第6章 基本时钟和定时器

学习要点

1. 了解总线、总线操作、指令周期的基本概念
2. 掌握MSP430G2553基本时钟模块的设置方法
3. 了解定时器的基本作用和使用方法
4. 掌握利用定时器比较功能实现PWM的方法

第6章 基本时钟和定时器

第1节 有关概念介绍

一、 主频，外频， 倍频系数

二、 T状态

三、 总线周期

四、 指令周期

第2节 基本时钟模块

一、基本时钟模块结构图 Diagram

二、基本时钟模块有关引脚

三、基本时钟模块相关寄存器 Registers

四、msp430G2553.h的符号定义 #define

五、基本时钟设置举例 Examples

第3节 定时器概述

一、定时器的作用

二、MSP430内部的定时器

三、TA的比较功能及PWM实现实例

第1节　有关概念介绍

一、 主频，外频，倍频系数

二、 T状态

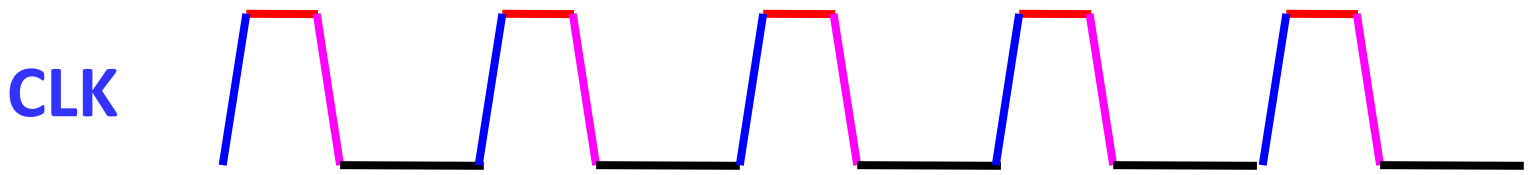
三、 总线周期

四、 指令周期

一、主频，外频，倍频系数

■ CPU是在时钟信号的控制下工作

时钟信号是一个按一定电压幅度、一定时间间隔发出的脉冲信号



f	32.768KHz	8MHz	133MHz
T	30.5ms	125ns	7.5ns

频率f：1秒内的脉冲个数

周期 T = 1/ f

占空比：高电平在一个周期中的比例

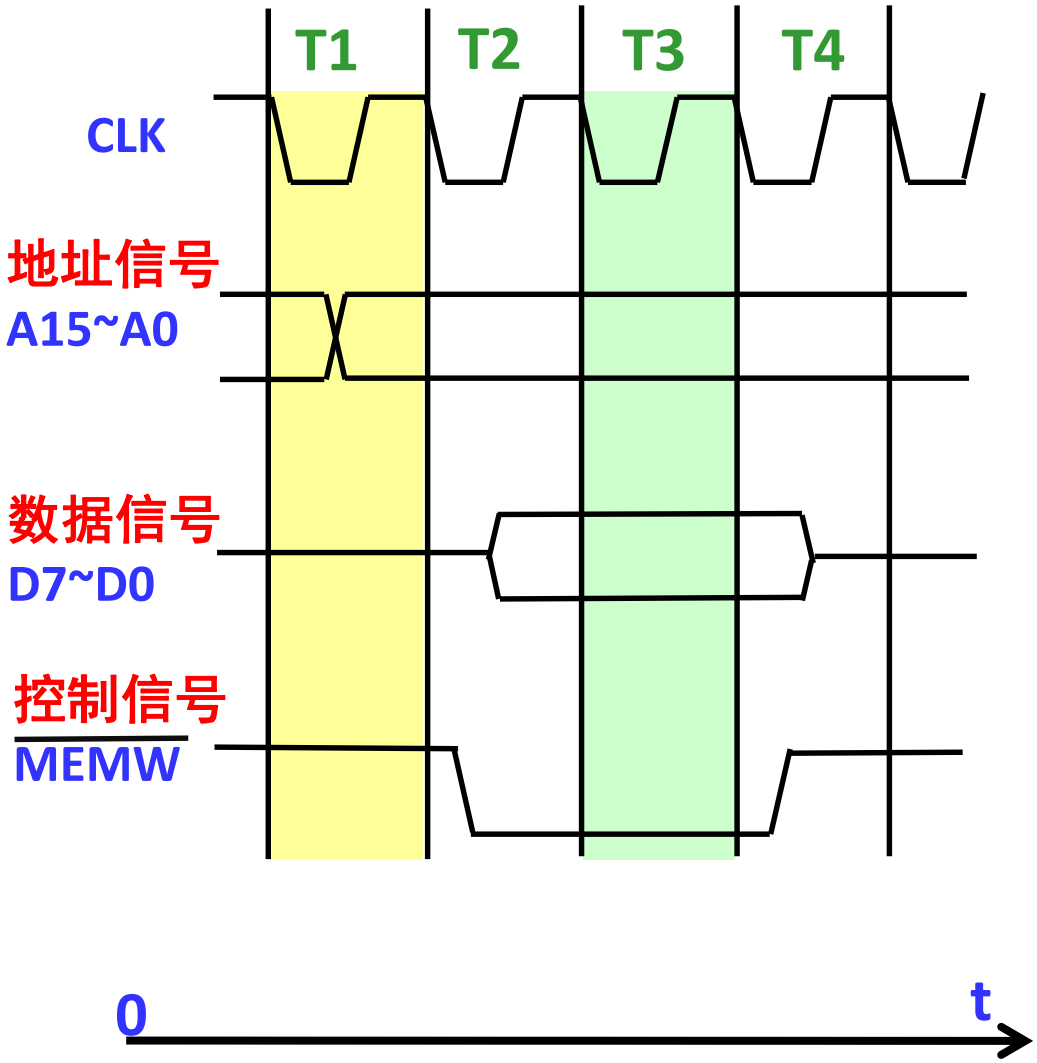
■ CPU所有的操作都以时钟信号为基准

CPU 按严格的时间标准发出地址，控制信号，

存储器、接口模块也按严格的时间标准送出或接收数据.

这个时间标准就是由时钟信号确定。

某总线上CPU向存储单元进行一次写操作的过程



信号变化过程：

①T1:

A15~A0上出现地址信号

②T2

$\overline{\text{MEMW}}$ 变低

③T2

D7~D0 上出现数据信号

④ T4

$\overline{\text{MEMW}}$变高,

数据写入 存储器单元

- CPU的主频或内频指控制CPU内部工作的时钟信号频率

  主频是表示CPU工作速度的重要指标，

  在 CPU其它性能指标相同时，主频越高，CPU 的速度越快

- CPU的外频或系统频率指控制CPU的外部总线工作的时钟频率。

  外频越高，微处理器与存储系统数据交换的速度越快，

  因而计算机的运行速度也越快。

- 倍频系数指CPU主频和外频的相对比例系数。

  8088/8086/80286/80386的主频和外频值相同;

  从80486DX2开始，CPU的主频和外频不再相同，

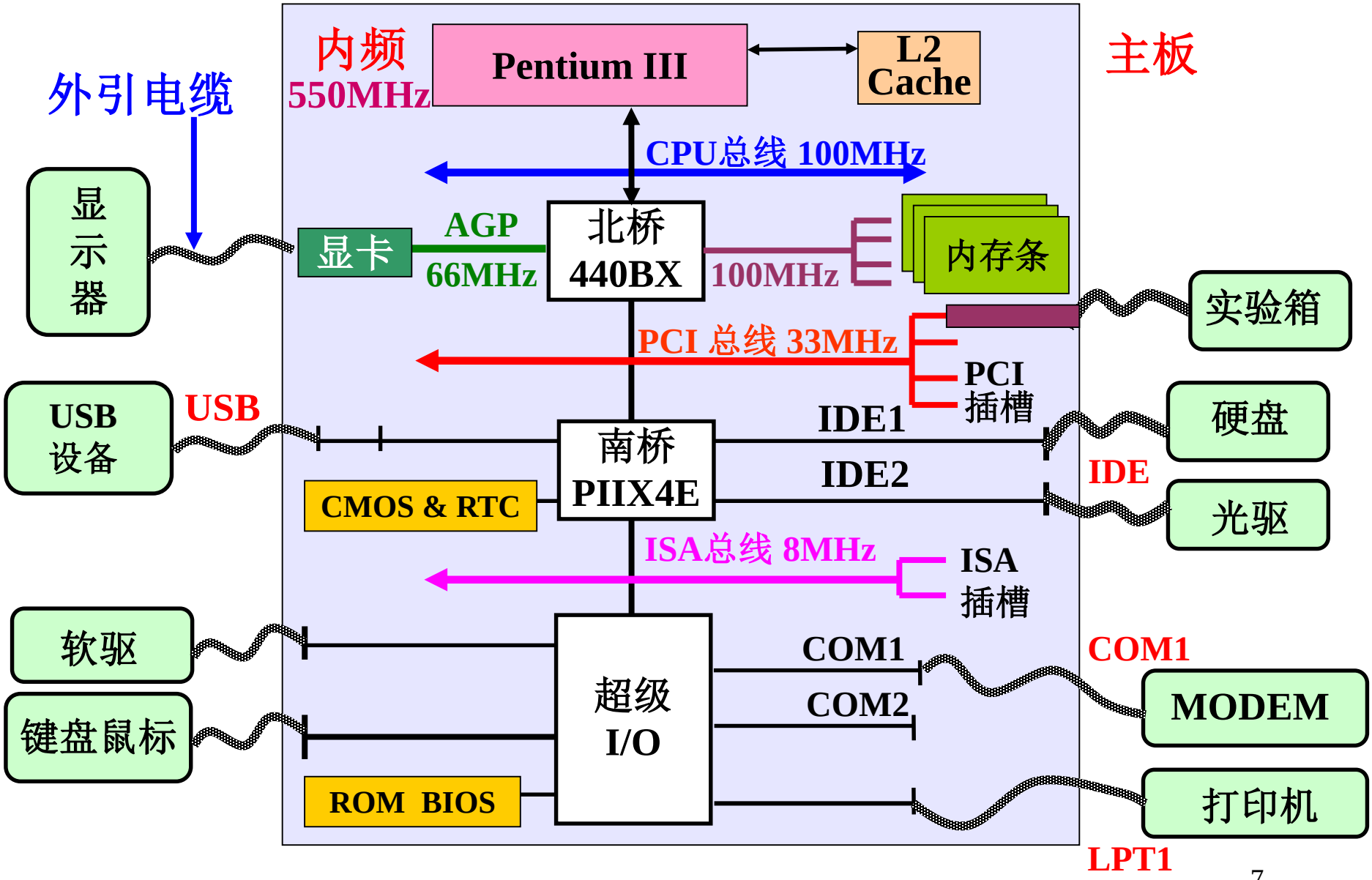
  将外频按一定的比例倍频后得到CPU的主频，即：

  CPU主频 = 外频 × 倍频系数

  现在PC机外频一般是200～333MHz，倍频是6~11

- PC机各子系统时钟(存储系统，显示系统，总线等)是由系统频率按照一定的比例分频得到。

倍频系数5.5

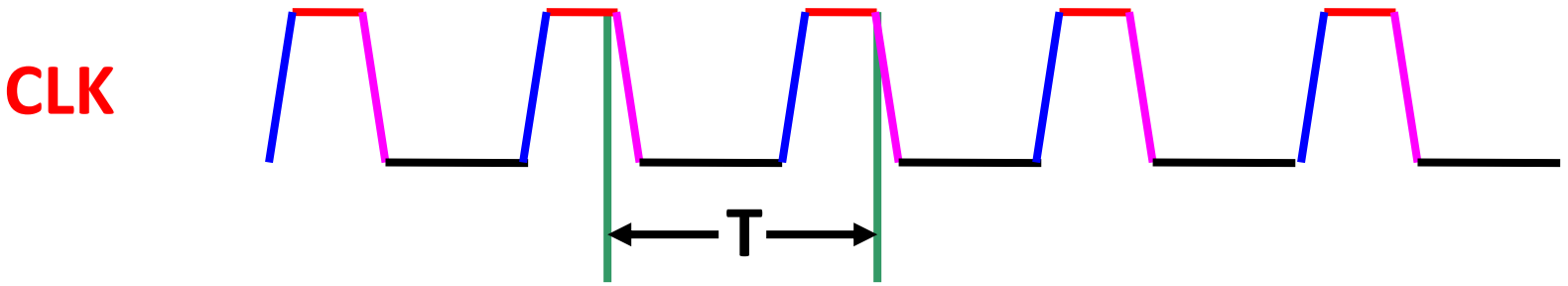

内频
550MHz
Pentium III
L2
Cache
主板
外引电缆
CPU总线 100MHz
显示器
显卡
AGP
66MHz
北桥
440BX
100MHz
内存条
实验箱
PCI 总线 33MHz
PCI
插槽
USB
设备
USB
IDE1
硬盘
南桥
PIIX4E
IDE2
IDE
CMOS & RTC
光驱
ISA总线 8MHz
ISA
插槽
软驱
COM1
COM1
超级
I/O
COM2
MODEM
键盘鼠标
ROM BIOS
打印机
LPT1

二、T状态

- 相邻两个脉冲之间的时间间隔，

  称为一个时钟周期，又称 T状态（T周期）。

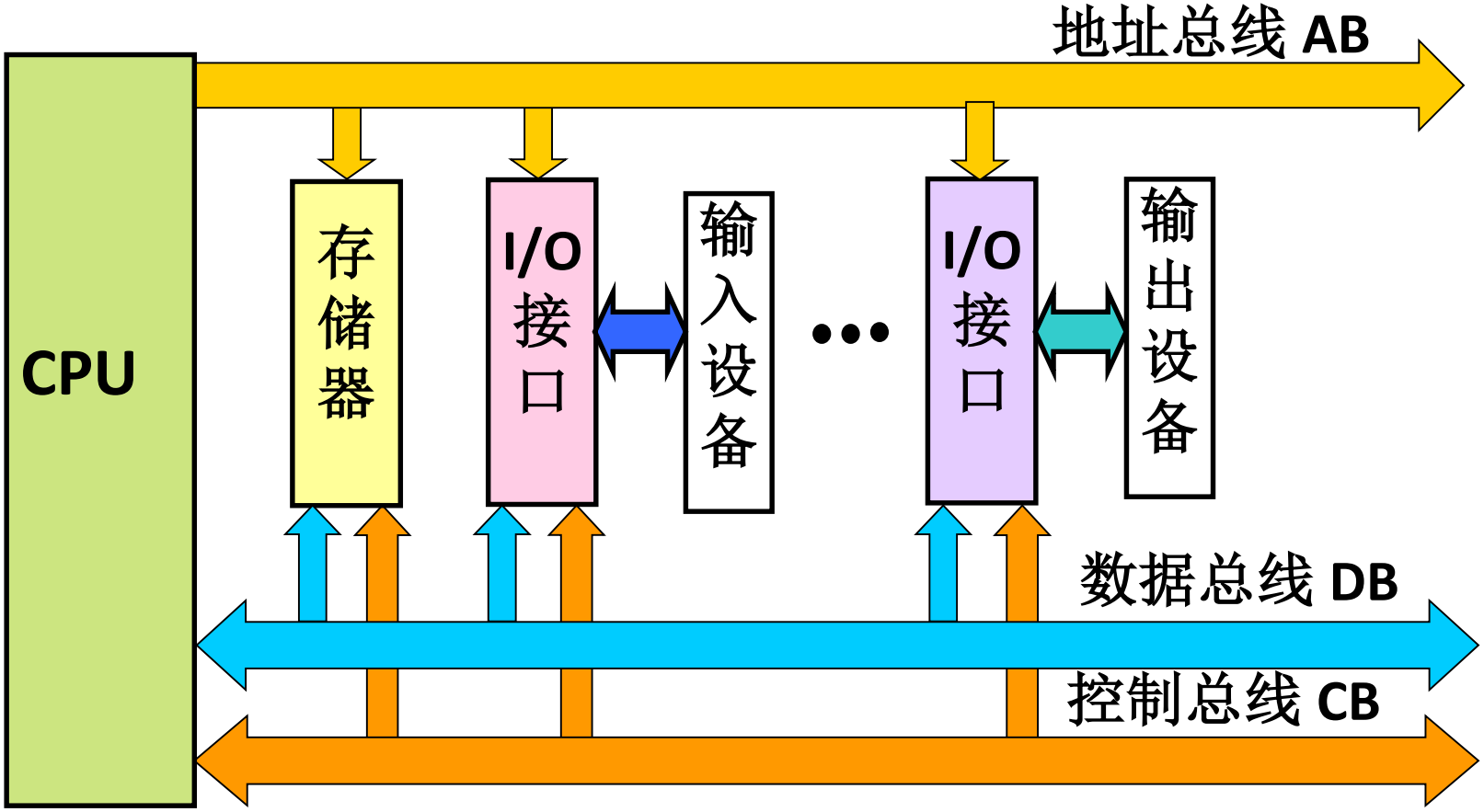
  时钟周期是CPU处理动作的最小时间单位。



- 每个T状态包括： 下降沿、低电平、上升沿、高电平

三、总线周期

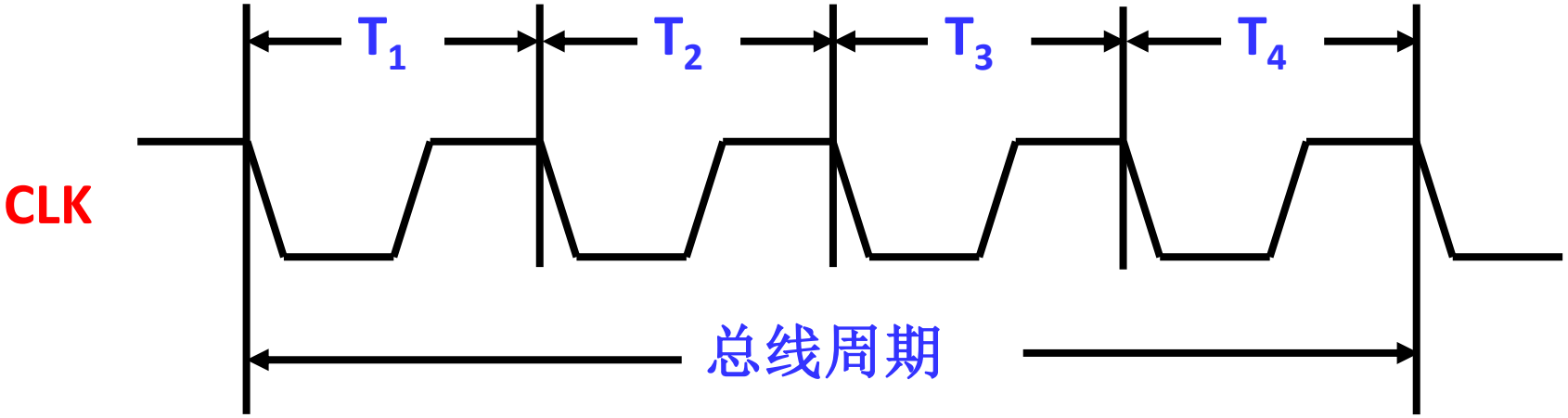
- CPU通过总线完成与存储器、I/O端口之间的操作，这些操作统称为总线操作。



- 执行一个总线操作所需要的时间称为总线周期。
  不同的总线操作有不同的总线周期。

总 线 操 作	总线周期
读存储器操作 (取指令、取操作数)	存储器读周期
写存储器操作 (将结果存放到内存)	存储器写周期
读 I/O 端口操作 (取 I/O 端口中的数)	I/O 端口读周期
写 I/O 端口操作 (往 I/O 端口写数)	I/O 端口写周期
中断响应操作	中断响应周期

- 一个基本的总线周期通常包含几个T状态，
  按时间的先后顺序分别称为T_1、T_2、T_3、T_4 ……



四、指令周期

- 执行一条指令所需要的时间称为指令周期。

  执行一条指令的时间:

  是取指令、执行指令、取操作数、存放结果所需时间的总和。

  用所需的时钟周期数表示。

例

MOV R4, R6	1个T周期
MOV 2(R5),R15	3个T周期
ADD 4(R5), 8(R15)	6个T周期

- 指令中操作数的寻址方式影响指令周期
- 参看 msp430x2xx 用户指南-中文.pdf P60、P61了解各指令和时钟周期个数的关系

内联函数 void __delay_cycles(unsigned long cycles)

- 通过指定CPU执行程序的周期数，控制延时时间的长短。其中参数cycles 指定延时的周期数。
- 执行 __delay_cycles(unsigned long cycles) 需要的时间为：
  = cycles*单片机CPU的T周期
  = cycles/单片机CPU的工作频率

在C程序中，可调用：

如

```
__delay_cycles(1);          //延时1个时钟周期
__delay_cycles(20);         //延时20个时钟周期
__delay_cycles(100);        //延时100个时钟周期
__delay_cycles(10000);      //延时10000个时钟周期
__delay_cycles(100000);     //延时100000个时钟周期
```

- **CPI (clock cycles per instruction)**

  执行每条指令所需的时钟周期数的平均值

  通常根据不同指令出现的频度, 乘上不同的系数, 求得的统计平均值

  一个程序的CPU时间

  = 一个程序的CPU时钟周期数 x 时钟周期时间

  = 指令数 x CPI x 时钟周期时间

  = 指令数 x CPI / 时钟频率

第2节　MSP430x2xx 基本时钟模块

Basic Clock Module

一、基本时钟模块结构图 *Diagram*

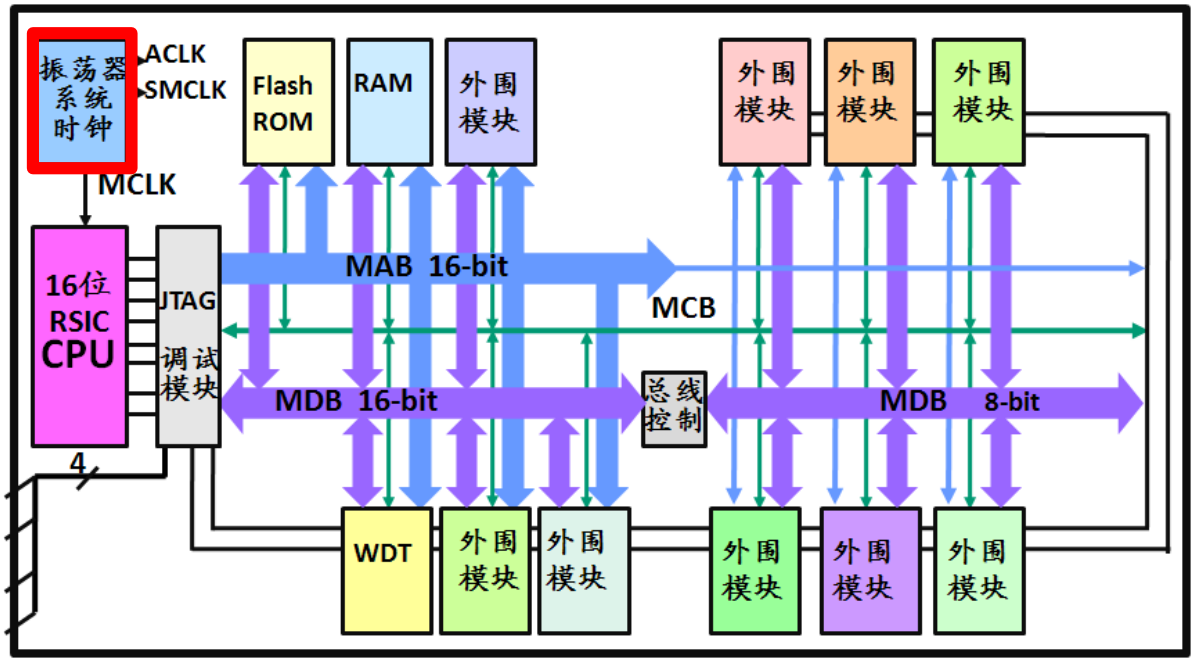
二、基本时钟模块有关引脚

三、基本时钟模块相关寄存器 *Registers*

四、msp430G2553.h的符号定义 *#define*

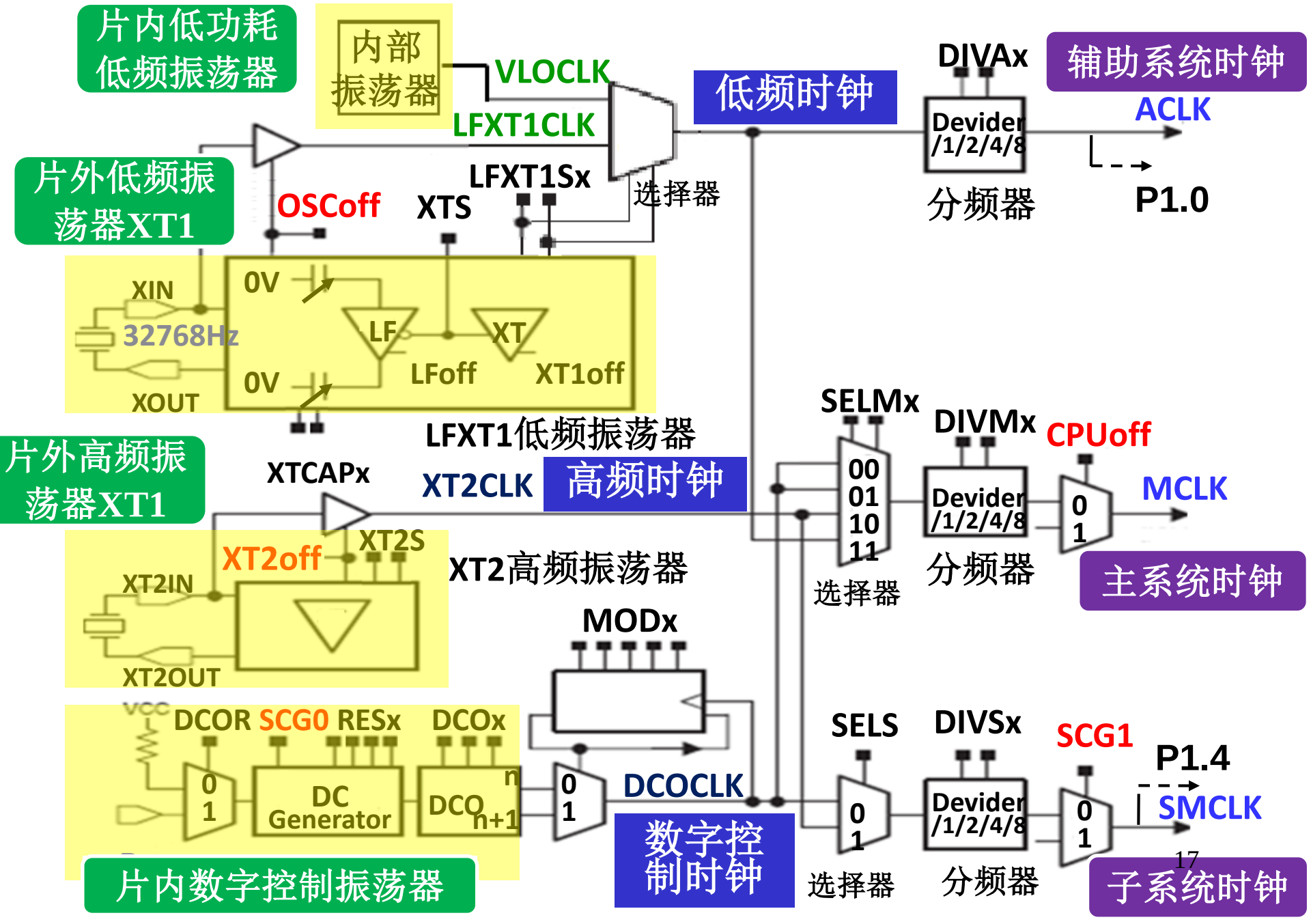
五、基本时钟设置举例 *Examples*

一、 基本时钟模块结构图



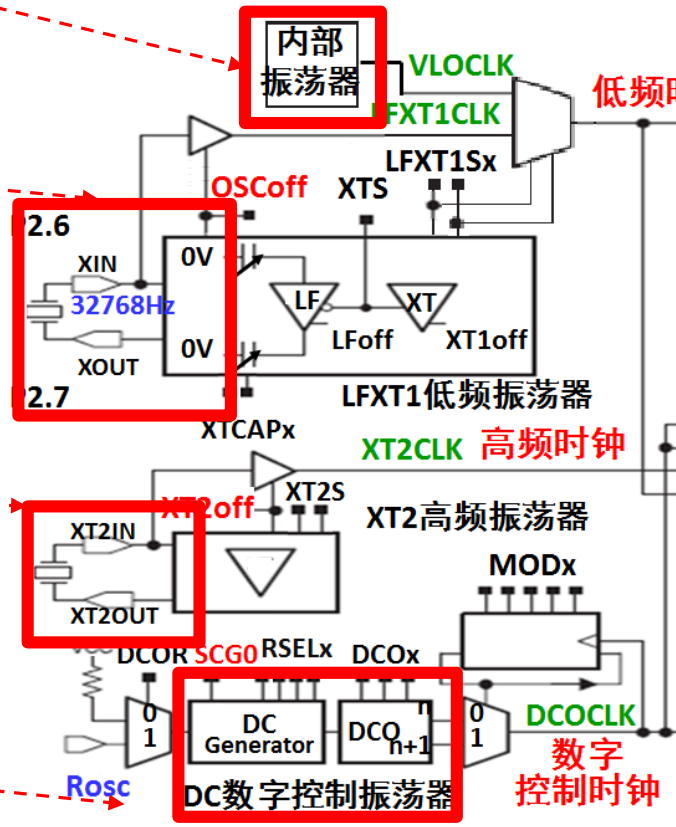
- 时钟信号是定时操作的基本信号，
  在时钟的作用下，各部件可以有条不紊地自动工作
- MSP430单片机基本时钟模块可由高频振荡器、低频振荡器、
  数字控制振荡器 DCO、锁频环FLL等部分构成
- 不同系列MSP430单片机包含的时钟模块会有不同，但都输出3种
  时钟信号: 辅助时钟ACLK、主时钟MCLK、子系统时钟SMCLK
- 多种时钟有利于实时应用系统对低功耗和快速响应外部事件要求

