TITLE

AUTHOR
Version 1.0
CREATEDATE

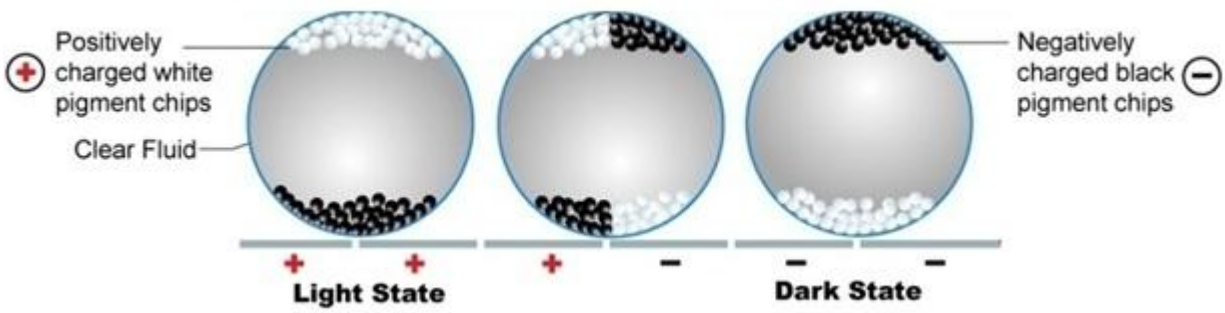
Table of Contents

Table of contents

Driving EInk Displays

EInk Segmented Display Technology

Electronic ink is made up of millions of tiny microcapsules, about the diameter of a human hair. Each microcapsule contains positively charged white particles and negatively charged black particles which are suspended in a clear fluid. When a positive or negative electric field is applied, corresponding particles move to the top of the microcapsule where they become visible to the user. This makes the surface appear white or black at that spot. Thus, driving an E Ink display follows a paradigm that is significantly different from the approach used to drive a traditional segmented TN-LCD.



E Ink Segmented displays function based on a difference in voltage potential between 0 and 5V. E Ink displays have a clear, conductive "Top Electrode" layer which is the basis for all switching. All other segments need to be driven in opposition to the Top Electrode to experience a change in state (black to white or white to black).

Standard pin mapping is as follows:

Pin 1 of the display = Top Electrode

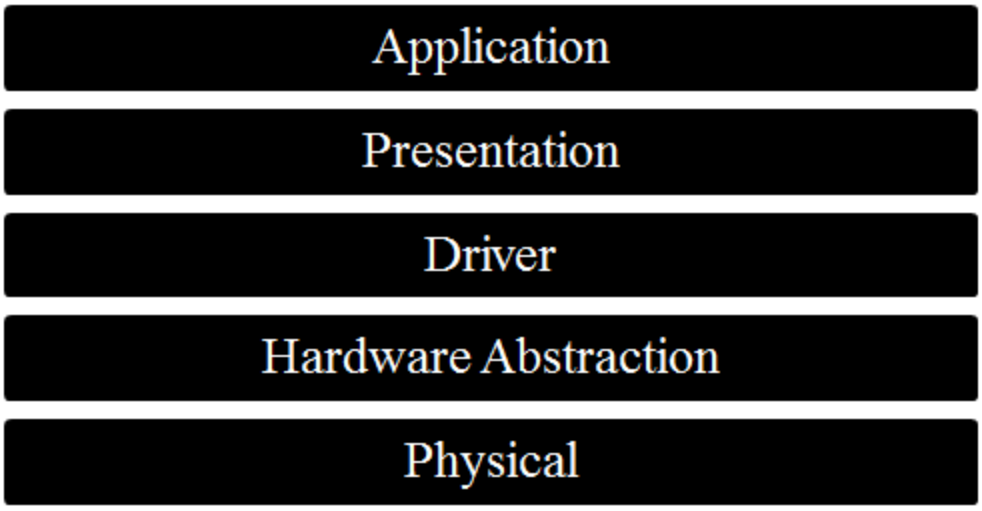
Pin 2 of the display = Field/background (driven like any other segment)

Pins 3 - n are for all other segments

Understanding the driver

Overview

The rEInkDriver is implemented as different layers as shown below:



The Physical Layer:

The physical layer consists of the actual pins that toggle voltages for each segment of the EInk display. This is done using the function SetOutput in the file rEInkDrive.c . The user should add more statements to this function in the already present format to extend functionality for the actual number of segments present. The initEInkHardware is another such function where user intervention is required to ensure that all pins connected to the EInk display segments are configured as output.

