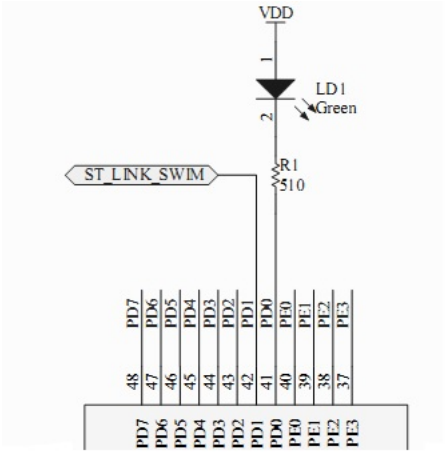
STM8 Tutorials – #2 Hello World Program

— June 10, 2015

Link to Part #1 in this Series

Without much talks, lets create our first "Hello World" application for STM8S Discovery Board. As practiced in embedded world, this time also, LED blinking is our way to say *Hello* to STM8S world.

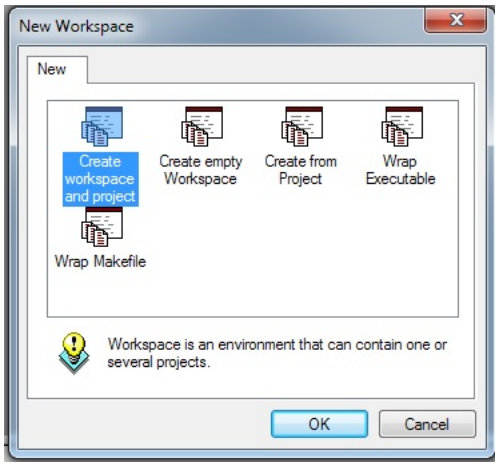
STM8S Discovery has an on-board Green Colored user LED connected to PD0. We will blink this led on and off with some delay in between.



