# Embedded Rust and the

MONOTRON

- Jonathan 'theJPster' Pallant
- ACCU, April 2019
# Preamble: Introductions

- @therealjpster (Twitter)
- @thejpster (Github)
- keybase.io/thejpster
# Preamble: What can I expect?

- A tale of obsession
- Right tool, to fix the wrong thing
# Agenda

- Act 1 - Embedded Rust <--
- Act 2 - The Idea
- Act 3 - The Implementation
- Act 4 - Spiralling out of control
- Act 5 - The Demo
Act 1: Rust 2018

===================================
# Act 1: Things you need for Embedded Rust

1. LLVM Backend
2. Target File
3. libcore
# Act 1: UART / GPIO Example

```rust
// USB Serial UART
let mut usb_uart = Serial::uart0(
p.UART0,
porta.pa1.into_af_push_pull::<gpio::AF1>(
   &mut porta.control
),
porta.pa0.into_af_push_pull::<gpio::AF1>(
   &mut porta.control
),
(),
(),
115200_u32.bps(),
NewlineMode::SwapLFtoCR,
&clocks,
&sc.power_control
);
```
pub fn free<F, R>(f: F) -> R
where
    F: FnOnce(&CriticalSection) -> R,
{
    let primask = register::primask::read();
    disable();
    let r = f(unsafe {
        &CriticalSection::new()
    });
    if primask.is_active() {
        unsafe { enable() }
    }
    r
}
# Act 1: Deref / Memory Mapped I/O Example

```rust
impl CBP {

    pub (crate) unsafe fn new() -> Self {
        CBP {
            _marker: PhantomData
        }
    }

    pub fn ptr() -> *const RegisterBlock { 0x8000_0000 as *const _ }
}

impl ops::Deref for CBP {
    type Target = RegisterBlock;

    fn deref(&self) -> &Self::Target {
        unsafe { &*Self::ptr() }
    }
}
```
# Act 1: Creating a new Project

- cargo new my_project
- Clone rust-embedded/cortex-m-quickstart
- cargo generate
# Act 1: Adding a HAL crate

- Hardware Abstraction Layer
- Some crates will Use the HAL...
- `fn new(spi: S) where S: spi::FullDuplex`
- Some crates will Impl the HAL...
- `impl spi::FullDuplex for TivaSPI {...}`
- Serial Ports, I2C, SPI, Timers, etc.
# Act 1: Running Embedded code on an OS

- Anyone can impl the Hal...
- impl spi::FullDuplex for LinuxDev {...}
- \#[cfg(feature)] macros
# Agenda

- Act 1 - Embedded Rust
- Act 2 - The Idea <--
- Act 3 - The Implementation
- Act 4 - Spiralling out of control
- Act 5 - The Demo
# Act 2: The Commodore 64

=================================

**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.
10 FOR X = 1 TO 5
20 PRINT "HELLO ACCU"
30 NEXT
RUN
HELLO ACCU
HELLO ACCU
HELLO ACCU
HELLO ACCU
HELLO ACCU
HELLO ACCU
# Act 2: Less is More

• For Sale: Baby shoes, never worn
Act 2: Goals for the project

- To distract me...
- Can you generate video with Rust?
- How much can you squeeze from one chip?
# Act 2: Candidate 1 - STM32F7 Discovery

- Cortex-M7 @ 216 MHz
- 1 MiB Flash
- 340 KiB SRAM
- Audio, Ethernet, SD/MMC
- Has a TFT controller...
- About £50
STM32F7 Discovery
# Act 2: Candidate 2 - Stellaris Launchpad

- Cortex-M4 @ 80 MHz
- 256 KiB Flash
- 32 KiB SRAM
- I2C, UART, SPI
- About £12
- There was one on my desk
# Act 2: Generating Analog Video

---

[Image: Monotron diagram showing horizontal and vertical blanking]
# Act 2: VGA Timing

- tinyvga.com/vga-timing
- 640 x 480 @ 60 Hz = 25.175 MHz
- 720 x 400 @ 70 Hz = 28.322 MHz
- 800 x 600 @ 60 Hz = 40.000 MHz
# Act 2: Rendering Mono/RGB Bitmaps

- Mono analog video

- RGB analog video
# Act 2: Text Mode

- A Font is a collection of tiny bitmaps
- Code Pages vs Unicode
- Rendering to a bitmap or in real-time
# Act 2: Text Attributes

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
# Agenda

* Act 1 - Embedded Rust
* Act 2 - The Idea
* Act 3 - The Implementation</br>  
* Act 4 - Spiralling out of control
* Act 5 - The Demo
for (ch, attr) in row.glyphs.iter() {  
  let index = (ch as isize) * 
  (MAX_FONT_HEIGHT as isize);  
  let w = unsafe { 
    *font_table.offset(index) \};  
  let rgb_addr = unsafe { 
    RGB_MAPS 
    .as_ptr() 
    .offset( 
      (attr.0 as isize) * 
      256_isize 
    ) + (w as isize)  
  } ;  
  let rgb_word = unsafe { *rgb_addr \};  
  hw.write_pixels(  
    rgb_word >> 16,  
    rgb_word >> 8,  
    rgb_word  
  );  
}
# Act 3: Implementation Failure...

