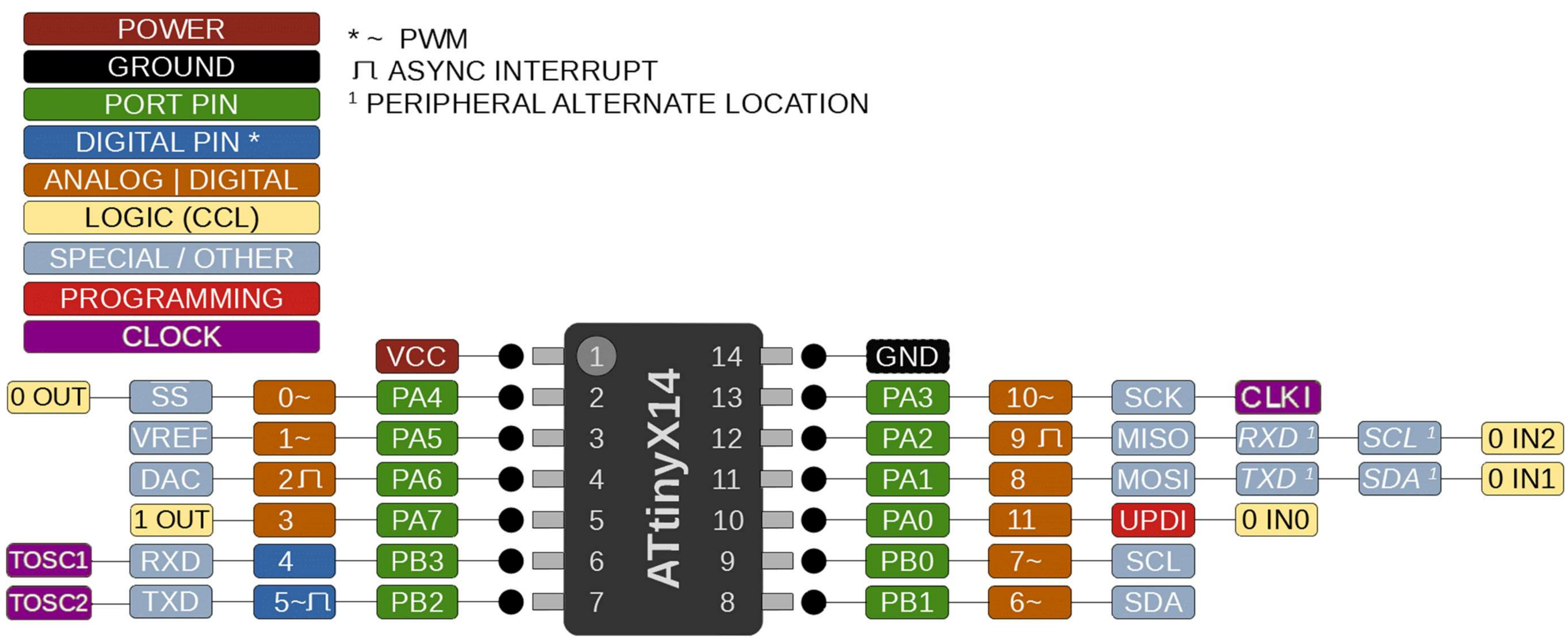

POWER
GROUND
PORT PIN
DIGITAL PIN *
ANALOG | DIGITAL
LOGIC (CCL)
SPECIAL / OTHER
PROGRAMMING
CLOCK
* ~ PWM
⎍ ASYNC INTERRUPT
1 PERIPHERAL ALTERNATE LOCATION
ATtinyX14
VCC 1
0 OUT SS 0~ PA4 2
VREF 1~ PA5 3
DAC 2⎍ PA6 4
1 OUT 3 PA7 5
TOSC1 RXD 4 PB3 6
TOSC2 TXD 5~⎍ PB2 7
14 GND
13 PA3 10~ SCK CLKI
12 PA2 9 ⎍ MISO RXD 1 SCL 1 0 IN2
11 PA1 8 MOSI TXD 1 SDA 1 0 IN1
10 PA0 11 UPDI 0 IN0
9 PB0 7~ SCL
8 PB1 6~ SDA

31. DAC - Digital-to-Analog Converter

31.1 Features

- 8-bit Resolution
- Up to 350 ksps Conversion Rate
- High Drive Capabilities (DAC0)
- Functioning as Input to Analog Comparator (AC) or Analog-to-Digital Converter (ADC)

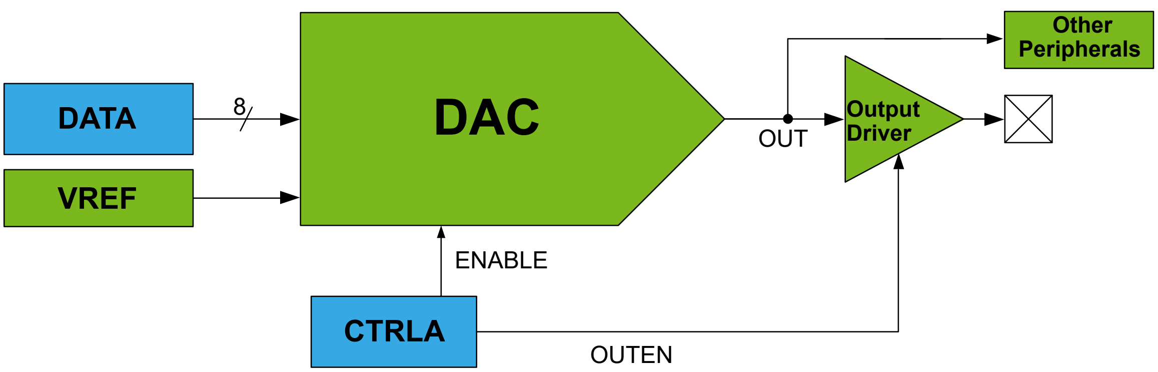
31.2 Overview

The Digital-to-Analog Converter (DAC) converts a digital value written to the Data (DAC.DATA) register to an analog voltage. The conversion range is between GND and the selected reference voltage.

The DAC features an 8-bit resistor-string type DAC, capable of converting 350,000 samples per second (350 ksps). The DAC uses the internal Voltage Reference (VREF) as the upper limit for conversion and has one continuous time output with high drive capabilities, which can drive a 5 kΩ and/or 30 pF load. The DAC conversion can be started from the application by writing to the Data Conversion registers.

