EECS192 Discussion Week 7
Programmable Real Time Unit and Linescan Camera

- Camera Connections
- PRU overview
- PRU <-> Linux interface
- PRU debugging
Beaglebone Blue Connections
TSL1401 line camera and BeagleBoneBlue Interface

Note AIN < 1.8 V!
Directions: pru_firmware and LineCamera

https://github.com/ucb-ee192/SkeletonBeagle/README.md

Skeleton code for reading TSL1401 Line Sensor on Beagle Bone Blue for EE192
Using PRU0 for Fast A/D read (about 8.2 us) Feb. 27, 2019.

1. Clone whole repo. git clone https://github.com/ucb-ee192/SkeletonBeagle
2. Compile and install PRU code
   i) cd pru_firmware
   ii) make all
   iii) sudo make install
3. compile and run LineCamera
   i) cd LineCamera
   ii) make
   iii) sudo ./LineCamera (must be root as accesses /dev/mem, etc)

This reads TSL1401 through shared memory buffer and saves to file linescans.csv
LEDs USR0...USR3 will flash when reading A/D. This particular .csv format is compatible
with Python linescanplot.py

Notes:
• The PRU0 has direct access to GPIO addresses without any device driver protection.
• UART1 RX is used as CLK. UART1 TX is used as SI. These GPIO pins are setup
  using rc_pinmux and rc_gpio_init, and should avoid Linux conflicts.
• The analog output from the line camera is in the range 0...3.3 V. A voltage divider is
  needed to keep the input voltage less than 1.8V on the A/D input.
• See LineCamera/TSL1401-BBBBlue-interface.png for wiring connections.
**GPIO Access Linux vs PRU**

<table>
<thead>
<tr>
<th>Use on Blue</th>
<th>config-pin name</th>
<th>Ball</th>
<th>MODE0</th>
<th>MODE1</th>
<th>MODE2</th>
<th>MODE3</th>
<th>MODE4</th>
<th>MODE5</th>
<th>MODE6</th>
<th>MODE7</th>
</tr>
</thead>
<tbody>
<tr>
<td>UARTs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UART1_TX</td>
<td>P9.24</td>
<td>D15</td>
<td>uart1_txd</td>
<td>mmc2_sdwp</td>
<td>dcan1_rx</td>
<td>I2C1_SCL</td>
<td>NC</td>
<td>pr1_uart0_txd</td>
<td>pr1_pru0_pru_r31_16</td>
<td>gpio0[15]</td>
</tr>
<tr>
<td>UART1_RX</td>
<td>P9.26</td>
<td>D16</td>
<td>uart1_rxd</td>
<td>mmc1_sdwp</td>
<td>dcan1_tx</td>
<td>I2C1_SDA</td>
<td>NC</td>
<td>pr1_uart0_rxd</td>
<td>pr1_pru1_pru_r31_16</td>
<td>gpio0[14]</td>
</tr>
</tbody>
</table>

**Linux** GPIO lines need to be set to mode 7 then output direction:

```c
int pin1 = UART1_HEADER_PIN_3;  // < P9_26, normally UART1 RX
rc_pinmux_set(pin1, PINMUX_GPIO);
rct_gpio_init (CHIP, pin1, GPIOHANDLE_REQUEST_OUTPUT);
```

**PRU** direct bit/register access

```c
#define GPIO0 0x44e07000
#define GPIO_CLEARDATAOUT 0x190
#define GPIO_SETDATAOUT 0x194
#define CLK (1 << 14)  // gpio[14] UART1_RX
#define SI (1 << 15)  // gpio[15] UART1_TX
*GPIO0_CLEAR = CLK;  // reset CLK line
*GPIO0_SET = CLK;
```
Programmable Real-time Unit

Used on BeagleBone Blue for RC servo and quad encoders
(hopefully real-time A/D for line camera)
### Shared 12 kb Address Space

<table>
<thead>
<tr>
<th><strong>PRU0</strong></th>
<th><strong>LINUX</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1_0000</td>
<td>For encoder</td>
</tr>
<tr>
<td></td>
<td><code>rc_pru_shared_mem_ptr();</code></td>
</tr>
<tr>
<td></td>
<td><code>shared_mem_32bit_ptr[16+1]</code></td>
</tr>
<tr>
<td>Flag = 1 to start conversion</td>
<td><code>shared_mem_32bit_ptr[16+2]</code></td>
</tr>
<tr>
<td>Pixel 0</td>
<td><code>shared_mem_32bit_ptr[16+2+127]</code></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Pixel 127</td>
<td></td>
</tr>
</tbody>
</table>