MSP430x2xx基本时钟模块结构图

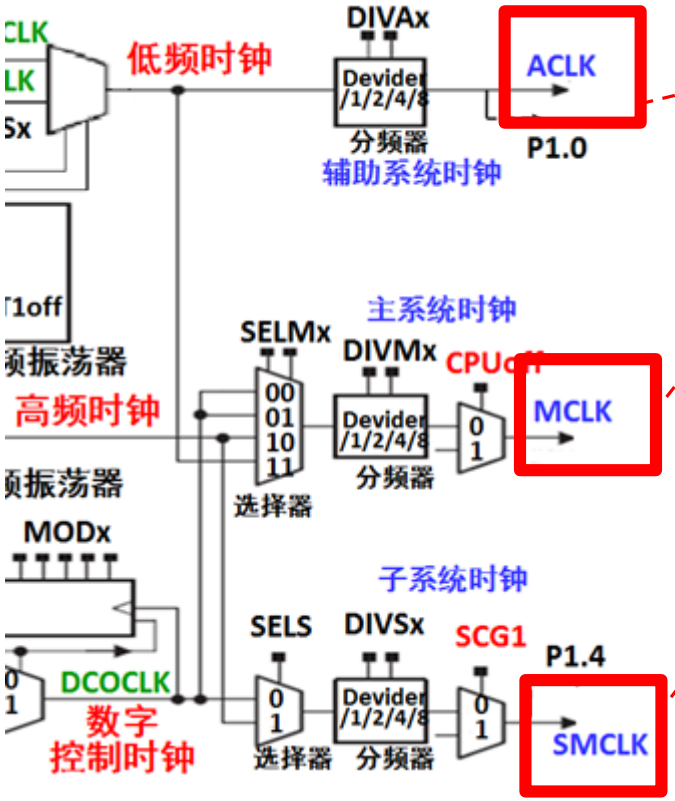


MSP430Gxxx的四个时钟源（振荡器）

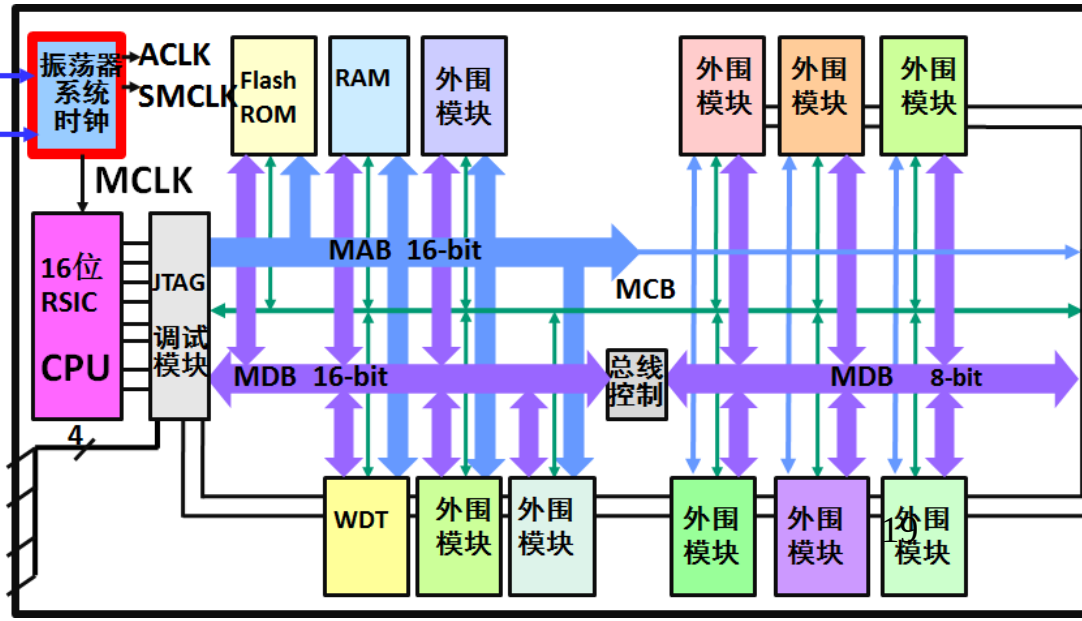
- VLOCLK　片内低功耗低频振荡器，频率为12KHz.
  受温度和工作电压影响大
- LFXT1CLK: XT1振荡器
  由外部通过XIN/XOUT引脚接入振荡器.
  可接低频振荡器，一般是32.768KHz
  有的也可接高频振荡器，频率在
  4MHz~16MHz
- XT2CLK　: XT2振荡器
  由外部通过XIT2N/XT2OUT引脚接入振荡器.
  接高频振荡器，频率在4MHz~16MHz
- DCOCLK　: 片内数字可控RC振荡器.
  能提供较宽的时钟频率，如从几十KHz到16Mhz，
  受温度和工作电压影响大



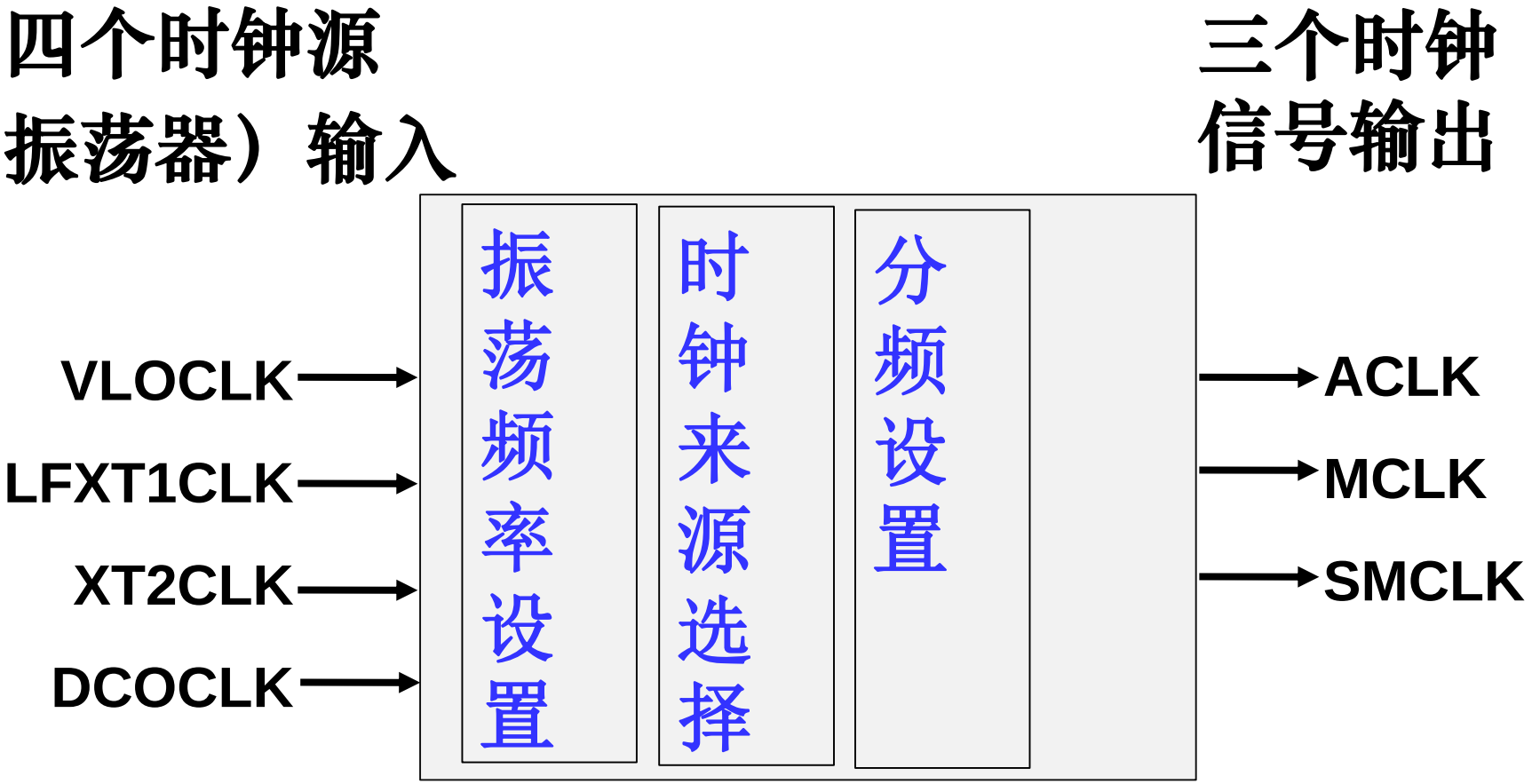
MSP430Gxxx的三个时钟信号



- **ACLK :** **辅助时钟**(Auxiliary clock)
  为需要低频时钟的外围模块提供时钟。
- **MCLK :** **主系统时钟**(main system clock)
  主要提供 CPU工作的时钟
  MCLK频率越高, CPU工作速度越快;
  在不用CPU时,可关闭 MCLK, 让CPU休眠, 降低功耗
- **SMCLK:** **子系统时钟**(Sub main clock)
  为需要高速时钟的外围模块提供时钟,
  在CPU 休眠时, 可由SMCLK提供时钟给这些模块

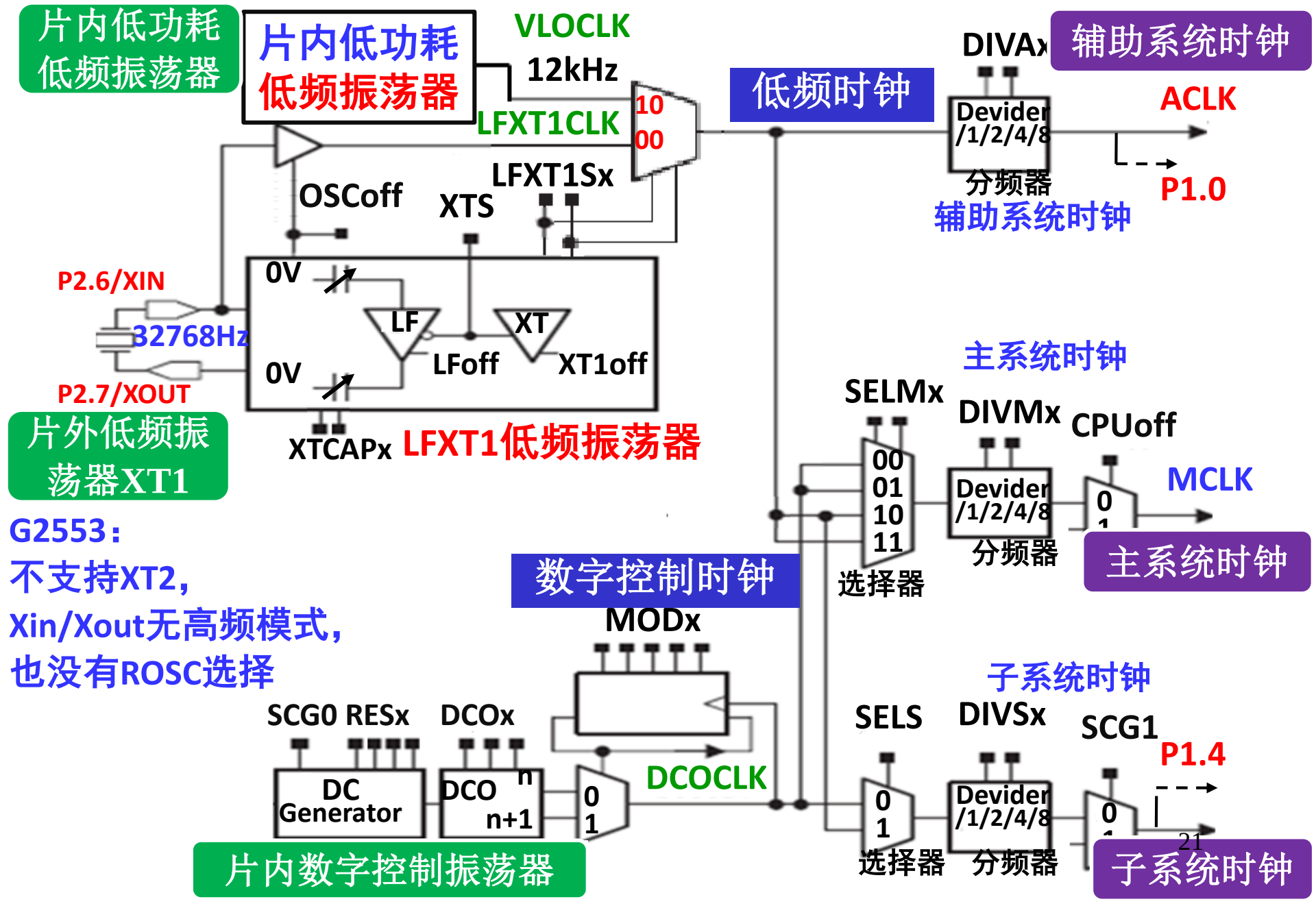


MSP430Gxxx的时钟源和时钟信号



MSP430G2553没有X2CLK

二、msp430G2553基本时钟模块有关引脚



- **LFXT1CLK:　XT1振荡器**

  由外部通过XIN/XOUT引脚接入振荡器.

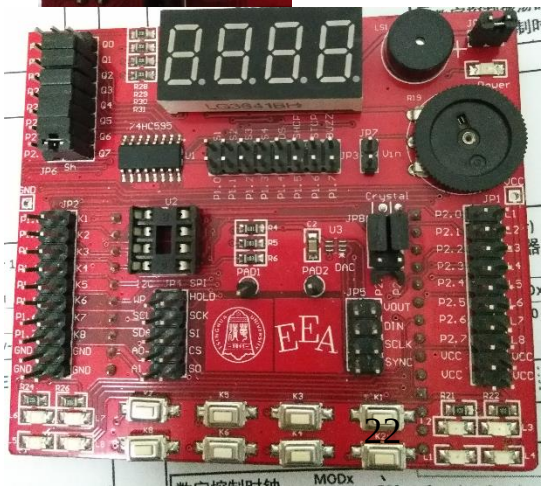
  可接低频振荡器，一般是32.768KHz

引脚名	功能	x	相关控制位的设置				
			P2DIR.x	P2SEL.6	P2SEL.7	P2SEL2.6	P2SEL2.7
P2.6	XIN	6	0	1	1	0	0
P2.7	XOUT	7	1	1	1	0	0

编程置引脚P2.6、P2.7做连接外部晶振的引脚

```
P2SEL |= BIT6;        //设P2.6为晶振引脚功能
P2SEL2 &= ~BIT6;
P2DIR &= ~BIT6;       //P2.6 XIN 输入

P2SEL |= BIT7;        //设P2.7为晶振引脚功能
P2SEL2 &= ~BIT7;
P2DIR |= BIT7;        //P2.7 Xou 输出
```

实验板上晶振



通过跳线将晶振接入P2.6、P2.7

ACLK和SMCLK可以由引脚输出到片外

引脚名	功能	相关控制位的设置				
		P1DIR.0	P1SEL.0	P1SEL2.0	ADC10AE.0 INCH.0=1	CAPD.0
P1.0	ACLK	1	1	0	0	0

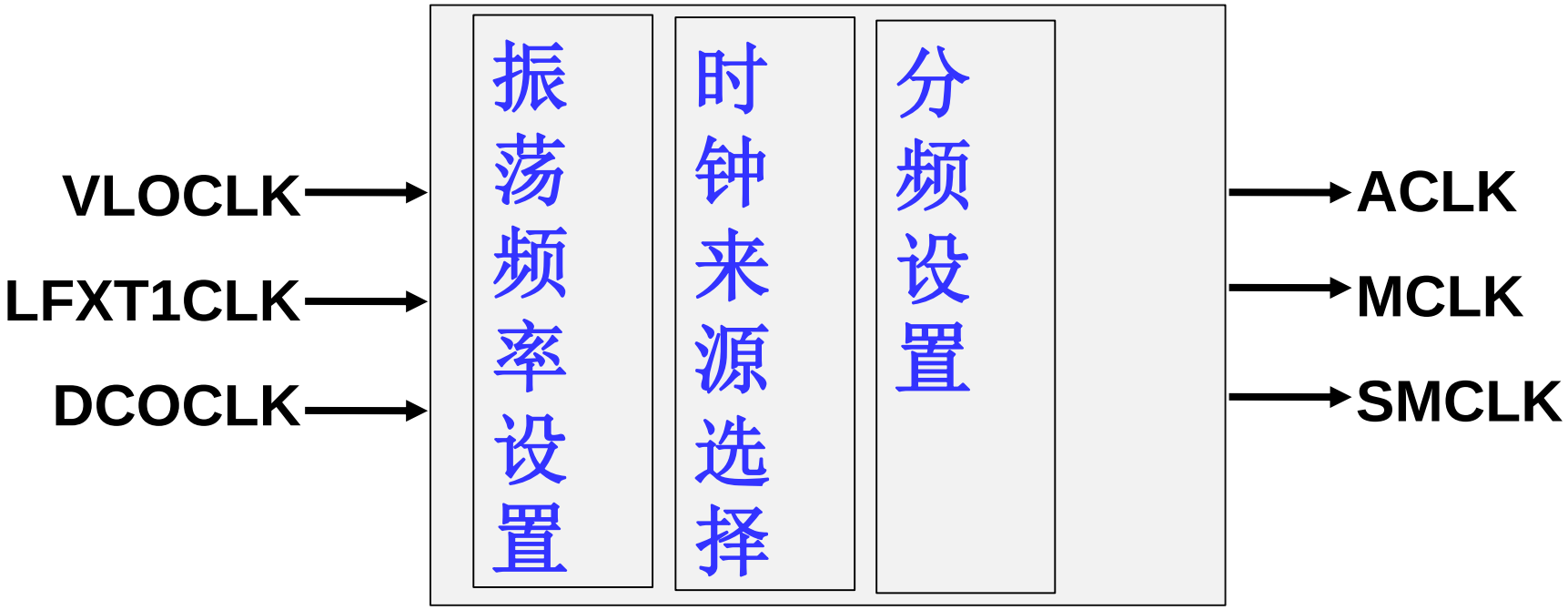
引脚名	功能	相关控制位的设置					
		P1DIR.4	P1SEL.4	P1SEL2.4	ADC10AE.4	JTAG Mode	CAPD.y
P1.4	SMCLK	1	1	0	0	0	0

编程置引脚P1.0、P1.4分别输出ACLK、SMCLK

```
P1SEL |=BIT0;          //P1.0 输出ACLK
P1SEL2 &= ~BIT0;
P1DIR |=BIT0;

P1SEL |=BIT4;          //P1.4 输出SMCLK
P1SEL2 &= ~BIT4;
P1DIR |=BIT4;
```

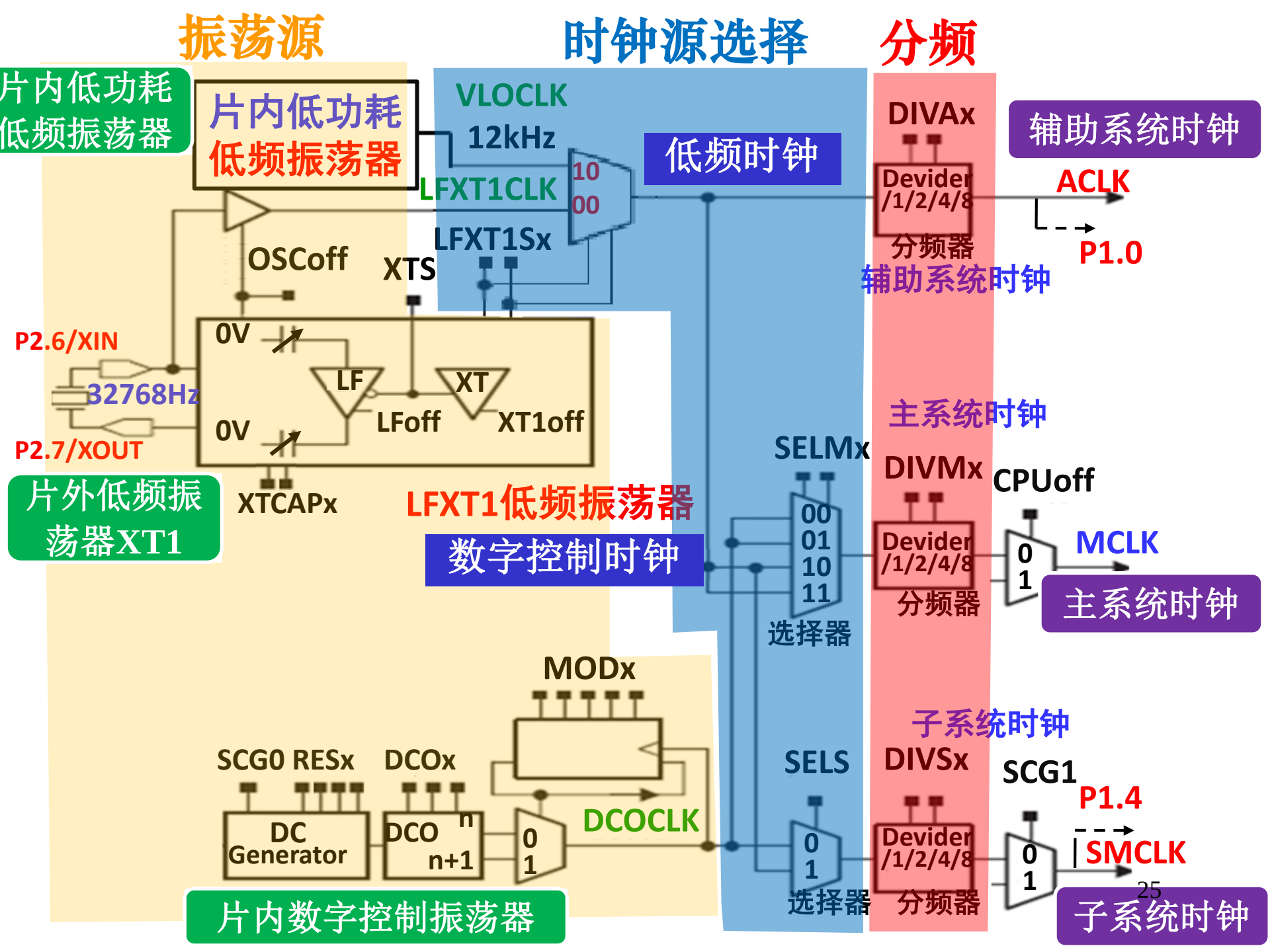
三、基本时钟模块相关寄存器

■ MSP430G2553



MSP430G2553时钟模块编程结构图

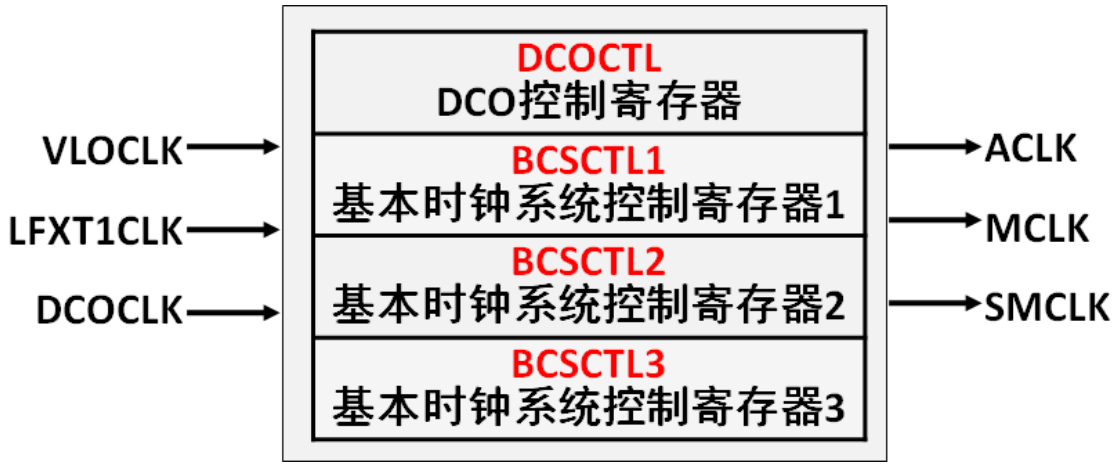
- 三个时钟源（振荡器）输入
- 三个时钟输出

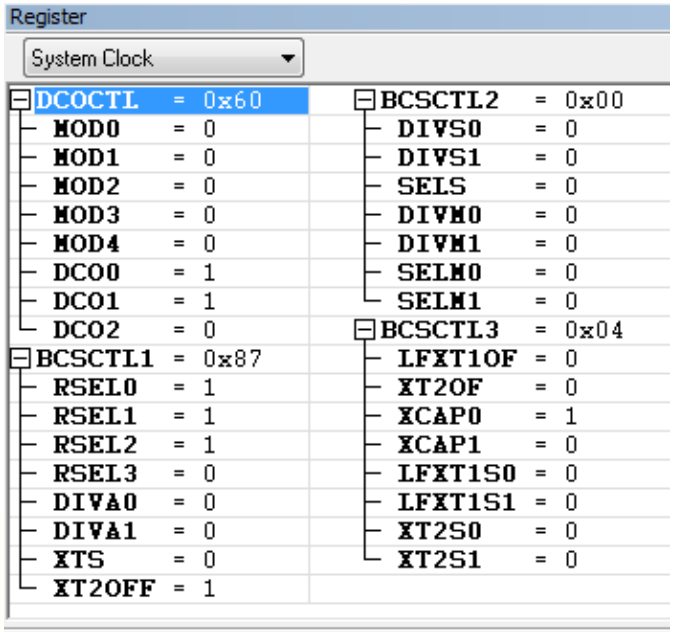

振荡源
时钟源选择
分频
片内低功耗
低频振荡器
片内低功耗
低频振荡器
VLOCLK
12kHz
LFXT1CLK
LFXT1Sx
10
00
低频时钟
DIVAx
辅助系统时钟
Devider
/1/2/4/8
分频器
ACLK
P1.0
辅助系统时钟
OSCoff
XTS
P2.6/XIN
32768Hz
P2.7/XOUT
0V
0V
LF
XT
LFoff
XT1off
XTCAPx
片外低频振
荡器XT1
LFXT1低频振荡器
数字控制时钟
主系统时钟
SELMx
DIVMx
CPUoff
00
01
10
11
选择器
Devider
/1/2/4/8
分频器
0
1
MCLK
主系统时钟
MODx
子系统时钟
SCG0 RESx
DCOx
SELS
DIVSx
SCG1
P1.4
DC
Generator
DCO
n
n+1
DCOCLK
0
1
选择器
Devider
/1/2/4/8
分频器
0
1
SMCLK
片内数字控制振荡器
子系统时钟

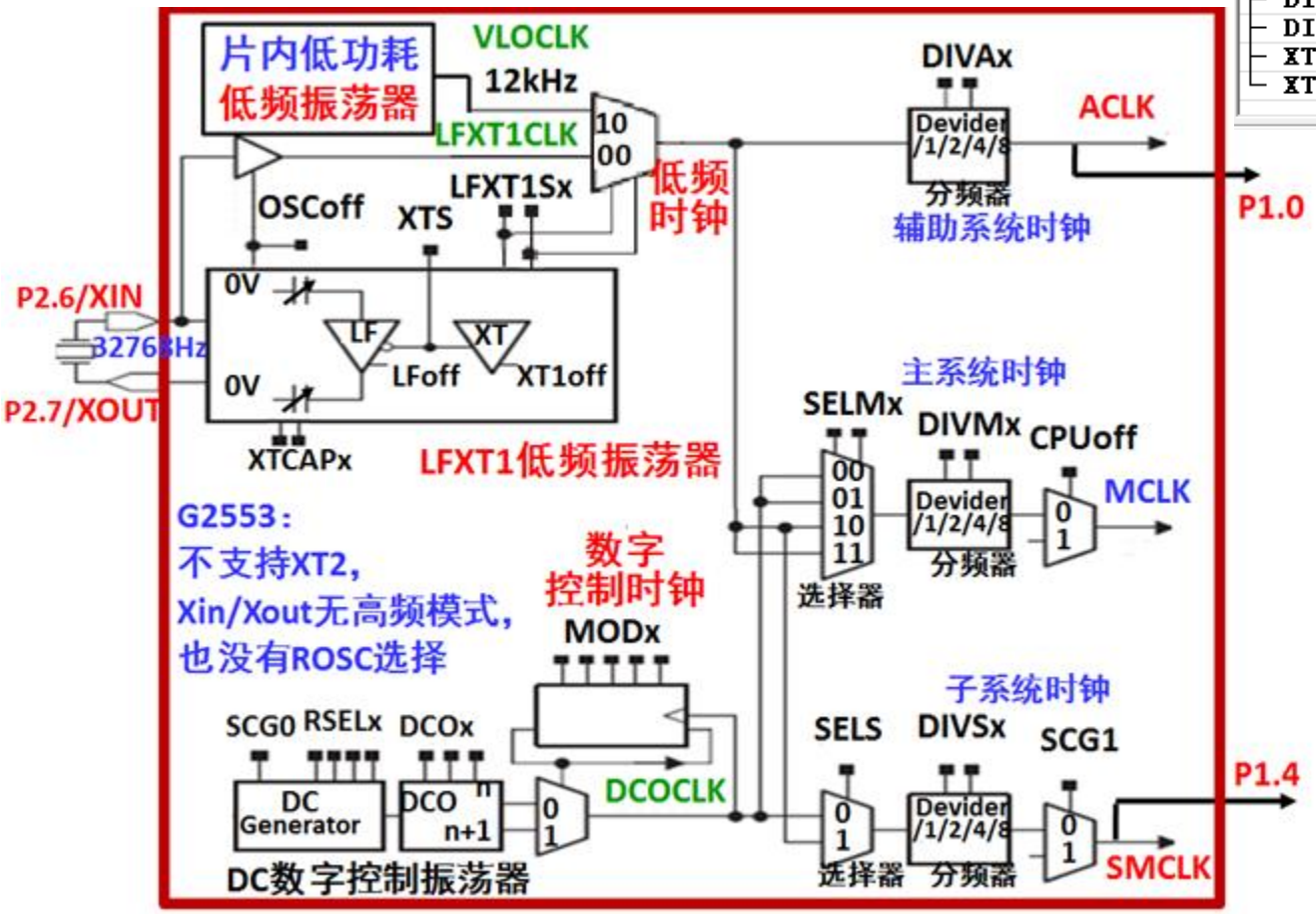
■ MSP430G2553内部有四个控制寄存器

MSP430G2553时钟模块编程结构图

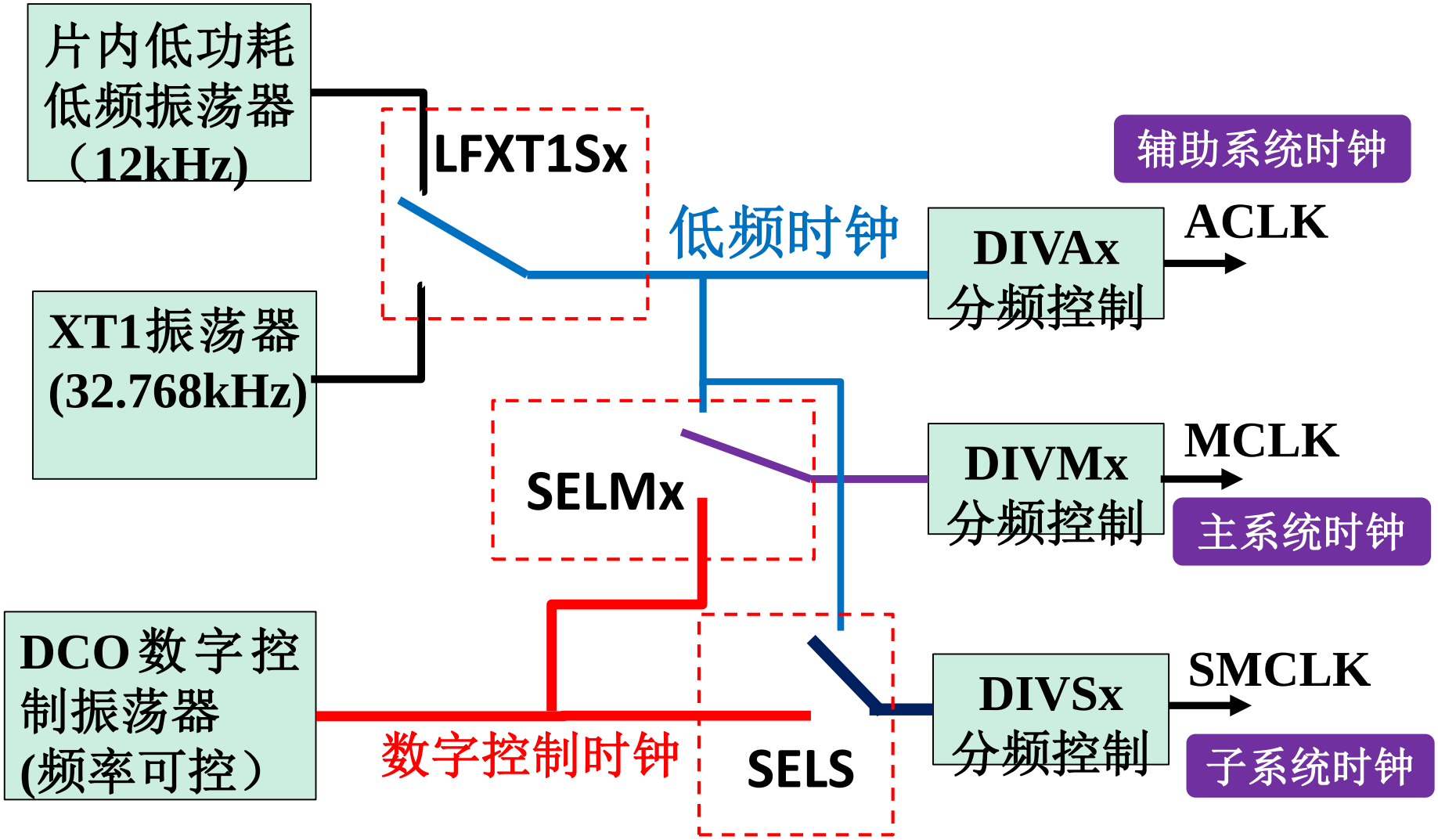
通过这4个端口寄存器，对于振荡源频率、振荡源选择、分频等进行控制，得到三个输出时钟信号







