Using Z8 Encore! XP® MCU for RMS Calculation

AN021602-0508

Abstract

This application note discusses an algorithm for computing the Root Mean Square (RMS) value of a sinusoidal AC input signal using the Z8 Encore! XP® MCU. The RMS application uses the internal oscillator of the Z8 Encore! XP® MCU as the system clock. The computed RMS value is displayed in a HyperTerminal application.

The source code file associated with this application note, `AN0216-SC01`, is available on `www.zilog.com`.

Z8 Encore! XP® 4K Series Flash Microcontrollers

Zilog's Z8 Encore! products are based on the new eZ8 CPU and introduce Flash memory to Zilog's extensive line of 8-bit microcontrollers. Flash memory in-circuit programming capability allows for faster development time and program changes in the field. The high-performance register-to-register based architecture of the eZ8 core maintains backward compatibility with Zilog's popular Z8 MCU. Z8 Encore! MCUs combine a 20 MHz core with Flash memory, linear-register SRAM, and an extensive array of on-chip peripherals.

The Z8 Encore! XP® 4K Series of devices support up to 4KB of Flash program memory and 1KB register RAM. An on-chip temperature sensor allows temperature measurement over a range of –40ºC to +105ºC. These devices include two enhanced 16-bit timer blocks featuring PWM and Capture and Compare capabilities. An on-chip Internal Precision Oscillator (5 MHz/32 kHz) can be used as a trimmable clock source requiring no external components. The Z8 Encore! XP® devices include 128 bytes of Non Volatile Data Storage (NVDS) memory where individual bytes can be written or read. The full-duplex UART, in addition to providing serial communications and IrDA encoding and decoding capability, also supports multidrop address processing in hardware.

The rich set of on-chip peripherals make the Z8 Encore! XP® MCUs suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Discussion

Measurement of voltage in an AC circuit can be complicated as compared to voltage measurement in a DC circuit. RMS is a common mathematical method used to define the effective voltage of an AC signal. In general, the RMS value is defined as the square root of the sum of squares of a set of quantities divided by the total number of quantities.

The RMS algorithm discussed in this application note calculates the square root of the sum of squares of the samples of a sinusoidal AC input signal divided by the total number of samples. Accurate calculation of the RMS value depends on the number of samples involved in the calculation. For better accuracy, a greater number of samples must be considered for calculating the RMS value of an input AC signal. There is a trade-off between accuracy and time, because the time required to calculate the RMS value of an input signal increases with the increase in the number of samples.

Theory of Operation

Figure 1 on page 2 illustrates the flow of the RMS algorithm discussed in this application note. The RMS value of a sinusoidal AC input signal is obtained by computing the square root of the average input signal over a period of the AC signal. The following equation is used to compute the RMS value:

RMS = Square Root($(V_1*V_1 + V_2*V_2 + V_3*V_3 + ... + V_{(n-1)}*V_{(n-1)} + V_n*V_n)/n$)

where

- n is the number of samples
- $V_1, V_2, V_3, ... ,V_n$ are the input samples

As described earlier, there is a trade-off between the processing speed of the microcontroller and the output accuracy, as the time required to compute the RMS value increases with the increase in the number of samples.

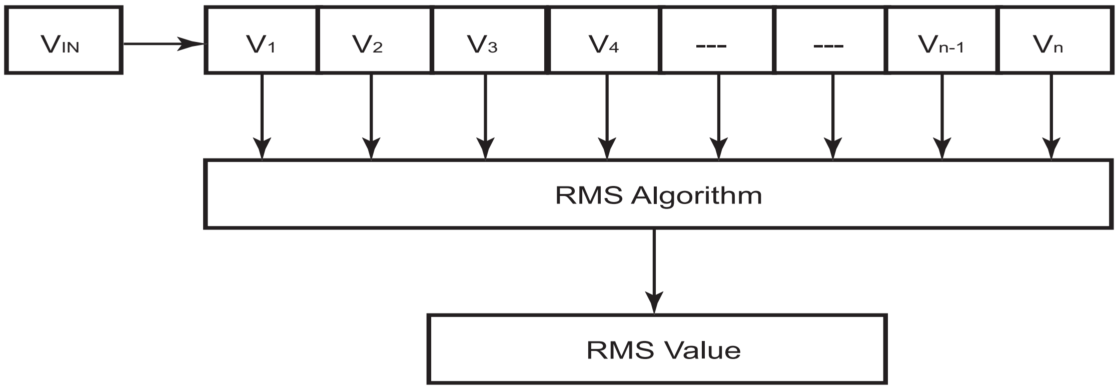

Figure 1. RMS Algorithm Flow

Description of Components

This section lists the external hardware components used in the RMS application, and describes their functionality in detail.

