CS452 F23 Lecture Notes

Lecture 02 - 12 Sep 2023

1. Raspberry Pi

1.1. Booting

- RPi's bootloader boots u-boot via the network
  - does some initialization, e.g, configures GPIO and UART for console output
- you load your image (tftpboot), call "go 0x200000"
  - this is essentially a call to the entry point of your code (kmain), located at 0x200000
  - you are on u-boot's stack

1.2. Memory

- ARM processor sees "low peripheral" address space (see BCM, Figure 1)
- VC memory (frame buffers) - about 76MB
- u-boot text, data, stack just below VC memory
- Your code and data should live in the lowest ARM memory segment, below the VC-reserved memory and u-boot
- Peripheral control registers mapped in range 0xFC000000 to 0xFF800000
  - BCM shows these as "legacy master" addresses: 0x7C000000 instead of 0xFC000000
  - See descriptions of each peripheral in BCM
    - example: UART0 registers mapped starting at 0xFE201000

1.3. Timers

- system time (BCM Ch 10) vs. ARM-side timers (Ch 12)
- we want system time - free-running, constant 1 MHz rate
- no time of day, just a free running counter
- read value of counter, or trigger interrupts via "compare" registers
  - for A0, no interrupts

1.4. GPIO amd serial hat

- GPIO (BCM Ch 5) allows function of BCM2711 pins to be controlled
- what we care about: UART0 and UART3 signals (TX, RX, CTS) need to be routed to pins expected by the serial hat
  - u-boot configures UART0 - line 1
  - $gpio_{init}()$ function in sample code (iotest) configures GPIO for UART3 - line 2
- no reason to mess with this

1.5. UART

- send FIFO (8 entries), receive FIFO (12 entries)
  - can enable/disable FIFO
  - if disabled, acts as if FIFO lengths are 1
- data register (DR)
- to write, write data into the DR. UART moves data into FIFO and starts transmitting
  - how long does it take to transmit a byte to the console? to the marklin controller?
- to read, reading from the DR gets the next item from receive FIFO
- what if transmit FIFO is full? What if receive FIFO is empty?
  - FR (flag register) flags indicating FIFO full, FIFO empty

1.6. Flow Control

- sending rate limited by how quickly RPi can write into DR, and how quickly UART can transmit
  - 2400 baud = 240 bytes/second
- serial line has two flow control signals (CTS, RTS) in addition to TX and RX
  - CTS is asserted by the Marklin/Console to indicate that it is ready to receive data
  - RTS is asserted by the RPi to indicate that it is ready to receive data
- flow control idea:
  - RPi should stop sending when CTS is deasserted
  - remote (marklin) should stop sending when RTS is asserted
- RPi UARTs support “hardware flow control”, if enabled in the UART control register
  - RPi will only send data from the transmit FIFO if CTS is asserted
  - RPI will deassert RTS when receive FIFO is full
- In practice
  - Marklin does not support flow control for sending (so RPi’s RTS status doesn’t matter)
    - should not matter, since RPi much faster
  - Marklin behavior
    - receive command
    - deassert CTS
    - process command
    - reassert CTS
    - repeat
  - problem: Marklin might not deassert CTS quickly enough after receiving command
- Problem Scenario (FIFO enabled)
  - RPi write command (two bytes) to DR, waits for FIFO empty before writing next command
  - UART sends the command (takes 8-10ms), FIFO now empty
  - race:
    - RPi writes next command into DR, starts transmitting
    - Marklin lowers CTS
  - if the RPi wins this race, it will start sending the next command before the Marklin is finished with the previous one (bad!)
    - and the RPi is fast…
- Solution for A0
  - RPi waits after sending a command
  - wait long enough for Marklin to both receive and process the command

Author: Ken Salem
Created: 2023-09-12 Tue 10:29