The Hardware Abstraction Layer:

The hardware abstraction layer is a simple set of preprocessor definitions that exist in the file rEInkHardwareAbstraction.h. This file is included in rEInkDrive.c and contains the names of each microcontroller port pin that maps to each segment of the EInk display. These definitions are replaced in the setOutput function when preprocessing of the source file occurs.

The Driver Layer:

The driver layer consists of the einkDriver function which is called periodically by either the timer interrupt or from the main loop. When not updating the screen, the driver is in the IDLE state checking if an update request is made to physically update the screen. When an update request is made, the state transitions to WAIT state. In the WAIT state, a call is made to UpdateScreen. The updateScreen function selects the voltage for each segment depending upon the user's choice of waveform and the previously applied voltage. There is also a delay between switching voltages. The selected voltage for each pin is applied using the setOutput function.

The Presentation Layer:

The presentation layeris useful when dealing with EInk displays that may have a lot of segments, digits, icons or combinations of the same. The einkDriver function reads the nextDisplay array to determine which segments need transitions. This task of setting the elements of nextDisplay array can be difficult when there are a lot of segments in a display or if the segments are not grouped together in the standard 8, 9, 14 or 16 segment format. This issue can be resolved by filling out information for the structure einkScreenDef which contains all the parameters to logically define the screen. The displayInZone uses the information passed through these parameters and a hexadecimal encoding to select set the elements in nextDisplay to "1".

Important Variables and definitions:

Some of the important variables to look at are:

updateRequest

curDisplay

nextDisplay

Some of the important preprocessor definitions that may change according to your requirements are:

SEG_DELAY

NUM_FENCES

PICKET_SHORT

NUM_SEGMENTS

TOP_PLANE

USING_PICKET_FENCE

USING_SLIDESHOWS

USING_GLOBAL_UPDATES

An important structure definition to look at:

einkScreenDef

Some important functions to look over are:

initEInkHardware

einkDriver

Connecting the hardware to EInk displays

EInk displays are usually terminated with stiffeners that can slide into FPC connectors. For the purpose of development, one may use the YLPDSKRL78EINK daughter board which connects to the YRPBRL78G14. The sample project provided is an IAR Workspace that runs on the YRPBRL78G14. In the case of the YRPBRL78G14, the segments are driven directly from the MCU pins. The EInk display requires atleast 5V to be able to drive the display. Note that an EInk segmented display cannot be driven using 3.3V.

Incorporating EInk driver into your application

Step 1: Defining the segments

In the file rEInkHardwareAbstraction.h you will find the preprocessor definitions for each pin connected to each segment of the EInk display screen.

You may add or delete some of these definitions depending upon the screen you choose to drive and the number of segments it needs.

Also make changes to the function SetOutput and initEInkHardware in the file rEInkDriver.c while noting the format of the previously written statements.

Step 2: Positioning the driver function and delay

Ensure that you have provision for atleast one interval timer interrupt. Within this periodic timer interrupt place the function call to einkDriver. Also ensure that you also calling the function TimerTick defined in timing.h.

Redefine the preprocessor definitions for PICKET_SHORT, NUM_FENCES, SEG_DELAY.

