### ################################################################################################################################

### LED

### ################################################################################################################################

### System\_stm32f10x.c

### /\*

### \* Istenen Kod 1.katman ama bunlara c dosyalarini kullanmadan kendin ulasmaya çalis

### \* Yani birinci katmani katmansiz hale getir.

### \*/

### ////////////////////////////////////////////////////////////////////////////////////////////

### 1.Katman

### ////////////////////////////////////////////////////////////////////////////////////////////

### int main(void)

### {

### EXS03\_initGPIO();//Giris Cikilari Ayarla

### void EXS03\_initGPIO(void)

{

### GPIO\_InitTypeDef GPIO\_InitStructure; //gpio.h dosyasına geçti.

### /\*\*

### \* @brief GPIO Init structure definition

### \*/

### typedef struct

### {

### uint16\_t GPIO\_Pin; /\*!< Specifies the GPIO pins to be configured.

### This parameter can be any value of @ref GPIO\_pins\_define \*/

### GPIOSpeed\_TypeDef GPIO\_Speed; /\*!< Specifies the speed for the selected pins.

### This parameter can be a value of @ref GPIOSpeed\_TypeDef \*/

### GPIOMode\_TypeDef GPIO\_Mode; /\*!< Specifies the operating mode for the selected pins.

### This parameter can be a value of @ref GPIOMode\_TypeDef \*/

### }GPIO\_InitTypeDef; //InitTypedef Burada

/\*\*

\* @brief Output Maximum frequency selection

\*/

typedef enum

{

GPIO\_Speed\_10MHz = 1,

GPIO\_Speed\_2MHz,

GPIO\_Speed\_50MHz

}**GPIOSpeed\_TypeDef;**

#define IS\_GPIO\_SPEED(SPEED) (((SPEED) == GPIO\_Speed\_10MHz) || ((SPEED) == GPIO\_Speed\_2MHz) || \

((SPEED) == GPIO\_Speed\_50MHz))

/\*\*

\* @brief Configuration Mode enumeration

\*/

typedef enum

{ GPIO\_Mode\_AIN = 0x0,

GPIO\_Mode\_IN\_FLOATING = 0x04,

GPIO\_Mode\_IPD = 0x28,

GPIO\_Mode\_IPU = 0x48,

GPIO\_Mode\_Out\_OD = 0x14,

GPIO\_Mode\_Out\_PP = 0x10,

GPIO\_Mode\_AF\_OD = 0x1C,

GPIO\_Mode\_AF\_PP = 0x18

}**GPIOMode\_TypeDef;**

#define IS\_GPIO\_MODE(MODE) (((MODE) == GPIO\_Mode\_AIN) || ((MODE) == GPIO\_Mode\_IN\_FLOATING) || \

((MODE) == GPIO\_Mode\_IPD) || ((MODE) == GPIO\_Mode\_IPU) || \

((MODE) == GPIO\_Mode\_Out\_OD) || ((MODE) == GPIO\_Mode\_Out\_PP) || \

((MODE) == GPIO\_Mode\_AF\_OD) || ((MODE) == GPIO\_Mode\_AF\_PP))

### /\*\*

### \* @brief General Purpose I/O

### \*/

### typedef struct

### {

### \_\_IO uint32\_t CRL;

### \_\_IO uint32\_t CRH;

### \_\_IO uint32\_t IDR;

### \_\_IO uint32\_t ODR;

### \_\_IO uint32\_t BSRR;

### \_\_IO uint32\_t BRR;

### \_\_IO uint32\_t LCKR;

### } GPIO\_TypeDef; //Typedef burda tanimlanmis

### /\* GPIOA-GPIOC Periph clock enable \*/

### #ifdef STM32F10X\_MD

### /\* GPIOA-GPIOC Periph clock enable \*/

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); // LED, BUTTON GPIO

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE); // LCD GPIO

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE); // LED, BUTTON, BUZZER GPIO

### #elif STM32F2XX

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE); // LED, BUTTON GPIO

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOB, ENABLE); // LCD GPIO

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOC, ENABLE); // LED, BUTTON, BUZZER GPIO

