Application Note

Performing Data Conversions Using Z8F64xx Series MCUs

AN035201-1212

Abstract

This application note demonstrates four methods for converting data from one format to another, and provides an API for each type of conversion. These APIs are subsequently ready to use in application-level programming. The four conversion methods are:

- Unsigned char/int/long to hexadecimal string, and vice versa
- Unsigned char/int/long to decimal string, and vice versa
- Signed char/int/long to decimal string, and vice versa
- Unsigned char/int/long to BCD, and vice versa

Note: The source code file associated with this application note, AN0352-SC01.zip, is available for download on zilog.com. This source code has been tested with version 5.0.0 of ZDSII for Z8 Encore! MCUs. Subsequent releases of ZDSII may require you to modify the code supplied with this application note.

Discussion

Data conversion refers to the process of translating data from one format to another. It is one of the common routines that are typically required in any application development scenario. For example, temperature data taken from a sensor must be shown in a display module such as an LCD panel or HyperTerminal. If this data is not converted, the displayed data will be unreadable because the raw data has not been converted to alphanumeric, human-readable characters; this crucial step in the scenario is where data conversion fits in.

The remainder of this section discusses the four data conversion methods. All data conversion APIs are described in Table 1 on page 4.

Unsigned to Hexadecimal String Conversion and Vice Versa

Displaying data in hexadecimal notation makes for an easy representation of binary numbers. To convert a numbered value to its hexadecimal string equivalent, observe the following procedure.

1. Divide the number by 16.
2. Get the remainder.
3. Convert the remainder to ASCII.

a. If the remainder is less than 10, add `0x30`. Otherwise, subtract 10, then add `0x41`.
b. Store the converted value into an array.

4. Repeat steps 1 through 3 until the number becomes zero.
5. Reverse the resulting array.

In hexadecimal string representation, two characters represent one hexadecimal value. To convert a hexadecimal string to its equivalent value, observe the following procedure.

1. Convert the first character to unsigned value. This value will become the upper nibble of the unsigned value.
   a. If the character is a digit (i.e., a value from 0 to 9), subtract `0x30`.
   b. Otherwise (i.e., if the character is a letter), subtract `0x41`, then add 10.
2. Convert the second character to an unsigned value. This value will become the lower nibble of the unsigned value.
3. Join the upper and lower nibbles; then append the acquired value as the LSB of the output value.
4. Repeat steps 1 through 3 for every succeeding two characters.

Unsigned to Decimal String Conversion and Vice Versa

Displaying data in decimal notation provides an easy-to-read mathematical value as a result of an operation or a reading taken from a sensor. Observe the following steps when converting data to its decimal string equivalent.

1. Divide the number by 10.
2. Get the remainder.
3. Convert the remainder to ASCII.
   a. Add `0x30` to the remainder.
   b. Store the converted value into an array.
4. Repeat steps 1 through 3 using the quotient from step 1 until the number becomes zero.
5. Reverse the resulting array.

To convert a decimal string back to its numerical value, perform the following steps:

1. Start at the last character in the string.
2. Initialize a factor to 1 to represent the current place value of the character being converted.
3. Get the lower nibble of the current character in the string and multiply it by the current value of the factor.
4. Add this acquired value to the output value.

5. Increase the value of the factor by a multiple of 10.
6. Repeat steps 3 through 5 for each succeeding character until the start of the array is reached.

Signed to Decimal String Conversion and Vice Versa

To convert a signed value to its decimal string equivalent, observe the following steps:

1. If the value to convert is negative, get its positive equivalent by multiplying it by –1.
2. Convert the value as unsigned data.

To convert a signed decimal string to its equivalent signed value, observe the following steps:

1. Take note of the sign of the value, which is the first character in the string.
2. Convert the value as an unsigned decimal string, using the string beginning at the second character.
3. If the sign acquired in step 1 is negative, multiply the result in step 2 by –1.

Unsigned to Binary Coded Decimal Conversion and Vice Versa

Observe the following procedure to convert an unsigned integer to its BCD equivalent.

1. Take note of the place value; start at the ones digit.
2. Divide the number by 10.
3. Get the remainder.
4. Shift the remainder by the place value and multiply by 8 (which is the number of bits per place value).
5. Add the result to the output value.
6. Repeat steps 2 through 5 until the number becomes zero.

Observe the following procedure to convert a BCD value to its equivalent unsigned integer value.

1. Initialize a factor to 1 to represent the current place value of the BCD data being converted.
2. Get the LSB and multiply it by the current factor value.
3. Add the result to the output value.
4. Increase the value of the factor by a multiple of 10.
5. Shift the BCD value to the right by 8.
6. Repeat steps 2 through 5 until the BCD value becomes zero.

Appendix A. APIs

Table 1 describes the APIs for each of the four data conversion methods.

Table 1. Data Conversion APIs

Function Name	Description
Unsigned to Hexadecimal String Conversion and Vice Versa	
ultoha	Converts an unsigned long value to its hexadecimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
uitoha	Converts an unsigned int value to its hexadecimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
uctoha	Converts an unsigned char value to its hexadecimal string equivalent and returns the number of characters written in the array excluding the null terminating character.
hatoul	Returns the unsigned long equivalent of the given hexadecimal string.
hatoui	Returns the unsigned int equivalent of the given hexadecimal string.
hatouc	Returns the unsigned char equivalent of the given hexadecimal string.
Unsigned to Decimal String Conversion and Vice Versa	
ultoda	Converts an unsigned long value to its decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
uitoda	Converts an unsigned int value to its decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
uctoda	Converts an unsigned char value to its decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
datoul	Returns the unsigned long equivalent of the given decimal string.
datoui	Returns the unsigned int equivalent of the given decimal string.
datouc	Returns the unsigned char equivalent of the given decimal string.
Signed to Hexadecimal String Conversion and Vice Versa	
ltoda	Converts a signed long value to its signed decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
itoda	Converts a signed int value to its signed decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
ctoda	Converts a signed char value to its signed decimal string equivalent and returns the number of characters written in the array, excluding the null terminating character.
datosl	Returns the signed long equivalent of the given decimal string.
datosi	Returns the signed int equivalent of the given decimal string.
datosc	Returns the signed char equivalent of the given decimal string.
Unsigned to BCD Conversion and Vice Versa	
uitobcd	Returns an unsigned long BCD equivalent of the given unsigned int value.
bcdtoui	Returns an unsigned int equivalent of the given BCD value.

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

©2012 Zilog, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z8, Z8 Encore! and Z8 Encore! XP are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.