# 07-2 Talking to the PRU

ARM to PRU communication via mmap()

# Overview

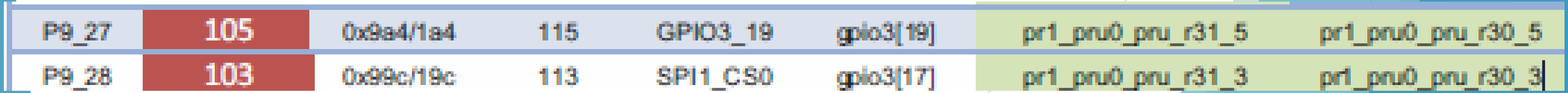
- Review PRU code
- Modify to read from PRU Data RAM
- Use mmap() to modify values

# Programming the PRU - c

```
#include <stdint.h>
#include <pru_cfg.h>
#include "resource_table_empty.h"

volatile register uint32_t __R30;
volatile register uint32_t __R31;
```

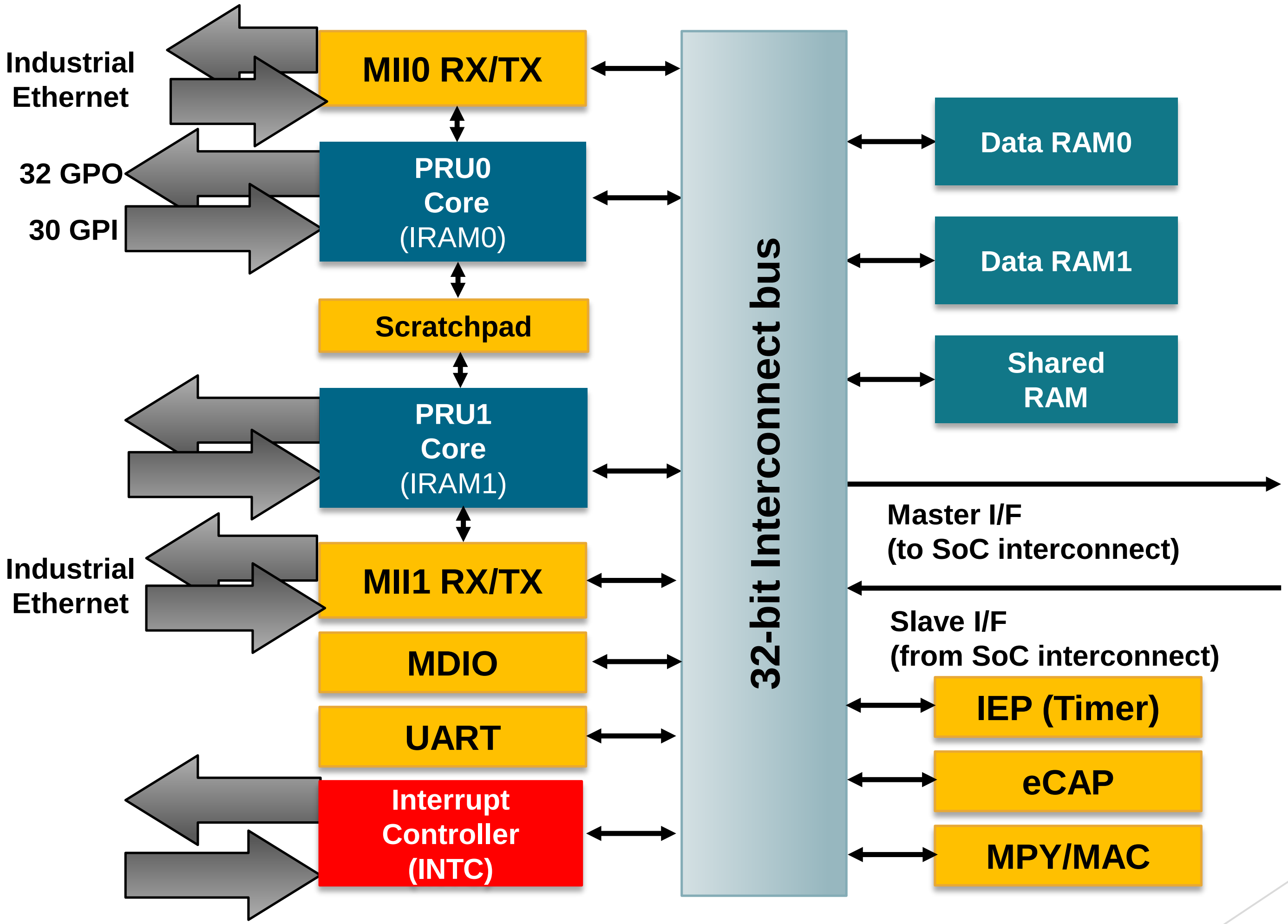
# Programming the PRU - c

```
void main(void)
{
    uint32_t gpio;
    /* Clear SYSCFG[STANDBY_INIT] to enable OCP master port */
    CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;

    gpio = 0x1<<5; // Select which pin to toggle.

    while (1) {
        __R30 |= gpio;     // Set the GPIO pin to 1
        __delay_cycles(100000000);
        __R30 &= ~gpio;        // Clearn the GPIO pin
        __delay_cycles(100000000);
    }
```



