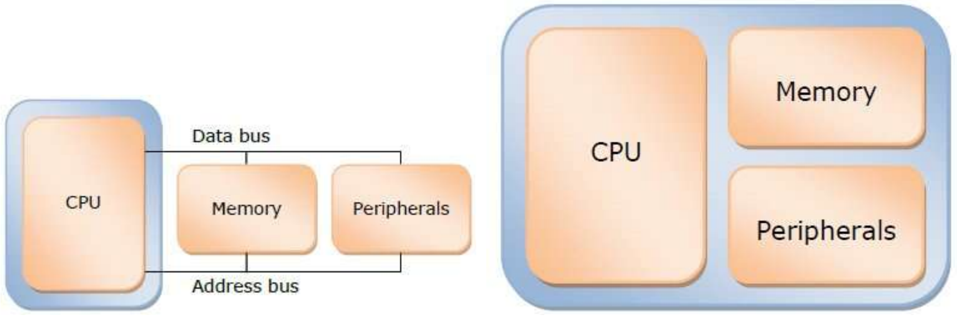
A Closer Look at the Embedded Computer

Embedded computer hardware is based on microprocessors or microcontrollers, and is made up of a processor, memory devices, and peripherals.

Processor: a processor (CPU in Figure 3) is known as the "brain" of the system. The processor executes the embedded program instructions by performing arithmetic operations, managing data flow, and generating the appropriate control signals. The Arithmetic Logic Unit (ALU) and the Control Unit (CU) make up the two parts of the processor that work together to fetch, decode and execute instructions.

Memory: memory devices store programs or data for use by the processor. Memory is primary classified based on persistence of data storage (also known as volatility). **Volatile memory** hold their contents only as long as power is supplied to the memory device and allow faster access to memory contents. **Nonvolatile memory**, also known as persistent memory, retain their contents even after power has been removed.

Peripherals: peripherals are IC devices that assist the processor. Most peripherals are used by the embedded computer to interface with the outside world. Of particular interest to the pacemaker system are timers, serial communication ports and parallel ports (discussed below). Interrupt controllers may also be advantageous.

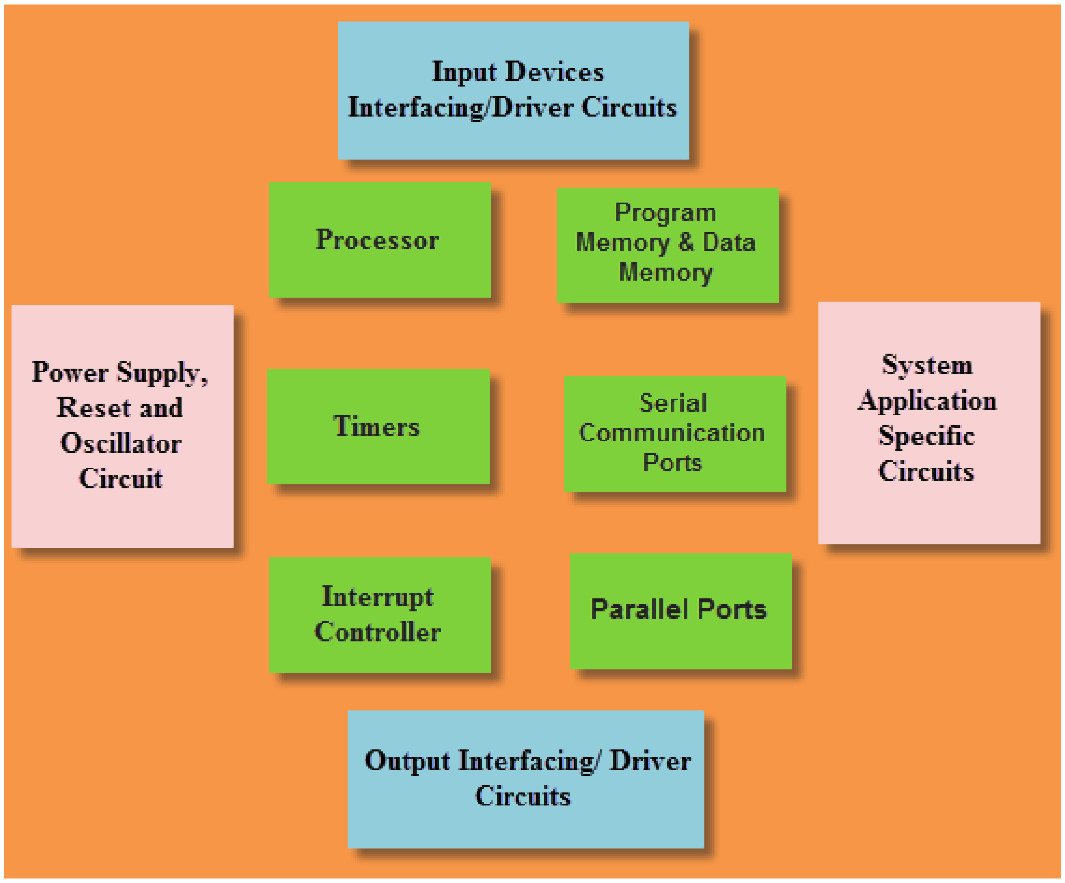


(a) A Microprocessor-Based System. (b) Microcontroller-Based System

Figure 3: High-level Comparison of the Microprocessor-Based System and Microcontroller-Based System.