31.2.1 Block Diagram

Figure 31-1. DAC Block Diagram



Note: Only DAC0 has an output driver for an external pin.

31.2.2 Signal Description

Signal	Description	Type
OUT	DAC output	Analog

31.2.3 System Dependencies

To use this peripheral, other parts of the system must be configured correctly, as described below.

Table 31-1. DAC System Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	No	-
Events	No	-

...........continued

Dependency	Applicable	Peripheral
Debug	Yes	UPDI

31.2.3.1 Clocks

This peripheral depends on the peripheral clock.

31.2.3.2 I/O Lines and Connections

Using the I/O lines of the peripheral requires configuration of the I/O pins.

Table 31-2. I/O Lines

Instance	Signal	I/O Line	Peripheral Function
DAC0	OUT	PA6	A

The DAC0 has one analog output pin (OUT) that must be configured before it can be used.

A DAC is also internally connected to the AC and the ADC. To use this internal OUT as input, both output and input must be configured in their respective registers.

31.2.3.3 Events

Not applicable.

31.2.3.4 Interrupts

Not applicable.

31.2.3.5 Debug Operation

This peripheral is unaffected by entering Debug mode.

If the peripheral is configured to require periodic service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

31.3 Functional Description

31.3.1 Initialization

To operate the DAC, the following steps are required:

1. Select the DAC reference voltage in the Voltage Reference (VREF) peripheral by writing the DAC and AC Reference Selection (DACnREFSEL) bits in the Control x (VREF.CTRLx) register.
2. The conversion range is between GND and the selected reference voltage.
3. Configure the further usage of the DAC output:
   3.1. Configure an internal peripheral (e.g., AC, ADC) to use the DAC output. Refer to the according peripheral's documentation.
   3.2. Enable the output to a pin by writing a '1' to the Output Enable (OUTEN) bit in the Control A (DAC.CTRLA) register. This requires a configuration of the Port peripheral.

   For DAC0, either one or both options are valid. Other instances of the DAC only support internal signaling.
4. Write an initial digital value to the Data (DAC.DATA) register.
5. Enable the DAC by writing a '1' to the ENABLE bit in the DAC.CTRLA register.

31.3.2 Operation

31.3.2.1 Enabling, Disabling and Resetting

The DAC is enabled by writing a '1' to the ENABLE bit in the Control A (DACn.CTRLA) register and disabled by writing a '0' to this bit.

The OUT output to a pin is enabled by writing the Output Enable (OUTEN) bit in the DACn.CTRLA register.

31.3.2.2 Starting a Conversion

When the DAC is enabled (ENABLE = '1' in DACn.CTRLA), a conversion starts as soon as the Data (DACn.DATA) register is written.

When the DAC is disabled (ENABLE = '0' in DACn.CTRLA), writing to the DACn.DATA register does not trigger a conversion. Instead, the conversion starts on writing a '1' to the ENABLE bit in the DACn.CTRLA register.

31.3.2.3 DAC as Source For Internal Peripherals

The analog output of the DAC is internally connected to both the AC and the ADC and is available to these peripherals when the DAC is enabled (ENABLE = '1' in DAC.CTRLA). When the DAC analog output is only being used internally, it is not necessary to enable the pin output driver (i.e., OUTEN = '0' in DAC.CTRLA is acceptable).

31.3.3 Sleep Mode Operation

If the Run in Standby (RUNSTDBY) bit in the Control A (DAC.CTRLA) register is written to '1' and CLK_PER is available, the DAC will continue to operate in Standby sleep mode. If the RUNSTDBY bit is '0', the DAC will stop the conversion in Standby sleep mode.

If the conversion is stopped in Standby sleep mode, the DAC and the output buffer are disabled to reduce power consumption. When the device is exiting Standby sleep mode, the DAC and the output buffer (if configured by OUTEN = '1' in DAC.CTRLA) are enabled again. Therefore, certain start-up time is required before a new conversion is initiated.

In Power-Down sleep mode, the DAC and output buffer are disabled to reduce the power consumption.

31.3.4 Configuration Change Protection

Not applicable.

31.4 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTDBY	OUTEN						ENABLE
0x01	DATA	7:0	DATA[7:0]							

31.5 Register Description

31.5.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	OUTEN						ENABLE
Access	R/W	R/W						R/W
Reset	0	0						0

Bit 7 – RUNSTDBY Run in Standby Mode
If this bit is written to '1', the DAC or output buffer will not automatically be disabled when the device is entering Standby sleep mode.

Bit 6 – OUTEN Output Buffer Enable
Writing a '1' to this bit enables the output buffer and sends the OUT signal to a pin.

Bit 0 – ENABLE DAC Enable
Writing a '1' to this bit enables the DAC.

31.5.2 DATA

Name: DATA
Offset: 0x01
Reset: 0x00
Property: -

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

Bits 7:0 – DATA[7:0] Data
This bit field contains the digital data, which will be converted to an analog voltage.

TB3210

Getting Started with Digital-to-Analog Converter (DAC)

Introduction

Author: Victor Berzan, Microchip Technology Inc.

The Digital-to-Analog Converter (DAC) converts a digital value written to the Data (DAC.DATA) register to an analog voltage. The output can be connected to a physical pin or used internally. The conversion range is between GND and the selected internal voltage reference (V_{REF}), provided by the Voltage Reference (VREF) peripheral module.

This technical brief describes how the DAC works on tinyAVR® 1-series and AVR® DA microcontrollers (MCUs). It covers the following use cases:

- **Generating Constant Analog Signal:**
  Illustrates how to initialize the DAC, set the voltage reference, set the DAC to output a specific constant voltage
- **Generating Sine Wave Signal:**
  Initializes the DAC, sets the voltage reference, updates the DAC output inside the infinite loop to generate sine wave samples.
- **Reading the DAC Internally with the ADC:**
  Shows how to initialize the DAC and ADC, set the voltage reference, configure the ADC to read the DAC output values. The DAC output voltage is incremented each step, and then it is read using the ADC.
- **Using DAC as Negative Input for AC:**
  Initializes the DAC and AC, configures the AC to use the negative input provided by the DAC, configures the DAC output value. The AC will compare the voltage on its positive input pin with the DAC voltage, and set the output pin to high or low, according to the compare result.