结构图中的
各控制位存放
在系统时钟模块
的端口寄存器中
（如图）


片内低功耗
低频振荡器
（12kHz)
LFXT1Sx
辅助系统时钟
低频时钟
DIVAx
分频控制
ACLK
XT1振荡器
(32.768kHz)
SELMx
DIVMx
分频控制
MCLK
主系统时钟
DCO数字控
制振荡器
(频率可控）
数字控制时钟
SELS
DIVSx
分频控制
SMCLK
子系统时钟

1） 振荡源频率设置

- VLOCLK ：片内低频振荡器，频率固定为12KHz.
  受温度和工作电压影响大

- LFXT1CLK: XT1振荡器，频率取决于外接的晶体振荡器，一般是32.768KHz

- DCOCLK ：片内数字可控RC振荡器.
  振荡频率数字可控
  受温度和工作电压影响大

MSP430G 系列单片机只能通过内部DCO 来获得高频时钟

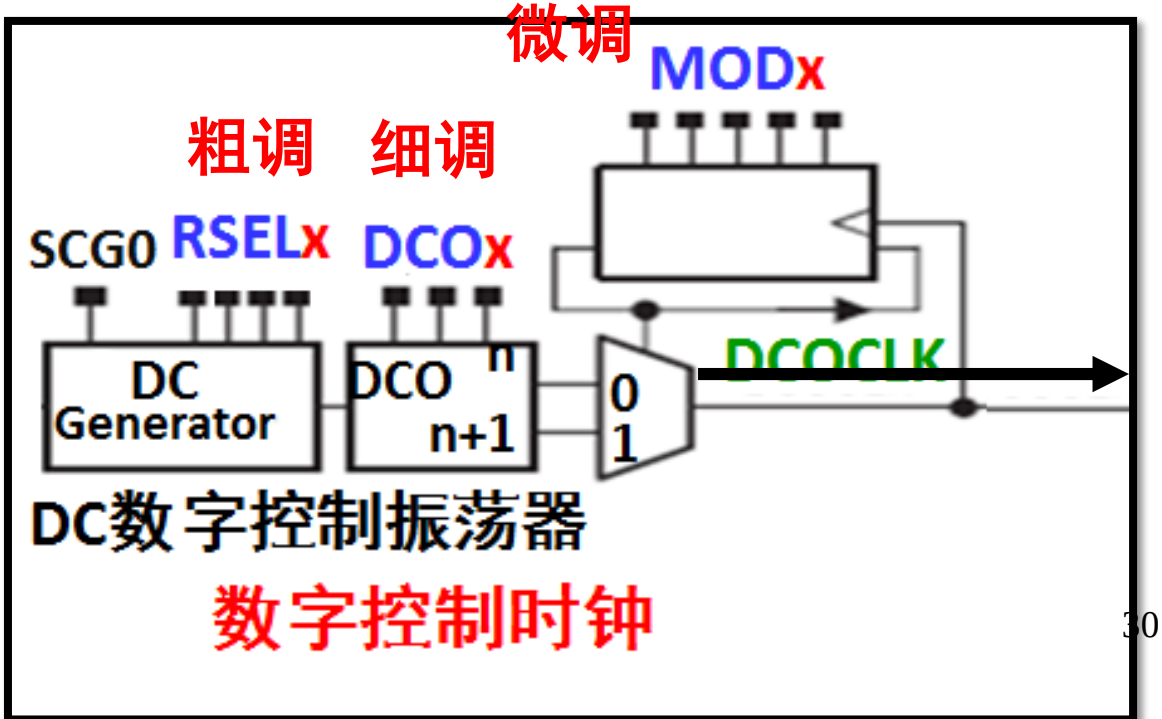
DCO振荡器的频率设定

能提供较宽的时钟频率，如从几十KHz到16Mhz，

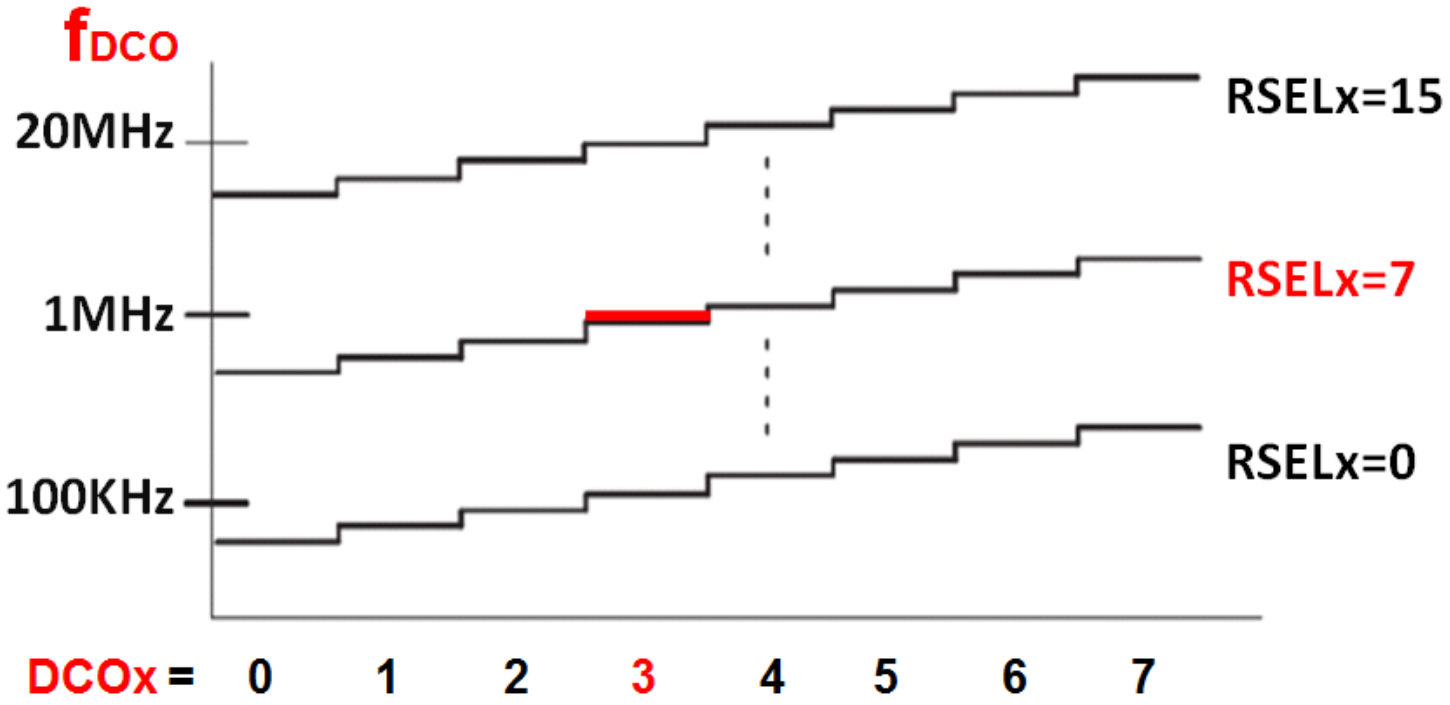
- 数字控制振荡器DCO内置系列电阻，供选择频率范围。

  DCOCTL和BCSCTL1寄存器中：

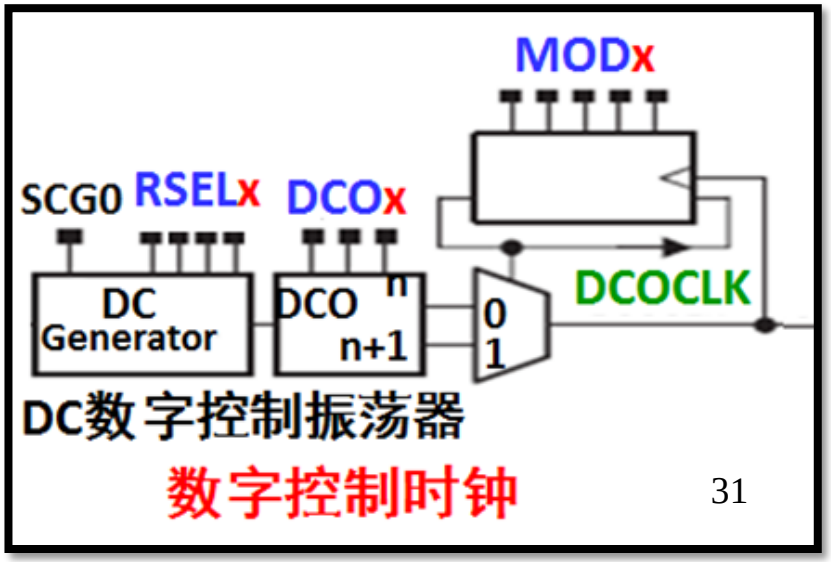
  - 4位RSELx负责粗调，16档;
  - 3位DCOx负责细调，共 8 档，档位步进约 10%
  - 5位MODx负责微调 (不介绍，可自学)



典型DCO振荡频率F_{DCO}范围及其分档设置示意图



其中DCOx和RSELx
在DCOCTL和BCSCTL1
中设置

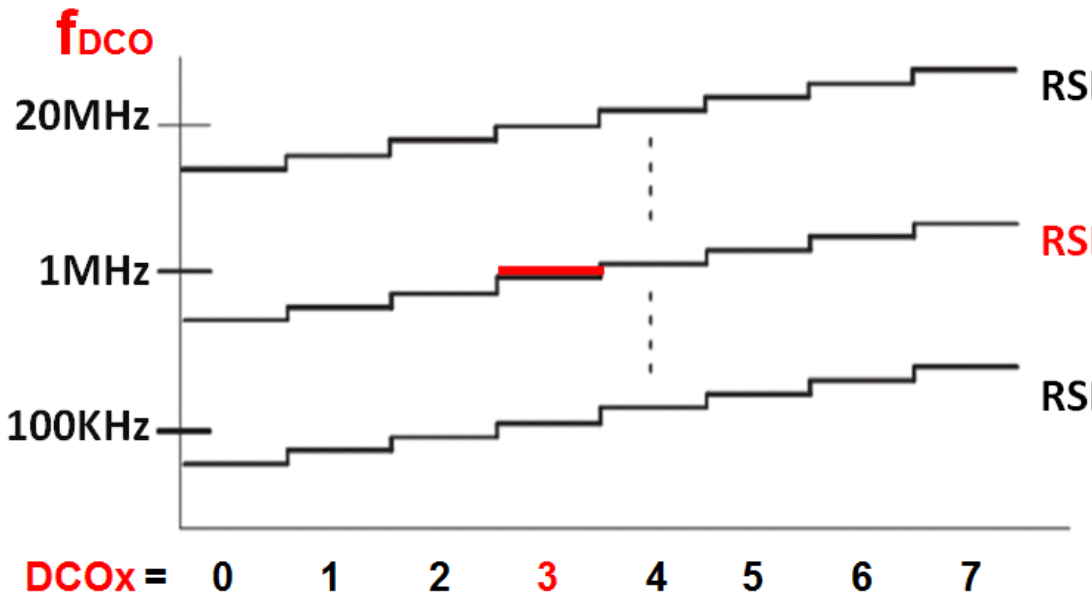


BCSCTL1 (Basic Clock System Control Register 1)

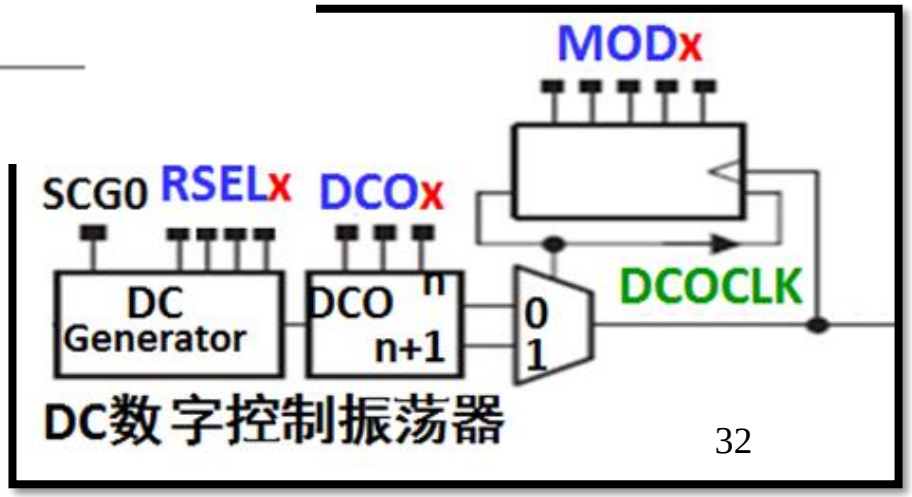
基本时钟系统控制寄存器1

7	6	5	4	3	2	1	0
		DIVAx		RSELx			
rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-1	rw-1	rw-1

设置RSELx值



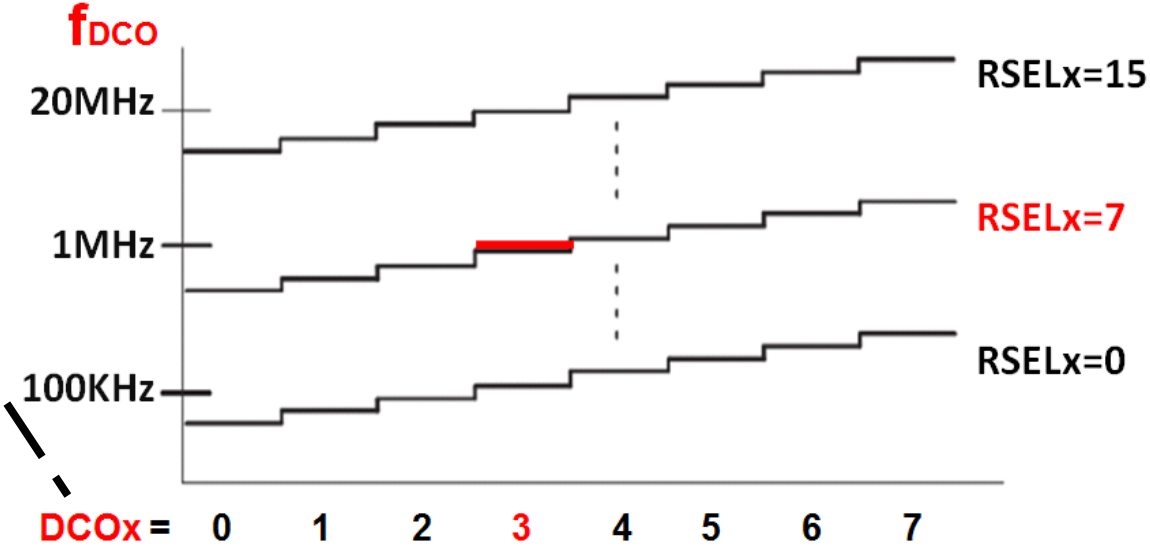
数字控制时钟

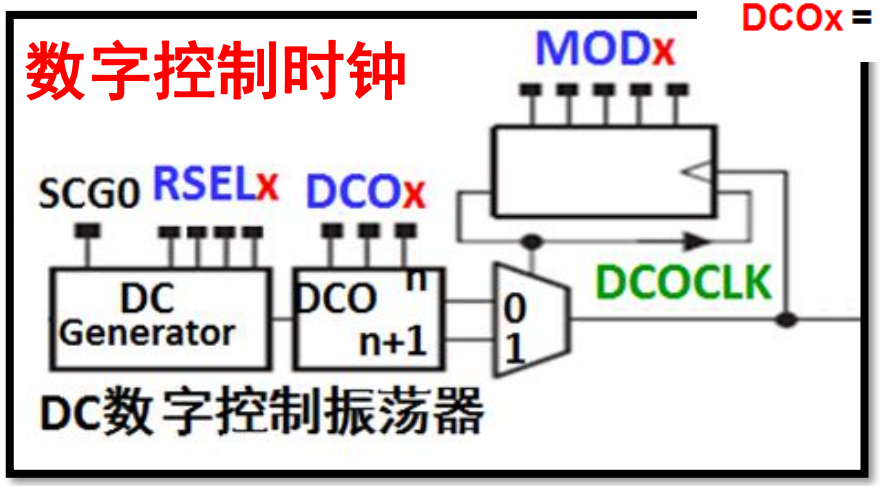


DCOCTL (DCO Control Register)
数字控制振荡器控制寄存器

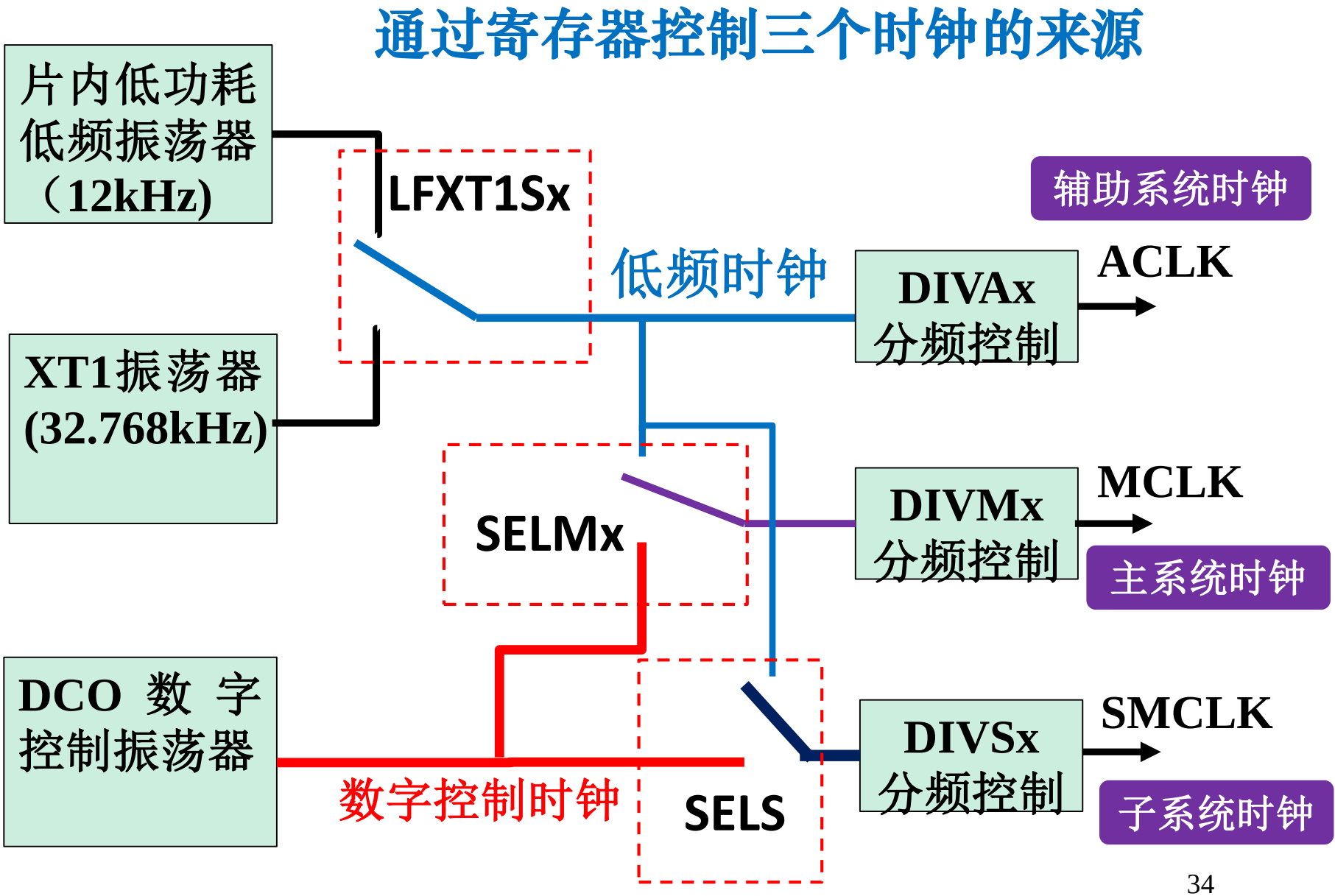
7	6	5	4	3	2	1	0
DCOx			MODx				
rw-0	rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0

设置DCOx值





2） 时钟源选择


通过寄存器控制三个时钟的来源
片内低功耗低频振荡器（12kHz)
LFXT1Sx
辅助系统时钟
低频时钟
DIVAx 分频控制
ACLK
XT1振荡器(32.768kHz)
SELMx
DIVMx 分频控制
MCLK
主系统时钟
DCO 数字控制振荡器
数字控制时钟
SELS
DIVSx 分频控制
SMCLK
子系统时钟

BCSCTL3 (Basic Clock System Control Register 3)
基本时钟系统控制寄存器3

7	6	5	4	3	2	1	0
		LFXT1Sx					
rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	r0	r-(1)

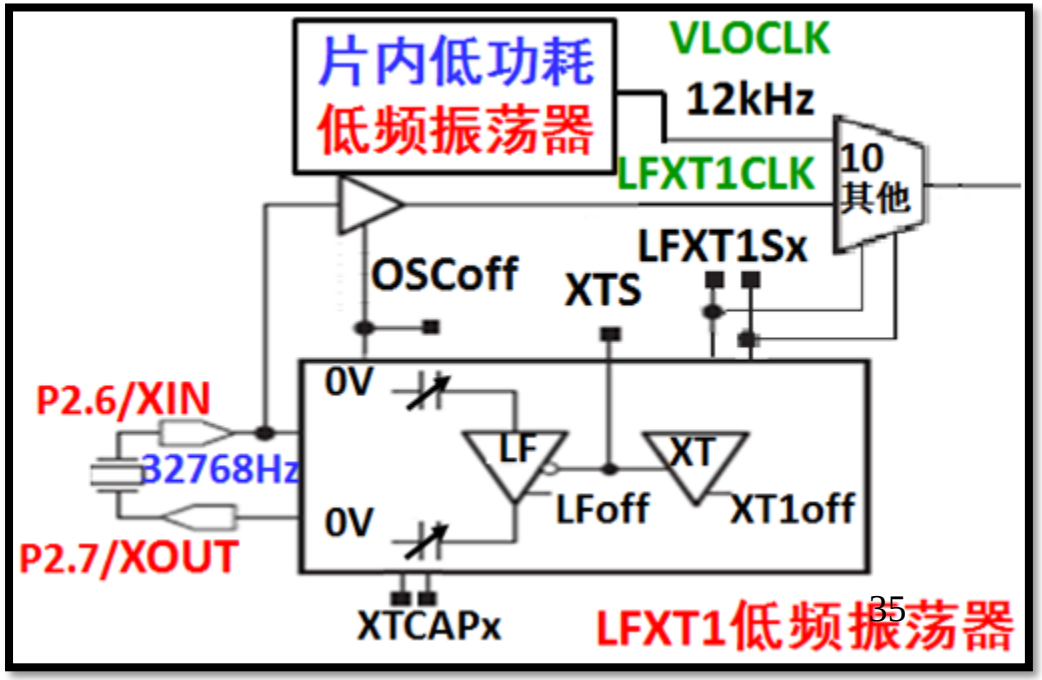
选择低频时钟

00： LFXT1外部晶振

10：VLOCLK，
12kHz

01、11： 保留未用

- DCOCTL DCO控制寄存器
- BCSCTL1 基本时钟系统控制寄存器1
- BCSCTL2 基本时钟系统控制寄存器2
- BCSCTL3 基本时钟系统控制寄存器3

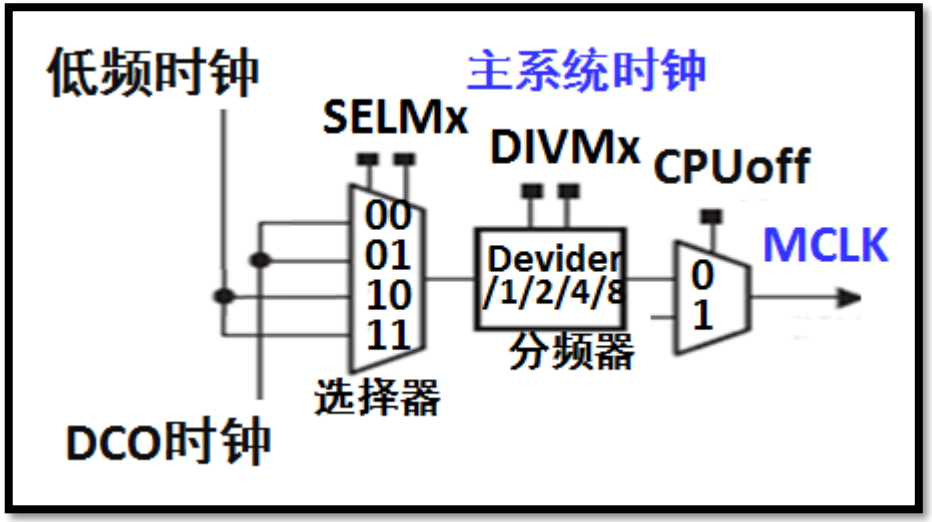


BCSCTL2 (Basic Clock System Control Register 2)
基本时钟系统控制寄存器2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0	rw-0	rw-0

选择MCLK
select MCLK
00: DCOCLK
01: DCOCLK
10:低频时钟
11:低频时钟

低频时钟为：
LFXT1CLK or
VLCLK



BCSCTL2 (Basic Clock System Control Register 2)

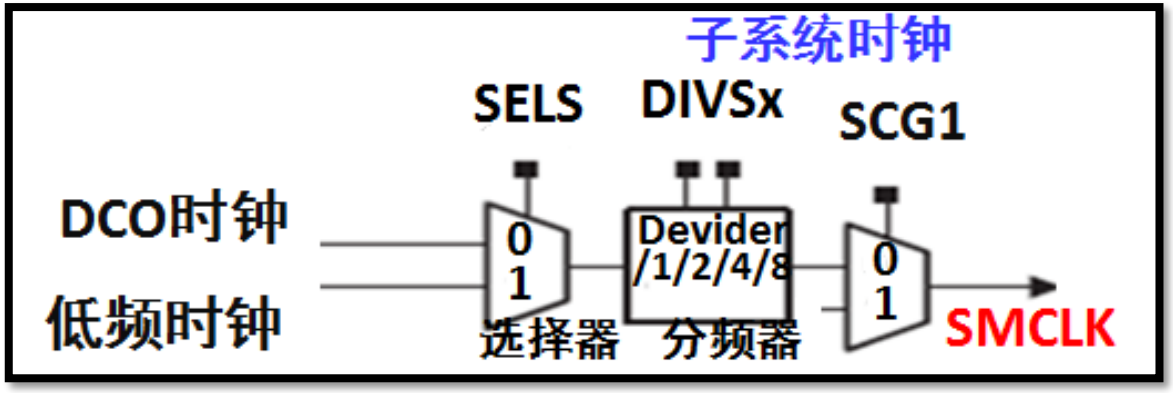
基本时钟系统控制寄存器2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0	rw-0	rw-0

