Казанский национальный исследовательский технический университет им. А.Н. Туполева

MSP430 Family

October 23, 2016, Bulat Valeev

Lection 1. Introduction. Technical pages overview.

Motivation

There were 12 billion embedded systems produced worldwide in the 2000 year.

Devices starting from simple GPS receivers to TV, tablets, phones, ECU in the cars, communication equipment are covered with definition embedded systems

Motivation

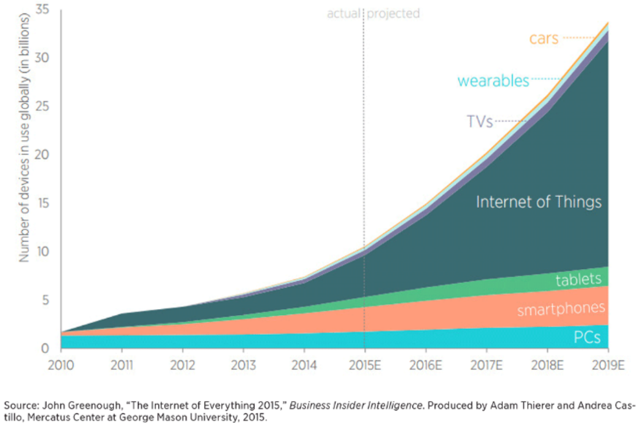


Figure: *Embedded devices statistics*[1]

Plan of the overall course

- Low performance MCU (MSP430G2553)
- Middle performance MCU (STM32F303VC)
- High performance MCU (I.MX6D /DM368)
- DSP (TMS320)
- FPGA
- Heterogeneous processors

Plan of the low performance MCU course

Lab 1· Introduction to the programming, technical pages

Lab 2· *C* coding in the MCU. Interrupts.

Lab 3· Interrupt based program

Lab 4· API writing

Lab 5· Debug

Lab 6· Calculations

Lab 7· Low power modes

Lab 8· Communication

Challenges

What you should know at the end of the course.

- Learn *C* coding for MCU's
- Know how use the periphery in the MSP430
- Study the MCU family implementation
- Train in debug for the MCU's

Introduction

Embedded systems development is one of the most challenging in terms of efficiency.

Main properties of the embedded systems: efficient, small, cheap.

There are a lot of low power MCU architectures. You will never know all of them. But you should know how migrate from one architecture to another.

Examples:

- ARM (Freecale, NXP,Infenion, STM, Atmel, TI)
- AVR (Atmel)
- MSP430 (TI)
- MCS 51 (Intel, Maxim, Atmel, NXP, TI)
- PIC (Microchip)
- STM8 (STM)

MCU families structure

MCU have base distribution model for development:

- Architecture (*ARM*)
- Core (Cortex-M4)
- Family (*STM32F*)
- Series (*STM32F*3)
- Device (*STM32F*303*VC*)
- Package (Property for the hardware development)

MCU families structure

Figure: *STM32F Family*[2]

MCU abstraction levels

We will use the 5 main abstraction levels of the MCU's [3] :

- Device (*Capacitor*, *FET*, *Bipolar transistor* etc.)
- Circuit (*Operational Amplifiers*, *Triggers* etc.)
- Gates (*OR*, *AND*, *NOT*, *XOR* etc.)
- Module (*ALU*, *RAM*, *ROM*, *Periphery blocks* etc.)
- System (*SoC*, *FPGA*, *DSP*, *CPU* etc.)

MSP430 family

In this course we will work with MSP430 family with MSP430G2553 mixed signal controller. The controller is produced by the Texas Instruments company and is developed for the computationally simple, energy efficient applications.
The main features of the controller are [4]:

- 16 bit RISC CPU
- Ultra low power (up to 0.1 μA)
- Fast wake up (up to 1 μs)
- Digitally controlled reference clock (up to 16 MHz)
- Universal serial communication interface(USCI)

MSP430 launchpad

The practical work starts with the evaluation board for the MSP430G2553 MCU called *MSP430 Launchpad*. The developing board allows to work with controller without hardware developing part and learn all possibilities faster. [5]

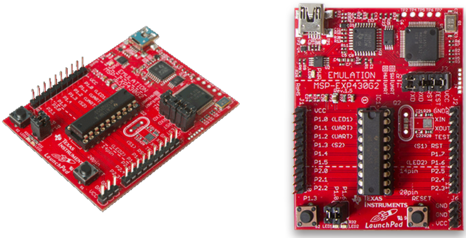


Figure: *MSP430 Launchpad*

Technical pages overview

Each controller starts with technical pages.

