MICROCHIP

TB3210

DAC 入门

简介

作者：Microchip Technology Inc.的 Victor Berzan

数模转换器（Digital-to-Analog Converter，DAC）可将写入数据寄存器（DAC.DATA）的数字值转换为模拟电压。输出可连接至物理引脚，也可在内部使用。转换范围介于 GND 和所选内部参考电压（V_{REF}）之间。

本技术简介说明了 DAC 如何在 tinyAVR® 1 系列单片机（MCU）上工作。其中涉及以下用例：

- **生成恒定模拟信号：**
  初始化 DAC，设置参考电压，设置 DAC 以输出特定的恒压。
- **生成正弦波信号：**
  初始化 DAC，设置参考电压，循环输出正弦波样本。
- **使用 ADC 在内部读取 DAC：**
  初始化 DAC 和 ADC，设置参考电压，设置 ADC 以读取 DAC，递增 DAC 输出，以及对每次步进用 ADC 读取该输出。
- **将 DAC 用作 AC 的负输入：**
  初始化 DAC 和 AC，将 AC 设置为从 DAC 获取其负输入，将 DAC 设置为输出特定值。AC 会将其正输入引脚上的电压与 DAC 电压进行比较，并根据比较结果将输出引脚设置为高电平或低电平。

注： 代码示例是在 ATtiny817 Xplained Mini（ATTINY817-XMINI）上开发的。

目录

1. 相关器件

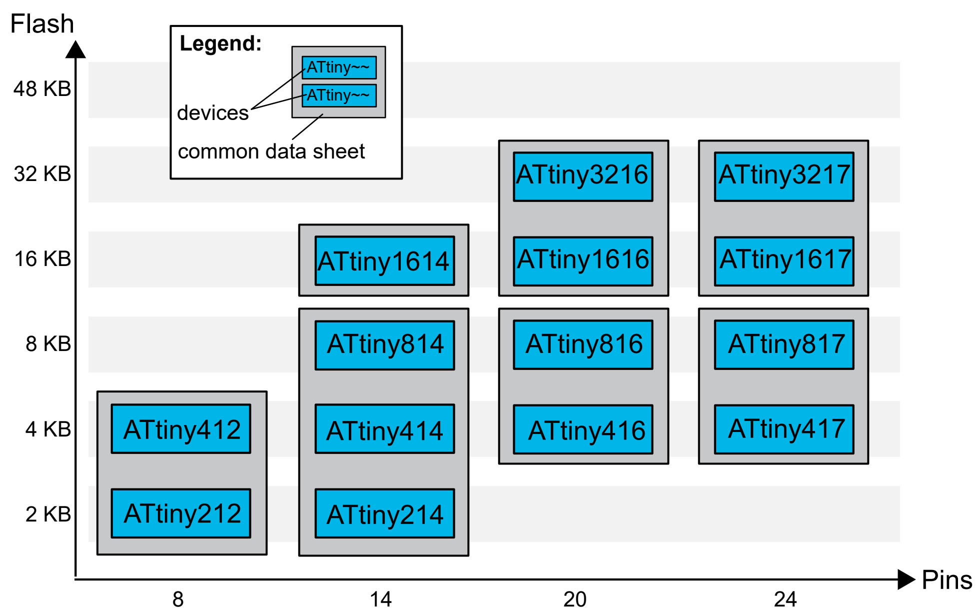
本章列出了文中涉及的相关器件。

1.1 tinyAVR® 1 系列

下图所示为 tinyAVR 1 系列器件，注明了不同的引脚数与存储器大小：

- 垂直向上移植无需修改代码，因为这些器件引脚兼容并可提供相同或更多的功能。而向下移植可能需要修改代码，因为某些外设的可用实例数减少。
- 水平向左移植会减少引脚数，进而减少可用的功能。

