



Designer or ViSi String Class Functions

DOCUMENT DATE: 7th May 2020
DOCUMENT REVISION: 1.1

Description

This Application Note is dedicated to providing the reader a simple and straight forward documentation about String Functions. The 4DGL code of the Designer project can be copied and pasted to an empty ViSi project and it will compile normally. The code can also be integrated to that of an existing ViSi project.

Before getting started, the following are required:

- Any of the following 4D Picaso display modules:

[uLCD-24PTU](#) [uLCD-32PTU](#) [uLCD-43\(P/PT/PCT\)](#)
[uLCD-28PTU](#) [uLCD-32WPTU](#) [uVGA-III](#)

and other superseded modules which support the Designer and/or ViSi environments.

- The target module can also be a Diablo16 display

[uLCD-35DT](#) [uLCD-70DT](#)

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor.

- [4D Programming Cable](#) or [uUSB-PA5](#)
- [micro-SD \(uSD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)

Content

Description	2
Content	2
Application Overview	3
Setup Procedure	3
Create a New Project	3
Introduction to String Functions	4
<i>Getting Acquainted with the 4DGL String Functions.....</i>	<i>4</i>
<i>Description of the String Class Functions.....</i>	<i>4</i>
Design the Project.....	5
<i>Numbers</i>	<i>5</i>
<i>Words or Byte</i>	<i>6</i>
<i>Characters</i>	<i>6</i>
<i>Appending of Strings.....</i>	<i>7</i>
<i>Putting of Bytes.....</i>	<i>7</i>
Run the Program.....	8
Proprietary Information	9
Disclaimer of Warranties & Limitation of Liability	9

Application Overview

This application note is intended to letting the reader understand the fundamental usage of the 4DGL String Functions through simple and straight forward exemplification. In addition, this application note also gives the reader the basic usage of these functions alongside loops and condition routines.

The underlying architecture of the Picaso and Diablo16 processors is 16-bits, this extends to the addresses used to access memory. This means that for strings, normal addresses and pointers cannot be used as strings need to be addressed in 8 bit increments.

String pointers and normal (word) pointers are not interchangeable. Normal (word) pointers can be converted to string pointers, using the `str_Ptr()` function. String pointers cannot be converted to word pointers.

Functions that use string pointers begin with `str_` (with the sole exception of `str_ptr`), also filenames in `file_` functions expect string pointers, but since one normally uses constants for this, you don't see `str_Ptr()` used much in those functions.

Setup Procedure

For instructions on how to launch Workshop 4, how to open a **Designer** project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of the application note

[Designer Getting Started - First Project](#)

For instructions on how to launch Workshop 4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of the application note

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

Create a New Project

For instructions on how to create a new **Designer** project, please refer to the section “**Create a New Project**” of the application note

[Designer Getting Started - First Project](#)

For instructions on how to create a new **ViSi** project, please refer to the section “**Create a New Project**” of the application note

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

Introduction to String Functions

Getting Acquainted with the 4DGL String Functions

String Class Functions

Summary of Functions in this section:

- `str_Ptr(&var)`
- `str_GetD(&ptr, &var)`
- `str_GetW(&ptr, &var)`
- `str_GetHexW(&ptr, &var)`
- `str_GetC(&ptr, &var)`
- `str_GetByte(ptr)`
- `str_GetWord(ptr)`
- `str_PutByte(ptr, val)`
- `str_PutWord(ptr, val)`
- `str_Match(&ptr, *str)`
- `str_MatchI(&ptr, *str)`
- `str_Find(&ptr, *str)`
- `str_FindI(&ptr, *str)`
- `str_Length(ptr)`
- `str_Printf(&ptr, *format)`
- `str_Cat(&destination, &Source)`
- `str_CatN(&ptr, str, count)`
- `str_ByteMove(src, dest, count)`
- `str_Copy(dest, src)`
- `str_CopyN(dest, src, count)`

Description of the String Class Functions

`str_Ptr(&var);`

Return a byte pointer to a word region.

`str_GetD(&ptr, &var);`

Convert number in a string to DWORD and put result in var

`str_GetW(&ptr, &var);`

Convert number in a string to WORD and put result in var.

`str_GetHexW(&ptr, &var);`

Convert hex number in a string to WORD and put result in var

`str_GetC(&ptr, &var);`

Get next valid ASCII char in a string to a variable var.

`str_GetByte(ptr);`

Get a byte from a string in a pointer ptr

`str_PutByte(ptr, value);`

Put a byte value into a string buffer at ptr

`str_PutWord(ptr, value);`

Put a word value into a byte buffer at ptr

`str_Match(&ptr, *str);`

Compares the string at position ptr in a string buffer to the string str, skipping over any leading spaces if required. If a match occurs, ptr is advanced to the first position past the match, else ptr is not altered. (**case sensitive matching)

`str_Find(&ptr, *str);`

Searches for string str in string buffer pointed to by ptr. (** case sensitive matching)

`str_Length(ptr);`

Returns the length of a string excluding terminator.

str_Printf(&ptr, *format);

This function prints a formatted string from elements derived from a structured byte region. There is only one input argument, the byte region pointer ptr which is automatically advanced as the format specifier string is processed.