The main technical documentation in the MCU programming.

- MCU Family User Guide (or Reference manual) [1]
- Datasheet for MCU [2]
- Device Erratasheet [3]
- Application notes [4]
- Code examples [4]

Technical pages overview


MSP430x2xx Family

User's Guide

Texas Instruments

Figure: *Your best friend for next lessons*

Algorithm construction over datasheet

Main description of the periphery and configuration process explained in the Family User Guide (or Reference Manual).

The developer must figure out the configuration process.

Unpredicted behaviour must be avoided by developer.

Algorithm construction over datasheet

Each technical paper for the family contains the registry values and addresses for programming.
Usually it looks like certain byte, where each bit perform certain action. There are three types of registers:

- Read-only
- Read and write
- Write-only

Read-only registers are used for GPIO input ports and state flags
Read and write registers are used for MCU configuration
Write only registers are peripheral output buffers, lock registers

Example of the registry

Here presented table from the MSP430 family datasheet. In the table are presented registers for the Timer A.

Table: Timer A registers

Register	Short Form	Register Type	Address	Initial state
Timer A control	TACTL	Read/Write	0160h	Reset with POR
Timer A counter	TAR	Read/Write	0170h	Reset with POR
Timer A capture/compare control 0	TACCTL0	Read/Write	0162h	Reset with POR
Timer A capture/compare 0	TACCR0	Read/Write	0172h	Reset with POR
Timer A capture/compare control 1	TACCTL1	Read/Write	0164h	Reset with POR
Timer A capture/compare 1	TACCR1	Read/Write	0174h	Reset with POR
Timer A capture/compare control 2	TACCTL2	Read/Write	0166h	Reset with POR
Timer A capture/compare 2	TACCR2	Read/Write	0176h	Reset with POR
Timer A interrupt vector	TAIV	Read only	012Eh	Reset with POR

Algorithm example: Timer

At first, you should write an algorithm of an action.
You must understand how MCU must perform any activity in a most safe way.
Any *C* code must base on the hardware steps which will made inside.
Example: We adjust timer overflow interval:

- Turn off interrupts
- Stop adjusted timer
- Set the timer overflow value
- Read the timer overflow value
- Reset timer counter
- Start adjusted timer
- Turn on interrupts

Technical pages for the timer A

12.2.1.1 Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally via TACLK or INCLK. The clock source is selected with the TASSELx bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the IDx bits. The timer clock divider is reset when TACLR is set.

12.2.2 Starting the Timer

The timer may be started, or restarted in the following ways:

- The timer counts when MCx > 0 and the clock source is active.
- When the timer mode is either up or up/down, the timer may be stopped by writing 0 to TACCR0. The timer may then be restarted by writing a nonzero value to TACCR0. In this scenario, the timer starts incrementing in the up direction from zero.

12.2.3 Timer Mode Control

The timer has four modes of operation as described in Table 12-1: stop, up, continuous, and up/down. The operating mode is selected with the MCx bits.

Table 12-1. Timer Modes

MCx	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TACCR0.
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TACCR0 and back down to zero.

12.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TACCR0, which defines the period, as shown in Figure 12-2. The number of timer counts in the period is TACCR0+1. When the timer value equals TACCR0 the timer restarts counting from zero. If up mode is selected when the timer value is greater than TACCR0, the timer immediately restarts counting from zero.

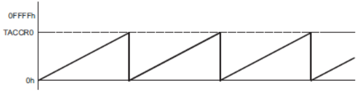


Figure 12-2. Up Mode

Technical pages for the timer A

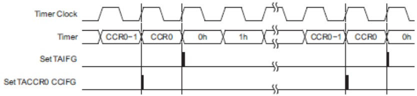


Figure 12-3. Up Mode Flag Setting

12.2.3.2 Changing the Period Register TACCR0

When changing TACCR0 while the timer is running, if the new period is greater than or equal to the old period, or greater than the current count value, the timer counts up to the new period. If the new period is less than the current count value, the timer rolls to zero. However, one additional count may occur before the counter rolls to zero.

12.2.3.3 Continuous Mode

In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in Figure 12-4. The capture/compare register TACCR0 works the same way as the other capture/compare registers.