Step-Down Transformer

The RMS application uses a step-down transformer to step-down the input AC voltage. The input to this transformer is line AC, and the output of this transformer is the required step-down AC voltage.

Bridge Rectifier

The RMS application uses a bridge rectifier to convert the sinusoidal AC input wave to a rectified full wave. Figure 2 on page 2 illustrates the output of the bridge rectifier.

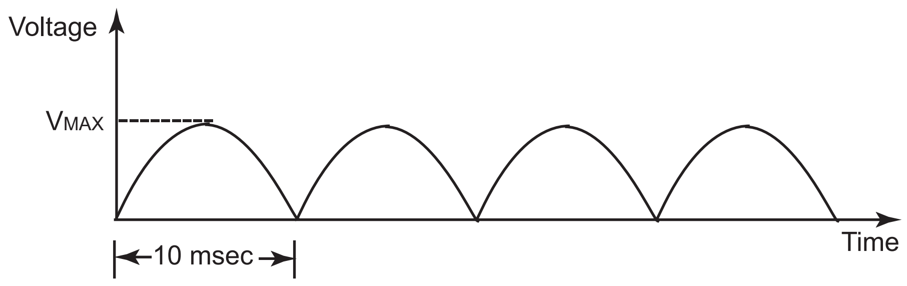

Figure 2. Output of Bridge Rectifier

Potential Divider Network

A potential divider circuit is a network comprising of two resistors, R_1 and R_2, connected in series. Figure 3 on page 3 illustrates a basic potential divider network. R_1 is a fixed resistance, and R_2 is a variable resistance. To fine-tune the output voltage and for better output accuracy, R_2 must be variable. If the output voltage is not critical to the application, then R_2 can be a fixed resistance.

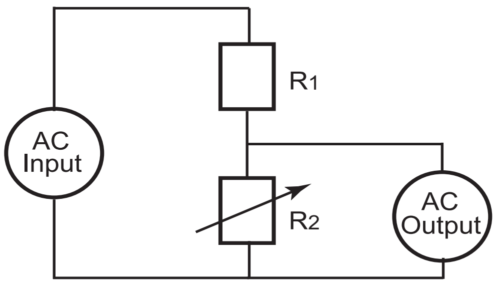

Figure 3. Potential Divider Circuit

Developing the RMS Application with the Z8 Encore! XP® MCU

This section discusses the hardware architecture and software implementation of the RMS application in detail.

Hardware Architecture

Figure 4 on page 3 illustrates a basic block diagram for the RMS application. The application uses a step-down transformer to step-down the line AC input of 0.0-110.0 volts. The output of the step-down transformer is a peak AC voltage of 0.0-6.0 volt. A pulse-shaping block, comprising a bridge rectifier and a potential divider network, converts the negative half-cycle of the step-down output signal to positive. In particular, the potential divider is used to fine-tune the output.

The on-chip ADC peripheral of the Z8 Encore! XP® MCU samples the output of the pulse-shaping block. The Z8 Encore! XP® MCU uses these samples to compute the RMS value of the input AC signal. The computed RMS value of the applied AC voltage is consequently displayed in a HyperTerminal application.

Refer to the schematic provided in Schematic Diagram on page 8 for a detailed connection diagram of the external circuitry.

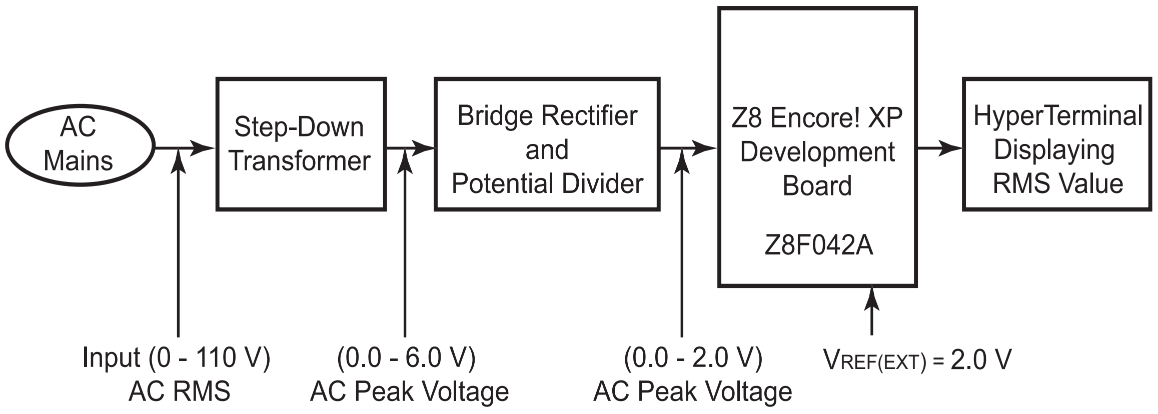
AC Mains
Step-Down Transformer
Bridge Rectifier and Potential Divider
Z8 Encore! XP Development Board
Z8F042A
HyperTerminal Displaying RMS Value
Input (0 - 110 V) AC RMS
(0.0 - 6.0 V) AC Peak Voltage
(0.0 - 2.0 V) AC Peak Voltage
VREF(EXT) = 2.0 V