Note: For the first three use cases described in this document, there are two code examples: One bare metal developed on ATtiny817, and one generated with MPLAB® Code Configurator (MCC) developed on AVR128DA48. For the last use case, there is one bare metal code example developed on ATtiny817.

View the ATtiny817 Code Examples on GitHub
Click to browse repository

View the AVR128DA48 Code Examples on GitHub
Click to browse repository

Table of Contents

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration on tinyAVR® 1-series devices may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 1-1. tinyAVR® 1-series Overview

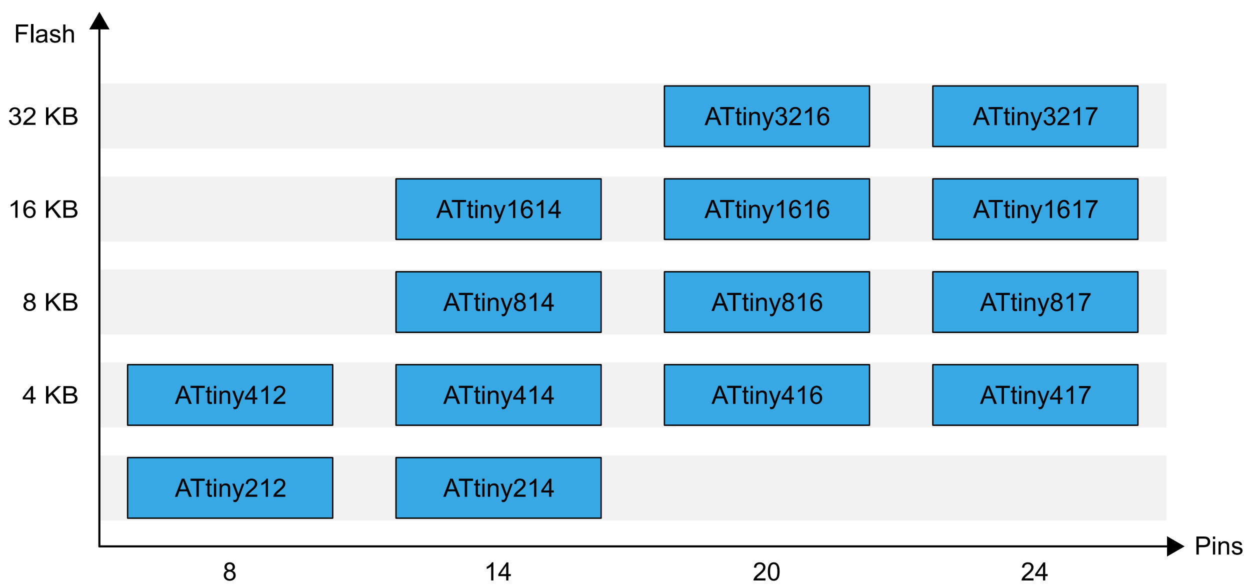
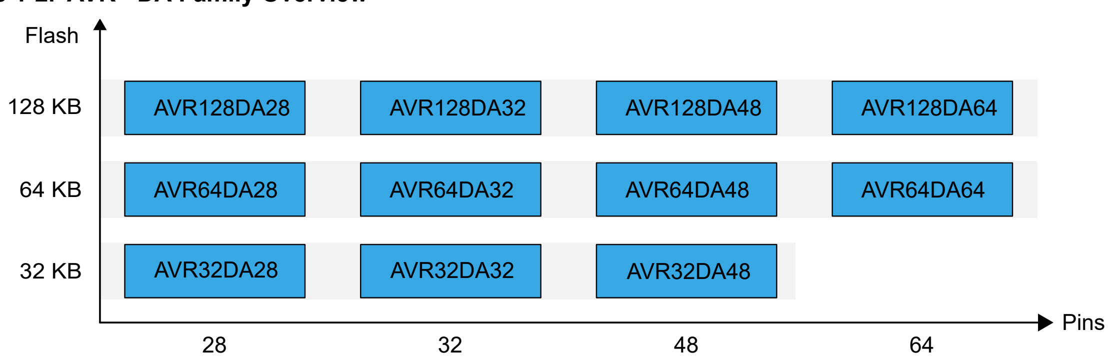


Figure 1-2. AVR® DA Family Overview



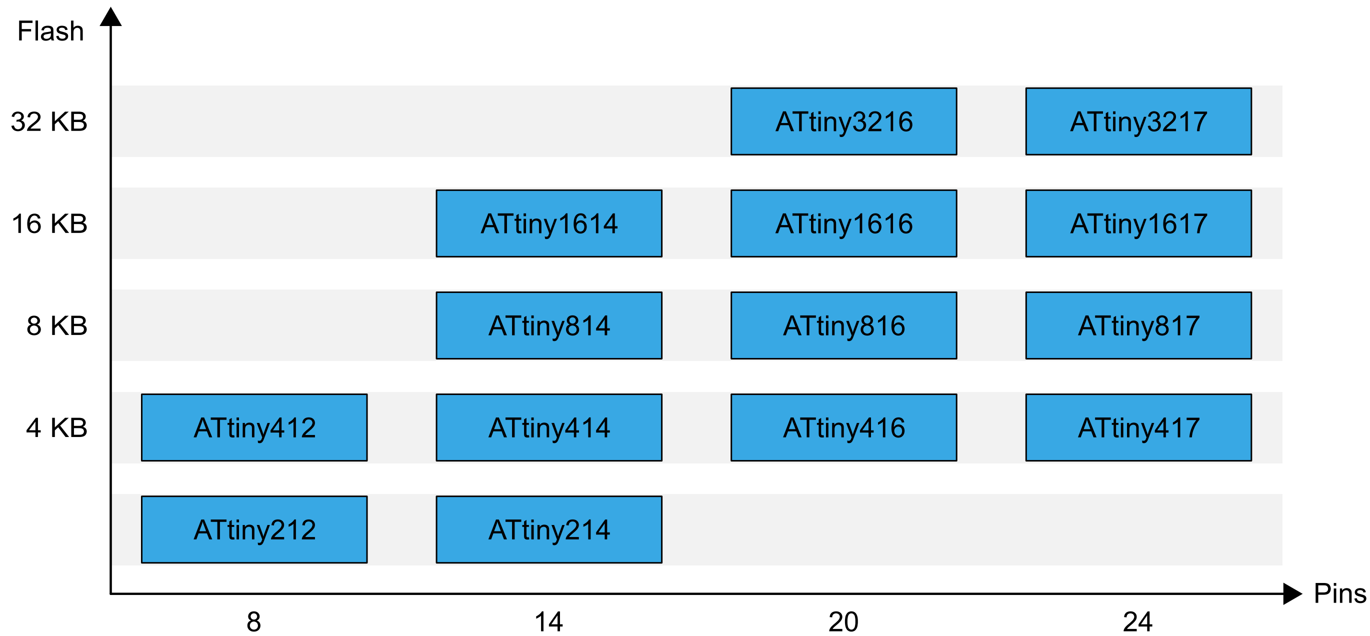
1.1 tinyAVR® 1-series

The following figure shows the tinyAVR 1-series devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration may require code modification due to fewer available instances of some peripherals.

- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1-3. tinyAVR® 1-series Overview



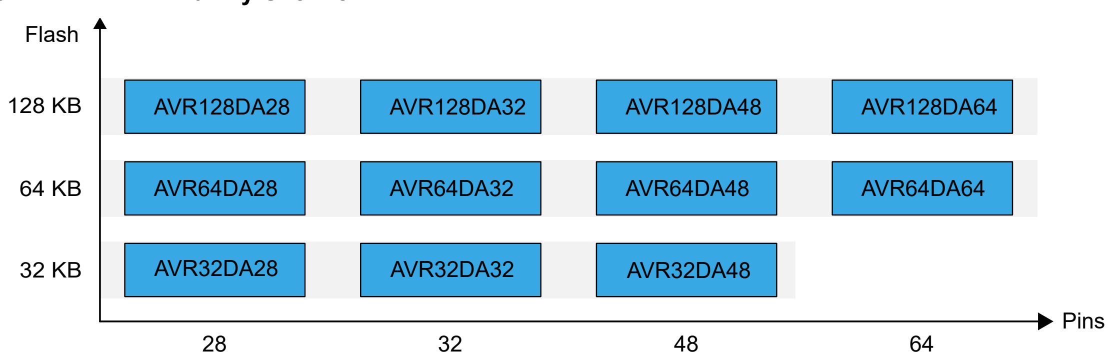
Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

1.2 AVR® DA Family Overview

The figure below shows the AVR® DA devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible
- Horizontal migration to the left reduces the pin count, and therefore, the available features

Figure 1-4. AVR® DA Family Overview

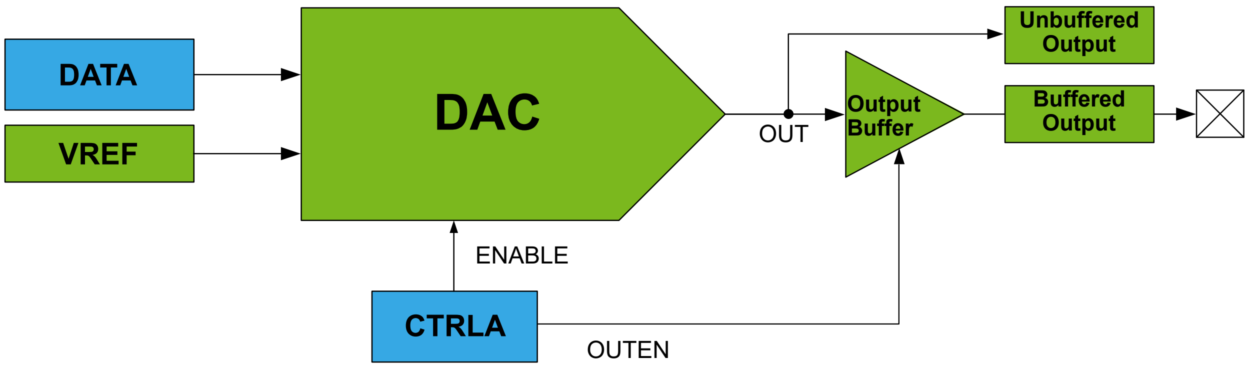


Devices with different Flash memory sizes typically also have different SRAM.

2. Overview

The DAC converts the digital value written to the Data (DACn.DATA) register into an analog voltage that will be available on the DAC output. The conversion range is between GND and the selected voltage reference in the Voltage Reference (VREF) peripheral module. The DAC has one continuous-time output with high drive capabilities. The DAC conversion can be started from the application by writing to the Data (DACn.DATA) register.

Figure 2-1. DAC Block Diagram



3. Generating Constant Analog Signal

The DAC can be used to generate a constant analog signal. The Voltage Reference (VREF) peripheral module is used to provide a voltage reference for the DAC. The DAC output ranges from 0V to $\frac{255 \times V_{REF}}{256}$.

The voltage reference V_{REF} can be selected from a list of predefined values:

Figure 3-1. CTRLA.DAC0REFSEL Values and Associated V_{REF} Voltages

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

The 4.34V was selected, to have the widest variation range:

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

A 25 μs delay is recommended after configuring the VREF and enabling the voltage reference output. To implement this delay, the `VREF_STARTUP_MICROS` macro definition is used.