图 1-1. tinyAVR® 1 系列概览



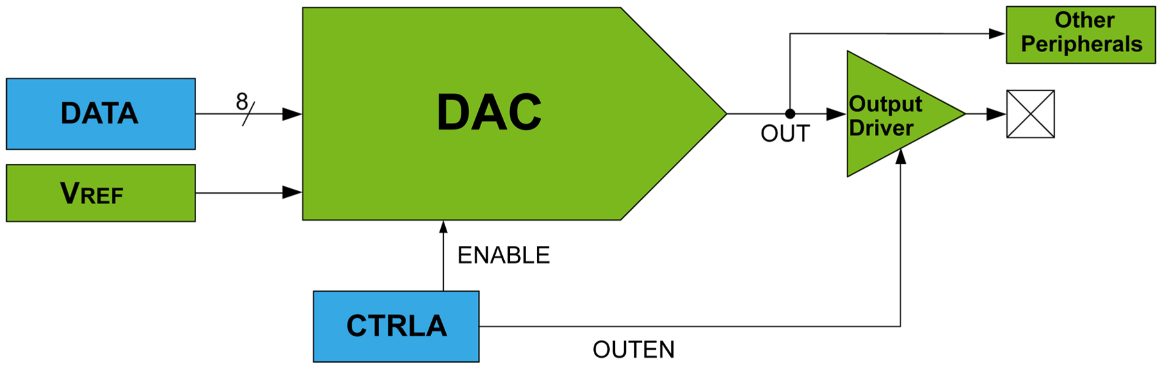
具有不同闪存大小的器件通常也具有不同的 SRAM 和 EEPROM。

2. 概述

DAC 具有一个 8 位电阻串型 DAC，每秒能够转换 350,000 个样本（350 ksps）。DAC 将内部参考电压（V_{REF}）用作转换上限。

DAC 具有一个具备高驱动能力的连续时间输出，能够驱动 5 kΩ 和/或 30 pF 的负载。可通过写入数据转换寄存器（DAC.DATA）来从应用程序启动 DAC 转换。

图 2-1. DAC 框图



3. 生成恒定模拟信号

可使用 DAC 生成恒定模拟信号。它将参考电压（V_{REF}）外设的输出用作正参考。DAC 输出范围为 0V 至 $\frac{255 \times V_{REF}}{256}$

可从预定义值列表中选择 V_{REF}：

图 3-1. CTRLA.DAC0REFSEL 值和相关的 V_{REF} 电压

Bit	7	6	5	4	3	2	1	0
		ADC0REFSEL[2:0]				DAC0REFSEL[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

bits 2:0 DAC0REFSEL[2:0]: DAC0 and AC0 Reference Select bits
These bits select the reference voltage for the DAC0 and AC0.

Value	Description
0x0	0.55V
0x1	1.1V
0x2	2.5V
0x3	4.3V
0x4	1.5V
other	Reserved

示例中选择了 4.34V，以便获得最大的变化范围：

```
VREF_CTRLA |= VREF_DAC0REFSEL_4V34_gc;
VREF_CTRLB |= VREF_DAC0REFEN_bm;
```

允许 VREF 后，建议采用 25 μs 的延迟。

图 3-2. 内部参考电压特性

Symbol	Description	Min.	Typ.	Max.	Unit
t_{start}	Start-up time	-	25	-	μs
$V_{DDINT055V}$	Power supply voltage range for INT055V	1.8	-	5.5	V
$V_{DDINT11V}$	Power supply voltage range for INT11V	1.8	-	5.5	
$V_{DDINT15V}$	Power supply voltage range for INT15V	1.9	-	5.5	
$V_{DDINT25V}$	Power supply voltage range for INT25V	2.9	-	5.5	
$V_{DDINT43V}$	Power supply voltage range for INT43V	4.75	-	5.5	

```
_delay_us(25);
```

DAC 输出可由其他外设在内部使用，也可连接至输出引脚。对于 ATtiny817，DAC 输出连接至引脚 A6（请参见图 3-3）。

图 **3-3. PORT** 函数复用

VQFN 24-pin	Pin Name (1,2)	Other/Special	ADC0	PTC(3)	AC0	DAC0	USART0
23	PA0	$\overline{\text{RESET}}$ UPDI	AIN0				
24	PA1		AIN1				TXD
1	PA2	EVOUT0	AIN2				RxD
2	PA3	EXTCLK	AIN3				XCK
3	GND						
4	VDD						
5	PA4		AIN4	X0/Y0			XDIR
6	PA5		AIN5	X1/Y1	OUT		
7	PA6		AIN6	X2/Y2	AINN0	OUT	
8	PA7		AIN7	X3/Y3	AINP0		
9	PB7						

DAC 输出引脚需要禁止数字输入缓冲器和上拉电阻，以减少负载。

```
PORTA.PIN6CTRL &= ~PORT_ISC_gm;
PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;
```

在此特定示例中，DAC 所需的输出为 0.5425V。为获得此输出，应采用以下公式：
$DATA = V_{OUT} \times 256/V_{REF} = 0.5425V \times 256/4.34V = 32 = 0x20$

在初始化时写入 DAC0.DATA 寄存器是可选操作；然而，使 DAC 从一开始就输出特定电压可能会很有用。

```
DAC0.DATA = 0x20;
```