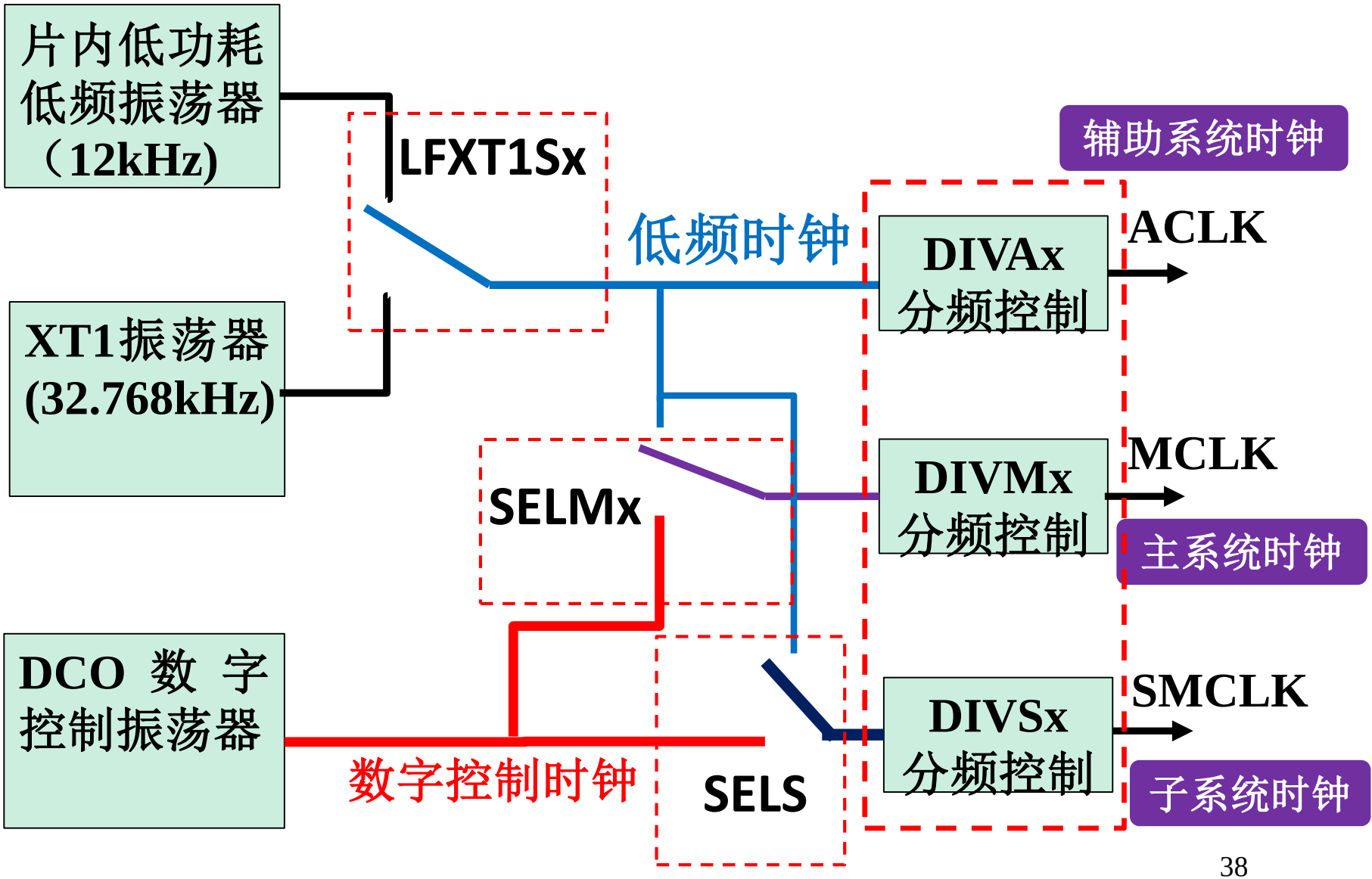
选择SMCLK

0:DCOCLK

1：低频时钟



3） 分频控制


片内低功耗
低频振荡器
（12kHz)
XT1振荡器
(32.768kHz)
DCO 数 字
控制振荡器
LFXT1Sx
SELMx
SELS
低频时钟
数字控制时钟
DIVAx
分频控制
DIVMx
分频控制
DIVSx
分频控制
辅助系统时钟
ACLK
MCLK
主系统时钟
SMCLK
子系统时钟

BCSCTL1 (Basic Clock System Control Register 1)

基本时钟系统控制寄存器1

7	6	5	4	3	2	1	0
		DIVAx		RSELx			
rw-(1)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-1	rw-1	rw-1

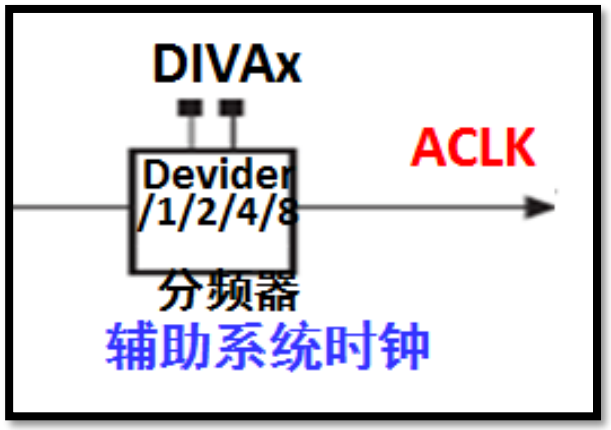
ACLK分频控制

00 : /1, 不分频

01 : /2, 2分频

10 : /4, 4分频

11 : /8, 8分频



BCSCTL2 (Basic Clock System Control Register 2)

基本时钟系统控制寄存器2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0	rw-0	rw-0

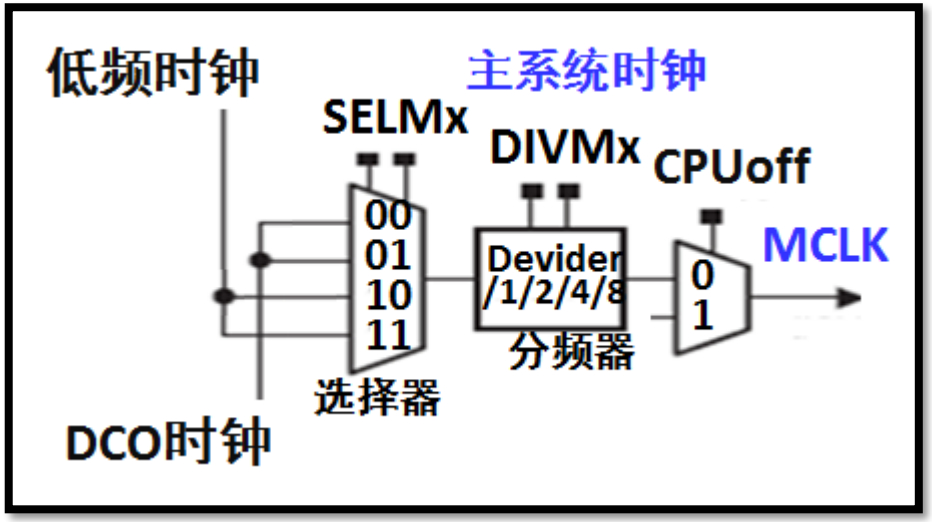
MCLK分频控制

00： /1

01： /2

10： /4

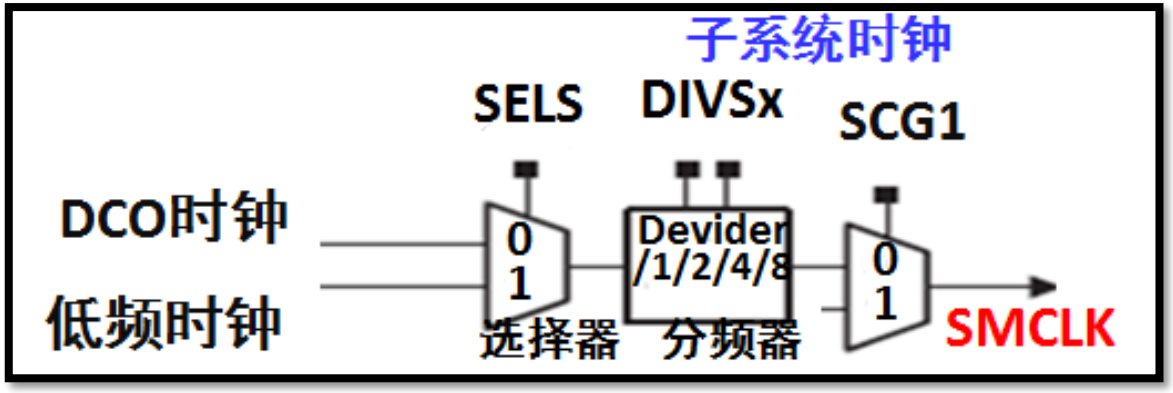
11： /8



BCSCTL2 (Basic Clock System Control Register 2)

基本时钟系统控制寄存器2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0	rw-0	rw-0



SMCLK分频控制

00： /1
01： /2
10： /4
11： /8

四、msp430g2553.h中的定义

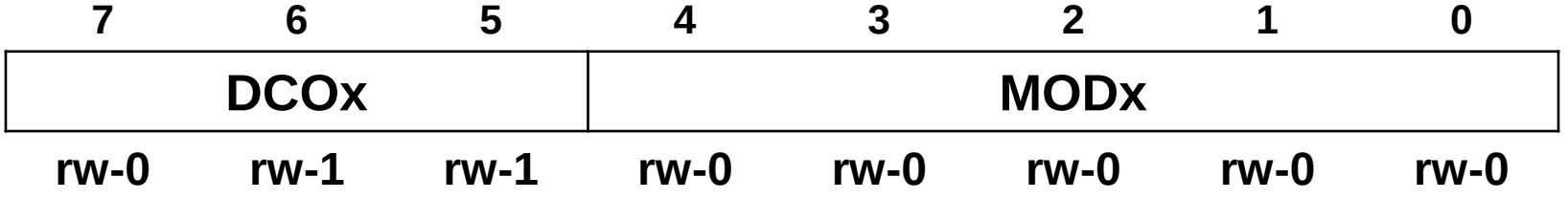
为编程方便，提高可读性，

根据各模块内部I/O寄存器各位的功能，

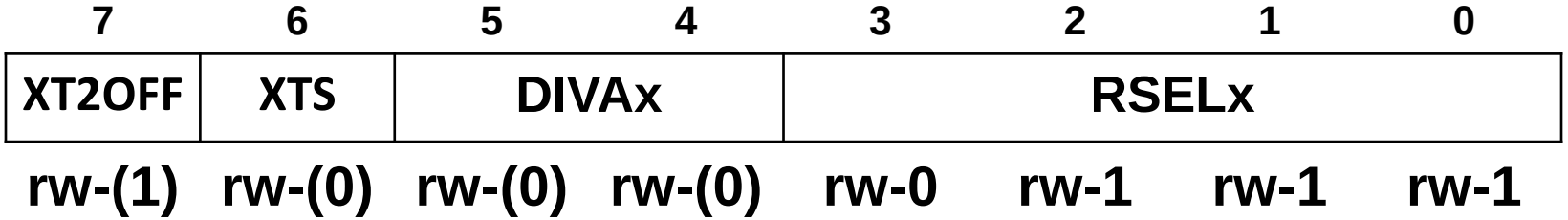
在头文件中设置有两种类型的符号定义

- 位定义
- 位值定义

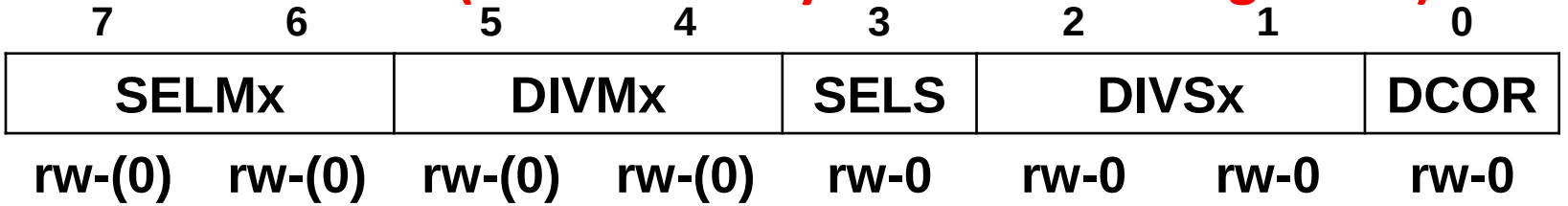
DCOCTL (DCO Control Register)



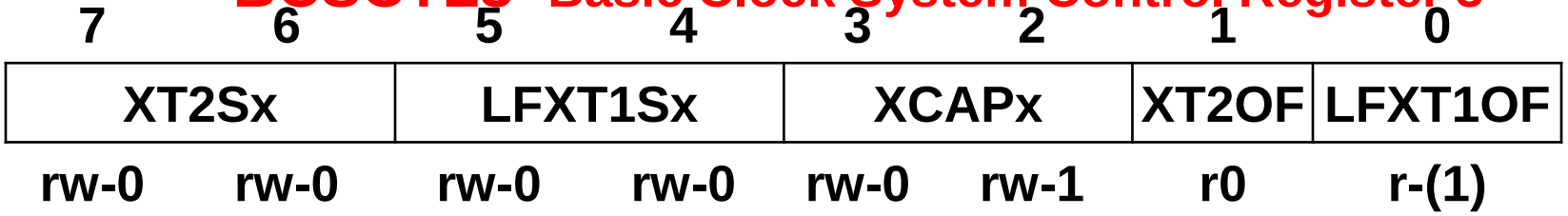
BCSCTL1 (Basic Clock System Control Register 1)



BCSCTL2 (Basic Clock System Control Register 2)

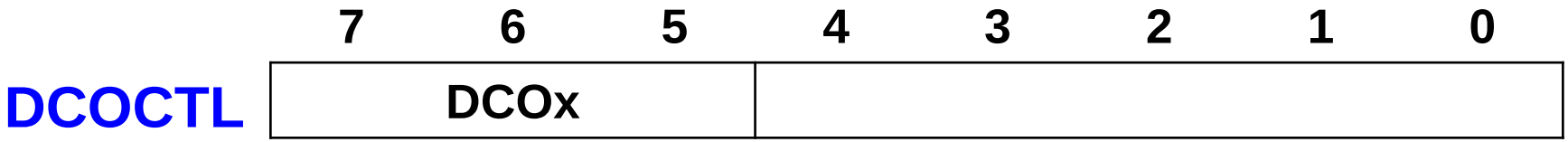


BCSCTL3 Basic Clock System Control Register 3



■ 端口寄存器的位定义

针对寄存器中的每一位用符号做了定义。
有了位定义后，可根据符号名知道该位的功能，
且不再用关心该位在寄存器中的具体位置



位定义

msp430G2553.h

```
#define DCO0          (0x20)     /* DCO Select Bit 0 */
#define DCO1          (0x40)     /* DCO Select Bit 1 */
#define DCO2          (0x80)     /* DCO Select Bit 2 */
```

	7	6	5	4	3	2	1	0
BCSCTL1			DIVAx		RSELx			

位定义

msp430g2553.h

```
#define  RSEL0        (0x01)     /* Range Select Bit 0 */
#define  RSEL1        (0x02)     /* Range Select Bit 1 */
#define  RSEL2        (0x04)     /* Range Select Bit 2 */
#define  RSEL3        (0x08)     /* Range Select Bit 3 */
#define  DIVA0        (0x10)     /* ACLK Divider 0 */
#define  DIVA1        (0x20)     /* ACLK Divider 1 */
```

BCSCTL2

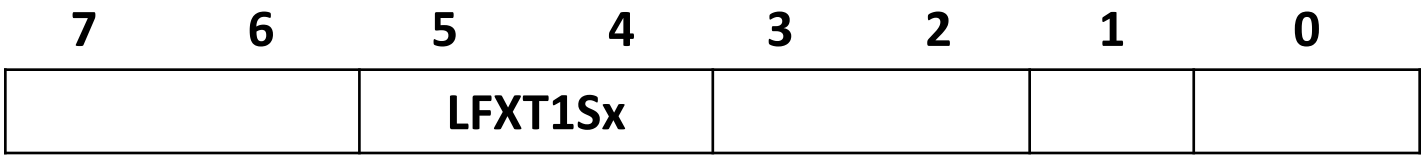
7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		

位定义

msp430g2553.h

```
#define  DIVS0        (0x02)      /* SMCLK Divider 0 */
#define  DIVS1        (0x04)      /* SMCLK Divider 1 */
#define  SELS         (0x08)  /* SMCLK Source Select 0:DCOCLK / 1:XT2CLK/LFXTCLK */
#define  DIVM0        (0x10)      /* MCLK Divider 0 */
#define  DIVM1        (0x20)      /* MCLK Divider 1 */
#define  SELM0        (0x40)      /* MCLK Source Select 0 */
#define  SELM1        (0x80)      /* MCLK Source Select 1 */
```

BCSCTL3



位定义

msp430g2553.h

```
#define  LFXT1S0   (0x10)        /* Mode 0 for LFXT1 (XTS = 0) */
#define  LFXT1S1   (0x20)        /* Mode 1 for LFXT1 (XTS = 0) */
```

■ 端口寄存器的位域值定义

有些功能需要2 位或以上位来设置，

如 BCSCTL2 中的SELMx，DIVMx, DIVSx。

用带下划线的宏定义完成2位或以上位的值域值定义，

将该值定义在寄存器相应的位置上。

BCSCTL2

7	6	5	4	3	2	1	0
SELMx		DIVMx		SELS	DIVSx		
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-0	rw-0	rw-0	rw-0

位域值定义

```
#define  DIVM_0      (0x00)  //MCLK Divider 0: /1 ,  0000 0000B
#define  DIVM_1      (0x10)  // MCLK Divider 1: /2 ,  0001 0000B
#define  DIVM_2      (0x20)  //MCLK Divider 2: /4 ,  0010 0000B
#define  DIVM_3      (0x30)  // MCLK Divider 3: /8 ,  0011 0000B
```

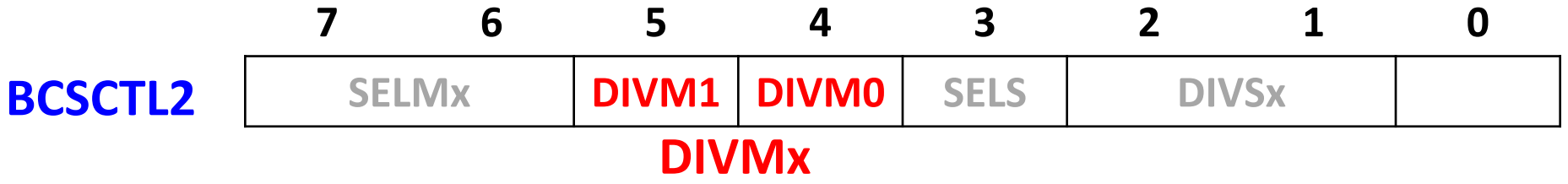
msp430g2553.h 位域值定义

```
#define   DIVA_0        (0x00)  /* ACLK Divider 0: /1 */
#define   DIVA_1        (0x10)  /* ACLK Divider 1: /2 */
#define   DIVA_2        (0x20)  /* ACLK Divider 2: /4 */
#define   DIVA_3        (0x30)  /* ACLK Divider 3: /8 */

#define   DIVS_0        (0x00)  /* SMCLK Divider 0: /1 */
#define   DIVS_1        (0x02)  /* SMCLK Divider 1: /2 */
#define   DIVS_2        (0x04)  /* SMCLK Divider 2: /4 */
#define   DIVS_3        (0x06)  /* SMCLK Divider 3: /8 */

#define   DIVM_0        (0x00)  /* MCLK Divider 0: /1 */
#define   DIVM_1        (0x10)  /* MCLK Divider 1: /2 */
#define   DIVM_2        (0x20)  /* MCLK Divider 2: /4 */
#define   DIVM_3        (0x30)  /* MCLK Divider 3: /8 */

#define   SELM_0        (0x00)  /* MCLK Source Select 0: DCOCLK */
#define   SELM_1        (0x40)  /* MCLK Source Select 1: DCOCLK */
#define   SELM_2        (0x80)  /* MCLK Source Select 2: VLOCLK/LFXTCLK */
#define   SELM_3        (0xC0)  /* MCLK Source Select 3: VLOCLK/LFXTCLK */

#define   LFXT1S_0      (0x00)  /* Mode 0 for LFXT1 : Normal operation */
#define   LFXT1S_1      (0x10)  /* Mode 1 for LFXT1 : Reserved */
#define   LFXT1S_2      (0x20)  /* Mode 2 for LFXT1 : VLO */
#define   LFXT1S_3      (0x30)  /* Mode 3 for LFXT1 : Digital input signal */
```

例 设置主系统时钟 MCLK的8 分频

比较下面四种方法：

```
BCSCTL2 |= 0x30;                    //需放到寄存器中才能知晓含义
BCSCTL2 |= BIT5+BIT4;               //同上
BCSCTL2 |= DIVM1+DIVM0;             //位定义方式，可读性好
BCSCTL2 |= DIVM_3;                  //位值方式，更清晰，可读性强
```



DIVMx

MCLK分频控制

00： /1

01： /2

10： /4

11： /8

注意：

在使用按位或操作符“|=”配置寄存器时，
要注意宏定义之间的“叠加”效应，
重新用宏定义配置寄存器前，一定要先清零。
对上电复位后寄存器默认值不是0，要特别注意。

例 先设定MCLK 分频为2，delay 一段时间后改为4 分频。

错误代码：

```
BCSCTL2 |= DIVM_1;    //因上电复位值为0，|= DIVM_1之后是2 分频
dealy( );       **01****
BCSCTL2 |= DIVM_2;          //在上条语句基础上再|= DIVM_2，
                **10****    //此时 DIVM的两位值为11，即8分频了
```

正确代码：

```
BCSCTL2 |= DIVM_1;          // 设置2 分频
dealy( );       **01****
BCSCTL2 &=~(DIVM0+DIVM1)   //将DIVM的两个控制位设置为0
                **00****
BCSCTL2 |= DIVM_2;              //设置4分频
                **10****
```

对端口寄存器操作小结

- 掌握通过查阅头文件，了解各端口寄存器的位、位值宏定义
- 尽量使用头文件中的定义去设置端口寄存器，提高可读性
- 深刻理解“|=、&=”等赋值结果，

  必要时应先清0 后赋值，避免误操作。
- 用变量设置端口寄存器时，

  应根据变量的值对应在寄存器位置，对变量移位后再赋值。

五、 时钟模块设置举例

例1. 分析上电复位时钟模块的设置

例2. 编程选择MCLK=外部晶振32768Hz/2

例3. 使用出厂的校验值，

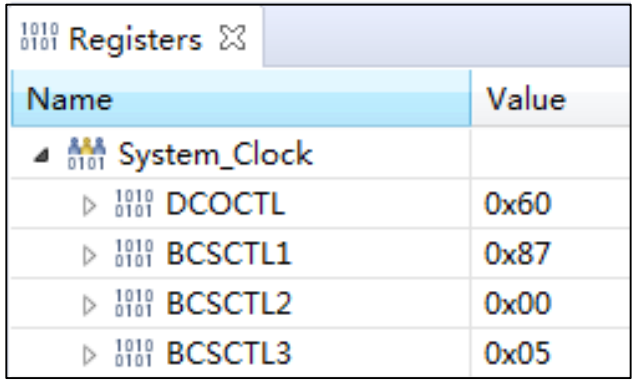
编程选择MCLK=DCO=8MHz

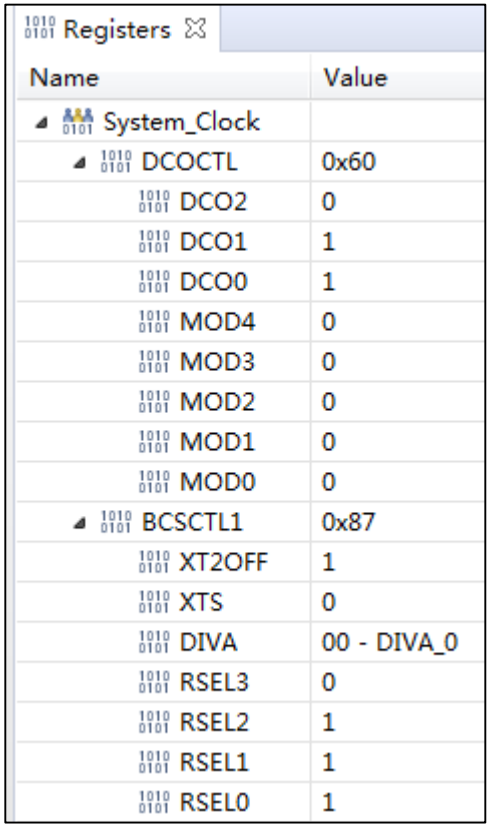
例1. 分析上电复位时钟模块的设置

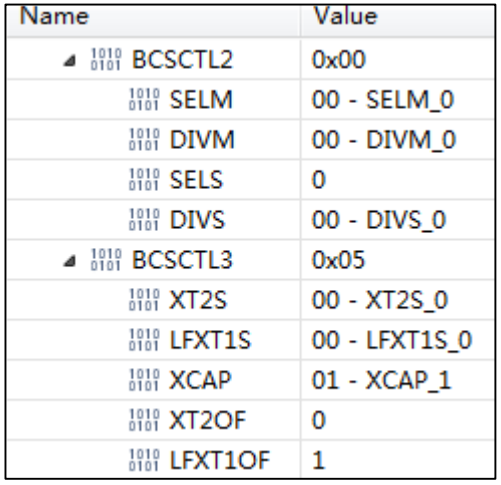
复位后，基本时钟模块相关寄存器初始值为：

端口寄存器	复位值
DCOCTL	0x60
BCSCTL1	0x87
BCSCTL2	0x00
BCSCTL3	0x05
P1SEL	0x00
P2SEL2	0xC0
IE1	0x00
IFG1	0x00

从MSP430x2xx user ‘s guider查到







从CCS/debug上查看

在CCS下的debug上查看

Registers

Name	Value
P2SEL	0xC0
P7	1
P6	1
P5	0
P4	0
P3	0
P2	0
P1	0
P0	0
P2SEL2	0x00
P7	0
P6	0
P5	0
P4	0
P3	0
P2	0
P1	0
P0	0

上电复位分析：

P2SEL的D6=1;

P2SEL2的D6=0;

P2SEL的D7=1;

P2SEL2的D7=0;

故P2.6\P2.7设为外部晶振引脚

DCOCTL 0x60

DCOx							
0	1	1	0	0	0	0	0
DCOx=3							

BCSCTL1 0x87

		DIVAx		RSELx			
1	0	0	0	0	1	1	1
		ACLK: 1分频		RSELx=7			

BCSCTL2 00h

SELMx		DIVMx		SELS	DIVSx		
0	0	0	0	0	0	0	0
MCLK:DCO		MCLK:1分频		Smclk:DCO	SMCLK:1分频		

BCSCTL3 05h

XT2Sx	LFXT1Sx						
	0	0	0	1	0		1

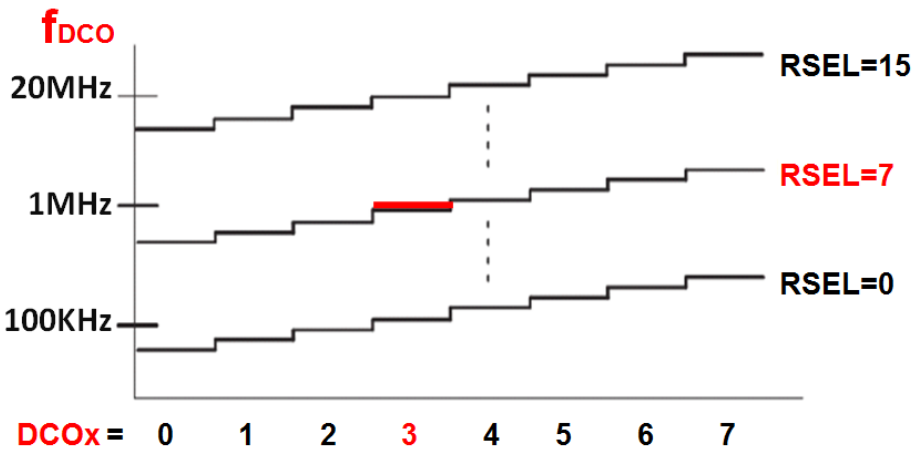
所以复位后，如果程序中没有对时钟模块的处理，则：
MCLK、SMCLK时钟源为DCO，ACLK时钟源为 LFXT1CLK
各时钟均为1分频

上电复位时钟模块的设置

Register

System Clock

DCOCTL	= 0x60	BCSCTL2	= 0x00
MOD0	= 0	DIVS0	= 0
MOD1	= 0	DIVS1	= 0
MOD2	= 0	SELS	= 0
MOD3	= 0	DIVM0	= 0
MOD4	= 0	DIVM1	= 0
DCO0	= 1	SELM0	= 0
DCO1	= 1	SELM1	= 0
DCO2	= 0	BCSCTL3	= 0x05
BCSCTL1	= 0x87	LFXT1OF	= 1
RSEL0	= 1	XT2OF	= 0
RSEL1	= 1	XCAP0	= 1
RSEL2	= 1	XCAP1	= 0
RSEL3	= 0	LFXT1S0	= 0
DIVA0	= 0	LFXT1S1	= 0
DIVA1	= 0	XT2S0	= 0
XTS	= 0	XT2S1	= 0
XT2OFF	= 1		



典型DCO频率范围及其分档图

MCLK =DCO ≈ 1MHz

SMCLK =DCO ≈ 1MHz

即DCOx=3，RSELx=7，对应档的DCOCLK频率，具体值需实测

ACLK = LFXT1CLK

= 32.768KHz //引脚P2.6 、P2.7接有32768Hz晶振频率情况

= 0 //引脚P2.6 、P2.7未接晶振情况

例2. 利用上电复位初始值

编程选择 MCLK = 外部晶振32768Hz / 2

1. 确认外部晶振连上
2. 清除OFIFG标志
3. 延时50us
4. 检测OFIFG标志，如果标志为1，回到步骤2
5. 设置选择外部晶振为MCLK，并根据需要设置分频

```
while ( (IFG1 &OFIFG) !=0 )                  //如果振荡失败
 {  IFG1 &= ~OFIFG  ;                        //清除振荡失败标志
          for (i=0;i<0xffff; i++);            //延时等待时钟稳定
  };
BCSCTL2 |= SELM1+DIVM0;    //MCLK选低频时钟，2 分频
```

