IndInf03: Ampelsteuerung mit Interrupts

Stefan Erceg
Version 1.0
30.10.2015

Module Index

Modules

Here is a list of all modules:

- CMSIS 5
  - Stm32f3xx_system 5
    - STM32F3xx_System_Private_Includes 5
    - STM32F3xx_System_Private_TypesDefinitions 5
    - STM32F3xx_System_Private_Defines 5
    - STM32F3xx_System_Private_Macros 6
    - STM32F3xx_System_Private_Variables 6
    - STM32F3xx_System_Private_FunctionPrototypes 6
    - STM32F3xx_System_Private_Functions 7

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

currentTrafficLight (Struct zur Speicherung des aktuellen Zustands und des aktuellen Events) 9

File Index

File List

Here is a list of all files with brief descriptions:

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/LED_States.c (Hier werden die verschiedenen Zustaende der LEDs verwaltet) 10

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/LED_States.h (Hier werden die verschiedenen Funktionen zum Verwalten der Zustaende der LEDs definiert) 13

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/main.c (Dieses File verwaltet die Interrupts und beinhaltet die Main-Funktion) 16

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/state_centric.c (Umsetzung einer State-Centric State Machine) 18

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/state_machine.h (Hier existieren Enums und Structs zum Definieren der Zustaende der LEDs und der Events) 19

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/stm32f3xx_it.c (Default Interrupt Service Routines) 21

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/system_stm32f3xx.c (CMSIS Cortex-M4 Device Peripheral Access Layer System Source File) 22

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**
- int **blink_counter**
- bool **interrupted**
- bool **night**

Detailed Description

Struct zur Speicherung des aktuellen Zustands und des aktuellen Events.

Der aktuelle Zustand und das aktuelle Event werden hier gespeichert. Ebenfalls werden hier Variablen fuer den Counter und die Zustaende "Unterbrochen" und "Nachtmodus" abgelegt.

Field Documentation

int currentTrafficLight::blink_counter

Variable fuer den Counter

LEDevent currentTrafficLight::currentEvent

Variable fuer das aktuelle Event

LEDstate currentTrafficLight::currentState

Variable fuer den aktuellen Zustand

bool currentTrafficLight::interrupted

Variable fuer die Ueberprufeung "Unterbrochen" oder nicht

bool currentTrafficLight::night

Variable fuer die Ueberprufeung "Nachtmodus aktiviert" oder nicht

---

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

- C:/Users/Stipe/workspace_5bhitt/IndInf02/src/**state_machine.h**

File Documentation

C:/Users/Stipe/workspace_5bhitt/IndInf02/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 **toggleRedLED** ()
  Die rote LED wird hier umgeschaltet.
- void **setGreenLED** ()
  Die gruene LED wird aktiviert.
- void **setBlinkingGreenLED** ()
  Die gruene LED wird auf blinkend gesetzt.
- void **toggleGreenLED** ()
  Die gruene LED wird hier umgeschaltet.
- void **setYellowLED** ()
  Die gelbe LED wird aktiviert.
- void **setBlinkingYellowLED** ()
  Die gelbe LED wird auf blinkend gesetzt.
- void **toggleYellowLED** ()
  Die gelbe LED wird hier umgeschaltet.

Detailed Description

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

Author:
Stefan Erceg

Version:
20151030

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 auf- und abgedreht. Davor werden alle LEDs auf "Off" gesetzt.

void setBlinkingYellowLED ()

Die gelbe LED wird auf blinkend gesetzt.

In dieser Funktion wird die gelbe LED auf- und abgedreht. Davor werden alle LEDs auf "Off" gesetzt.

void setGreenLED ()

Die gruene LED wird aktiviert.

In dieser Funktion wird die gruene LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setRedLED ()

Die rote LED wird aktiviert.

In dieser Funktion wird die rote LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setRedYellowLEDs ()

Die rote und gelbe LED werden aktiviert.

In dieser Funktion werden die rote und gelbe LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setYellowLED ()

Die gelbe LED wird aktiviert.

In dieser Funktion wird die gelbe LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void toggleGreenLED ()

Die gruene LED wird hier umgeschaltet.

In dieser Funktion wird die gruene LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

void toggleRedLED ()

Die rote LED wird hier umgeschaltet.

In dieser Funktion wird die rote LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

void toggleYellowLED ()

Die gelbe LED wird hier umgeschaltet.

In dieser Funktion wird die gelbe LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

C:/Users/Stipe/workspace_5bhitt/IndInf02/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 **toggleRedLED** ()
  Die rote LED wird hier umgeschaltet.
- void **setGreenLED** ()
  Die gruene LED wird aktiviert.
- void **setBlinkingGreenLED** ()
  Die gruene LED wird auf blinkend gesetzt.
- void **toggleGreenLED** ()
  Die gruene LED wird hier umgeschaltet.
- void **setYellowLED** ()
  Die gelbe LED wird aktiviert.
- void **setBlinkingYellowLED** ()
  Die gelbe LED wird auf blinkend gesetzt.
- void **toggleYellowLED** ()
  Die gelbe LED wird hier umgeschaltet.

