DAC 使用指南

概述

本手册适用范围：

适用范围	
SPC1125 系列	SPC1125，SPC1128
SPC1168 系列	SPC1155， SPC1156， SPC1158， SPC1168，SPD1148， SPD1178， SPD1188， SPD1163，SPM1173
SPC2168 系列	SPC2168，SPC2165，SPC2166，SPC1198
SPC1169 系列	SPC1169，SPD1179，SPD1176
SPC2188 系列	SPC1185，SPC2188

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
C/0	2024-08-29	苏杭	Released	1. 首次发布。

术语或缩写

术语或缩写	描述
MCU	Microcontroller Unit，微控制器单元

1 SPC1168/SPC2168/SPC1169 系列

1.1 特性

包含 10 比特 DAC 和可以驱动容性负载的缓冲器（Buffer）。DAC 输出可以配置成直流电压和交流波形，如三角波、方波等。DAC 的输出可由下面表达式计算：

$$\mathrm{VOUT} = \frac{AVDD}{2^{10}} * CODE \quad (AVDD\text{典型值为 } 3.3V)$$

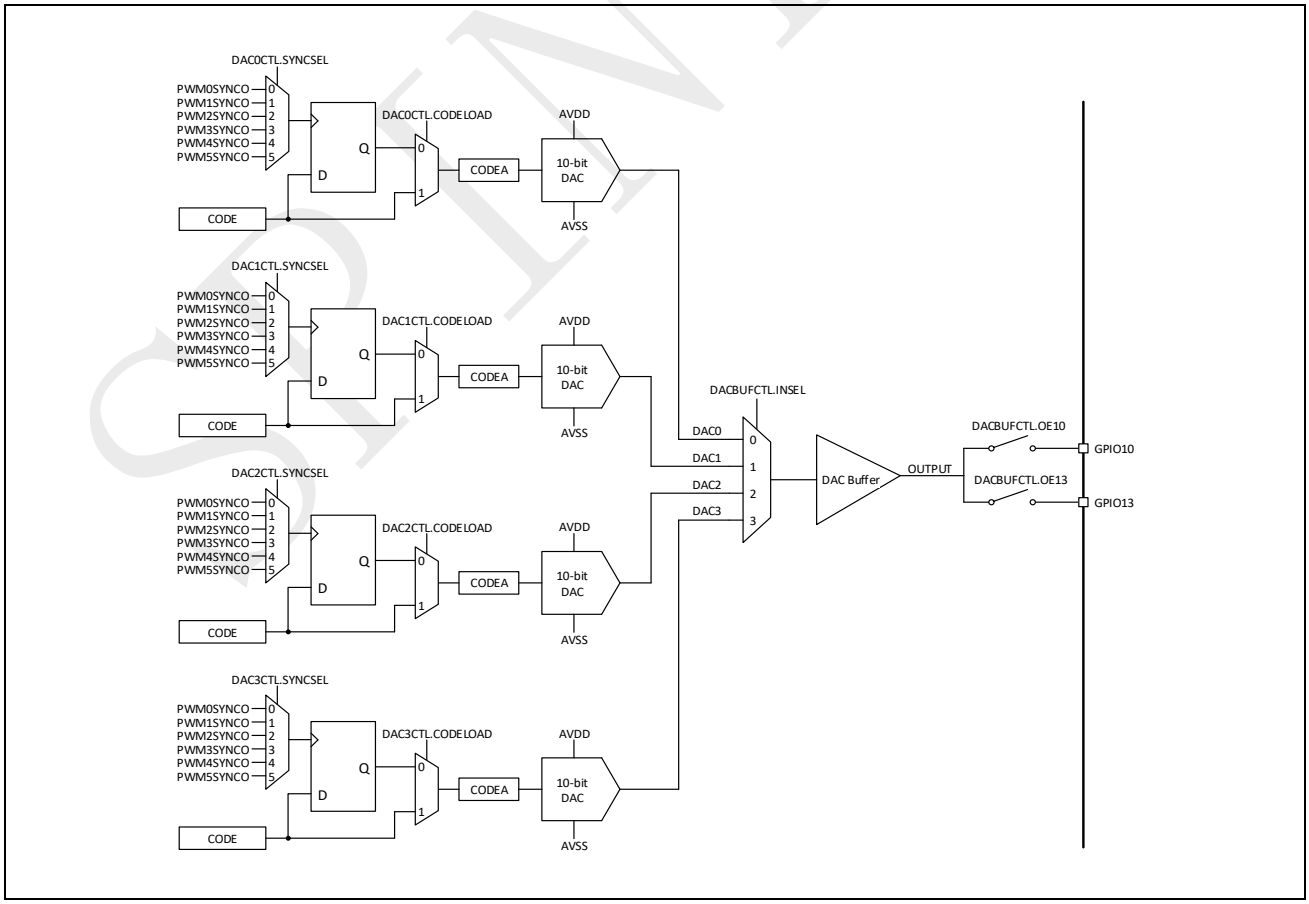
1.2 功能描述

如图 1-1 到图 1-3 所示。

- 当 DACxCTL.CODELOAD = 0 时，开启影子模式。此时写入 DACxCODE 寄存器的码值并不会立即生效，而是要等到由 DACxCTL.SYNCSEL 所选定的 PWMySYNCO 事件发生时才会更新 DACxCODEA。
- 当 DACxCTL.CODELOAD = 1 时，开启即时模式。此时写入 DACxCODE 寄存器的码值立即同步更新到 DACxCODEA。

