Home
STM32F4 links
SPL libs
HAL libs
Tutorials
ESP8266 & ESP32
About

About
LinkedIn
Twitter
Google+
Facebook
Github
Instagram

,
STM32F42
DISCOVER
/ STM32F7
/ STM32F7
DISCOVER

purpose microcontrollers
STM32G series - Next generation general purpose microcontrollers
STM32L series - Low-Power series
STM32MP series - Microprocessors
STM32W series - Wireless microcontrollers
STM32U series - Ultra-Low-Power series

PREVIOUS STORY
1-Wire over UART

OMMENTS

TOP POSTS

TouchGFX simulator development in Visual Studio Code with CMake

T

RECENT COMMENTS

- How to use UART DMA RX on STM32F103 MCU on STM32 tutorial: Efficiently receive UART data using DMA
- STM32Cube Ecosystem | Andreas' Blog on How to properly enable/disable interrupts in ARM Cortex-M?
- STM32 delay ms function : Software delay vs HAL Delay function on Library 03- STM32F4 system clock and delay functions
- Project: EOGee – Programming the EOGlass microcontrollers | Matt's Projects on Tutorial – Jump to system memory from software on STM32
- c – Control AMIS-30543 with STM32F030R8 via SPI – ThrowExceptions on Library 02- STM32F429 Discovery GPIO tutorial with onboard leds and button

CATEGORIES

- ARM Cortex-M
- ESP8266 & ESP32
- Random
- STM32F4
- STM32F4 Discovery

or
ial
:
C
o
nt
ro
l
W
S
2
8
1
2

SUPPORT

Donate

SEARCH

Search ... Search

TWITTER

Posts from @tilz0R

Nothing to see here - yet

When they post, their posts will show up here.

View on X

RECENT POSTS

CALENDAR

June 2018

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

Follow me on Google+

le ds wi th S T M 3 2

BY TILZOR · PUBLISHED JUNE 3, 2018 · UPDATED OCTOBER 5, 2018

In

M	T	W	T	F	S	S
25	26	27	28	29	30	
« Apr				Sep »		

ARCHIVES

- December 2022
- August 2022
- October 2018
- September 2018
- June 2018
- April 2018
- December 2017
- November 2017
- July 2017
- April 2017
- March 2017
- November 2016
- July 2016
- April 2016
- January 2016
- December 2015

explain how to drive WS2812B with STM32 using TIM PWM and DMA peripherals in the most efficient way by using minimum amount of RAM, required to process all leds.

There

already available on the web. If you have no experience with WS2812B leds, I strongly recommend you to read the blog post. It is very well written, but it has one major issue.

of RAM for each LED, which is **96** bytes per LED. If you have 100 leds, that's almost **10k** of RAM.

> *STM32 hardware allows you good way to scale down this*

3 bytes *per LED, which is* ***300 bytes at 100 leds*** *=* ***32x*** *more efficient! In addition to this memory, we also need temporary buffer to store 2 LED raw PWM*

*DMA hardware. All together we need **3 * LEDS_Count + 24 * sizeof(uint32) bytes.***

Double buffering DM

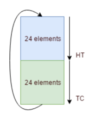
DMA hardware in all STM32 families allows you double buffering mode. In this mode, you have 2 memory addresses and DMA will switch between memories at the end of block transaction. Handling some additional code in our project. Second option we have is to use single buffer mode and rely on **DMA Transfer-Complete** and **Half-Transfer Complete** events. They are called at the transfer and in the middle of block transfer, respectively.

Imagine we have array of **48** elements. If we configure DMA to transfer **48** elements we will:

- receive Half-Transfer Complete event (**HT**

24 elements were transferred,
- receive Transfer Complete event (**TC event**) after all elements were transferred.

Since we are receiving 2 events in the middle/end of block, we can use single but interact with it as double buffer data. DMA in STM32 also supports **circular** mode, which basically means that once we are at the end of block transfer, DMA will start from the and will transfer more data (this is visible on picture below with left arrow).



STM32 DMA HT and TC events

We will use this feature for our raw

PWM for each LED.

LED memory footprint

Each LED has RGB format for color,

24 bits of data (or **3 bytes**). To have information for all leds, we need **3 * leds_count** big array with **R0,G0,B0,R1,G1,B1,R2,G2,B2,...** color structure.