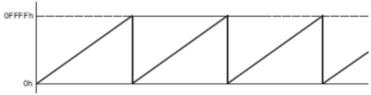
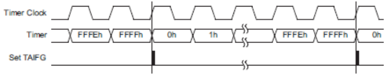


Figure 12-4. Continuous Mode

The TAIFG interrupt flag is set when the timer *counts* from 0FFFFh to zero. Figure 12-5 shows the flag set cycle.



Technical pages for the timer A

12.3.1 TACTL, Timer_A Control Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Unused	Bits 15-10	Unused
TASSELx	Bits 9-8	Timer_A clock source select 00 TACLK 01 ACLK 10 SMCLK 11 INCLK (INCLK is device-specific and is often assigned to the inverted TBCLK) (see the device-specific data sheet)
IDx	Bits 7-6	Input divider. These bits select the divider for the input clock. 00 /1 01 /2 10 /4 11 /8
MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00 Stop mode: the timer is halted. 01 Up mode: the timer counts up to TACCR0. 10 Continuous mode: the timer counts up to 0FFFFh. 11 Up/down mode: the timer counts up to TACCR0 then down to 0000h.
Unused	Bit 3	Unused
TACLR	Bit 2	Timer_A clear. Setting this bit resets TAR, the clock divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.
TAIE	Bit 1	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0 Interrupt disabled 1 Interrupt enabled
TAIFG	Bit 0	Timer_A interrupt flag 0 No interrupt pending 1 Interrupt pending

Result

Here presented the description in registers and C code for the timer interval change function
Input value: var.

- Initialize variables $var1, var2$
- Set GIE to 0
- Save $MC1$ and $MC0$ to $var1$
- Set $MC0$ and $MC1$ to 0
- Set the $TACCR0$ to var
- Read the $TACCR0$ to $var2$
- Set TAR to 0
- Set $MC0$ and $MC1$ to $var1$
- Set GIE to 1

Code example:Timer

```
void TimASetSpeed(int var)
{
int var1,var2; //Initialize variables var1,var2
__bic_SR_register(GID); // Set GIE to 0
var1=TACTL &(MC1|MC0);//Save MC1, MC0 to var1
TACTL=TACTL &~(MC1|MC0);//Set MC0, MC1 to 0
TACCR0=var;//Set the TACCR0 from var
var2=TACCR0;//Read the TACCR0 to var2
TAR=0;//Set TAR to 0
TACTL=TACTL|var1; //Set MC0, MC1 from var1
__bis_SR_register(GIE);//Set GIE to 1
return;
}
```

Current work :UART

Write the algorithm which will initialize the UART in MP430 controller with 9600 baudrate.

Hint: Use clock $BRCLK = 1MHz$

Use the Family User Guide for MSP430x2xx MCU's with part Universal serial interface in the UART mode.

Result

The resulting algorithm to configure the UART is presented here

- Turn off the UART
- Set the baud-rate
- Set the low frequency baud-rate settings
- Configure GPIO pins
- Turn on UART
- Turn on RX interrupt

Result

The resulting algorithm to configure the UART is presented here

- Set the UCSWRST bit
- Configure the UCA0BR0 and UCA0BR1 bytes
- Configure the UCA0MCTL byte
- Configure P1SEL and P2SEL bytes
- Reset the UCSWRST bit
- Set the UCA0RXIE bit

Home task: Clock generation

Write the algorithm which will initialize different clock frequency.

Hint: Use Digitally controlled oscillator

Use the Family User Guide for MSP430x2xx MCU's with part Basic clock module+. Deadline is xx.xx.xxxx.

Thanks for your attention

Reference slide

- Shawn Hymel. https://www.linkedin.com/pulse/why-javascript-good-embedded-systems-shawn-hymel
- Roman Popov. http://www.compel.ru/lib/ne/2011/2/4-mikrokontrolleryi-stm32-s-nulya/
- Anish Goel. http://www.slideshare.net/anishgoel/eda-1744474
- Texas Instruments. $http://www.ti.com/lsds/ti/microcontrollers\_16-bit\_32-bit/msp/overview.page$
- Texas Instruments. $http://www.ti.com/tool/msp-exp430g2\#descriptionArea$

Reference slide

- Texas Instruments. http://www.ti.com/lit/pdf/slau144
- Texas Instruments. http://www.ti.com/lit/gpn/msp430g2553
- Texas Instruments. http://www.ti.com/lit/pdf/slaz440
- Texas Instruments. http://www.ti.com/product/MSP430G2553/technicaldocuments
- Texas Instruments. http://43oh.com/2010/08/10-beginner-msp430-tutorials-and-counting/