注意： 为防止外部晶振故障或未接，可将while {} 循环改为有限次检测，MCLK继续使用DCO，或给出信号提醒晶振失效信号，如蜂鸣响或LED闪。

例3. 使用出厂的预校正频率设置值，

编程使MCLK=DCO=8MHz

- DCO 中的RSELx 的粗调和DCOx 的细调都是非线性的。

  如何保证DCO 输出频率精度呢？

- 每一块MSP430G2553单片机的校验参数都不一样

  单片机在出厂时都提供了1/8/12/16MHz

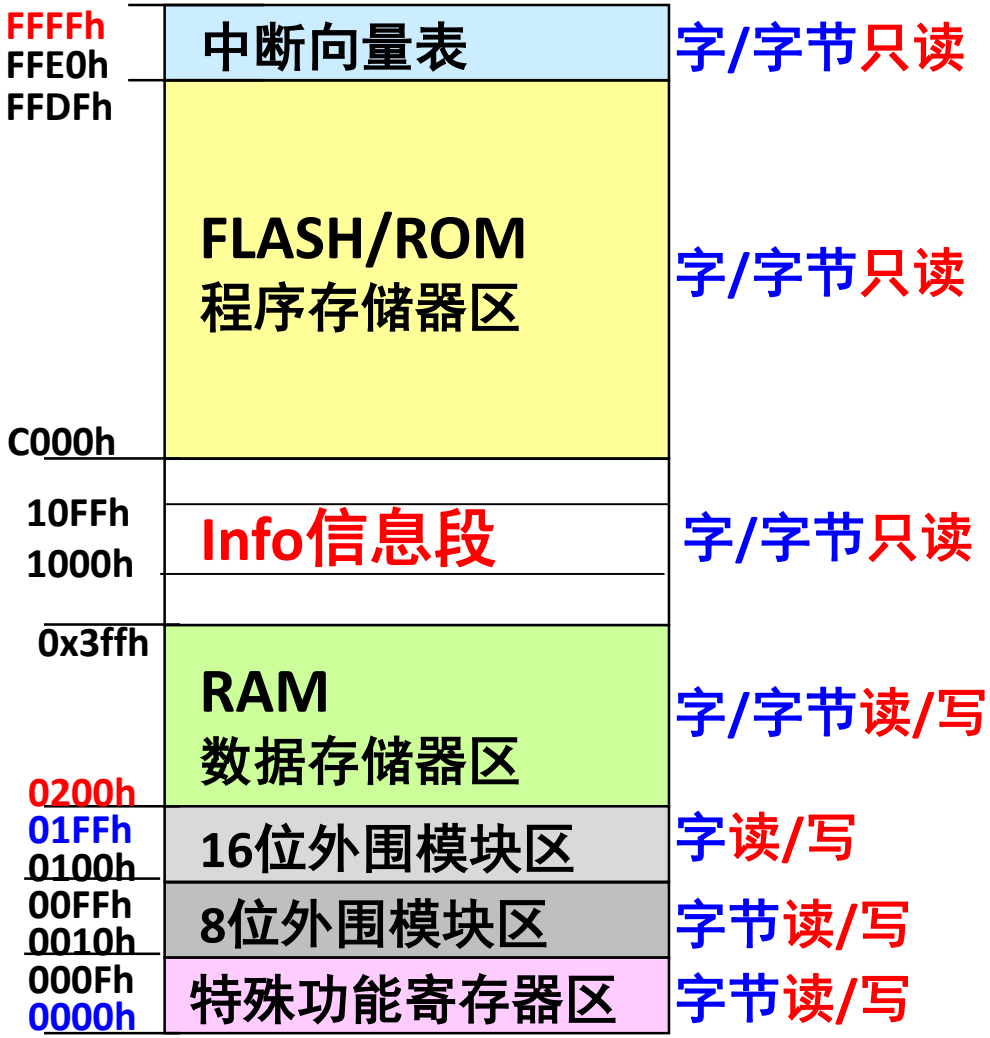
  这4 个频率的对应的DCOx、RSELx、MODx的预校验值，

  并将这些参数存放在单片机内的FLASH 的信息段(info段)中。

MSP430G2xx3的存储系统里的信息存储器

MSP430		G2153 G2113	G2253 G2213	G2353 G2313	G2453 G2413	G2553 G2513
ROM	容量	1KB	2KB	4KB	8KB	16KB
	中断向量	64B, 0xFFFF~0xFFC0				
	代码存储器	0xFFBF~ x0FC00	0xFFBF~ 0xF800	FFBF~ F000	0xFFBF~ 0xE000	0xFFBF~ 0xC000
	信息存储器	256B, 0x10FF~0x1000				
RAM	容量	256B	256B	256B	512B	512B
	范围	0x02FF~0x0200			0x03FF~0200	
16位外围模块		01FF~0100H				
8位外围模块		00FF~0010H				
特殊功能寄存		000F~0000H				

MSP430G2553的存储器结构



- 每组参数占用两个存储单元，从0x10F8单元开始，从低到高依次存放16MHz、 12MHz、 8MHz、 1MHz DCO振荡频率的DCOCTL和BCSCTL1校验值

ROM地址	头文件中变量定义	存放校验值
0x10FF	CALBC1_1MHZ	1MHz的BCSCTL1
0x10FE	CALDCO_1MHZ	1MHzDCOCTL
0x10FD	CALBC1_8MHZ	8MHz的BCSCTL1
0x10FC	CALDCO_8MHZ	8MHzDCOCTL
0x10FB	CALBC1_12MHZ	12MHz的BCSCTL1
0x10FA	CALDCO_12MHZ	12MHzDCOCTL
0x10F9	CALBC1_16MHZ	16MHz的BCSCTL1
0x10F8	CALDCO_16MHZ	16MHzDCOCTL

- 用户可用这些预校验值设置DCOCTL和BCSRCTL1，获得1/8/12/16MHz的DCO时钟信号

msp430g2553.h的定义

```
SFR_8BIT(CALDCO_16MHZ);   /* DCOCTL  Calibration Data for 16MHz */
SFR_8BIT(CALBC1_16MHZ);   /* BCSCTL1 Calibration Data for 16MHz */
SFR_8BIT(CALDCO_12MHZ);   /* DCOCTL  Calibration Data for 12MHz */
SFR_8BIT(CALBC1_12MHZ);   /* BCSCTL1 Calibration Data for 12MHz */
SFR_8BIT(CALDCO_8MHZ);    /* DCOCTL  Calibration Data for 8MHz */
SFR_8BIT(CALBC1_8MHZ);    /* BCSCTL1 Calibration Data for 8MHz */
SFR_8BIT(CALDCO_1MHZ);    /* DCOCTL  Calibration Data for 1MHz */
SFR_8BIT(CALBC1_1MHZ);    /* BCSCTL1 Calibration Data for 1MHz */
```

msp430g2553.cmd的设置

```
CALDCO_16MHZ = 0x10F8;
CALBC1_16MHZ = 0x10F9;
CALDCO_12MHZ = 0x10FA;
CALBC1_12MHZ = 0x10FB;
CALDCO_8MHZ  = 0x10FC;
CALBC1_8MHZ  = 0x10FD;
CALDCO_1MHZ  = 0x10FE;
CALBC1_1MHZ  = 0x10FF;
```

编程利用校验值，设置DCO的振荡频率为8MHz

```
BCSCTL1 = CALBC1_8MHZ;
DCOCTL  = CALDCO_8MHZ;
```

为防止参数已被擦除，可加上判断语句排除，防止误操作用的。

```
if ( CALBC1_8MHZ != 0xff )   // Flash 被擦除后每位都是1
{          BCSCTL1=CALBC1_8MHZ;
           DCOCTL=CALDCO_8MHZ;
};
```

第3节　定时器概述及设计实例

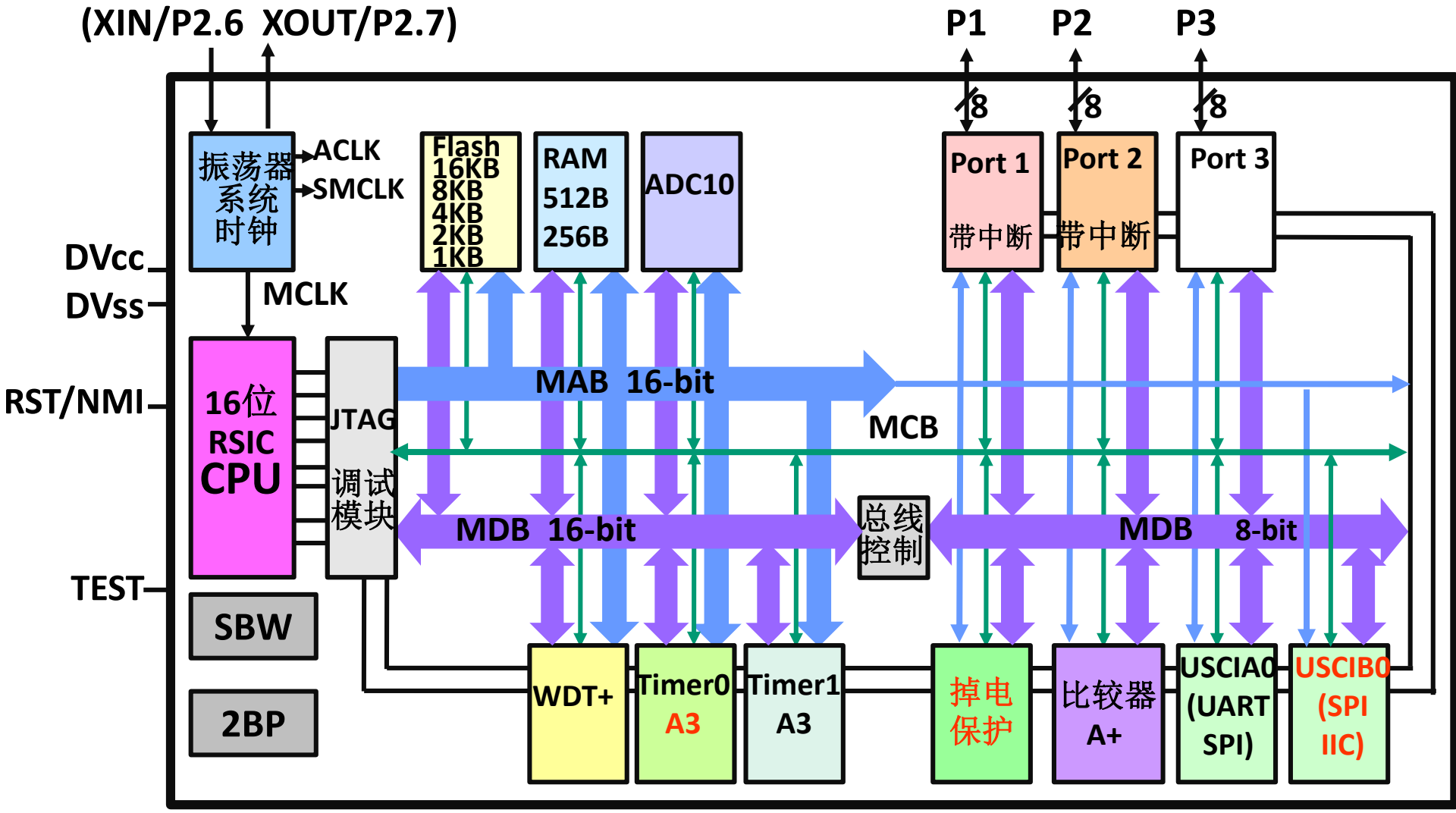
一、定时器的基本作用

二、MSP430G2553的定时器A

三、 TA的比较功能及PWM实现实例

MSP430G2553内部的定时器

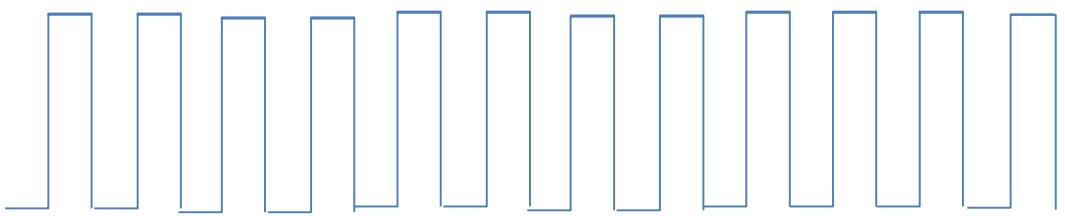
内部包含定时器TA0、TA1、看门狗定时器WDT

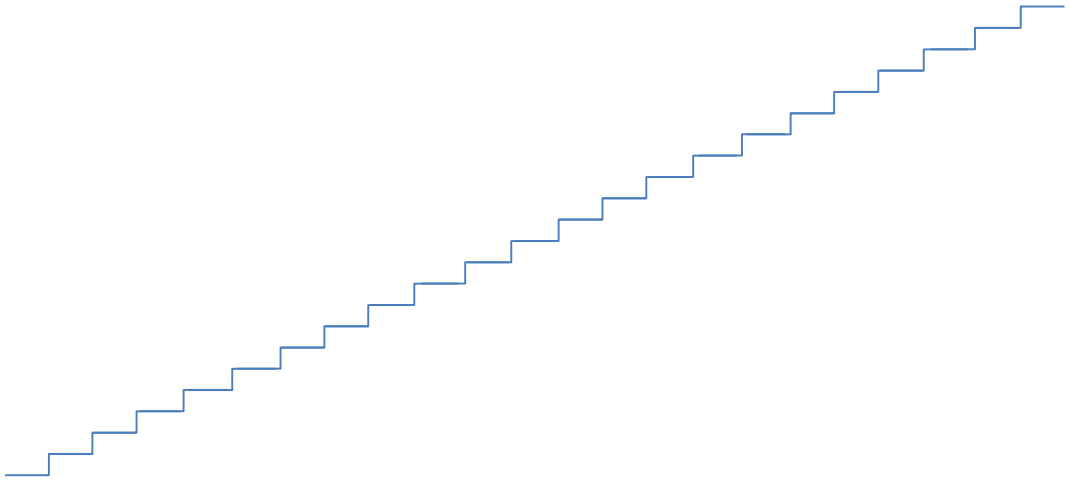


一、定时器的作用

时钟信号CLK是一个周期固定的方波信号(脉冲信号)

利用时钟信号可以做什么?

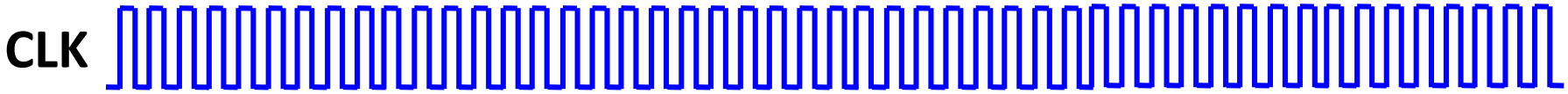


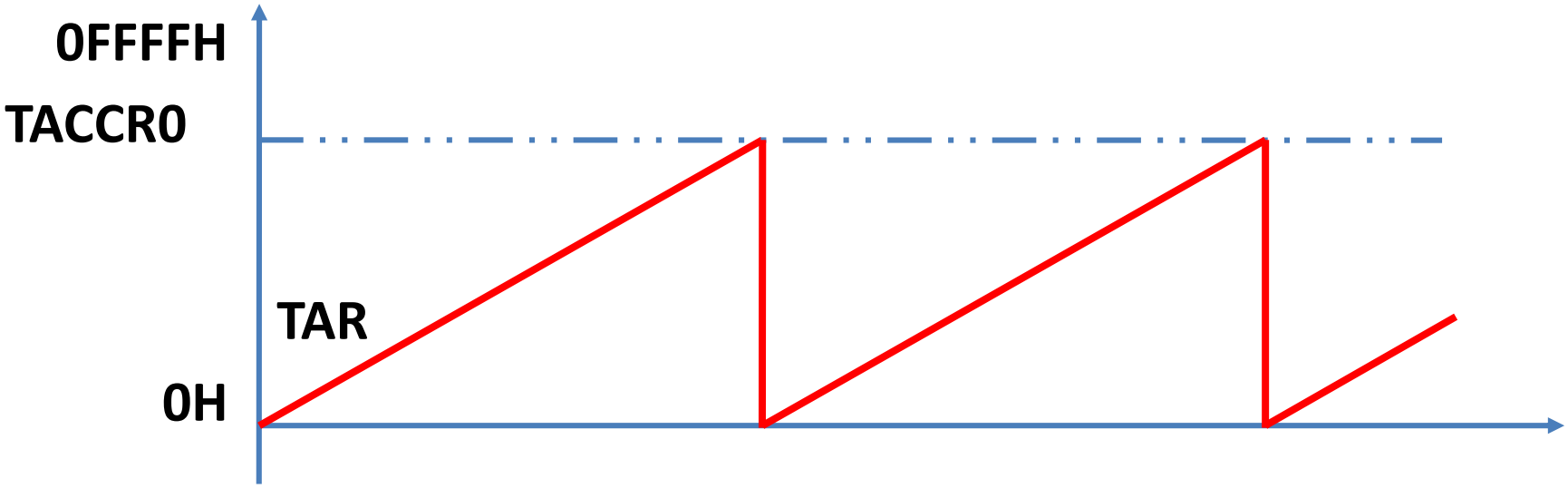


1. 定时
2. PWM脉宽调制(比较输出)
3. 事件发生时刻的捕捉(输入捕捉)
4. 对外部事件计数(输入捕捉)
5. 测量脉宽(输入捕捉)
6. 速度测量、周期/频率测量(输入捕捉)

计数器: TAR

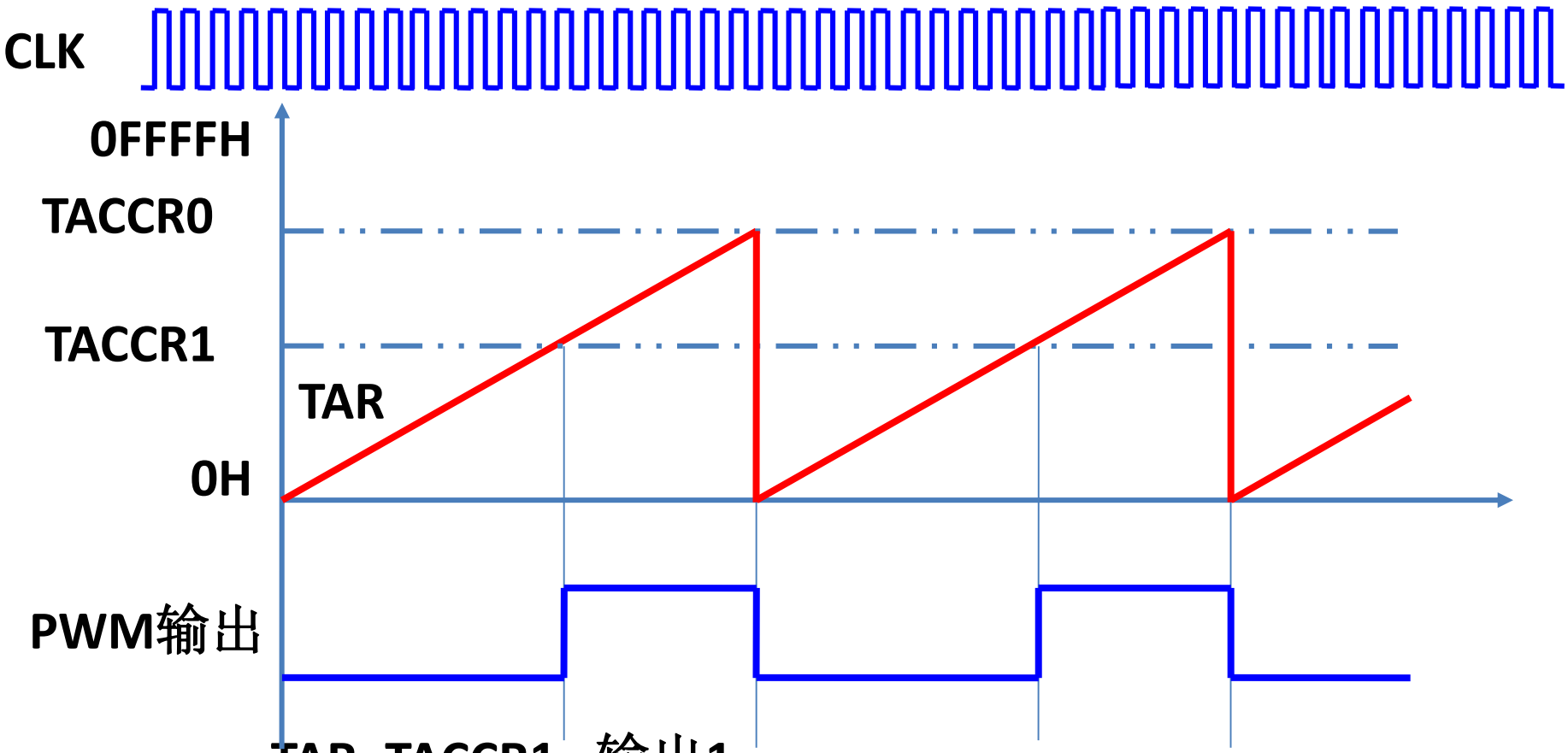
■ 1）定时





TAR计数脉冲个数，
到设定的TACCR0值，发中断申请
定时t= TACCR0*T，　T为一个脉冲的周期
每来一次中断申请，意味着定时到了
调整TACCR0可调整定时时间长短

■ 2） Pulse Width Modulation脉宽调制输出(比较输出)