Figure 4. Block Diagram Illustrating the RMS Application Hardware Setup

Software Implementation

The software program for the RMS application is implemented in four modules: the data sampling module, the data acquisition module, the RMS calculation module, and the RMS display module. Refer to the source code, AN0216-SC01, available on `www.zilog.com`

Data Sampling Module

In the application described in this document, the on-chip ADC peripheral of the Z8 Encore! XP® MCU samples the rectified AC input signal (i.e., the output of the pulse-shaping block). The ADC operates in the continuous mode of operation, and generates samples at the rate of 450us.

To synchronize the speed of the RMS calculation algorithm with the ADC peripheral, the data acquisition module reads only 64 samples per cycle (i.e., the module reads every 8th sample).

The next ADC sample to be read is specified using the following define statement:

```
#define READ_SAMPLE_NO   8
```

The total number of ADC samples read per cycle is specified using the following define statement:

```
#define N   63
```

The right shift value depends on the total number of ADC samples *N*, and is specified using the following define statement:

```
#define RSHIFT    6
```

The value of `RSHIFT` depends on the value of `N` (size of buffer). For example, if

- `N is 127, then RSHIFT is 7`
- `N is 63, then RSHIFT is 6`

The ratio between `N` and `RSHIFT` is $N{:}2^{RSHIFT}$.

Users can modify all of the above define statements to suit application requirements.

Figure 8 on page 10 is a flowchart illustrating the ADC Interrupt Service Routine (ISR).

Data Acquisition Module

The data acquisition module reads data from the ADC High and Low registers, converts this data into one byte, and alternately stores this byte in the even and odd buffers. The buffer size varies in powers of 2 (1^2, 2^2, 4^2, 8^2, etc.), and depends on the number of samples used for RMS calculation. The buffer size, in powers of 2, facilitates the reduction of the processing time required for mathematical operations (multiplication and division, specifically). The number of samples considered in the application described in this document is 8^2, or 64.

Figure 9 on page 11 is a flowchart illustrating the read operation of the ADC samples.

RMS Calculation Module

The RMS calculation module reads the stored samples in the even and odd buffers, alternatively. The RMS calculation module performs the following operations on the ADC samples:

1. Squares the read sample.
2. Divides the squared value by the number of samples (using right shift operation).
3. Adds the value obtained in step 2 to the previous sum.

   Example: sum=sum+new sample
4. Repeats the above steps for all of the buffer contents.
5. Computes the square root of the final value.

The total time required to calculate the RMS value is less than 14ms at a frequency of 5.5MHz.

Figure 10 on page 12 is a flowchart illustrating the RMS calculation algorithm.

RMS Display Module

The RMS display module converts the calculated RMS value to ASCII, and displays this value in the HyperTerminal application.

Testing

This section lists the setup and equipment used to test the RMS application. The test results obtained are also listed in Table 1 on page 6.

Setup

A basic setup to test the RMS algorithm using the Z8 Encore! XP® MCU is illustrated in Figure 5 on page 5. The setup comprises of the input block, the Z8 Encore! XP® 4K Series Development Kit (Z8F04A28100KIT), and the HyperTerminal application. The input block consists of a step-down transformer, a bridge rectifier, and a potential divider network.

Refer to the schematic provided in Schematic Diagram on page 8 for a detailed connection diagram of the input block.