Figure 3-2. Internal Voltage Reference Characteristics

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(VREF_STARTUP_MICROS);
```

The DAC output can be used internally by other peripherals, or it can be linked to an output pin. For the ATtiny817, the DAC output is connected to pin A6 (see Figure 3-3).

Figure 3-3. PORT Function Multiplexing

VQFN 24-pin	Pin Name	Other/Special	ADC0	PTC	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						

The DAC output pin needs to have the digital input buffer and the pull-up resistor disabled to reduce its load.

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

The desired output for the DAC in this particular example is 0.5425V. To achieve it, the following equation is applied:

$$DATA = V_{OUT} \times 256/V_{REF} = 0.5425V \times 256/4.34V = 32 = 0x20$$

Writing to the DAC0.DATA register at initialization is optional; however, it may be useful to make the DAC output a specific voltage from the beginning.

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

Enabling DAC, Output Buffer and Run in Standby sleep mode:

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

Important: If Run in Standby sleep mode is enabled, the DAC will continue to run when the MCU is in Standby sleep mode.

Starting a Conversion

When the DAC is enabled (ENABLE = `1` in DAC.CTRLA), a conversion starts as soon as the Data (DAC.DATA) register is written.

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

After conversion, the output keeps its value of $\frac{DATA \times V_{REF}}{256}$ until the next conversion, as long as the DAC is running. Any change in V_{REF} selection will immediately change the DAC output (if enabled and running).

Tip: The full code example is also available in the Appendix section.

View the ATtiny817 Code Example on GitHub
Click to browse repository

An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

View the AVR128DA48 Code Example on GitHub
Click to browse repository

4. Generating Sine Wave Signal

The DAC can be used to obtain a sine wave signal by generating a defined number of discrete samples of the desired analog signal. Each sample is characterized by a voltage level and by a time duration. This mechanism provides an approximation of the desired sine wave signal.

In this application, the number of steps used is 100. To obtain a signal frequency of 50 Hz (translated into a period of 20 ms), the time duration between two signal samples will be (1000000/Desired Frequency)/Number of Steps [µs] = (1000000/50 Hz)/100 (steps) [µs] = 200 µs.

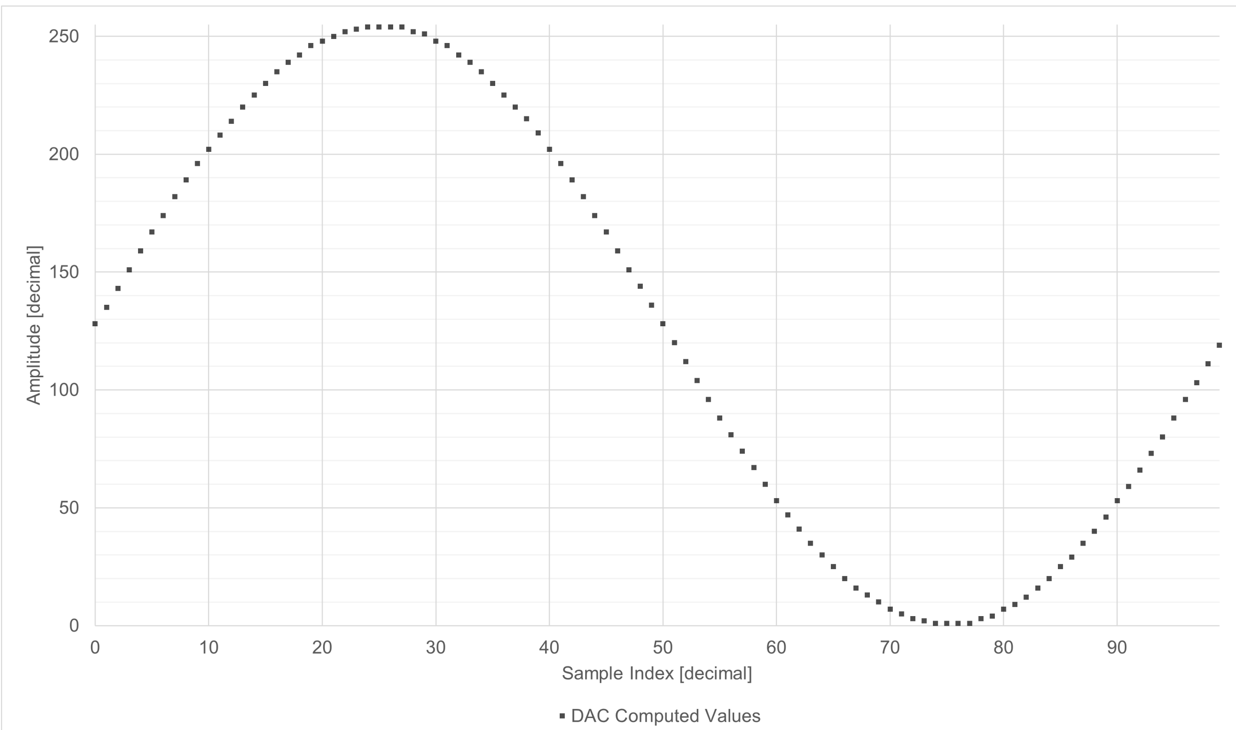