使能DAC和输出缓冲器，并允许它们在待机模式下运行：

```
DAC0.CTRLA = DAC_ENABLE_bm | DAC_OUTEN_bm | DAC_RUNSTDBY_bm;
```

重要： 如果允许在待机模式下运行，则当 MCU 处于待机休眠模式时，DAC 将继续运行。

启动转换
如果使能 DAC（DAC.CTRLA 中 ENABLE = 1），则一旦写入数据寄存器（DAC.DATA），就会开始转换。

```
DAC0.DATA = 0x20;
```

转换后，只要 DAC 在运行，输出值就会保持为 $\frac{\mathrm{DATA} \times \mathrm{V_{REF}}}{256}$，直到下次转换。如果 V_{REF} 选择发生任何改变，DAC 输出也会立即改变（如果使能并运行 DAC）。

View Code Example on GitHub
Click to browse repository

提示： 如需获取完整的代码示例，也可参见附录部分。

4. 生成正弦波信号

可使用 DAC 生成正弦波信号。

初始化 V_{REF} 和 DAC 之后，只需写入 DAC0.DATA 寄存器即可更改输出值。

要生成正弦波，请先计算正弦波样本并将其存储在缓冲器中：

```
for(uint16_t i = 0; i < SINE_WAVE_STEPS; i++)
{
    sineWave[i] = SINE_DC_OFFSET + SINE_AMPLITUDE * sin(i * M_2PI / SINE_WAVE_STEPS);
}
```

然后将其以特定的时间间隔循环写入 DAC0.DATA：

```
while (1)
{
    DAC0.DATA = sineWave[i++]; i
    = i % SINE_WAVE_STEPS;
    _delay_us(STEP_DELAY_MICROS);
}
```

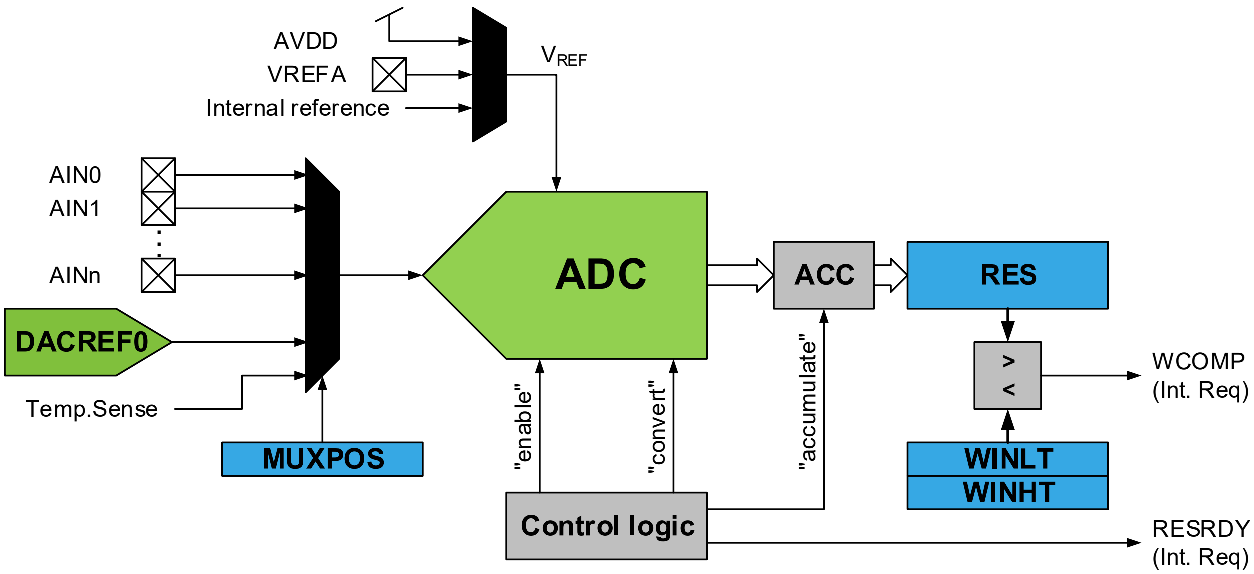
View Code Example on GitHub
Click to browse repository

提示： 如需获取完整的代码示例，也可参见附录部分。

5. 使用 ADC 在内部读取 DAC

如图 5-1 所示，ADC 输入可以连接至 DAC 的输出。

图 5-1. 模数转换器框图



如果将 ADC 与来自 V_{REF} 的正参考配合使用，则还必须配置相应的参考选择：

图 5-2. VREF.CTRLA 寄存器

Bit	7	6	5	4	3	2	1	0
		ADC0REFSEL[2:0]				DAC0REFSEL[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

bits 6:4 ADC0REFSEL[2:0]: ADC0 Reference Select bits
These bits select the reference voltage for the ADC0.

