IndInf01: Ampelsteuerung

Stefan Erceg
Version 1.0
01.10.2015

Module Index

Modules

Here is a list of all modules:

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Module Documentation

CMSIS

Modules

- **Stm32f3xx_system**

Detailed Description

Stm32f3xx_system

Modules

- **STM32F3xx_System_Private_Includes**
- **STM32F3xx_System_Private_TypesDefinitions**
- **STM32F3xx_System_Private_Defines**
- **STM32F3xx_System_Private_Macros**
- **STM32F3xx_System_Private_Variables**
- **STM32F3xx_System_Private_FunctionPrototypes**
- **STM32F3xx_System_Private_Functions**

Detailed Description

STM32F3xx_System_Private_Includes

STM32F3xx_System_Private_TypesDefinitions

STM32F3xx_System_Private_Defines

Macros

- #define **HSE_VALUE** ((uint32_t)8000000)
- #define **HSI_VALUE** ((uint32_t)8000000)
- #define **VECT_TAB_OFFSET** 0x0

Detailed Description

Macro Definition Documentation

#define HSE_VALUE ((uint32_t)8000000)

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

#define HSI_VALUE ((uint32_t)8000000)

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

#define VECT_TAB_OFFSET 0x0

< Uncomment the following line if you need to relocate your vector Table in Internal SRAM. Vector Table base offset field. This value must be a multiple of 0x200.

STM32F3xx_System_Private_Macros

STM32F3xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 8000000
- __IO const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}

Detailed Description

Variable Documentation

__IO const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}

uint32_t SystemCoreClock = 8000000

STM32F3xx_System_Private_FunctionPrototypes

STM32F3xx_System_Private_Functions

Functions

- void **SystemInit** (void)
  Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.
- void **SystemCoreClockUpdate** (void)
  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Detailed Description

Function Documentation

void SystemCoreClockUpdate (void)

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note:

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.
- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the **HSI_VALUE(*)**
- If SYSCLK source is HSE, SystemCoreClock will contain the **HSE_VALUE(**)**
- If SYSCLK source is PLL, SystemCoreClock will contain the **HSE_VALUE(**)** or **HSI_VALUE(*)** multiplied/divided by the PLL factors.

(*) HSI_VALUE is a constant defined in stm32f3xx_hal.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(**) HSE_VALUE is a constant defined in stm32f3xx_hal.h file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters:

None	

Return values:

None	

void SystemInit (void)

Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.

Parameters:

None	

Return values:

None	

Data Structure Documentation

currentTrafficLight Struct Reference

Struct zur Speicherung des aktuellen Zustands und des aktuellen Events.

`#include <state_machine.h>`

Data Fields

- **LEDstate currentState**
- **LEDevent currentEvent**

Detailed Description

Struct zur Speicherung des aktuellen Zustands und des aktuellen Events.

Der aktuelle Zustand und das aktuelle Event werden hier gespeichert.

Field Documentation

LEDevent currentTrafficLight::currentEvent

LEDstate currentTrafficLight::currentState

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

- C:/Users/Stipe/workspace_5bhitt/IndInf01/src/**state_machine.h**

File Documentation

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/event_centric.c File Reference

Umsetzung einer Event-Centric State Machine.

```
#include "state_machine.h"
```

Functions

- void **trafficLightSystemWithEvent** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer Event-Centric State Machine.

Detailed Description

In diesem File existiert die Funktion, in welcher die Zustaende der LEDs und die Events mittels Umsetzung einer Event-Centric State Machine in einem switch-case Konstrukt verwaltet werden.

