AN2551

Noise Countermeasures for ADC Applications

Introduction

Author: Rupali Honrao, Microchip Technology Inc.

This application note will explain how and when to use the powerful noise suppression features available on the Microchip tinyAVR® 0- and 1-series, and megaAVR® 0-series ADCs. In these ADCs, the input signal is fed through a Sample-and-Hold circuit which ensures that the input voltage to the ADC is held at a constant level during sampling.

The ADC supports sampling in bursts where a configurable number of conversion results are accumulated into a single ADC result (sample accumulation). Further, a sample delay can be configured to tune the ADC sampling frequency associated with a single burst. This is done to tune the sampling frequency away from any harmonic noise aliased with the ADC sampling frequency (within the burst) from the sampled signal. An automatic sampling delay variation feature can be used to randomize this delay to slightly change the time between samples.

In this case, a test and verification setup is suggested. The topics covered include using the ADC hardware sample accumulator to filter out zero mean random noise, and surpassing harmonic noise through tuned sampling delays or automatic sampling delay variation. A test setup for generating noisy signals is suggested and directions on how to illustrate the results in the Atmel Studio Data Visualizer are provided.

The example code for replicating the results described in this application note is available from Atmel | START:

- Periodic noise generation and noise filtering with sample accumulation, sampling delay, and automatic sampling delay.

Additional details on ADC performance and general configuration are available in the device data sheet.

Features

- ADC Sample Accumulation up to 64 Samples per Conversion
- Generating Noise with the AVR® Microcontroller, Using PWM, and Adding it to the Input Signal
- Filter Random Noise and Harmonic Noise
- Sampling Delay
- Automatic Sampling Delay
- Plotting Graph of ADC Samples in *Data Visualizer*

Table of Contents

1. Relevant Devices

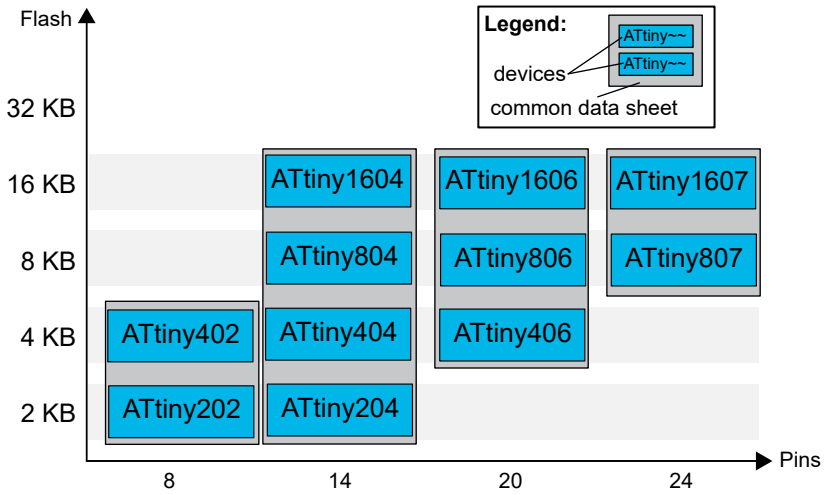
This chapter lists the relevant devices for this document.

1.1 tinyAVR 0-series

The figure below shows the tinyAVR 0-series, 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 1-1. tinyAVR® 0-series Overview



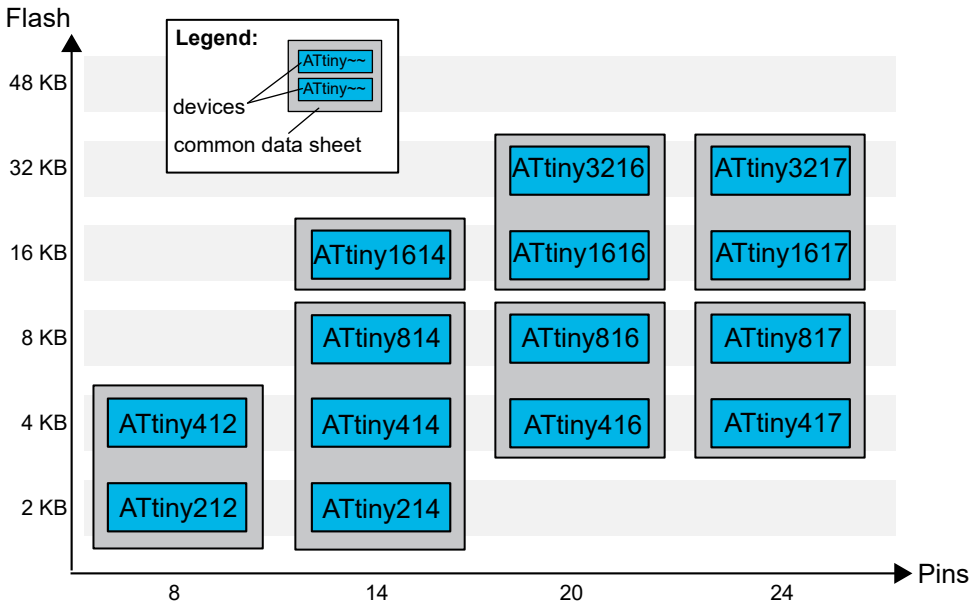
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.2 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 1-2. tinyAVR® 1-series Overview



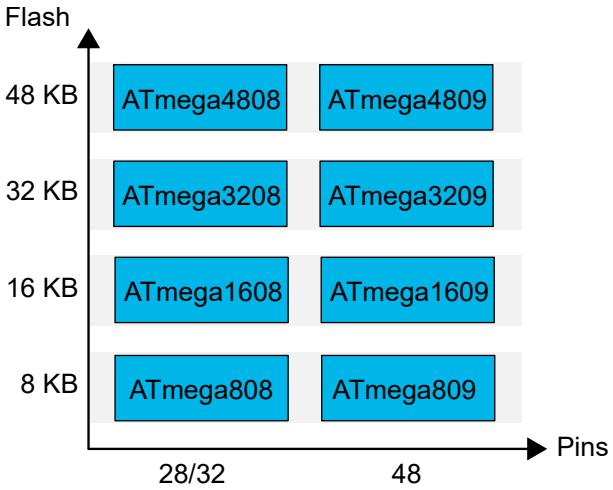
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.3 megaAVR® 0-series

The figure below shows the megaAVR 0-series 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 1-3. megaAVR® 0-series Overview



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

2. Overview

Microchip tinyAVR® 0- and 1-series, and megaAVR® 0-series devices feature a successive approximation Analog-to-Digital Converter (ADC) with a maximum conversion rate of 150 ksps with 8-bit resolution, or 115 ksps with 10-bit resolution. The ADC has a flexible input multiplexer allowing single-ended ground referenced measurements of multiple internal and external input sources. Measurements can be done against a selection of internal voltage references (0.55V, 1.1V, 2.5V, and 4.3V) or directly against V_{DD}.

Noise countermeasures supported by the ADC are:

- Hardware sample accumulator
- Sampling delay
- Automatic sampling delay variation

2.1 Hardware Sample Accumulator

The ADC can be configured to automatically accumulate a number of sampling results for a single conversion trigger by writing the Sample Accumulation Number Select (SAMPNUM) bit field in the Control B (ADC.CTRLB) register. Upon receiving a conversion trigger, $2^{SAMPNUM}$ sampling results will be accumulated in the ADC Result register (ADC.RES) before the Result Ready (RESRDY) bit in the Interrupt Flags (ADC.INTFLAGS) register is set. The ADC supports accumulation of up to 64 sampling results.

2.2 Sampling Delay

The Sample Delay Selection bit field (SAMPDLY) in the Control B (ADC.CTRLB) register can be written to insert a number of delay cycles between consecutive ADC conversions. When used together with hardware sample accumulation, SAMPDLY can be used to de-alias the ADC bursting frequency from harmonic noise frequency components present in the sampled analog signal.

2.3 Automatic Sampling Delay Variation

Automatic Sampling Delay Variation (ASDV) together with sample accumulation can be helpful in suppressing harmonic noise with unknown base frequency. If ADSV is enabled, the ADC will cycle through the supported sampling delay configurations, increasing the delay by one ADC clock (CLK_ADC) cycle for every conversion. The varying sampling delay can provide noise attenuation over a broader frequency range compared to a fixed sampling delay, but does so at the cost of a reduced attenuation factor. In situations where the harmonic noise frequency is known or measurable, it is recommended to use a sampling delay tuned to suppress the present frequency components. ADSV can be enabled by writing the ADSV bit to the Control D (ADC.CTRLD) register.

3. Theory: Noise Suppression

3.1 Signal with Random Noise

Many MCU applications involve measuring analog signals. In the ideal case of a completely noiseless signal, achieving a high-quality digital representation of the signal is simple: single ADC conversions triggered with a fixed time interval are sufficient. In reality, most analog signals are affected by noise, and modern Microchip ADCs provide functionality that can be used to increase the signal-to-noise ratio.

Figure 3-1 illustrates a noisy signal. Single conversions evenly spaced in time result in a noisy digital representation of the signal. A potential solution could be to filter the acquired samples in software, but this would require additional CPU resources. A better option would be to use the hardware sample accumulator supported by the ADC.

Figure 3-2 illustrates how a single ADC conversion trigger results in a burst of up to 64 successive ADC conversions. Each conversion will be accumulated in hardware and the average value of the accumulated samples can be calculated by dividing the accumulated result by burst size. Because the sampled noise has a zero mean, the averaged result will be close to the actual signal value.

Here, ADC feature hardware sample accumulator can be used for averaging by configuring ADC to accumulate m samples automatically. ADC sampling rate is affected by the number of samples accumulated. Total sampling time for m samples is the multiplication by m (the number of samples taken) of the sampling time for one sample.

Consider a DC signal mixed with random noise as shown in Figure 3-1.

