I^2C, PWM and Hardware Interfacing with the Beagleboard

David Baty, Mark Jacobson and Tom Most

Overview

- Hardware Interfacing
  - Expansion Connector
  - Powering the Beagle
  - Level Shifting
  - I^2C Devices
- I^2C
  - Enabling I2C2 on the Beagle
  - Basic access methods
- PWM
  - Configuring the mux
  - Overview of registers and technique
  - Demonstration

Beagleboard Expansion Connector (J3)

- Easy to access connector for breadboarding, etc
- Contains a variety of signals
  - 1.8v and 5v
  - I2C, UART, MMC, McBSP, GPIO
  - **ALL I/O is 1.8v**
- Beagle can be powered solely with 5v pin
- Desired signals must be muxed correctly to physical pins
- Smaller expansion headers expose more signals
  - Harder to interface with
  - Typically used for LCD interface

Level Shifting

- Level shifting will be required for most applications
- MAX3378 -16Mbps Quad Bi-directional Level Shifter
  - Confirmed working
  - Used on several commercial expansion boards
  - Difficult to breadboard
- PCA9306 - Level shifter for I2C
- Single MOSFET Bi-Directional level shifter with BSS88 or BSS138 (NXP Appnote AN97055)

Expansion Board Design

- DXF available on wiki to align mounting holes and expansion headers
- Suggested Components
  - 5v power supply (TPS5420 or other switcher)
  - 5v I2C header
  - 5v GPIO header
  - I/O Expander for switches and LEDs

- See "BeagleBoard Hardware Interfacing" on eLinux wiki for more information

Confirmed Functional I^2C Devices

- AD7991: 4 channel 12-bit ADC
  - Good default configuration
  - Auto sequencing
- ADXL345: 3 axis accelerometer
  - Can used directly with 1.8V I2C with proper supply
  - Sparkfun sells a version for breadboards/prototypes
- PCA8574AD: 8-bit IO Expander
  - Kernel module allows for transparent use
  - Can drive LEDs
  - 3.3V and 5V

I^2C Overview

- Off-chip bus for communicating with low-speed peripherals
  - Common Applications: I/O expanders, Analog-to-digital converters, Miscellaneous sensors
  - 100kHz, 400kHz, 3.4Mhz modes available
- Simple to use, requires only two pins
  - SCL - Clock
  - SDA - Data
- Pull-up resistors used to idle the bus high
- 7-bit address and a read/write bit

Enabling I2C2 on the Beagleboard

- Three I^2C buses available, used to control a variety of on-board peripherals, including the DVI display driver and power sequencing
- I2C2 pinned out to 0.1" spaced expansion header
  - Disabled by default due to lack of pull-up resistors
  - Can be enabled by recompiling the kernel with "Beagleboard I2C2" selected in the configuration menu
  - Appears as /dev/i2c-2 when enabled

Communication with I^2C Peripherals

- Accomplished using standard syscalls
  - open - returns a file handle passed to the next three
  - ioctl
  - read - returns number of bytes actually read
  - write - returns number of bytes actually written
- Errors can be checked via errno
- See "Interfacing with I2C Devices" on eLinux wiki for more details

Configuring the Beagleboard MUX

1. Pick pin to mux and what it needs to be set to (look in BBSRM 8.19). Note the ball that the pin is connected to (e. g. ball AB25 for expansion header pin 10).
2. Figure out the kernel name for that ball. Search for it in .../arch/arm/mach-omap2/mux34xx.c in the omap3_cbb_ball array (this is the right package for the Beagle). For AB25, this is GPT10_PWMEVT.
3. Add an entry for it in .../arch/arm/mach-omap2/board-omap3beagle.c in the board_mux struct with the mode:

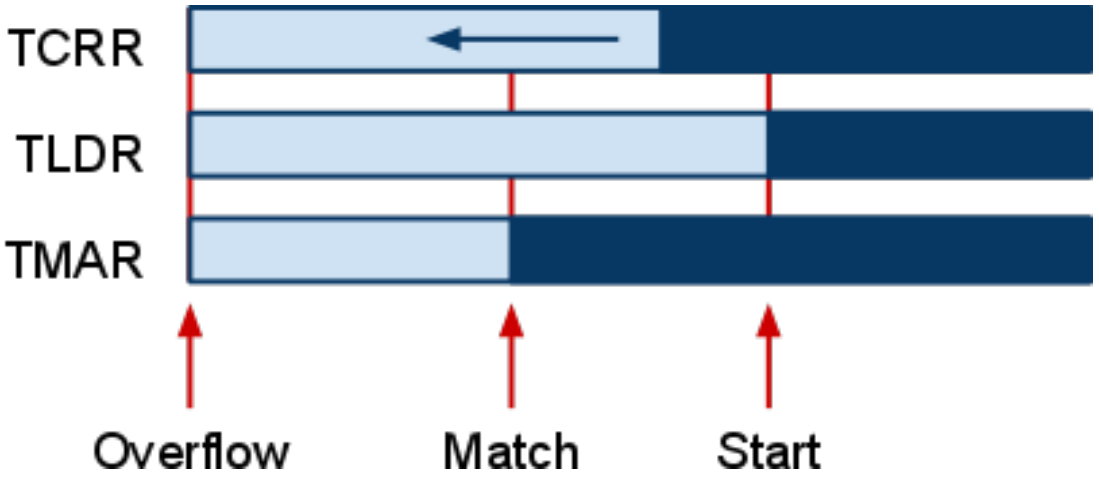
```
static struct omap_board_mux board_mux[] __initdata = {
    OMAP3_MUX(UART2_RTS, OMAP_MUX_MODE2|OMAP_PIN_OUTPUT),
    { .reg_offset = OMAP_MUX_TERMINATOR },
};
```

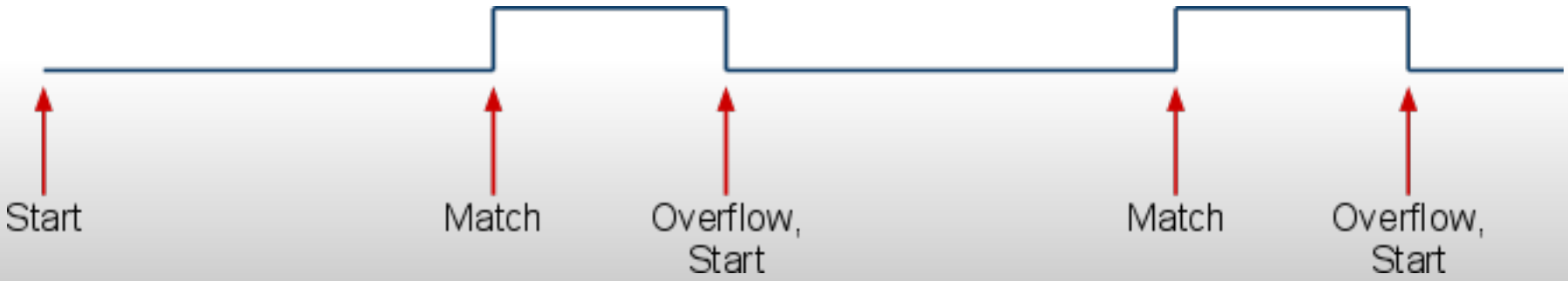
PWM Overview

- OMAP3530 has 12 timers that can be used for PWM.
- 3 pinned out on the Beagle (exp. header 4, 6, and 10)
  - 2 can be connected to the 13 MHz system clock (10 and 11, exp. header pins 10 and 6)
  - The rest use a 32 kHz clock
- Each timer is a bunch of 32-bit registers
  - TCLR Control register
  - TCRR Counter
  - TLDR Load register (fills TCRR when it overflows)
  - TMAR Compare value
- To get PWM, set the timer to count endlessly. When it equals TMAR or overflows, toggle the output.

PWM Continued

- Frequency is (0xffffffff - TLDR) * clock_frequency
- Duty cycle is (TMAR - TLDR) / (0xffffffff - TLDR)





PWM Demonstation

- By using mmap() we can gain direct access to the timer registers
- This allows trivial control of hobby servomotors, which take a 50 Hz PWM signal:

Duty cycle	**Effect**
0.050	Full speed in one direction
0.075	No movement
0.100	Full speed in opposite direction

Questions?

Further Reading

- http://elinux.org/Interfacing_with_I2C_Devices
- http://elinux.org/BeagleBoard_Hardware_Interfacing
- http://elinux.org/BeagleBoardPWM