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

# Data RAM (8KB)


Industrial Ethernet
MII0 RX/TX
32 GPO
30 GPI
PRU0 Core (IRAM0)
Scratchpad
PRU1 Core (IRAM1)
Industrial Ethernet
MII1 RX/TX
MDIO
UART
Interrupt Controller (INTC)
32-bit Interconnect bus
Data RAM0
Data RAM1
Shared RAM
Master I/F (to SoC interconnect)
Slave I/F (from SoC interconnect)
IEP (Timer)
eCAP
MPY/MAC

# Data RAM address

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

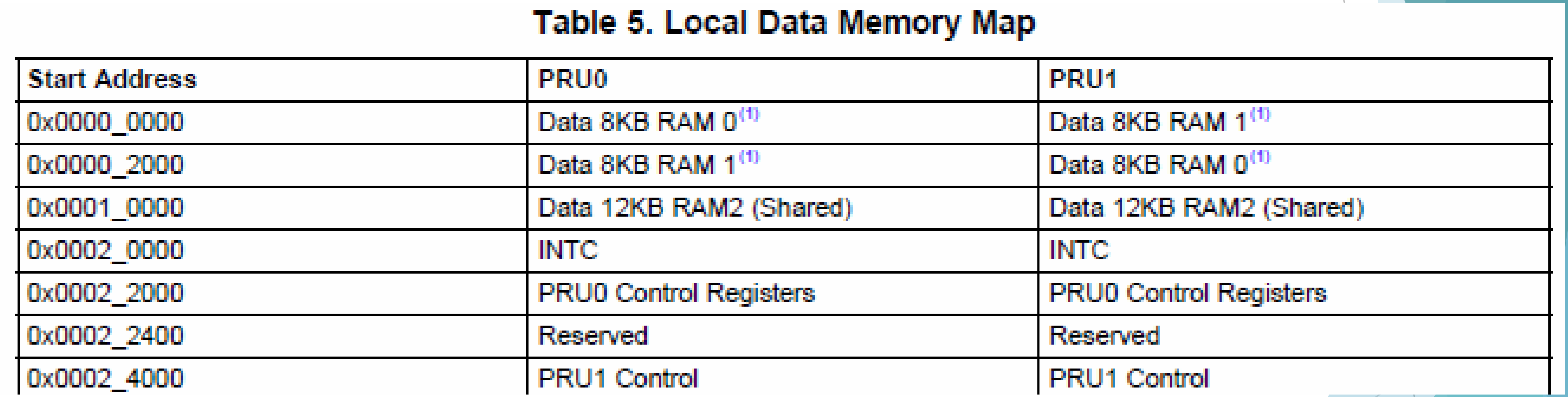


**Table 5. Local Data Memory Map**

| 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

- 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$ ./pwm_setup.sh`~~

```
bone$ source pwm_setup.sh
```