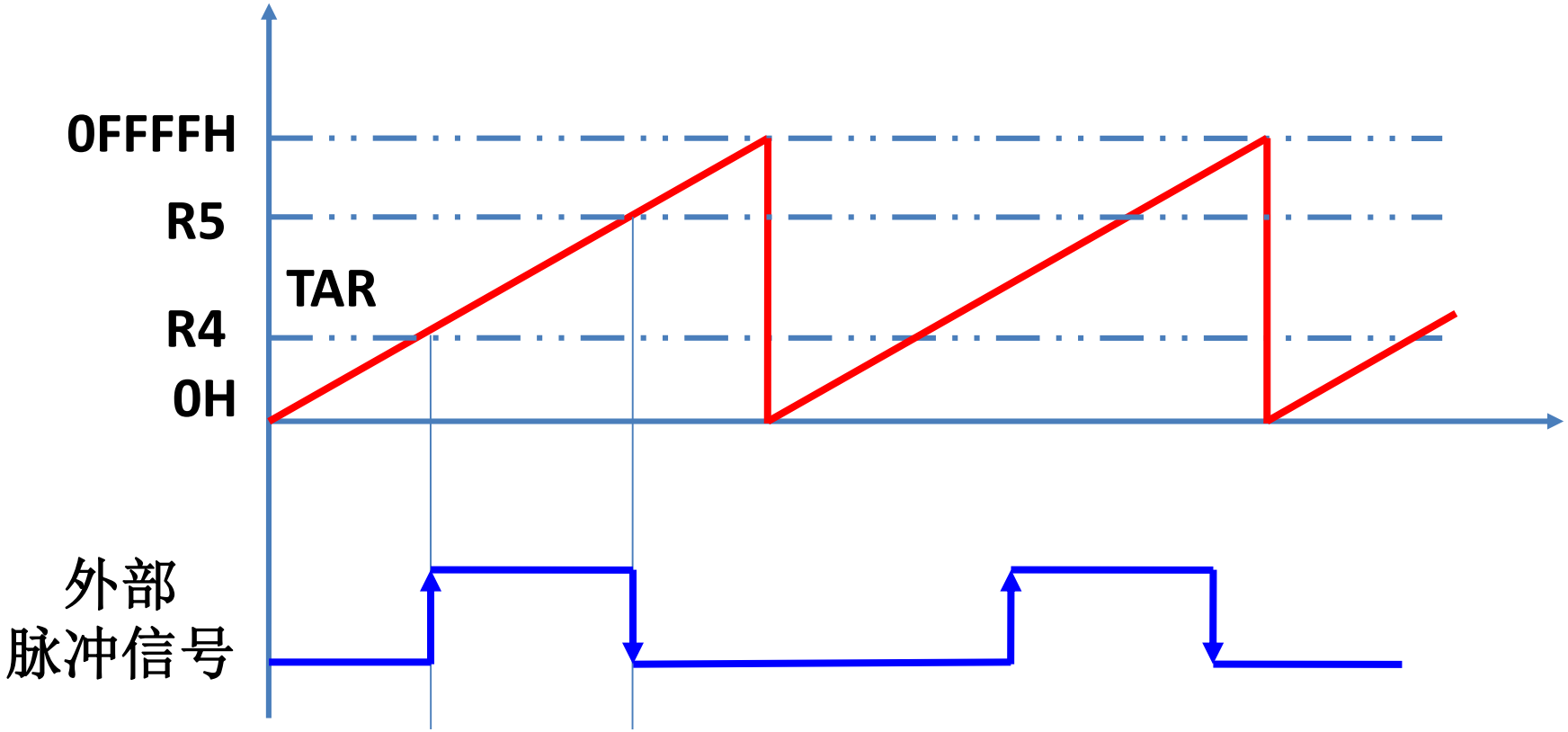
TAR=TACCR1，输出1

TAR=TACCR0，输出0，并置TAR=0

改变TACCR0，改变脉冲周期

改变TACCR1，改变占空比

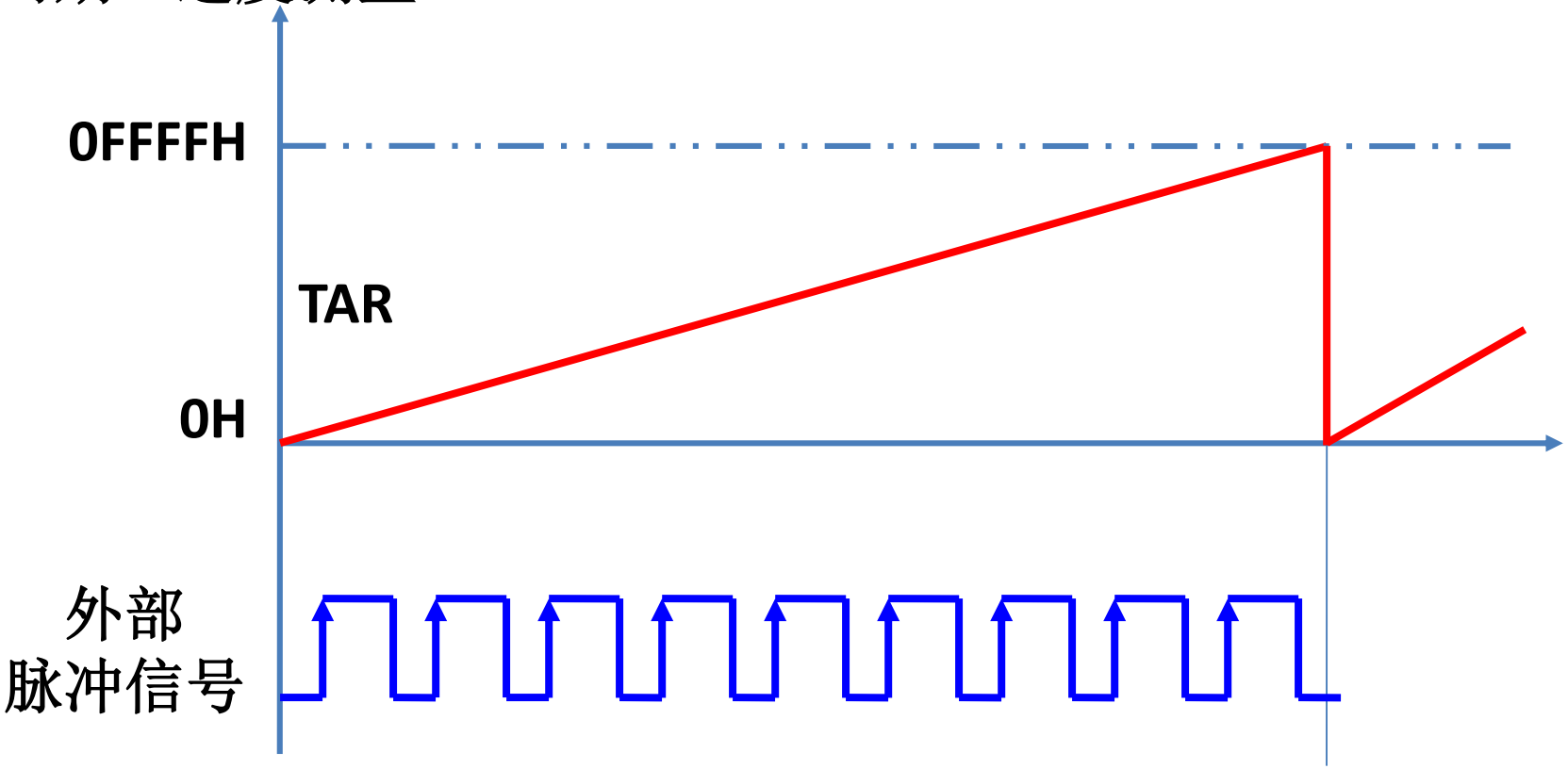
■ 3）测量脉宽（输入捕捉）



事件产生时（如上升沿，下降沿），
将上升沿产生时刻的TAR计数值记录下来，如R4
再将下降沿产生时刻的TAR计数值记录下来，如R5
两者相减，即可得到脉冲宽度

■ 4） 频率、周期、速度测量功能

■ 利用定时和输入捕捉(或外部中断）结合， 可以进行频率/周期、速度测量

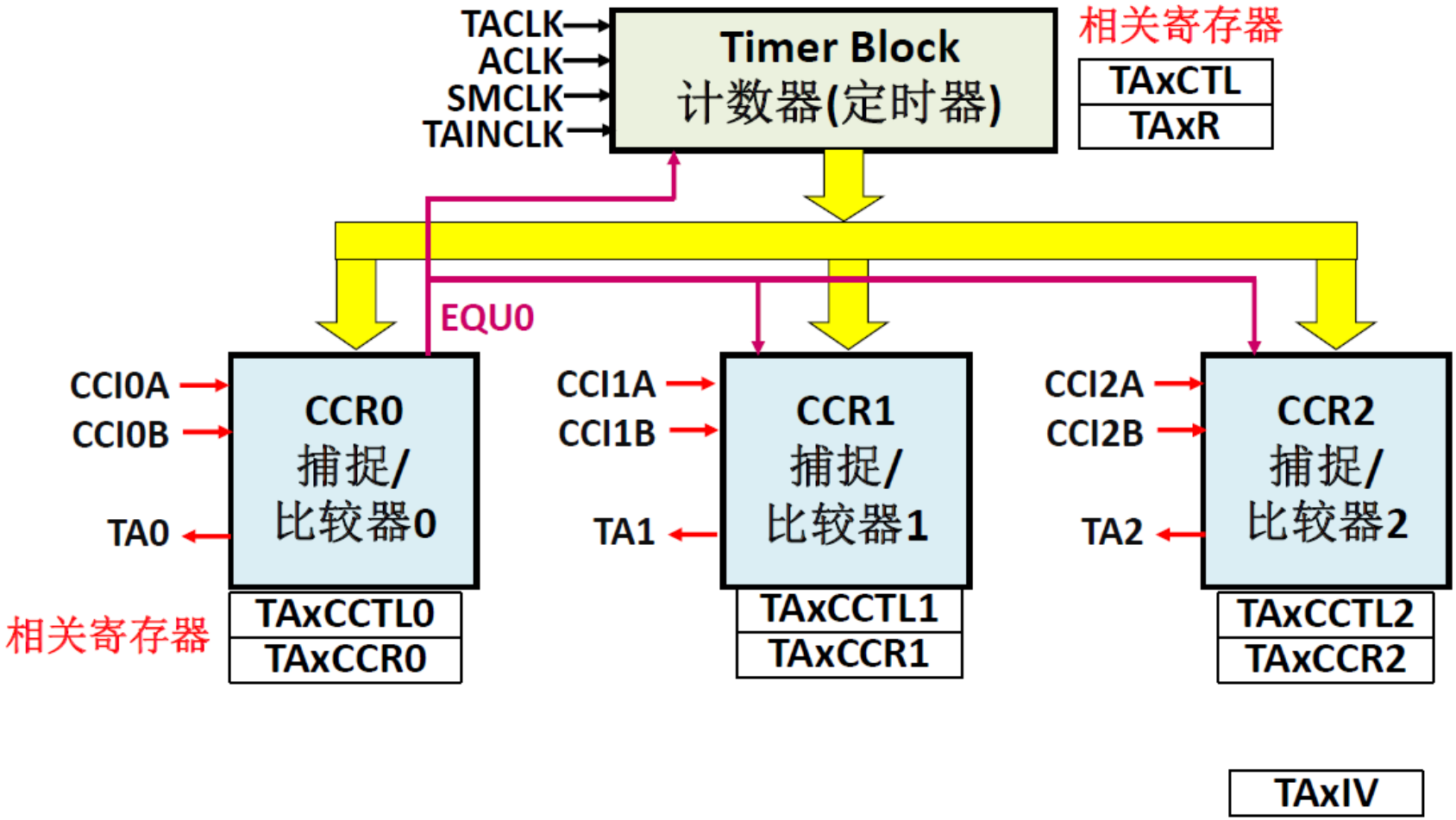


计数一定时间内(如10s)，外部事件发生的次数，即可获取频率

二、MSP430G2553的定时器A

1、TA的特点及结构

由一个计数定时器单元和3个捕捉/比较器单元构成



定时器的操作　通过定时器相关的配置寄存器完成

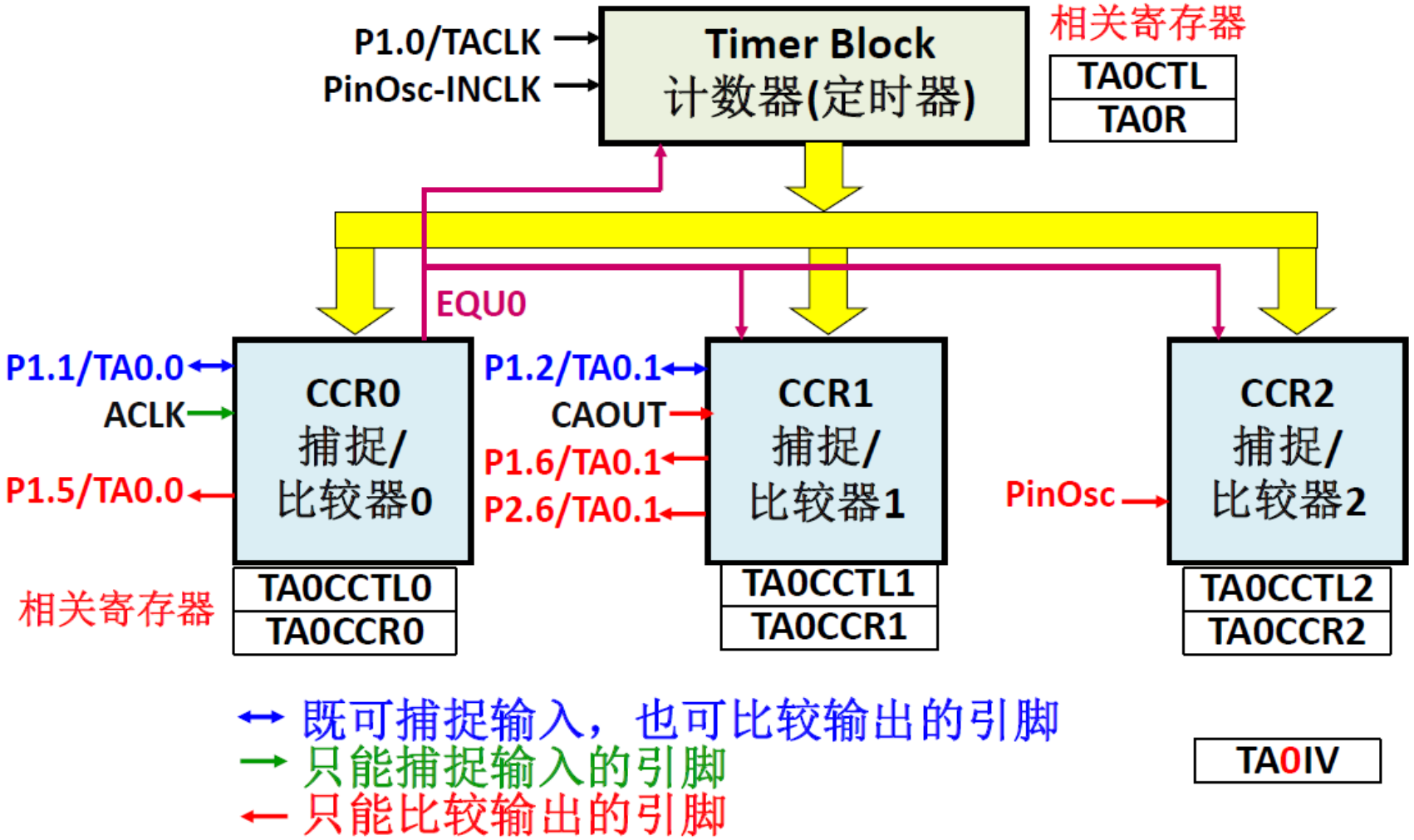
内部包含定时器TA0、TA1

每个定时器的寄存器如下

- 控制器寄存器：TAxCTL
- 计数器: TAxR
- 捕获比较控制寄存器：TAxCCTLx （存在多个，编号0,1,2）
- 捕获比较寄存器：TAxCCRx（存在多个，编号0,1,2）
- 中断向量寄存器TAIV

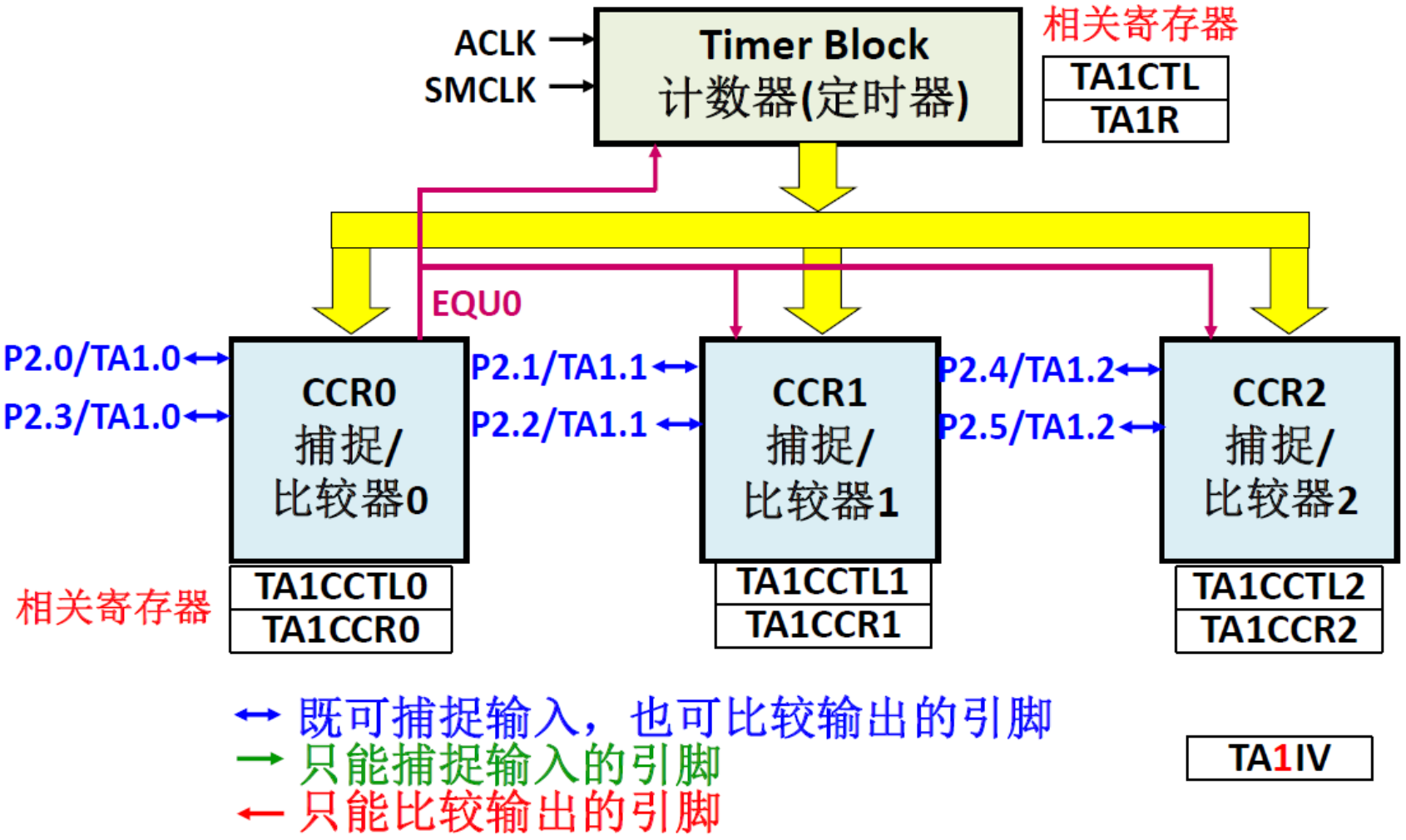
TA0的结构示意图

由一个计数定时器单元和3个捕捉/比较器单元构成



TA1的结构示意图

由一个计数定时器单元和3个捕捉/比较器单元构成



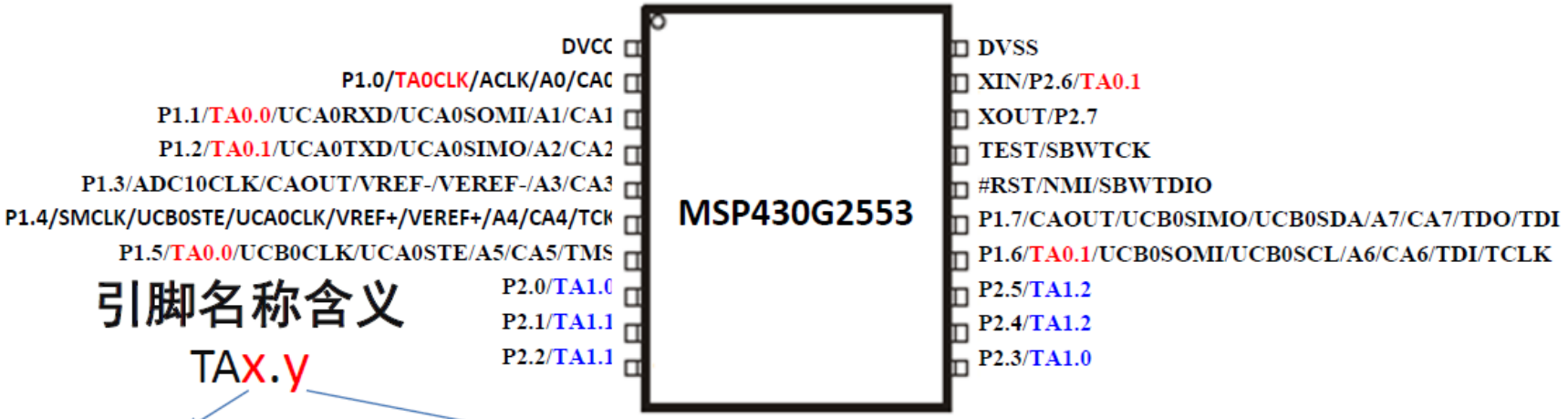
定时器A相关寄存器

寄 存 器	缩写	读写类型
TA控制寄存器	TAxCTL	读/写
TA计数寄存器	TAxR	读/写
TA捕捉/比较控制寄存器0	TAxCCTL0	读/写
TA捕捉/比较寄存器0	TAxCCR0	读/写
TA捕捉/比较控制寄存器1	TAxCCTL1	读/写
TA捕捉/比较寄存器1	TAxCCR1	读/写
TA捕捉/比较控制寄存器2	TAxCCTL2	读/写
TA捕捉/比较寄存器2	TAxCCR2	读/写
TA中断向量寄存器	TAxIV	只读

CC：Capture/Compare

X=0,1 可以是T0_A3，或T1_A3

20引脚双列直插式

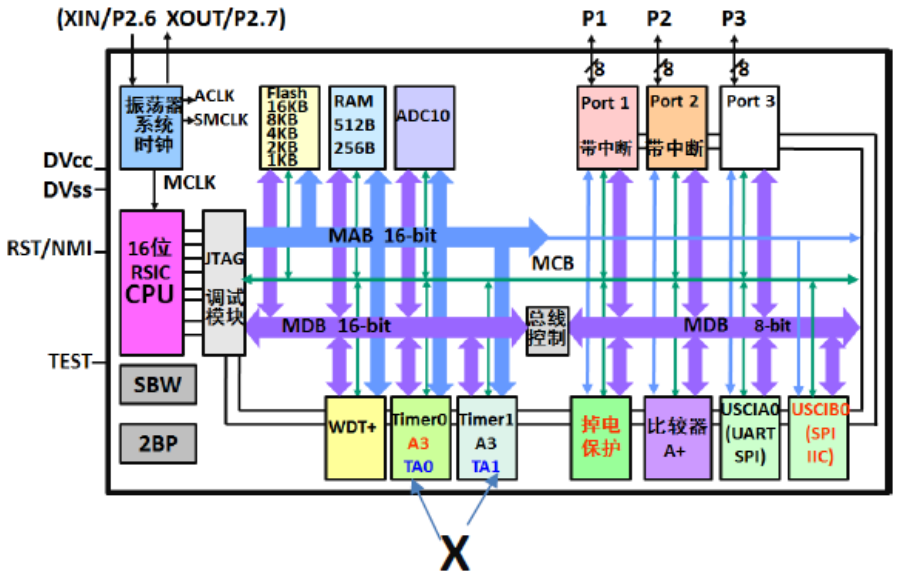


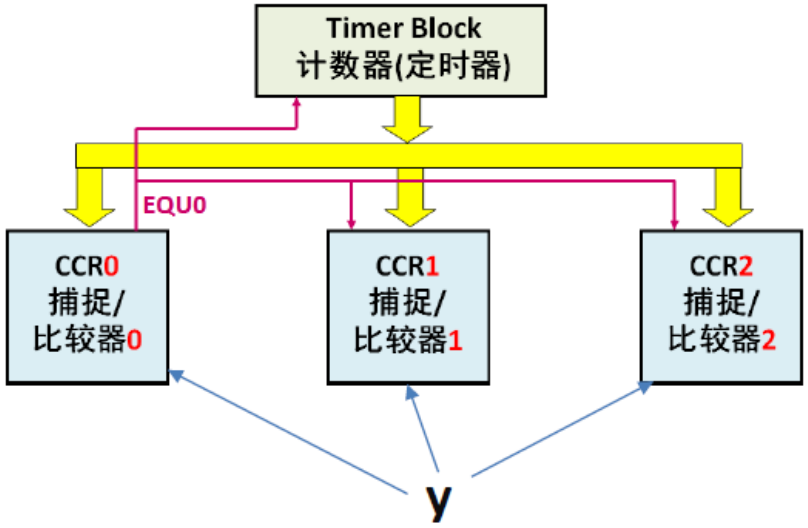
引脚名称含义

TAx.y

x 表示哪个定时器

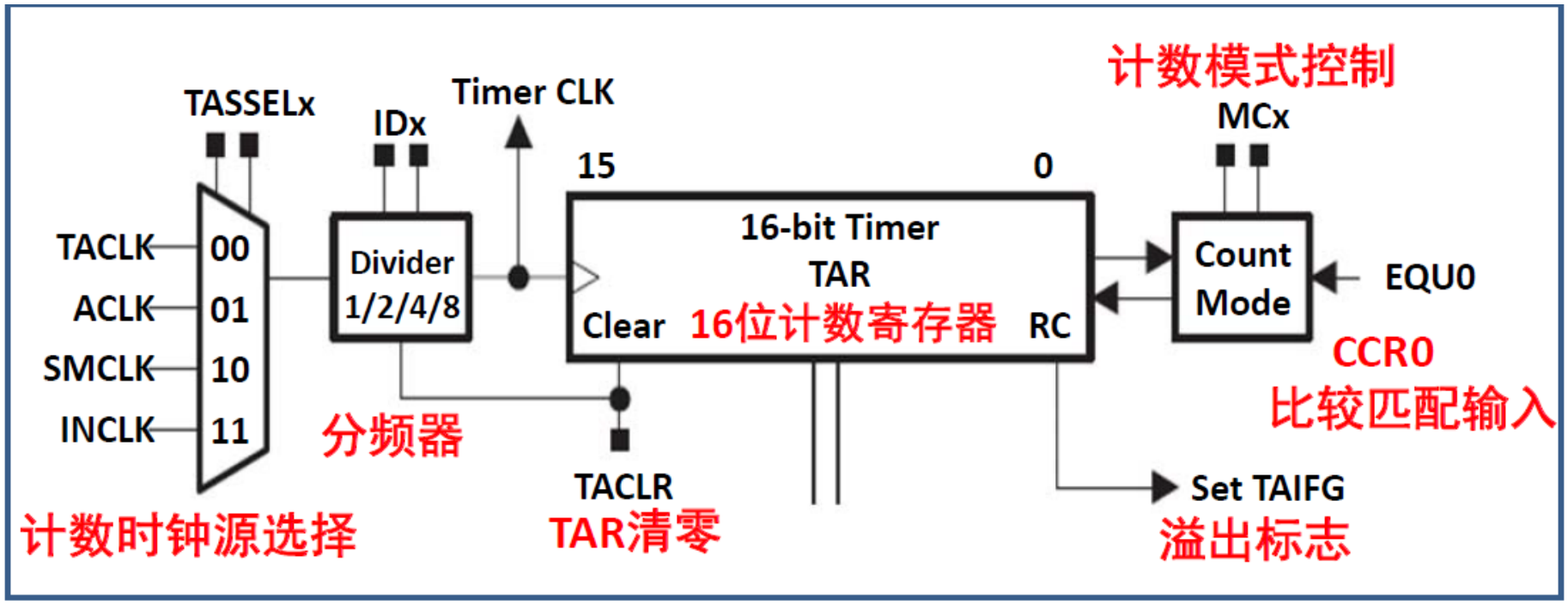
y 表示 定时器中的哪个捕捉/比较器





2、TA的计数单元

TA计数定时器部分结构图



图中控制位(如TASSELx、MCx等)中的 x 表示有2位以上, 按顺序可为 0、1、2、....

相关寄存器:

TAxCTL: TA控制寄存器

TAxR: TA计数寄存器

寄存器名称中的 x可为0或1，分别对应定时器TA0，TA1

TA计数寄存器 TAxR

- TAR是1个16位计数寄存器，其内容可读可写;
- Timer CLK时钟 的上升沿触发1次计数器TAR计数，
- 由计数方式决定TAR自动随时钟个数加 1 或减 1、
  到何值时设置溢出中断标志；
- TAR 可由程序设置初值，可由控制寄存器的TACLR 位清零

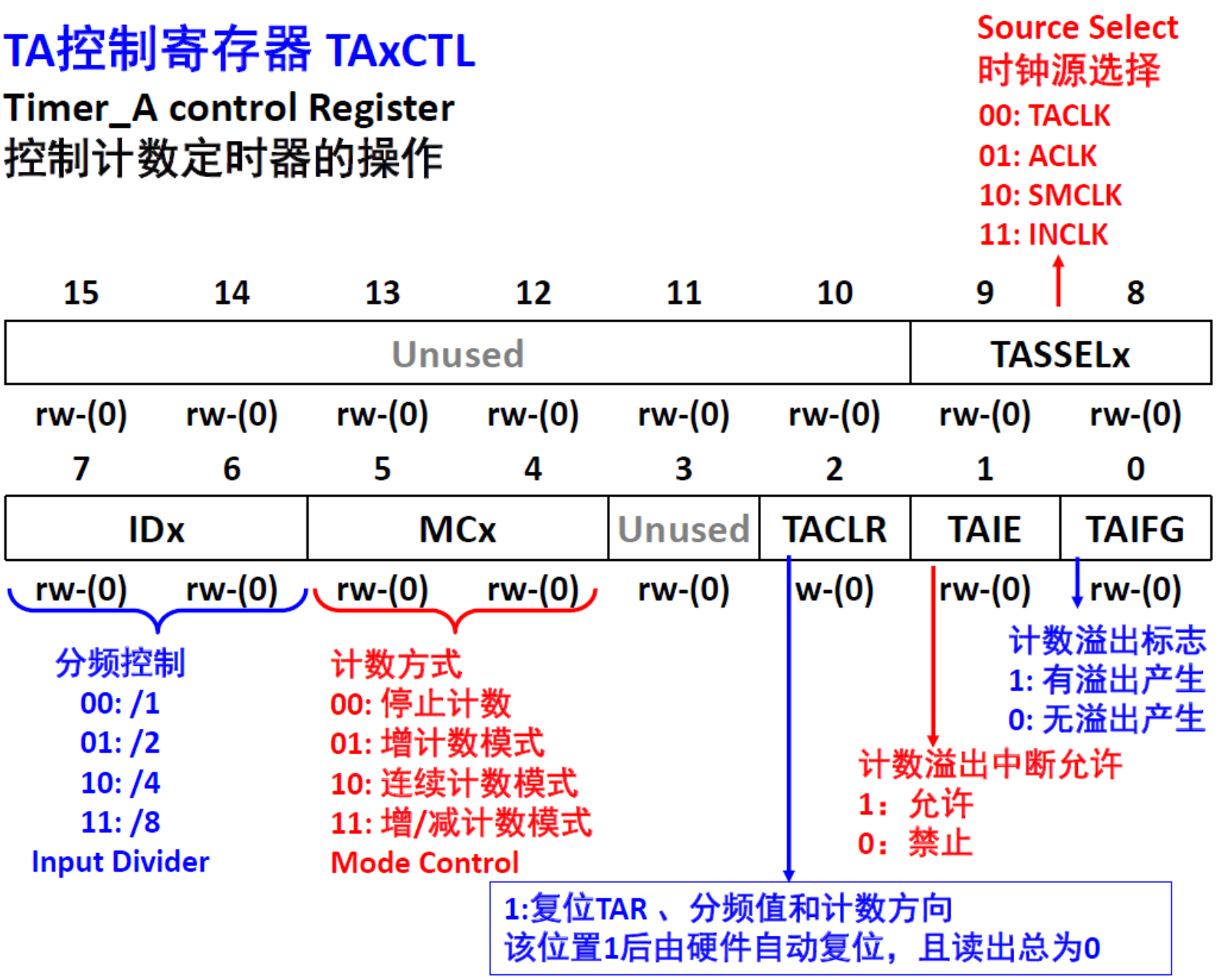
15	14	13	…	…	…	1	0
TAxR							
rw-(0)	rw-(0)	rw-(0)	…	…	rw-(0)	rw-(0)	rw-(0)

注意： TAxR 中x分别对应定时器TA0，TA1，x可为0或1

TA控制寄存器 TAxCTL

Timer_A control Register

控制计数定时器的操作



TA的4种计数方式 Count Mode

MCx

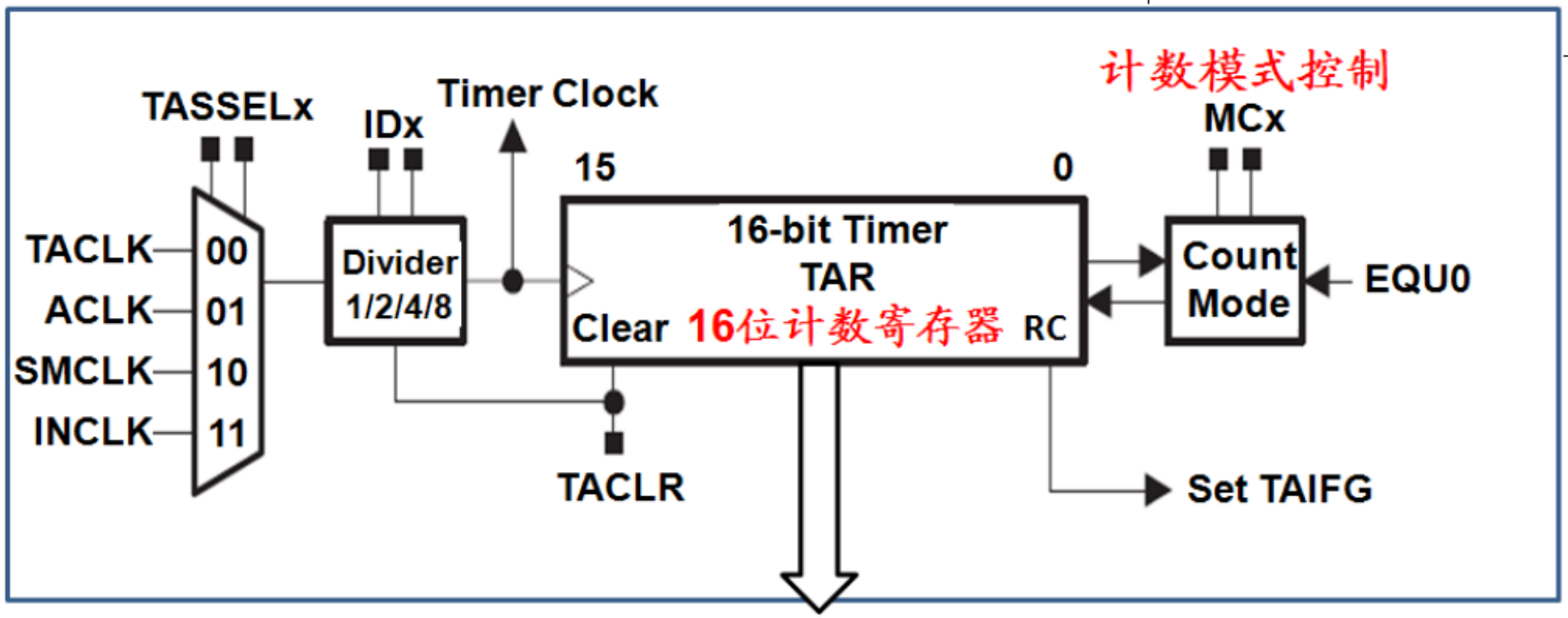
0 0: 停止计数　　stop mode

0 1: 增计数模式　　up mode

1 0: 连续计数模式　　continuous mode

1 1: 增/减计数模式　　up/down mode (不要求)

```
enum {
  TAIFG    = 0x0001,
  TAIE     = 0x0002,
  TACLR    = 0x0004,
  MC0      = 0x0010,
  MC1      = 0x0020,
  ID0      = 0x0040,
  ID1      = 0x0080,
  TASSEL0 = 0x0100,
  TASSEL1 = 0x0200
};
```



1) 停止方式　MCx= 00

Stop mode : the timer is halted

- 定时器暂停计数，并不复位TA，TAxR计数寄存器保持不变，所有寄存器的内容在停止方式结束后都可用。
- 当不使用Timer时，

  将Timer配置为Stop mode，可降低芯片的功耗
- MCU复位后定时器 A 的计数方式为 stop mode

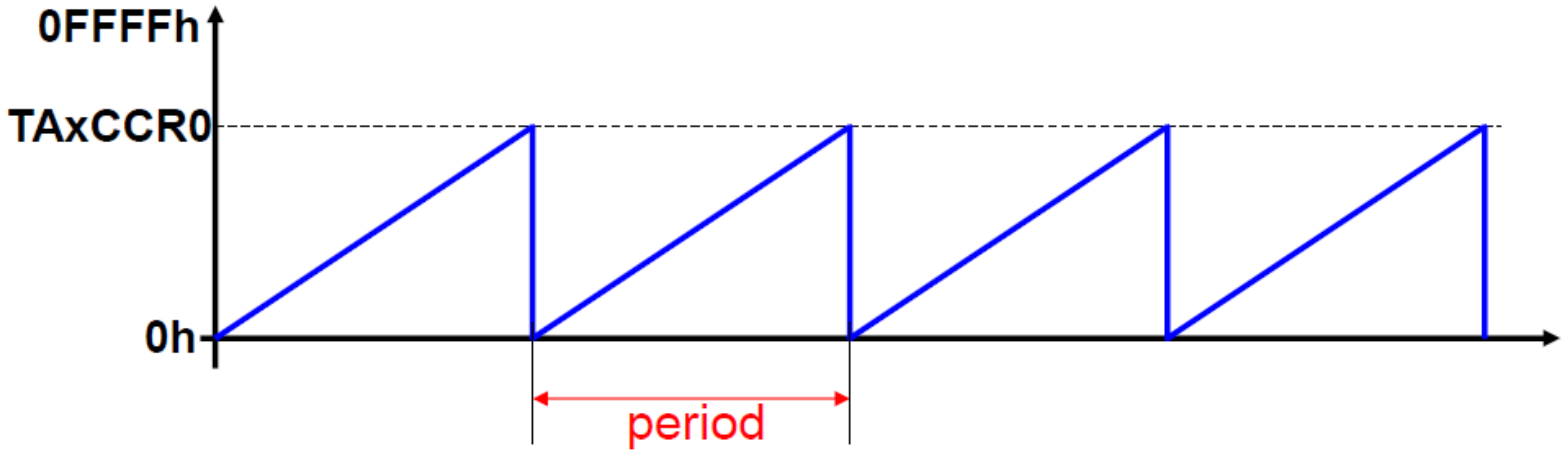
例　编程设置TA0为停止方式

C语言　　TA0CTL &= ~(MC1+MC0);