Value	Description
0x0	0.55V
0x1	1.1V
0x2	2.5V
0x3	4.3V
0x4	1.5V
other	Reserved

bits 2:0 DAC0REFSEL[2:0]: DAC0 and AC0 Reference Select bits
These bits select the reference voltage for the DAC0 and AC0.

Value	Description
0x0	0.55V
0x1	1.1V
0x2	2.5V
0x3	4.3V
0x4	1.5V
other	Reserved

现在，完整的 V_{REF} `init` 将如下所示：

```
VREF_CTRLA |= VREF_DAC0REFSEL_4V34_gc;
VREF_CTRLB |= VREF_DAC0REFEN_bm;
VREF_CTRLA |= VREF_ADC0REFSEL_4V34_gc;
VREF_CTRLB |= VREF_ADC0REFEN_bm;
_delay_us(VREF_STARTUP_MICROS);
```

然后必须初始化 ADC：

```
ADC0.CTRLC = ADC_PRESC_DIV4_gc
            | ADC_REFSEL_INTREF_gc
            | ADC_SAMPCAP_bm;

ADC0.CTRLA = ADC_ENABLE_bm
            | ADC_RESSEL_10BIT_gc;
```

为了使用 ADC 读取 DAC，必须将 ADC 的 MUXPOS 寄存器设置为 0x1C。

图 5-3. MUXPOS DAC 输出选择

Bit	7	6	5	4	3	2	1	0
				MUXPOS[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

bits 4:0 MUXPOS[4:0]: MUXPOS bits
This bit field selects which single-ended analog input is connected to the ADC. If these bits are changed during a conversion, the change will not take effect until this conversion is complete.

Value	Name	Description
`0x0B`	AIN11	ADC input pin 11
`0x1C`	DAC0	DAC0
`0x1D`	INTREF	Internal reference (from VREF peripheral)
`0x1E`	TEMPSENSE	Temperature sensor
`0x1F`	GND	0V (GND)
`Other`	-	Reserved

```
ADC0.MUXPOS = ADC_MUXPOS_DAC0_gc;
```

通过将相应的位写入 ADC0.COMMAND 来启动 ADC 转换：

```
ADC0.COMMAND = ADC_STCONV_bm;
```

转换完成后，硬件将置位 ADC0.INTFLAGS 中的 `ADC_RESRDY_bm`。

```
while (!(ADC0.INTFLAGS & ADC_RESRDY_bm))
{
    ;
}
```

用户必须通过写入“1”来清除标志：

```
ADC0.INTFLAGS = ADC_RESRDY_bm;
```

可从 ADC0.RES 读取 ADC 数据。

可将 DAC 输出设置为不同的值，并使用 ADC 在 while 循环中读取：

```
while (1)
{
    adcVal = ADC0_read();
    dacVal++;
    DAC0_setVal(dacVal);
}
```

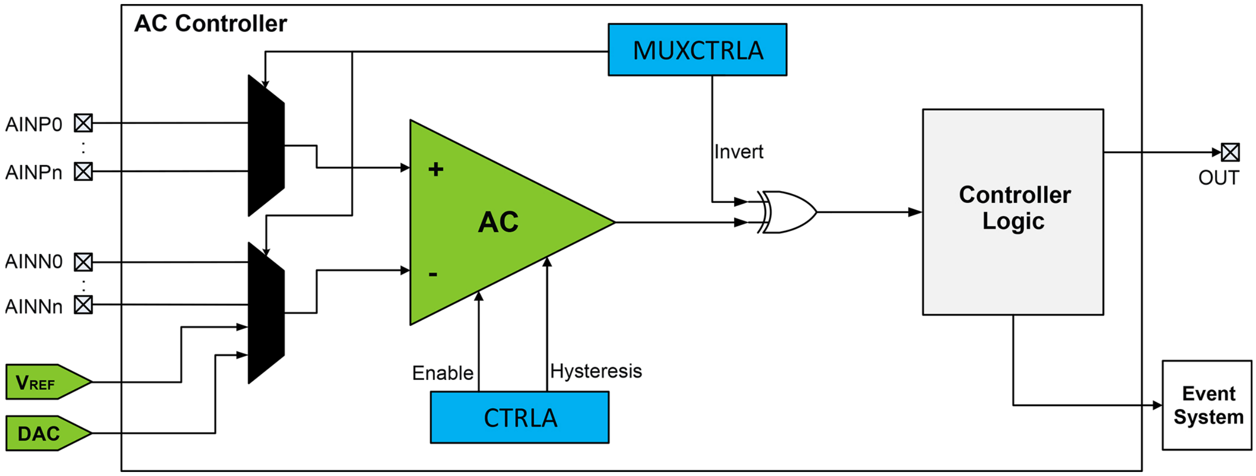
View Code Example on GitHub
Click to browse repository