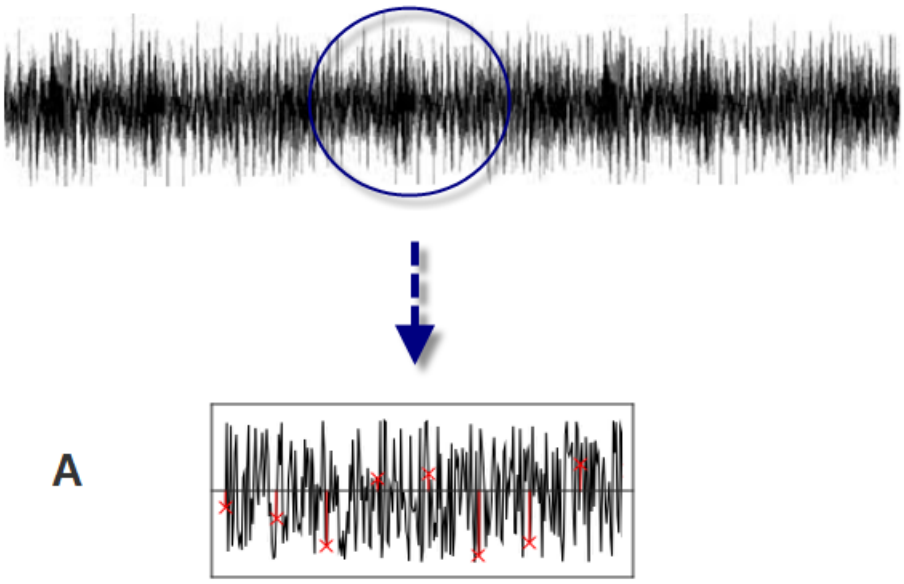
Figure 3-1. DC Signal Mixed with Random Noise

If the signal is zoomed, consider ADC samples which are taken as indicated in part 'A' of Figure 3-2.

Consider oversampling is done with eight samples. This means that a single burst contains eight samples.

In a single burst, eight samples are taken, which are shown with red markings in Figure 3-2.

Figure 3-2. Detailed Noise Signal



As oversampling is done with multiple samples, the average result of all the sampled values will be approximately equal to the original DC signal. That means it results in zero mean noise. Increasing the burst size (accumulating more samples) helps to flatten out more peak signals and results in more suppression in noise.

3.2 Signal with Periodic Noise

The potential scenarios where the periodic noise occurs are switch mode regulators or PWM signals used for motor or LED control, etc.

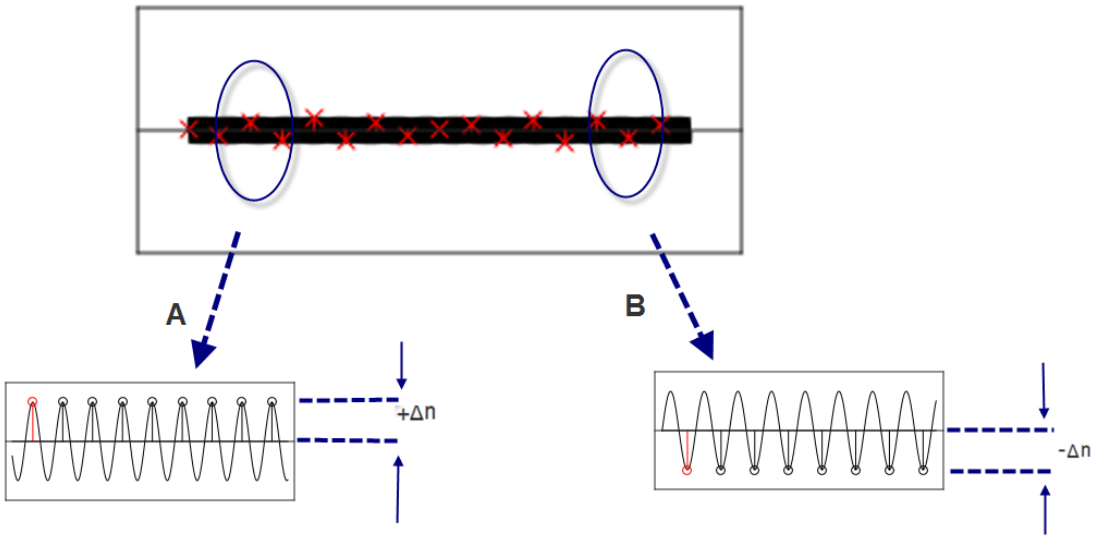
With the feature hardware sample accumulator, average value of the accumulated samples can be calculated by dividing the accumulated result by burst size (number of samples accumulated) and resulting in a zero mean sampled noise.

Consider a DC signal mixed with periodic noise and an accumulation of multiple samples shown with red markings, as in Figure 3-3.

If the signal is zoomed, consider ADC samples which are taken in part 'A' and part 'B' in Figure 3-3.

As a hardware sample accumulator is used, the average result of ADC samples in part 'A' will be $\cong +\Delta n$. In part 'B', the average result of ADC samples will be $\cong -\Delta n$. As each individual sample differ from zero with equal probability of being either positive or negative, the accumulated noise samples will approach zero, and the noise is successfully suppressed.

Figure 3-3. DC Signal Mixed with Periodic Noise: Sample Accumulator



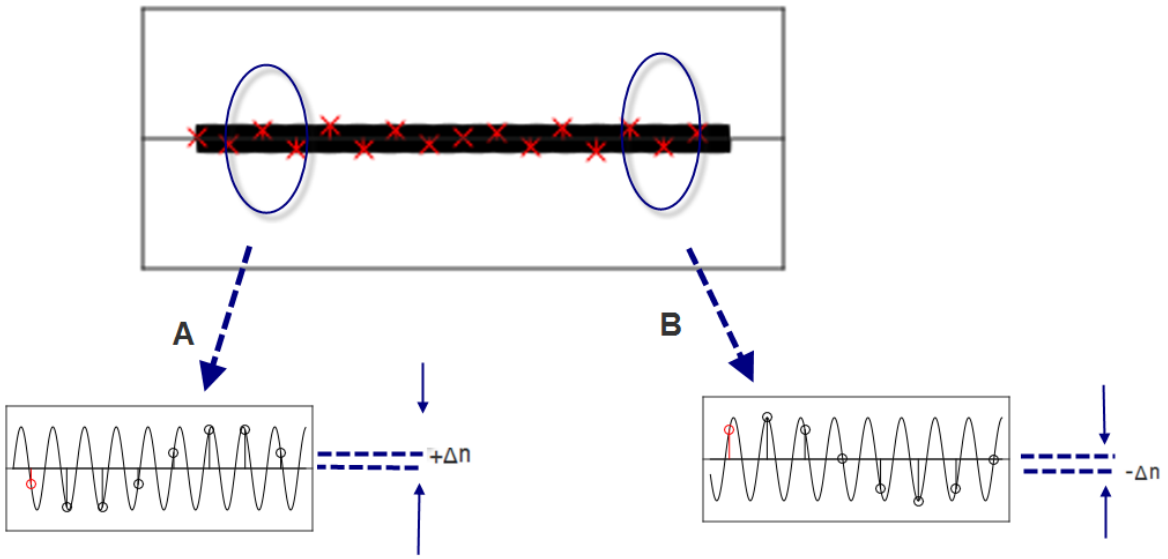
Sampling Delay or Automatic Sampling Delay Variation:

When the periodic noise frequency is known, the sampling rate can be adjusted to tune the sampling frequency away from any periodic or harmonic noise aliased with the ADC sampling frequency (within the burst) from the sampled signal.

In case of unknown periodic noise frequency, an automatic sampling delay variation feature can be used to randomize this delay to slightly change the time between samples and achieve better noise suppression. In this case, broader suppression range is achieved but with reduced noise attenuation.

If sampling delay is added and sampling frequency is adjusted, consider that the actual read ADC samples are as shown in part 'A' and 'B' in Figure 3-4. In this scenario, the average value of samples taken in batch 'A' will be close to zero and the average value of the samples taken in batch 'B' will be equally close to zero, and it results in suppressing the noise signal.

Figure 3-4. DC Signal Mixed with Periodic Noise: Sampling Delay



The sampling delay and sampling length can be adjusted using the Sample Delay bit field in the Control D (ADC.CTRLD) register and the Sample Length bit field in the Sample Control (ADC.SAMPCTRL) register. Both of these control the ADC sampling time in CLK_ADC cycles.

Total sampling time is given by:

$$\text{SampleTime} = \frac{(2 + \text{SAMPDLY} + \text{SAMPLEN})}{f_{\text{CLK\_ADC}}}$$

In Free-Running mode, the sampling rate R_S is calculated by:

$$\text{SampleRate} = \frac{f_{\text{CLK\_ADC}}}{(13 + \text{SAMPDLY} + \text{SAMPLEN})}$$

Sampling Delay Selection bits in the ADC.CTRLD register define the delay between consecutive ADC samples. The SAMPDLY field can also be automatically modified from one sampling cycle to another, by setting the ASDV bit. The delay is expressed as CLK_ADC cycles. When the ASDV bit is not enabled, the SAMPDLY value can be configured from 0 to 15. 0 means no sampling delay, 1 is a delay of 1 cycle, and so on. In the setup, it is required to tune this value correctly for obtaining the desired results and to suppress the present frequency components. It is needed to configure this value by a trial-and-error method so that +Δn or -Δn is close to zero.

From the above figures it can be seen that with a correctly tuned value of the sampling delay (SMPDLY) between consecutive samples, the average result of ADC samples approach the noise free signal.

With an increase in sampling delay or number of samples accumulated, the sampling frequency is reduced.

4. Add Noise to the Signal

To verify the ADC features, artificial noise can be generated with various methods and then this noise can be added to the input signal.

- Generating periodic noise with the AVR, using PWM, and adding it to the input signal.
- Generating random noise with the AVR, using DAC, and adding it to the input signal.

Note: Pseudo-random noise can be generated from any device with onboard DAC.