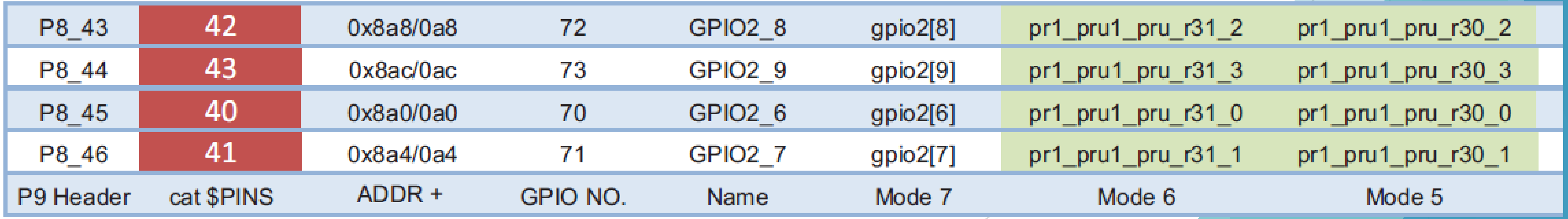
# pwm_setup.sh

```
#!/bin/bash
#
export PRUN=0
export TARGET=pwm1
echo PRUN=$PRUN
echo TARGET=$TARGET
```

# pwm_setup.sh

```
# Configure the PRU pins based on which Beagle is running
machine=$(awk '{print $NF}' /proc/device-tree/model)
echo -n $machine
if [ $machine = "Black" ]; then
    echo " Found"
    pins="P9_31 P9_29 P9_30 P9_28"
elif [ $machine = "Blue" ]; then
    echo " Found"
    pins=""
elif [ $machine = "PocketBeagle" ]; then
    echo " Found"
    pins="P1_36 P1_33 P2_32 P2_30"
else
    echo " Not Found"
```

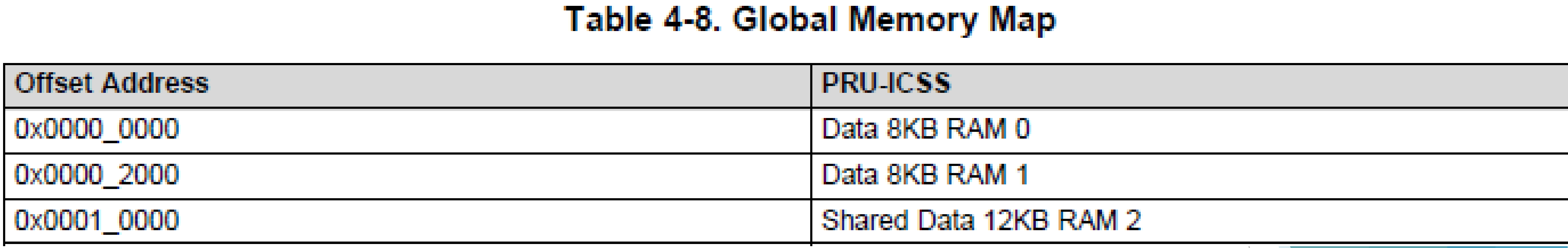
```
bone$ cat /proc/device-tree/model
TI AM335x BeagleBone Green Wireless
```



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| P8_43 | 42 | 0x8a8/0a8 | 72 | GPIO2_8 | gpio2[8] | pr1_pru1_pru_r31_2 | pr1_pru1_pru_r30_2 |
| P8_44 | 43 | 0x8ac/0ac | 73 | GPIO2_9 | gpio2[9] | pr1_pru1_pru_r31_3 | pr1_pru1_pru_r30_3 |
| P8_45 | 40 | 0x8a0/0a0 | 70 | GPIO2_6 | gpio2[6] | pr1_pru1_pru_r31_0 | pr1_pru1_pru_r30_0 |
| P8_46 | 41 | 0x8a4/0a4 | 71 | GPIO2_7 | gpio2[7] | pr1_pru1_pru_r31_1 | pr1_pru1_pru_r30_1 |
| P9 Header | cat $PINS | ADDR + | GPIO NO. | Name | Mode 7 | Mode 6 | Mode 5 |