Format specifiers:

%c	character	%s	string of characters
%d	signed decimal	%u	unsigned decimal
%lu	long unsigned decimal	%x	hex byte
%X	hex word	%lX	hex long
%b	binary word	%lb	long binary word

str_Cat(&destination, &source);

Appends a copy of the source string to the destination string.

str_CatN(&ptr, str, count);

The number of characters copied is limited by "count".

str_ByteMove(src, dest, count);

Copy bytes from "src" to "dest", stopping only when "count" is exhausted.

str_Copy(dest, src);

Copy a string from "src" to "dest", stopping only when the end of source string "src" is encountered (0x00 terminator).

str_CopyN(dest, src, count);

Copy a string from "src" to "dest", stopping only when "count" is exhausted, or end of source string "str" is encountered (0x00 string terminator).

Design the Project

Numbers

// CONVERT NUMBERS IN A STRING TO WORD AND DISPLAY RESULT

```

var buffer[50]; // 100 character buffer for a source string
var vars[3];    // variable array for results
var p;          // string pointer
var n;

func main()
gfx_ScreenMode(PORTRAIT);
txt_MoveCursor(3,0);

print("CONVERT NUMBER IN A STRING TO WORD AND DISPLAY RESULT");
to(buffer); print("0x7834 0b11011001 12345 datum"); //stream data from print
                                                    //function to variable array
                                                    //buffer
p := str_Ptr(buffer);                            // raise string pointer for the
                                                    //string functions

while(str_GetW(&p, &vars[n++]) != 0); //convert all numerical value from string
//pointer p into a word and save to vars until
// non-numerical is detected

txt_MoveCursor(6,12);
print(vars[0],"\n", vars[1],"\n", vars[2],"\n"); // display all numerical value from
                                                    // result of numerical detection
                                                    //values are in decimal format
str_Printf (&p, "%s\n" );                        // numbers extracted, now just print
                                                    // remainder of string

txt_MoveCursor(12,13);
str_Printf(&p, "\nRemainder\n%s" );              //display remainder of
                                                    // string in pointer p

repeat
forever
endfunc

```

Words or Byte

// INSERTING AND GETTING OF WORDS FROM A STRING

```
var p;                                // string pointer variable
numbers[10];                          // array for 20 characters
var numbers[10];

func main()

    p:= str_Ptr(numbers);              // raise a string pointer
    str_PutWord (p + 1, 10);           // put a word value of 100 in string
                                        // pointer p @ the 2nd byte
    str_PutWord (p + 3, 400);          // put a word value of 100 in string
                                        // pointer p @ the 3rd byte

    print( str_GetWord( p + 1), "\n" ); // 'peek' the array for the byte @
                                        //2 and 3 and display after
    print( str_GetWord( p + 3), "\n" );

repeat
forever

endfunc
```

Characters

// READING OF CHARACTERS FROM A STRING

```
var p;                                // string pointer
var n;
var char;
var buffer[100];                      // 200 character buffer for a source string

func main()

    to(buffer); print("4D SYSTEMS");
    p := str_Ptr(buffer);              // raise a string pointer p for
                                        //strings in array buffer

    while(str_GetC(&p, &char))
        print("\np=", p, " char is ", [CHR] char); // use the print characters
    wend

    print("\nEnd of string");

repeat
forever

endfunc
```

Appending of Strings

// SAMPLE PROGRAM TO DEMONSTRATE SIMPLE APPENDING OF STRING

```

var buffer1[100], buffer2[100];
var p,q;

func main()
    txt_MoveCursor(0,1);
    print("PROGRAM FOR APPENDING STRING" );
    to(buffer1); print(" 4D SYSTEMS");           // save string to buffer1
    txt_MoveCursor(3,0);
    q:=str_Ptr(buffer1);                         // assign q as pointer
    print("VALUE OF BUFFER1:");
    txt_MoveCursor(5,5);
    str_Printf(" ", buffer1);

    to (buffer2); print(" TECHNOLOGY" );         //save string to buffer2
    txt_MoveCursor(7,0);
    print("VALUE OF BUFFER2:");
    p := str_Ptr (buffer2);                     //assign p as pointer
    txt_MoveCursor(9,5);
    str_Printf(" ", buffer2);

    txt_MoveCursor(20,0);
    print("APPEND BUFFER1 W/ BUFFER2 ");
    str_Copy(q +10, p);                         // copy buf2 to buf1.
                                                // Position to 11th of buf2
    str_Printf(&q, "\n\n%s");                   // print all characters in pointer q

repeat
forever
endfunc

```

Putting of Bytes

// PUTTING CHARACTERS/BYTES IN STRING POINTER

```

func main()
var buffer[100]; // 200 character buffer for a source string
var n, p;

p:= str_Ptr(buffer); //assign p as pointer to buffer

str_PutByte(p++, 'S'); // put char S in position zero of buffer
str_PutByte(p++, 'Y'); // offset placement of Y in p by 1
str_PutByte(p++, 'S' ); // offset placement of Y in p by 2
str_PutByte(p++, 'T' ); // offset placement of Y in p by 3
str_PutByte(p++, 'E' ); // offset placement of Y in p by 4
str_PutByte(p++, 'M' ); // offset placement of Y in p by 5
str_PutByte(p++, 0); // terminate string with 0
txt_MoveCursor(3, 5);
print([STR] buffer); //display string content of buffer

repeat
forever
endfunc

```

Run the Program

For instructions on how to save a **Designer** project, how to connect the target display to the PC, how to select the program destination, and how to compile and download a program, please refer to the section “**Run the Program**” of the application note

[Designer Getting Started - First Project](#)

For instructions on how to save a **ViSi** project, how to connect the target display to the PC, how to select the program destination, and how to compile and download a program, please refer to the section “**Run the Program**” of the application note

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

The uLCD-32PTU and uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication 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.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.