The example code to generate random noise:

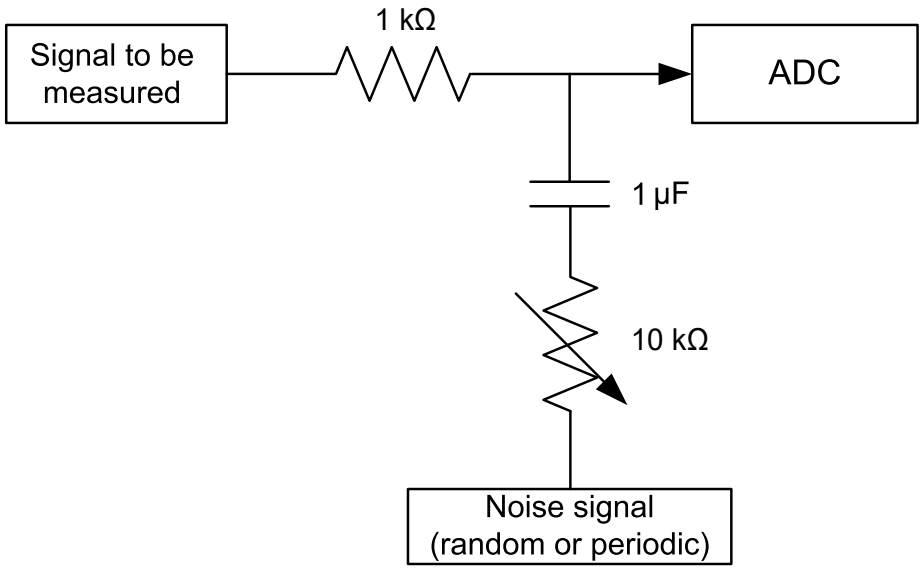
```
unsigned int i;
unsigned int k;
unsigned char random_buf[512];
for (i=0;i<512;i++)
{
    random_buf[i]= rand()%256;
}
dac_init();
while (1)
{
    DAC.DATA = random_buf[k++];
    if (k>512)
    {
        k=0;
    }
}
```

`random_buf[512]` is an array of random numbers 0 to 255 using standard library function '`rand()`'.

To generate a true random noise signal, sufficiently high DAC conversion rate is needed. It can be ensured by selecting the CPU clock frequency of the device which generates random DAC noise higher than the device using it as input noise signal.

The circuit illustrated in Figure 4-1 can be used to add noise to the signal to be measured.

Figure 4-1. Adding Noise to the Signal Circuit



Note: Precautions may be taken so that the voltage level of the mixed signal may not go above the selected ADC reference voltage. Device characteristics do not recommend input voltage signal above ADC reference voltage.

For periodic noise, a PWM signal can be generated. To get the worst-case scenario of periodic noise in the test setup, the PWM frequency can be selected close to the ADC sampling frequency.

In the example source code, the default CPU clock is 3.33 MHz. ADC clock is 3.33 MHz/4 = 832.5 kHz.

In Free-Running mode, the sampling rate R_S is calculated by:

$$\text{SampleRate} = \frac{f_{\text{CLK\_ADC}}}{(13 + \text{SAMPDLY} + \text{SAMPLEN})}$$

So, R_S is 832.5 kHz/13 = 64 kHz.

In the example source code, the PWM signal is generated with a 62 kHz frequency which is close to the ADC sampling frequency.

5. Demonstrate Noise Filtering

The noise filtering is demonstrated by using an example source code and plotting a graph of ADC samples in *Data Visualizer* in Atmel Studio.

The example source code generates PWM noise using the TCA timer. This PWM signal is added as noise to the 'signal to be measured', as shown in Figure 4-1. A DC signal from the potentiometer is used as 'signal to be measured'. This mixed signal (signal + noise signal) is given as input signal to ADC. It will be sampled and the ADC result value is sent through USART to the serial terminal of the *Data Visualizer* and a graph of the ADC samples is plotted in the *Data Visualizer*.

Different graphs with different noise filtering configurations such as sample accumulation for 1 or 64 samples, sampling delay, and automatic sampling delay are plotted. From these graphs, it can be observed how the ADC result count range gets reduced when noise is suppressed with configured ADC features.

A detailed explanation is provided in the further sections.

5.1 Source Code Overview

Source code overview using ATmega4809 Xplained Pro:

- CPU clock: (default) 3.33 MHz.
- Peripherals used:
  - ADC, TCA, USART, V_{REF}.
  - ADC input channel is AIN 5: Pin PD5, 10-bit ADC resolution.
  - TCA: PWM signal is generated on pin PA0: 62 kHz, 50% duty cycle.
  - USART: TXD PC0, Baud rate: 19200, ADC result is sent to serial terminal.
  - V_{REF} selects the ADC reference voltage to 2.5V.

The project configured in Atmel START generates peripheral driver functions and files, as well as a '`main()`' function that initializes all drivers.

- Driver header and source files are in the *src* and *include* folder.
- In `atmel_start.c` file, the function '`atmel_start_init()`' initializes MCU, drivers, and middleware in the project.

Source code overview using ATtiny817 Xplained Pro:

- CPU clock: (default) 3.33 MHz.
- Peripherals used:
  - ADC, TCA, USART, V_{REF}.
  - ADC input channel is AIN 5: Pin PA5, 10-bit ADC resolution.
  - TCA: PWM signal is generated on pin PB0: 62 kHz, 50% duty cycle.
  - USART: TXD PB2, Baud rate: 19200, ADC result is sent to the serial terminal.
  - V_{REF} selects the ADC reference voltage to 2.5V.

The project configured in Atmel START generates peripheral driver functions and files, as well as a '`main()`' function that initializes all drivers.

- Driver header and source files are in the *src* and *include* folder.

- In `atmel_start.c` file, the function '`atmel_start_init()`' initializes MCU, drivers, and middleware in the project.

5.1.1 Macro Definitions

Below are the macro definitions in the source code in `main.c` file listed.

- **HARMONIC_NOISE**

```
#define HARMONIC_NOISE 1
```

`1`: Generate PWM signal as a periodic noise.

`0`: Do not generate PWM signal.

- **PWM_FRQ**

```
#define PWM_FRQ 62000
```

Configuration of PWM frequency, 62 kHz.

To get the worst-case scenario of periodic noise in the test setup, the PWM frequency can be selected close to ADC sampling frequency.

- **ADC_64X_ACCUMULATOR_ENABLE**

```
#define ADC_64X_ACCUMULATOR_ENABLE 1
```

Enable configuration of consecutive sample accumulation.

`1`: 64 consecutive sample accumulation.

`0`: No multiple sample accumulation (only one ADC sample accumulation).

Note that this example shows a result graph with the ADC sample accumulation of 1 sample or 64 samples by using only the `ADC_64X_ACCUMULATOR_ENABLE` macro.

The Microchip tinyAVR® 1-series and megaAVR® 0-series support sample accumulation of 1, 2, 4, 8, 16, 32, and 64. For sample accumulation other than 1 or 64, the code needs to be changed accordingly.

- **SAMPLING_DELAY**

```
#define SAMPLING_DELAY 5
```

`0`: No sampling delay between consecutive samples in one single burst.

1 to 15: Configures the sampling delay between consecutive samples in one single burst. The delay is expressed as CLK_ADC cycles, 1 being a delay of one cycle.

Note: In the test setup, maximum noise filtering is achieved when the sampling delay value has been configured to 5. It has been configured using trial and error to tune this value correctly for obtaining the desired results and to suppress the present frequency components. It may vary from one setup to another.

- **ENABLE_ASDV**

```
#define ENABLE_ASDV 0
```

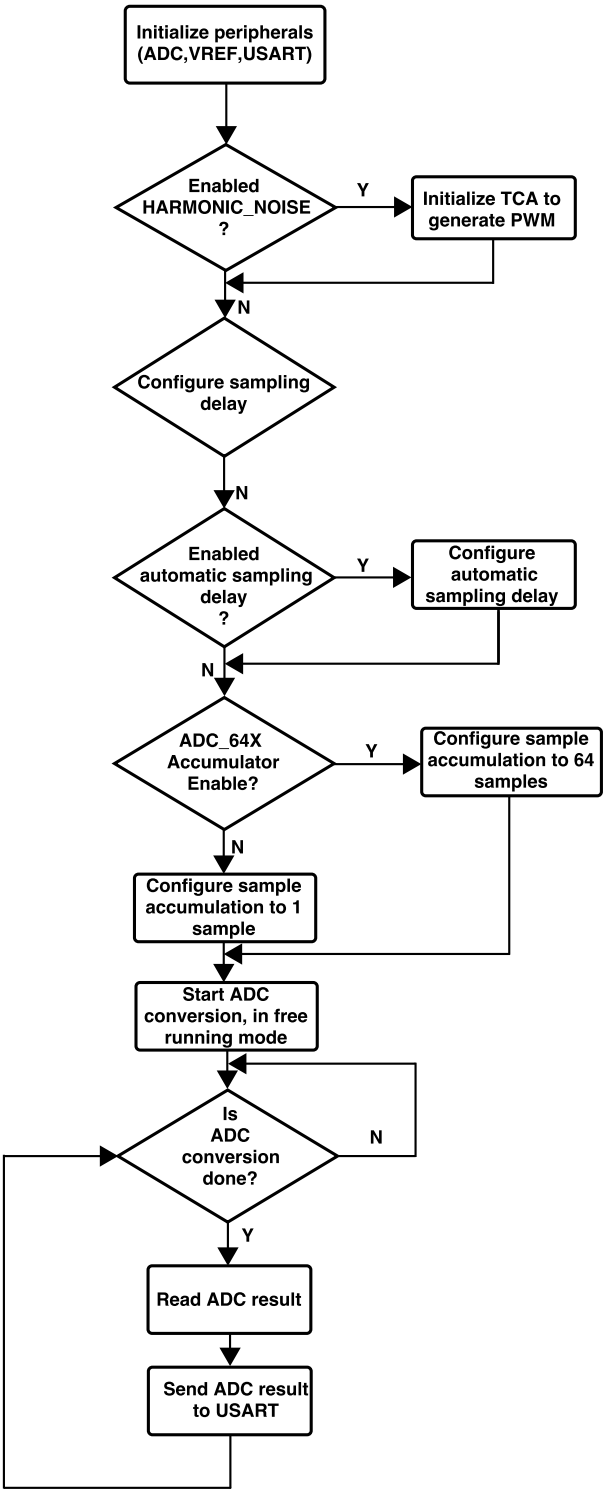
`1`: Enable the automatic sampling delay between consecutive samples in one single burst.

`0`: Do not enable automatic sampling delay variation.