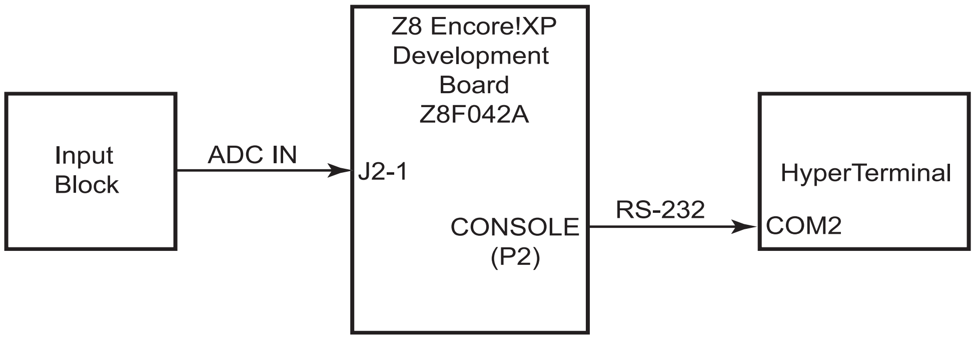

Figure 5. Test Setup for the RMS Application

Equipment Used

- Z8 Encore! XP® 4K Series Development Kit (28-pin) with full ANSI C Compiler (Z8F04A28100KIT)
- Zilog Developer Studio II - Z8 Encore! (ZDS II-IDE)
- A PC equipped with the HyperTerminal application configured to the following settings:
  - 38400 bps baud rate
  - 8 data bits
  - No parity
  - One stop bit
  - No flow control

Results

Table 1 on page 6 lists the readings of the input AC (RMS) voltage, the output AC (RMS) voltage, and the percentage error in the calculation of the RMS value using the Z8 Encore! XP® MCU.

A variable transformer was used to vary the input AC voltage, and the corresponding output was measured. The percentage error in the computation of the RMS value was found to be negligible. The performance was as expected.

Table 1. Observation Table

Input AC (RMS) in Volts	Output AC (RMS) in Volts	Percentage Error (%)
120	120	0.0
115	115	0.0
110	110	0.0
105	105	0.9
100	100	0.0
95	95	0.0
90	90	0.0
85	84	1.1
80	79	1.2
75	74	1.3
70	69	1.4
65	64	1.5
60	59	1.6
55	54	1.8

Note: *The percentage error in the calculation of the RMS value can be further minimized by using improved circuitry at the ADC end. The voltage drop in the AC main must vary linearly with the input to the ADC for zero percent error.*

Summary

The Z8 Encore! XP® MCU features the UART and the ADC ports for communication. The module-based software implementation allows users to directly use the code or modify the code with ease. The software modules are modifiable, and are usable with other microcontrollers of the Z8 Encore! family.

In the application described in this document, the ADC peripheral of the Z8 Encore! XP® MCU operates in the continuous mode of operation, and produces a final RMS value based on 8-bit data at a frequency of 5.5 MHz. The RMS algorithm is very simple, and uses the Root Mean Square mathematical method to calculate the voltage of a sinusoidal AC input signal.

Appendix A—References

Further details about the Z8 Encore! products can be found in the references listed in Table 2.

Table 2. List of References

Topic	Document Name
eZ8® CPU	eZ8® CPU User Manual (UM0128)
Z8 Encore! XP® 4K Series Microcontrollers	Z8 Encore! XP® 8K and 4K Series Product Specification (PS0228)
	Z8 Encore! XP® F042A Series Development Kit User Manual (UM0166)
ZDS II-IDE	Zilog Developer Studio II-Z8 Encore! User Manual (UM0130)

Appendix B—Schematic Diagram

Figure 6 illustrates a schematic diagram for the RMS application interface described in this application note.