提示： 如需获取完整的代码示例，也可参见附录部分。

6. 将 DAC 用作 AC 的负输入

如图 6-1 所示，AC 可将 DAC 输出用作负输入。

图 6-1. 模拟比较器框图



在此特定示例中，用于 DAC 的 V_{REF} 设置为 1.5V：

```
VREF.CTRLA |= VREF_DAC0REFSEL_1V5_gc;
VREF.CTRLB |= VREF_DAC0REFEN_bm;
_delay_us(VREF_STARTUP_MICROS);
```

AC 设置为从 DAC 获取负输入，从 PA7 引脚获取正输入。

VQFN 24-pin	Pin Name (1,2)	Other/Special	ADC0	PTC(3)	AC0	DAC0	USART0
23	PA0	RESET UPDI	AIN0				
24	PA1		AIN1				TXD
1	PA2	EVOUT0	AIN2				RxD
2	PA3	EXTCLK	AIN3				XCK
3	GND						
4	VDD						
5	PA4		AIN4	X0/Y0			XDIR
6	PA5		AIN5	X1/Y1	OUT		
7	PA6		AIN6	X2/Y2	AINN0	OUT	
8	PA7		AIN7	X3/Y3	AINP0		
9	PB7						

```
AC0.MUXCTRLA = AC_MUXNEG_DAC_gc
             | AC_MUXPOS_PIN0_gc;
```

AC 的输出位于引脚 PA5 上。

VQFN 24-pin	Pin Name [1,2]	Other/Special	ADC0	PTC[3]	AC0	DAC0	USART0
23	PA0	RESET UPDI	AIN0				
24	PA1		AIN1				TXD
1	PA2	EVOUT0	AIN2				RxD
2	PA3	EXTCLK	AIN3				XCK
3	GND						
4	VDD						
5	PA4		AIN4	X0/Y0			XDIR
6	PA5		AIN5	X1/Y1	OUT		
7	PA6		AIN6	X2/Y2	AINN0	OUT	
8	PA7		AIN7	X3/Y3	AINP0		
9	PB7						

```
AC0.CTRLA = AC_ENABLE_bm
            | AC_OUTEN_bm |
            AC_RUNSTDBY_bm |
            AC_LPMODE_bm;
```

对于此特定示例，初始化 V_{REF}、DAC 和 AC 之后，将 DAC 输出设置为 1.4V：

```
DAC0_setVal(DAC_DATA_1V4);
```

AC 会将 PA7 的电压与 1.4V 进行比较，并根据比较结果设置其输出。

View Code Example on GitHub
Click to browse repository

提示： 如需获取完整的代码示例，也可参见附录部分。

7. 参考资料

1. ATtiny417/817——采用独立于内核的外设和 picoPower® 技术的 AVR® 单片机（DS40001901）
2. ATtiny817 Xplained Mini 评估板

8. 附录

例 8-1. 生成恒定模拟信号的代码示例

```
/* 3.33 MHz（延迟所需）*/
#define F_CPU                          (3333333UL)
/* DAC 值 */
#define DAC_EXAMPLE_VALUE              (0x20)
/* VREF 启动时间 */
#define VREF_STARTUP_MICROS            (25)

#include <avr/io.h>
#include <util/delay.h>

void           VREF_init(void);
void           DAC0_init(void);
void DAC0_setVal(uint8_t val);

void VREF_init(void)
{
    /* 参考电压为 4.34V */
    VREF.CTRLA |= VREF_DAC0REFSEL_4V34_gc;
    /* 是否使能参考 DAC0/AC0: 使能 */
    VREF.CTRLB |= VREF_DAC0REFEN_bm;
    /* 等待 VREF 启动时间 */
    _delay_us(VREF_STARTUP_MICROS);
}

void DAC0_init(void)
{
    /* 禁止数字输入缓冲器 */
    PORTA.PIN6CTRL &= ~PORT_ISC_gm;
    PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
    /* 禁止上拉电阻 */
    PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    /* 使能 DAC和输出缓冲器，并允许它们在待机模式下运行 */
    DAC0.CTRLA = DAC_ENABLE_bm | DAC_OUTEN_bm | DAC_RUNSTDBY_bm;
}

void DAC0_setVal(uint8_t val)
{
    DAC0.DATA = val;
}

int main(void)
{
    VREF_init();

    DAC0_init();

    DAC0_setVal(DAC_EXAMPLE_VALUE);

    while (1)
    {
        ;
    }
}
```