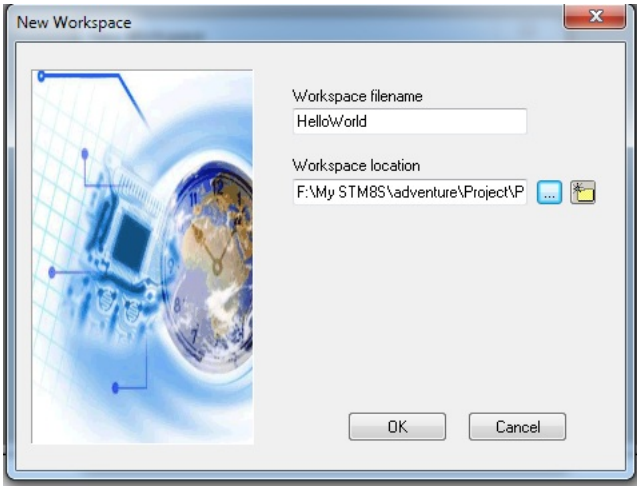
STM8S Discovery LED connected to PD0

STEP 1: STVD Workspace/Project Creation

Create a new STVD Workspace from File menu

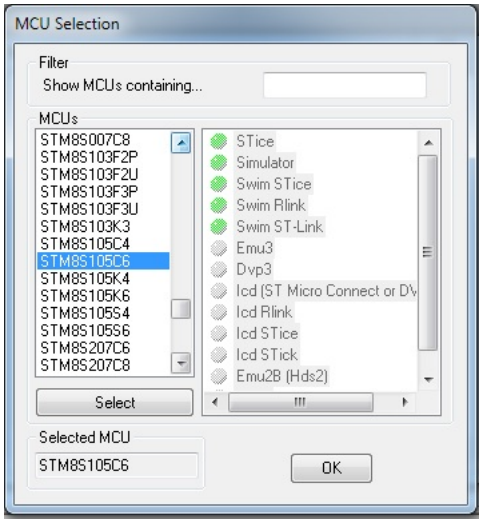


Step -1 Create Workspace

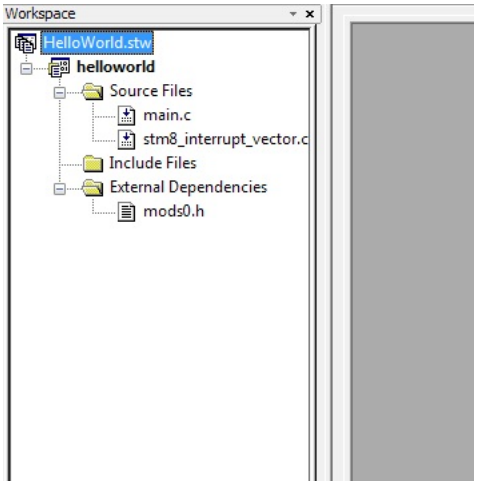


Step -2 Project Name

Step -3 Project Toolchain selection as Cosmic STM8



Step -4 Choose STM8S105C6 as Target MCU

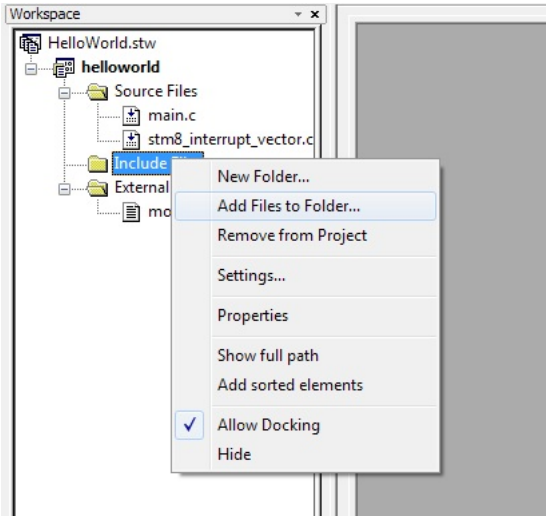


Step – 5 Review Project File Tree

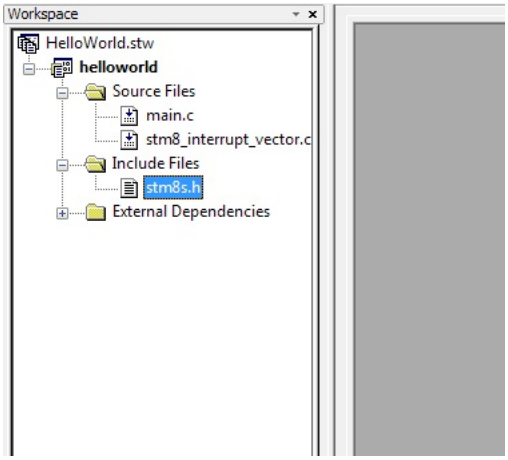
STEP – 2 Adding STM8S Register Definition header stm8s.h

To compile any C project for STM8S can not be done without mapping register definition into suitable names. This file is very important and it will be there in all most all the upcoming projects. It maps register addresses with suitable names. For example GPIOA base register is at 0×5000, this file has definition to map 0×5000 to a name as “GPIOA”.

DDownload stm8s.h file here.



Add stm8s.h file into your project



Verify that stm8s.h has been added in your project

STEP – 3 Writing the Code

```
#include "stm8s.h"


void myDelay(void);
```

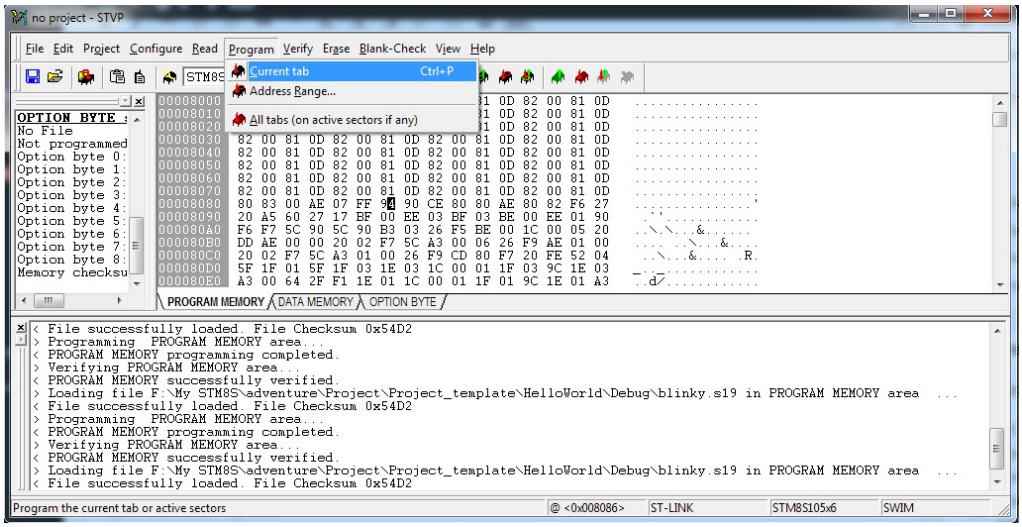
```
void myDelay()
{
   int i,j;
   for(i=0;i<1000;i++)
   {
       for(j=0;j<100;j++);
       }
}

main()
{
     GPIOD->DDR |= 0x01;  // PD.0 as Output
     GPIOD->CR1 |= 0x01;  // PD.0 as Push Pull Type Output
     while (1)
     {
            GPIOD->ODR |=1<<0;  // PD.o = 1
            myDelay();
            GPIOD->ODR &= ~(1<<0); // PD.0 = 0
            myDelay();
     }
}
```

