文字列パッケージ

Str_Pack

説明書

2009 年 10 月 15 日　　1.07 版

版　数　管　理　表

版数	日　付	変　更　内　容
1.00	2005.12.7	初版
1.03	2006.7.11	strini,get_cmd_param を追加
1.04	2008.3.5	strdup を追加、strpack_ver 返却値の誤記を訂正
	2008.3.10	readchr, writechr を追加
1.05	2008.5.19	readstr に ret_cd 追加、bool2str を追加
1.06	2009.2.3	writestr に２系統出力を追加(例 ファイルと STD_OUTPUT) writestr0 を追加
1.07	2009.10.15	１章、２章を全面リライト iskanji を追加

目 次

1. はじめに

Str_Packとは VHDL String型の変数に対してC言語ライクな操作を可能にするprocedureと functionをまとめたパッケージで、Model*Sim*上で動作します。本書はその使用方法を説明します。

Str_Pack は Model*Sim* 5.8e, Model*Sim* 6.1f, Model*Sim* 6.5c(いずれも OS は Windows XP SP.2)での動作を確認しています。

2. 使用例(サンプルテストベンチ)の実行方法

以下の手順で同梱の使用例を実行することができます。

1．アーカイブファイルを解凍し、任意のディレクトリーに sim をフォルダごとコピーする。

sim 配下のフォルダー構成

```
sim              … サンプルソースファイル、起動用マクロファイル格納フォルダー
  ├── str_pack   … 文字列パッケージ本体格納フォルダー
  ├── data_r     … サンプルがリードするファイル格納フォルダー
  └── data_w     … サンプルがライトするファイル格納フォルダー
```

2．Model*Sim*を起動して１でコピーした sim ディレクトリに移動する。

3．do コマンドでマクロファイル(tb_str_pack1.do または tb_str_pack2.do)を起動する。

2.1 使用例(１) tb_str_pack1.do

Str_Pack を使った基本的な以下の４つのテストを順番に行います。

表 2-1 使用例（１）テスト項目

項目	入力ファイル	出力ファイル	動作
1	なし	ex1_out.txt	本パッケージに含まれる procedure, function の基本的な動作を確認します。 ex1_out.txt にソース記述と出力結果を出力します。
2	ex2_in.txt	ex2_out.txt	ex2_in.txt から空白で区切られた2桁の16進文字列を切り出し、ex2_out.txt に出力します。
3	ex3_in.bin	ex3_out.bin ex3_out.txt	ex3_in.bin からバイナリーデータを読み込み、ex3_out.bin にバイナリーコピー、 ex3_out.txt に1バイト毎に16進文字列に変換したテキストを出力します。
4	ex4_in.txt	ex4_out.txt	ex4_in.txt から TAB または SPACE で区切られた文字列を抽出し、ex4_out.txt に出力します。

2.2 使用例(２) tb_str_pack2.do

Str_Pack を使った映像系の以下の３つのテストを並行して行います。

表 2-2 使用例（２）テスト項目

項目	入力ファイル	出力ファイル	動作
1	NTSC_in_00.bmp	なし	NTSC_in_00.bmp(720×487 ピクセル)の映像データを読み込み、 ITU-R BT.656 のタイミングで R/G/B(4:4:4)データを出力します。
2	なし	なし	1から出力された R/G/B(4:4:4)データを Y/Cb/Cr(4:2:2)に変換します。 ※この項目は項目１の出力を項目３の入力に変換するためのもので、 Str_Pack は使用していません。
3	なし	NTSC_out_01.bmp	2から出力された Y/Cb/Cr(4:2:2)データを R/G/B(4:4:4)に変換し、 NTSC_out_01.bmp(720×487 ピクセル)に出力します。