5.1.2 Application Flow Diagram

The overall application flow diagram looks as shown in Figure 5-1.

Figure 5-1. Application Flow Diagram



5.2 Results with Graph

Results will be shown by plotting different graphs in *Atmel Studio Data Visualizer* and by configuring advanced ADC features. Here are plotted different graphs with different noise filtering configurations such as sample accumulation for 1 or 64 samples, sampling delay, and automatic sampling delay variation. From these graphs, it can be observed how the ADC result count range gets reduced when noise is suppressed with configured ADC features.

Note: Refer to 7. Appendix A: Plotting Graph in Data Visualizer for more details on how to plot graphs in *Atmel Studio Data Visualizer*.

5.2.1 Signal with No Noise

Task: The graph will be plotted in *Data Visualizer* using a DC signal without noise.

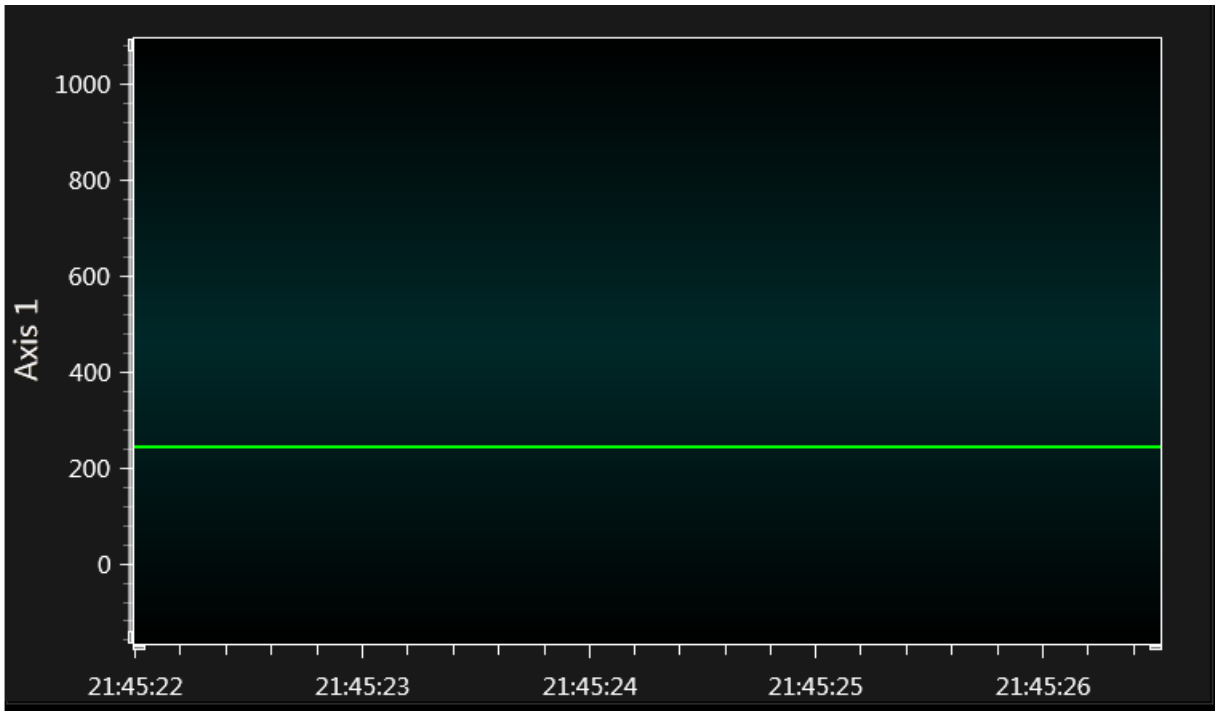
In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  0
#define ADC_64X_ACCUMULATOR_ENABLE   0
#define SAMPLING_DELAY 0
#define ENABLE_ASDV  0
```

When a DC signal of 0.6V is connected to ADC input pin PD5, a graph in *Data Visualizer* is plotted as below.

Note: No noise has been added to the input signal.

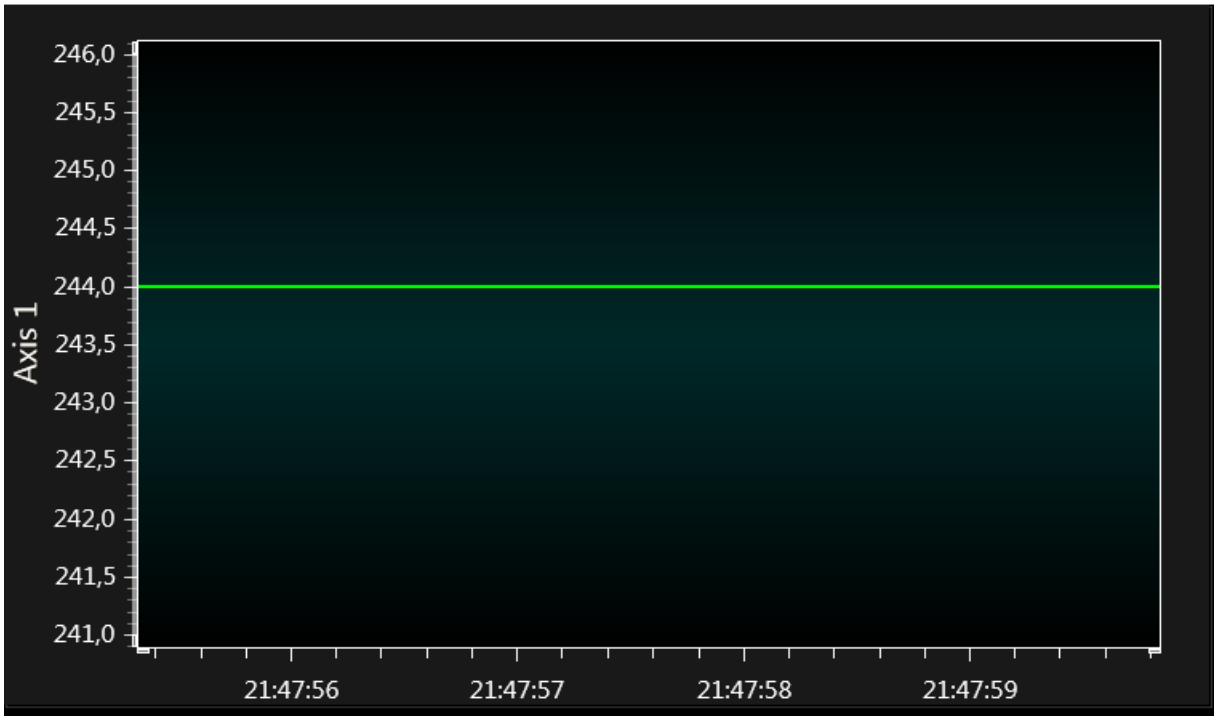
Figure 5-2. DC Signal with No Noise



From the graph, the ADC result value can been seen around 240 and no noise has been observed.

If the graph is zoomed into, it can be seen that the ADC count is 244 (see Figure 5-3).

Figure 5-3. Zoomed DC Signal with No Noise



The ADC reference voltage is 2.5V and the ADC resolution is 10 bit.

Ideally measured ADC count may be (1023 x 0.6)/2.5 = 245.

Note: Refer to Appendix A: Plotting graph in Data Visualizer.

5.2.2 Signal with Random Noise

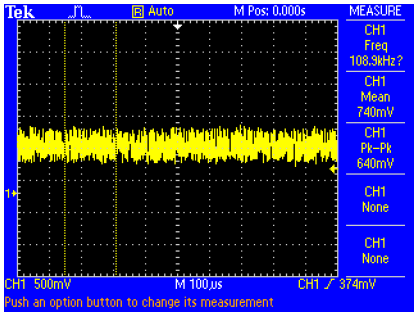
Task: Two graphs will be plotted in *Data Visualizer* using a DC signal mixed with random noise.

- A graph with 1-sample accumulation.
- A graph with 64-sample accumulation.

Test setup: The random noise is generated using a device with DAC and added this random noise to the DC signal to be measured, as shown in Figure 4-1.

A signal with random noise is shown in Figure 5-4. The DC signal level is 740 mV and the noise amplitude is 640 mV pk-pk.

Figure 5-4. Signal with Random Noise Oscilloscope Capture



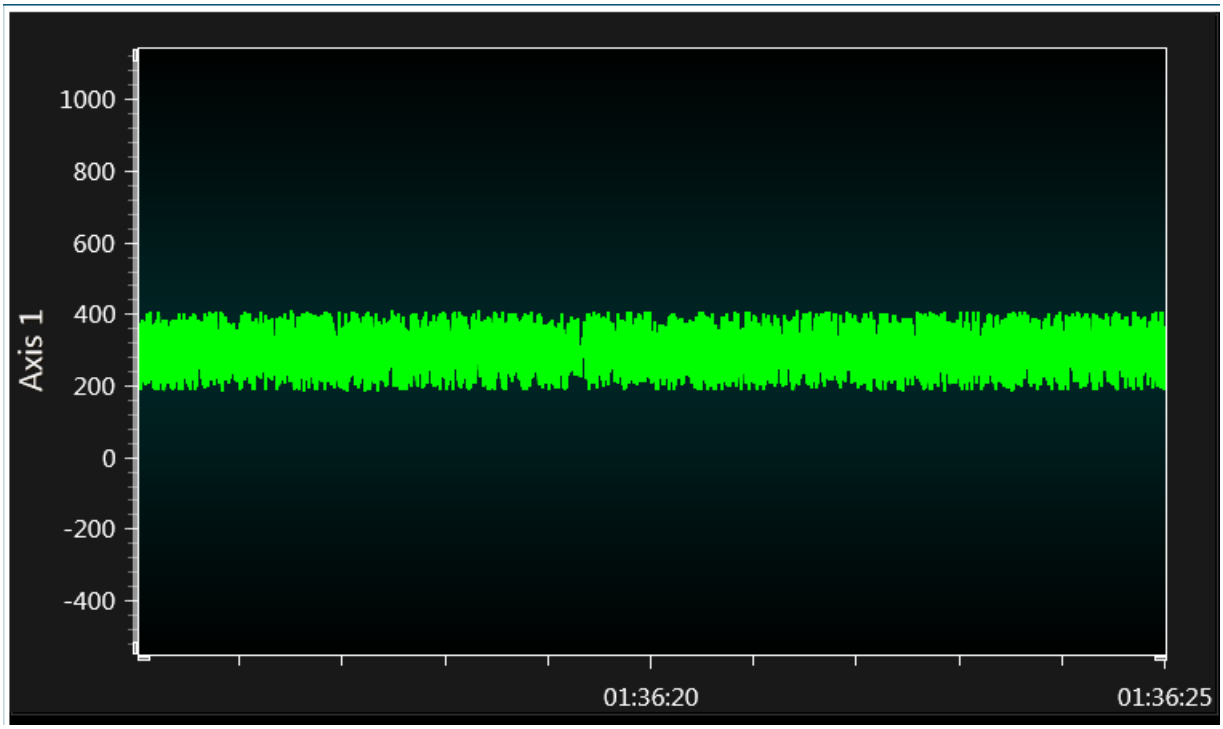
1. A graph with 1-sample accumulation:

In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  0
#define ADC_64X_ACCUMULATOR_ENABLE   0
#define SAMPLING_DELAY 0
#define ENABLE_ASDV  0
```

After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-5:

Figure 5-5. Signal with Random Noise: 1-Sample Accumulation



ADC count can be seen varying from 200 to 400 because of random noise. That means ADC count is varying ±100 counts (400 to 200 → 300 ±100 counts).

