/\*

\* STM32F4xx\_gpio\_driver.c

\*

\* Created on: Mar 5, 2021

\* Author: 99003772

\*/

**#include** "STM32F4xx\_gpio\_driver.h"

//I. peripheral clock enable or disable

**void** **GPIO\_PeripheralClK**(GPIO\_RegDef \*pGPIOx,uint8\_t EnorDI)//EnorDI is enable or disable

{

**if**(EnorDI == ENABLE)

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PCLOCKENABLE;

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PCLOCKENABLE;

}

**else**

{

**return** ;

}

}

**else**

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_PCLOCKDISABLE;

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_PCLOCKDISABLE;

}

**else**

{

**return** ;

}

}

}

//II. Init and DeInit

**void** **GPIO\_Init**(GPIO\_Handle \*pGPIOHandle)

{

//1.Configuring Pin Mode

uint32\_t temp;

temp = (pGPIOHandle->pin\_config->GPIO\_pin\_mode) << (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->MODER = ~(0x3)<< (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->MODER = temp;

//2. config pin Otyper

uint16\_t temp1;

temp1 = (pGPIOHandle->pin\_config->GPIO\_otype) << (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->OTYPER = ~(0x1)<< (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->OTYPER = temp1;

//3.Config OSpeed

temp = (pGPIOHandle->pin\_config->GPIO\_speed) << (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->SPEEDR = ~(0x1)<< (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->SPEEDR = temp;

//4.Config PUPD

temp = (pGPIOHandle->pin\_config->GPIO\_pupd) << (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->PUPDR = ~(0x1)<< (2\*pGPIOHandle->pin\_config->GPIO\_pin\_number);

pGPIOHandle->pGPIOx->PUPDR = temp;

//4.Config Alt fn

uint32\_t temp2;

temp = pGPIOHandle->pin\_config->GPIO\_pin\_number/8;

temp2= pGPIOHandle->pin\_config->GPIO\_pin\_number%8;

pGPIOHandle->pin\_config->GPIO\_altfn[temp]=~(0xf) << (4\*temp2);

pGPIOHandle->pin\_config->GPIO\_altfn[temp]=(uint8\_t)(pGPIOHandle->pin\_config->GPIO\_altfn) << (4\*temp2);

}

**void** **GPIO\_DeInit**(GPIO\_Handle \*pGPIOx);

//III. Read/Write to/from port/pin

uint16\_t **GPIO\_Read\_from\_port**(GPIO\_RegDef \*pGPIOx)

{

uint16\_t temp=0;

temp= (uint16\_t)pGPIOx->IDR;

**return** temp;//temp value goes to microprocessor

}

uint8\_t **GPIO\_Read\_from\_pin**(GPIO\_RegDef \*pGPIOx,uint8\_t GPIO\_pin\_number)

{

uint8\_t temp=0;

temp=(uint8\_t)(pGPIOx->IDR >> GPIO\_pin\_number &(0x00000001));

**return** temp;

}

**void** **GPIO\_Write\_to\_port**(GPIO\_RegDef \*pGPIOx,uint16\_t value)

{

pGPIOx->ODR=value;

}

**void** **GPIO\_Write\_to\_pin**(GPIO\_RegDef \*pGPIOx,uint8\_t GPIO\_pin\_number,uint16\_t value)

{

**if**(value == ENABLE)

{

pGPIOx->ODR|=(1<<GPIO\_pin\_number);

}

**else**

{

pGPIOx->ODR&=~(1<<GPIO\_pin\_number);

}

}

//IV. Toggle

**void** **GPIO\_toggle**(GPIO\_RegDef \*pGPIOx,uint8\_t GPIO\_pin\_number)

{

pGPIOx->ODR^=(1<<GPIO\_pin\_number);

}