3. Procedure & Function 一覧

表 3-1 Procedure & Function 一覧表

名前	返却値	機能	パラメータ名		方向	型
strini	なし	str_io を nul 文字で埋める。	variable	str_io	inout	string
strcpy	なし	str_in を str_out にコピーする。	variable	str_out	out	string
			constant	str_in	in	string
strcat	なし	str_in1 と str_in2 を結合して str_in1 にセットする。	variable	str_in1	inout	string
			constant	str_in2	in	string
strcat2	なし	str_in1 と str_in2 を結合して str_out にセットする。	variable	str_out	out	string
			constant	str_in1	in	string
			constant	str_in2	in	string
strdup(1)	string	文字列(str_in)の先頭の文字を num 文字繰り返した文字列を返す。	constant	num	in	integer
			constant	str_in	in	string
strdup(2)	string	文字(chr_in)を num 文字繰り返した文字列を返す。	constant	num	in	integer
			constant	chr_in	in	character
strlen	integer	str_in の文字数を計算する。	constant	str_in	in	string
strcmp	integer	str_in1 と str_in2 の大小比較をする。	constant	str_in1	in	string
			constant	str_in2	in	string
leftstr	string	str_in の左から num 文字を切り出す。	constant	str_in	in	string
			constant	num	in	integer
rightstr	string	str_in の右から num 文字を切り出す。	constant	str_in	in	string
			constant	num	in	integer
midstr	string	str_in の左から num1 番目の文字から num2 文字を切り出す。	constant	str_in	in	string
			constant	num1	in	integer
			constant	num2	in	integer
instr	integer	str_in1 の num 文字目から str_in2 を探す。	constant	str_in1	in	string
			constant	str_in2	in	string
			constant	num	in	integer
hstr2std	std_logic_vector	16 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。	constant	str_in	in	string
			constant	num	in	integer
dstr2std	std_logic_vector	符号つき 10 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。	constant	str_in	in	string
			constant	num	in	integer
bstr2std	std_logic_vector	2 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。	constant	str_in	in	string
			constant	num	in	integer
hstr2int	integer	16 進文字列 str_in を integer に変換する。	constant	str_in	in	string
dstr2int	integer	符号つき 10 進文字列 str_in を integer に変換する。	constant	str_in	in	string
bstr2int	integer	2 進文字列 str_in を integer に変換する。	constant	str_in	in	string
std2hstr	string	std_logic_vector(std_in)を 16 進文字列(1 to std_in'length + 3) / 4)に変換する。	constant	std_in	in	std_logic_vector