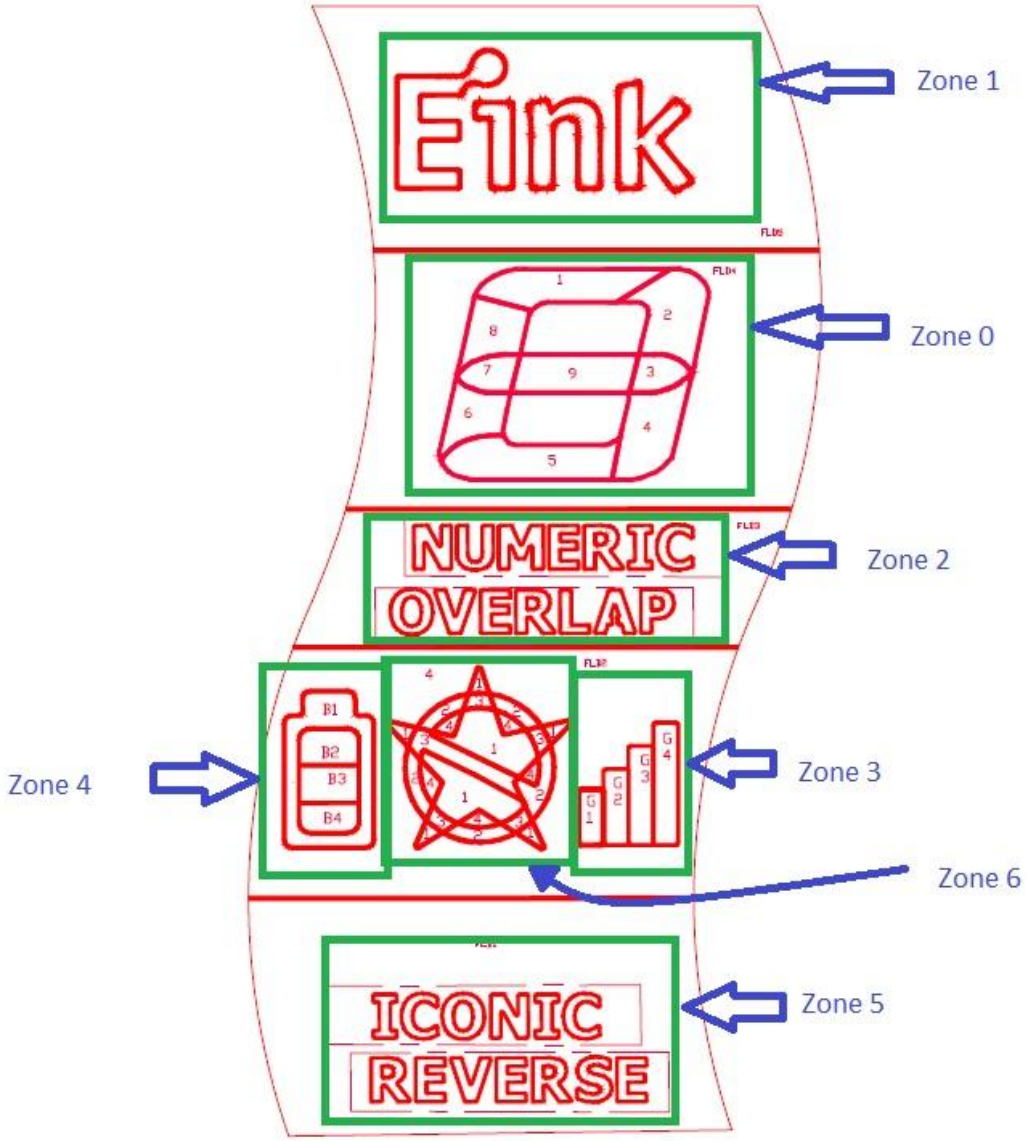
Make sure that you are utilizing atleast one driving waveform by defining either USING_PICKET_FENCE, USING_SLIDESHOWS or USING_GLOBAL_UPDATES.

Step 3: Testing: Causing an update

Once Step Step 1: Defining the segments and Step 2: Positioning the driver function and delay have been completed, set some of the elements of the nextDisplay array to 1 and set variable updateRequest to 1. NOTE: Ensure that your timer interrupt is running. This should result in a physical change on the EInk display screen. If needed, make a call to ClearDisplay to clear all segments on screen. This is preferred after power up.

