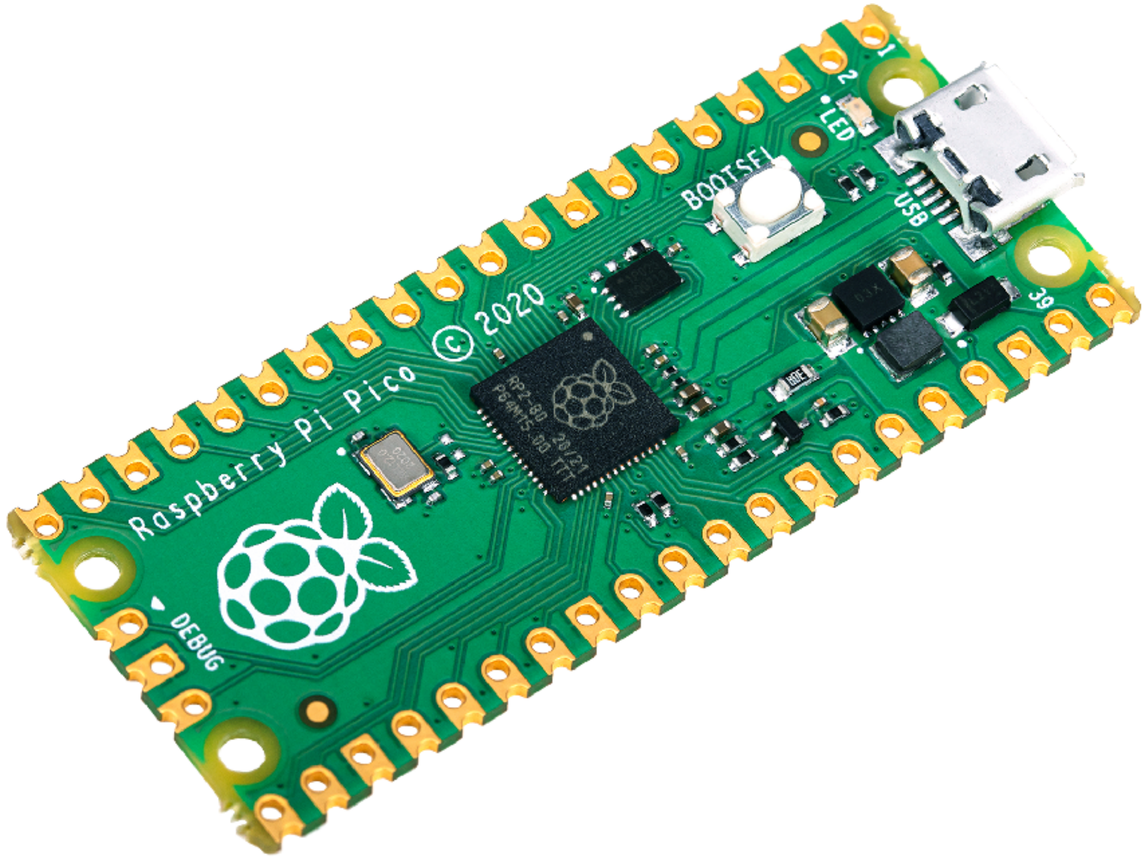
