

```

        if (numBytesInCharacter <= 0)
            Abort "Bug in CharactersInUTF8String"      // Should not happen
        endif
        numCharacters += 1
        byteOffset += numBytesInCharacter
    while(1)

        return numCharacters
End

Function/S UTF8CharacterAtPosition(str, charPos)
    String str
    Variable charPos

    if (charPos < 0)
        // Print "charPos is invalid"                  // For debugging only
        return ""
    endif

    Variable length = strlen(str)
    Variable byteOffset = 0
    do
        if (byteOffset >= length)
            // Print "charPos is out-of-range"          // For debugging only
            return ""
        endif
        if (charPos == 0)
            break
        endif
        Variable numBytesInCharacter = NumBytesInUTF8Character(str, byteOffset)
        byteOffset += numBytesInCharacter
        charPos -= 1
    while(1)

        numBytesInCharacter = NumBytesInUTF8Character(str, byteOffset)
        String result = str[byteOffset, byteOffset+numBytesInCharacter-1]
        return result
End

Function DemoCharacterByCharacter()
    String str = "<string containing non-ASCII text>

    Variable numCharacters = UTF8CharactersInString(str)
    Variable charPos

    for(charPos=0; charPos<numCharacters; charPos+=1)
        String character = UTF8CharacterAtPosition(str, charPos);
        Printf "CharPos=%d, char=\"%s\"\r", charPos, character
    endfor
End

```

See Also: [Text Encodings](#) on page III-459, [String Indexing](#) on page IV-16

Working With Binary String Data

Although Igor strings were originally conceived to store human-readable text such (e.g., "Hello, World"), advanced users have found them useful as containers for binary data. Such binary data is usually read from binary files, obtained from the Internet (e.g., via [FetchURL](#) or [URLRequest](#)), or obtained via data acquisition.