### #endif

### /\*

### \*\*\*\*\*\*\*\*\*LED1->PA0, LED2->PA1, LED3-> PC2, LED4 -> PC3 \*\*\*\*\*\*\*\*\*

### \*/

### #ifdef STM32F2XX

### GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 ;

### GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

### GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT; //Output

### GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; // ve PushPull

### GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

### GPIO\_Init(GPIOA, &GPIO\_InitStructure);

### GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2 | GPIO\_Pin\_3;

### GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

### GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT; //Output

### GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; // ve PushPull

### GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

### GPIO\_Init(GPIOC, &GPIO\_InitStructure);

### /\*\*

### \* @brief Initializes the GPIOx peripheral according to the specified

### \* parameters in the GPIO\_InitStruct.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_InitStruct: pointer to a GPIO\_InitTypeDef structure that

### \* contains the configuration information for the specified GPIO peripheral.

### \* @retval None

### \*/

### void GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct)

### {

### uint32\_t currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;

### uint32\_t tmpreg = 0x00, pinmask = 0x00;

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_MODE(GPIO\_InitStruct->GPIO\_Mode));

### assert\_param(IS\_GPIO\_PIN(GPIO\_InitStruct->GPIO\_Pin));

### }

### #define HSE\_VALUE ((uint32\_t)8000000)

### delay\_init(); //Delay icin konfigurasyonlari yap

### void delay\_init(void)

### {

### RCC\_ClocksTypeDef RCC\_ClocksStatus;

### RCC\_GetClocksFreq(&RCC\_ClocksStatus);

### SysTick\_CLKSourceConfig(SysTick\_CLKSource\_HCLK\_Div8); /\* HCLK/8 \*/

### SysTick\_ITConfig(DISABLE);

### delay\_fac\_us = RCC\_ClocksStatus.HCLK\_Frequency / 8000000; // 8Mhz Kristal için

### delay\_fac\_ms = RCC\_ClocksStatus.HCLK\_Frequency / 8000; // 8Mhz Kristal için

### }

### while(1){

### GPIO\_SetBits(GPIOA , GPIO\_Pin\_0);

### /\*\*

### \* @brief Sets the selected data port bits.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_Pin: specifies the port bits to be written.

### \* This parameter can be any combination of GPIO\_Pin\_x where x can be (0..15).

### \* @retval None

### \*/

### void GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

### {

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

### 

### GPIOx->BSRR = GPIO\_Pin;

### }

**GPIO\_ResetBits(GPIOA , GPIO\_Pin\_1);**

**GPIO\_ResetBits(GPIOC , GPIO\_Pin\_2);**

**GPIO\_ResetBits(GPIOC , GPIO\_Pin\_3);**

**break;**

### /\*\*

### \* @brief Clears the selected data port bits.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_Pin: specifies the port bits to be written.

