Bare Metal GPIO
Topics

1) How does the BeagleBone’s CPU do GPIO?
2) How can we access registers?
TI AM335x Sitara GPIO Basics
Data Sheets

- BeagleBone Green Schematic  
  - (for BBG wiring)
- BeagleBone Green SRM [system reference manual]  
  - (for BBG hardware/system description)
- ..  
  - (for library description)
- ..  
  - (for all CPU’s sub-systems and registers)
- AM3359 Data Sheet  
  - (for pin MUXs)
TI AM335x Sitara GPIO

- GPIO0 - GPIO3 modules
  - ..

- Pins are physical connections to the CPU
  - each pin can be input, output, and generate interrupt.
  - ..
  - Example: BBG’s User Push button is 
    lcd_data2, gpmc_a2, 
    pr1_mii0_txd3, ehrpwm2_trizone_input, 
    pr1_pru1_pru_r30_2, pr1_pru1_pru_r31_2, 
    gpio2_8
TI AM335x Sitara GPIO

• Each GPIO module has memory-mapped registers
  – Changing the values in these control registers changes the voltages on the pins.

• 2 ways to driving pins:
  – ..
    Write bit to 1 for on, 0 for off.
    • Use bit-twiddling to change bits.
  – ..
    • Write 1 to bit in SET register to turn on.
    • Write 1 to bit in CLEAR register to turn off.
    • (Writing 0 has no effect.)
1. Initialize GPIO module
   - Enable clocks on GPIO modules
   - PIN muxing (defaults to I/O: "mode 7", listed last on schematic)
   - Enable GPIO modules
   - Module reset

2. Set pin direction:
   - GPIO_OE: ..
     - Set bit to 1 for input, 0 for output.
3a. Read:
- GPIO_DATAIN: read values from pins
- Pseudocode:
  unsigned int val = PortA’s GPIO_DATAIN

3b. Write: (ex on next slide)
- GPIO_DATAOUT: standard register for driving pins.
- GPIO_SETDATAOUT/GPIO_CLEARDATAOUT: set & clear semantics on output.
GPIO Write Example

- **Ex: Drive pins with bit-twiddling**
  - Write value to port’s GPIO_DATAOUT register.
    
Pseudocode to turn on LED1:
    PortA GPIO_DATAOUT |= (1 << LED1_PIN);

- **Ex: Drive pins with set and clear registers**
  - To turn on a bit, write to GPIO_SETDATAOUT reg.
    
Pseudocode to turn on LED1:
    PortA GPIO_SETDATAOUT = (1 << LED1_PIN);
  - To turn off a bit, write to GPIO_CLEARDATAOUT reg.
    
Pseudocode to turn off LED1:
    PortA GPIO_CLEARDATAOUT = (1 << LED1_PIN);
Register Access
Each GPIO module has ..
to control it’s direction, input/output, etc.

Each GPIO module has ..

Each register has an..

Specific register’s address is..
Register Access in C

// Defined by StarterWare
#define SOC_GPIO_1_REGS     (0x4804C000)
    // ..(from TI Datasheet - Sec 2: Memory Map - p181)
    // “SOC” = System on Chip

#define GPIO_DATAOUT        (0x13C)
    // ..(from TI Datasheet - Sec 25.4 GPIO Registers - p4871)

    // How to access a memory mapped register
#define HWREG(x)            (*((volatile unsigned int *)(x)))

    // User Provided:
#define LED1_PIN            (22)

    // Use in code:
HWREG(SOC_GPIO_1_REGS + GPIO_DATAOUT) |= (1 << LED1_PIN);
C Data Types for Bits

- C Guarantees
  - Size of: char <= short <= int <= long <= long long
    (Our GCC has int at 32 bits)

- What if you *need* 32 bits
  - Ex: for 32 GPIO pin states
  - Use the stdint types:
    ```c
    #include <stdint.h>
    uint32_t pins;  // unsigned, at least 32 bits.
    int8_t distance;  // signed
    ```
  - These guarantee to be at least the size indicated.
Debouncing a Read

• Button (and such) inputs often “bounce”
  – The value will bounce between a 0 and a 1 for a little before settling to be the desired value.

• Debounce so you only..
  – Ex: Require the same value for 3 reads in a row before accepting it as a debounced value.
Debouncing a Read

- Debouncing possible to be done by hardware
  - Set pin to input
  - Set debounce time in GPIO_DEBOUNCINGTIME (# 32kHz clock cycles to debounce for)
  - Debouncing time = (DEBOUNCETIME + 1) x 31 us
  - Set bit in GPIO_DEBOUNCENABLE to turn on.