**button.c**

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* Project: HEIA-FR / Embedded Systems 1 Laboratory

\*

\* Purpose: Initialise the buttons s1 and test it

\*

\* Author: valentin Pahrisa

\* Date: 06.11.2017

\*/

#include "button.h"

#include <am335x\_gpio.h>

#define SW\_PIN 15

#define SW\_GPIO AM335X\_GPIO1

//Init method

void button\_init(){

am335x\_gpio\_init(SW\_GPIO);

am335x\_gpio\_setup\_pin\_in(SW\_GPIO, SW\_PIN, AM335X\_GPIO\_PULL\_NONE, true);

}

// Test the press

bool button\_s1\_is\_pressed(){

bool is\_pressed = !am335x\_gpio\_get\_state(SW\_GPIO, SW\_PIN);

return is\_pressed;

}

**button.h**

#pragma once

#ifndef BUTTON\_H

#define BUTTON\_H

#include <stdbool.h>

/\*\*

\* method to initialize the resoures of button

\*/

extern void button\_init();

/\*\*

\* method to restart counter when is pressed

\*/

extern bool button\_s1\_is\_pressed();

#endif

**dmtimer1.c**

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* This module is based on the software library developped by Texas Instruments

\* Incorporated - http://www.ti.com/ for its AM335x starter kit.

\*

\* Project: HEIA-FR / Embedded Systems 1+2 Laboratory

\*

\* Abstract: AM335x DTTimer1

\*

\* Purpose: This module implements basic services to drive the AM335x DMTimer1

\*

\* Author: Valentin Pharisa

\* Date: 04.12.2017

\*/

#include <stdbool.h>

#include <am335x\_clock.h>

#include "dmtimer1.h"

// -- Internal types and constant definition ----------------------------------

// DMTimer TIOCP\_CFG register bit definition

#define TIOCP\_CFG\_SOFTRESET (1<<1)

// DMTimer TISTAT register bit definition

#define TISTAT\_RESETDONE (1<<0)

// DMTimer TCLR register bit definition

#define TCLR\_ST (1<<0)

#define TCLR\_AR (1<<1)

/\*\*

\* DMTimer1 Register Definition

\*/

struct timer1\_ctrl {

uint32\_t tidr; // 00

uint32\_t res1[3]; // 04-0c

uint32\_t tiocp\_cfg; // 10

uint32\_t tistat; // 14

uint32\_t tisr; // 18

uint32\_t tier; // 1c

uint32\_t twer; // 20

uint32\_t tclr; // 24

uint32\_t tcrr; // 28

uint32\_t tldr; // 2c

uint32\_t ttgr; // 30

uint32\_t twps; // 34

uint32\_t tmar; // 38

uint32\_t tcar1; // 3c

uint32\_t tsicr; // 40

uint32\_t tcar2; // 44

uint32\_t tpir; // 48

uint32\_t tnir; // 4c

uint32\_t tcvr; // 50

uint32\_t tocr; // 54

uint32\_t towr; // 5c

};

/\*\*

\* DMTimer1 registers

\*/

// define the timer access variable here...

static volatile struct timer1\_ctrl\* timer1 = (volatile struct timer1\_ctrl\*) 0x44e31000;

static bool is\_initialized = false;

// -- Public methods definition -----------------------------------------------

void dmtimer1\_init()

{

static bool is\_initialized = false;

if (is\_initialized) return;

// configure DMTimer1 for internal use (resolution 24MHz)

am335x\_clock\_enable\_timer\_module (AM335X\_CLOCK\_TIMER1);

timer1-> tiocp\_cfg = TIOCP\_CFG\_SOFTRESET;

while ((timer1-> tistat & TISTAT\_RESETDONE) == 0);

timer1->tldr = 0;

timer1->tcrr = 0;

timer1->ttgr = 0;

timer1->tclr = TCLR\_AR | TCLR\_ST;

is\_initialized = true;

}

// ----------------------------------------------------------------------------

uint32\_t dmtimer1\_get\_counter()

{

return timer1->tcrr;

}

// ----------------------------------------------------------------------------

uint32\_t dmtimer1\_get\_frequency()

{

return 24000000;

}

**dmtimer1.h**

#pragma once

#ifndef DMTIMER1\_H

#define DMTIMER1\_H

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* This module is based on the software library developped by Texas Instruments

\* Incorporated - http://www.ti.com/ for its AM335x starter kit.

