08-1 Talking to the PRU

ARM to PRU communication via mmap()

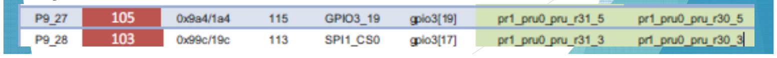
Overview

- Review PRU code
- Modify to read from PRU0 Data RAM
- Use mmap() to modify values

Programming the PRU - c

- You can use either C or assembly, Let's do both

```
void main(void){
…
    while(!(__R31&(1<<3))) {
        __R30 |= 1<<5;
        __delay_cycles(TIME);
        __R30 &= ~(1<<5);
        __delay_cycles(TIME);
    }
    __delay_cycles(TIME);  // Give some time for press to release
    // Call assembly language
    start(TIME);
    __halt();
}
```

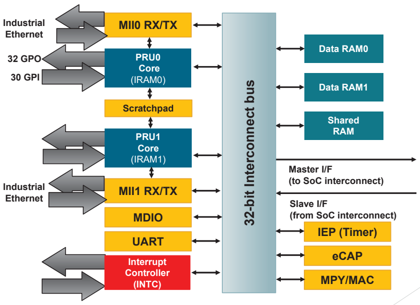


P9_27	105	0x9a4/1a4	115	GPIO3_19	gpio3[19]	pr1_pru0_pru_r31_5	pr1_pru0_pru_r30_5
P9_28	103	0x99c/19c	113	SPI1_CS0	gpio3[17]	pr1_pru0_pru_r31_3	pr1_pru0_pru_r30_3

Programming the PRU - assembly

```
start:
    set     r30, r30.t5     ; turn on the output pin (LED on)
    mov     r0, r14         ; store the length of the delay in REG0
delayon:
    sub     r0, r0, 1        ; Decrement REG0 by 1
    qbne delayon, r0, 0      ; Loop to DELAYON, unless REG0=0
ledoff:
    clr     r30, r30.t5      ; clear the output bin (LED off)
    mov     r0, r14         ; Reset REG0 to the length of the delay
delayoff:
    sub     r0, r0, 1        ; decrement REG0 by 1
    qbne delayoff, r0, 0     ; Loop to DELAYOFF, unless REG0=0
    qbbc start, r31, 3       ; is the button pressed? If not, loop
end:
    jmp     r3               ; r3 contains the return address
```

Data RAM (8KB)



Data RAM address

- From: **AM335x PRU-ICSS Reference Guide**

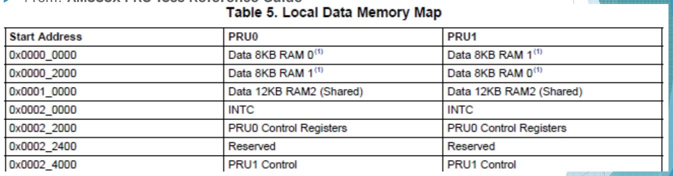


Table 5. Local Data Memory Map

Start Address	PRU0	PRU1
0x0000_0000	Data 8KB RAM 0[1]	Data 8KB RAM 1[1]
0x0000_2000	Data 8KB RAM 1[1]	Data 8KB RAM 0[1]
0x0001_0000	Data 12KB RAM2 (Shared)	Data 12KB RAM2 (Shared)
0x0002_0000	INTC	INTC
0x0002_2000	PRU0 Control Registers	PRU0 Control Registers
0x0002_2400	Reserved	Reserved
0x0002_4000	PRU1 Control	PRU1 Control

Data RAM address, free

Set in install.sh

- In the **Makefile** you find:

```
LINKER_COMMAND_FILE=./AM335x_PRU.cmd
LIBS=--library=$(PRU_SUPPORT)/lib/rpmsg_lib.lib
INCLUDE=--include_path=$(PRU_SUPPORT)/include –
         include_path=$(PRU_SUPPORT)/include/am335x
STACK_SIZE=0x100
HEAP_SIZE=0x100
```