名前	返却値	機能	パラメータ名		方向	型
std2dstr	string	std_logic_vector(std_in)を符号つき 10 進文字列(1 to 11)に変換する。	constant	std_in	in	std_logic_vector
std2bstr	string	std_logic_vector(std_in)を 2 進文字列(1 to std_in'length)に変換する。	constant	std_in	in	std_logic_vector
int2hstr	string	integer(int_in)を 16 進文字列(1 to 8)に変換する。	constant	int_in	in	integer
int2dstr	string	integer(int_in)を 10 進文字列(1 to 11)に変換する。	constant	int_in	in	integer
int2bstr	string	integer(int_in)を 2 進文字列(1 to 32)に変換する。	constant	int_in	in	integer
bool2str(1)	string	boolean(bool_in)を文字列("1"/"0")に変換	constant	bool_in	in	boolean
bool2str(2)	string	boolean(bool_in)を変換モード(mode_in)に従って文字列に変換	constant	bool_in	in	boolean
			constant	mode_in	in	integer
ascii(1)	integer	文字列(str_in)の先頭の文字を ASCII コードに変換する。	constant	str_in	in	string
ascii(2)	integer	文字(chr_in)を ASCII コードに変換する。	constant	chr_in	in	character
char(1)	string	ASCII コード(int_in)を1文字の文字列に変換する。	constant	int_in	in	integer
char(2)	character	ASCII コード(int_in)を文字に変換する。	constant	int_in	in	integer
iskanji(1)	boolean	文字列(str_in)の先頭文字が S-JIS 漢字コードの1バイト目かどうかの判定をする。	constant	str_in	in	string
iskanji(2)	boolean	文字(chr_in)が S-JIS 漢字コードの1バイト目かどうかの判定をする。	constant	chr_in	in	character
strpack_ver	integer	本パッケージのバージョンを返す。	なし			
readstr(1)	なし	ファイルから1行リードして str_out にセットする	FILE	f_id		text
			variable	str_out	out	string
			variable	ret_cd	out	boolean
readstr(2)	なし	ファイルから1行リードして str_out にセットする	FILE	f_id		text
			variable	str_out	out	string
writestr(1)	なし	文字列(str_in)をファイルにライトする	FILE	f_id		text
			constant	str_in	in	string
writestr(2)	なし	1文字(chr_in)をファイルにライトする	FILE	f_id		text
			constant	chr_in	in	string
writestr(3)	なし	文字列(str_in)を2つのファイルにライトする	FILE	f_id1		text
			FILE	f_id2		text
			constant	str_in	in	string
writestr(4)	なし	1文字(chr_in)を2つのファイルにライトする	FILE	f_id1		text
			FILE	f_id2		text
			constant	chr_in	in	string
writestr0(1)	なし	文字列(str_in)をファイルにライトする(改行しない)	FILE	f_id		text
			constant	str_in	in	string
writestr0(2)	なし	1文字(chr_in)をファイルにライトする(改行しない)	FILE	f_id		text
			constant	chr_in	in	string
writestr0(3)	なし	文字列(str_in)を2つのファイルにライトする(改行しない)	FILE	f_id1		text
			FILE	f_id2		text
			constant	str_in	in	string
writestr0(4)	なし	1文字(chr_in)を2つのファイルにライトする(改行しない)	FILE	f_id1		text
			FILE	f_id2		text

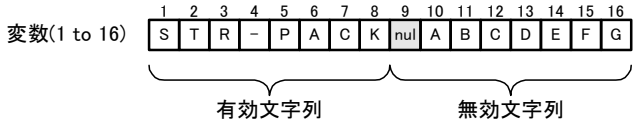
名前	返却値	機能	パラメータ名		方向	型
			constant	chr_in	in	string
readchr	なし	ファイルから1文字バイナリーリードして chr_out にセットする	FILE	f_id		FCHR
			variable	chr_out	out	character
writechr	なし	1文字(chr_in)をファイルにバイナリーライトする	FILE	f_id		FCHR
			constant	chr_in	in	character
get_cmd_param	なし	コマンドファイルから1行リードして、param(0～10)にセットする。	FILE	f_id		text
			variable	eof_flg	out	boolean
			variable	param_num	out	integer
			variable	param	out	PARAM_TYPE

4. Procedure & Function 詳細

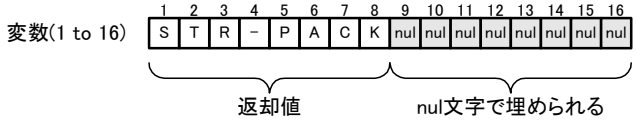
4.1 文字列の扱い

本パッケージでは string 型変数(または定数)の左端から最初に nul 文字(=00h)がセットされている手前までを有効な文字列として扱い、それ以降の文字は無効とする。

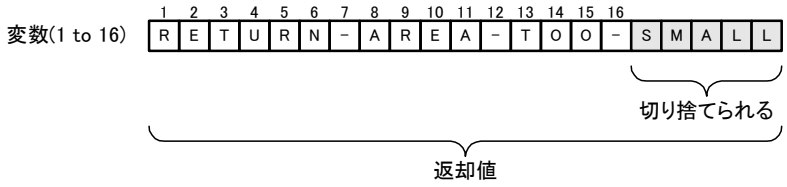
本パッケージから返される文字数が文字領域サイズより少ない場合、余白は nul 文字で埋められる。逆に返される文字数の方が多い場合、はみ出した部分の文字は切り捨てられる。