Detailed Description

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

Author:

Stefan Erceg

Version:

20151030

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 auf- und abgedreht. Davor werden alle LEDs auf "Off" gesetzt.

void setBlinkingYellowLED ()

Die gelbe LED wird auf blinkend gesetzt.

In dieser Funktion wird die gelbe LED auf- und abgedreht. Davor werden alle LEDs auf "Off" gesetzt.

void setGreenLED ()

Die gruene LED wird aktiviert.

In dieser Funktion wird die gruene LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setRedLED ()

Die rote LED wird aktiviert.

In dieser Funktion wird die rote LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setRedYellowLEDs ()

Die rote und gelbe LED werden aktiviert.

In dieser Funktion werden die rote und gelbe LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void setYellowLED ()

Die gelbe LED wird aktiviert.

In dieser Funktion wird die gelbe LED aktiviert. Davor werden alle LEDs auf "Off" gesetzt.

void toggleGreenLED ()

Die gruene LED wird hier umgeschaltet.

In dieser Funktion wird die gruene LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

void toggleRedLED ()

Die rote LED wird hier umgeschaltet.

In dieser Funktion wird die rote LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

void toggleYellowLED ()

Die gelbe LED wird hier umgeschaltet.

In dieser Funktion wird die gelbe LED umgeschaltet. Dazu wird die Funktion "BSP_LED_Toggle" aufgerufen.

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/main.c File Reference

Dieses File verwaltet die Interrupts und beinhaltet die Main-Funktion.

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

Functions

- void **EXTI0_Config** ()
  Die Funktion dient zum Konfigurieren der Interrupts.
- int **main** (void)
- void **HAL_GPIO_EXTI_Callback** (uint16_t GPIO_Pin)
  Die Funktion dient zum Verwalten des Zustandes, nachdem der User-Button gedrueckt wurde.
- void **HAL_SYSTICK_Callback** (void)
  Die Funktion dient zum Ausfuehren der Methode, welche eine State-Centric State Machine umsetzt.

Variables

- **currentTrafficLight trafficLight**
- **currentTrafficLight * pointerForTrafficLight**

Detailed Description

In diesem File ist einerseits die Main-Funktion vorhanden, welche den STM und die LEDs initialisiert und Default-Werte fuer die Variablen setzt. Anderseits existieren hier ebenfalls Funktionen zum Verwalten der Interrupts.

Author:
Stefan Erceg

Version:
20151030

Function Documentation

void EXTI0_Config (void)

Die Funktion dient zum Konfigurieren der Interrupts.

In dieser Funktion werden alle fuer die Interrupts relevanten Eigenschaften gesetzt. Es werden daher die GPIO-Clock aktiviert, der User-Button konfiguriert und die Prioritaet fuer einen bestimmten Interrupt definiert.

void HAL_GPIO_EXTI_Callback (uint16_t *GPIO_Pin*)

Die Funktion dient zum Verwalten des Zustandes, nachdem der User-Button gedrueckt wurde.

In dieser Funktion wird definiert, was das Programm ausfuehren soll, nachdem der User-Button gedrueckt wurde. Dabei wird ueberprueft, ob der Nachtmodus aktiviert ist oder nicht und je nachdem wird der Zustand der LEDs verwaltet.

void HAL_SYSTICK_Callback (void)

Die Funktion dient zum Ausfuehren der Methode, welche eine State-Centric State Machine umsetzt.

Diese Funktion wird jede Millisekunde aufgerufen. Dabei wird die Methode, welche die State-Centric State Machine umsetzt, ausgefuehrt.

int main (void)

Hier erfolgt unter anderem die Initialisierung des STMs und der LEDs.

In der main-Funktion werden der STM und die LEDs initialisiert. Ebenfalls wird die EXTI0_Config-Methode aufgerufen und Default-Werte fuer die Variablen des Structs, der unter anderem den aktuellen Zustand der LED speichert, gesetzt.

Variable Documentation

currentTrafficLight* pointerForTrafficLight

wird beim Aufruf der State Machine als Parameter uebergeben

currentTrafficLight trafficLight

stellt die Struct zum Verwalten der aktuellen Zustaende dar

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/state_centric.c File Reference

Umsetzung einer State-Centric State Machine.

```
#include <stdbool.h>
#include "LED_States.h"
#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:

20151030

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/IndInf02/src/state_machine.h File Reference

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

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

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:

20151030

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.

C:/Users/Stipe/workspace_5bhitt/IndInf02/src/stm32f3xx_it.c File Reference

Default Interrupt Service Routines.

```
#include "stm32f3xx_hal.h"
#include "stm32f3xx.h"
#include "stm32f3xx_it.h"
#include "stm32f3_discovery.h"
```

Functions

- void **SysTick_Handler** (void)
  This function handles SysTick Handler.
- void **EXTI0_IRQHandler** (void)

Detailed Description

Default Interrupt Service Routines.

Author:
Ac6

Version:
V1.0

Date:
02-Feb-2015

Function Documentation

void EXTI0_IRQHandler (void)

void SysTick_Handler (void)

This function handles SysTick Handler.

Parameters:

None	

Return values:

None	

C:/Users/Stipe/workspace_5bhitt/IndInf02/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:

© COPYRIGHT(c) 2015 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics 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 HOLDER 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.

Index

INDEX