//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// main\_spi\_lis3dh.c - Example demonstrating how to configure SSI0 in SPI master

// mode to read accelerometer data from LIS3DH sensor.

//

// Copyright (c) 2010-2011 Texas Instruments Incorporated. All rights reserved.

// Software License Agreement

//

// Texas Instruments (TI) is supplying this software for use solely and

// exclusively on TI's microcontroller products. The software is owned by

// TI and/or its suppliers, and is protected under applicable copyright

// laws. You may not combine this software with "viral" open-source

// software in order to form a larger program.

//

// THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS.

// NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT

// NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

// A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY

// CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL

// DAMAGES, FOR ANY REASON WHATSOEVER.

//

// This is part of revision 8034 of the Stellaris Firmware Development Package.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_ssi.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/ssi.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/uart.h"

**#include** "utils/uartstdio.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

//! \addtogroup ssi\_examples\_list

//! <h1>SPI Master (spi\_master)</h1>

//!

//! This example shows how to configure the SSI0 as SPI Master. The code will

//! send three characters on the master Tx then polls the receive FIFO until

//! 3 characters are received on the master Rx.

//!

//! This example uses the following peripherals and I/O signals. You must

//! review these and change as needed for your own board:

//! - SSI0 peripheral

//! - GPIO Port A peripheral (for SSI0 pins)

//! - SSI0CLK - PA2

//! - SSI0Fss - PA3

//! - SSI0Rx - PA4

//! - SSI0Tx - PA5

//!

//! The following UART signals are configured only for displaying console

//! messages for this example. These are not required for operation of SSI0.

//! - UART0 peripheral

//! - GPIO Port A peripheral (for UART0 pins)

//! - UART0RX - PA0

//! - UART0TX - PA1

//!

//! This example uses the following interrupt handlers. To use this example

//! in your own application you must add these interrupt handlers to your

//! vector table.

//! - None.

//!

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// The error routine that is called if the driver library encounters an error.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#ifdef** DEBUG

**void**

\_\_error\_\_(**char** \*pcFilename, uint32\_t ui32Line)

{

UARTprintf("Error at line %d of %s\n", ui32Line, pcFilename);

**while**(1)

{

}

}

**#endif**

**#define** NUM\_SSI\_DATA 6 //Store 16bit of data for X,Y and Z axis

**#define** NUM\_SSI\_CFG 4 //Store Configuration Parameters

**#define** LIS3DH\_RANGE\_16\_G 0b11 // +/- 16g

**#define** LIS3DH\_RANGE\_8\_G 0b10 // +/- 8g

**#define** LIS3DH\_RANGE\_4\_G 0b01 // +/- 4g

**#define** LIS3DH\_RANGE\_2\_G 0b00 // +/- 2g (default value)

**#define** LIS3DH\_DATARATE\_400\_HZ 0b0111 // 400Hz

**#define** LIS3DH\_DATARATE\_200\_HZ 0b0110 // 200Hz

**#define** LIS3DH\_DATARATE\_100\_HZ 0b0101 // 100Hz

**#define** LIS3DH\_DATARATE\_50\_HZ 0b0100 // 50Hz

**#define** LIS3DH\_DATARATE\_25\_HZ 0b0011 // 25Hz

**#define** LIS3DH\_DATARATE\_10\_HZ 0b0010 // 10 Hz

**#define** LIS3DH\_DATARATE\_1\_HZ 0b0001 // 1 Hz

**#define** LIS3DH\_DATARATE\_POWERDOWN 0

**#define** LIS3DH\_DATARATE\_LOWPOWER\_1K6HZ 0b1000

**#define** LIS3DH\_DATARATE\_LOWPOWER\_5KHZ 0b1001

**#define** BAUD 38400

**void** **InitConsole**(**void**)

{

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

// U can also use UARTStdioConfig to initialize the UART for console I/O.

// UARTStdioConfig(0, 115200, 16000000);

}

uint8\_t ulDataTx[NUM\_SSI\_DATA];

uint8\_t ulDataRx[NUM\_SSI\_DATA];

uint8\_t ulCFGTx[NUM\_SSI\_CFG];

**int** **main**(**void**)

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

InitConsole();

**UARTprintf**("SSI ->\n");

**UARTprintf**("Mode: SPI\n");

**UARTprintf**("Data: 8-bit\n\n");

ulDataRx[0]=1;

//

// The SSI0 peripheral and port A must be enabled for use.