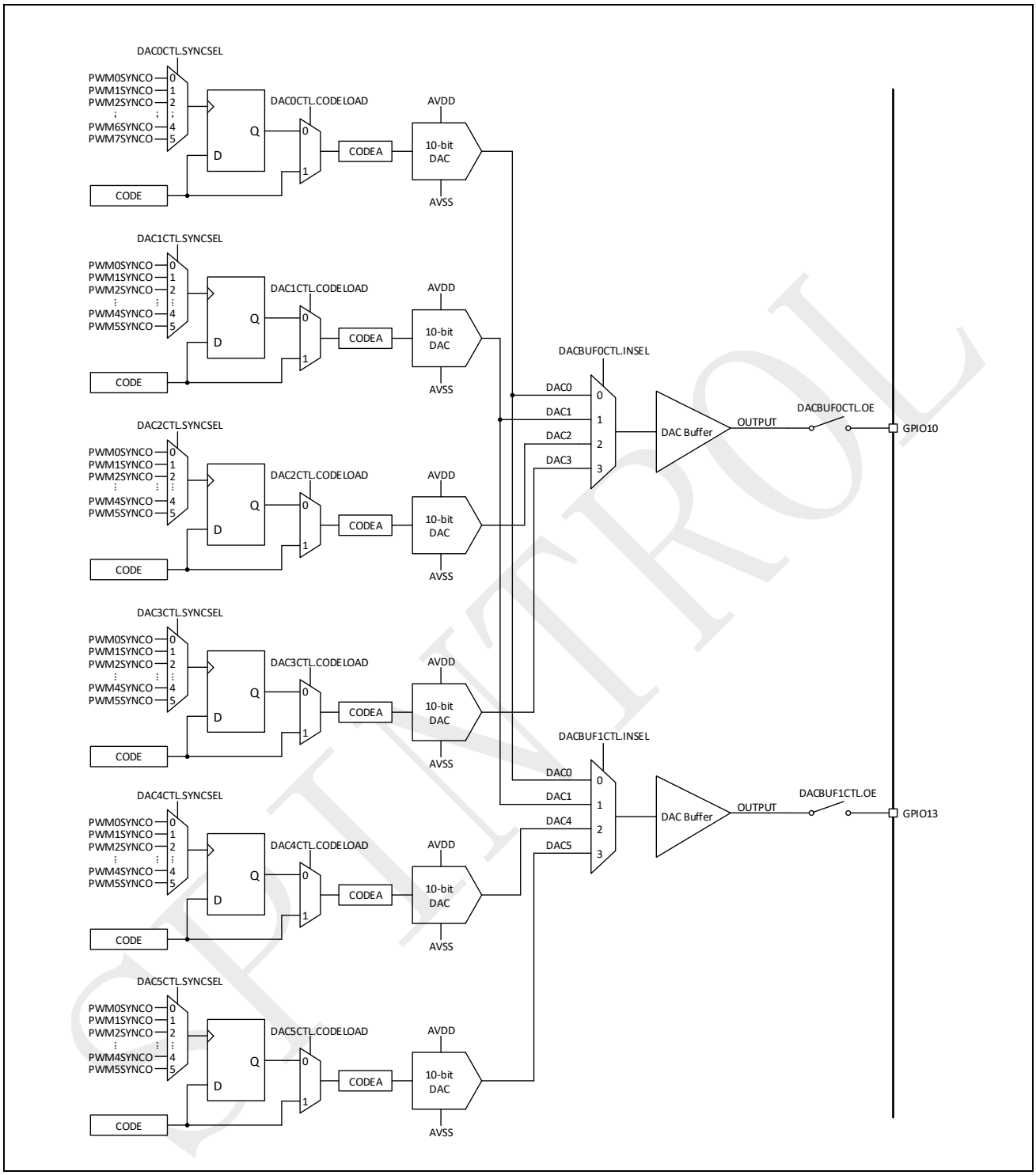
产生的模拟电压既可直接传给 COMP，亦可经过 DAC buffer 直接输出到对应 IO。

