Appendix J: Code for PRU

fpgamem.hp

```
// fpgamem.hp

#ifndef __PRUCODE_FPGA_HP__
#define __PRUCODE_FPGA_HP__

// Definitions

// Refer to this mapping in the file - pruss_intc_mapping.h
#define PRU0_PRU1_INTERRUPT     17
#define PRU1_PRU0_INTERRUPT     18
#define PRU0_ARM_INTERRUPT      19
#define PRU1_ARM_INTERRUPT      20
#define ARM_PRU0_INTERRUPT      21
#define ARM_PRU1_INTERRUPT      22

#define CONST_PRUCFG            C4
#define CONST_PRUDRAM           C24
#define CONST_PRUSHAREDRAM      C28
#define CONST_DDR               C31

// Address for the Constant table Block Index Register (CTBIR)
#define CTBIR           0x24020

// Address for the Constant table Programmable Pointer Register 0(CTPPR_0)
#define CTPPR_0         0x24028

// Address for the Constant table Programmable Pointer Register 1(CTPPR_1)
#define CTPPR_1         0x2402C

// Macros

.macro  LD32
.mparam dst,src
    LBBO    dst,src,#0x00,4
.endm

.macro  LD16
.mparam dst,src
    LBBO    dst,src,#0x00,2
.endm

.macro  LD8
.mparam dst,src
    LBBO    dst,src,#0x00,1
.endm

.macro ST32
.mparam src,dst
    SBBO    src,dst,#0x00,4
.endm

.macro ST16
.mparam src,dst
```

```
    SBBO      src,dst,#0x00,2
.endm

.macro ST8
.mparam src,dst
    SBBO      src,dst,#0x00,1
.endm

// *************************************
// *     Global Structure Definitions     *
// *************************************

.struct Global
    .u32 regPointer
    .u32 regVal
.ends


// *************************************
// *      Global Register Assignments     *
// *************************************

.assign Global, r2, *, global

#endif // __PRUCODE_FPGA_HP__
```

fpgamem.p

```
// fpgamem.p

// Authors: Haolin Li, Joris Van Kerrebrouck
// Acknowledgment: Intec Design, University Ghent
// Project Name:   Etherscope v1.0

//
// BBB Schematic  BBB port  Assign  Bit
// ---------------  --------  ------  ------------
// LCD_DATA0       P8.45     D0            PRU1_R31_0
// LCD_DATA1       P8.46     D1            PRU1_R31_1
// LCD_DATA2       P8.43     D2            PRU1_R31_2
// LCD_DATA3       P8.44     D3            PRU1_R31_3
// LCD_DATA4       P8.41     D4            PRU1_R31_4
// LCD_DATA5       P8.42     D5            PRU1_R31_5
// LCD_DATA6       P8.39     D6            PRU1_R31_6
// LCD_DATA7       P8.40     D7            PRU1_R31_7
// LCD_PCLK         P8.28     CLK          PRU1_R31_10
// LCD_VSYNC      P8.27     START  PRU1_R30_8
// LCD_HSYNC      P8.29     EN     PRU1_R31_9

.origin 0
.entrypoint START

#include "fpgamem.hp"
#define MASK0 0x000000ff

// Memory location where to store the data to be acquired:
#define ACQRAM 0x00010004

// Length of acquisition:
```

```
#define RECORDS 4096     //1024

START:
// Enable OCP master port
LBCO      r0, CONST_PRUCFG, 4, 4
CLR        r0, r0, 4          // Clear SYSCFG[STANDBY_INIT] to enable OCP master port
SBCO     r0, CONST_PRUCFG, 4, 4

// Configure the programmable pointer register for PRU1
// by setting c28_pointer[15:0] field to 0x0100.
// This will make C28 point to 0x00010000 (PRU shared RAM). see 'fpgamem.hp'
MOV      r0, 0x00000100
MOV      r1, CTPPR_0
ST32     r0, r1

// Configure the programmable pointer register for PRU1
// by setting c31_pointer[15:0] field to 0x0010.
//This will make C31 point to 0x80001000 (DDR memory). see 'fpgamem.hp'
MOV     r0, 0x00100000
MOV     r1, CTPPR_1
ST32    r0, r1

MOV    r7, RECORDS // This will be the loop counter to read the entire set of data
MOV    r4, MASK0    //r4 is the mask for the lowest 8bits

MOV      r2.w0,0x0000
SBCO     0x0, CONST_PRUDRAM, 0, 2   //clear flag for data RAM1
MOV      r3,0x00002000
SBCO     0x0, CONST_PRUDRAM, r3, 2  //clear flag for data RAM0

MOV    r3,2      //index
MOV    r8,20000
SET    r30.t8

LOOOP1:
    WBS    r31.t9

    // 8bits
    WBS    r31.t10     //wait for clock rising edge-> data arrives
    WBC    r31.t10     //read data at falling edge
    MOV    r2.b0, r31.b0     //read input register, and copy to r2

    SBCO   r2.b0, CONST_PRUDRAM, r3, 1
    ADD     r3,r3,1

    SUB      r7,r7,1
    QBNE   LOOOP1,r7,0
    CLR      r30.t8
    SBCO   0x0001, CONST_PRUDRAM,0, 2     //set flag high

    MOV     r3, 0x00002000       //local address in dataRAM0
    JMP      WAIT0

LOOOP0:

    //8bits
    WBS     r31.t9

    WBS     r31.t10     //wait for clock rising edge-> data arrives
```

```
    WBC     r31.t10     //read data at falling edge
    MOV     r2.b0, r31.b0     //read input register, and copy to r2

    SBCO    r2.b0, CONST_PRUDRAM, r3, 1
    ADD     r3,r3,1

    SUB       r7,r7,1
    QBNE    LOOOP0,r7,0
    CLR       r30.t8
    MOV     r3, 0x00002000

    SBCO    0x0001, CONST_PRUDRAM,r3, 2    //set flag high

    SUB       r8,r8,1
    QBNE    WAIT1,r8,0
    JMP       EXIT
WAIT1:
    //wait until the flag is cleared
    LBCO    r12.w0, CONST_PRUDRAM,0,2
    QBNE    WAIT1,r12.w0,0x0000

    MOV     r7, RECORDS
    MOV     r3,2
    SET       r30.t8
    JMP       LOOOP1
WAIT0:
    //wait until the flag is cleared
    LBCO    r12.w0, CONST_PRUDRAM,r3,2
    QBNE    WAIT0,r12.w0,0x0000

    ADD     r3,r3,2
    MOV     r7, RECORDS
    SET       r30.t8
    JMP       LOOOP0

EXIT:
// Send notification to Host for program completion
   MOV      r31.b0, PRU1_ARM_INTERRUPT + 16

// Halt the processor
   HALT
```