#### Linux LineCamera.c line 116:
```
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 1;
// set flag to start conversion by PRU
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 1);
```

#### PRU main_pru0.c line 117:
```
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 0);
// loop until command
//... read 128 pixels ...
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 0;
// reset to zero
```
Linescan telemetry file format

Caution with .csv format: spaces also act as delimiters

linescans.20190302-235143.csv

time (us)   linescan_near   velocity
2957

5857
Display Linescan file

On laptop or desktop: python linescanplot.py

Note: Variable timing from Linux causes overexposure
Delay measurement using rc library

Note: Variable timing from Linux causes overexposure

```
Linux LineCamera.c

start_time = rc_nanos_since_boot();

for(j = 0; j< 1000; j++)
    {shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 1;
     // set flag to start conversion by PRU
     while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 1);
     // wait for PRU to zero word
     for(i = 0; i< 128; i++){
        linescan[i]=
            (int) shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+2+i];
        // copy data
    }
}

end_time = rc_nanos_since_boot();
run_time = end_time - start_time;
```
A to D using **PRU**

```c
#define ENCODER_MEM_OFFSET 16
#define PRU_SHARED_MEM 0x10000 // shared memory with Cortex A8
#define GPIO0 0x44e07000
#define GPIO_CLEARDATAOUT 0x190
#define GPIO_SETDATAOUT 0x194
#define CLK (1 << 14) // gpio[14] = TSL1401 clock, UART1_RX
#define SI (1 << 15) // gpio[15] = start integration, UART1_TX

// access shared memory with Linux:
shared_mem_32bit_ptr = (volatile unsigned int*) PRU_SHARED_MEM;

// signal to Linux process that PRU is ready by clear []
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET] = 0x0;
...

for (i = 0; i < 130; i++)
{
    *GPIO0_CLEAR = CLK; // reset CLK line
    shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+2+i] = read_adc(0);
    *GPIO0_SET = CLK; // rising edge of CLK gives next pixel
    __delay_cycles(200); // allow hold time - 1 us
}

Need at least 129 clocks for proper operation
```
PRU Debugging details
(for reference)
Debugging PRU

1. git clone https://github.com/RRvW/prudebug-rl
2. missing readline.h
   sudo apt-get install libreadline-dev
3. make
4. sudo ./prudebug

https://markayoder.github.io/PRUCookbook/04debug/debug.html

Assembly code: build/main_pru0.lst
Symbol table: file.map

<table>
<thead>
<tr>
<th>page</th>
<th>address</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000200</td>
<td>main</td>
</tr>
<tr>
<td>0</td>
<td>00000324</td>
<td>read_adc</td>
</tr>
</tbody>
</table>

Maybe need Beaglebone reset if get:
ERROR: failed to run rc_encoder_pru_init
main_pru0.c line 117:

116    while(1)
117    {while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1]==0);
119    *GPIO0_SET = SI ; // set SI

main_pru0.lst line 697:

697; 117 | { while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 0); // loop
698; | until command to read 128 times
701;--------------------------------------------------------------------------------
702 00000078 000000F1002081          LBBO      &r1, r0, 0, 4         ; [ALU_PRU] |117| $O$v1
703 0000007c 0000005700E1FF          QBEQ      ||$C$L6||, r1, 0x00   ; [ALU_PRU] |117|
706

**PRU Instruction offset address = 0x200 (from file.map) +0x78 = 0x278**
**Since 4 bytes per instruction, PRU instruction memory address = 0x278/4 = 0x9e**

```
PRU0> dis 0x9e
Absolute addr = 0xd09e, offset = 0x009e, Len = 16
[0x009e] 0xf1002081 >> LBBO R1, R0, 0, 4
[0x009f] 0x5700e1ff       QBEQ     ||$C$L6||, R1, 0x00   ; [ALU_PRU] |117|
[0x00a0] 0xe1043297 SBBO R23, R18, 4, 4
```
Prudebug main_pru0.c

PRU0> dis 0x9e
Absolute addr = 0xd09e, offset = 0x009e, Len = 16
[0x009e] 0xf1002081 >> LBBO R1, R0, 0, 4
[0x009f] 0x5700e1ff QBEQ -1, R1, 0
[0x00a0] 0xe1043297 SBBO R23, R18, 4, 4

Breakpoint after while()

PRU0> br 1 0xa0
PRU0> gss
Running (will run until a breakpoint is hit or a key is pressed)....

Now linecamera will wait for gss to read next frame

Errata: LineCamera.c line 64:
if(rc_encoder_pru_init()) {...}

PRU needs to be running, e.g. reset flag to pass this init.