- **The compiler uses the first 0x200 bytes of RAM**

```
bone$ ~~./setup.sh~~
bone$ source setup.sh
```

Changes - C

#define pwmParams   0x200

```
#include "pwmParams.h"

volatile register unsigned int __R30;
volatile register unsigned int __R31;
int *RAM0  = (int *) pwmParams;
int i;
int time;
```

Points to Data RAM0

Changes - C

Must be in loop

Must be passed a constant

```
while(!(__R31&(1<<3))) {
    time = RAM0[0];   // On time
    for(i=0; i<time; i++) {
        __R30 |= 1<<5;
        // __delay_cycles must be passed a const, so we have to do our own loop
        // Must have something in loop, otherwise it optimized out.
    }
    time = RAM0[1];
    for(i=0; i<time; i++) {
        __R30 &= ~(1<<5);
    }
}
__delay_cycles(TIME);// Give some time for press to release
// Call assembly language
start(RAM0);
```

Changes - assembly

Load Bye Burst, Offset

Number of bytes

```
start: (Old)
    set     r30, r30.t5      ; turn on the output pin (LED on)
    mov     r0, r14          ; store the length of the delay in REG0
delayon:
    sub     r0, r0, 1        ; Decrement REG0 by 1
    qbne    delayon, r0, 0   ; Loop to DELAYON, unless REG0=0

start: (New)
    lbbo    &r0, r14, 0, 4    ; Load the length of the delay in r0
    set     r30, r30.t5      ; turn on the output pin (LED on)

delayon:
    sub     r0, r0, 1        ; Decrement REG0 by 1
    qbne    delayon, r0, 0   ; Loop to DELAYON, unless REG0=0
```

Changes 2

Offset

```
ledoff: (Old)
    clr     r30, r30.t5          ear the output bin (LED off)
    mov     r0, r14              et REG0 to the length of the delay
delayoff:
    sub     r0, r0, 1            decrement REG0 by 1
    qbne delayoff, r0, 0         ; Loop to DELAYOFF, unless REG0=0
    qbbc start, r31, 3           ; is the button pressed? If not, loop

ledoff: (New)
    clr     r30, r30.t5      ; clear the output bin (LED off)
    lbbo    &r0, r14, 4, 4   ; Load the length of the delay in r0

delayoff:
    sub     r0, r0, 1        ; decrement REG0 by 1
    qbne    delayoff, r0, 0  ; Loop to DELAYOFF, unless REG0=0

    qbbc    start, r31, 3    ; is the button pressed? If not, loop
```

On the ARM side

- Memory is shared between the PRU and the ARM
- From: **AM335x Sitara™ Processors Technical Reference Manual**

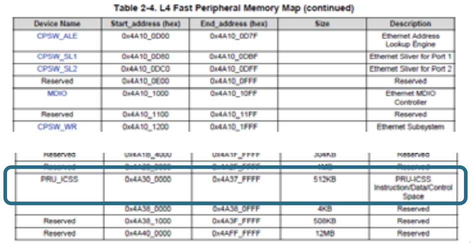


Table 2-4. L4 Fast Peripheral Memory Map (continued)

Device Name	Start_address (hex)	End_address (hex)	Size	Description
CPSW_ALE	0x4A10_0D00	0x4A10_0D7F		Ethernet Address Lookup Engine
CPSW_SL1	0x4A10_0D80	0x4A10_0DBF		Ethernet Sliver for Port 1
CPSW_SL2	0x4A10_0DC0	0x4A10_0DFF		Ethernet Sliver for Port 2
Reserved	0x4A10_0E00	0x4A10_0FFF		Reserved
MDIO	0x4A10_1000	0x4A10_10FF		Ethernet MDIO Controller
Reserved	0x4A10_1100	0x4A10_11FF		Reserved
CPSW_WR	0x4A10_1200	0x4A10_1FFF		Ethernet Subsystem
Reserved	0x4A18_4000	0x4A1F_FFFF	304KB	Reserved
Reserved	0x4A20_0000	0x4A2F_FFFF	1MB	Reserved
PRU_ICSS	0x4A30_0000	0x4A37_FFFF	512KB	PRU-ICSS Instruction/Data/Control Space
	0x4A38_0000	0x4A38_0FFF	4KB	Reserved
Reserved	0x4A38_1000	0x4A3F_FFFF	508KB	Reserved
Reserved	0x4A40_0000	0x4AFF_FFFF	12MB	Reserved