例 8-2. 生成正弦波信号的代码示例

```
/* 3.33 MHz（延迟所需）*/
#define F_CPU                          (3333333UL)
/* VREF 启动时间 */
#define VREF_STARTUP_MICROS            (25)
/* 正弦波周期的步进数 */
#define SINE_WAVE_STEPS                (100)
/* 正弦波幅度 */
#define SINE_AMPLITUDE                 (127)
```

```
    /* 正弦波 DC 偏移 */
    #define SINE_DC_OFFSET                         (128)
    /* 2*PI */
    #define M_2PI                                  (2 * M_PI)
    /* 正弦波频率 */
    #define OUTPUT_FREQ                            (50)
    /* 合成循环的步进延迟 */
    #define STEP_DELAY_MICROS                      ((1000000 / OUTPUT_FREQ) /
SINE_WAVE_STEPS)

    #include <avr/io.h>
    #include <util/delay.h>
    #include <math.h>

    /* 用于存储正弦波样本的缓冲器 */
    uint8_t sineWave[SINE_WAVE_STEPS];

    void sineWaveInit(void);
    void VREF_init(void);
    void DAC0_init(void);
    void DAC0_setVal(uint8_t val);

    void sineWaveInit(void)
    {
        for(uint16_t i = 0; i < SINE_WAVE_STEPS; i++)
        {
            sineWave[i] = SINE_DC_OFFSET + SINE_AMPLITUDE * sin(i * M_2PI /
 SINE_WAVE_STEPS);
        }
    }

    void VREF_init(void)
    {
        /* 参考电压为 4.34V */
        VREF.CTRLA |= VREF_DAC0REFSEL_4V34_gc;
        /* 是否使能参考DAC0/AC0: 使能 */
        VREF.CTRLB |= VREF_DAC0REFEN_bm;
        /* 等待 VREF 启动时间 */
        _delay_us(VREF_STARTUP_MICROS);
    }

    void DAC0_init(void)
    {
        /* 禁止数字输入缓冲器 */
        PORTA.PIN6CTRL &= ~PORT_ISC_gm;
        PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
        /* 禁止上拉电阻 */
        PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

        /* 默认值 */
        DAC0.DATA = SINE_DC_OFFSET;
        /* 使能DAC和输出缓冲器，并允许它们在待机模式下运行 */
        DAC0.CTRLA = DAC_ENABLE_bm | DAC_OUTEN_bm | DAC_RUNSTDBY_bm;
    }

    void DAC0_setVal(uint8_t val)
    {
        DAC0.DATA = val;
    }

    int main(void)
    {
        uint16_t i = 0;
        VREF_init();
        DAC0_init();

        sineWaveInit();

        while (1)
        {
            DAC0_setVal(sineWave[i++]); i
            = i % SINE_WAVE_STEPS;
            _delay_us(STEP_DELAY_MICROS);
```

```
        }
    }
```