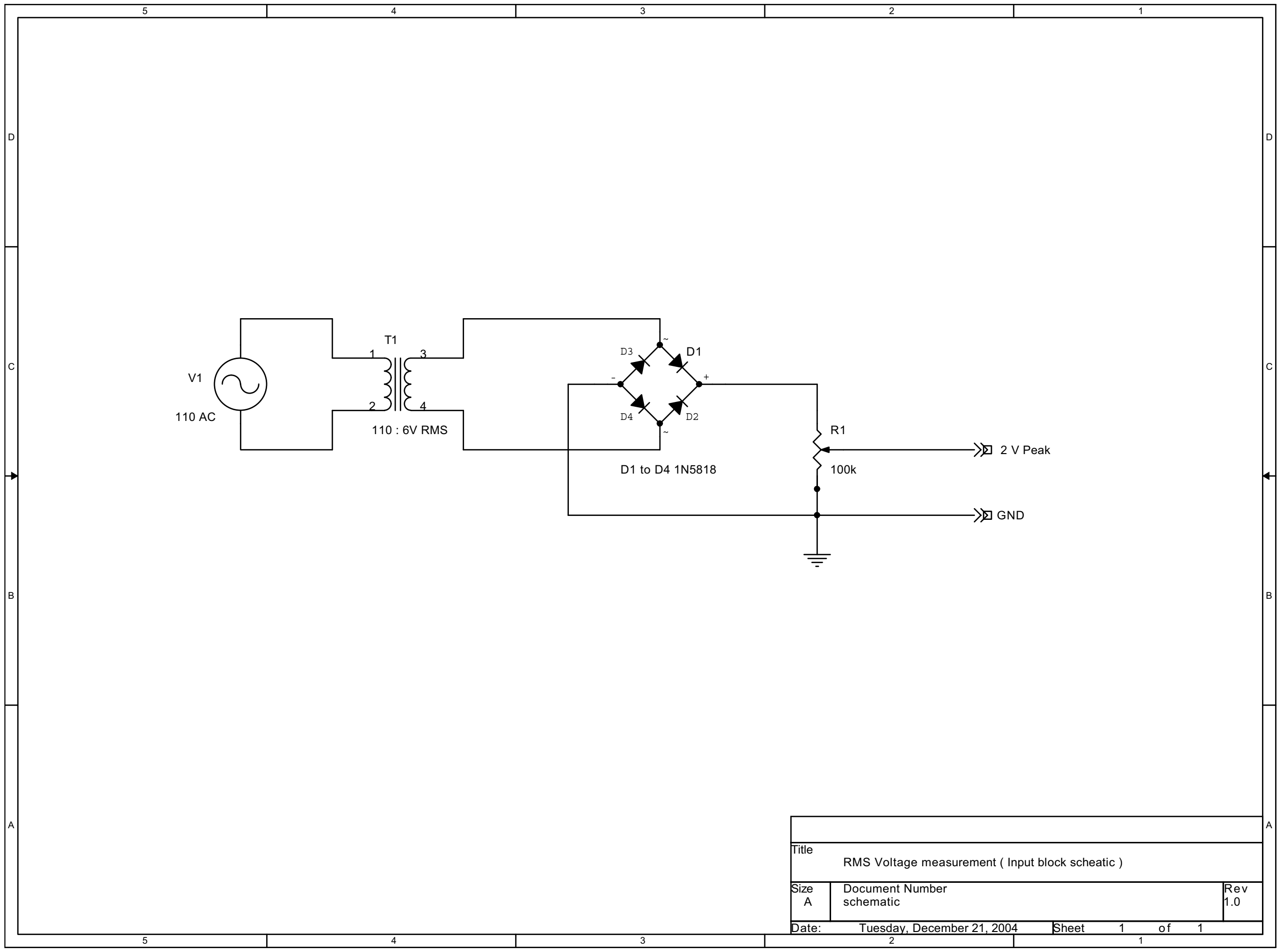

Figure 6. Schematic Illustrating RMS Application Interface

Appendix C—Flowcharts

This appendix provides flowcharts for the RMS application described in this document. Figure 7 is a flowchart for the main routine of the RMS algorithm in which the calculated RMS value is displayed in the HyperTerminal application.