// Enable the SSI0 peripheral

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_SSI0);

// The SSI0 peripheral is on Port A and pins 2,3,4 and 5.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

// This function/s configures the pin muxing on port A pins 2,3,4 and 5

**GPIOPinConfigure**(GPIO\_PA2\_SSI0CLK);

**GPIOPinConfigure**(GPIO\_PA3\_SSI0FSS);

**GPIOPinConfigure**(GPIO\_PA4\_SSI0RX);

**GPIOPinConfigure**(GPIO\_PA5\_SSI0TX);

**GPIOPinTypeSSI**(GPIO\_PORTA\_BASE, GPIO\_PIN\_5 | GPIO\_PIN\_4 | GPIO\_PIN\_3 |

GPIO\_PIN\_2);

// Deselect the Slave

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

// Configure and enable the SSI port for SPI master mode.

**SSIClockSourceSet**(SSI0\_BASE,SSI\_CLOCK\_SYSTEM);

**SSIConfigSetExpClk**(SSI0\_BASE, **SysCtlClockGet**(), SSI\_FRF\_MOTO\_MODE\_0,

SSI\_MODE\_MASTER, 1000000, 8);

**SSIEnable**(SSI0\_BASE);

// Read any residual data from the SSI port.

**while**(**SSIDataGetNonBlocking**(SSI0\_BASE, &ulDataRx[0]));

// Display indication that the SSI is transmitting data.

**UARTprintf**("Sent:\n ");

// Send the data using the "blocking" put function.

// Initialize the data to send.

ulDataTx[0] = (0x0F & ~0x0C); //ASK WHO AM I?

//Make SSI pin low - enable slave device

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

// Send data to slave to ask "Who am I"?

**SSIDataPut**(SSI0\_BASE,&ulDataTx[0]);

**while**(**SSIBusy**(SSI0\_BASE));

**SSIDataGet**(SSI0\_BASE, &ulDataRx[0]);

// Deselect the slave

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

**UARTprintf**(" ", ulDataRx[0]);

**if** (ulDataRx[0] == 0x33)

**UARTprintf**("Device ID'%X' found:\n", ulDataRx[0]);

**else**

**UARTprintf**("Device not found:\n");

// Configuration Sequence

// enable all axes, normal mode

// 100Hz rate

ulCFGTx[0] = 0x57;

ulCFGTx[1] = 0x00; // High res, BDU enabled, range = 4g @ 0x21 (CTRL2)

ulCFGTx[2] = 0x00; // DRDY on INT1 @ 0x22 (CTRL3)

ulCFGTx[3] = 0x00; //set Range to 2Gs @ 0x24 (CTRL4)

// Configure LIS3D

// enable all axes, normal mode and 400Hz rate

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0); // CS = 0 enable

**SSIDataPut**(SSI0\_BASE,0x20 & ~0xC0);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3); // CS =1 disable

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

**SSIDataPut**(SSI0\_BASE,&ulCFGTx[0]);

**while**(**SSIBusy**(SSI0\_BASE));

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

// High res & BDU enabled

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0); // CS = 0 enable

**SSIDataPut**(SSI0\_BASE,0x23 & ~0xC0);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3); // CS =1 disable

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

**SSIDataPut**(SSI0\_BASE,&ulCFGTx[1]);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

// DRDY on INT1

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0); // CS = 0 enable

**SSIDataPut**(SSI0\_BASE,0x22 & ~0xC0);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3); // CS =1 disable

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

**SSIDataPut**(SSI0\_BASE,&ulCFGTx[2]);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

//set Range to 4Gs

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0); // CS = 0 enable

**SSIDataPut**(SSI0\_BASE,0x1F & ~0xC0);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3); // CS =1 disable

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

**SSIDataPut**(SSI0\_BASE,&ulCFGTx[3]);

**while**(**SSIBusy**(SSI0\_BASE)){}

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

//Repeat for all data registers

**while**(1)

{

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,0);

//Read Data from 0x28 ... 0x2D

**SSIDataPut**(SSI0\_BASE,0x28|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[0]);

**SSIDataPut**(SSI0\_BASE,0x29|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[1]);

**SSIDataPut**(SSI0\_BASE,0x2A|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[2]);

**SSIDataPut**(SSI0\_BASE,0x2B|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[3]);

**SSIDataPut**(SSI0\_BASE,0x2C|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[4]);

**SSIDataPut**(SSI0\_BASE,0x2D|0x80); //read data

**while**(**SSIBusy**(SSI0\_BASE)){}

**SSIDataGet**(SSI0\_BASE, &ulDataRx[5]);

**GPIOPinWrite**(GPIO\_PORTA\_BASE,GPIO\_PIN\_3,GPIO\_PIN\_3);

// Change divisor based on range (2g=16380, 4g= 8190, 8g=4096, 16g=1365)

**UARTprintf**("X-axis:'%f' ", (**float**) (ulDataRx[1]<<8 | ulDataRx[0])/16380);

**UARTprintf**("Y-axis:'%f' ", (**float**) (ulDataRx[3]<<8 | ulDataRx[2])/16380);

**UARTprintf**("Z-axis:'%f' ", (**float**) (ulDataRx[5]<<8 | ulDataRx[4])/16380);

}

**return**(0);

}