图 1-1：DAC 示意图



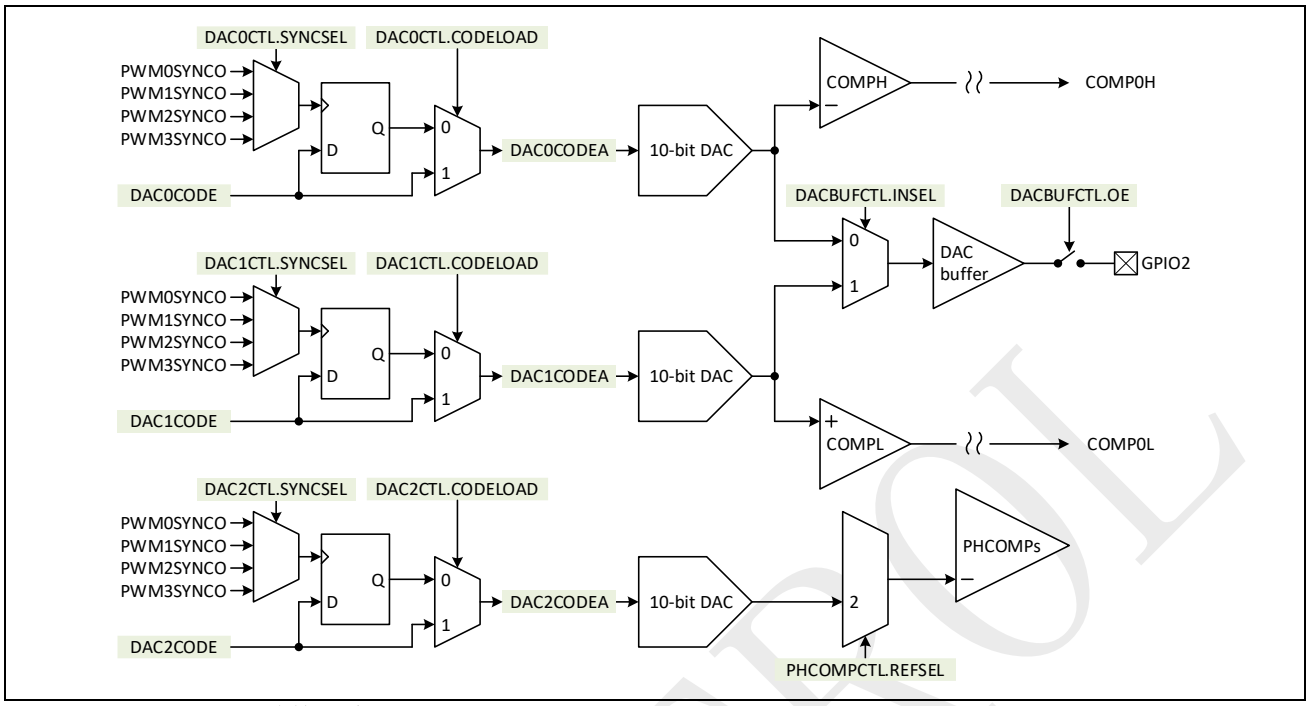
[1] SPC1168 DAC 结构示意图。

图 1-2：DAC 示意图



[1] SPC2168 DAC 结构示意图。

图 1-3：DAC 示意图



[1] SPC1169 DAC 结构示意图。

1.3 功能实例

1.3.1 输出恒定电压

1.3.1.1 功能需求

通过 DAC buffer 将 DAC 电压输出到引脚上。

1.3.1.2 功能实现

通过 DAC 在 GPIO2 输出 1.65V 电压：

Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);
    printf("Enter the test\n");

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 1.65V  */
    COMP_SetDACValue10Bit(DAC0, 512);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO();

    /* Init ADC and set DAC buffer as the positive input of the ADC CH0 */
    ADC_EasyInit1(ADC, ADC_CH0, ADC_IN_DAC0, ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Enable the global INT for the ADC */
    NVIC_EnableIRQ(ADCCH0_IRQn);

    /* Use software to trigger ADC CH0 to work */
    ADC_ForceChannelSOC(ADC, ADC_CH0);