(a) 対象となる範囲



(b) 領域サイズ > 返却される文字数



(c) 領域サイズ < 返却される文字数

図 4-1 文字列の扱い

4.2 変数の型

各 Procedure, Function の【使用例】で用いられる変数の型は次の通り。

表 4-1 【使用例】で用いられる変数の型

変数名	型
str1, str2, str3	string(1 to 16)
str4, str5	string(1 to 32)
chr1, chr2	character
std1, std2	std_logic_vector(15 downto 0)
int1, int2	integer
bool0, bool1	boolean
str_buf	string(1 to 256)
Fid0, Fid1	text(TEXTIO で用いるファイル変数)
Fid2, Fid3	FCHR(バイナリーI/O で用いるファイル変数)
param	PARAM_TYPE (str_pack 内で array (0 to 10) of string(1 to 256)と定義されている。)

4.3 strini

strini(variable str_io : inout string)

【機能】

str_io を nul 文字で埋める。

【返却値】

なし。

【使用例】

```
strini(str1);
```

str1 に""がセットされる。

4.4 strcpy

strcpy(variable str_out : out string; constant str_in : in string)

【機能】

str_in を str_out にコピーする。

【返却値】

なし。

【使用例】

```
strcpy(str1,"12345");
```

str1 に"12345"がセットされる。

4.5 strcat

strcat(variable str_in1 : inout string; constant str_in2 : in string)

【機能】

str_in1 と str_in2 の文字数合計がを結合して str_in1 にセットする。

【返却値】

なし。

【使用例】

```
strcpy(str1,"12345");
strcpy(str2,"56789");
strcat(str1, str2);
```

str1 に"1234556789"がセットされる。

4.6 strcat2

strcat2(variable str_out : out string; constant str_in1 : in string; constant str_in2 : in string)

【機能】

str_in1 と str_in2 を結合して str_out にセットする。

【返却値】

なし。

【使用例】

```
strcpy(str1,"12345");
strcpy(str2,"56789");
strcat2(str3, str1, str2);
```

str3 に"1234556789"がセットされる。

4.7 strdup

strdup(constant num : in integer; constant str_in : in string)

strdup(constant num : in integer; constant chr_in : in character)

【機能】

文字列(str_in)の先頭の文字、または文字(chr_in)を num 文字繰り返した文字列を返す。(num>0)

【返却値】

繰り返した文字列(string)

【使用例】

```
str1 := strdup(16, "A");
```

str1 に"AAAAAAAAAAAAAAAA"がセットされる。

4.8 strlen

strlen(constant str_in : in string)

【機能】

str_in の文字数を計算する。

【返却値】

文字数(integer)。

【使用例】

```
int1 := strlen("123");
```

int1 に 3 がセットされる。

4.9 strcmp

strcmp(constant str_in1 : in string; constant str_in2 : in string)

【機能】

str_in1 と str_in2 の大小比較をする。

左端から比較して行き、最初に一致しない文字の ascii コード大小関係で判定する。

str_in1 > str_in2 の場合：+1

str_in1 < str_in2 の場合：-1

str_in1 = str_in2 の場合： 0

【返却値】

比較結果(integer)。

【使用例】

```
int1 := strcmp("a123", "a5678");
```

int1 に-1 がセットされる。

4.10 leftstr

leftstr(constant str_in : in string; constant num : in integer)

【機能】

str_in の左から num 文字を切り出す。(num>０)

【返却値】

切り出された文字列(string)

【使用例】

```
str1 := leftstr("123456789abcdefghijklmn", 16);
```

str1 に"123456789abcdefg"がセットされる。

4.11 rightstr

rightstr(constant str_in : in string; constant num : in integer)

【機能】

str_in の右から num 文字を切り出す。(num>０)

【返却値】

切り出された文字列(string)

【使用例】

```
str1 := rightstr("123456789abcdefghijklmn", 16);
```

str1 に"89abcdefghijklmn"がセットされる。

4.12 midstr

leftstr(constant str_in : in string; constant num2 : in integer; constant num2 : in integer)