# pwm_setup.sh

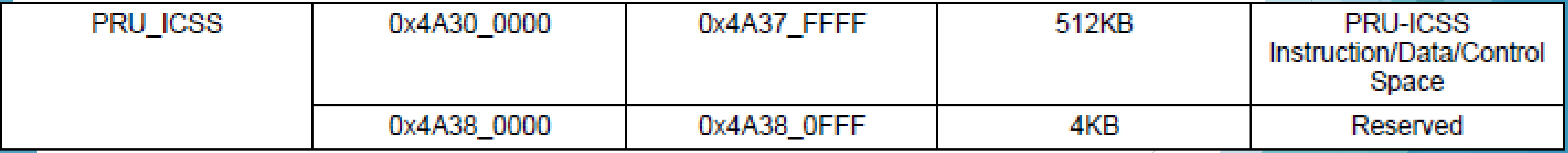
```
for pin in $pins
do
    echo $pin
    config-pin $pin pruout
    config-pin -q $pin
done
```



### Table 4-8. Global Memory Map

| Offset Address | PRU-ICSS |
|---|---|
| 0x0000_0000 | Data 8KB RAM 0 |
| 0x0000_2000 | Data 8KB RAM 1 |
| 0x0001_0000 | Shared Data 12KB RAM 2 |

- You would like to control the frequency and duty cycle of the PWM without recompiling
- Have the PRU read the *on* and *off* times from a shared memory location.
- Each PRU has is own 8KB of data memory (DRAM) and 12KB of shared memory (SHAREDMEM) that the ARM processor can also access
- The DRAM 0 address is 0x0000 for PRU 0. The same DRAM appears at address 0x4A300000 as seen from the ARM processor.



| | | | | |
|---|---|---|---|---|
| PRU_ICSS | 0x4A30_0000 | 0x4A37_FFFF | 512KB | PRU-ICSS Instruction/Data/Control Space |
| | 0x4A38_0000 | 0x4A38_0FFF | 4KB | Reserved |

# pwm3.c

```
// This code does MAXCH parallel PWM channels.
// It's period is 3 us
#include <stdint.h>
#include <pru_cfg.h>
#include "resource_table_empty.h"

#define MAXCH 4   // Maximum number of channels

volatile register uint32_t __R30;
volatile register uint32_t __R31;
```

# pwm3.c

```
void main(void)
{
    uint32_t ch;
    uint32_t on[]  = {1, 2, 3, 4};// Number of cycles to stay on
    uint32_t off[] = {4, 3, 2, 1};// Number to stay off
    uint32_t onCount[MAXCH];      // Current count
    uint32_t offCount[MAXCH];

    /* Clear SYSCFG[STANDBY_INIT] to enable OCP master port */
    CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;

    // Initialize the channel counters.
    for(ch=0; ch<MAXCH; ch++) {
        onCount[ch] = on[ch];
        offCount[ch]= off[ch];
    }
```

# pwm3.c

```
void main(void)

…

while (1) {

          for(ch=0; ch<MAXCH; ch++) {

               if(onCount[ch]) {

                    onCount[ch]--;

                    __R30 |= 0x1<<ch;        // Set the GPIO pin to 1

               } else if(offCount[ch]) {

                    offCount[ch]--;

                    __R30 &= ~(0x1<<ch);     // Clear the GPIO pin

               } else {

                    onCount[ch] = on[ch];    // Reset counts

                    offCount[ch]= off[ch];

               }

          }    }}
```