Step 4: Using the presentation layer

The presentation layer provides a convenient way of selecting which segments to turn ON/OFF on the EInk screen for different purposes e.g. displaying numbers or alphabets. Each display can be logically viewed as consisting of different zones as shown in the figure below.



Each zone can be viewed as consisting of one or multiple digits; where each digit can have one or mutliple segments. For e.g. As seen in Figure 3, Zone 0 has 1 digit with 9 segments per digit. Zone 6 consists of 1 digit and 4 segments for each digit. Use this information for filling the arrays in YRPDSKRL78EINK_Display.c Once the arrays and the hexadecimal encodings are filled in, use a function to set up the einkScreenDef structure and call a function in main to set it up. The screen definition is now ready to be used by the presentation layer. You may now call the function displayInZone to set up the segments to turn ON or OFF in order to display a string on the EInk display screen.

Questions/Issues/Recommendations:

Feel free to contact us for suggestions/recommendation/bug reports at www.bnssolutions.com or am.renesas.com

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all documented files with brief descriptions:

Class Documentation

einkScreenDef Struct Reference

Structure for holding organized information about the EInk display screen.

`#include <rPresentation.h>`

Public Attributes

- unsigned char screenName [6]
- unsigned char numZones
- unsigned char * digitsPerZone
- unsigned char * segmentsPerDigit
- unsigned char numSegments
- unsigned char * segmentInfo
- unsigned char activeEncodingType
- unsigned char * activeEncoding
- unsigned char numHexEncodingsAvailable
- unsigned char * hexEncodingInfo
- unsigned char * hexEncodingMap [NUM_MAPPINGS]

Detailed Description

Structure for holding organized information about the EInk display screen.

Member Data Documentation

unsigned char* einkScreenDef::activeEncoding

Pointer to the last encoding used

unsigned char einkScreenDef::activeEncodingType

Used to save the last encoding type used (7,8,14 or 16 bit)

unsigned char* einkScreenDef::digitsPerZone

Identify number of digits in each zone

unsigned char* einkScreenDef::hexEncodingInfo

Identify the type of encoding (used during switching the type)

unsigned char* einkScreenDef::hexEncodingMap[NUM_MAPPINGS]

Hexadecimal Encoding for displaying a digit (7-8 bit)

unsigned char einkScreenDef::numHexEncodingsAvailable

Identify the number of encodings available(used during switching the type)

unsigned char einkScreenDef::numSegments

Used to specify number of segments used by the display

unsigned char einkScreenDef::numZones

Number of displayZones available on screen

unsigned char einkScreenDef::screenName[6]

Used to identify screen name

unsigned char* einkScreenDef::segmentInfo

Used to provide segment numbers in ascending order of zones and digit Left to right

unsigned char* einkScreenDef::segmentsPerDigit

Identify number of segments in each digit of a particular zone

---

The documentation for this struct was generated from the following file:

- C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/presentation/rPresentation.h

---

sw_flag_u Union Reference

A union for accessing the switch states in either bit mode or byte mode (Not relevant to the driver)

`#include <switches.h>`

Public Attributes

- uint8_t byteAccess
  Access information using a byte.
- struct {
- unsigned **sw1**: 1
- unsigned **sw1Hold**: 1
- unsigned **sw2**: 1
- unsigned **sw2Hold**: 1
- unsigned **sw3**: 1
- unsigned **sw3Hold**: 1
- unsigned **__pad0__**: 2
- };

Access information bitwise using a struct.

Detailed Description

A union for accessing the switch states in either bit mode or byte mode (Not relevant to the driver)

The documentation for this union was generated from the following file:

- C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/application/switches.h

---

tmr_flag_u Union Reference

A union to access timer information in either bitwise or byte mode (Only delayFsm may be related to timing).