Author:

Stefan Erceg

Version:

20151001

Function Documentation

void trafficLightSystemWithEvent (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer Event-Centric State Machine.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer Event-Centric State Machine wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der das Event zu Beginn und in einer if-Abfrage der Status geprueft werden. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/LED_States.c File Reference

Hier werden die verschiedenen Zustaende der LEDs verwaltet.

```
#include "stm32f3xx.h"
#include "stm32f3_discovery.h"
#include "LED_States.h"
```

Functions

- void **resetAllLEDs** ()
  Alle LEDs werden auf "Off" gesetzt.
- void **setRedLED** ()
  Die rote LED wird aktiviert.
- void **setRedYellowLEDs** ()
  Die rote und gelbe LED werden aktiviert.
- void **setGreenLED** ()
  Die gruene LED wird aktiviert.
- void **setBlinkingGreenLED** ()
  Die gruene LED wird auf blinkend gesetzt.
- void **setYellowLED** ()
  Die gelbe LED wird aktiviert.
- void **setBlinkingYellowLED** ()
  Die gelbe LED wird auf blinkend gesetzt.

Detailed Description

In diesem File existieren Funktionen, die fuer das Leuchten der spezifischen LEDs zustaendig sind. Zu Beginn jeder Set-Funktion werden alle LEDs reseted.

Author:
Stefan Erceg

Version:
20150928

Function Documentation

void resetAllLEDs ()

Alle LEDs werden auf "Off" gesetzt.

Diese Funktion dient zum Ausschalten aller LEDs und wird zu Beginn aller setLED-Funktionen aufgerufen.

void setBlinkingGreenLED ()

Die gruene LED wird auf blinkend gesetzt.

In dieser Funktion wird die gruene LED mittels einer for-Schleife auf blinkend gesetzt. Nachdem die gruene LED ein- bzw. ausgeschaltet wird, wird eine Verzoegerung von 0,5 Sekunden eingestellt.

void setBlinkingYellowLED ()

Die gelbe LED wird auf blinkend gesetzt.

In dieser Funktion wird die gelbe LED mittels einer for-Schleife auf blinkend gesetzt. Nachdem die gelbe LED ein- bzw. ausgeschaltet wird, wird eine Verzoegerung von 0,5 Sekunden eingestellt.

void setGreenLED ()

Die gruene LED wird aktiviert.

In dieser Funktion wird die gruene LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setRedLED ()

Die rote LED wird aktiviert.

In dieser Funktion wird die rote LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setRedYellowLEDs ()

Die rote und gelbe LED werden aktiviert.

In dieser Funktion werden die rote und gelbe LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setYellowLED ()

Die gelbe LED wird aktiviert.

In dieser Funktion wird die gelbe LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/LED_States.h File Reference

Hier werden die verschiedenen Funktionen zum Verwalten der Zustaende der LEDs definiert.

Functions

- void **resetAllLEDs** ()
  Alle LEDs werden auf "Off" gesetzt.
- void **setRedLED** ()
  Die rote LED wird aktiviert.
- void **setRedYellowLEDs** ()
  Die rote und gelbe LED werden aktiviert.
- void **setGreenLED** ()
  Die gruene LED wird aktiviert.
- void **setBlinkingGreenLED** ()
  Die gruene LED wird auf blinkend gesetzt.
- void **setYellowLED** ()
  Die gelbe LED wird aktiviert.
- void **setBlinkingYellowLED** ()
  Die gelbe LED wird auf blinkend gesetzt.

Detailed Description

In dem Header-File werden alle Funktionen definiert, die fuer das Leuchten der spezifischen LEDs zustaendig sind. Ebenfalls existiert eine Funktion zum Reseten der LEDs.

Author:

Stefan Erceg

Version:

20150928

Function Documentation

void resetAllLEDs ()

Alle LEDs werden auf "Off" gesetzt.

Diese Funktion dient zum Ausschalten aller LEDs und wird zu Beginn aller setLED-Funktionen aufgerufen.

void setBlinkingGreenLED ()

Die gruene LED wird auf blinkend gesetzt.

In dieser Funktion wird die gruene LED mittels einer for-Schleife auf blinkend gesetzt. Nachdem die gruene LED ein- bzw. ausgeschaltet wird, wird eine Verzoegerung von 0,5 Sekunden eingestellt.

void setBlinkingYellowLED ()

Die gelbe LED wird auf blinkend gesetzt.

In dieser Funktion wird die gelbe LED mittels einer for-Schleife auf blinkend gesetzt. Nachdem die gelbe LED ein- bzw. ausgeschaltet wird, wird eine Verzoegerung von 0,5 Sekunden eingestellt.

void setGreenLED ()

Die gruene LED wird aktiviert.