# On the ARM side

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

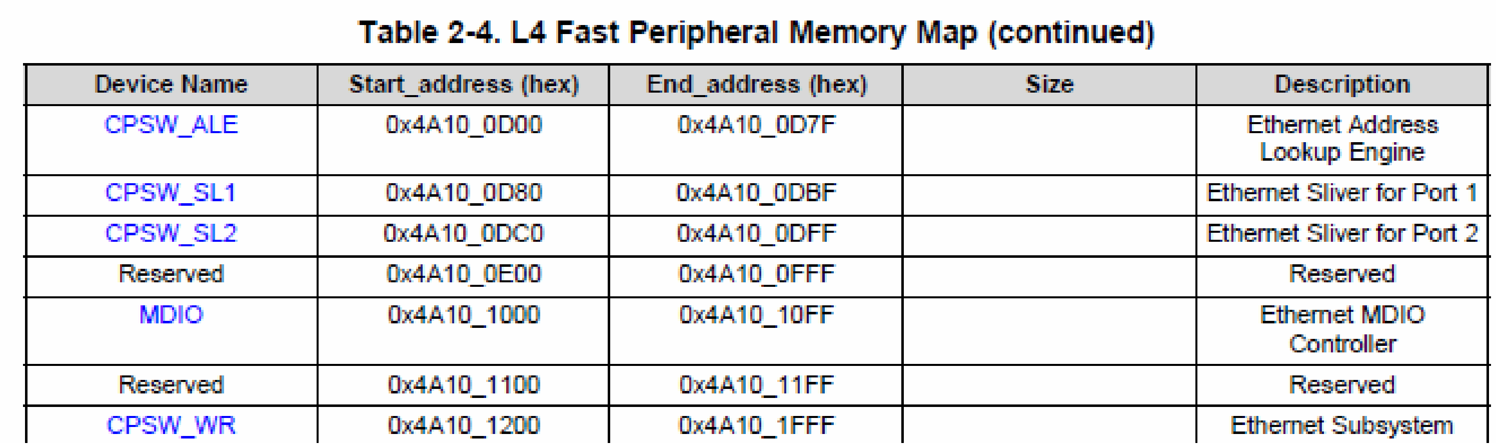
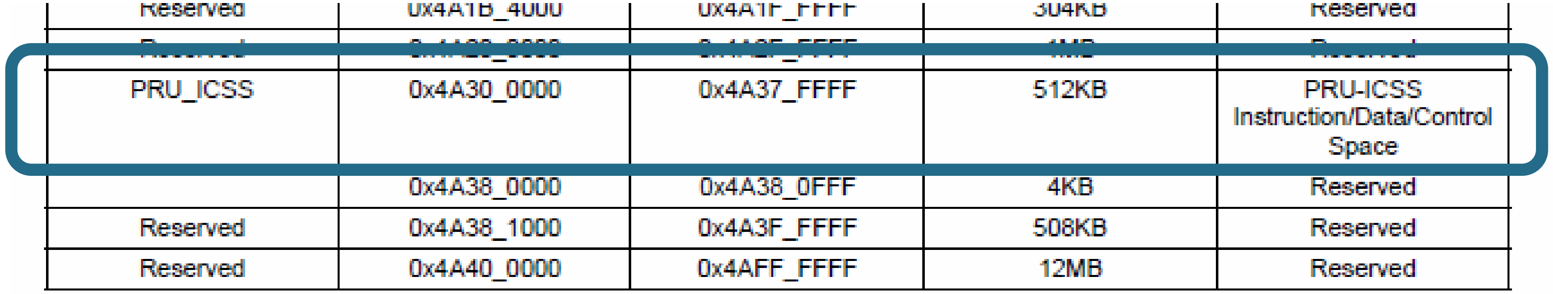