STEP – 4 Compile and Build the Project

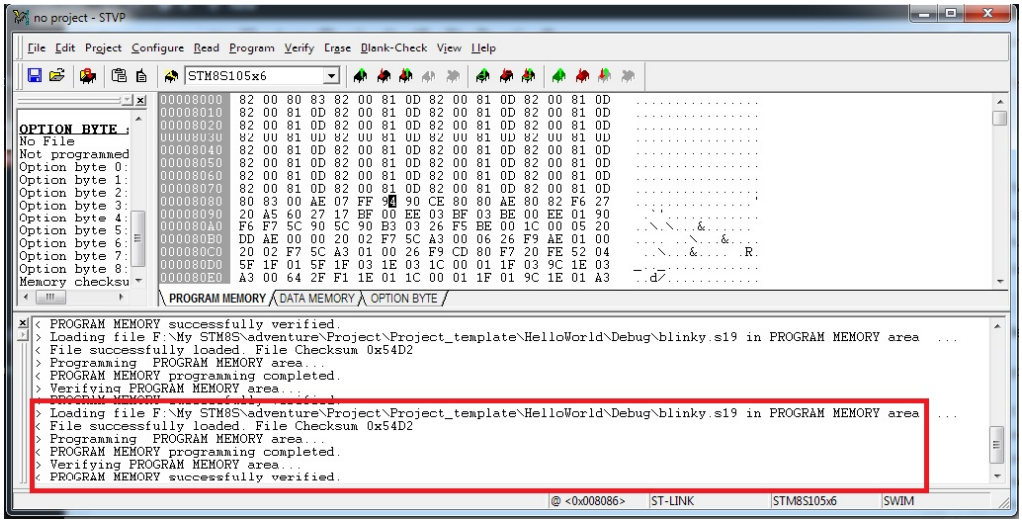
Compile and build the necessary machine files by pressing F7. If build process does not produce any errors, our code is successfully cross compiled and machine code is now ready to download on mcu program memory.

STEP – 5 Download the code onto MCU Program Memory

Now, open STVP and open .s19 file from the Debug folder inside project directory.



After opening the .s19 file, choose Program current tab from STVP Menu options.



STVD after successful device programming

> *LED LD1 on Discovery board should be blinking now with a delay defined by mydelay() function. If you are facing any troubles in achieving desired out put in this case, you can put your issue in the comment section below.*

« Previous Article

STM8S Tutorials - #1 Tools

Next Article »

Low cost STM8S103F3P6 Board - Review and Getting Started Guide

ABOUT AUTHOR

Devesh Samaiya

Chief Tinkerer at Lonely Night Projects

f

8 COMMENTS

vaibhav
June 23, 2015 at 5:28 am Reply

no 3rd Tutorial

Robothito
April 14, 2016 at 4:29 am Reply

Nice tutorial
but… I have ST Visual Develop and Cosmic but “stm8s.h” does not exist anywhere…

Where is this file?
I have completely different definitions on each chip particular h file (like “STM8S903K.h”)

Can you help me with any comment?

IndianYouthful
August 14, 2016 at 5:23 pm Reply

The stm8s.h that I am using is available for download in this post itself….anyways here is the link h ttps://www.dropbox.com/s/e3ac4fa5w48iwnj/stm8s.h?dl=0

Amita
October 5, 2016 at 10:04 am Reply

Nice. Waiting for your other projects like these.

Rohit
December 6, 2016 at 6:54 am Reply

GPIOD->DDR |= 0×01; // PD.0 as Output
GPIOD->CR1 |= 0×01; // PD.0 as Push Pull Type Output

What is the meaning of GPIOD, DDR, CR1, PD.0??
Any guide for this ? sorry for such silly question but i am a newbie to STM8

Devesh Samaiya
January 17, 2017 at 5:31 am Reply

GPIOD is a structure found in STM8S.h, DDR is the data direction register which specifies if the po rt is going to be used as output or input. CR is the control register which sets various aspects of th

e port and PD.0 indicates that we are dealing with 0th bit of PORTD. Refer to STM8S reference m anual for more details on various registers.

Hyuckjin
December 15, 2016 at 1:19 am Reply

Compilation is fine. But the LED does not work.
Stm8s.h is the library from st.