    while (1)
    {

    }
}
```

[1] 示例代码适用于 SPC1169，其它系列产品的示例代码会根据实际需求进行补充。

1.3.2 斜坡发生器

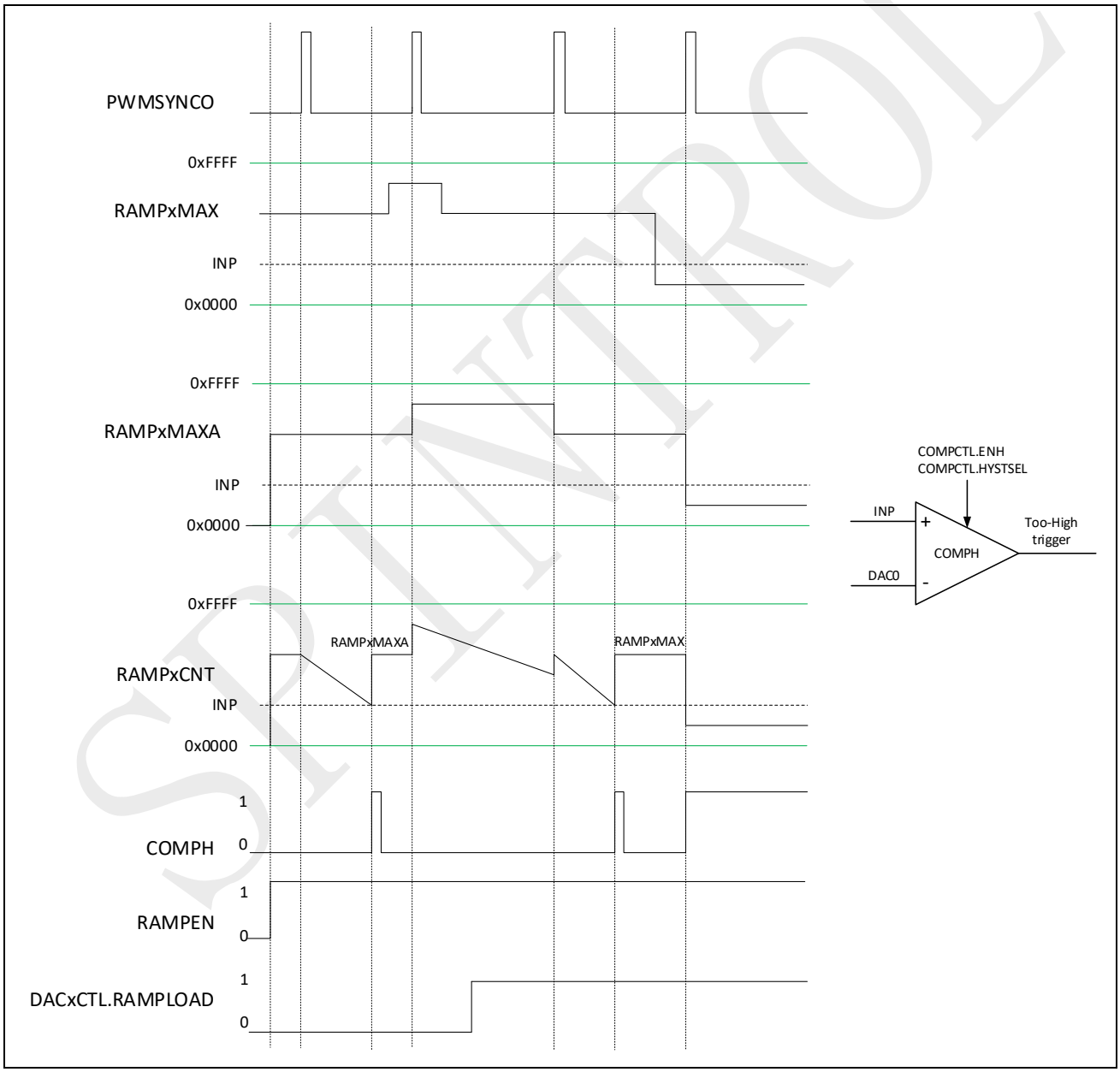
1.3.2.1 功能需求

芯片输出斜坡电压。

1.3.2.2 功能实现

PWM，DAC，COMPH 可以共同构成斜坡发生器，产生斜坡电压 RAMPxCNT，如图 1-4 所示。

图 1-4：斜坡发生器

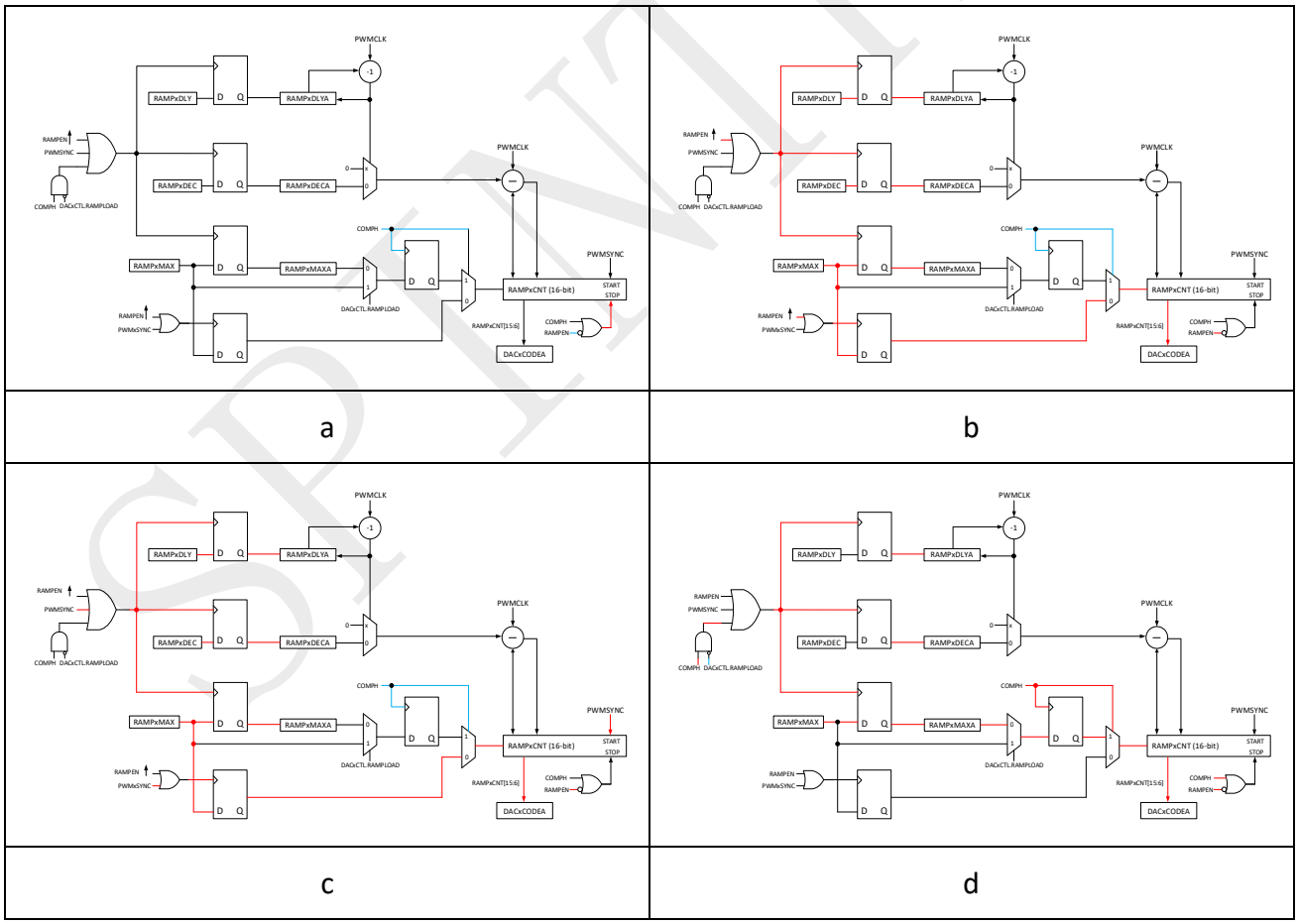


- 因为 DAC0 只能接到 COMPH 的负端，为了保证默认状态无 COMPH 事件，必须保证默认状态 DAC0CODEA(RAMP0CNT[15:6])电压 > INP；

- 初始条件 COMPH 是禁止的（通过软件设定），因此虽然 INP > DAC0CODEA(RAMP0CNT[15:6]) 电压，但是没有对应的 COMPH 事件；同时 RAMPEN 为低，停止 RAMPxCNT 的递减计数，如图 1-5a；
- 下一时刻，RAMPEN 产生上升沿脉冲，由于此时 COMPH 仍是禁止，没有 COMPH 事件，从而控制下路连通，RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器；当 RAMPxCNT 加载完成后，其[15:6]位构成 DACxCODEA，在其模拟电压稳定后，使能 COMPH，此时 DAC0CODEA(RAMP0CNT[15:6])电压>INP，没有 COMPH 事件，COMPH 事件维持低电平图 1-5b；
- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-5c；
- 当 DACxCODEA 电压递减到<INP 时，COMPH 事件触发，由于 RAMPLOAD 为 0，RAMPxMAXA 通过触发器直接加载到 RAMPxCNT，同时各寄存器初始值均加载到对应的活跃寄存器，COMPH 事件消失，同时 COMPH 关闭 RAMPxCNT 的递减计数图 1-5d；

其余时刻动作可在此基础类推。

图 1-5：根据 PWM 同步输出信号递减电平



示例演示了通过定时器产生 SYNCI 信号并通过 SYNCO 触发 DAC 递减计数，从而产生斜坡电压。

实验时，COMPH 对应正端电压必须大于 0V，否则负端电压无法降落到正端电压之下。

Example Code

```
int main()
{
    FLASH_WALLOW();

#if defined(SPC1158)
    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_100MHZ);
#else
    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
#endif

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DAC0CTL_BIT_CODELOAD_DIRECT_MODE);

    /* set DAC0 output to 3000mV  */
    COMP_SetDACVoltage(DAC0, 3000);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0, FALSE, TRUE);

    /* Select the PWM synchronous output signal for DAC */
    COMP_SetDACSyncEvent(DAC0, SEL_PWM0);

    CLOCK_EnableModule(PWM_MODULE);

    /* Connect the input sync signal with output */
    PWM_SetSyncOutEvent(PWM0, SYNC_SYNCI_AND_FRCSYNC);