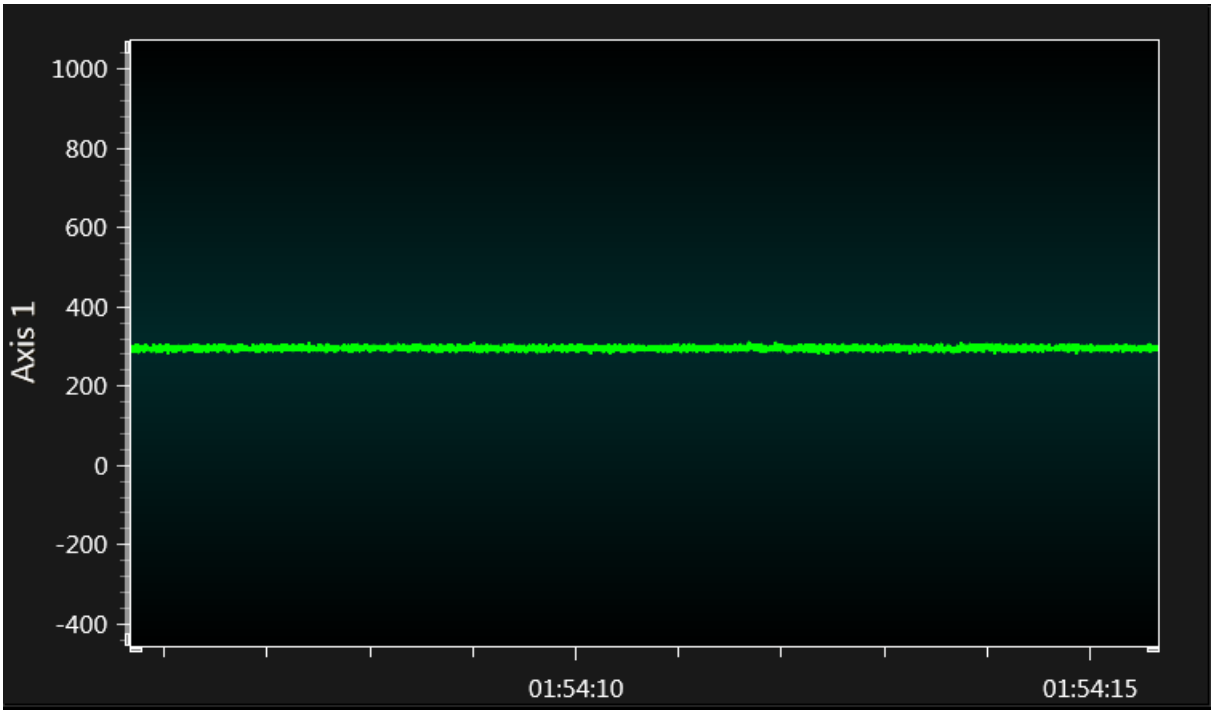
2. A graph with 64-sample accumulation:

In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  0
#define ADC_64X_ACCUMULATOR_ENABLE   1
#define ENABLE_SMP_DLY 0
#define ENABLE_AUTOTMATIC_SAMP_DLY  0
```

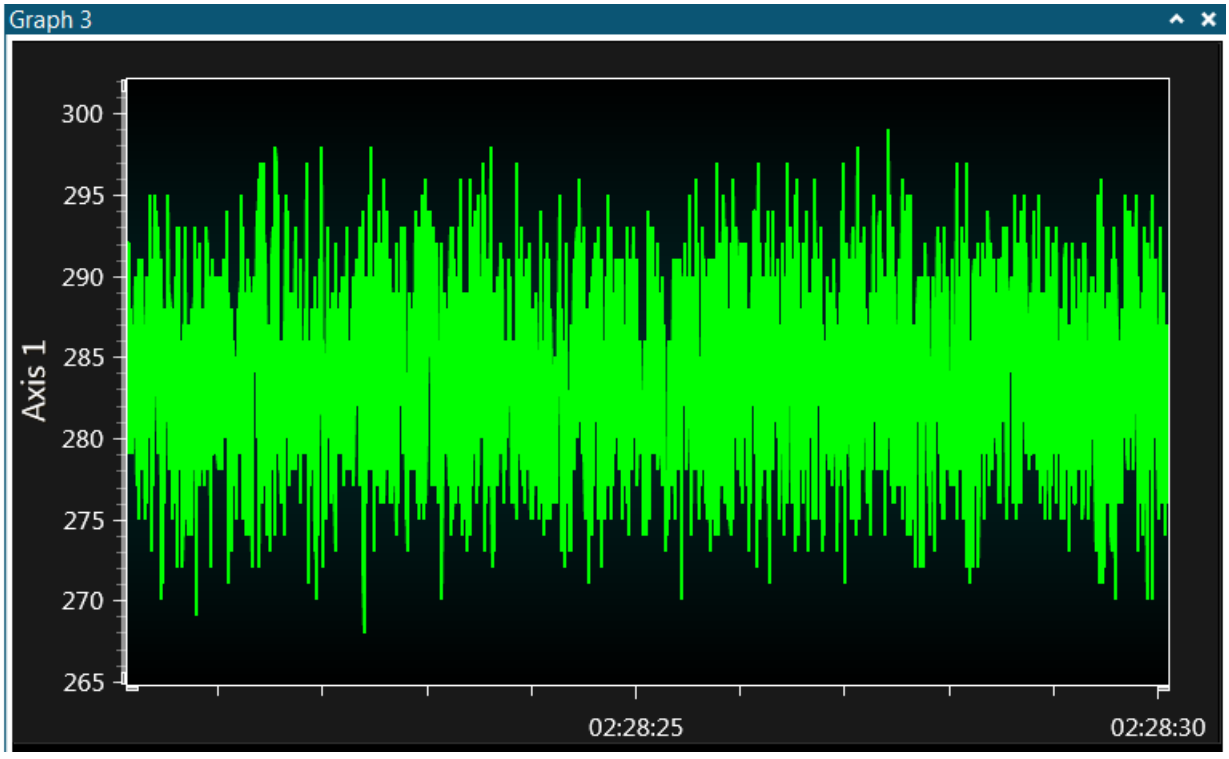
After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-6:

Figure 5-6. Signal with Random Noise: 64-Sample Accumulation



From the graph, it can be observed that by oversampling with 64 ADC samples accumulation, the ADC result count range has been reduced. If the signal is zoomed, the image is as shown in Figure 5-7:

Figure 5-7. Zoomed Signal with Random Noise: 64-Sample Accumulation



The ADC result count range is varying between 295 to 270. That means ADC count is varying ±13 counts (295 to 270 → 282 ±13 counts).

5.2.3 Signal with Periodic Noise

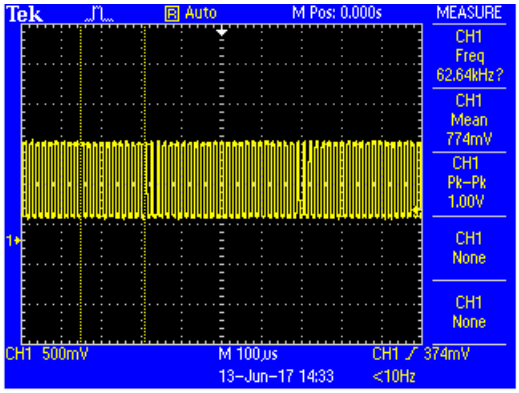
Task: Four graphs will be plotted in *Data Visualizer* using a DC signal mixed with periodic noise.

- A graph with 1 ADC sample accumulation.
- A graph with 64 ADC samples accumulation.
- A graph with 64 ADC samples and automatic sampling delay variation.
- A graph with 64 ADC samples and sampling delay.

Test setup: The PWM signal is generated using peripheral TCA and added this PWM noise to the DC signal to be measured as shown in Figure 4-1.

The signal with PWM noise is shown in Figure 5-8:

Figure 5-8. Signal with PWM Noise, Oscilloscope Capture



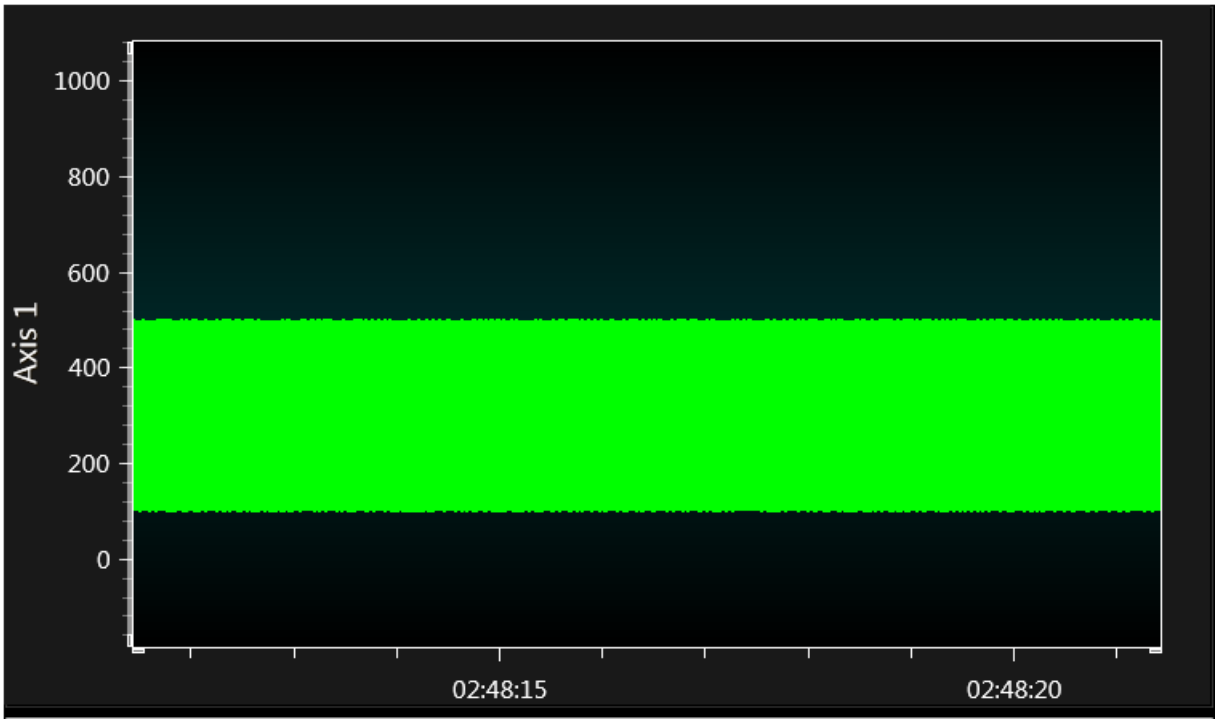
1. A graph with 1-sample accumulation:

In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  1
#define ADC_64X_ACCUMULATOR_ENABLE   0
#define SAMPLING_DELAY 0
#define ENABLE_ASDV  0
```