`#include <timing.h>`

Public Attributes

- uint8_t byteAccess
  Access information as a byte.
- struct {
- unsigned **delayFsm**: 1
- unsigned **twoSec**: 1
- unsigned **demo**: 1
- unsigned **swScan**: 1
- unsigned **__pad0__**: 4
- };
  Access information bitwise using a structure.

Detailed Description

A union to access timer information in either bitwise or byte mode (Only delayFsm may be related to timing).

The documentation for this union was generated from the following file:

- C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/EInkDriverFiles/timing.h

---

File Documentation

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/application/application.c File Reference

Description: This file contains the application.

```
#include "r_cg_macrodriver.h"
#include "rEInkInclude.h"
#include "rPresentation.h"
#include "YRPDSKRL78EINK_Display.h"
#include "eink_demos.h"
#include "switches.h"
#include "application.h"
```

Functions

- void application (void)
  The function called from main to do all the tasks.

Detailed Description

Description: This file contains the application.

Filename: application.c Creation Date: October 29, 2012 Author: O. Raut

REVISION HISTORY:

Modified: ------ Date: ---------- Reason: ----------

Function Documentation

void application (void)

The function called from main to do all the tasks.

Description: This file contains the function prototype.

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/EInkDriverFiles/extern.h File Reference

Description: This file contains all the externalally defined variables that may be needed through out the project.

Variables

- uint8_t curDisplay [NUM_SEGMENTS]
  This array holds current segments that have been turned ON(1)/OFF(0)
- uint8_t nextDisplay [NUM_SEGMENTS]
  This array holds next segments that should be turned ON(1)/OFF(0)
- uint8_t updateRequest
  This variable, when set to true activates the driver and makes the physical update to the screen.
- uint8_t changingState

A variable that indicates that we are updating the display.

- uint8_t updateType
  The display waveform to use to update the display.
- uint8_t einkNextDir
  Next direction of drive. 0 is forward. 1 is reverse. (when driving the lines directly from the MCU pins)

Detailed Description

Description: This file contains all the externalally defined variables that may be needed through out the project.

Filename: extern.h

Creation Date: October 31, 2012

Author: O. Raut

REVISION HISTORY:

Modified: -------

Date: -----------

Reason: ----------

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/EInkDriverFiles/rEInkDriver.c File Reference

Description: C file that contains the state machine and functions required to control the EInk display screen using a Renesas microcontroller.

```
#include "rEInkInclude.h"
```

Functions

- void ClearDisplay (uint8_t dir)
  Set up the display array to blank.
- void initEInkHardware (void)
  Initializes the required peripherals; i.e. Output pin segments and peripheral timer. This function is part of the Hardware abstraction layer.
- void SetOutput (uint8_t *newData, uint8_t dir, uint8_t topPlane)
  This function sets the actual values of the segments. This function is part of the Hardware abstraction layer.
- static void ChangeEinkState (uint8_t next)
  This function changes the current state of the FSM to the one indicated in the argument.
- static void UpdateScreen (void)
  This function is called through the Finite state machine to manage the toggling of each pin connected to the EInk segmented display screens.
- void einkDriver (void)
  Finite state machine, called each time thru main. Used to update and keep the screen in check.

Variables

- const uint8_t allZeros [NUM_SEGMENTS] = {0}
  An array of all Zeroes (used to set pin voltage to 0)
- uint8_t changingState = 0
  A variable that indicates that we are updating the display.
- uint8_t updateNum
  A simple counter to make sure that a global update is done every few physical updates to the screen.
- static uint8_t curState
  A variable that indicates the state of the FSM.
- uint8_t nextState
  The next state of the FSM.
- static uint8_t updateStep = 0
  The current step in toggling the waveforms.
- uint8_t curDisplay [NUM_SEGMENTS] = {0}
  This array holds current segments that have been turned ON(1)/OFF(0)
- uint8_t nextDisplay [NUM_SEGMENTS] = {0}
  This array holds next segments that should be turned ON(1)/OFF(0)
- uint8_t updateType = EINK_PICKET_FENCE
  The display waveform to use to update the display.
- static uint8_t updateTypeDefault = EINK_PICKET_FENCE
  The default display waveform to use to update the display. The driver starts with this waveform and can then be changed using updateType.
- uint8_t updateRequest = FALSE
  This variable, when set to true activates the driver and makes the physical update to the screen.
- static uint8_t einkDir = BACKGROUND_WHITE
  Current direction of drive. 0 is forward. 1 is reverse. (when driving the lines directly from the MCU pins)
- uint8_t einkNextDir = BACKGROUND_WHITE
  Next direction of drive. 0 is forward. 1 is reverse. (when driving the lines directly from the MCU pins)