【機能】

str_in の左から num1 番目の文字から num2 文字を切り出す。(num1, num2>０)

【返却値】

切り出された文字列(string)

【使用例】

```
str1 := midstr("123456789abcdefghijklmn", 4, 16);
```

str1 に"456789abcdefghij"がセットされる。

4.13 instr

instr(constant str_in1 : in string; constant str_in2 : in string; constant num : in integer)

【機能】

str_in1 の num 文字目から str_in2 を探す。num が省略された場合は１文字目から探す。

【返却値】

見つかった場合： 文字の位置(integer)、見つからなかった場合： ０(integer)。

【使用例】

```
strcpy(str1,"123456123456789");
strcpy(str2,"456");
int1 := instr(str1, str2);
int2 := instr(str1, str2, 5);
```

int1 に 4、int2 に 10 がセットされる。

4.14 hstr2std

hstr2std(constant str_in : in string; constant num : integer)

【機能】

16 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。

【返却値】

変換値(std_logic_vector)。

ビット幅は num ビット。変換値ビット数>num の場合、下位 num ビットが返却される。変換値ビット数<num の場合、上位に'0'が埋められる。

【使用例】

```
std1 := hstr2std("1234", 16);
```

std1 に x"1234"("0001:0010:0011:0100")がセットされる。

4.15 dstr2std

dstr2std(constant str_in : in string; constant num : integer)

【機能】

符号つき 10 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。

変換用ワークに integer 型を使用しているので、その範囲を超える数の変換は不可。

【返却値】

変換値(std_logic_vector)。

ビット幅は num ビット。変換値ビット数>num の場合、下位 num ビットが返却される。変換値ビット数<num の場合、上位に'0'が埋められる。

【使用例】

```
std1 := dstr2std("1234", 16);
std2 := dstr2std("-1234", 16);
```

std1 に x"04D2"("0000:0100:1101:0010")、

std2 に x"FB2E"("1111:1011:0010:1110")がセットされる。

4.16 bstr2std

dstr2std(constant str_in : in string; constant num : integer)

【機能】

2 進文字列 str_in を std_logic_vector(num-1 downto 0)に変換する。

【返却値】

変換値(std_logic_vector)。

ビット幅は num ビット。変換値ビット数>num の場合、下位 num ビットが返却される。変換値ビット数<num の場合、上位に'0'が埋められる。

【使用例】

```
std1 := dstr2std("0001001000110100", 16);
```

std1 に x"1234"("0001:0010:0011:0100")がセットされる。

4.17 hstr2int

hstr2int(constant str_in : in string)

【機能】

16 進文字列 str_in を integer に変換する。integer 型の範囲を超える数の変換は不可。

【返却値】

変換値(integer)。

【使用例】

```
int1 := hstr2int("04D2");
```

int1 に 1234 がセットされる。

4.18 dstr2int

dstr2int(constant str_in : in string)

【機能】

符号つき 10 進文字列 str_in を integer に変換する。integer 型の範囲を超える数の変換は不可。

【返却値】

変換値(integer)。

【使用例】

```
int1 := dstr2int("1234");
int2 := dstr2int("-1234");
```

int1 に 1234、int2 に-1234 がセットされる。

4.19 bstr2int

bstr2int(constant str_in : in string)

【機能】

2 進文字列 str_in を integer に変換する。integer 型の範囲を超える数の変換は不可。

【返却値】

変換値(integer)。

【使用例】

```
int1 := dstr2int("0000010011010010");
```

int1 に 1234 がセットされる。

4.20 std2hstr

std2hstr(constant std_in : in std_logic_vector)

【機能】

std_logic_vector(std_in)を16進文字列(1 to std_in'length + 3) / 4)に変換する。

【返却値】

変換値(string)。

【使用例】

```
str1(1 to 4) := std2hstr(x"1234");
```

str1(1 to 4)に"1234"がセットされる。

4.21 std2dstr

std2dstr(constant std_in : in std_logic_vector)