\*

\* Project: HEIA-FR / Embedded Systems 1+2 Laboratory

\*

\* Abstract: AM335x DTTimer1

\*

\* Purpose: This module implements basic services to drive the AM335x DMTimer1

\*

\* Author: Daniel Gachet

\* Date: 19.11.2017

\*/

#include <stdint.h>

/\*\*

\* method to initialize the AM335x Timer resources.

\* should be called prior any other methods.

\*/

extern void dmtimer1\_init();

/\*\*

\* method to get the current AM335x DMTimer1 counter value in tick

\*

\* @return counter value in ticks

\*/

extern uint32\_t dmtimer1\_get\_counter();

/\*\*

\* method to get the frequency of the timer source clock

\* @return timer clock frequency in Hz

\*/

extern uint32\_t dmtimer1\_get\_frequency();

#endif

**main.c**

**seg7.c**

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* Project: HEIA-FR / Embedded Systems 1 Laboratory

\*

\* Abstract: 7-Segment Display Device Driver

\*

\* Purpose: This module implements a method to diplay a value in range

\* of -99 to 99 on a two 7-segments display of the HEIA-FR

\* extension board of the Beaglebone black.

\*

\* Author: Valentin Pharisa

\* Date: 06.11.2017

\*/

#include <stdbool.h>

#include <am335x\_gpio.h>

#include "seg7.h"

// pin definition for 7-segment access

#define DIG\_GPIO AM335X\_GPIO2

#define DIG1 (1<<2)

#define DIG2 (1<<3)

#define DIG\_ALL (DIG1 | DIG2)

#define DP\_GPIO AM335X\_GPIO2

#define DP1 (1<<4)

#define DP2 (1<<5)

#define DP\_ALL (DP1 | DP2)

#define SEG\_GPIO AM335X\_GPIO0

#define SEG\_A (1<<4)

#define SEG\_B (1<<5)

#define SEG\_C (1<<14)

#define SEG\_D (1<<22)

#define SEG\_E (1<<23)

#define SEG\_F (1<<26)

#define SEG\_G (1<<27)

#define SEG\_ALL (SEG\_A | SEG\_B | SEG\_C | SEG\_D | SEG\_E | SEG\_F | SEG\_G)

// macro to compute number of elements of an array

#define ARRAY\_SIZE(x) (sizeof(x) / sizeof(x[0]))

// structure to initialize gpio pins used by 7-segment

static const struct gpio\_init {

enum am335x\_gpio\_modules module;

uint32\_t pin\_nr;

bool state;

} gpio\_init[] = {

{DIG\_GPIO, 2, false}, // DIG1

{DIG\_GPIO, 3, false}, // DIG2

{DP\_GPIO, 4, false}, // DP1

{DP\_GPIO, 5, false}, // DP2

{SEG\_GPIO, 4, false}, // SEGA

{SEG\_GPIO, 5, false}, // SEGB

{SEG\_GPIO, 14, false}, // SEGC

{SEG\_GPIO, 22, false}, // SEGD

{SEG\_GPIO, 23, false}, // SEGE

{SEG\_GPIO, 26, false}, // SEGF

{SEG\_GPIO, 27, false}, // SEGG

};