Figure 3 illustrates the hardware configurations of a microprocessor-based system and a microcontroller-based system. A **microprocessor unit (MPU)** is an integrated circuit (IC) that contains all (and only) the functions of the Central Processing Unit (CPU) of a computer. Memory and peripherals are connected externally to the MPU chip to build an embedded computer. In contrast, a **microcontroller unit (MCU)** has a CPU along with integrated memory and/or peripheral interfaces on a single chip. MCUs are used to perform applications that have tasks that are fixed and predefined, whereas MPUs are more desirable where intensive processing, and perhaps a flexible hardware configuration, is required. When it comes to size, MCUs are more desirable in applications where a small size is a key requirement. All else equal, MCUs generally consume less power than MPUs because they are manufactured with power management circuitry.

For our purposes, the pacemaker system is based on a MCU called the **FRDM-K64F (K64F)**. The K64F has most of the components shown in Figure 4 built into the board. For more information on the board's features, refer to: `https://os.mbed.com/platforms/FRDM-K64F/` and `https://inst.eecs.berkeley.edu/~ee192/sp18/files/FRDMK64FUG.pdf`.



Input Devices Interfacing/Driver Circuits
Processor
Program Memory & Data Memory
Power Supply, Reset and Oscillator Circuit
Timers
Serial Communication Ports
System Application Specific Circuits
Interrupt Controller
Parallel Ports
Output Interfacing/ Driver Circuits

Figure 4: Embedded System — Internal Block Diagram.

Power Supply, Reset and Oscillator Circuit: generates a reset signal when power is supplied to the system. The K64F receives power through a USB cable connected to your personal computer.

Program Memory & Data Memory: the instructions to be executed by the processor are stored in **program memory**; the data being manipulated by the processor is stored in **data memory**. Celebrated for its high speed, long life and electrical programmability, flash memory is the most common program memory type in embedded systems. **Flashing** is a term that refers to the process of writing a program to flash memory. Once the K64F is flashed, it will execute the program every time the board is powered on. A typical flash memory layout has a bootloader area and a program area. A bootloader is a type of **firmware** (software that is "firm" — not intended to change) that is executed before the main program. The bootloader contains the code that communicates with your personal computer, accepts the main program file/binary, and writes it to the program area.

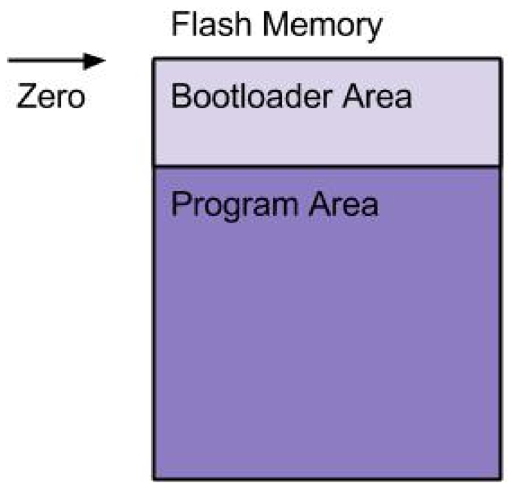


Figure 5: Flash Memory Layout.

Timers: a timer is a peripheral that counts processor cycles or clocks to measure elapsed time. A timer is essential for the pacemaker system because it helps the pulse generator measure or generate time intervals that synchronize the timing of the pulse generation or sensing operation.

Serial Communication Ports: provide an interface for serial communication with external devices. Serial communication is a process where data is sent one bit at a time over a communication channel. Serial communication is used by the K64F when you flash or debug pacemaker software, or when you operate the DCM. Serial communication will be explored in more detail in **Tutorial 3 — Serial Communication**.

Interrupt Controller: an interrupt controller listens to peripherals for events and reports to processor once an event, an "interrupt", occurs. The use of interrupts is often contrasted with a process called polling. Polling is when the software actively checks the status of external devices. Interrupts improve the performance and power consumption of the embedded computer because less operations are required from the processor.

Parallel Ports: a parallel port, or **General Purpose Input and Output (GPIO)**, is an input output communication interface that allows the embedded computer to exchange digital information with the external world. The pins that come out of the microcontroller board are mapped to GPIO. Pins (or pin headers) provide electrical contact with the GPIO. A pinout is a visual map of the microcontroller pins to their functions. Functions include digital I/O, analog I/O and Pulse Width Modulation (PWM) (more on PWM in **Tutorial 2 — Pacing Basics**). Figure 6 is a pinout of the FRDM-K64F microcontroller. The K64F has 16 digital pins and 6 analog pins. The digital pins are prefixed with a "D", whereas the analog pins are prefixed with an "A".

