EMBSYS 110, Spring 2017

- Design & Optimization of Embedded & Real-Time Systems.
- Lawrence Lo, Embedded Software Engineer.
- Currently focused on medical devices.
- Previous experiences include data center management and multimedia extension systems.
- Welcome to the class!

Course Outline

- Week 1
  - Introduction to software design.
  - Review of ARM programming model and architecture.
- Week 2
  - Object-oriented design and UML.
  - Essential C++ techniques.
- Week 3
  - QP statechart framework (guest speaker).
- Week 4
  - QP internals.
  - Application framework design.

Course Outline...

- Week 5
  - Example project.
  - Design patterns and heuristics.
- Week 6
  - Robustness and error handling strategies.
  - Debugging, profiling and testing.
- Week 7
  - Optimization (zero-copy, DMA, multiple instance, non-blocking).
  - Cache.

Course Outline...

- Week 8
  - More optimization and power management.
  - Bootloader and firmware upgrade.
  - Embedded security.
  - Design examples.
- Week 9
  - Review of embedded software architecture (with and without QP).
  - Unit and integration testing.
  - Software development processes (safety-critical and Agile).
  - Open source toolchains and other statechart tools.
  - Embedded vs cloud.
- Week 10
  - Review and conclusion.

Textbook and References

- Textbook
  - Samek, Miro. Practical UML Statecharts in C/C++, 2nd Edition. Newnes. 2008. Available at:

    http://www.state-machine.com/psicc2/index.html.
  - Gomaa, Hassan. Real-Time Software Design for Embedded Systems. Cambridge University Press. 2016. Available online at UW Library.
- References
  - Lippman, Stanley B., Lajoie, Josee, C++ Primer, 3rd Edition. Addison Wesley. 1998.
  - Various publications as introduced in the class.

Grading

- Participation
  - Students must attend at least 8 of the 10 lectures.
- Pass/Fail based on:
  - 75% for assignments (individual effort).
  - 25% for course project (individual effort).

Introduction to Software Design

- Discussion
  - What is SW design?
  - How do you do design?
  - What is output of the design process?
- My mentor once told me:
  - Our job is simple - just program the micro-controller to do what you think it should do.
  - Obvious it appears, but it brings up two keys points:
    - Do we know by ourselves what we want the MCU to do?
      (And how do we know? Can we capture it?)
    - Is there a way to convey that *knowledge* to program code?

Introduction to Software Design

- SW design is abstract that can be hard to visualize.
- SW design is
  - A precise description of system behaviors that fulfill the requirements (model).
  - A software architecture that supports the execution of the model (framework).
- A designer needs to
  - Understand how the system should behave. (Obvious?)
  - Capture (document) the behavioral model. (Why?)
  - Convert the model to executable code. (How?)

A Design Story

- Statecharts in the Making: A Personal Account.

  http://www.wisdom.weizmann.ac.il/~harel/papers/Statecharts.History.pdf

- Statecharts: A Visual Formalism for Complex Systems.

  http://www.wisdom.weizmann.ac.il/~dharel/SCANNED.PAPERS/Statecharts.pdf

  Ranked 55th as most cited computer science articles.

A Design Story

- See Statecharts in the Making.pdf
- Quoted from: Harel, David. Statecharts in the Making: A Personal Account. Communications of the ACM. March 2009.
- “An avionics system is a great example of what Amir Pnueli and I later identified as a reactive system [HP85]. The aspect that dominates such a system is its reactivity; its event-driven, control-driven, event-response nature, often including strict time constraints, and often exhibiting a great deal of parallelism. A typical reactive system is not particularly data intensive or calculation-intensive. So what is/was the problem with such systems? In a nutshell, it is the need to provide a clear yet precise description of what the system does, or should do. Specifying its behavior is the real issue.”

A Design Story...

- “In my naïve eyes, this looked like a bizarre situation, because it was obvious that someone, eventually, would make a decision about what happens when you press a certain button under a certain set of circumstances. However, that person might very well turn out to be a low-level programmer whose task it was to write some code for some procedure, and who inadvertently was making decisions that influenced crucial behavior on a much higher level.”
- The goal was to try to find, or to invent for these experts, a means for simply saying what they seemed to have had in their minds anyway.