Detailed Description

Description: C file that contains the state machine and functions required to control the EInk display screen using a Renesas microcontroller.

Filename: rEInkDriver.c Creation Date: Nov 15, 2012 Original Author of Algorithm: T. Weisberger Modified and documented by: O. Raut

REVISION HISTORY: 07/24/2012 TSW: Added timer flag structure and added InitTimer to facilitate the structure.

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/EInkDriverFiles/rEInkDriver.h File Reference

Description:This file is the header file for the driver layer of the EInk driver.

```
#include "rEInkHardwareAbstraction.h"
```

Defines

- #define USING_PICKET_FENCE
  Defining this preprocessor definition will compile the code for using picket fence waveform.
- #define USING_SLIDESHOWS
  Defining this preprocessor definition will compile the code for using slideshow waveform.
- #define USING_GLOBAL_UPDATES
  Defining this preprocessor definition will compile the code for using global update waveform.
- #define TRUE (1)
  Compiler independent specification of TRUE.
- #define FALSE (0)
  Compiler independent specification of FALSE.
- #define NUM_SEGMENTS (36)
  The number of segments physically present on the hardware (not display)
- #define GLOBAL_REFRESH_HOLDOFF (10)
  Number of physical screen writes before a global update gets applied.
- #define BACKGROUND_WHITE (0)
  Just a preprocessor definition to make source file readable.
- #define BACKGROUND_BLACK (1)
  Just a preprocessor definition to make source file readable.
- #define NUM_FENCES (40)
  The number of toggles present in a picket fence update type waveform.
- #define DIR_OUTPUT_ON (0)
  MCU Specific value for turning a pin into input mode.
- #define DIR_OUTPUT_OFF (1)
  MCU Specific value for turning a pin into input mode.
- #define SEG_DELAY 200
  Specify the amount of delay used when using all other waveforms except PICKET_FENCE.
- #define PICKET_SHORT (450 / NUM_FENCES)
- #define TOP_PLANE EINK_P05
  Indicate which pin is connected to the common electrode.
- #define TOP_PLANE_DIR EINK_P05_DIR
  Related to the preprocessor definition TOP_PLANE.

Enumerations

- enum waveformType { **EINK_SLIDE_SHOW_FV**, **EINK_SLIDE_SHOW_RV**, **EINK_PICKET_FENCE**, **EINK_GLOBAL**, **EINK_INVERT** }
- *This enum contains the update waveform driving patterns. Each driving pattern has it's own application. For e.g.: SLIDE_SHOWs are good for driving icons where as PICKET_FENCE is good for driving alphanumeric digit.* enum driverState { **FSM_IDLE**, **FSM_WAIT**, **FSM_START_UPDATE**, **FSM_UPDATE_COMPLETE** }

This enum lists the different states that the driver can be in. Functions

- void ClearDisplay (uint8_t dir)
  Set up the display array to blank.
- void einkDriver (void)
  Finite state machine, called each time thru main. Used to update and keep the screen in check.

- void initEInkHardware (void)
  Initializes the required peripherals; i.e. Output pin segments and peripheral timer. This function is part of the Hardware abstraction layer.

Detailed Description

Description:This file is the header file for the driver layer of the EInk driver.