    /* Enable PWM SYNC by the signal coming from TIMER1 */
    PWM_EnableSyncFromTIMER1(INC_PWM0 );

    printf("PWMCFG->TMR1SYNCIEN is %x\n", PWMCFG->TMR1SYNCEN.all);
```

Example Code

```
    /* Init TIMER1 */
    TIMER_Init(TIMER1, 1);

    /* Set TIMER1 generate SYNC to PWM when count down to 0 */
    TIMER_EnablePWMSync(TIMER1);

    /* Open Global INT for TIMER1 */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Run(TIMER1);

    COMP->RAMP0DLY.all = 200;
    COMP->RAMP0DEC.all = 1;
    COMP->DAC0CTL.bit.RAMPEN = ENABLE;

    /* Enable Comparator clock */
    CLOCK_EnableModule(COMP_MODULE);

    /* Enable the ADC Bandgap */
    ADC_EnableBandgap();

    /* Enable comparator */
    COMP_Enable(COMP_0_HI);

    /* Set Input channel */
    COMP_SelectChannel(COMP_0_HI, COMP0_FROM_ADC0);

    COMP->COMP0CTL.bit.REFSEL = 0;

    /* Set comparator output = Filtered */
    COMP_SetOutputType(COMP_0_HI, COMP_OUTPUT_FILTERED);

    /* Set Filter window - Threshld value = 0.6 * Window Size */
    COMP_SetFilterWindowTimeNs(COMP_0_HI, 200, 200 * 3 / 5);

    /* Clear latched filter status */
    COMP_ClearAllFilterOutputStatus();

    printf("COMP->DAC0CODE is %d\n", COMP->DAC0CODE.all);

    COMP->RAMP0MAX.all = (COMP->DAC0CODE.all << 6U);

    printf("COMP->RAMP0MAX is %d\n", COMP->RAMP0MAX.all);

    while (1)
    {

    }
}

void TIMER1_IRQHandler()
{
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例代码适用于 SPC1168。

Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);
    printf("Enter the test\n");

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 3000mV  */
    COMP_SetDACVoltage(DAC0, 3000);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO();

    /* Select the PWM synchronous output signal for DAC */
    COMP_SetDACSyncEvent(DAC0, SEL_PWM0);

    CLOCK_EnableModule(PWM_MODULE);

    /* Connect the input sync signal with output */
    PWM_SetSyncOutEvent(PWM0, SYNCO_SYNCI_AND_FRCSYNC);

    /* Enable PWM SYNC by the signal coming from TIMER1 */
    PWM_EnableSyncFromTIMER1(INC_PWM0 );

    printf("PWMCFG->TMR1SYNCIEN is %x\n", PWMCFG->TMR1SYNCIEN);

    /* Init TIMER1 */
    TIMER_Init(TIMER1, 1);

    /* Set TIMER1 generate SYNC to PWM when count down to 0 */
    TIMER_EnablePWMSync(TIMER1);

    /* Open Global INT for TIMER1 */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    COMP_SetDACRampDelay(DAC0, 200);
    COMP_SetDACRampDecrement(DAC0, 1);
    COMP_EnableDACRamp(DAC0);

    /* Set DPGAP as input to comparator, set Too High threshold is 3000mV,
filter window is 200ns */
    COMP_Init(COMP_H, COMP_FROM_ANA_IN0, 3000, 200);
```

Example Code

```
    printf("COMP->DAC0CODE is %d\n", COMP->DAC0CODE);

    COMP_SetDACRampReloadValue(DAC0, (COMP->DAC0CODE << 6U));

    printf("COMP->RAMP0MAX is %d\n", COMP->RAMP0MAX);

    while (1)
    {

    }
}


void TIMER1_IRQHandler()
{
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例代码适用于 SPC1169，其它系列产品的示例代码会根据实际需求进行补充。

2 SPC2188 系列

2.1 特性

SPC2188 提供了 5 个 10 位 DAC 和 1 个 12 位 DAC，可以通过数字码值的控制，输出静态电压或者动态波形。同时还提供了 2 个可以驱动容性负载的 DAC 缓冲器。