### \* This parameter can be any combination of GPIO\_Pin\_x where x can be (0..15).

### \* @retval None

### \*/

### void GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

### {

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

### 

### GPIOx->BRR = GPIO\_Pin;

### }

### delay\_ms(250);

### void delay\_ms(uint16\_t nms)

### {

### uint32\_t temp = delay\_fac\_ms \* nms;

### if (temp > 0x00ffffff)

### {

### temp = 0x00ffffff;

### }

### SysTick\_SetReload(temp);

### SysTick\_CounterCmd(SysTick\_Counter\_Clear);

### SysTick\_CounterCmd(SysTick\_Counter\_Enable);

### do

### {

### Status = SysTick\_GetFlagStatus(SysTick\_FLAG\_COUNT);

### }while (Status != SET);

### SysTick\_CounterCmd(SysTick\_Counter\_Disable);

### SysTick\_CounterCmd(SysTick\_Counter\_Clear);

### }

### }

### }

### ////////////////////////////////////////////////////////////////////////////////////////////

### 2.Katman

### ////////////////////////////////////////////////////////////////////////////////////////////

### void EXS03\_initGPIO(void)

{

### GPIO\_InitTypeDef GPIO\_InitStructure;

### /\* GPIOA-GPIOC Periph clock enable \*/

### #ifdef STM32F10X\_MD

### /\* GPIOA-GPIOC Periph clock enable \*/

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); // LED, BUTTON GPIO

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE); // LCD GPIO

### RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE); // LED, BUTTON, BUZZER GPIO

### #elif STM32F2XX

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE); // LED, BUTTON GPIO

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOB, ENABLE); // LCD GPIO

### RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOC, ENABLE); // LED, BUTTON, BUZZER GPIO

### #endif

### /\*

### \*\*\*\*\*\*\*\*\*LED1->PA0, LED2->PA1, LED3-> PC2, LED4 -> PC3 \*\*\*\*\*\*\*\*\*

### \*/

### #ifdef STM32F2XX

### GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 ;

### GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

### GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT; //Output

### GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; // ve PushPull

### GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

### GPIO\_Init(GPIOA, &GPIO\_InitStructure);

### GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2 | GPIO\_Pin\_3;

### GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

### GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT; //Output

### GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; // ve PushPull

### GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

### GPIO\_Init(GPIOC, &GPIO\_InitStructure);

### }

### #define HSE\_VALUE ((uint32\_t)8000000)

### 

### void delay\_init(void)

### {

### RCC\_ClocksTypeDef RCC\_ClocksStatus;

### RCC\_GetClocksFreq(&RCC\_ClocksStatus);

### SysTick\_CLKSourceConfig(SysTick\_CLKSource\_HCLK\_Div8); /\* HCLK/8 \*/

### SysTick\_ITConfig(DISABLE);

### delay\_fac\_us = RCC\_ClocksStatus.HCLK\_Frequency / 8000000; // 8Mhz Kristal için

### delay\_fac\_ms = RCC\_ClocksStatus.HCLK\_Frequency / 8000; // 8Mhz Kristal için

### }

### 

### 

### void delay\_ms(uint16\_t nms)

### {

### uint32\_t temp = delay\_fac\_ms \* nms;

### if (temp > 0x00ffffff)

### {

### temp = 0x00ffffff;

### }

### SysTick\_SetReload(temp);

### SysTick\_CounterCmd(SysTick\_Counter\_Clear);

### SysTick\_CounterCmd(SysTick\_Counter\_Enable);

### do

### {

### Status = SysTick\_GetFlagStatus(SysTick\_FLAG\_COUNT);

### }while (Status != SET);

### SysTick\_CounterCmd(SysTick\_Counter\_Disable);

### SysTick\_CounterCmd(SysTick\_Counter\_Clear);

### }

### ////////////////////////////////////////////////////////////////////////////////////////////

### 3.Katman

### ////////////////////////////////////////////////////////////////////////////////////////////

### /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---------------GPIO---------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### ################################################################

### stm32f10x\_gpio.c dosyasi

### ################################################################

### /\*\*

### \* @brief Initializes the GPIOx peripheral according to the specified

### \* parameters in the GPIO\_InitStruct.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_InitStruct: pointer to a GPIO\_InitTypeDef structure that

### \* contains the configuration information for the specified GPIO peripheral.

### \* @retval None

### \*/

### void GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct)

### {

### uint32\_t currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;

### uint32\_t tmpreg = 0x00, pinmask = 0x00;

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_MODE(GPIO\_InitStruct->GPIO\_Mode));

### assert\_param(IS\_GPIO\_PIN(GPIO\_InitStruct->GPIO\_Pin));

### 

### 

### 

### /\*\*

### \* @brief Sets the selected data port bits.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_Pin: specifies the port bits to be written.