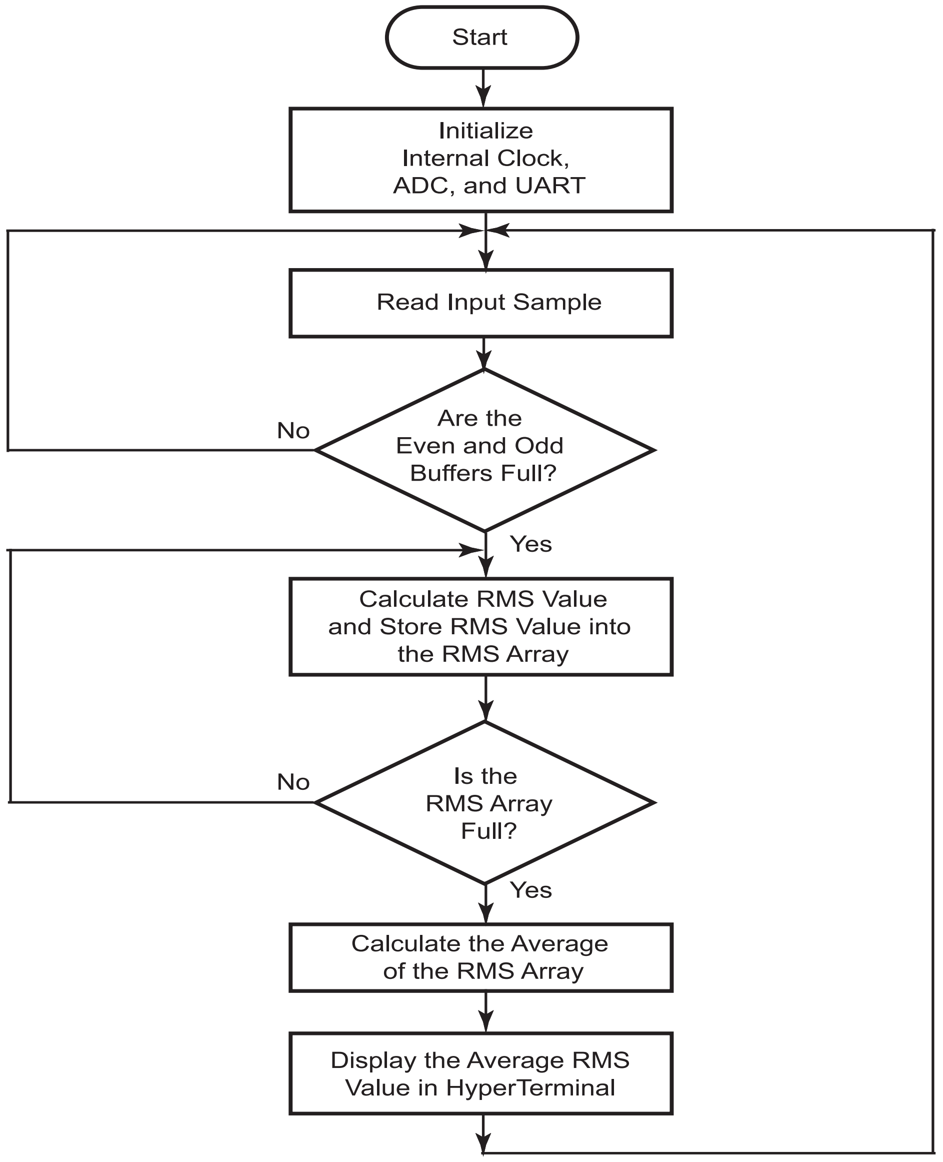
Start
Initialize Internal Clock, ADC, and UART
Read Input Sample
Are the Even and Odd Buffers Full?
No
Yes
Calculate RMS Value and Store RMS Value into the RMS Array
Is the RMS Array Full?
No
Yes
Calculate the Average of the RMS Array
Display the Average RMS Value in HyperTerminal

Figure 7. Main Routine

Figure 8 is flowchart illustrating the ADC interrupt service routine.

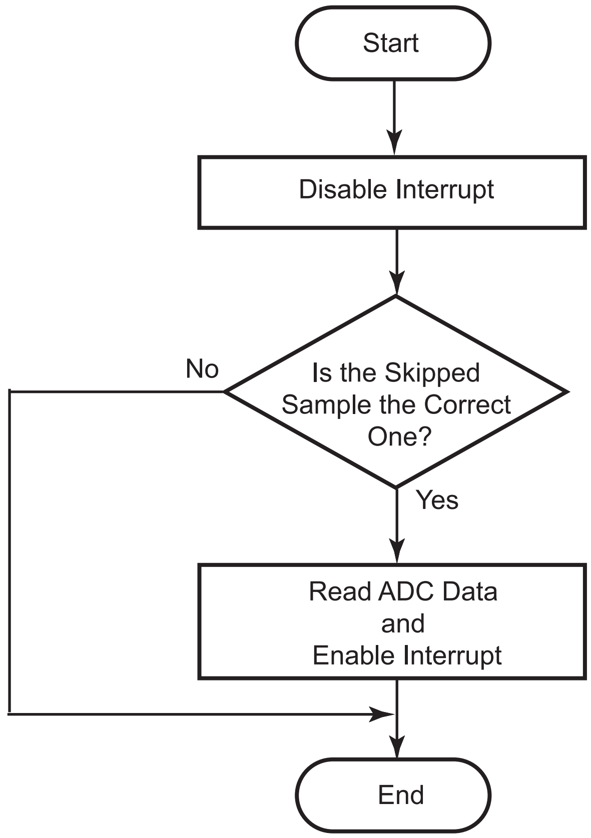
Start
Disable Interrupt
Is the Skipped Sample the Correct One?
No
Yes
Read ADC Data and Enable Interrupt
End

Figure 8. ADC Interrupt Service Routine

Figure 9 is a flowchart to read the ADC samples.