2) 增计数方式(锯齿波方式) MCx= 01

Up mode 必需要CCR0(比较方式)协助

Timer CLK的上升沿触发 TAR加1

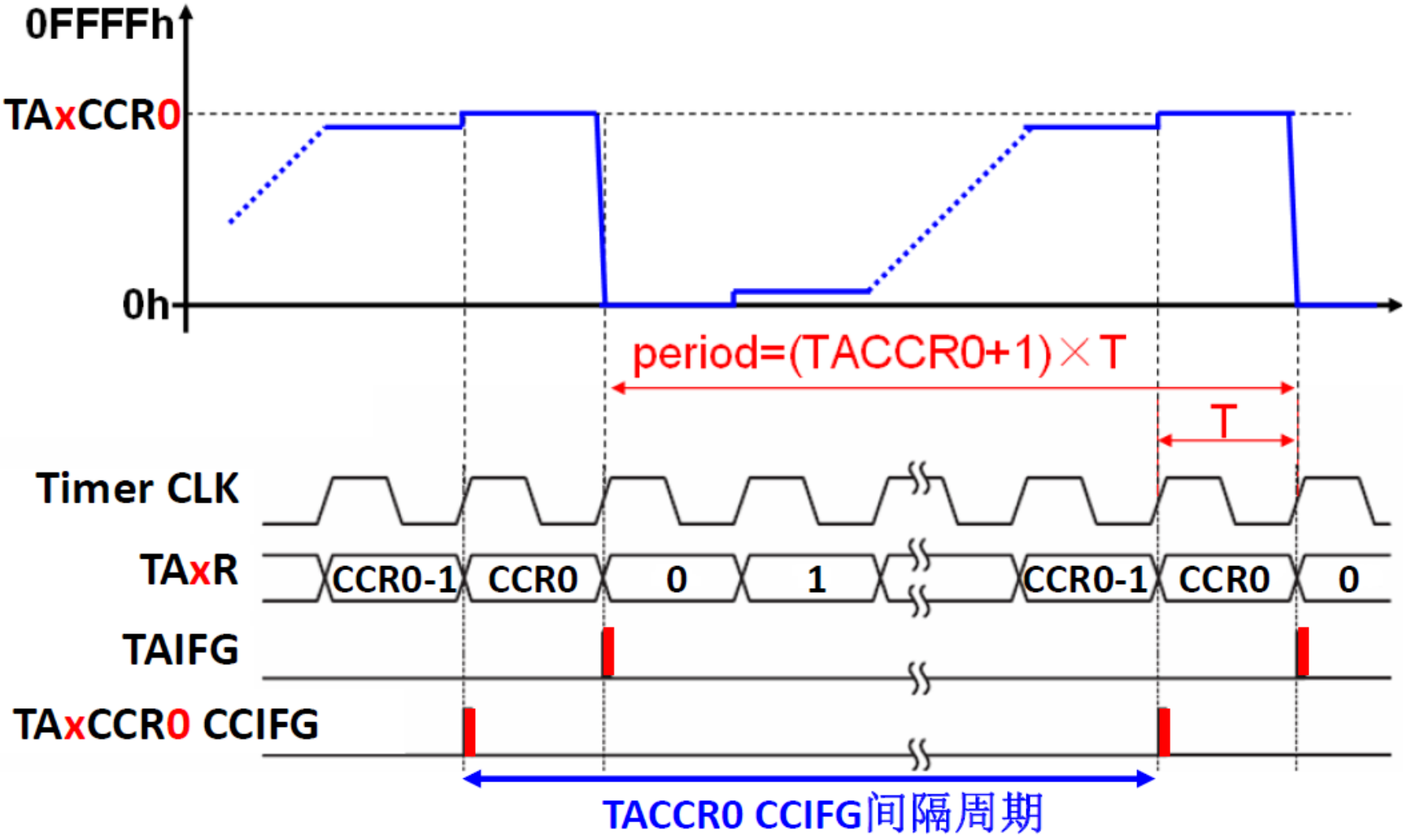


period = (TACCR0 +1)×T

T = Timer Clock 的周期

TA0CTL |= MC_1;

增计数方式下标志位的设置



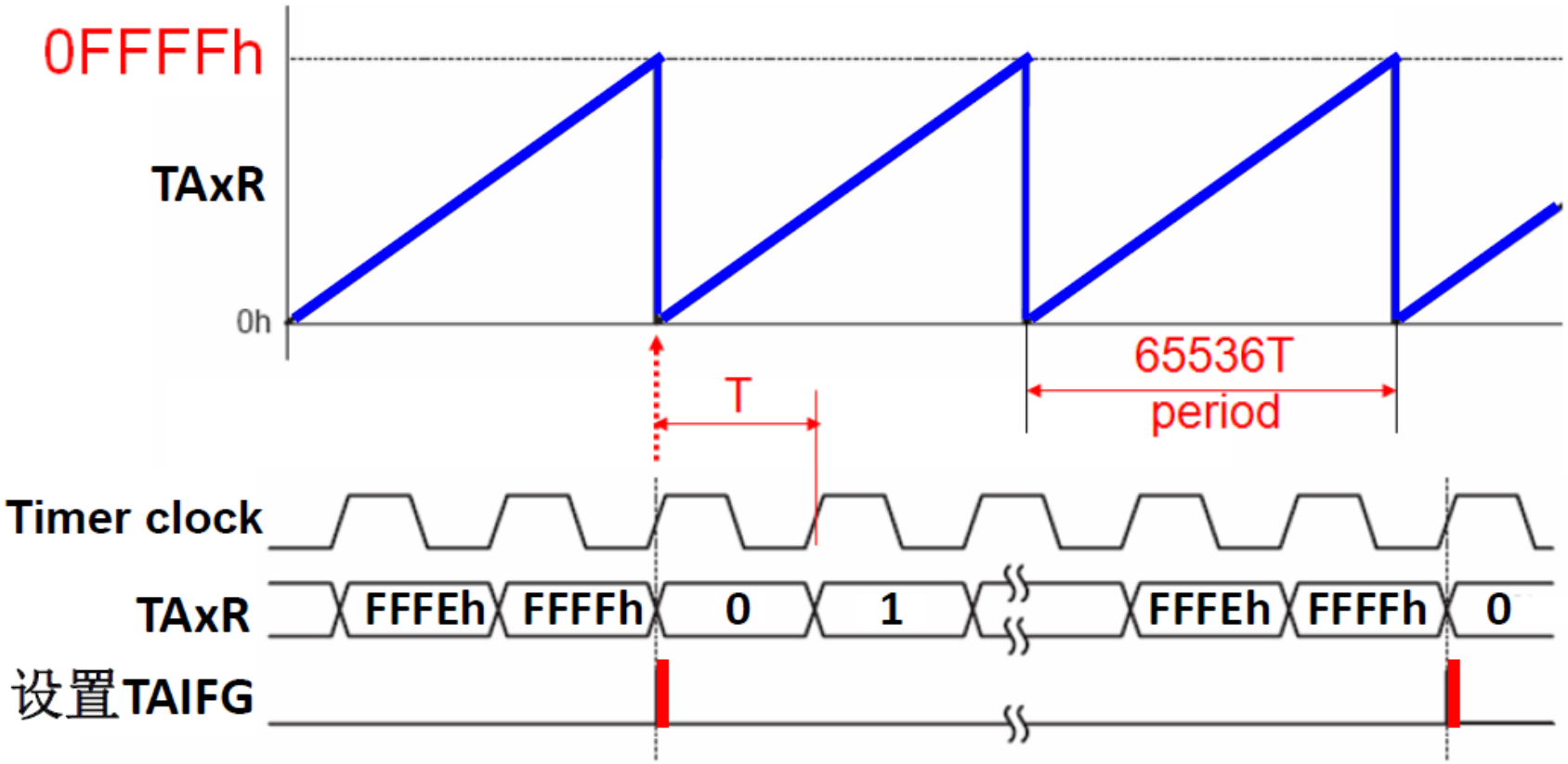
当TAR计数到 TACCR0时，置TACCR0 CCIFG =1

当TAR计数从值TACCR0变为0时，置 TAIFG =1

3) 连续计数方式(最大锯齿波方式) MCx= 10

Continuous mode

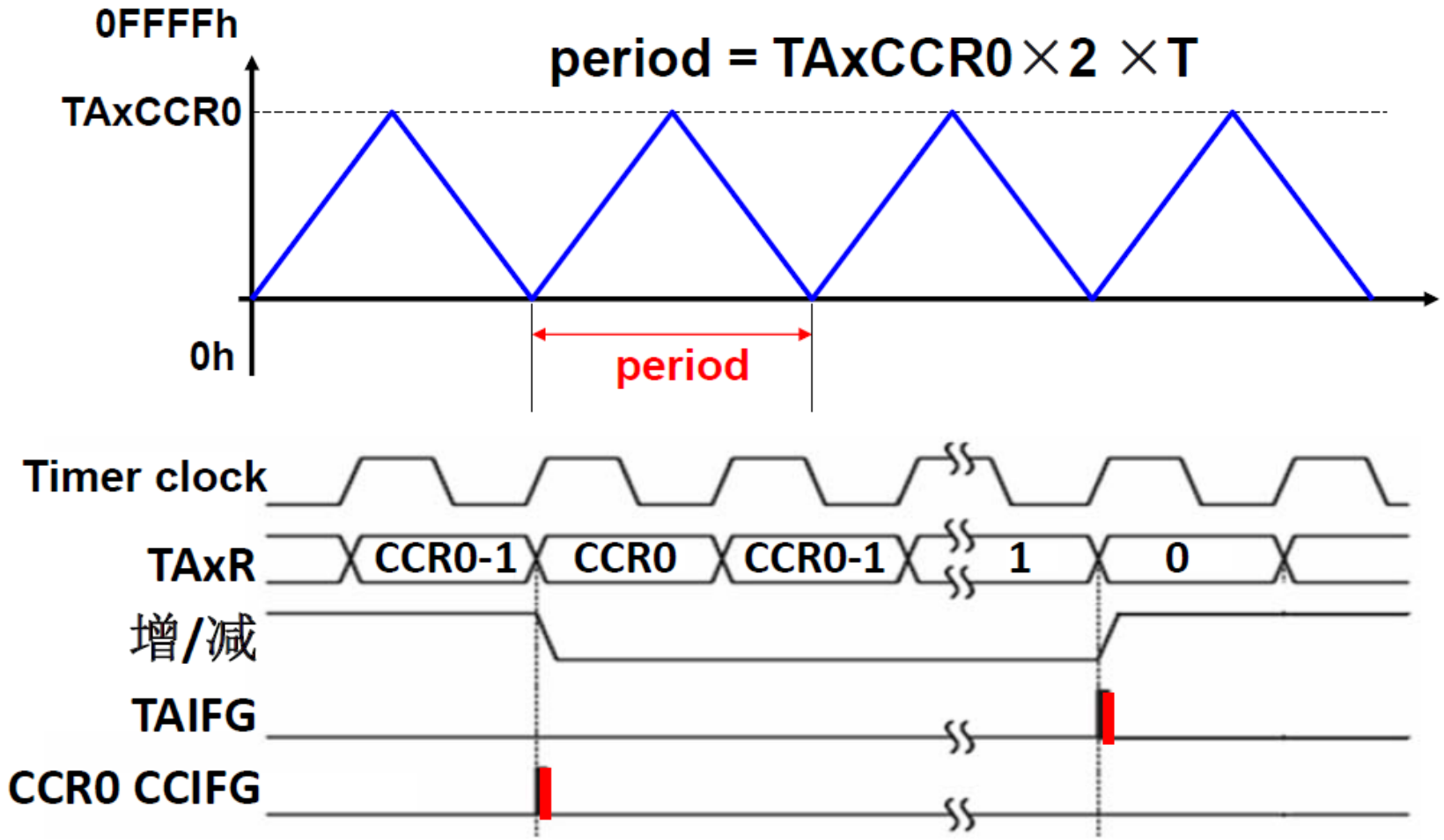
本方式不需要 CCR0 协助，使用方便，
但难以得到所希望的 period



当TAxR计数从值0FFFFh变为0时，置 TAIFG =1

4) 增/减计数方式(三角波方式) MCx= 11

Up/Down mode 需要 CCR0 协助



当TAxR增计数到 TAxCCR0时，置TAxCCR0 CCIFG =1, 并开始减计数，

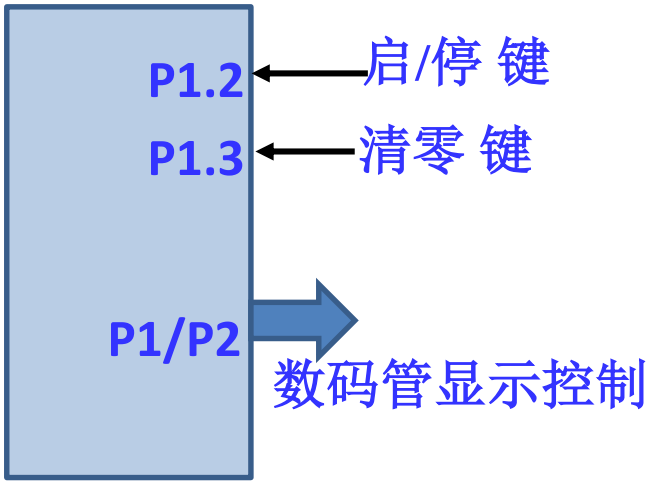
当TAxR减计数从值1变为0时，置 TAIFG =1, 并开始增计数

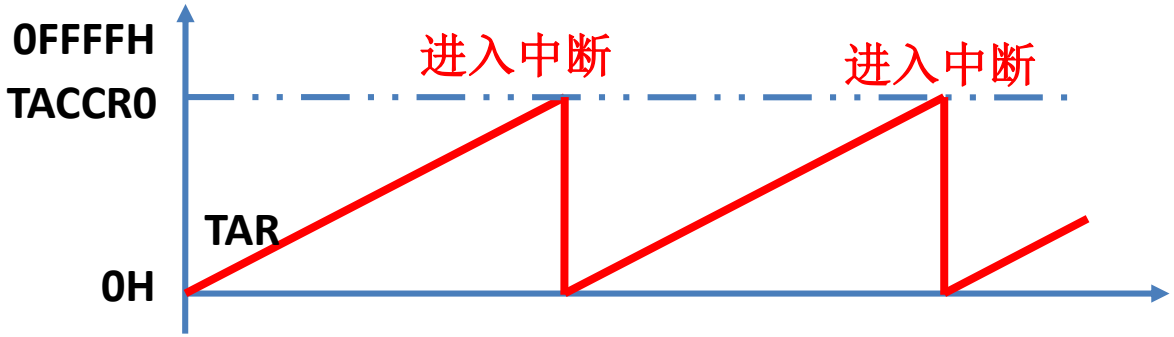
示例：典型的定时中断产生方法

```
#include "msp430.h"
void main( void )
{
   WDTCTL = WDTPW + WDTHOLD;
  _DINT();
  P1DIR|=BIT0; //选择P1.0为管脚输出

  TA0CCTL0=CCIE; //设定定时器CCR0中断使能
  TA0CTL=TASSEL_2+MC_1; //选择SMCLK时钟源,UP模式
  TA0CCR0= 1000;  //设定计数值 1000*1us=1ms计数周期
 _EINT();     //开中断
  LPM0;
 // while(1);
}
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A(void)
{
   P1OUT^=BIT0;  //通过P1.0反观察
}
```

设计实例：用定时中断实现一个秒表（精度1/100s）

方法：利用定时中断



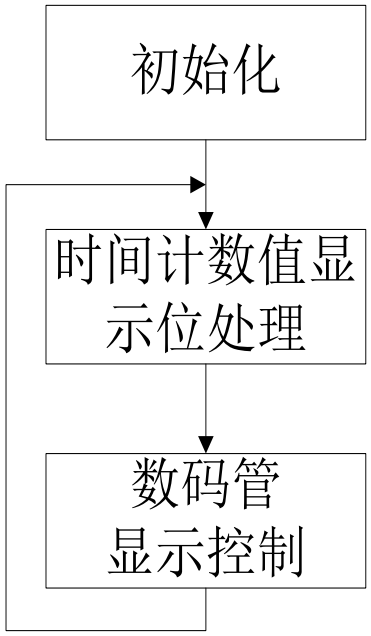


设计一个定时中断，每0.01s进入一次定时中断，将计数值加1

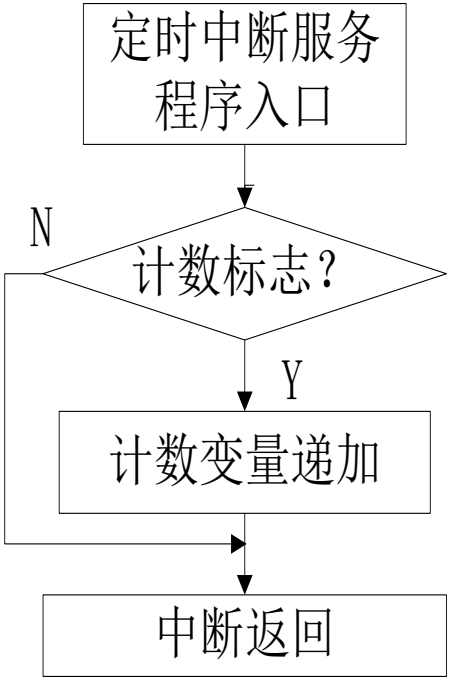
利用TA0CCR0中断

```
BCSCTL1 |= DIVA_0;  //ACLK采用外部晶振，1分频
TA0CCTL0=CCIE; //设定定时器CCR0中断使能
TA0CTL=TASSEL_1+MC_1; //选择ACLK时钟源, UP模式
TA0CCR0= 327;  //设定计数值 328*1/32768=0.01s计数周期
```

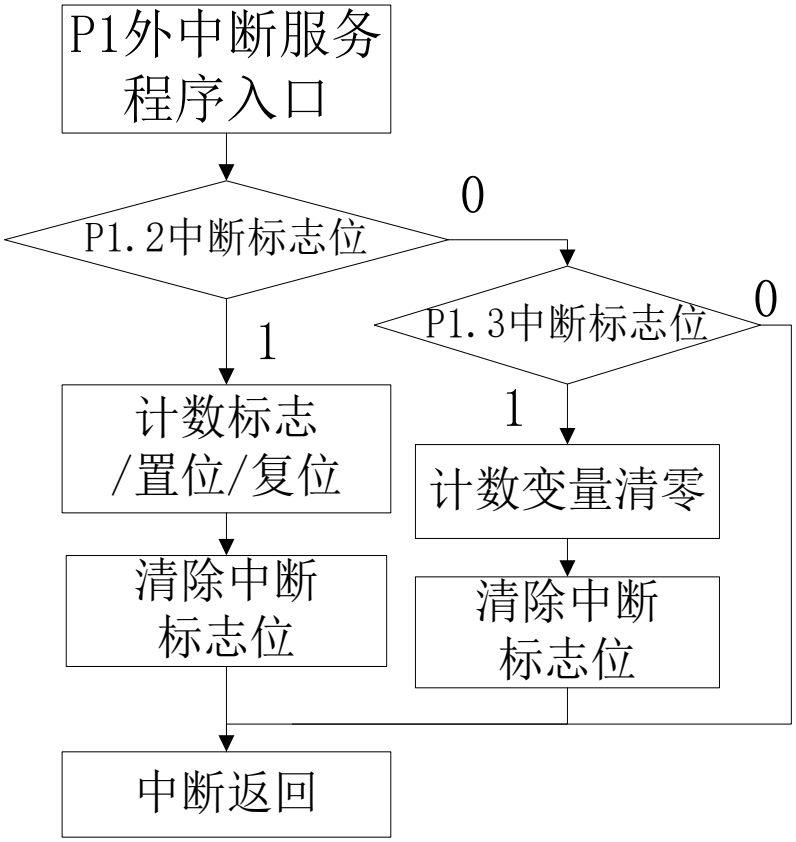
主函数



定时中断服务程序



外中断服务程序



三、TA的比较功能及PWM实现实例

捕获比较控制寄存器TACCTLx

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
CMx		CCISx		SCS	SCCI	Unused	CAP
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG

CAP=0 比较模式

- **一般用于PWM输出控制**

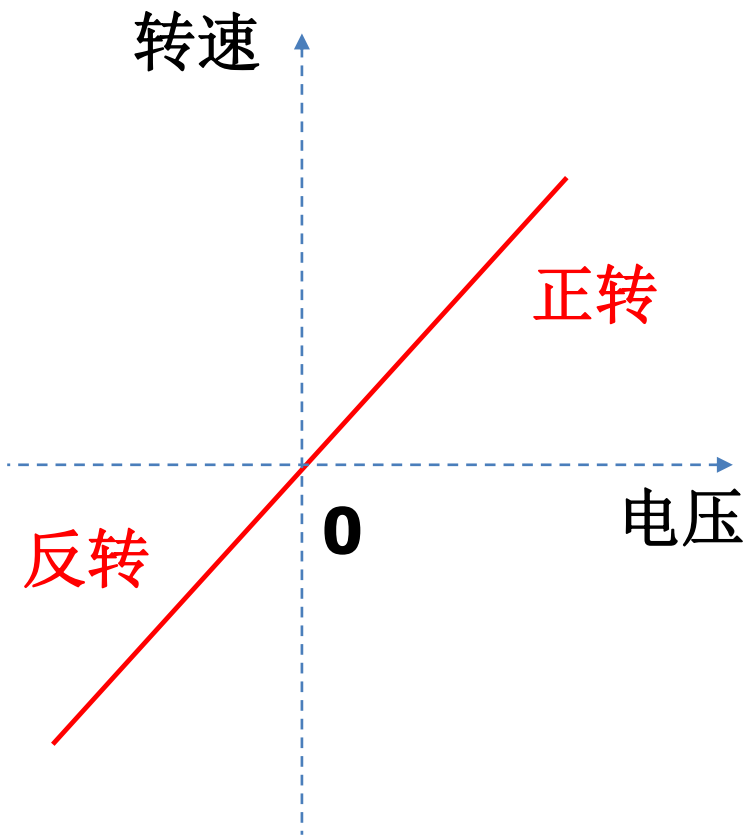
CAP=1 捕获模式

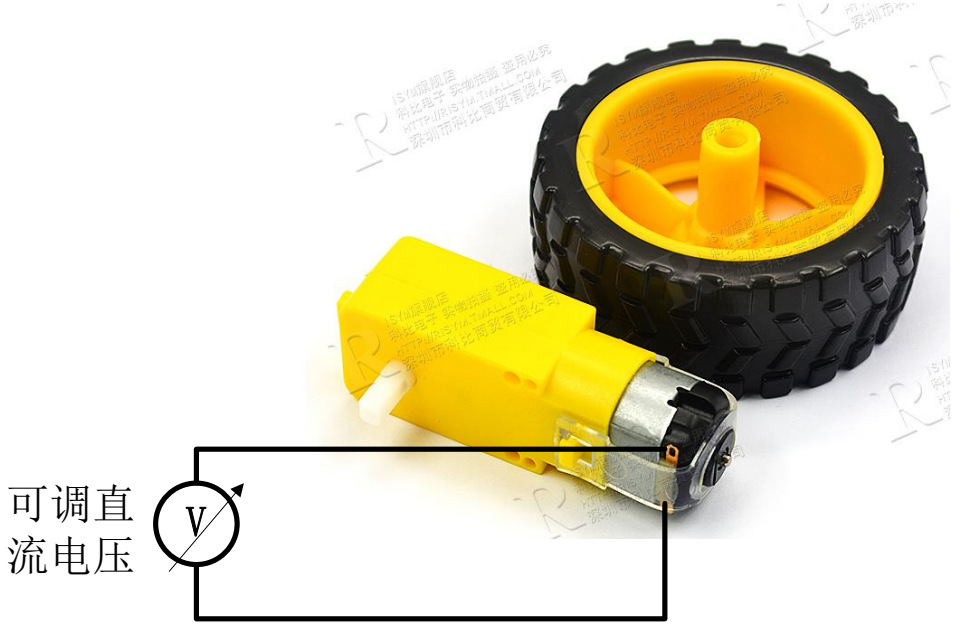
- 事件捕捉（变化沿）
- 脉冲宽度测量

- **TA0和TA1分别有多个捕获比较控制寄存器，编号为0,1,2**
- **使用时需要根据所使用的捕获比较控制寄存器，选择相应的管脚作为输出或输入（参考器件数据手册）**

实例：小车轮速控制

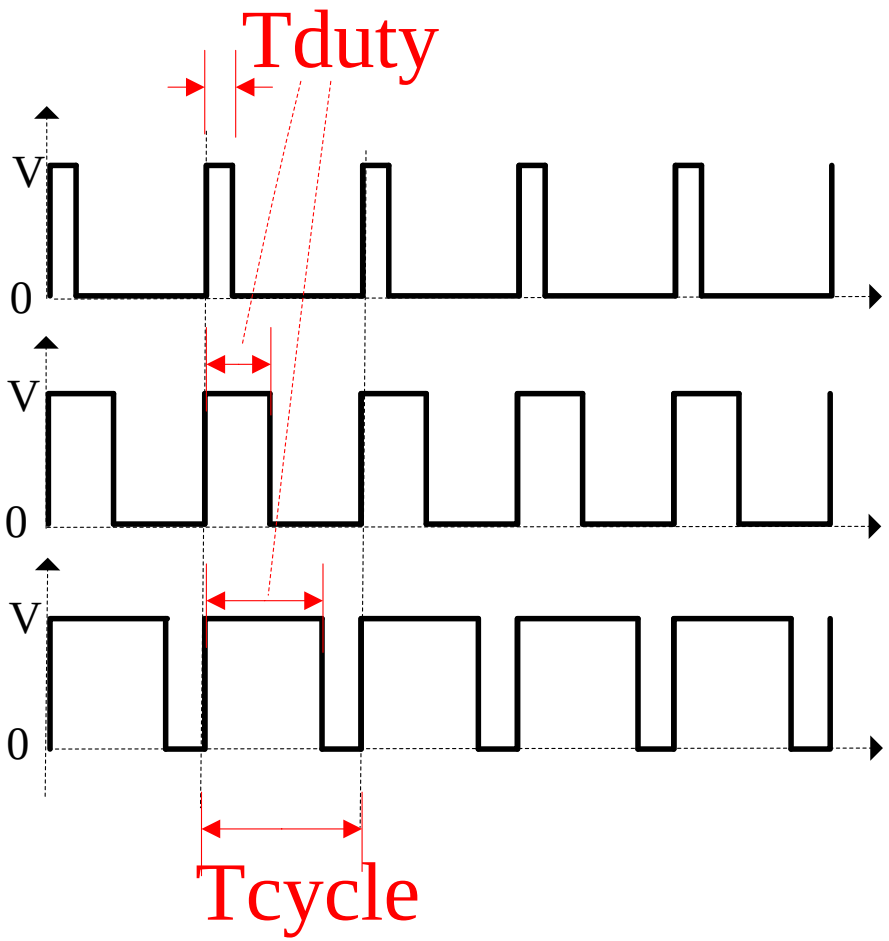
直流电机的调速





如何实现可调电压?