例 8-3. 使用 ADC 在内部读取 DAC 的代码示例

```
/* 3.33 MHz（延迟所需）*/
#define F_CPU                          (3333333UL)
/* VREF 启动时间 */
#define VREF_STARTUP_MICROS            (25)

#include <avr/io.h>
#include <util/delay.h>

void VREF_init(void);
void DAC0_init(void);
void ADC0_init(void);
uint16_t ADC0_read(void);
void DAC0_setVal(uint8_t val);

void VREF_init(void)
{
    /* 参考电压为 4.34V */
    VREF_CTRLA |= VREF_DAC0REFSEL_4V34_gc;
    /* 是否使能参考DAC0/AC0：使能 */
    VREF_CTRLB |= VREF_DAC0REFEN_bm;
    /* 参考电压为 4.34V */
    VREF_CTRLA |= VREF_ADC0REFSEL_4V34_gc;
    /* 是否使能参考ADC0：使能 */
    VREF_CTRLB |= VREF_ADC0REFEN_bm;
    /* 等待 VREF 启动时间 */
    _delay_us(VREF_STARTUP_MICROS);
}

void DAC0_init(void)
{
    /* 使能 DAC */
    DAC0.CTRLA = DAC_ENABLE_bm;
}

void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc        /* CLK_PER 除以 4 */
               | ADC_REFSEL_INTREF_gc     /* 参考 VDD */
               ADC_SAMPCAP_bm;            /* 样本电容选择：使能*/

       ADC0.CTRLA = ADC_ENABLE_bm         /*  是否使能 ADC：使能*/
               | ADC_RESSEL_10BIT_gc;     /* 10 位模式 */

    /* 选择 ADC 通道 */
    ADC0.MUXPOS = ADC_MUXPOS_DAC0_gc;
}
uint16_t ADC0_read(void)
{
    /* 启动 ADC 转换 */
    ADC0.COMMAND = ADC_STCONV_bm;

    /* 等待 ADC 转换完成 */
    while ( !(ADC0.INTFLAGS & ADC_RESRDY_bm) )
    {
        ;
    }

    /* 通过写入 1 清除中断标志：*/
    ADC0.INTFLAGS = ADC_RESRDY_bm;

    return ADC0.RES;
}

void DAC0_setVal(uint8_t val)
{
```

```
        DAC0.DATA = val;
    }

    int main(void)
    {
        uint8_t dacVal = 0;
        uint16_t adcVal = 0;

        VREF_init();
        DAC0_init();
        ADC0_init();

        /* 等待 VREF 启动时间 */
        _delay_us(VREF_STARTUP_MICROS);

        DAC0_setVal(dacVal);

        while (1)
        {
            adcVal = ADC0_read();

            /* 使用 adcVal 执行操作 */

            dacVal++;
            DAC0_setVal(dacVal);
        }
    }
```

例 8-4. 将 DAC 用作 AC 的负输入的代码示例

```
    /* 3.33 MHz（延迟所需）*/
    #define F_CPU                          (3333333UL)
    /* 输出 1.4V（VREF = 1.5V） */
    #define DAC_DATA_1V4                   (239)
    /* VREF 启动时间 */
    #define VREF_STARTUP_MICROS            (25)

    #include <avr/io.h>
    #include <util/delay.h>

    void VREF_init(void);
    void DAC0_setVal(uint8_t val);
    void DAC0_init(void);
    void AC0_init(void);

    void VREF_init(void)
    {
        /* 参考电压为 1.5V */
        VREF.CTRLA |= VREF_DAC0REFSEL_1V5_gc;
        /* 是否使能参考 DAC0/AC0: 使能 */
        VREF.CTRLB |= VREF_DAC0REFEN_bm;
        /* 等待 VREF 启动时间 */
        _delay_us(VREF_STARTUP_MICROS);
    }

    void DAC0_setVal(uint8_t val)
    {
        DAC0.DATA = val;
    }

    void DAC0_init(void)
    {
        /* 禁止数字输入缓冲器 */
        PORTA.PIN6CTRL &= ~PORT_ISC_gm;
        PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
        /* 禁止上拉电阻 */
        PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

        /* 使 能 DAC和输出缓冲器，并允许它们在待机模式下运行 */
        DAC0.CTRLA = DAC_ENABLE_bm | DAC_OUTEN_bm | DAC_RUNSTDBY_bm;
    }
```

```
void AC0_init(void)
{
    /* 来自 DAC0 的负输入 */
    /* 来自引脚 PA7 的正输入 */
    AC0.MUXCTRLA = AC_MUXNEG_DAC_gc
                 | AC_MUXPOS_PIN0_gc;

    /* 使能 PA5 上输出，低功耗模式 */
    AC0.CTRLA = AC_ENABLE_bm
                 | AC_OUTEN_bm
                 | AC_RUNSTDBY_bm
                 | AC_LPMODE_bm;
}

int main(void)
{
    /* 分压器 -> PA7 上的 VDD/2 输入 */
    /* PA5 的 LED 上的 AC 输出 */
    /* 当电池电压低于 2.8V（PA7 电压低于 1.4V）时 LED 熄灭 */

    VREF_init();
    DAC0_init();
    AC0_init();

    /* 输出 1.4V（@ VREF = 1.5V）*/
    DAC0_setVal(DAC_DATA_1V4);

    while (1)
    {
        ;
    }
}
```

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

商标

ISBN：978-1-5224-8001-3

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲

公司总部
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
电话：480-792-7200
传真：480-792-7277
技术支持：
www.microchip.com/support
网址：
www.microchip.com