To generate the “steps” that approximate the sine wave, the integer values to be written to the DAC0.DATA register will be computed using the formula below:

$$sineWave[i] = DC\,Offset + A \cdot sin\left(\frac{\mathrm{i} \cdot 2\pi}{Number\ of\ Steps}\right)$$

,where *i* is the sample index and takes values from 0 to Number of Steps, *A* is the signal amplitude (the maximum possible value is the DAC reference voltage), and the *Number of Steps* represents the number of samples used to construct the sine wave signal.

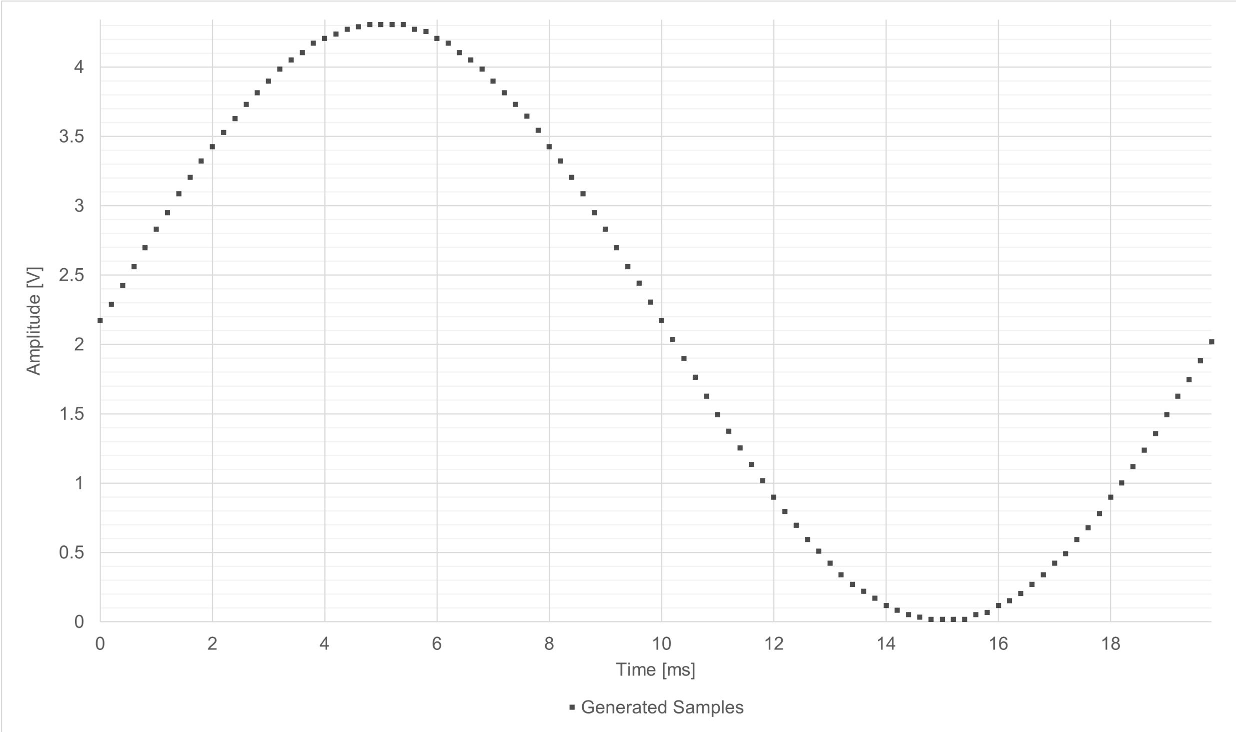
To generate a sine wave signal between 0V and 4.34V, the DAC must convert integers ranged between 0 and 255. Therefore, because the DAC will only generate positive voltage values, the DC Offset will be 128 (this is also the mean value of the sine wave signal). To obtain the maximum of 4.34V, the amplitude must be 127. The data to be written to the DAC0.DATA register is presented on the graph below.

Figure 4-1. DAC Input Digital Values



The ideal voltage values that will be available on the DAC output are presented on the graph below.

Figure 4-2. DAC Output Analog Values



To avoid spending time inside the infinite loop by computing the signal samples in real-time, the samples are computed at the beginning of the application and stored in a Look-up Table. The function implementation for this computation is presented below.

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

The data is then written to the DAC0.DATA register inside the infinite loop using a 200 μs delay, as described above.

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

Tip: The full code example is also available in the Appendix section.

View the ATtiny817 Code Example on GitHub
Click to browse repository

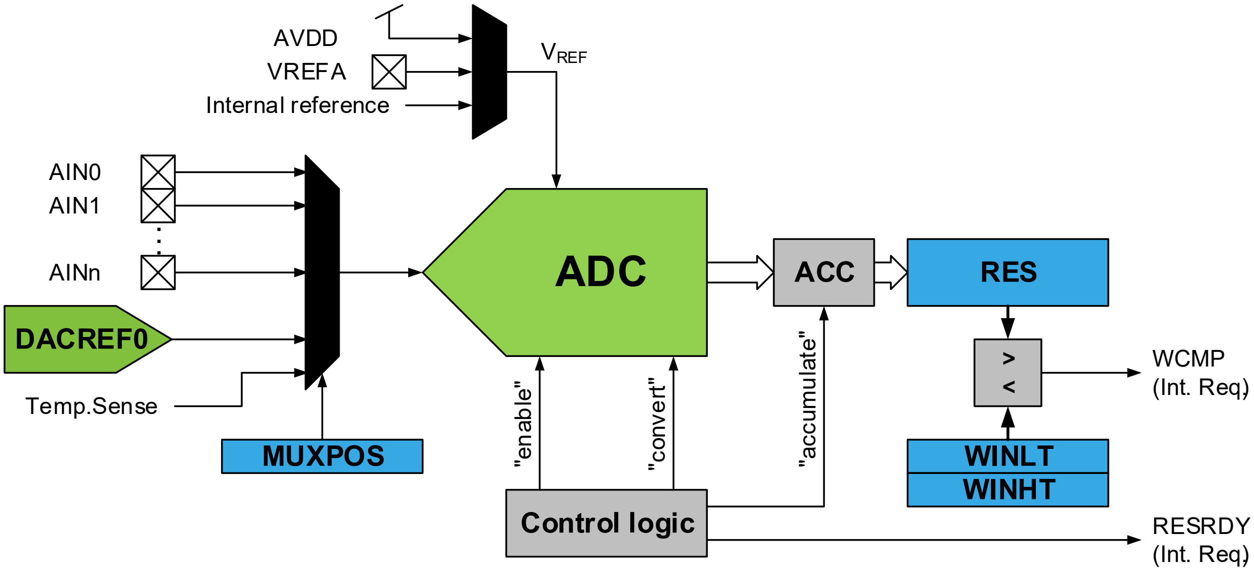
An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

View the AVR128DA48 Code Example on GitHub
Click to browse repository

5. Reading the DAC Internally with the ADC

As seen in Figure 5-1, the ADC input can be connected to the DAC output. Therefore, the DAC conversion result (a digital value) can be transmitted directly to the ADC. Then, the ADC will convert the analog result provided by the DAC again, obtaining a digital result.

Figure 5-1. Analog-to-Digital Converter Block Diagram



The VREF is configured to provide the voltage reference for the ADC0 and DAC0, as presented below.

Figure 5-2. VREF.CTRLA Register

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

The VREF peripheral module is initialized as illustrated in the code snippet below.