/\* 7-segment: segment definition

+-- SEG\_A --+

| |

SEG\_F SEG\_B

| |

+-- SEG\_G --+

| |

SEG\_E SEG\_C

| |

+-- SEG\_D --+

\*/

static const uint32\_t seg7[] = {

(SEG\_A | SEG\_B | SEG\_C | SEG\_D | SEG\_E | SEG\_F | 0 ), // 0

(0 | SEG\_B | SEG\_C | 0 | 0 | 0 | 0 ), // 1

(SEG\_A | SEG\_B | 0 | 0 | SEG\_E | SEG\_D | SEG\_G ), // 2

(SEG\_A | SEG\_B | SEG\_C | SEG\_D | 0 | 0 | SEG\_G ), // 3

(0 | SEG\_B | SEG\_C | 0 | 0 | SEG\_F | SEG\_G ), // 4

(SEG\_A | 0 | SEG\_C | SEG\_D | 0 | SEG\_F | SEG\_G ), // 5

(SEG\_A | 0 | SEG\_C | SEG\_D | SEG\_E | SEG\_F | SEG\_G ), // 6

(SEG\_A | SEG\_B | SEG\_C | 0 | 0 | 0 | 0 ), // 7

(SEG\_A | SEG\_B | SEG\_C | SEG\_D | SEG\_E | SEG\_F | SEG\_G ), // 8

(SEG\_A | SEG\_B | SEG\_C | SEG\_D | SEG\_F | SEG\_G | 0 ), // 9

(SEG\_A | SEG\_B | SEG\_C | SEG\_E | SEG\_F | SEG\_G | 0 ), // A

(0 | 0 | SEG\_C | SEG\_D | SEG\_E | SEG\_F | SEG\_G ), // b

(SEG\_A | 0 | 0 | SEG\_D | SEG\_E | SEG\_F | 0 ), // C

(0 | SEG\_B | SEG\_C | SEG\_D | SEG\_E | 0 | SEG\_G ), // d

(SEG\_A | 0 | 0 | SEG\_D | SEG\_E | SEG\_F | SEG\_G ), // E

(SEG\_A | 0 | 0 | 0 | SEG\_E | SEG\_F | SEG\_G ), // F

(0 ),

};

// 7-segment display: digit value

static struct digit {

uint32\_t seg7;

uint32\_t dot;

uint32\_t digit;

} display[2] = {

[0] = {.digit = DIG2,},

[1] = {.digit = DIG1,},

};

// -----------------------------------------------------------------------------

// implementation of public methods

// -----------------------------------------------------------------------------

void seg7\_init()

{

// initialize gpio modules

am335x\_gpio\_init(DIG\_GPIO);

am335x\_gpio\_init(DP\_GPIO);

am335x\_gpio\_init(SEG\_GPIO);

// configure gpio pins as output

for (int i=ARRAY\_SIZE(gpio\_init)-1; i>=0; i--) {

am335x\_gpio\_setup\_pin\_out(

gpio\_init[i].module,

gpio\_init[i].pin\_nr,

gpio\_init[i].state);

}

}

// -----------------------------------------------------------------------------

void seg7\_display\_value (int value)

{

uint32\_t dot = 0;

if (value <0){

dot = DP1;

value = -value;

}

display[0].seg7 = seg7[value%10];

display[1].seg7 = seg7[(value/10)%10];

display[1].dot= dot;

}

// -----------------------------------------------------------------------------

void seg7\_refresh\_display()

{

static unsigned digit = 0;

// turn off all segments

am335x\_gpio\_change\_states (DIG\_GPIO, DIG\_ALL, false);

am335x\_gpio\_change\_states (DP\_GPIO, DP\_ALL, false);

am335x\_gpio\_change\_states (SEG\_GPIO, SEG\_ALL, false);

//turn on segment for

am335x\_gpio\_change\_states (DIG\_GPIO, display[digit].digit, true);

am335x\_gpio\_change\_states (SEG\_GPIO, display[digit].seg7, true);

am335x\_gpio\_change\_states (DP\_GPIO, display[digit].dot, true);

digit = (digit+1) % 2;

}

**seg7.h**

#pragma once

#ifndef SEG7\_H

#define SEG7\_H

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* Project: HEIA-FR / Embedded Systems 1 Laboratory

\*

\* Abstract: 7-Segment Display Device Driver

\*

\* Purpose: This module implements a method to diplay a value in range

\* of -99 to 99 on a two 7-segments display of the HEIA-FR

\* extension board of the Beaglebone black.

\*

\* Author: Daniel Gachet

\* Date: 05.11.2017

\*/

/\*\*

\* method to initialize the resoures of the 7-segment display

\* this method shall be called prior any other.