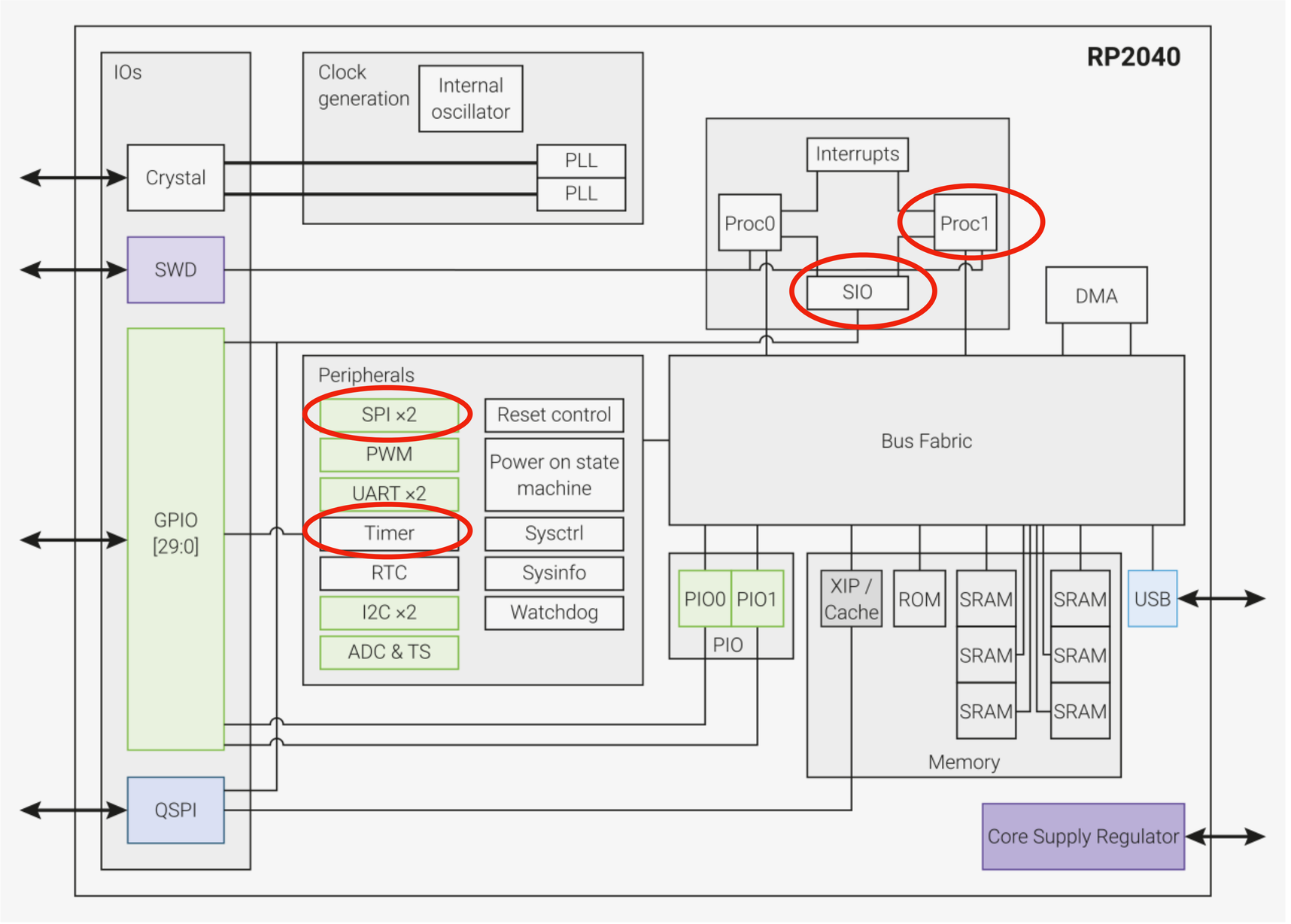