亚特兰大
德卢斯，佐治亚州
电话：678-957-9614
传真：678-957-1455

奥斯汀，德克萨斯州
电话：512-257-3370

波士顿
韦斯特伯鲁，马萨诸塞州
电话：774-760-0087
传真：774-760-0088

芝加哥
艾塔斯卡，伊利诺伊州
电话：630-285-0071
传真：630-285-0075

达拉斯
阿迪森，德克萨斯州
电话：972-818-7423
传真：972-818-2924

底特律
诺维，密歇根州
电话：248-848-4000

休斯顿，德克萨斯州
电话：281-894-5983

印第安纳波利斯
诺布尔斯维尔，印第安纳州
电话：317-773-8323
传真：317-773-5453
电话：317-536-2380

洛杉矶
米慎维荷，加利福尼亚州
电话：949-462-9523
传真：949-462-9608
电话：951-273-7800

罗利，北卡罗来纳州
电话：919-844-7510

纽约，纽约州
电话：631-435-6000

圣何塞，加利福尼亚州
电话：408-735-9110
电话：408-436-4270

加拿大 - 多伦多
电话：905-695-1980
传真：905-695-2078

亚太地区

澳大利亚 - 悉尼
电话：61-2-9868-6733

中国 - 北京
电话：86-10-8569-7000

中国 - 成都
电话：86-28-8665-5511

中国 - 重庆
电话：86-23-8980-9588

中国 - 东莞
电话：86-769-8702-9880

中国 - 广州
电话：86-20-8755-8029

中国 - 杭州
电话：86-571-8792-8115

中国 - 香港特别行政区
电话：852-2943-5100

中国 - 南京
电话：86-25-8473-2460

中国 - 青岛
电话：86-532-8502-7355

中国 - 上海
电话：86-21-3326-8000

中国 - 沈阳
电话：86-24-2334-2829

中国 - 深圳
电话：86-755-8864-2200

中国 - 苏州
电话：86-186-6233-1526

中国 - 武汉
电话：86-27-5980-5300

中国 - 西安
电话：86-29-8833-7252

中国 - 厦门
电话：86-592-2388138

中国 - 珠海
电话：86-756-3210040

亚太地区

印度 - 班加罗尔
电话：91-80-3090-4444

印度 - 新德里
电话：91-11-4160-8631

印度 - 浦那
电话：91-20-4121-0141

日本 - 大阪
电话：81-6-6152-7160

日本 - 东京
电话：81-3-6880-3770

韩国 - 大邱
电话：82-53-744-4301

韩国 - 首尔
电话：82-2-554-7200

马来西亚 - 吉隆坡
电话：60-3-7651-7906

马来西亚 - 槟榔屿
电话：60-4-227-8870

菲律宾 - 马尼拉
电话：63-2-634-9065

新加坡
电话：65-6334-8870

台湾地区 - 新竹
电话：886-3-577-8366

台湾地区 - 高雄
电话：886-7-213-7830

台湾地区 - 台北
电话：886-2-2508-8600

泰国 - 曼谷
电话：66-2-694-1351

越南 - 胡志明市
电话：84-28-5448-2100

欧洲

奥地利 - 韦尔斯
电话：43-7242-2244-39
传真：43-7242-2244-393

丹麦 - 哥本哈根
电话：45-4485-5910
传真：45-4485-2829

芬兰 - 埃斯波
电话：358-9-4520-820

法国 - 巴黎
电话：33-1-69-53-63-20
传真：33-1-69-30-90-79

德国 - 加兴
电话：49-8931-9700

德国 - 哈恩
电话：49-2129-3766400

德国 - 海尔布隆
电话：49-7131-72400

德国 - 卡尔斯鲁厄
电话：49-721-625370

德国 - 慕尼黑
电话：49-89-627-144-0
传真：49-89-627-144-44

德国 - 罗森海姆
电话：49-8031-354-560

以色列 - 若那那市
电话：972-9-744-7705

意大利 - 米兰
电话：39-0331-742611
传真：39-0331-466781

意大利 - 帕多瓦
电话：39-049-7625286

荷兰 - 德卢内市
电话：31-416-690399
传真：31-416-690340

挪威 - 特隆赫姆
电话：47-72884388

波兰 - 华沙
电话：48-22-3325737

罗马尼亚 - 布加勒斯特
电话：40-21-407-87-50

西班牙 - 马德里
电话：34-91-708-08-90
传真：34-91-708-08-91

瑞典 - 哥德堡
电话：46-31-704-60-40

瑞典 - 斯德哥尔摩
电话：46-8-5090-4654

英国 - 沃金厄姆
电话：44-118-921-5800
传真：44-118-921-5820