### \* This parameter can be any combination of GPIO\_Pin\_x where x can be (0..15).

### \* @retval None

### \*/

### void GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

### {

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

### 

### GPIOx->BSRR = GPIO\_Pin;

### }

### /\*\*

### \* @brief Clears the selected data port bits.

### \* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.

### \* @param GPIO\_Pin: specifies the port bits to be written.

### \* This parameter can be any combination of GPIO\_Pin\_x where x can be (0..15).

### \* @retval None

### \*/

### void GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

### {

### /\* Check the parameters \*/

### assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));

### assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));

### 

### GPIOx->BRR = GPIO\_Pin;

### }

### ////////////////////////////////////////////////////////////////////////////////////////////

### 4.Katman

### ////////////////////////////////////////////////////////////////////////////////////////////

### ################################################################

### stm32f10x.h temel kütüphane

### ################################################################

### /\*\*

### \* @brief General Purpose I/O

### \*/

### typedef struct

### {

### \_\_IO uint32\_t CRL;

### \_\_IO uint32\_t CRH;

### \_\_IO uint32\_t IDR;

### \_\_IO uint32\_t ODR;

### \_\_IO uint32\_t BSRR;

### \_\_IO uint32\_t BRR;

### \_\_IO uint32\_t LCKR;

### } GPIO\_TypeDef; //Typedef burda tanimlanmis

### /\*\* @addtogroup Peripheral\_memory\_map

### \* @{

### \*/

### #define PERIPH\_BASE ((uint32\_t)0x40000000) /\*!< Peripheral base address in the alias region \*/

### #define APB2PERIPH\_BASE (PERIPH\_BASE + 0x10000)

### #define GPIOA\_BASE (APB2PERIPH\_BASE + 0x0800) //GPİO ADRES

/\*\* @addtogroup Peripheral\_declaration

\* @{

\*/

### #define GPIOA ((GPIO\_TypeDef \*) GPIOA\_BASE) // GPİO TANIMLAMA

### ################################################################

### stm32f10x\_gpio.h kütüphanesi

### ################################################################

### /\*\*

### \* @brief GPIO Init structure definition

### \*/

### typedef struct

### {

### uint16\_t GPIO\_Pin; /\*!< Specifies the GPIO pins to be configured.

### This parameter can be any value of @ref GPIO\_pins\_define \*/

### GPIOSpeed\_TypeDef GPIO\_Speed; /\*!< Specifies the speed for the selected pins.

### This parameter can be a value of @ref GPIOSpeed\_TypeDef \*/

### GPIOMode\_TypeDef GPIO\_Mode; /\*!< Specifies the operating mode for the selected pins.

### This parameter can be a value of @ref GPIOMode\_TypeDef \*/

### }GPIO\_InitTypeDef; //InitTypedef Burada

### /\*\* @defgroup GPIO\_pins\_define

### \* @{

### \*/

### #define GPIO\_Pin\_0 ((uint16\_t)0x0001) /\*!< Pin 0 selected \*/

### #define GPIO\_Pin\_1 ((uint16\_t)0x0002) /\*!< Pin 1 selected \*/

### #define GPIO\_Pin\_2 ((uint16\_t)0x0004) /\*!< Pin 2 selected \*/

### #define GPIO\_Pin\_3 ((uint16\_t)0x0008) /\*!< Pin 3 selected \*/

### #define GPIO\_Pin\_4 ((uint16\_t)0x0010) /\*!< Pin 4 selected \*/

### #define GPIO\_Pin\_5 ((uint16\_t)0x0020) /\*!< Pin 5 selected \*/

### #define GPIO\_Pin\_6 ((uint16\_t)0x0040) /\*!< Pin 6 selected \*/

### #define GPIO\_Pin\_7 ((uint16\_t)0x0080) /\*!< Pin 7 selected \*/

### #define GPIO\_Pin\_8 ((uint16\_t)0x0100) /\*!< Pin 8 selected \*/

### #define GPIO\_Pin\_9 ((uint16\_t)0x0200) /\*!< Pin 9 selected \*/

### #define GPIO\_Pin\_10 ((uint16\_t)0x0400) /\*!< Pin 10 selected \*/

### #define GPIO\_Pin\_11 ((uint16\_t)0x0800) /\*!< Pin 11 selected \*/

### #define GPIO\_Pin\_12 ((uint16\_t)0x1000) /\*!< Pin 12 selected \*/

### #define GPIO\_Pin\_13 ((uint16\_t)0x2000) /\*!< Pin 13 selected \*/

### #define GPIO\_Pin\_14 ((uint16\_t)0x4000) /\*!< Pin 14 selected \*/

### #define GPIO\_Pin\_15 ((uint16\_t)0x8000) /\*!< Pin 15 selected \*/

### #define GPIO\_Pin\_All ((uint16\_t)0xFFFF) /\*!< All pins selected \*/

### void GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct);

### void GPIO\_StructInit(GPIO\_InitTypeDef\* GPIO\_InitStruct);

### void GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin);

### void GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin);

### //