In dieser Funktion wird die gruene LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setRedLED ()

Die rote LED wird aktiviert.

In dieser Funktion wird die rote LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setRedYellowLEDs ()

Die rote und gelbe LED werden aktiviert.

In dieser Funktion werden die rote und gelbe LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

void setYellowLED ()

Die gelbe LED wird aktiviert.

In dieser Funktion wird die gelbe LED aktiviert und danach eine Verzoegerung von 2 Sekunden eingestellt.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/main.c File Reference

Dieses File ruft die Funktion einer bestimmten State auf.

```
#include "stm32f3xx.h"
#include "stm32f3_discovery.h"
#include "state_machine.h"
```

Functions

- int **main** (void)
  Hier erfolgt die Initialisierung des STMs und der LEDs.

Detailed Description

In diesem File ist die main-Funktion vorhanden, in welcher der STM und die LEDs initialisiert und die Funktion einer bestimmten State Machine (defaultmaessig: Event Centric State Machine) aufgerufen wird.

Author:
Stefan Erceg

Version:
20150928

Function Documentation

int main (void)

Hier erfolgt die Initialisierung des STMs und der LEDs.

In der Main-Funktion werden der STM und die LEDs initialisiert und die Funktion einer bestimmten State Machine (defaultmaessig: Event Centric State Machine) aufgerufen.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/state_centric.c File Reference

Umsetzung einer State-Centric State Machine.

```
#include "state_machine.h"
```

Functions

- void **trafficLightSystem** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer State-Centric State Machine.

Detailed Description

In diesem File existiert die Funktion, in welcher die Zustaende der LEDs und die Events mittels Umsetzung einer State-Centric State Machine in einem switch-case Konstrukt verwaltet werden.

Author:
Stefan Erceg

Version:
20151001

Function Documentation

void trafficLightSystem (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer State-Centric State Machine.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer State-Centric State Machine wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der der Status zu Beginn und in einer if-Abfrage das Event geprueft werden. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/state_centric_with_hidden_transitions.c File Reference

Umsetzung einer State-Centric State Machine With Hidden Transitions.

```
#include "state_machine.h"
```

Functions

- void **trafficLightSystemWithTransition** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer State-Centric State Machine With Hidden Transitions.

Detailed Description

In diesem File existiert die Funktion, in welcher die Zustaende der LEDs und die Events mittels Umsetzung einer State-Centric State Machine With Hidden Transitions in einem switch-case Konstrukt verwaltet werden.

Author:
Stefan Erceg

Version:
20151001

Function Documentation

void trafficLightSystemWithTransition (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer State-Centric State Machine With Hidden Transitions.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer State-Centric State Machine With Hidden Transitions wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der (nur) der Status geprueft wird. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/state_machine.h File Reference