A Design Story...

- “Still, it was pretty easy to see that just having one big state machine describing what is going on would be fruitless, and not only because of the number of states, which, of course, grows exponentially in the size of the system. Even more important seemed to be the pragmatic point of view, whereby a formalism in which you simply list all possible states and specify all the transitions between them is unstructured and non-intuitive; it has no means for modularity, hiding of information, clustering, and separation of concerns, and was not going to work for the kind of complex behavior in the avionics system. And if you tried to draw it visually you’d get spaghetti of the worst kind. It became obvious pretty quickly that it could be beneficial to come up with some kind of structured and hierarchical extension of the conventional state machine formalism.”

A Design Story...

- “the two main ideas in statecharts are hierarchy and orthogonality”

More Stories

- Unintended Acceleration Report Appendix A Software

  https://one.nhtsa.gov/About-NHTSA/Press-Releases/NHTSA%E2%80%93NASA-Study-of-Unintended-Acceleration-in-Toyota-Vehicles

More Stories

- From Protocol Specification to Statechart to Implementation.

  https://ml.jpl.nasa.gov/papers/wagstaff/wagstaff-protocol-statecharts-08.pdf
- Automatic Code Generation for Instrument Flight Software.

  https://ml.jpl.nasa.gov/papers/wagstaff/wagstaff-autocoding-08.pdf
- UML Statechart Coding for the Mars Space Lab.

  http://flightsoftware.jhuapl.edu/files/2012/FSW12_Benowitz.pdf

Embedded System Architecture

- ARM Cortex-M4 Core.
- On-chip Peripherals.
- On-board Peripherals.

Embedded System Architecture

- ARM Cortex-M4 Generic User Guide:
- http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf
- STM32F401 Reference Manual:

  http://www.st.com/content/ccc/resource/technical/document/reference_manual/5d/b1/ef/b2/a1/66/40/80/DM00096844.pdf/files/DM00096844.pdf/jcr:content/translations/en.DM00096844.pdf
- STM32 Nucleo-64 Board User Manual:

  http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

ARM Cortex-M4 Core

- ARM Cortex-M4 Core
  - See ARM Cortex-M4 Generic User Guide.pdf
  - **Programmers model**
    - Processor modes, stacks and core registers.
  - **Memory model**
    - Memory map, endianness. (Section 2.2 p. 2-12)
    - Compare to STM32F401 Reference Manual Table 1 and Table 3.

ARM Cortex-M4 Core

- **Exception model**
  - Exception number, IRQ number, priority, vector address. (See Table 2-16 and Figure 2-2)
  - Exception entry and return.
  - NVIC (Nested Vectored Interrupt Controller).
  - Compare different ways to disable/enable interrupts.
    - __disable_irq() / CPSID I (Table 3-2, Section 3.12.2 bottom) (via PRIMASK).
    - __set_BASEPRI() (via BASEPRI) (more flexible).
    - NVIC_EnableIRQ() (via NVIC/SCB) (individual).

ARM Cortex-M4 Core

- **Other Cortex-M4 Peripherals**
  - SysTick
  - System Control Block (SCB)
  - Memory Protection Unit (MPU)

On-chip Peripherals (STM32F401)

- On-chip Peripherals (STM32F401)
  - See STM32F401 Reference Manual.pdf
  - See memory map in Table 1 and Table 3.

    Compare to memory map of ARM Cortex-M4.
  - Note down address of internal Flash and SRAM.
  - Note down boot address.

    Compare to VTOR of System Control Block of ARM Cortex-M4.
  - Check out EXTI and how to disable interrupt at chip level (Section 10.3.1 on page 208).

On-board Peripherals (Nucleo)

- On-board Peripherals (Nucleo)
  - See STM32 Nucleo-64 User Manual.pdf.
  - What peripherals are available on board?
  - How to connect to external modules using SPI, I2C, PWM connectors?

Review of SW Architecture

- Polling loop.
- Foreground-background with interrupts.
- Multi-threaded with RTOS.
- What's the next level up?

Source Code

- Demo project can be downloaded from:

  https://github.com/galliumstudio/stm32f401-nucleo-iar-demo