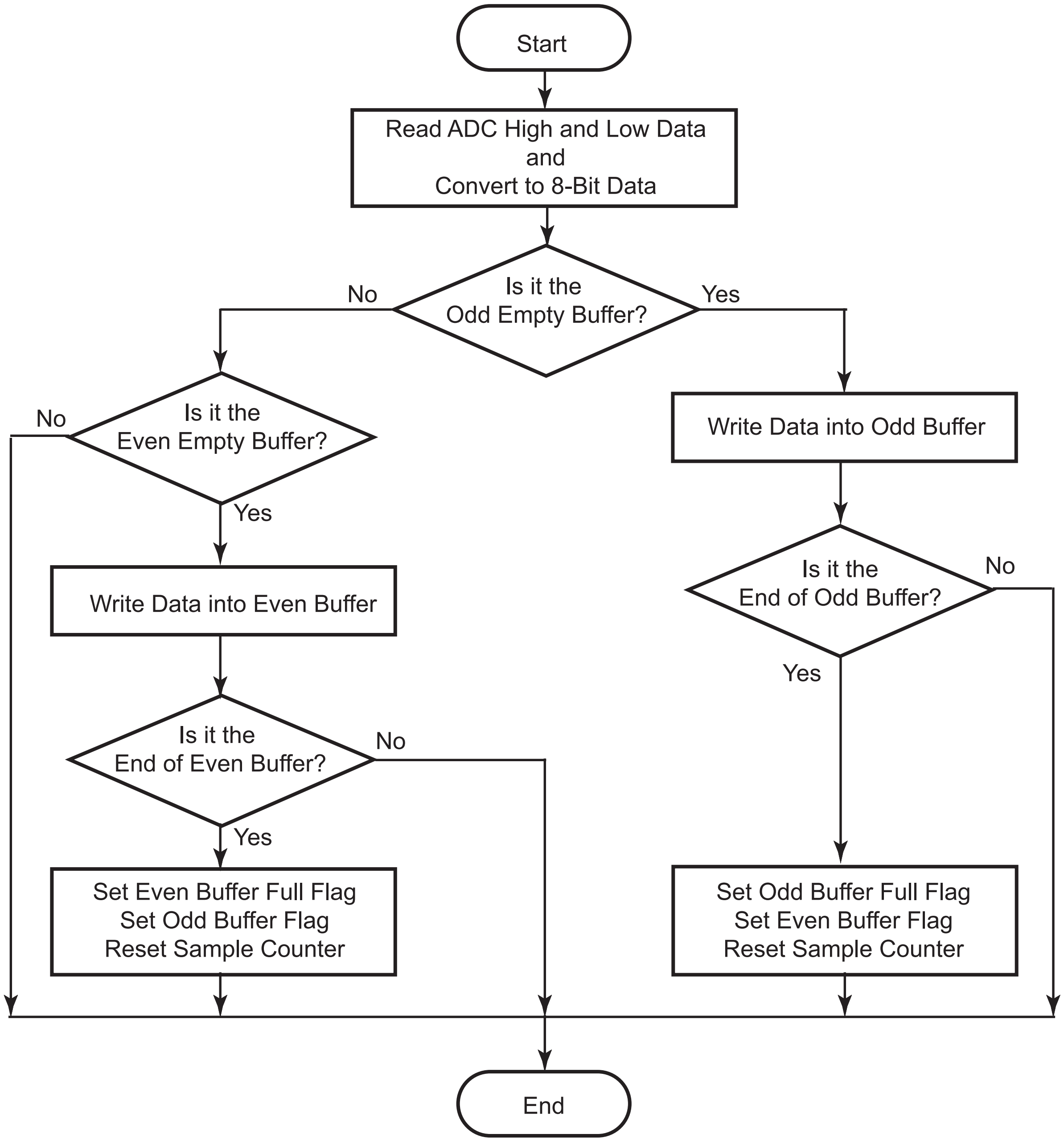
Start
Read ADC High and Low Data and Convert to 8-Bit Data
Is it the Odd Empty Buffer?
No
Yes
Is it the Even Empty Buffer?
No
Yes
Write Data into Odd Buffer
Write Data into Even Buffer
Is it the End of Odd Buffer?
No
Yes
Is it the End of Even Buffer?
No
Yes
Set Even Buffer Full Flag
Set Odd Buffer Flag
Reset Sample Counter
Set Odd Buffer Full Flag
Set Even Buffer Flag
Reset Sample Counter
End

Figure 9. Routine to Read ADC Samples

Figure 10 illustrates the RMS calculation algorithm in which the RMS mathematical method is applied to the input data stored in either the odd or the even buffer.