Table 2-4. L4 Fast Peripheral Memory Map (continued)

| 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 | 0x4A1B_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 – pwm-test.c

```
#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

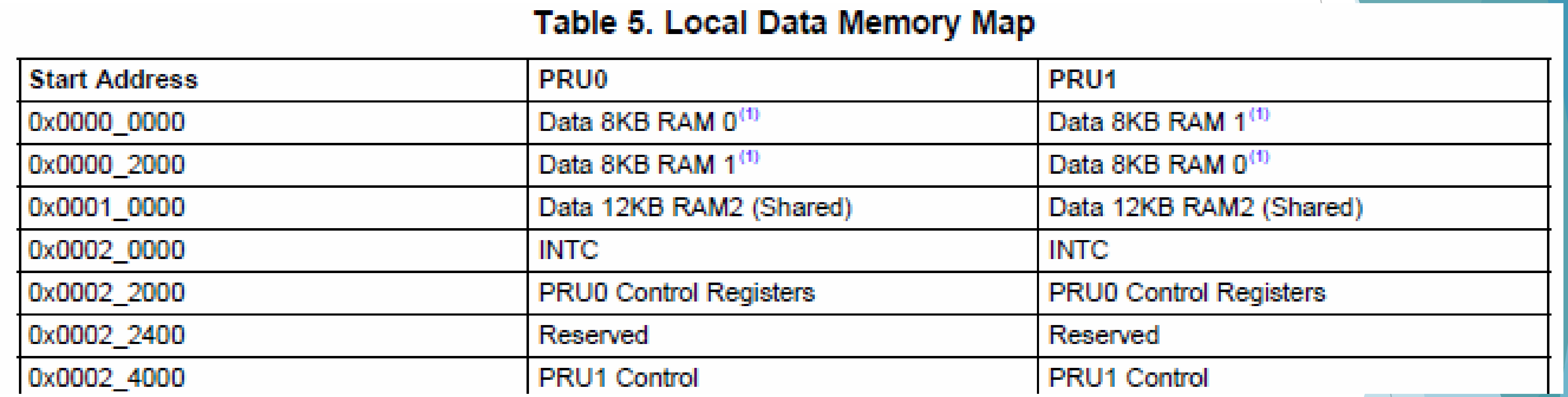


Table 5. Local Data Memory Map

| 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 |

# main() – pwm3.c

```c
int main(int argc, char *argv[])
{
    unsigned int  *pru;         // Points to start of PRU memory.
    int  fd;
    printf("Servo tester\n");

    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);
```

# main() – pwm3.c

```
int main(int argc, char *argv[])
..
    printf ("Using /dev/mem.\n");

    pru0DRAM_32int_ptr =     pru + PRU0_DRAM/4 + 0x200/4;  // Points to 0x200 of PRU0 memory
    pru1DRAM_32int_ptr =     pru + PRU1_DRAM/4 + 0x200/4;  // Points to 0x200 of PRU1 memory
    prusharedMem_32int_ptr = pru + PRU_SHAREDMEM/4;  // Points to start of shared memory

    int i;
    for(i=0; i<MAXCH; i++) {
        start_pwm_count(i, i+1, 20-(i+1));
    }

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

word address

Convert byte address to word address

HEAP + STACK

# Start_pwm_count

```
/*******************************************************************************
* int start_pwm_count(int ch, int countOn, int countOff)
*
* Starts a pwm pulse on for countOn and off for countOff to a single channel (ch)
*******************************************************************************/
int start_pwm_count(int ch, int countOn, int countOff) {
    unsigned int *pruDRAM_32int_ptr = pru0DRAM_32int_ptr;

    printf("countOn: %d, countOff: %d, count: %d\n",
        countOn, countOff, countOn+countOff);
    // write to PRU shared memory
    pruDRAM_32int_ptr[2*(ch)+0] = countOn;  // On time
    pruDRAM_32int_ptr[2*(ch)+1] = countOff; // Off time
    return 0;
}
```

# Running it

```
bone$ cd PRUCookbook/docs/05blocks/code
bone$ source pwm_setup.sh
bone$ make
bone$ ./pwm-test
```