Filename: rEInkPresentation.h

Creation Date: October 31, 2012

Author: O. Raut

REVISION HISTORY:

Modified: -------

Date: -----------

Reason: ----------

Define Documentation

#define PICKET_SHORT (450 / NUM_FENCES)

brief Specify the time delay between toggling a line when using picket fence waveform.

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/EInkDriverFiles/rEinkInclude.h File Reference

Description: This file is the common include header file.

```
#include "r_cg_macrodriver.h"
#include "rEInkHardwareAbstraction.h"
#include "rEInkDriver.h"
#include "extern.h"
#include "timing.h"
```

Detailed Description

Description: This file is the common include header file.

Filename: rEInkInclude.h

Creation Date: October 31, 2012

Author: O. Raut

REVISION HISTORY:

Modified: -------

Date: -----------

Reason: ----------

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/presentation/rPresentation.c File Reference

Description:This file contains the source code for the state machine to drive an EInk segmented display.

```
#include "rEInkInclude.h"
#include "rPresentation.h"
#include <string.h>
#include <stdlib.h>
```

Functions

- uint8_t displayInZone (uint8_t zone, uint8_t *str, uint8_t formatOptions, struct einkScreenDef *thisScreen)
  This function is part of the presentation layer for the EInk Segmented display driver. It is responsible for setting up the next segments to turn ON or OFF by using screen definitions and information. This function cannot handle floating point instructions, however, there is a preprocessor statement which can be turned on to handle the same. Note that an updateRequest is not made through this function.

Variables

- uint8_t nextDisplay []
  This array holds next segments that should be turned ON(1)/OFF(0)

Detailed Description

Description:This file contains the source code for the state machine to drive an EInk segmented display.

Filename: rEInkDriver.c Creation Date: October 29, 2012 Author: O. Raut

REVISION HISTORY:

Modified: ------ Date: ---------- Reason: ----------

Function Documentation

uint8_t displayInZone (uint8_t *zone*, uint8_t * *str*, uint8_t *formatOptions*, struct einkScreenDef * *thisScreen*)

This function is part of the presentation layer for the EInk Segmented display driver. It is responsible for setting up the next segments to turn ON or OFF by using screen definitions and information. This function cannot handle floating point instructions, however, there is a preprocessor statement which can be turned on to handle the same. Note that an updateRequest is not made through this function.

Select correct active encoding here

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/presentation/rPresentation.h File Reference

Description:This file is the header file for the presentation layer of the EInk driver. Calling the function displayInZone is not necessary and the driver is independent of the same.

Classes

- struct einkScreenDef

Structure for holding organized information about the EInk display screen.

Defines

- #define FORMAT_ACTION (0xF0)
  A macro that list the actions to be performed on the arguement passed to the function for the purpose of being displayed. Refer function displayInZone().
- #define FORMAT_RETAIN_OTHER (0x20)
  Foramtting option that specifies that the status of the other segments should be maintained.
- #define FORMAT_LZ_BLANK (0x40)
  Foramtting option that specifies that all leading zeros within the string must be set to the value LCD_BLANK.
- #define LCD_BLANK (27)
  This preprocessor definition defines the index in the hexadecimal encoding look-up-table to use when blanking a character.
- #define NUM_MAPPINGS (4)
  This preprocessor definition defines the total number of hexadecimal encoding maps available to the presentation layer.

Enumerations

- enum { **NO_ERR** = 0, **ERR_NO_BYTES_IN_DISPLAY_BUFFER**, **ERR_NO_HEXENCODING_AVAILABLE** }

Functions

- uint8_t displayInZone (uint8_t zone, uint8_t *str, uint8_t count, struct einkScreenDef *thisScreen)
  This function is part of the presentation layer for the EInk Segmented display driver. It is responsible for setting up the next segments to turn ON or OFF by using screen definitions and information. This function cannot handle floating point instructions, however, there is a preprocessor statement which can be turned on to handle the same. Note that an updateRequest is not made through this function.