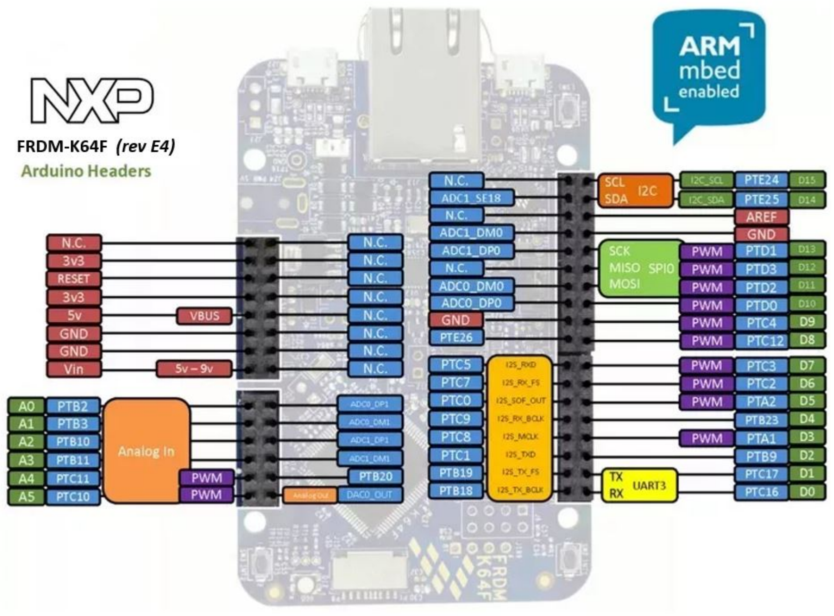


Figure 6: FRDM-K64F pinout.

System Application Specific Circuits: circuits designed for a particular application.

Other Peripherals: the K64F is also manufactured with an on-board **accelerometer** — a device that measures the acceleration of a moving entity. In the context of the pacemaker system, the on-board accelerometer provides an indication of a person's physical activity level.

I/O Interfacing/Driver Circuits: a driver circuit controls other electric components or circuits, and is usually connected to GPIO pins. The driver circuit you will be using for the pacemaker system is called the pacemaker shield. The pacemaker shield is a **printed circuit board (PCB)** that is mounted on the K64F. A PCB is a self-contained electrical circuit with components and conductors contained within the same mechanical structure. More information on the pacemaker shield can be found in the document "pacemaker-shield-explained" on Avenue.

Summary — Pacemaker System Description

In addition to understanding the pacemaker as an embedded system, a pacemaker is, to a certain extent, a **control system** because it manages, commands, directs and regulates the behaviour of another system — the Cardiac Conduction System. The pacemaker is a **real-time system** because it is subject to hard real time constraints. Lastly, the pacemaker is a **safety-critical system** because it has the potential of harming human beings.

There we have it! Altogether, a pacemaker is a safety-critical, real-time control embedded system.

1.2 The Project

Vision Improve familiarity with model-driven development and apply principles of software design.

Goal Design and implement system that operates a cardiac pacemaker under the specified modes.

Scope Design and implement the embedded pacemaker software, and the driver software and user interface for the DCM. You are also required to verify and document your software.

Deliverables

- MATLAB Simulink® model implementing the pacemaker operating modes
- DCM software
- Technical Documentation
- Live demonstration

Development Paradigm
A **model-driven development (MDD)** paradigm will be used develop the pacemaker software. MDD "uses models as the primary artifact in the development process" (definition from: IGI Global). MDD is important in embedded systems because it reduces development time, improves collaborative development, leads to increased software quality and reliability, and produces software that is less error prone when compared to traditional design processes. MDD is prominent in embedded system development across many industries today.

About the Pacemaker System Requirements Specification (SRS)
The deliverables for the project are based on the Pacemaker SRS. The Pacemaker SRS is an industry standard requirements document published by Boston Scientific in 2007. The SRS was used in a certification challenge problem issued by the Software Certification Consortium (SCC) called the Pacemaker Formal Methods Challenge to create certifiably safe, secure and dependable software for the pacemaker safety critical system. **Formal methods** involves using mathematical techniques when specifying, developing and verifying software, and emphasizes correctness and proof as a measure system integrity.

2 Tutorial: Getting Familiar with the Project Requirements

1. Review theory on the Cardiac Conduction System and pacemakers by reading the following resources:
   - What Is A Pacemaker?: The Cardiac Conduction System and the Artificial Pacemaker
   - Model Based Design Pacemaker (these are slides summarizing this tutorial and the theory of the pacemaker)
2. Review the **Deliverable 1** handout and corresponding sections in the PACEMAKER SRS requirements document. Ask your TA if you have any questions.

3 Revision History

Version	Date	Modification	Modified by
1.0	Sept. 14, 2020	Initial Document Creation	Kehinde, Michael
2.0	Sept. 2, 2025	Update Instructions and Schedule	Zavaleta, Angela