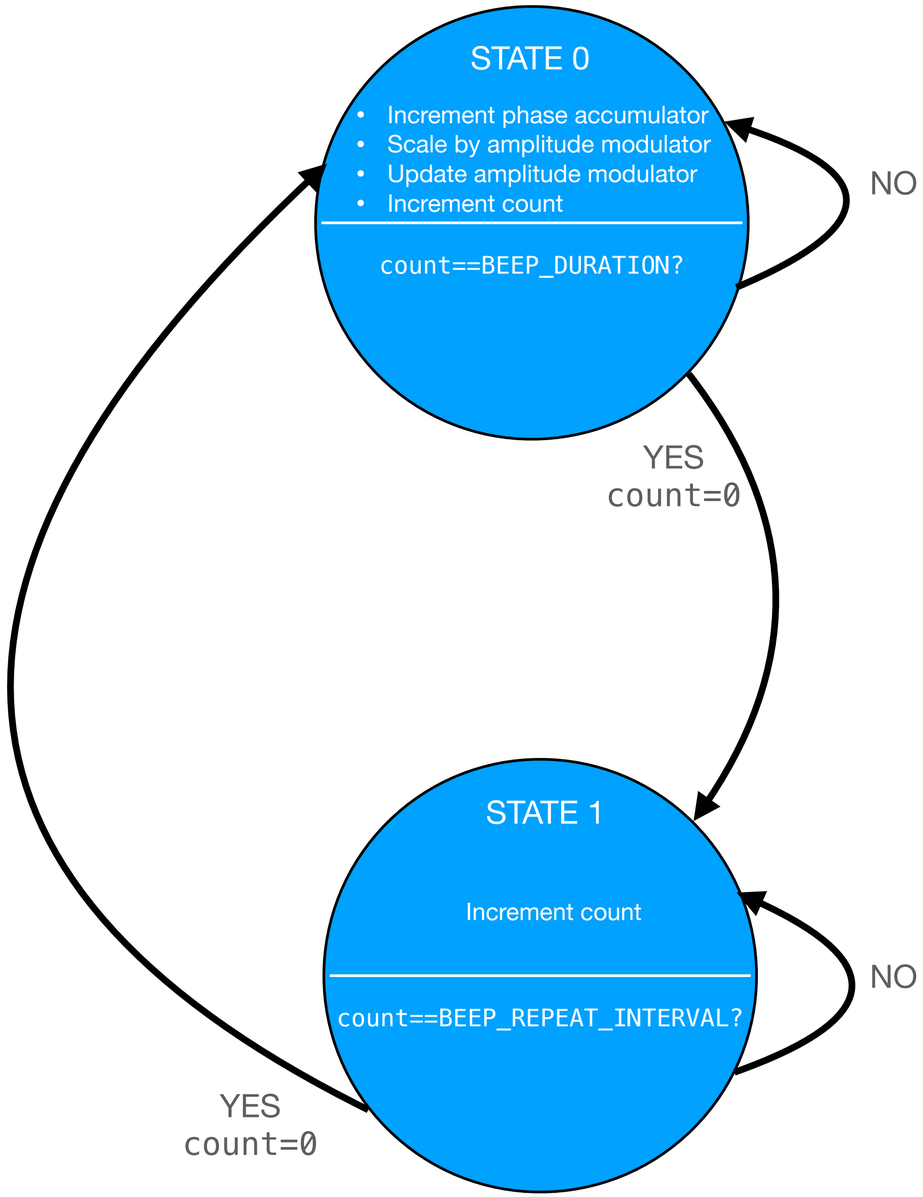
Raspberry Pi Pico (RP2040)