脉冲宽度调制　　PWM： Pulse Width Modulation



等效（周期平均）直流电压为：

$$V_o = \frac{T_{duty}}{T_{cycle}} V$$

Tcycle：固定，决定了脉冲频率

Tduty：连续变化，实现Vo的连续变化控制

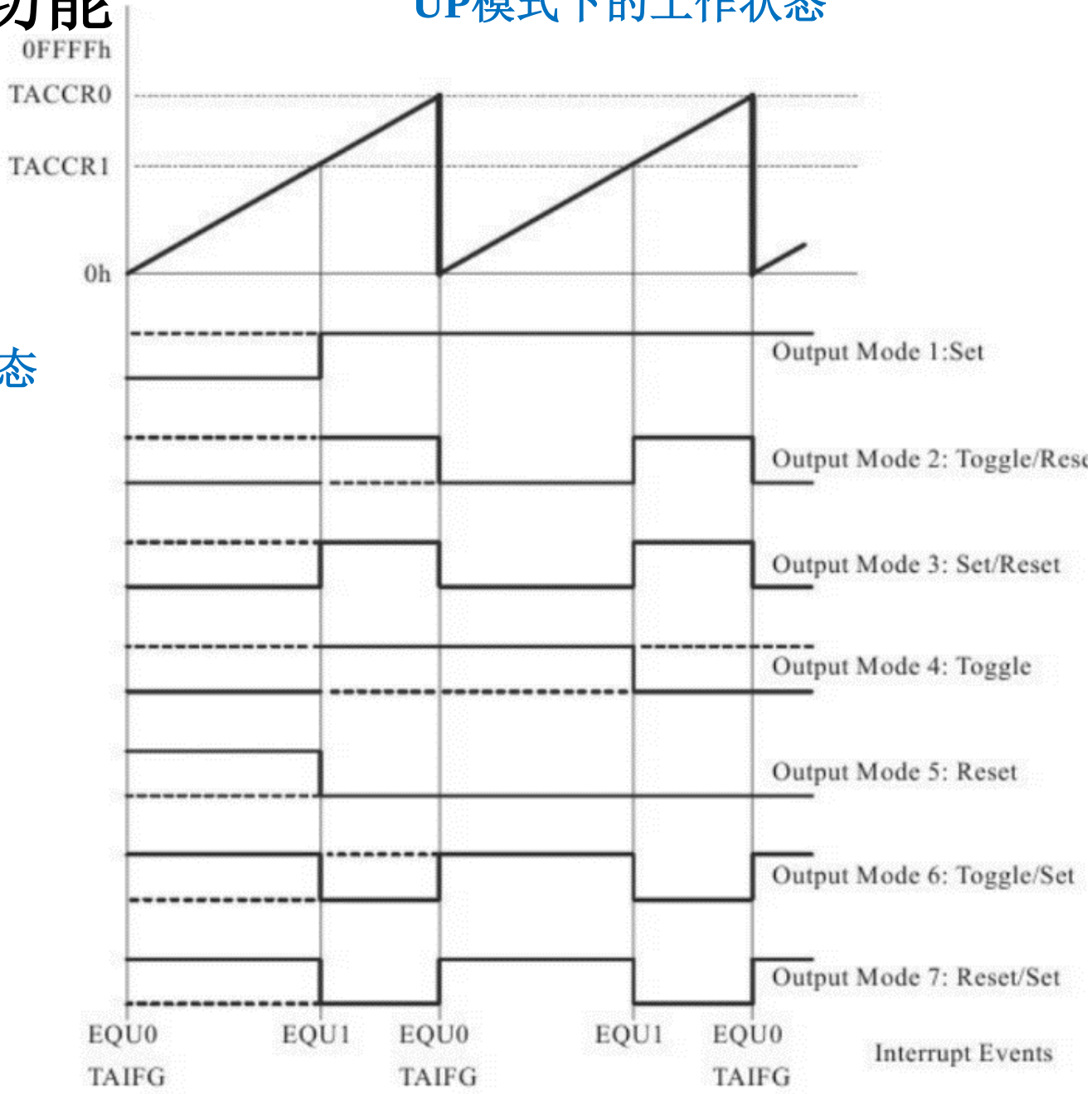
定时器的比较功能

捕获比较控制寄存器TAxCCTLx

CAP=0 比较模式

OUTMODx: 8种工作状态

UP模式下的工作状态



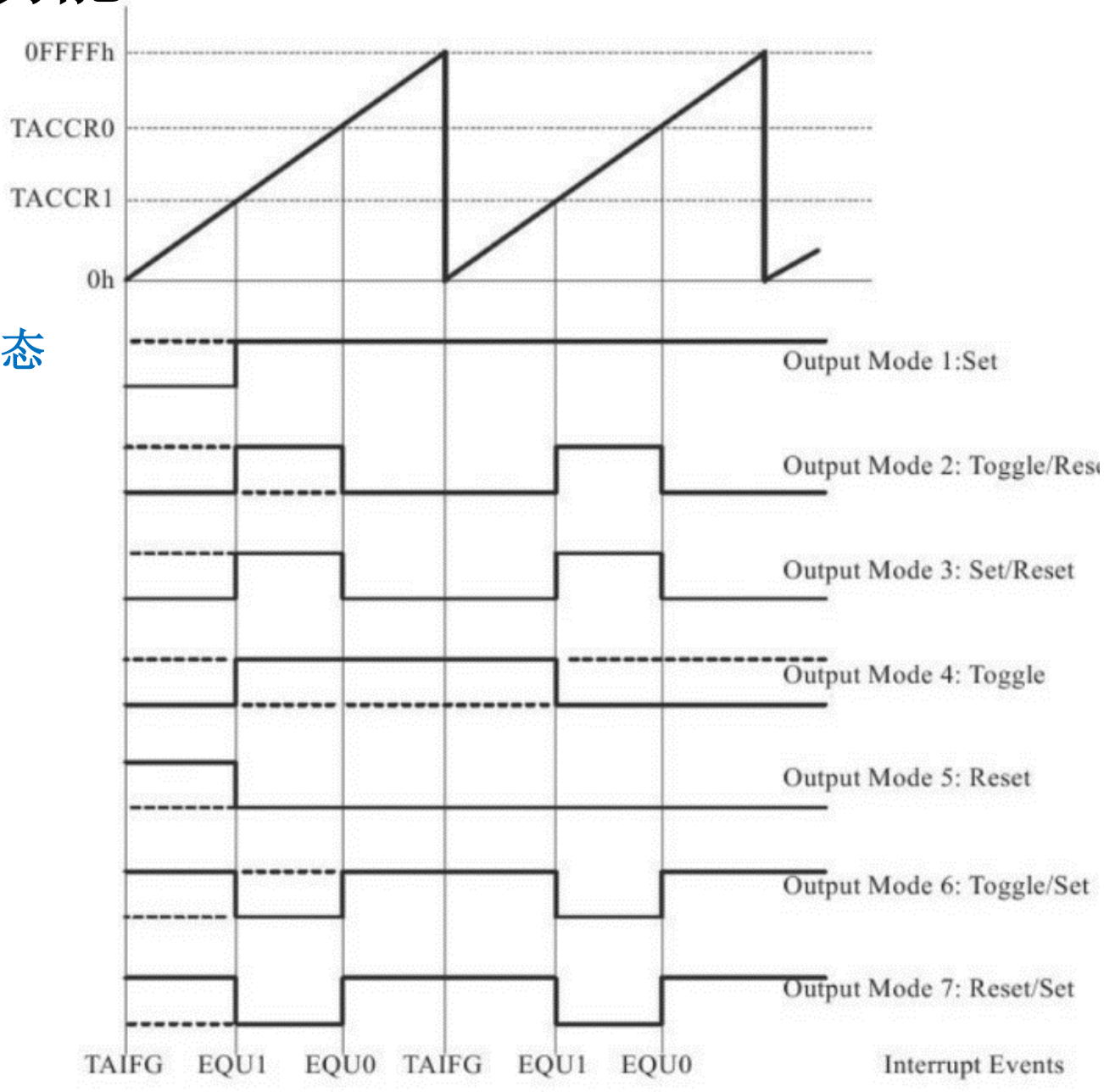
定时器的比较功能

捕获比较控制寄存器TAxCCTLx

CAP=0 比较模式

OUTMODx: 8种工作状态

Continuous模式下的工作状态



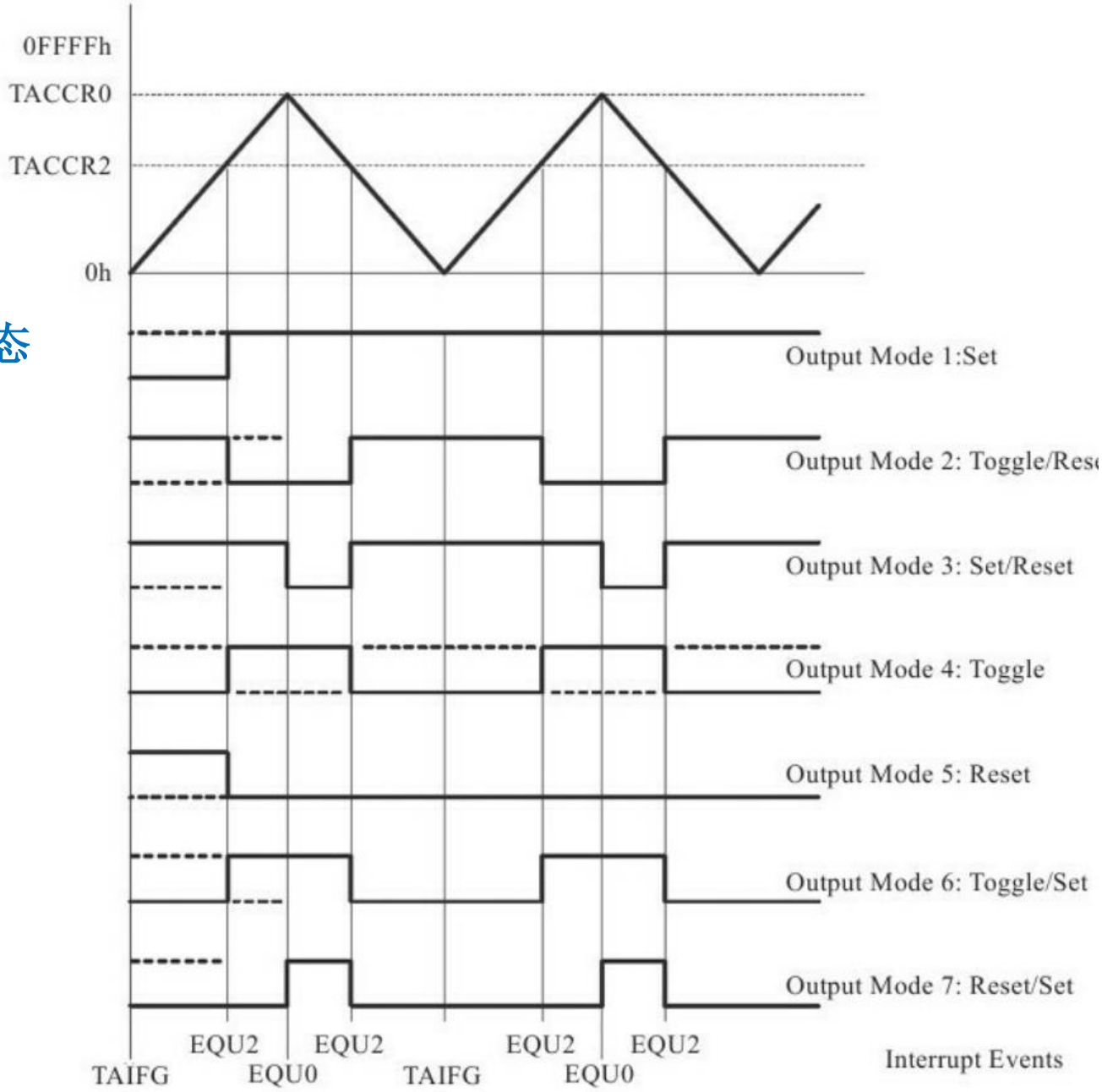
定时器的比较功能

捕获比较控制寄存器TAxCCTLx

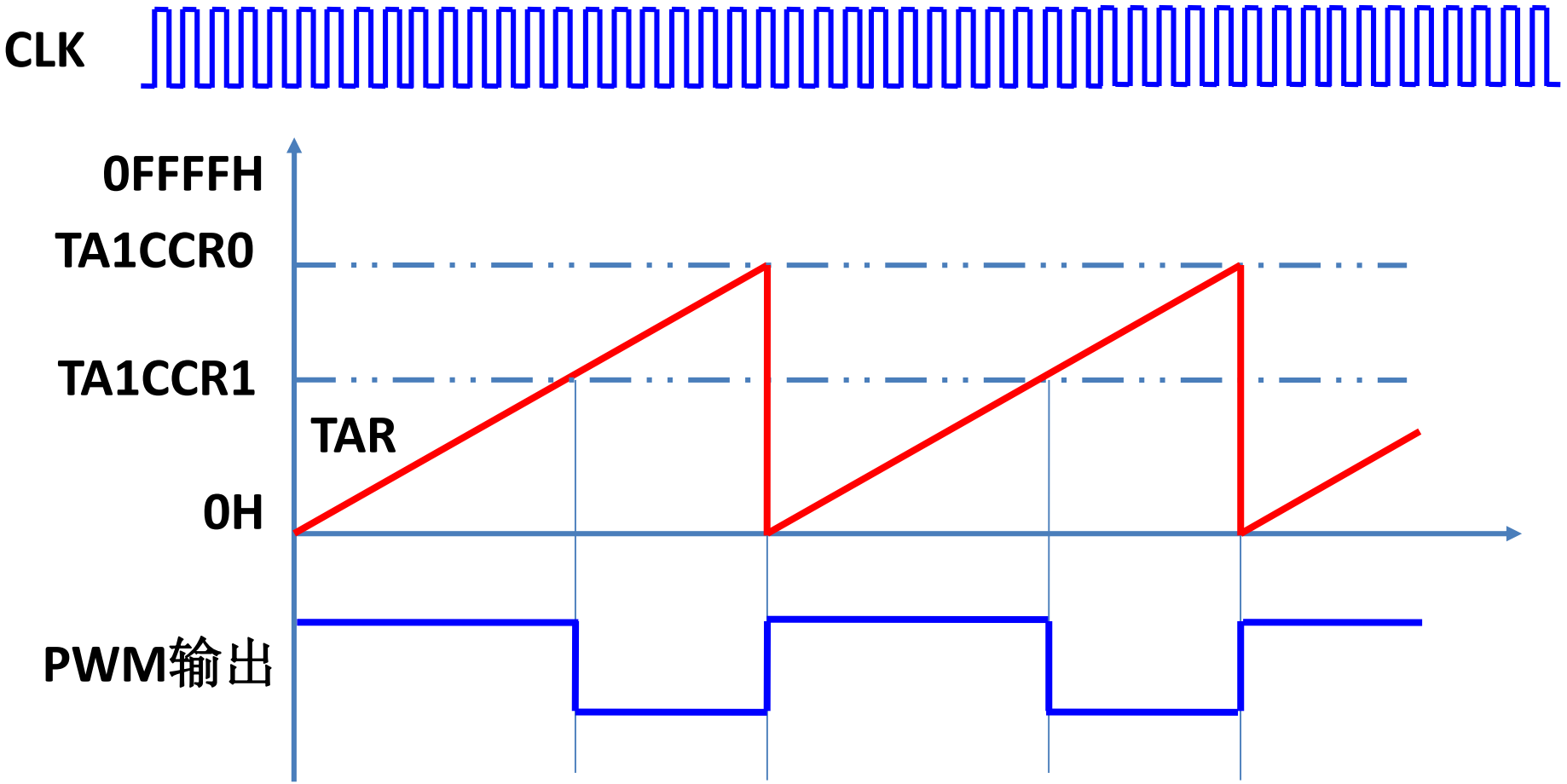
CAP=0 比较模式

OUTMODx: 8种工作状态

Up-down模式下的工作状态



PWM输出(比较输出) 以定时器TA1为例 （TA1CCR1）



TAR1<TA1CCR1，输出1

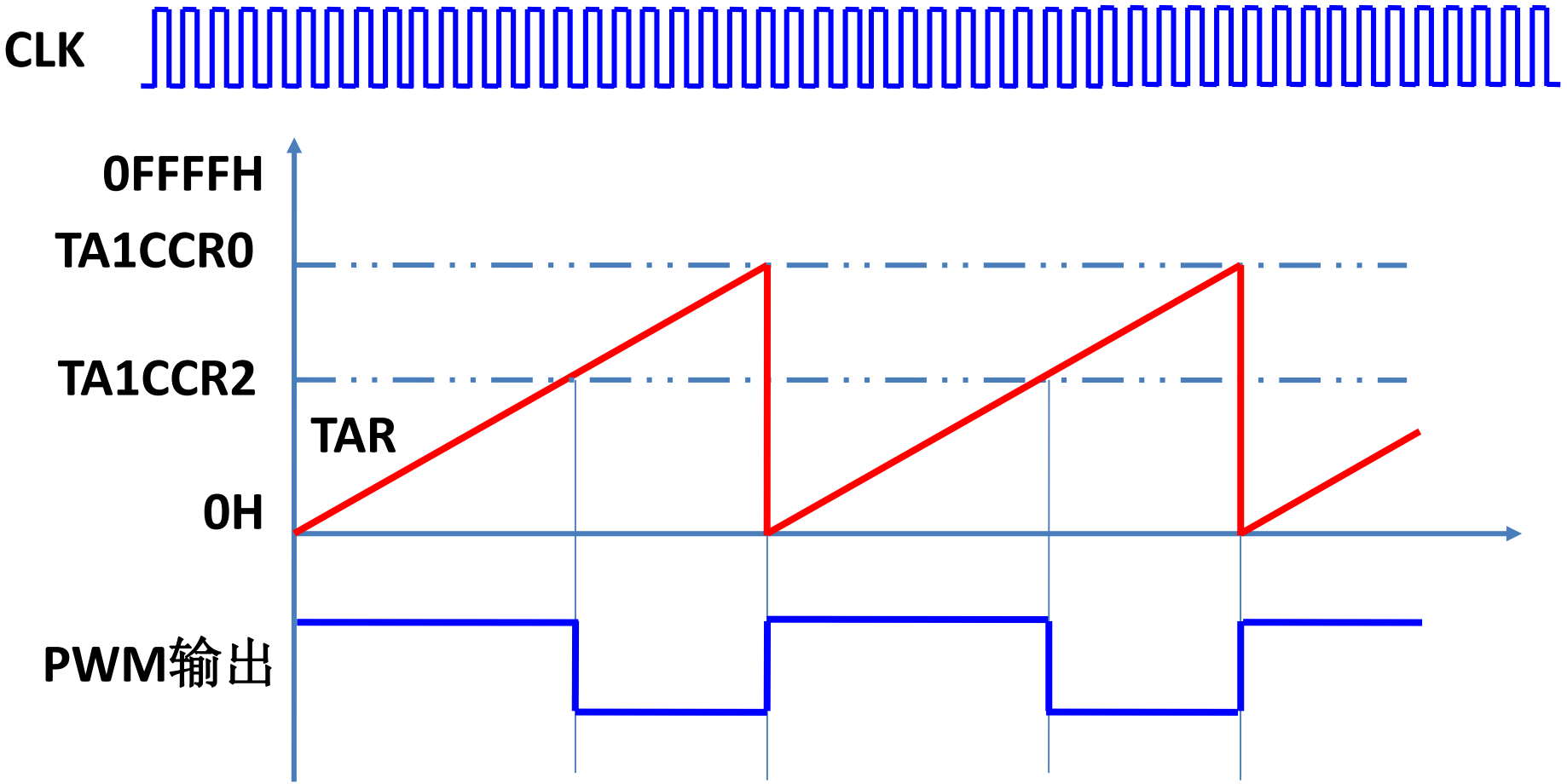
TAR1>TA1CCR1，输出0

TAR1=TACCR0，置TAR1=0

改变TA1CCR1，改变占空比

改变TA1CCR0，改变脉冲周期

PWM输出(比较输出) 以定时器TA1为例（TA1CCR2）



TAR1<TA1CCR2，输出1

TAR1>TA1CCR2，输出0

TAR1=TACCR0，置TAR1=0

改变TA1CCR2，改变占空比

改变TA1CCR0，改变脉冲周期

示例：利用比较模式产生一个固定占空比的脉冲信号

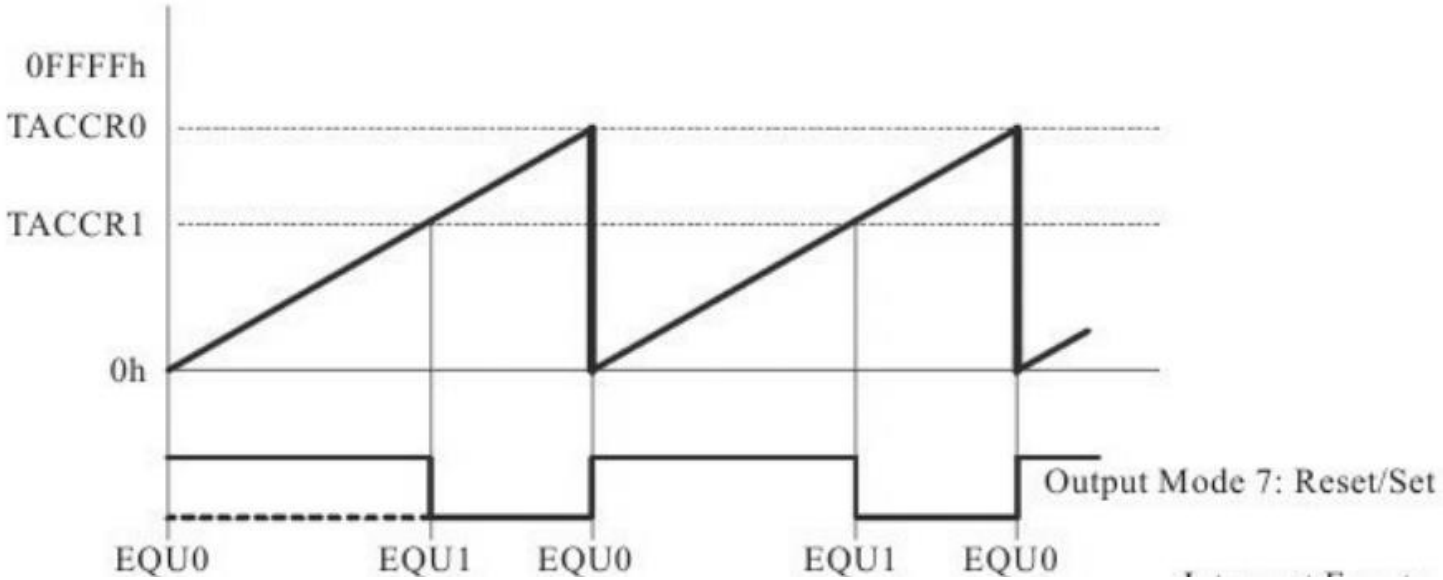
```
#include "msp430.h"
void main( void )
{
 WDTCTL = WDTPW + WDTHOLD;

  P1DIR|=BIT2;   //P1.2选择  TA0.1功能
  P1SEL|=BIT2;

  TA0CTL=TASSEL_2+MC_1; //选择SMCLK作为时钟源，缺省1us周期；选择UP模式
  TA0CCTL1=OUTMOD_7; //选择模式7 Reset/Set模式
  TA0CCR0=1000;  //CCR0控制周期，1000对应1kHz
  TA0CCR1=800;   //CCR1控制占空比，占空比0.8
  LPM0; //进入低功耗模式
}
```

P1.2管脚功能

P1.2/
TA0.1/
UCA0TXD/
UCA0SIMO/
A2/
CA2

为什么能够从P1.2管脚输出PWM脉冲信号？

注意：需要选择与所用的捕获比较寄存器对应的管脚，并选择功能



P1.2管脚功能

P1.2/
TA0.1/
UCA0TXD/
UCA0SIMO/
A2/
CA2

```
TA0CCTL1=OUTMOD_7; //选择模式7 Reset/Set模式
TA0CCR0=1000;  //CCR0控制周期，1000对应1kHz
TA0CCR1=800;   //CCR1控制占空比，占空比0.8
```

根据器件手册查找与所选择定时器对应的管脚

MSP430G2553：TA0.1, TA0.2 TA1.1, TA1.2（对应于使用的定时器和比较捕获控制寄存器编号）

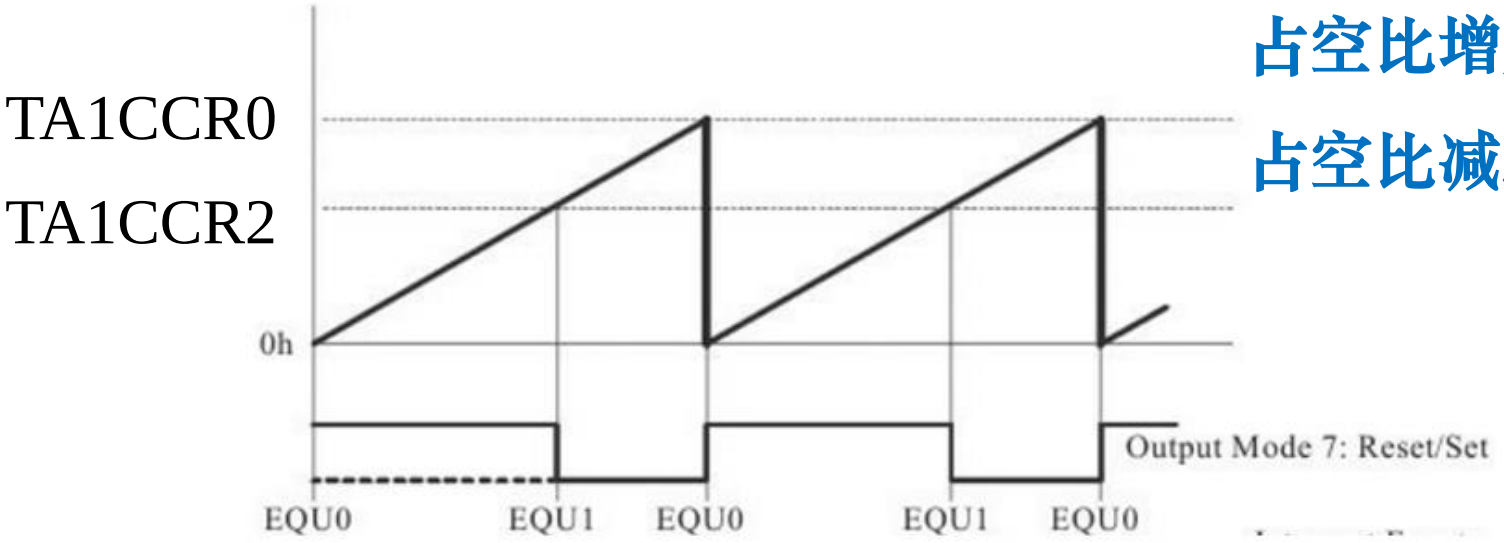
PxSEL设置为所选择的管脚功能

示例：利用按键，产生一个占空比变化的PWM信号

利用定时器的比较功能，输出一个PWM脉冲信号，控制车轮速度

占空比增大：转速增加

占空比减小：转速降低

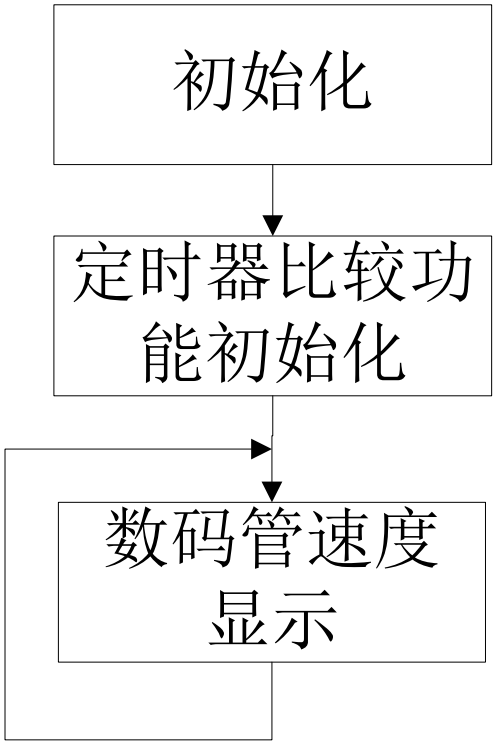


P2.4/
TA1.2

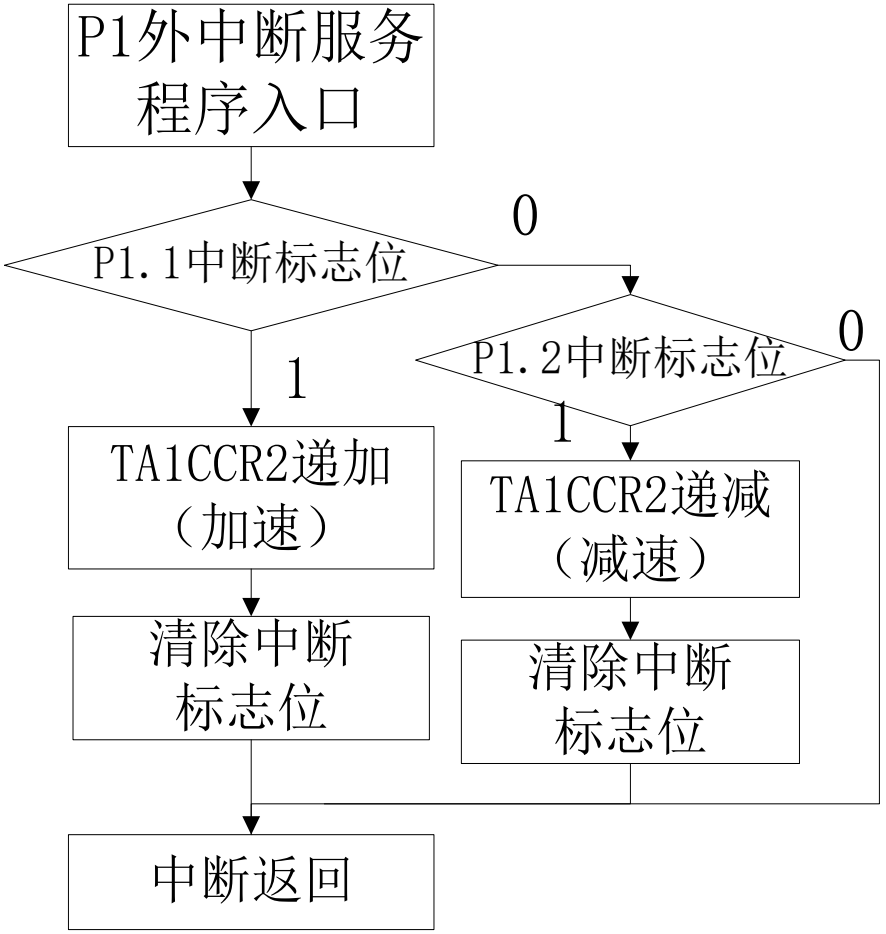
P2SEL |= (BIT4);
P2DIR |= (BIT4); **//P2.4作为TA1.2 PWM功能输出**

TA1CTL = TASSEL_2 + MC_1+ID_2; **//SMCLK时钟， UP模式，4分频， SMCLK时钟周期4us**
TA1CCR0 = Tcycle; **//设置PWM周期**
TA1CCTL2 = OUTMOD_7; **//模式7 reset/set模式**
TA1CCR2 = Tduty; **//设置初始占空比。**

主函数


初始化
定时器比较功能初始化
数码管速度显示

外中断服务程序


P1外中断服务程序入口
P1.1中断标志位
0
1
P1.2中断标志位
0
1
TA1CCR2递加（加速）
TA1CCR2递减（减速）
清除中断标志位
清除中断标志位
中断返回

以下几个方面来掌握定时器：

1. 时钟信号的选择(内部时钟、外部时钟)
2. 计数寄存器开始计数、停止计数的控制
3. 计数溢出的产生、标志置位、分中断允许的设置
4. 比较匹配、输入捕捉相关寄存器的设置
5. 比较匹配、输入捕捉的产生、标志置位、分中断允许的设置
6. 比较匹配对输出波形的控制(PWM)

关于定时器的详细使用方法可查阅相关参考书、器件手册和所提供的示例程序