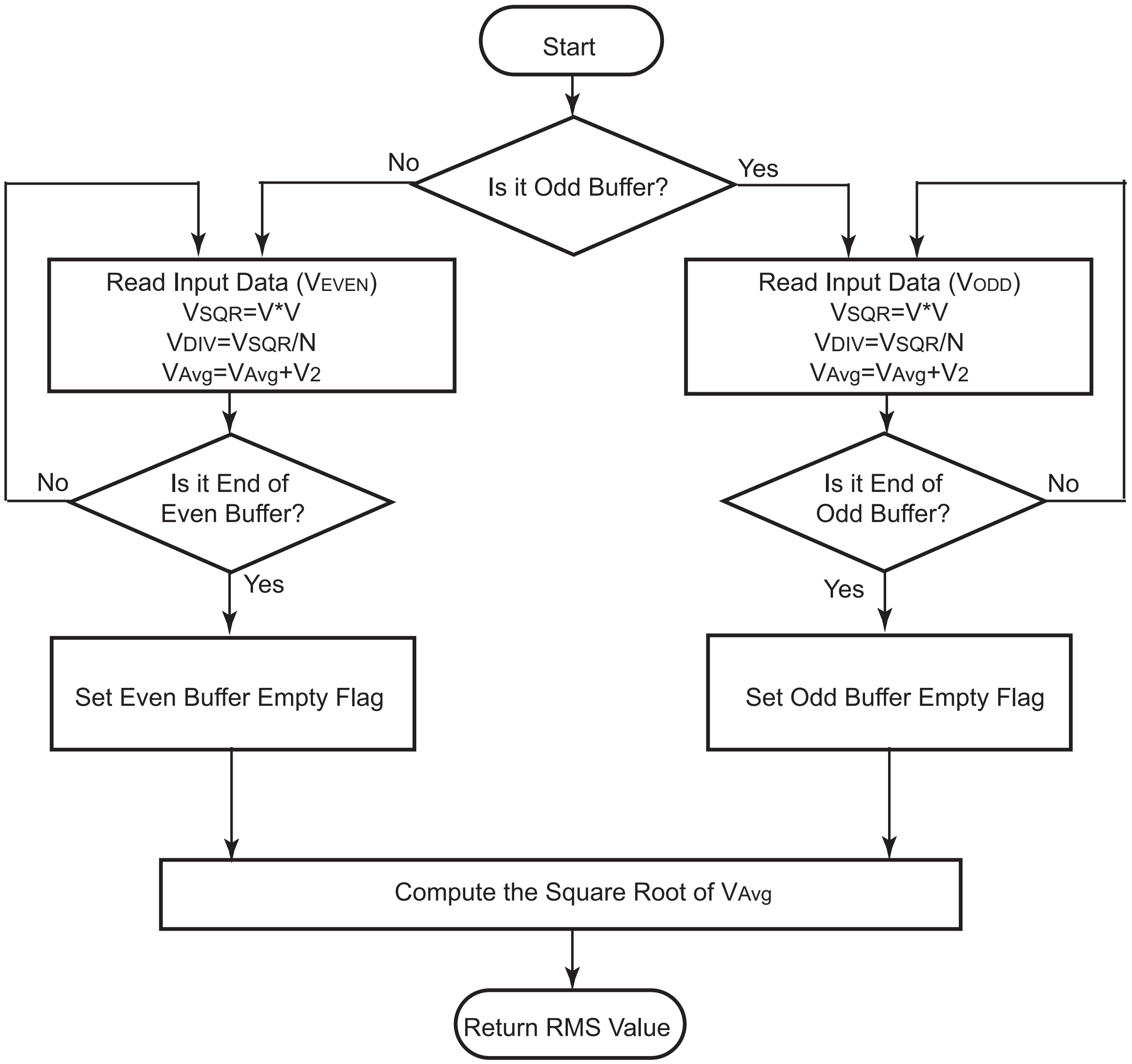

Figure 10. RMS Calculation Algorithm

Warning: DO NOT USE IN LIFE SUPPORT