Hier existieren Enums und Structs zum Definieren der Zustaende der LEDs und der Events.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "LED_States.h"
```

Data Structures

- struct **currentTrafficLight**

Struct zur Speicherung des aktuellen Zustands und des aktuellen Events.

Enumerations

- enum **LEDstate** { **RED**, **RED_YELLOW**, **GREEN**, **GREEN_BLINK**, **YELLOW**, **YELLOW_BLINK** }*Enums fuer die verschiedenenen Zustaende der LEDs werden erstellt.*
- enum **LEDevent** { **STOP**, **PREPAREFORGOING**, **GO**, **PREPAREFORWAITING**, **CAUTION**, **ERR** }*Enums fuer die verschiedenenen Events werden erstellt.*

Functions

- void **trafficLightSystem** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer State-Centric State Machine.
- void **trafficLightSystemWithTransition** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer State-Centric State Machine With Hidden Transitions.
- void **trafficLightSystemWithEvent** (**currentTrafficLight** *t_light)
  Funktion zum Ausfuehren einer Event-Centric State Machine.

Detailed Description

In dem Header-File werden Enums fuer die Zustaende der LEDs und die Events erstellt. Ebenfalls existiert ein Struct, in dem der aktuelle Zustand und das aktuelle Event gespeichert werden.

Author:
Stefan Erceg

Version:
20150928

Enumeration Type Documentation

enum LEDevent

Enums fuer die verschiedenenen Events werden erstellt.

Alle moeglichen Events fuer die LEDs werden hier definiert.

Enumerator

STOP Enum fuer das Event "gelb zu rot"

PREPAREFORGOING Enum fuer das Event "rot zu rot-gelb"

GO Enum fuer das Event "rot-gelb zu gruen"

PREPAREFORWAITING Enum fuer das Event "gruen zu gruen-blinkend"

CAUTION Enum fuer das Event "gruen-blinkend zu gelb"

ERR Enum fuer den Status "Error" falls etwas schief gegangen ist

enum LEDstate

Enums fuer die verschiedenenen Zustaende der LEDs werden erstellt.

Alle moeglichen Zustaende der LEDs, wie z.B. nur rote LED aktiv oder rote und gelbe LEDs aktiv, werden hier definiert.

Enumerator

RED Enum fuer die rote LED

RED_YELLOW Enum fuer die rote und gelbe LED

GREEN Enum fuer die gruene LED

GREEN_BLINK Enum fuer die blinkende gruene LED

YELLOW Enum fuer die gelbe LED

YELLOW_BLINK Enum fuer die blinkende gelbe LED

Function Documentation

void trafficLightSystem (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer State-Centric State Machine.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer State-Centric State Machine wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der der Status zu Beginn und in einer if-Abfrage das Event geprueft werden. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

void trafficLightSystemWithEvent (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer Event-Centric State Machine.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer Event-Centric State Machine wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der das Event zu Beginn und in einer if-Abfrage der Status geprueft werden. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

void trafficLightSystemWithTransition (currentTrafficLight * *t_light*)

Funktion zum Ausfuehren einer State-Centric State Machine With Hidden Transitions.

Parameters:

t_light	dient zum Verwalten des aktuellen Zustands der LED und des aktuellen Events

Das Konzept einer State-Centric State Machine With Hidden Transitions wird hier umgesetzt. Dabei wird ein switch-case Konstrukt aufgebaut, bei der (nur) der Status geprueft wird. Der Status aendert sich, sobald das spezifische LED Delay ausgelaufen ist.

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/stm32f3xx_it.c File Reference

Default Interrupt Service Routines.

```
#include "stm32f3xx_hal.h"
#include "stm32f3xx.h"
#include "stm32f3xx_it.h"
```

Functions

- void **SysTick_Handler** (void)
  This function handles SysTick Handler.

Detailed Description

Default Interrupt Service Routines.

Author:
Ac6

Version:
V1.0

Date:
02-Feb-2015

Function Documentation

void SysTick_Handler (void)

This function handles SysTick Handler.

Parameters:

None	

Return values:

None	

C:/Users/Stipe/workspace_5bhitt/IndInf01/src/system_stm32f3xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

```
#include "stm32f3xx.h"
```

Macros

- #define **HSE_VALUE** ((uint32_t)8000000)
- #define **HSI_VALUE** ((uint32_t)8000000)
- #define **VECT_TAB_OFFSET** 0x0

Functions

- void **SystemInit** (void)
  Setup the microcontroller system Initialize the FPU setting, vector table location and the PLL configuration is reset.
- void **SystemCoreClockUpdate** (void)
  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 8000000
- __IO const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}

Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

Author:

MCD Application Team

Version:

V1.2.0

Date:

19-June-2015

1. This file provides two functions and one global variable to be called from user application:
   - **SystemInit()**: This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f3xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - **SystemCoreClockUpdate()**: Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then **SystemInit()** function is called, in "startup_stm32f3xx.s" file, to configure the system clock before to branch to main program.

3. This file configures the system clock as follows:

Supported STM32F3xx device

System Clock source | HSI

SYSCLK(Hz) | 8000000

HCLK(Hz) | 8000000

AHB Prescaler | 1

APB2 Prescaler | 1

APB1 Prescaler | 1

USB Clock | DISABLE

==========

Attention:

Index

INDEX