Detailed Description

Description:This file is the header file for the presentation layer of the EInk driver. Calling the function displayInZone is not necessary and the driver is independent of the same.

Filename: rEInkPresentation.h

Creation Date: October 31, 2012

Author: O. Raut

REVISION HISTORY:

Modified: -------

Date: -----------

Reason: ----------

Function Documentation

uint8_t displayInZone (uint8_t *zone*, uint8_t * *str*, uint8_t *count*, struct einkScreenDef * *thisScreen*)

This function is part of the presentation layer for the EInk Segmented display driver. It is responsible for setting up the next segments to turn ON or OFF by using screen definitions and information. This function cannot handle floating point instructions, however, there is a preprocessor statement which can be turned on to handle the same. Note that an updateRequest is not made through this function.

Select correct active encoding here

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/presentation/YRPDSKRL78EINK_Display.c File Reference

Description:This file contains the information that the presentation layer function can use to set up the array nextDisplay.

```
#include "rEInkInclude.h"
#include "rPresentation.h"
#include "YRPDSKRL78EINK_Display.h"
#include "r_cg_macrodriver.h"
```

Functions

- void setupeinkScr001 (void)
  Function to setup the EInk screen definition.

Variables

- struct einkScreenDef einkScr001
- uint8_t segmentsPerDigit []
- uint8_t digitsPerZone []
- uint8_t segmentInfo []
- uint8_t hexEncodingInfo []
- const uint8_t hexEncoding01 []
- const uint8_t hexEncoding02 []
- const uint8_t hexEncoding03 []
- const uint8_t hexEncoding04 []

Detailed Description

Description:This file contains the information that the presentation layer function can use to set up the array nextDisplay.

Filename: YRPDSKRL78EINK_Display.c

Creation Date: October 29, 2012

Author: O. Raut

REVISION HISTORY:

Modified: ------

Date: ----------

Reason: ----------

Variable Documentation

uint8_t digitsPerZone[]

```
Initial value:
{
    3,
    1,
    1,
    1,
    1,
    1,
    1,
    1,
}
```

Digit information per zone

struct einkScreenDef einkScr001

Structure defined to represent EInk screen on YRDKRL78G14

const uint8_t hexEncoding01[]

Hexadecimal encoding look up table 01

const uint8_t hexEncoding02[]

```
Initial value: {
    0x00,
    0x80,
    0x40,
    0xC0,
}
```

Hexadecimal encoding look up table 02

const uint8_t hexEncoding03[]

```
Initial value: {
    0x00,
    0x80,
```

```
    0x40,
    0xC0,
    0x20,
    0x60,
    0xA0,
    0xE0,

}
```

Hexadecimal encoding look up table 03

const uint8_t hexEncoding04[]

```
Initial value: {
    0x00,
    0x80,
}
```

Hexadecimal encoding look up table 04

uint8_t hexEncodingInfo[]

```
Initial value: {
    7,
    2,
    3,
    1,
}
```

Type of hexadecimal encoding

uint8_t segmentInfo[]

Segment information

uint8_t segmentsPerDigit[]

```
Initial value:
{
    7,
    2,
    2,
    3,
    1,
    1,
    1,
    1,
}
```

Segment information per digit

C:/Users/oraut/Documents/Workspace/Renesas/EInkDriverFinal/YLPDSKRL78EINK/user_src/presentation/YRPDSKRL78EINK_Display.h File Reference

Description:This file is the header to the screen definition file.

Functions

- void setupeinkScr001 (void)
  Function to setup the EInk screen definition.

Variables

- struct einkScreenDef einkScr001

Detailed Description

Description:This file is the header to the screen definition file.

Filename: YRPDSKRL78EINK_Display.h

Creation Date: October 31, 2012

Author: O. Raut

REVISION HISTORY:

Modified: -------

Date: ----------

Reason: ----------

Variable Documentation

struct einkScreenDef einkScr001

Structure defined to represent EInk screen on YRDKRL78G14

Index

INDEX