\*/

extern void seg7\_init();

/\*\*

\* method to display a value [-99..99] on the the 7-segments display

\* for negative value, a dot will be displayed.

\*

\* @param value value to display

\*/

extern void seg7\_display\_value (int value);

/\*\*

\* method to refresh the 7-segment display

\*/

extern void seg7\_refresh\_display();

#endif

**wheel.c**

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* Project: HEIA-FR / Embedded Systems 1 Laboratory

\*

\* Purpose: Initiliaze the wheel s5 and detect which direction it take and the push s5 button

\*

\* Author: Valentin Pahrisa

\* Date: 06.11.2017

\*/

#include "wheel.h"

#include <am335x\_gpio.h>

#define SW\_PIN 2

#define SW\_GPIO AM335X\_GPIO0

#define CHA\_PIN 1

#define CHA\_GPIO AM335X\_GPIO2

#define CHB\_PIN 29

#define CHB\_GPIO AM335X\_GPIO1

static int former\_state = 0; // State old

static enum wheel\_direction transition[4][4]={

{WHEEL\_STILL, WHEEL\_STILL, WHEEL\_STILL, WHEEL\_STILL},

{WHEEL\_LEFT, WHEEL\_STILL, WHEEL\_STILL, WHEEL\_RIGHT},

{WHEEL\_RIGHT, WHEEL\_STILL, WHEEL\_STILL, WHEEL\_LEFT},

{WHEEL\_STILL, WHEEL\_STILL, WHEEL\_STILL, WHEEL\_STILL}

};

static int get\_new\_state(){

int new\_state = 0;

if(am335x\_gpio\_get\_state(CHA\_GPIO, CHA\_PIN)) new\_state +=1;

if(am335x\_gpio\_get\_state(CHB\_GPIO, CHB\_PIN)) new\_state +=2;

return new\_state;

}

void wheel\_init(){

// Initialise GPIO modules

am335x\_gpio\_init(SW\_GPIO);

am335x\_gpio\_init(CHA\_GPIO);

am335x\_gpio\_init(CHB\_GPIO);

am335x\_gpio\_setup\_pin\_in(SW\_GPIO, SW\_PIN, AM335X\_GPIO\_PULL\_NONE, true);

am335x\_gpio\_setup\_pin\_in(CHA\_GPIO, CHA\_PIN, AM335X\_GPIO\_PULL\_NONE, true);

am335x\_gpio\_setup\_pin\_in(CHB\_GPIO, CHB\_PIN, AM335X\_GPIO\_PULL\_NONE, true);

former\_state = get\_new\_state();

}

// Get the wheel state

enum wheel\_direction wheel\_get\_direction(){

int new\_state = get\_new\_state();

enum wheel\_direction dir = transition[former\_state][new\_state]; // new direction in tab

former\_state = new\_state; // old state becom new

return dir;

}

// test if the wheel button s5 is pressed

bool wheel\_button\_is\_pressed(){

bool is\_pressed = !am335x\_gpio\_get\_state(SW\_GPIO, SW\_PIN);

return is\_pressed;

}

**wheel.h**

#pragma once

#ifndef WHEEL\_H

#define WHEEL\_H

/\*\*

\* Copyright 2017 University of Applied Sciences Western Switzerland / Fribourg

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*

\* Project: HEIA-FR / Embedded Systems 1 Laboratory

\*

\* Abstract: 7-Segment Display Device Driver

\*

\* Purpose: This module implements a method to diplay a value in range

\* of -99 to 99 on a two 7-segments display of the HEIA-FR

\* extension board of the Beaglebone black.

\*

\* Author: valentin Pahrisa

\* Date: 06.11.2017

\*/

#include <stdbool.h>

enum wheel\_direction {WHEEL\_STILL, WHEEL\_RIGHT, WHEEL\_LEFT};

/\*\*

\* method to initialize the wheel button

\*/

extern void wheel\_init();

/\*\*

\* method to say wich direction we turn the weel

\*/

extern enum wheel\_direction wheel\_get\_direction();

/\*\*

\* method to say when button is pressed (Not uses)

\*/

extern bool wheel\_button\_is\_pressed();

#endif