```
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);
```

Then, the ADC is initialized as presented below.

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

ADC0.CTRLA = ADC_ENABLE_bm
           | ADC_RESSEL_10BIT_gc;
```

To read the DAC with the ADC, the MUXPOS register of the ADC must be set to `0x1C`.

Figure 5-3. MUXPOS DAC Output Selection

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;
```

The ADC conversion is started by writing the corresponding bit to the ADC0.COMMAND.

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

When the conversion is done, the Result Ready (RESRDY) flag in ADC0.INTFLAGS will be set by the hardware.

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

The flag must be cleared by software, as presented below.

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

The ADC conversion result is available in the ADC0.RES register.

The DAC output can be set to different values and read using the ADC inside the infinite loop.

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

Tip: The full code example is also available in the Appendix section.

View the ATtiny817 Code Example on GitHub
Click to browse repository

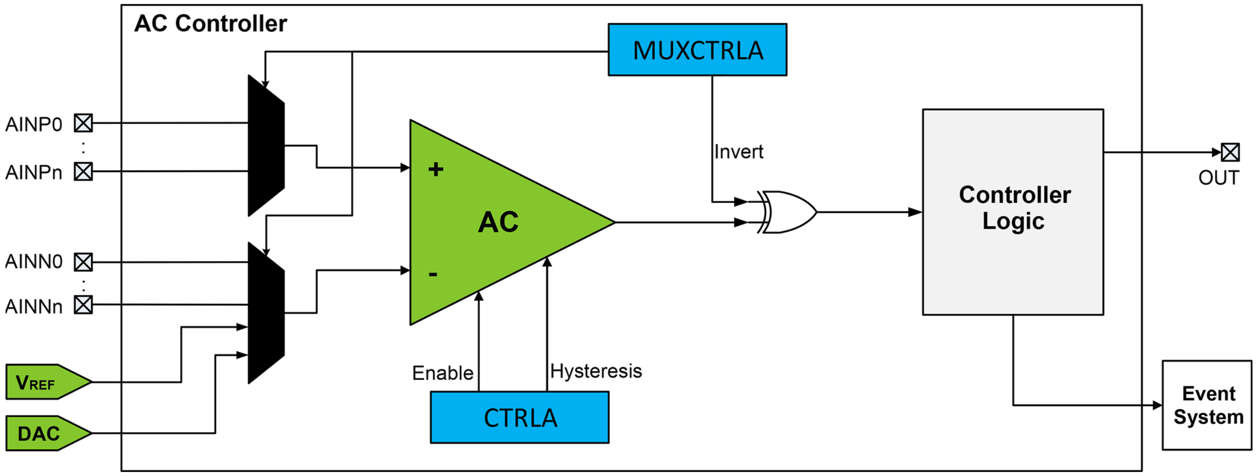
An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

View the AVR128DA48 Code Example on GitHub
Click to browse repository

6. Using DAC as Negative Input for AC

As seen in Figure 6-1, the AC can use the DAC output as the negative input. Therefore, the user can generate the AC negative voltage reference by simply writing the corresponding digital value to the DATA register of the DAC peripheral module. Then, the DAC will generate the conversion result, a constant voltage, which will be directly connected to the AC input.

Figure 6-1. Analog Comparator Block Diagram



In this code example, the voltage reference for the DAC module is configured at 1.5V.

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

The AC is configured to receive its negative input from DAC and positive input from pin PA7.

VQFN 24-pin	Pin Name	Other/Special	ADC0	PTC	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;
```

Pin PA5 is configured as the AC output.

VQFN 24-pin	Pin Name	Other/Special	ADC0	PTC	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						

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

After initializing the VREF, DAC, and AC, the DAC output is set to 1.4V.

```
DAC0_setVal(DAC_DATA_1V4);
```

The AC will compare the voltage from PA7 to 1.4V and set its output according to the comparison result.

Tip: The full code example is also available in the Appendix section.

View the ATtiny817 Code Example on GitHub
Click to browse repository

This application can be implemented on AVR128DA48 by using the internal reference voltage generator (DACREF) feature of the AC. There is no need to configure the DAC to provide a negative input for the AC on this architecture.

7. References

1. ATtiny417/817 - AVR® Microcontroller with Core Independent Peripherals and picoPower® Technology (DS40001901)
2. ATtiny817 Xplained Mini board
3. ATtiny817 product page: www.microchip.com/wwwproducts/en/ATTINY817
4. AVR128DA48 product page: www.microchip.com/wwwproducts/en/AVR128DA48
5. AVR128DA48 Curiosity Nano Evaluation Kit product page: https://www.microchip.com/Developmenttools/ProductDetails/DM164151
6. AVR128DA28/32/48/64 Data Sheet
7. Getting Started with the AVR® DA Family

8. Appendix

Example 8-1. Generating Constant Analog Signal Code Example

```
/* 3.33 MHz (needed for delay) */
#define F_CPU                           (3333333UL)
/* DAC value */
#define DAC_EXAMPLE_VALUE               (0x20)
/* VREF Startup time */
#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)
{
    /* Voltage reference at 4.34V */
    VREF.CTRLA |= VREF_DAC0REFSEL_4V34_gc;
    /* DAC0/AC0 reference enable: enabled */
    VREF.CTRLB |= VREF_DAC0REFEN_bm;
    /* Wait VREF start-up time */
    _delay_us(VREF_STARTUP_MICROS);
}

void DAC0_init(void)
{
    /* Disable digital input buffer */
    PORTA.PIN6CTRL &= ~PORT_ISC_gm;
    PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
    /* Disable pull-up resistor */
    PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    /* Enable DAC, Output Buffer, Run in Standby */
    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)
    {
        ;
    }
}
```

Example 8-2. Generating Sine Wave Signal Code Example

```
    /* 3.33 MHz (needed for delay) */
    #define F_CPU                                (3333333UL)
    /* VREF Startup time */
    #define VREF_STARTUP_MICROS                  (25)
    /* Number of steps for a sine wave period */
    #define SINE_WAVE_STEPS                      (100)
    /* Sine wave amplitude */
    #define SINE_AMPLITUDE                       (127)
    /* Sine wave DC offset */
    #define SINE_DC_OFFSET                       (128)
    /* 2*PI */
    #define M_2PI                                (2 * M_PI)
    /* Frequency of the sine wave */
    #define OUTPUT_FREQ                          (50)
    /* Step delay for the synthesis loop */
    #define STEP_DELAY_MICROS                    ((1000000 / OUTPUT_FREQ) /
SINE_WAVE_STEPS)

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

    /* Buffer to store the sine wave samples */
    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)
    {
        /* Voltage reference at 4.34V */
        VREF.CTRLA |= VREF_DAC0REFSEL_4V34_gc;
        /* DAC0/AC0 reference enable: enabled */
        VREF.CTRLB |= VREF_DAC0REFEN_bm;
        /* Wait VREF start-up time */
        _delay_us(VREF_STARTUP_MICROS);
    }

    void DAC0_init(void)
    {
        /* Disable digital input buffer */
        PORTA.PIN6CTRL &= ~PORT_ISC_gm;
        PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
        /* Disable pull-up resistor */
        PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

        /* default value */
        DAC0.DATA = SINE_DC_OFFSET;
        /* Enable DAC, Output Buffer, Run in Standby */
        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);
        }
    }
```