After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-9:

Figure 5-9. Periodic Noise: 1-Sample Accumulation



It is observed that the ADC result count value is varying around 100 to 450.

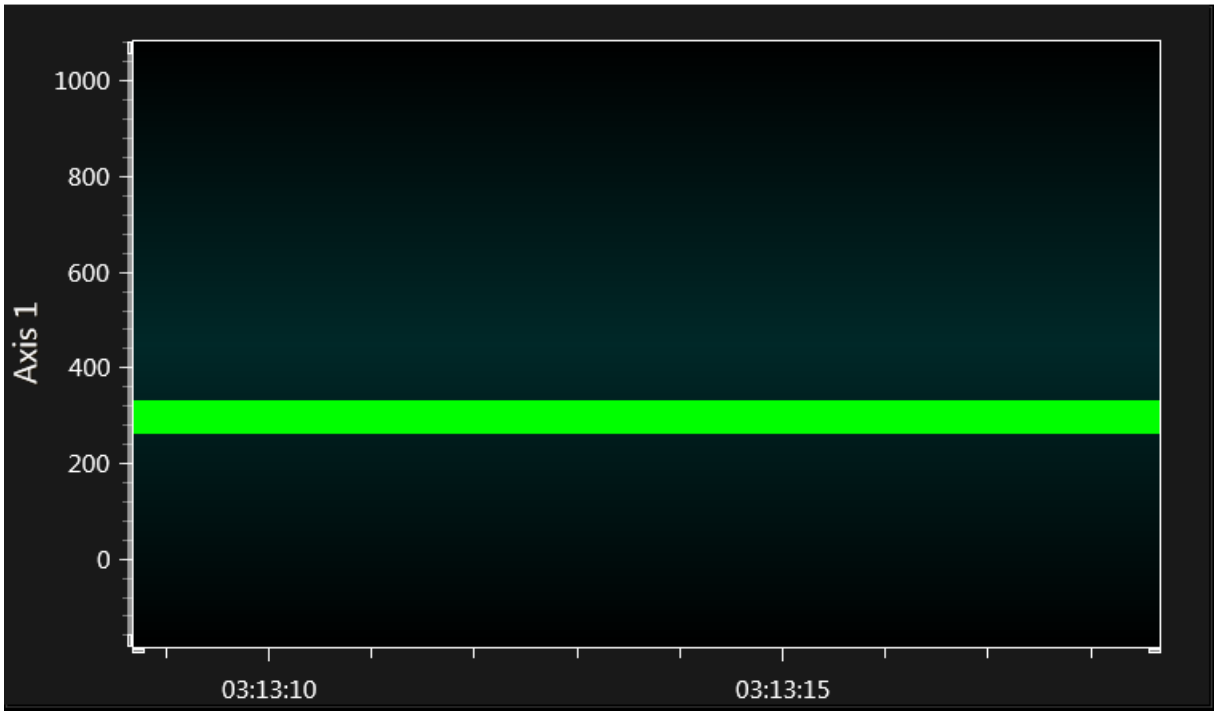
2. A graph with 64-sample accumulation:

In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  1
#define ADC_64X_ACCUMULATOR_ENABLE   1
#define SAMPLING_DELAY 0
#define ENABLE_ASDV  0
```

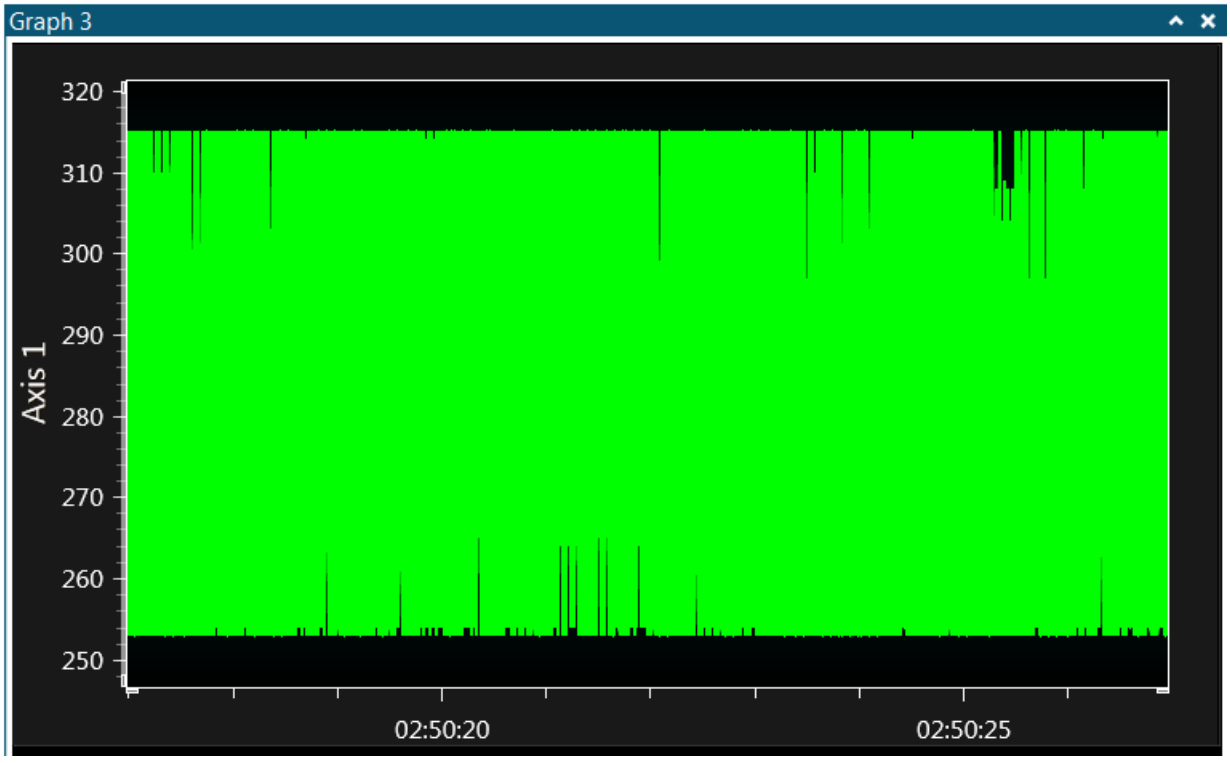
After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-10:

Figure 5-10. Periodic Noise: 64-Sample Accumulation



If the above signal is zoomed, the image is as shown in Figure 5-11:

Figure 5-11. Zoomed Periodic Noise: 64-Sample Accumulation



With oversampling, the ADC result count range has been reduced and the value is varying from 255 to 315. That means the ADC result count range is approx. ±30 counts.

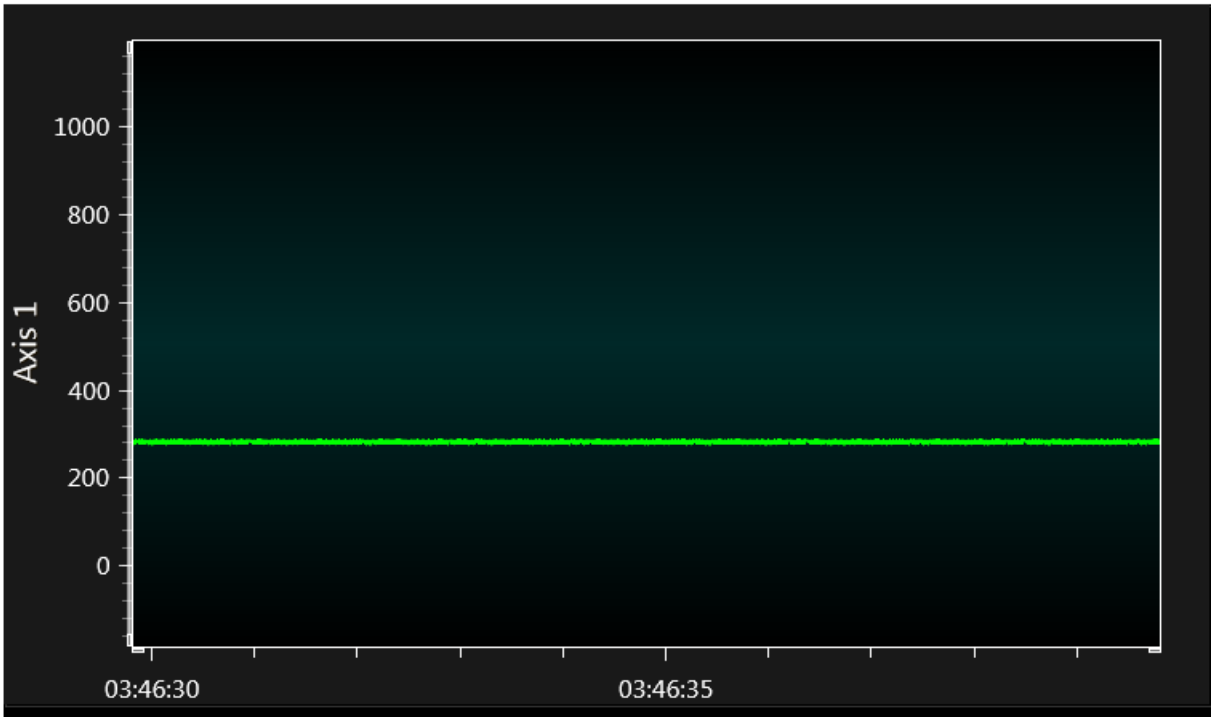
3. A graph with 64-sample accumulation and automatic sampling delay:

In the example code, the configuration of macro definitions is as below:

```
#define HARMONIC_NOISE  1
#define ADC_64X_ACCUMULATOR_ENABLE   1
#define SAMPLING_DELAY 0
#define ENABLE_ASDV  1
```

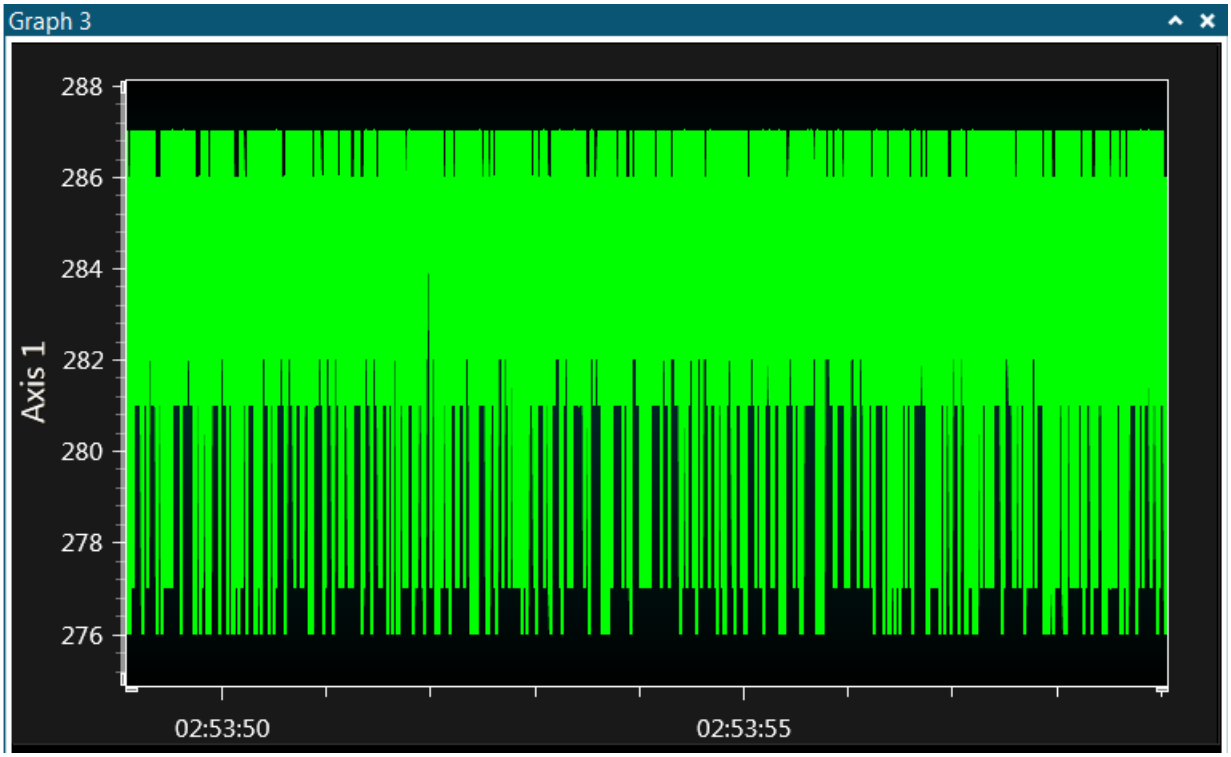
After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-12:

Figure 5-12. Periodic Noise: 64-Sample Accumulation and Automatic Sampling Delay



If the above signal is zoomed, the image is as shown in Figure 5-13:

Figure 5-13. Zoomed Periodic Noise: 64-Sample Accumulation and Automatic Sampling Delay



With oversampling and adding automatic sampling delay, the ADC result count range has been further reduced and the value is varying from 276 to 287. This means that the ADC result count range is approx. ±6 counts.

4. A graph with 64-sample accumulation and sampling delay:

In the example code, the configuration of macro definitions is as below:

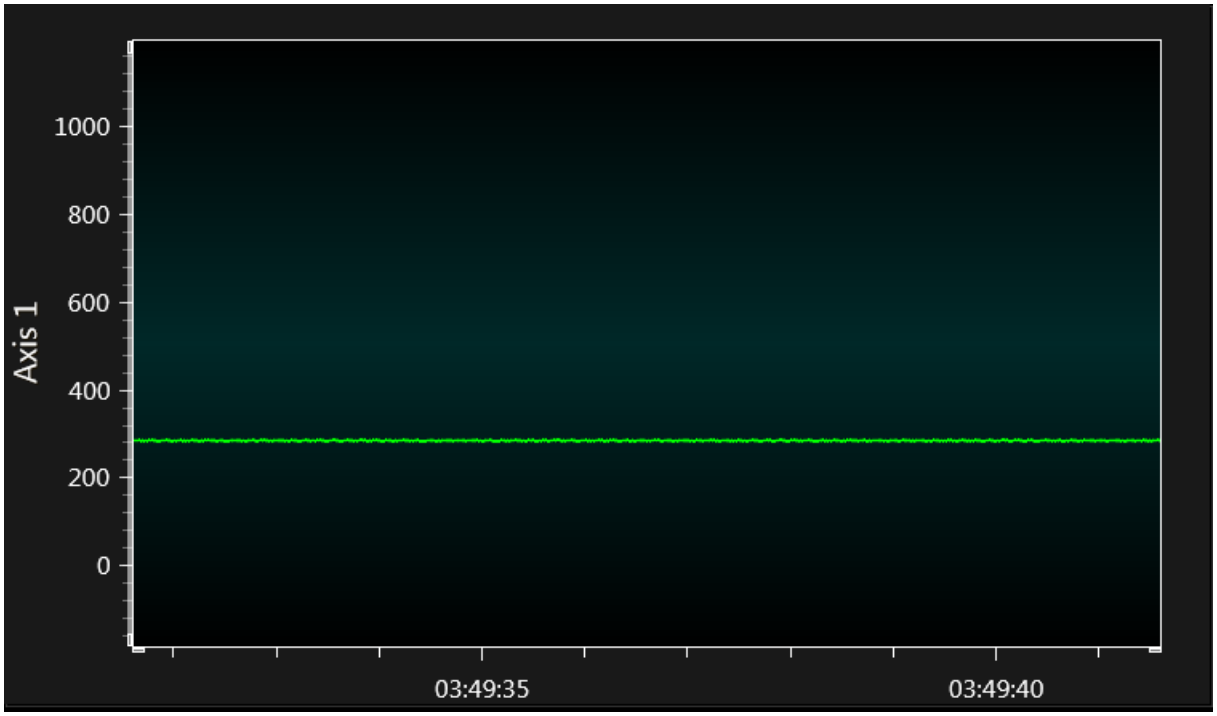
```
#define HARMONIC_NOISE  1
#define ADC_64X_ACCUMULATOR_ENABLE   1
#define SAMPLING_DELAY 5
#define ENABLE_ASDV  0
```

In the test setup, to achieve maximum noise filtering the value of sampling delay (`#define SAMPLING_DELAY`) has been configured to 5. It has been configured using trial and error and it may vary from one setup to another.

When sampling delay is configured to 5, the resulting ADC sampling frequency is 3.33 MHz/(13+ 5) = 46 kHz.

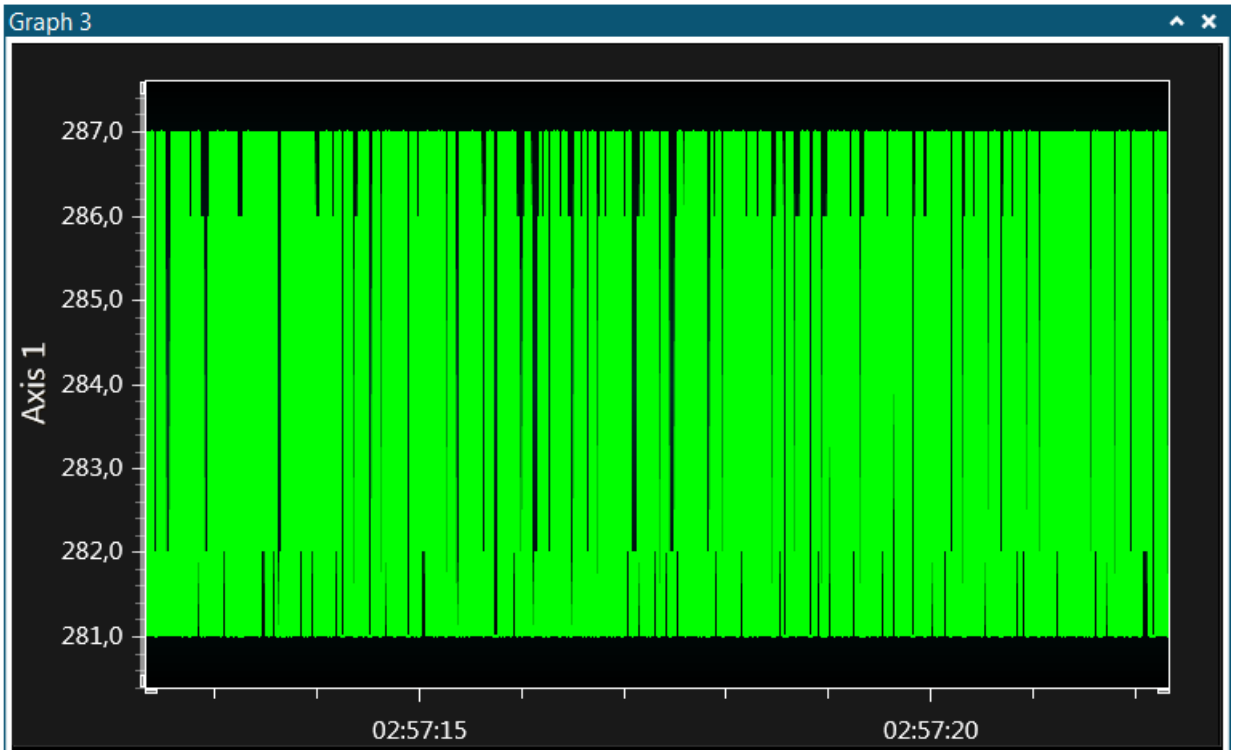
After downloading the code with the above macro definitions, the *Data Visualizer* graph is as shown in Figure 5-14:

Figure 5-14. Periodic Noise: 64-Sample Accumulation and Sampling Delay



If the above signal is zoomed, the image is as shown in Figure 5-15:

Figure 5-15. Zoomed Periodic Noise: 64-Sample Accumulation and Sampling Delay



With oversampling and adding sampling delay, the ADC result count range has been further reduced and the value is varying from 281 to 287. This means the ADC result count range is ±3 counts, which shows that the configured ADC features result in suppressing the noise signal and achieving better ADC results.

6. Get Source Code from Atmel | START

The example code is available through Atmel | START, which is a web-based tool that enables configuration of application code through a Graphical User Interface (GUI). The code can be downloaded for both Atmel Studio and IAR Embedded Workbench® via the direct example code-link below or the *Browse examples* button on the Atmel | START front page.

Atmel | START web page: http://start.atmel.com/

Example Code

- Noise Countermeasures for ADC applications
  - http://start.atmel.com/#example/Atmel:noise_countermeasure:1.0.0::Application:Noise_Countermeasures_for_ADC_Applications:
- Noise Countermeasures for ADC applications with megaAVR 0-series
  - http://start.atmel.com/#example/Atmel:noise_countermeasure_megaavr:1.0.0::Application:Noise_Countermeasures_for_ADC_Applications_with_megaAVR_0-series:

Click *User guide* in Atmel | START for details and information about example projects. The *User guide* button can be found in the example browser, and by clicking the project name in the dashboard view within the Atmel | START project configurator.

Atmel Studio

Download the code as an `.atzip` file for Atmel Studio from the example browser in Atmel | START, by clicking *Download selected example*. To download the file from within Atmel | START, click *Export project* followed by *Download pack*.

Double click the downloaded `.atzip` file and the project will be imported to Atmel Studio 7.0.

IAR Embedded Workbench