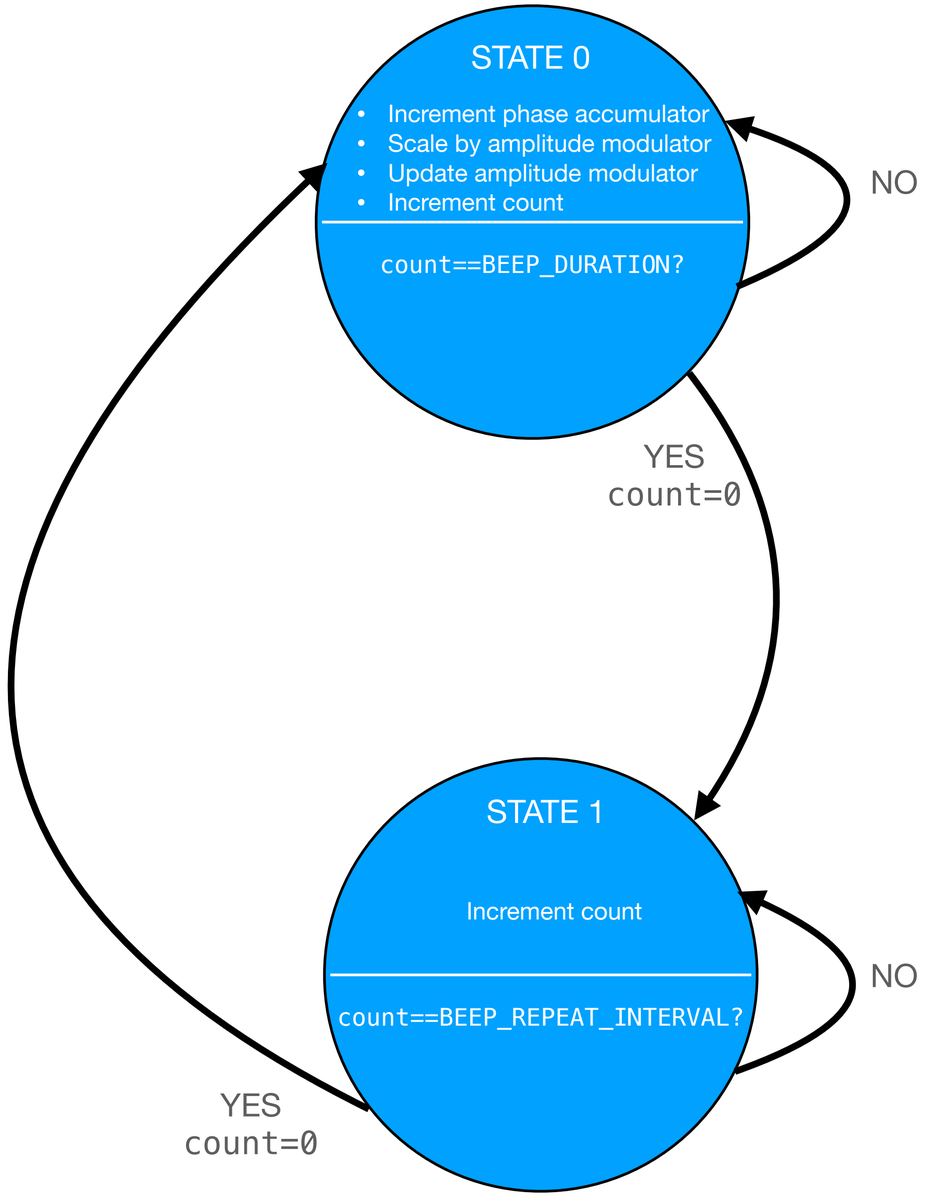
Lecture 5: Multicore and Protothreads


RP2040
IOs
Crystal
SWD
GPIO
[29:0]
QSPI
Clock
generation
Internal
oscillator
PLL
PLL
Interrupts
Proc0
Proc1
SIO
DMA
Peripherals
SPI ×2
PWM
UART ×2
Timer
RTC
I2C ×2
ADC & TS
Reset control
Power on state
machine
Sysctrl
Sysinfo
Watchdog
Bus Fabric
PIO0
PIO1
PIO
XIP /
Cache
ROM
SRAM
SRAM
SRAM
SRAM
SRAM
SRAM
Memory
USB
Core Supply Regulator

Today’s objectives:

- Multicore
- Protothreads


STATE 0
Increment phase accumulator
Scale by amplitude modulator
Update amplitude modulator
Increment count
count==BEEP_DURATION?
NO
YES
count=0
STATE 1
Increment count
count==BEEP_REPEAT_INTERVAL?
NO
YES
count=0



How will you augment this?

- **3 states**: `chirp, syllable pause, chirp pause`
- Both `syllable pause` and `chirp pause` always transition to `chirp`
- `chirp` transitions to one or the other based on the number of syllables it has already produced

Protothreads

a lightweight, stackless threading library created by Adam Dunkels

Threading libraries

A non-threaded microcontroller program generally has the following structure:

- Includes
- Global variable declarations and macros
- ISR’s
- (Maybe) some helper functions
- Main - configures hardware peripherals, then repeatedly executes a set of instructions

Threading libraries

A non-threaded microcontroller program generally has the following structure:

- Includes
- Global variable declarations and macros
- ISR’s
- (Maybe) some helper functions
- Main - configures hardware peripherals, then repeatedly executes a set of instructions

A *threaded* microcontroller program generally has the following structure:

- Includes
- Global variable declarations and macros
- ISR’s
- (Maybe) some helper functions
- Threads - a collection of *separate* execution environments
- Main - configures hardware peripherals, and adds threads to the scheduler, which context switches among threads

When are threading libraries nice?

Suppose that you are writing a complicated concurrent program, like in this class.

In such a program, you want for certain pieces of code to only execute when certain conditions are met (e.g. a user has entered a value, a certain amount of time has passed, a variable has assumed some value, etc.).

You could accomplish this with a state machine, but those can grow cumbersome.

A threading library makes your code more modular, and makes it easier to add conditional code to your application.

Protothreads

- Non-preemptive
- Stackless
- Based on C macros that utilize *Duff's Device*

To the code!