Example 8-3. Reading DAC Internally With ADC Code Example

```
    /* 3.33 MHz (needed for delay) */
    #define F_CPU                           (3333333UL)
    /* VREF Startup time */
    #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)
    {
        /* Voltage reference at 4.34V */
        VREF_CTRLA |= VREF_DAC0REFSEL_4V34_gc;
        /* DAC0/AC0 reference enable: enabled */
        VREF_CTRLB |= VREF_DAC0REFEN_bm;
        /* Voltage reference at 4.34V */
        VREF_CTRLA |= VREF_ADC0REFSEL_4V34_gc;
        /* ADC0 reference enable: enabled */
        VREF_CTRLB |= VREF_ADC0REFEN_bm;
        /* Wait VREF start-up time */
        _delay_us(VREF_STARTUP_MICROS);
    }

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

    void ADC0_init(void)
    {
        ADC0.CTRLC = ADC_PRESC_DIV4_gc      /* CLK_PER divided by 4 */
                   | ADC_REFSEL_INTREF_gc   /* VDD reference */
                   | ADC_SAMPCAP_bm;        /* Sample Capacitance Selection:
enabled */

        ADC0.CTRLA = ADC_ENABLE_bm          /* ADC Enable: enabled */
                   | ADC_RESSEL_10BIT_gc;   /* 10-bit mode */

        /* Select ADC channel */
        ADC0.MUXPOS = ADC_MUXPOS_DAC0_gc;
    }

    uint16_t ADC0_read(void)
    {
        /* Start ADC conversion */
        ADC0.COMMAND = ADC_STCONV_bm;

        /* Wait until ADC conversion done */
        while ( !(ADC0.INTFLAGS & ADC_RESRDY_bm) )
        {
            ;
        }
```

```
    /* Clear the interrupt flag by writing 1: */
    ADC0.INTFLAGS = ADC_RESRDY_bm;

    return ADC0.RES;
}

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

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

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

    /* Wait VREF start-up time */
    _delay_us(VREF_STARTUP_MICROS);

    DAC0_setVal(dacVal);

    while (1)
    {
        adcVal = ADC0_read();

        /* do something with the adcVal */

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

Example 8-4. Using DAC as Negative Input for AC Code Example

```
/* 3.33 MHz (needed for delay) */
#define F_CPU                         (3333333UL)
/* 1.4V output @ VREF = 1.5V */
#define DAC_DATA_1V4                  (239)
/* VREF Startup time */
#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)
{
    /* Voltage reference at 1.5V */
    VREF.CTRLA |= VREF_DAC0REFSEL_1V5_gc;
    /* DAC0/AC0 reference enable: enabled */
    VREF.CTRLB |= VREF_DAC0REFEN_bm;
    /* Wait VREF start-up time */
    _delay_us(VREF_STARTUP_MICROS);
}

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

void DAC0_init(void)
{
    /* Disable digital input buffer */
    PORTA.PIN6CTRL &= ~PORT_ISC_gm;
    PORTA.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;
```

```
    /* Disable pull-up resistor */
    PORTA.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    /* Enable DAC, Output Buffer, Run in Standby */
    DAC0.CTRLA = DAC_ENABLE_bm | DAC_OUTEN_bm | DAC_RUNSTDBY_bm;
}

void AC0_init(void)
{
    /* Negative input from DAC0 */
    /* Positive input from pin PA7 */
    AC0.MUXCTRLA = AC_MUXNEG_DAC_gc
                 | AC_MUXPOS_PIN0_gc;

    /* Enable, Output on PA5, Low Power mode */
    AC0.CTRLA = AC_ENABLE_bm
                 | AC_OUTEN_bm
                 | AC_RUNSTDBY_bm
                 | AC_LPMODE_bm;
}

int main(void)
{
    /* Voltage divider -> VDD/2 input on PA7 */
    /* AC output on LED on PA5 */
    /* LED turns OFF when battery is below 2.8V (PA7 below 1.4V) */

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

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

    while (1)
    {
        ;
    }
}
```

9. Revision History

Document Revision	Date	Comments
B	03/2021	Updated the GitHub repository links, the *References* section, and the use cases sections. Added the *AVR® DA Family Overview* and *Revision History* sections. Added MCC versions for each use case, running on AVR128DA48. Other minor corrections.
A	05/2019	Initial document release.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-7853-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra’anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820

http://darcy.rsgc.on.ca/ACES/MinskyCircle/Minsky%20circle%20drawing.htm

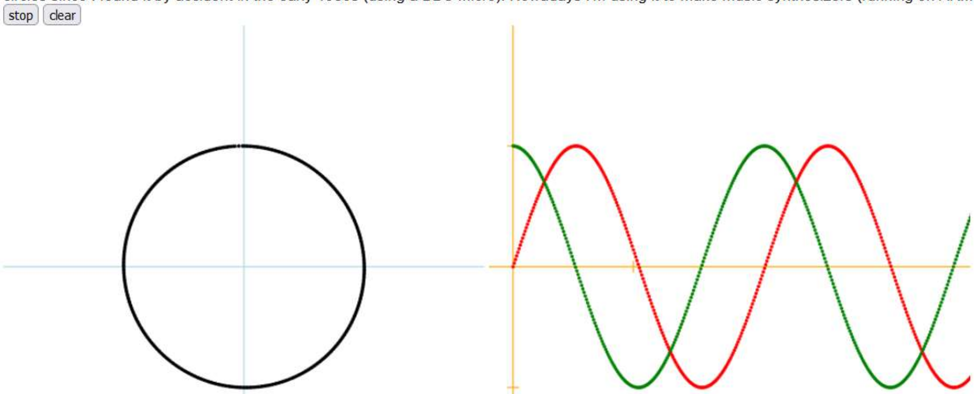
Fast iterative circles (and ellipses, and other figures).

Here's the entire algorithm to compute points on an elliptical arc, very quickly:

```
while(true)
{
    x += d * y;
    y -= d * x;
}
```

Attributed to Marvin Minsky, 1972: HAKMEM, MIT AI Memo 239 (HTML version here). Also on a PDP-1, David Mapes talks about finding it independently. I've been using this to make circles since I found it by accident in the early 1980s (using a BBC Micro). Nowadays I'm using it to make music synthesizers (running on ARM Cortex M4).

stop clear



See Also: Technoblogy: http://www.technoblogy.com/show?22HF

https://www.daycounter.com/Calculators/Sine-Generator-Calculator.phtml

Daycounter - Engineering Resources About Articles Calculators Circuits Lab Book Part Sources Site Map

Sine Look Up Table Generator Calculator

This calculator generates a single cycle sine wave look up table. It's useful for digital synthesis of sine waves.

Sine Look Up Table Generator Input	
Number of points	1023
Max Amplitude	65535
Numbers Per Row	8
Delimeter (Text Value Separator)	,
Format	Hex

Calculate