Plastic Logic Micro-Controller E-Paper Software Guide for UC8156 based evaluation kits

1 Table of Contents

2 Licensing

The majority of the software in this codebase was written by Plastic Logic and is currently licensed under a restrictive license for early adopters. For the avoidance of confusion: This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Some key support functionality is provided by third parties who have their own licenses. The third party

components are: FatFs – A FAT file-system driver. This is used to access configuration and image data stored on a micro SD card on the reference microcontroller hardware. The license for FatFS can be found here: http://elm-chan.org/fsw/ff/en/appnote.html#license, it is not restrictive. Sample code - This is sample source code made freely available by the microcontroller vendor. The copyright notices vary from source file to source file but are not restrictive other than limiting the use of such processor specific sample code to a given range of processor devices. Please see Appendix A for license text.

3 Glossary

CCS

Code Composer Studio, an integrated development environment from Texas Instruments that can be used to develop code for the MSP430 microcontroller

EPD

Electrophoretic display. Such displays retain the last image driven to them and only require power to change the image

EPDC

Electrophoretic display controller. A specialized display timing controller required for updating electrophoretic displays. The EPDC is responsible for applying the correct waveform to each pixel, according to the current and target images

FFC

Flexible flat cable (http://en.wikipedia.org/wiki/Flat_Flex_Cable)

GPIO

General-purpose input/output. A user-controllable pin that can be defined at runtime as either an input or an output

HVPMIC

High voltage power management IC. A chip that converts a single (typically battery) voltage into the various higher voltages required by the display

I2C

Inter-Integrated Circuit, a standard two-wire multimaster serial bus intended for communication with low-speed peripherals

MSP430

A low-power microcontroller from Texas Instruments

Parrot board

An evaluation board from Plastic Logic containing a MSP430 microcontroller

PGM

A portable graphics file format

SPI

Serial Peripheral Interface, a standard four-wire serial bus that operates in full duplex mode

USCI

Universal Serial Communication Interface. MSP430 serial communications interface that supports multiple serial communication modes with one hardware module

VCOM

Display-specific common electrode voltage. Each Plastic Logic display is supplied with the correct voltage that must be applied by the control electronics

Waveform

A display-specific data file that defines how the display updates

4 Configuring the display type

The code includes a number of features and demonstrations that can be configured at run time via the use of settings in the display_type.txt file.

The following example defines a S011_T1.1 display, see all possible displays in chapter Display Types:

```
S011_T1.1
```

5 Code Structure

5.1 Host Abstraction Layer

The host abstraction layer isolates the platform neutral code from the underlying platform hardware. The abstraction layers are kept as self-contained and thin as practical. While interrupts and timers are listed their availability is not required to create a working system.

5.1.1 Host GPIO Interface

The GPIO interface provides a way to reserve and define a GPIO pin on the host processor at run time. On small microcontrollers pins are typically either GPIOs or connected to a single special purpose hardware unit e.g. an I2C unit. Some, or all, of the GPIOs supported may be able to generate interrupts. The GPIO interface records which GPIOs are already defined but not the mode in which they are configured. This allows the code to trap errors where a pin is defined multiple times, or used before being defined. GPIO pins are typically used to control the power sequence hardware and manipulate signals in the serial and parallel interface to the EPDC controller.

5.1.2 Host I2C Interface

The host I2C interface provides access to an I2C interface physically attached to the host processor. Only a single I2C interface is supported by the code. A host I2C interface may not be required if the system is configured to use the EPDC SPI-I2C bridge feature instead. Examples of devices connected to I2C include the HVPMIC, temperature sensors, and EEPROMs.

5.1.3 Host SPI Interface – SD Card

The host SPI-SDCard interface provides access to an SPI interface that is connected to the SD Card. The SD Card is operated at 20MHz. If additional hardware is available in the host processor the SD Card could be operated in 4 bit parallel mode for improved data transfer speed.