N 位 DAC 的输出电压由 DACxCODEA 寄存器控制，可以计算为：

$$V_{DACx} = V_{AVDD} \times DACxCODEA / 2^N$$

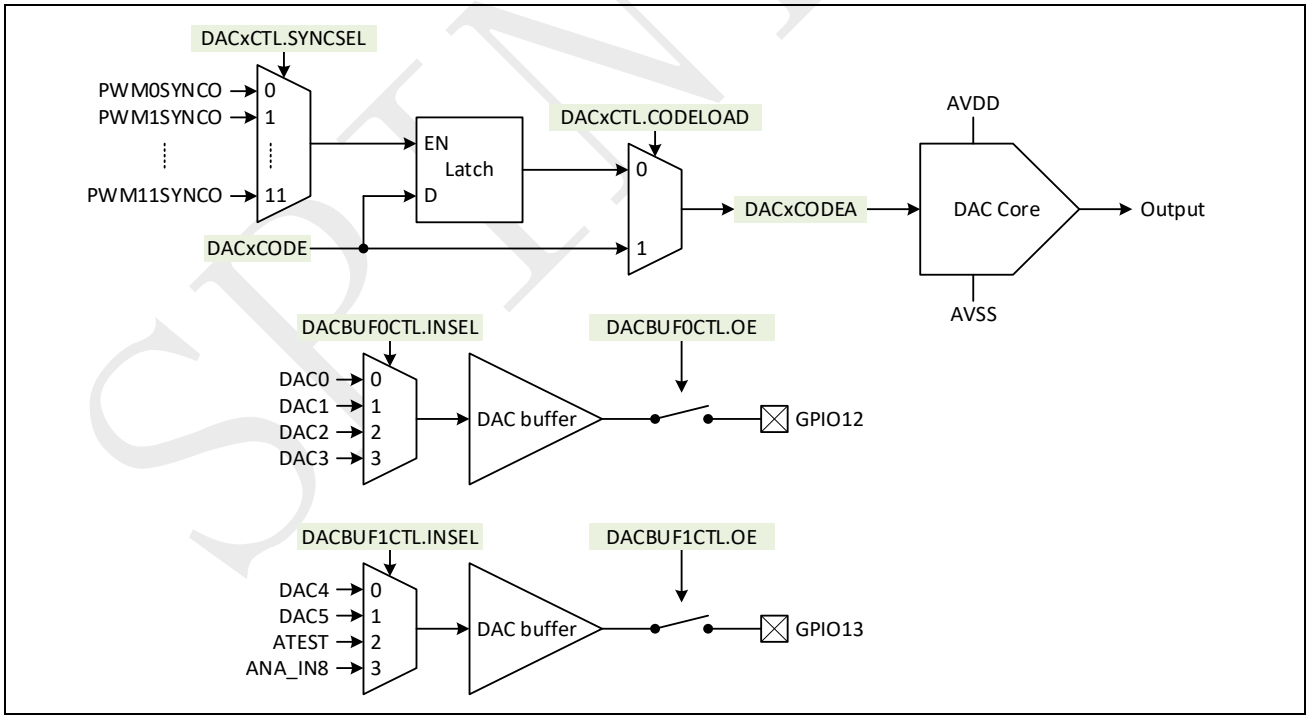
2.2 功能描述

DAC 及缓冲器的框图如图 2-1 所示。

- 当 DACxCTL.CODELOAD = 0 时，开启影子模式。此时写入 DACxCODE 寄存器的码值并不会立即生效，而是要等到由 DACxCTL.SYNCSEL 所选定的 PWMySYNCO 事件发生时才会更新 DACxCODEA。
- 当 DACxCTL.CODELOAD = 1 时，开启即时模式。此时写入 DACxCODE 寄存器的码值立即同步更新到 DACxCODEA。

产生的模拟电压既可直接传给 COMP，亦可经过 DAC buffer 直接输出到对应 IO。

图 2-1：DAC 示意图



2.3 功能实例

2.3.1 输出恒定电压

2.3.1.1 功能需求

通过 DAC buffer 将 DAC 电压输出到引脚上。

2.3.1.2 功能实现

通过 DAC 在 GPIO12 输出 1.65V 电压。

Example Code

```
#include "spc2188.h"
#include <stdio.h>


/* ADC result convert to voltage can calculate as the follow */
#define                     ValueToVoltage(x)           ((x*3339)/8192)


int32_t                     i32VSP;                             /* Result Value */

int main(void)
{
    CLOCK_InitWithRCO(240000000U);

    Delay_Init();

    /*
    * Initial the UART
    */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* Initial ADC and set DAC buffer as the negative input of the ADC CH0 */
    ADC_Init(ADC1, ADC_CH0, ADC1_SH0_P_GND, ADC1_SH0_N_DACBUF0,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 1.65V  */
    COMP_SetDACCode(DAC0, 512);

    /* DAC Buffer Initial, only DACBUF0_FROM_DAC0 ~ DACBUF0_FROM_DAC3 connect to
DACBUF0*/
    COMP_SetDACBuffer0Input(DACBUF0_FROM_DAC0);
    COMP_EnableDACBuffer(DACBUF0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO(DACBUF0);
```

Example Code

```
    /* Enable ADC1_IRQn */
    NVIC_EnableIRQ(ADC1CH0_IRQn);

    while(1)
    {
        /* Use software to trigger ADC CH0 to work */
        ADC_ForceChannelSOC(ADC1,ADC_CH0);

        Delay_Ms(500);
    }
}


void ADC1CH0_IRQHandler(void)
{
    /* Result gotten from 'ADC_GetChannelResult()' can be a negative value or a
positive value */
    i32VSP = ADC_GetChannelResult(ADC1,ADC_CH0);
    printf("ADC data is %d\n",i32VSP);
    printf("voltage = %d mV\n", ValueToVoltage(i32VSP));

    /* Clear SOC INT */
    ADC_ClearChannelInt(ADC1, ADC_CH0);
}
```