- Fringing effect.
Act 3: Would you like to see a demo?
# Act 3: Serial Input

- Keyboards are tiny computers
- Talking to them is non-trivial
- So, I cheated...
# Act 3: Command Line Interface

- REPL?
- BASIC?
- Keep it simple...

```javascript
Item {
  item_type: ItemType::Callback(beep),
  command: "beep",
  help: Some("<freq> <len>"),
},

> beep
Error: Not enough arguments

> beep 440 60
Playing 440 Hz for 60 frames
```
Act 3: PS/2 Keyboard (fail!)

- Clock Signal (from Keyboard)
- Data Signal (bi-directional)
- Open-Collector (can hold clock low)
- Scan Codes, ugh!
- Interrupts @ 10 kHz are bad for video
Act 3: Joystick

--------------------
# Act 3: Memory Layout

---

*Application*

0x8000_8000

0x8000_2000

0x8000_1000

0x8000_0000

*data*

0x0004_0000

0x0003_8xxx

0x0002_xxx

0x0000_0200

0x0000_0000

*text*
# Act 3: Application Binary Interface

- 0x2000 - 0x2003: Pointer to init fn
- 0x2004 - 0x2FFF: Don’t care!
- Structure of function pointers
# Act 3: Application Binary Interface

- `putchar(char) -> int`
- `puts(const char*) -> int`
- `readc() -> int`
- `wfvbi()`
- `kbhit() -> int`
- `move_cursor(row, col)`
- `play(freq, chan, wave, vol) -> int`
- `change_font(font)`
- `get_joystick() -> u8`
# Act 3: Audio

---

- Square Wave Beeps

---

- PWM and Audio Filter

---

- Basic Tunes

---

- Three-channel wavetable synthesiser

---

- Tested on Linux with Pulse Audio
# Act 3: Storage Options

3.5 inch Floppy Disks

held 720 KiB or 1440 KiB of data.
# Act 3: Microsoft FAT Filesystems

Partition 1

Reserved
MBR

File A
Space
File A
Root Dir
FAT
VBR / BPB
# Act 3: SD Card

- Appear as an array of 512-byte blocks
- Can be partitioned (or not)
- Can work in SPI mode (slowly)

- Super cheap!

- GH: thejpster/embedded-sdmmc-rs
# Agenda

- Act 1 - Embedded Rust
- Act 2 - The Idea
- Act 3 - The Implementation
- Act 4 - Spiralling out of control
- Act 5 - The Demo
Act 4: Demo veroboard

----------------------------------
# Act 4: Designing a PCB

- Is...
- ...hard
- ...really time consuming
- ...an open-ended project
- ...quite good fun?
# Act 4: RS-232 Serial Port

- It's not a DB9!
- +/- 5V to 15V signalling
- RX / TX / GND
- RTS / CTS
- DTR / DSR
- RI / DCD

- Could hook up old Modems?
- Serial mice?
- Linux on Monotron!
# Act 4: MIDI Port

- Atari ST had one...
- MIDI is just a UART!
- 31,250 bps
- 5V signalling, opto-isolated
# Act 4: Real Time Clocks

- CMOS Batteries
- TM4C has one...
- ... but no coin cell input
- Crystal capacitance is fun
- Inter-Integrated Circuit / TWI
- Sec/Min/Hour/DOW/Day/Month/Year
# Act 4: Keyboards, revisited

- Can’t do 10 kHz data AND video
- How did IBM solve this?
- Intel i8042
- I could add an I/O processor!
# Act 4: AtMega 328

- 23 pins ...
- (If you include RST and XTAL1/2)
# Act 4: IEEE-1284 Parallel Port

---

<table>
<thead>
<tr>
<th>SEL</th>
<th>PE</th>
<th>BUSY</th>
<th>ACK</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>DO</th>
<th>STRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Act 4: Inventing a programming language

- BASIC
- Python
- Javascript
- Pascal
- REXX
- Euphoria
# Act 4: Monotronian

```javascript
01 fn main(args)
02    len = length(args)
03    for x = 1 to len
04        if args[x] == "--help"
05            print_help()
06            return
07        elif args[x] == "--verbose"
08            verbose = verbose + 1
09        else
10            process_file(args[x])
11        endif
12    endfor
13 endfn
```
# Act 4: Closing Thoughts

- github.com/the.jpster
- keybase.io/the.jpster
- Come say hi!
- (I have Rust Embedded flyers)
- Think about how you write code
# Agenda

- Act 1 – Embedded Rust
- Act 2 – The Idea
- Act 3 – The Implementation
- Act 4 – Spiralling out of control
- Act 5 – The Demo <--