Makefile script

```
 #!/bin/bash

echo "Start compiling ..."

pasm -V3 -cdl  -b fpgamem.p
```

pru.h

```
#ifndef PRU_H
#define PRU_H

char convertdata(unsigned char);

unsigned int InitPRU(void);   /* Initialize PRU and PRU Interrupt */
int   InitMEM(void);  /* Initialize PRU DATARAM mapping */
```

```
void ExePRU(void);  /* Execute the bin file */

void communication1(unsigned char*);  /* Communication Protocol and Data Transfer*/
void communication0(unsigned char*);  /* Communication Protocol and Data Transfer*/

void ExitPRU(void);

#endif
```

pru.c

```
/* * pru.c
*
* Copyright (C) 2014 ADLab - http://www.analogdigitallab.org/
* Authors:        Haolin Li, Joris Van Kerrebrouck
* Acknowledgment: Intec Design, University Ghent
* Project Name:       Etherscope v1.0
* Create Date:        23:32:38 02/27/2013
*/

/*****************************************************************************
* Include Files *
*****************************************************************************/

// Standard header files
#include <stdio.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>

#include "pru.h"

// Driver header file
#include "prussdrv.h"
#include <pruss_intc_mapping.h>


/*****************************************************************************
* Local Macro Declarations *
*****************************************************************************/
#define PRU_NUM 1
#define OFFSET_MEM0 0x00002000
#define OFFSET_SHAREDRAM 0x00010000


/*****************************************************************************
* Local Function Declarations *
*****************************************************************************/



/*****************************************************************************
* Global Variable Definitions *
*****************************************************************************/
static void *pruDataMem;
static unsigned char *pruDataMem_short0; //AM33XX_DATA 8KB RAM0
static unsigned char *pruDataMem_short1; //AM33XX_DATA 8KB RAM1
```

```
int num=4096;

/******************************************************************************
* Global Function Definitions *
******************************************************************************/
char convertdata(unsigned char d){
    // Convert 2-complementary binary data to decimal
        int sign;
        char converted;
        sign = (d >> 7) & 0x01;
        if (sign == 1){ converted = d - 128; } //negative
        else { converted = d + 128; }    //positive
        return converted;
}


unsigned int InitPRU(void){
        unsigned int ret;
        tpruss_intc_initdata pruss_intc_initdata = PRUSS_INTC_INITDATA;
        printf("\nINFO: Starting %s PRU.\r\n", "Node.js -> C++ -> C -> PRU");

        /* Initialize the PRU */
        prussdrv_init();

        /* Open PRU Interrupt */
        ret = prussdrv_open(PRU_EVTOUT_1);
        if (ret)
        {
                printf("prussdrv_open open failed\n");
                return (ret);
        }
        /* Get the interrupt initialized */
        prussdrv_pruintc_init(&pruss_intc_initdata);
        return ret;  // ret=0 if PRU successfully initialized
}


int InitMEM() {

        prussdrv_map_prumem(PRUSS0_PRU1_DATARAM, &pruDataMem);

        //assign the the data RAM address to two pointers
        pruDataMem_short1 = (unsigned char*)pruDataMem; //AM33XX_DATA 8KB RAM1,Glocal Memor
        pruDataMem_short0 = (unsigned char*)(pruDataMem - OFFSET_MEM0); //AM33XX_DATA 8KB R


        return(0);
}


void ExePRU(void){
        /* Execute the bin file */

        prussdrv_exec_program(PRU_NUM, "./fpgamem.bin");
        printf("\tINFO: Executing ...\r\n");
        // give some time for the PRU code to execute
        sleep(1);
```

```
}


void communication1(unsigned char*  data){
        unsigned int x;
        while (pruDataMem_short1[0] == 0 && pruDataMem_short1[1] == 0);
        for (x = 0; x < num; x++)
        {
                data[x] = convertdata(pruDataMem_short1[x + 1]);
        }
        pruDataMem_short1[0]=0;pruDataMem_short1[1] = 0;
}


void communication0(unsigned char* data){
        unsigned int x;
        while (pruDataMem_short0[0] == 0 && pruDataMem_short0[1] == 0);
        for (x = 0; x < num; x++)
        {
                data[x] = convertdata(pruDataMem_short0[x + 1]);
                // data[x] = convertdata(pruDataMem_short0[x + 1]);

        }
         pruDataMem_short0[0] = 0;   pruDataMem_short0[1] = 0;
}


void ExitPRU(){
        /* Disable PRU and close memory mapping*/
        prussdrv_pru_disable(PRU_NUM);
        prussdrv_exit();
}
```