Omkar Teli
May 8, 2017 at 9:04 am Reply

I am writing simple SPI c code for STM8S103F3 but i am not getting anything on SPI pins, seems to be some pr oblem in code or board has problem. Kindly help me to solve the problem. I searched a lot & lot but no good tut orials available on STM8. For your reference i am adding my c code here…

……SPI code
……

```
#include “stm8s.h”

void SPI_SendData1(uint8_t Data)
{
SPI->DR = Data; /* Write in the DR register the data to be sent*/
}

FlagStatus SPI_GetFlagStatus1(SPI_Flag_TypeDef SPI_FLAG)
{
FlagStatus status = RESET;
/* Check the status of the specified SPI flag */
if ((SPI->SR & (uint8_t)SPI_FLAG) != (uint8_t)RESET)
{
status = SET; /* SPI_FLAG is set */
}
else
{
status = RESET; /* SPI_FLAG is reset*/
}

/* Return the SPI_FLAG status */
return status;
}

void SPI_Init1(SPI_FirstBit_TypeDef FirstBit, SPI_BaudRatePrescaler_TypeDef BaudRatePrescaler, SPI_Mode
_TypeDef Mode, SPI_ClockPolarity_TypeDef ClockPolarity, SPI_ClockPhase_TypeDef ClockPhase, SPI_DataD
irection_TypeDef Data_Direction, SPI_NSS_TypeDef Slave_Management, uint8_t CRCPolynomial)
{

/* Frame Format, BaudRate, Clock Polarity and Phase configuration */
SPI->CR1 = (uint8_t)((uint8_t)((uint8_t)FirstBit | BaudRatePrescaler) |
(uint8_t)((uint8_t)ClockPolarity | ClockPhase));

/* Data direction configuration: BDM, BDOE and RXONLY bits */
SPI->CR2 = (uint8_t)((uint8_t)(Data_Direction) | (uint8_t)(Slave_Management));

if (Mode == SPI_MODE_MASTER)
{
SPI->CR2 |= (uint8_t)SPI_CR2_SSI;
}
else
{
SPI->CR2 &= (uint8_t)~(SPI_CR2_SSI);
}

/* Master/Slave mode configuration */
SPI->CR1 |= (uint8_t)(Mode);

/* CRC configuration */
SPI->CRCPR = (uint8_t)CRCPolynomial;
}

void SPI_Cmd1(FunctionalState NewState)
{
if (NewState != DISABLE)
{
SPI->CR1 |= SPI_CR1_SPE; /* Enable the SPI peripheral*/
}
else
{
SPI->CR1 &= (uint8_t)(~SPI_CR1_SPE); /* Disable the SPI peripheral*/
}
}

void SPI_DeInit1()
{
```

```
SPI->CR1 = SPI_CR1_RESET_VALUE;
SPI->CR2 = SPI_CR2_RESET_VALUE;
SPI->ICR = SPI_ICR_RESET_VALUE;
SPI->SR = SPI_SR_RESET_VALUE;
SPI->CRCPR = SPI_CRCPR_RESET_VALUE;
}

void CLK_PeripheralClockConfig1(CLK_Peripheral_TypeDef CLK_Peripheral, FunctionalState NewState)
{
if (((uint8_t)CLK_Peripheral & (uint8_t)0×10) == 0×00)
{
if (NewState != DISABLE)
{
/* Enable the peripheral Clock */
CLK->PCKENR1 |= (uint8_t)((uint8_t)1 <PCKENR1 &= (uint8_t)(~(uint8_t)(((uint8_t)1 <PCKENR2 |= (uint8_t)
((uint8_t)1 <PCKENR2 &= (uint8_t)(~(uint8_t)(((uint8_t)1 <CKDIVR &= (uint8_t)(~CLK_CKDIVR_HSIDIV);

/* Set High speed internal clock prescaler */
CLK->CKDIVR |= (uint8_t)HSIPrescaler;
}

void main(void)
{
CLK_HSIPrescalerConfig1(CLK_PRESCALER_HSIDIV1); //16 MHz Internal sys clock
CLK_PeripheralClockConfig1(CLK_PERIPHERAL_SPI, ENABLE); //Enables the SPI peripheral clock
SPI_DeInit1(); //All SPI registers are reset
SPI_Cmd1(DISABLE); //SPI Disable
SPI_Init1(SPI_FIRSTBIT_MSB, SPI_BAUDRATEPRESCALER_4, SPI_MODE_MASTER,
SPI_CLOCKPOLARITY_LOW, SPI_CLOCKPHASE_1EDGE, SPI_DATADIRECTION_2LINES_FULLDUPLEX,
SPI_NSS_HARD, 0×00); //SPI CR1 and SPI CR2 are set
SPI_Cmd1(ENABLE); //SPI Enable

while (1)
{
SPI_Cmd1(ENABLE); //SPI Enable
while(SPI_GetFlagStatus1(SPI_FLAG_BSY)); //Wait till the SPI is busy
while(!SPI_GetFlagStatus1(SPI_FLAG_TXE)); //Wait till transmit buffer is empty
SPI_SendData1(0×55); //Sending data on SPI
SPI_Cmd1(DISABLE); //SPI Disable
}
}
```

LEAVE A REPLY

*Your email address will not be published. Required fields are marked **

Name (required):

Email (required):

Website

Math Captcha
58 + = 63

Your comment (required):

Post Comment