2.3.2 斜坡发生器

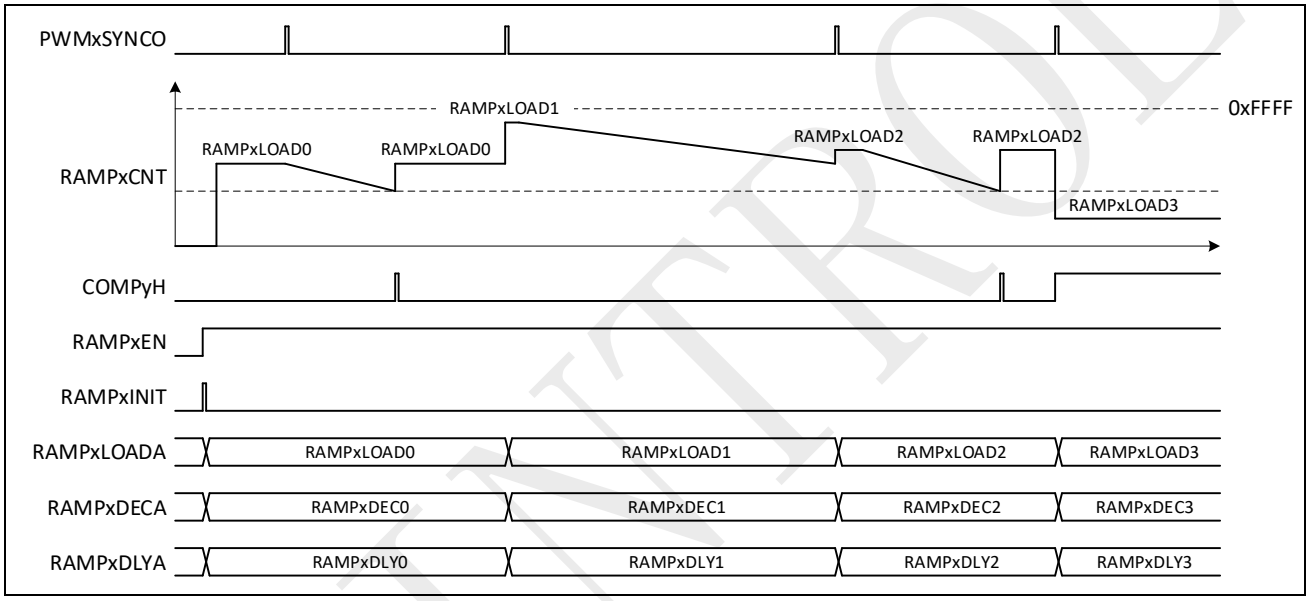
2.3.2.1 功能需求

芯片输出斜坡电压。

2.3.2.2 功能实现

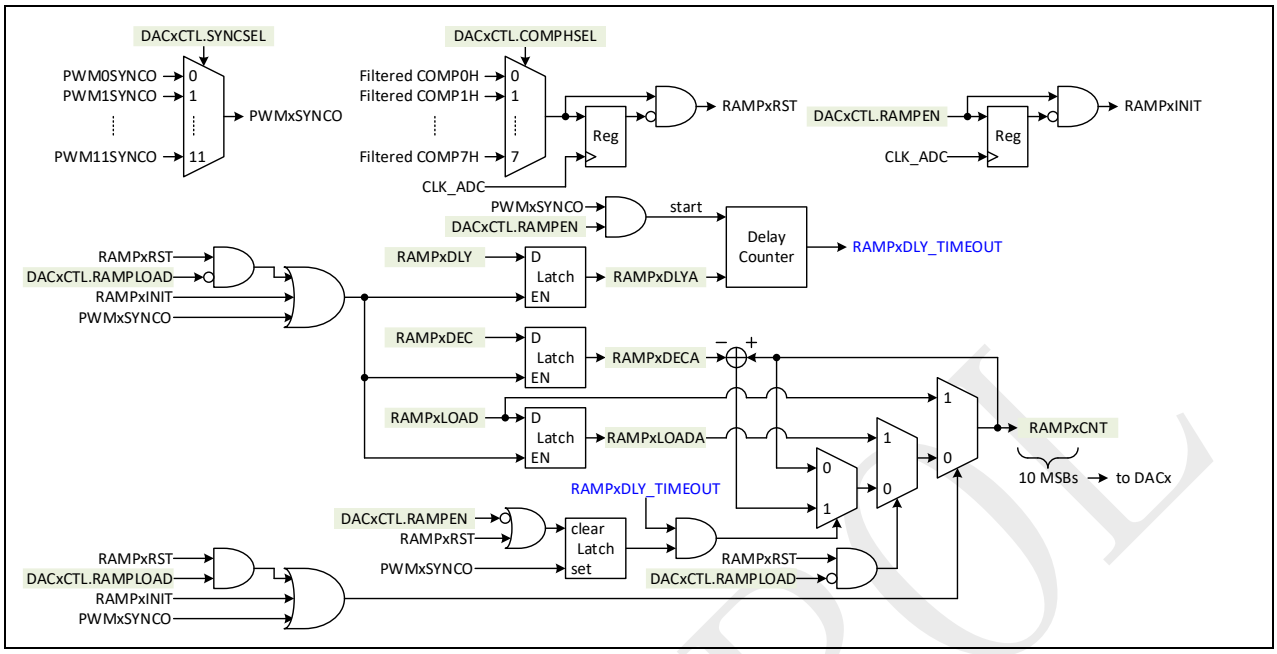
PWM，DAC，COMPH 可以共同构成斜坡发生器，产生斜坡电压 RAMPxCNT，如图 2-2 所示。

图 2-2：斜坡发生器



- 通过 DACxCTL.RAMPEN 使能斜坡发生器后，触发初始化加载影子寄存器 RAMPxDLY、RAMPxDEC、RAMPxLOAD 到对应的有效值寄存器 RAMPxDLYA、RAMPxDECA、RAMPxLOADA；
- RAMPxCNT 计数器维持初值不变；
- 在 PWMxSYNCO 事件触发后，延时计数器从其初值 RAMPxDLYA 开始递减计数，直到超时；
- 延时结束后，RAMPxCNT 计数器以 RAMPxDECA 为步长递减计数；
- 由于 RAMPxCNT 的高 10 位用作 DACx 的码值，因此其输出的电压递减，即 COMPyH 的参考电压递减；
- 随着 RAMPxCNT 的减小，COMPyH 输出在某个时候会变高，这使得 RAMPxCNT 回到初值，在新的 PWMxSYNCO 事件到来后重复延时等待和递减计数的过程；

图 2-3：根据 PWM 同步输出信号递减电平



示例演示了通过定时器产生 SYNCI 信号并通过 SYNCO 触发 DAC 递减计数，从而产生斜坡电压。

实验时，COMPH 对应正端电压必须大于 0V，否则负端电压无法降落到正端电压之下。

Example Code

```
#include "spc2188.h"
#include <stdio.h>

int main(void)
{
    CLOCK_InitWithRCO(240000000U);

    Delay_Init();

    /*
    * Initial the UART
    */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 3000mV  */
    COMP_SetDACVoltage(DAC0, 3000);

    /* DAC Buffer Initial, only DACBUF0_FROM_DAC0 ~ DACBUF0_FROM_DAC3 connect to
DACBUF0*/
    COMP_SetDACBuffer0Input(DACBUF0_FROM_DAC0);
    COMP_EnableDACBuffer(DACBUF0);
```