On the ARM side

```
#define PRU_ADDR       0x4A300000         // Start of PRU memory Page 184 am335x TRM
#define PRU_LEN        0x80000            // Length of PRU memory
#define PRU0_DRAM      0x00000            // Offset to DRAM
#define PRU1_DRAM      0x02000
#define PRU_SHAREDMEM  0x10000            // Offset to shared memory

unsigned int  *pru0DRAM_32int_ptr;        // Points to the start of local DRAM
unsigned int  *pru1DRAM_32int_ptr;        // Points to the start of local DRAM
unsigned int  *prusharedMem_32int_ptr;    // Points to the start of the shared memory
```

Data RAM address

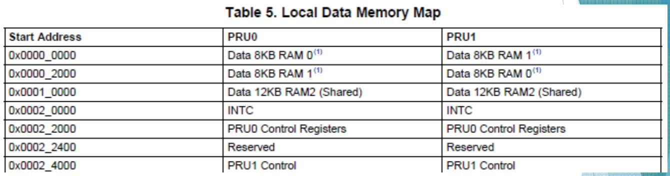


Table 5. Local Data Memory Map

Start Address	PRU0	PRU1
0x0000_0000	Data 8KB RAM 0[(1)]	Data 8KB RAM 1[(1)]
0x0000_2000	Data 8KB RAM 1[(1)]	Data 8KB RAM 0[(1)]
0x0001_0000	Data 12KB RAM2 (Shared)	Data 12KB RAM2 (Shared)
0x0002_0000	INTC	INTC
0x0002_2000	PRU0 Control Registers	PRU0 Control Registers
0x0002_2400	Reserved	Reserved
0x0002_4000	PRU1 Control	PRU1 Control

mmap()

```
#include "pwmParams.h“
fd = open ("/dev/mem", O_RDWR | O_SYNC);
if (fd == -1) {
    printf ("ERROR: could not open /dev/mem.\n\n");
    return 1;
}
pru = mmap (0, PRU_LEN, PROT_READ | PROT_WRITE, MAP_SHARED, fd, PRU_ADDR);
if (pru == MAP_FAILED) {
    printf ("ERROR: could not map memory.\n\n");
    return 1;
}
close(fd);
printf ("Using /dev/mem.\n");

pru0DRAM_32int_ptr =     pru + PRU0_DRAM/4 + pwmParams/4;   // Points to PRU0 memory
pru1DRAM_32int_ptr =     pru + PRU1_DRAM/4 + pwmParams/4;   // Points to PRU1 memory
prusharedMem_32int_ptr = pru + PRU_SHAREDMEM/4;// Points to start of shared memory
```

word address

Convert byte address to word address

HEAP + STACK

mmap()

```
int ch =0;    // We only have channel 0
pru0DRAM_32int_ptr[2*(ch)+0] = onCount;     // On time
pru0DRAM_32int_ptr[2*(ch)+1] = offCount;    // Off time

if(munmap(pru, PRU_LEN)) {
   printf("munmap failed\n");
} else {
   printf("munmap succeeded\n");
}
```

Running it

- Setup PRU: http://elinux.org/EBC_Exercise_30_PRU_via_remoteproc_and_RPMsg

```
bone$ cd exercises/pru/examples/pwm1
bone$ git pull
bone$ ./install.sh
bone$ source setup.sh
bone$ make && make install
bone$ ./pwm-test onCount offCount
```