5.1.4 Host Interrupt Interface

The interrupt interface supports the processing of interrupts. The code currently does not use interrupts but the first usage will be for notifying the code that the EPDC is ready to accept a new command by the assertion of the HRDY line. The abstraction is still to be defined

5.1.5 Host Timer Interface

The timer interface provides platform specific timer implementations. Currently delays are coded as busy loops. A more power efficient mechanism will follow in a future release.

5.2 MSP430 Specific Host Interfaces

5.2.1 GPIO Interface

This is the reference implementation for the GPIO host interface and can be found in msp430/msp430-gpio.c. It supports the configuration of all features on all pins that can be configured. It is only possible to configure one pin at a time in a port. It is not possible to define the configuration of multiple pins in a port with one call – e.g. when defining an 8 bit bus as output or input. The code attempts to verify the request as much as it can. Much of the error checking code can be disabled once the porting process to a new platform has been completed and the platform configuration is stable.

5.2.2 I2C Interface

A single I2C interface is supported. I2C is only supported in USCI modules and the chosen UCSI module is defined in the msp430/mlsp430-i2c.c source file by setting the macros USCI_UNIT and USCI_CHAN as required. The code will then reconfigure itself to reference the correct I2C unit. In addition to specifying which UCSI module to use the I2C SDA and SCL pins need to be connected to the USCI unit by defining the appropriate pins as PL_GPIO_SPECIAL in the pl_gpio_config_list() call.

5.2.3 SPI Interface – EPDC

SPI is supported in both USCI_A and USCI_B modules and the chosen USCI module is defined in the msp430/msp430-spi.c source file by setting the macros USCI_UNI and USCI_CHAN as required. The code will then reconfigure itself to reference the correct SPI unit. In addition to specifying which USCI module to use the SPI_CLK, SPI_MOSI and SPI_MISO pins need to be

connected to the USCI unit by defining the appropriate pins as PL_GPIO_SPECIAL in the pl_gpio_config_list() call. Note that it is possible to use both the USCI_A and USCI_B units, i.e. USCI_A0 and USCI_B0 are physically different hardware units. A single SPI interface is supported for EPDC controller communications. Multiple controllers can be connected to this bus and are selected using their chip select lines as required. This interface runs at 20Mbps reliably. Due to the need to keep the EPDC chip selected for the duration of the image data transfer the EPDC controller must be placed on a separate bus to the SD card so that multiple blocks can be read from the SD card.

5.2.4 SPI Interface – SD Card

SPI is supported in both USCI_A and USCI_B modules and the chosen USCI module is defined in the msp430/msp430-sdcard.c source file by setting the macros USCI_UNI and USCI_CHAN as required. The code will then reconfigure itself to reference the correct SPI unit. In addition to specifying which USCI module to use the SPI_CLK, SPI_MOSI and SPI_MISO pins need to be connected to the USCI unit by defining the appropriate pins as PL_GPIO_SPECIAL in the pl_gpio_config_list() call. Note that it is possible to use both the USCI_A and USCI_B units. i.e. USCI_A0 and USCI_B0 are physically different hardware units. A single SPI interface is supported for transferring data from the micro SD card slot. This interface runs at 20Mbps reliably.

5.2.5 UART Interface

UART mode is supported in the USCI_A module and the code handling this can be found in the msp430\msp430-uart.c source file. In the sample code, the UART interface is used only to handle stderr and stdout, and then only if CONFIG_UART_PRINTF is defined (in config.h). In Code Composer Studio it is not possible simply to override putc() and puts(), and instead a device has to be registered (see msp430_uart_register_files()).

5.2.6 Porting the Existing Code to another MSP430 Processor

Porting the existing code to a design which requires a different pin out is relatively straightforward. The necessary configuration information is mainly contained in the main.c file. To reconfigure the reference code follow the sequence below:

1. Determine which USCI units will be used in the new configuration. Ensure the unit is suitable for its intended purpose
2. Determine which pins are associated with the chosen USCI units
3. Determine which pins will be used for the EPDC SPI signals HRDY, HDC, and RESET
4. Determine which pin(s) will be used for the EPDC SPI chip select
5. Determine which pins may be necessary to control the power supplies
6. In each of the msp430/msp430-spi.c, msp430/msp430-sdcard.c, mps430/msp430-i2c.c and msp430/msp430-uart.c
    a. Define USCI_UNIT and USCI_CHAN as required

b. Modify the definitions for the pins so they match the chosen UCSI unit, e.g.:

```
#define USCI_UNIT B
#define USCI_CHAN 0
// Pins from MSP430 connected to the SD Card
#define SD_CS MSP430_GPIO(5,5)
#define SD_SIMO MSP430_GPIO(3,1)
#define SD_SOMI MSP430_GPIO(3,2)
#define SD_CLK MSP430_GPIO(3,3)
```

7. In main.c, define the EPDC SPI interface signals, e.g.:

```
// Remaining EPDC interface pins
#define EPDC_HDC MSP430_GPIO(1,3)
#define EPDC_HRDY MSP430_GPIO(2,7)
#define EPDC_RESET MSP430_GPIO(5,0)
```

8. In main.c, define the power control and EPDC chip select pins, e.g.:

```
#define B_HWSW_CTRL MSP430_GPIO(1,2)
#define B_POK MSP430_GPIO(1,0)
#define B_PMIC_EN MSP430_GPIO(1,1)
#define EPDC_CS_0 MSP430_GPIO(3,6)
```

Recompile the code and it has now been retargeted to the new pin assignments.

5.3 Plastic Logic Evaluation Hardware

5.3.1 Display Types

The code supports the following Plastic Logic display types. Additional displays will be supported as required.

Display Type	Resolution	Notes
S011_T1.1	148x70	1,1" 150ppi
S014_T1.1	180x100	1,38" 150ppi
S021_T1.1	240x146	2,1" 150ppi
S031_T1.1	312x74	3,1" 105ppi

5.3.2 Parrot - MSP430 Processor Board

The Parrot board docks with the Ruddock2 motherboard to provide access to the display interfaces. It has the same form factor and connector pin out as a BeagleBone allowing the processors to be easily swapped for evaluation or development work. The Parrot board can also be used without the Ruddock2 by connecting it directly to the Z6, Z7 or Raven boards via the 24-pin "serial" interface.

The board has the following features:

- MSP430F5438A, clocked at 20MHz
- A 32KHz oscillator for low power operation
- micro SD card socket
- On-board reset switch

- JTAG programming header (an adapter may be required to mate with the MSP-FET430UIF programmer)
- All 100 processor pins available on debug headers
- On-board power regulation and power socket (can also be powered from USB)
- The board has 1 LED for power good and another connected to a pin on the processor for status indication
- 24-pin "serial" interface to Z6, Z7 and Raven boards
- Provision for an SPI daisy-chain of MSP430 boards using 2 SPI channels (upstream and downstream)

6 Relevant Data Sheets and Additional Resources

6.1 Third Party Datasheets and Resources

TI MSP430 tools and software

http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/tools_software.page

7 Appendix A - License Text

The license text for third party components is reproduced below:

7.1 FatFs

```
/*----------------------------------------------------------------------------/
* / FatFs - FAT file system module R0.08a (C)ChaN, 2010
* /-----------------------------------------------------------------------------/
* / FatFs module is a generic FAT file system module for small embedded systems.
* / This is a free software that opened for education, research and commercial
* / developments under license policy of following terms.
* /
* / Copyright (C) 2010, ChaN, all right reserved.
* /
* / * The FatFs module is a free software and there is NO WARRANTY.
* / * No restriction on use. You can use, modify and redistribute it for
* / personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.
* / * Redistributions of source code must retain the above copyright notice.
* /
* /-----------------------------------------------------------------------------/
*/
```

7.2 Texas Instruments

```
/* --COPYRIGHT--,BSD_EX
* Copyright (c) 2012, Texas Instruments Incorporated
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* * Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* * Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the follo wing disclaimer in the
* documentation and/or other materials provided with the distribution.
*
* * Neither the name of Texas Instruments Incorporated nor the names of
* its contributors may be used to endorse or promote products derived
* from this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*****************************************************************************
*/
```