WS2812

led sequence

I will not go deep into control sequence, to set color you basically need to respect rules below:

- PWM signal must

or **1.25us** per **bit**. To transfer data for 1 LED, you need **30us**

- Before you start PWM sequence for all leds, 50us reset pulse (pulse low) is required (**40 periods at 800kHz**)
- Transfer

each led of **33% (logical 0) or 67% (logical 1)** duty-cycle on PWM
  - There is no dead-time between end of first led and beginning of second led!
- After PWM

pulse again (**40 periods at 800kHz**)

STM32 DMA, TIM + PWM im

e
m
en
ta
tio
n

Implementation is done using single timer + PWM output on one of its channels. DMA is used to transfer data from compare register for PWM output. Single 48 words long array is used for data transfer, acting as double-buffer DMA. On the beginning, memory is configured with all zeros for reset transfered with DMA to PWM channel and we have to wait for **TC event**. Once we have the TC event, we can start preparing a data for first LED. To send **24bits** and according to bit values, we set duty-cycle to either **33% or 67%.** When we are configuring the first LED, we also need to prepare the same for second LED and of main buffer.

Next step is to start the DMA transfer in circular mode and wait for HT event. When we receive HT event (half transfer of 48 elements transfered), we know was completely transferred to PWM and we no longer need this memory for it, thus we can start prepare memory for third LED. Later, we will receive TC event, which means that successfully transferred to PWM and we no longer need second part of memory for second LED. At this point, data for third led started to transfer from memory to PWM and second part of our 2-LED array to configure forth LED data. When we transfer all leds, we can simply stop the DMA transfer and configure reset pulse again to send zeros for **50us low pulse**.

- Send reset pulse, wait for TC event, DMA is in **normal** mode
- Prepare first LED data in first 24 section, prepare second LED data in

start DMA transfer in **circular** mode and wait for HT event

- **HT** event received, first LED data were transferred, prepare third LED data to first 24 section, wait for **TC** event
- **TC** event

data were transferred, prepare forth LED data to second 24 section, wait for **HC** event

- **HC** event received, third LED data were transferred, prepared fifth LED data to first 24 section,

TC event

- **TC** event received, forth LED data were transferred, prepare sixth LED data to second 24 section, wait for **HC** event

Pattern is that ODD LEDs are in top

memory (on image above is **blue** part), while EVEN LEDs are on bottom part of big memory (**green** part). This is very well visible, but now we have to finish transfers example for **2** or **3** leds.

- 2 leds
  - Prepare first LED data in first 24 section, prepare second LED data in second 24 section, start DMA transfer in circular mode and wait f

- **HT** event received, we already have second led data in second part of memory and DMA is already transferring it now to PWM registers => **DO NOTHING** now, just wait for

- **TC** event received, **manually STOP DMA**
- 3 leds
    - Prepare first LED data in first 24 section, prepare second LED data in second 24 section, start DMA transfer in circular mode and

HT event

- **HT** event received, we have second led data in second part of memory, but we still need to prepare third LED data in first 24 section. Prepare the data

TC event

- **TC** event received, we have fully transfer 2 leds, but we still have to send third led data which are already prepared and have just started with transfer =>**DO**

for **HC** event
  - HC event received, we transferred 3x LED data => **manually STOP DMA**

These 2 examples clearly show that we have to manually take care of when to either on **TC** or **HT** events.

Example

Example code is highly-optimized for specific MCU and reference manual should be used when porting. **NUCLEO-**

used as experiment with settings below:

- PWM PIN: **PB3**, connected to **TIM2_CH2**. DMA is served via **DMA1 Stream6, channel 3**
- MCU uses internal HSI clock, increased via PLL to **84** MHz.
- TIM2

with no prescaler with period of **105** ticks (**104** written in **TIM2->ARR** register)
- Example uses 8**leds**.
- Example is coded in Keil uvision and is below **32kB** which means you may

free of charge.

Download code from Github account.

Share this to other users:

Tags:

control

led

leds

stm32

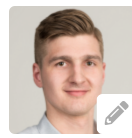
stm32f4

tim

tim2

timer

ws2812b



tilz0R

Owner of this site. Application engineer, currently employed by STMicroelectronics. Exploring latest technologies and owner of different libraries posted on Github.

YOU

LIKE.
..

Library 07- DAC Converter on STM32F4

HAL Library 24- RTC for STM32Fxxx

HAL Library 35- GPS parser for STM32F

AUGUST 23, 2015

APRIL

JULY
16,
2016

5
Comn

ogin ▼

J
d

LOG IN
WITH

OR
SIGN
UP
WITH
DISQUS

Name

Best

M

About
LinkedIn
Twitter
Google+
Facebook
Github
Instagram