Example Code

```
    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO(DACBUF0);

    CLOCK_EnableModule(PWM_MODULE);

    /* Connect the input sync signal with output */
    PWM_SetSyncOutEvent(PWM0, SYNCO_SYNCI_AND_FRCSYNC);

    /* Enable PWM SYNC by the signal coming from TIMER1 */
    PWM_EnableSyncFromTIMER1(INC_PWM0 );

    printf("PWMCFG->TMR1SYNCIEN is %x\n", PWMCFG->TMR1SYNCIEN);

    /* Init TIMER1 */
    TIMER_Init(TIMER1, 1);

    /* Set TIMER1 generate SYNC to PWM when count down to 0 */
    TIMER_EnablePWMSync(TIMER1);

    /* Open Global INT for TIMER1 */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    COMP_SetDACRampDelay(DAC0, 200);
    COMP_SetDACRampDecrement(DAC0, 1);
    COMP_EnableDACRamp(DAC0);

    /* Set PGAP as input to comparator, set Too High threshold is 3000mV, filter
window is 200ns */
    COMP_Init(COMP0_H, COMP0_FROM_ANA_IN6, COMP0_REF_DAC0_DAC1, 3000, 200);

    printf("COMP->DAC0CODE is %d\n", COMP->DAC0CODE);

    COMP_SetDACRampReloadValue(DAC0, (COMP->DAC0CODE << 6U));

    while (1)
    {

    }
}


void TIMER1_IRQHandler()
{
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

3 SPC1125 系列

3.1 特性

SPC1128 包含 3 个 10 比特 DAC。DAC 的输出可由下面表达式计算：

$$\text{VOUT} = \frac{AVDD}{2^{10}} * CODE \quad (AVDD\text{典型值为 } 3.3V)$$

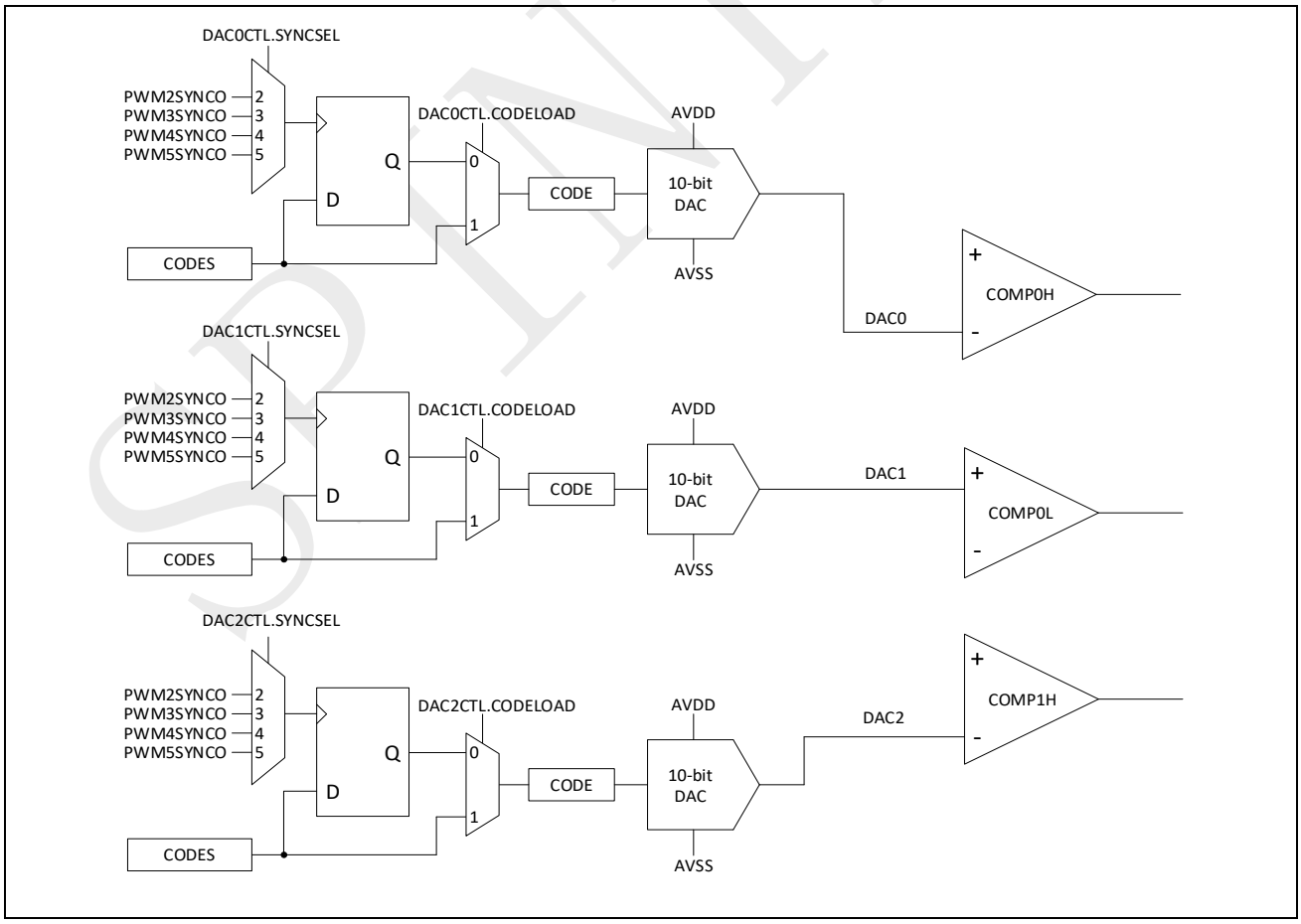
3.2 功能描述

DAC 结构框图如图 3-1 所示。

- 当 DACxCTL.CODELOAD = 0 时，开启影子模式。此时写入 DACxCODE 寄存器的码值并不会立即生效，而是要等到由 DACxCTL.SYNCSEL 所选定的 PWMySYNCO 事件发生时才会更新 DACxCODEA。
- 当 DACxCTL.CODELOAD = 1 时，开启即时模式。此时写入 DACxCODE 寄存器的码值立即同步更新到 DACxCODEA。

产生的模拟电压既可直接传给 COMP。

图 3-1：DAC 示意图



3.3 功能实例

3.3.1 输出恒定电压

3.3.1.1 功能需求

通过 DAC 产生 1.65V 电压，并送入 ADC 测量。

3.3.1.2 功能实现

表 3-1：代码路径

MCU 产品类型	代码路径
SPC1125，SPC1128	0_Examples\DAC_to_ADC