For information on how to import the project in IAR Embedded Workbench, open the Atmel | START User Guide, select *Using Atmel Start Output in External Tools*, and *IAR Embedded Workbench*. A link to the Atmel | START User Guide can be found by clicking *Help* from the Atmel | START front page or *Help And Support* within the project configurator, both located in the upper right corner of the page.

7. Appendix A: Plotting Graph in *Data Visualizer*

Note: For detailed information on *Data Visualizer*, refer to the Data Visualizer User's Guide.

In the example source code, the ADC result value is sent through USART to the serial terminal of the *Data Visualizer* and this serial terminal data is fed as input to plot the graph.

The data streamer protocol is used to send the ADC result to serial terminal.

How to use data streamer protocol to send 16-bit value:

The ADC has been configured for 10 bits and this 10-bit ADC result needs to be sent to an 8-bit USART. As one ADC result value will be sent as two bytes, a data streamer protocol has been used to send the ADC result to a serial terminal as below, so that one 16-bit value will be used to plot the graph.

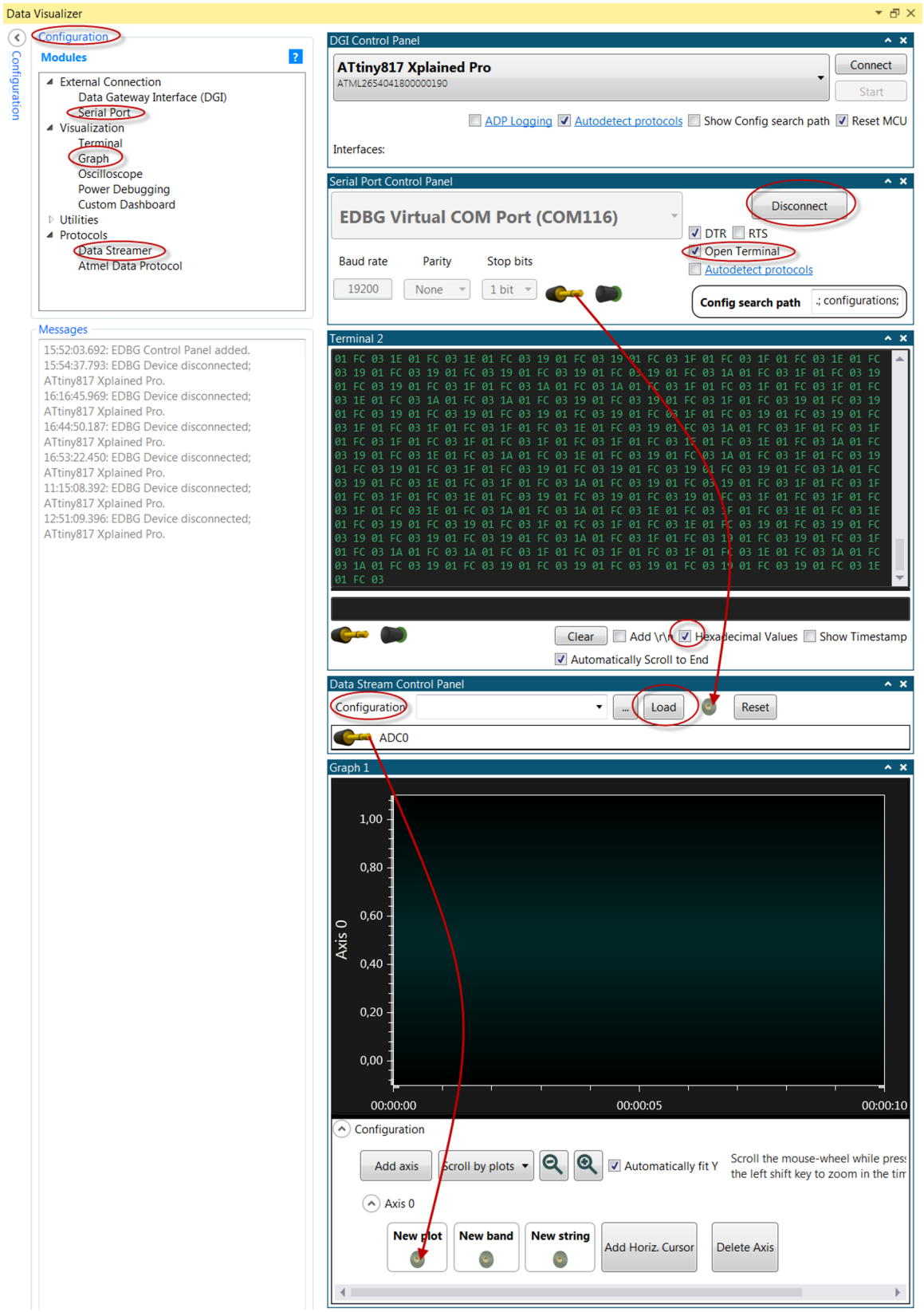
```
USART_0_write(0x03);  //START
USART_0_write(adc_data_LSB);
USART_0_write(adc_data_MSB);
USART_0_write(0xFC); //END
```

***Data Visualizer* Configuration:**

- Open *Atmel Studio → Tools → Data Visualizer*.
- In *Data Visualizer*, open *Configuration → External Connection → Serial Port*.
- Select the EDBG Virtual COM port, and then select *Connect*.
- Open *Configuration → Visualization → Data Streamer*.
- In *Data Stream Control Panel*, under *Configuration*, browse to the configuration .txt file and then select *Load*.
  Note: The `.txt` file to be loaded in for *Data Streamer* has configuration: D, 1, 1, ADC0.

  Note: For more details on *Data Streamer*, refer to the Data Visualizer User's Guide-Data Stream Protocol.
- Open *Configuration → Visualization → Graph*.
- Drag the connections as shown with red arrows in Figure 7-1 to plot the graph.

Figure 7-1. Graph Plotting



To adjust the Y-axis in the graph, follow the points below:

- Under *Configuration* in *Graph*, deselect *Automatically Fit Y*.
- Click somewhere inside the plot area.
- Scroll the mouse-wheel while pressing or holding the Ctrl key.

Note: For more details on *Data Visualizer → Graph*, refer to the Data Visualizer User's Guide-Graph.

8. Revision History

Doc. Rev.	Date	Comments
C	10/2018	Updated figures 1-1, 1-2, 1-3 in chapter "Relevant Devices". Fixed grammar and punctuation.
B	02/2018	Added support for tinyAVR 0-series and megaAVR 0-series.
A	09/2017	Initial document release.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **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 (FAQ), technical support requests, online discussion groups, Microchip consultant 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

Customer Change Notification Service

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

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://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 specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “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

Trademarks

ISBN: 978-1-5224-3672-0

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

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: http://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: 45-4450-2828 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: 49-7131-67-3636 **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