【機能】

std_logic_vector(std_in)を符号つき10進文字列(1 to 11)に変換する。

変換用ワークにinteger型を使用しているので、その範囲を超える数の変換は不可。

【返却値】

変換値(string)。結果は0サプレス左詰めされる。負数の場合は先頭に"-"記号が付加される。

【使用例】

```
str1 := std2dstr(x"1234");
str2 := std2dstr(x"9234");
str3 := std2dstr(x"09234");
```

str1に"4660"、str2に"-28108"、str3に"37428"がセットされる。

4.22 std2bstr

std2bstr(constant std_in : in std_logic_vector)

【機能】

std_logic_vector(std_in)を2進文字列(1 to std_in'length)に変換する。

【返却値】

変換値(string)。

【使用例】

```
str1 := std2bstr(x"1234");
```

str1に"0001001000110100"がセットされる。

4.23 int2hstr

int2hstr(constant int_in : in integer)

【機能】

integer(int_in)を16進文字列(1 to 8)に変換する。

【返却値】

変換値(string)。

【使用例】

```
str1(1 to 8) := int2hstr(1234);
str2(1 to 8) := int2hstr(-1234);
```

str1に"000004D2"、str2に"FFFFFB2E"がセットされる。

4.24 int2dstr

int2dstr(constant int_in : in integer)

【機能】

integer(int_in)を10進文字列(1 to 11)に変換する。

変換用ワークにinteger型を使用しているので、その範囲を超える数の変換は不可。

【返却値】

変換値(string)。結果は0サプレス左詰めされる。負数の場合は先頭に"-"記号が付加される。

【使用例】

```
str1(1 to 11) := int2dstr(1234);
str2(1 to 11) := int2dstr(-1234);
```

str1に"1234"、str2に"-1234"がセットされる。

4.25 int2bstr

int2bstr(constant int_in : in integer)

【機能】

integer(int_in)を2進文字列(1 to 32)に変換する。

【返却値】

変換値(string)。

【使用例】

```
str4 := std2bstr(1234);
str5 := std2bstr(-1234);
```

str4に"00000000000000000000010011010010"、

str5に"11111111111111111111101100101110"がセットされる。

4.26 bool2str(1)

bool2str(constant bool_in : in integer)

【機能】

boolean(bool_in)を文字列("1"/"0")に変換する。

【返却値】

変換値(string)。

【使用例】

```
bool0 := FALSE;
bool1 := TRUE;
strcpy(str1, bool2str(bool0));
strcpy(str2, bool2str(bool1));
```

str1に"0"、str2に"1"がセットされる。

4.27 bool2str(2)

bool2str(constant bool_in : in integer; constant mode_in : in integer)

【機能】

boolean(bool_in)を変換モード(mode_in)に従って文字列に変換する。(mode_inは0～2)

【返却値】

変換値(string)。

【使用例】

```
bool0 := FALSE;
bool1 := TRUE;
strcpy(str1, bool2str(bool0), 0);
strcpy(str2, bool2str(bool1), 0);
```

str1に"0"、str2に"1"がセットされる。

```
bool0 := FALSE;
bool1 := TRUE;
strcpy(str1, bool2str(bool0), 1);
strcpy(str2, bool2str(bool1), 1);
```

str1に"F"、str2に"T"がセットされる。

```
bool0 := FALSE;
bool1 := TRUE;
strcpy(str1, bool2str(bool0), 2);
strcpy(str2, bool2str(bool1), 2);
```

str1に"FALSE"、str2に"TRUE "がセットされる。

4.28 ascii

```
ascii(constant str_in : in string)
ascii(constant chr_in : in character)
```

【機能】

文字列(str_in)の先頭の文字、または文字(chr_in)を ASCII コードに変換する。

【返却値】

ASCII コード(integer)。

【使用例】

```
int1 := ascii("1234");
int2 := ascii('a');
```

int1 に 49、int2 に 97 がセットされる。

4.29 iskanji

```
iskanji(constant str_in : in string)
iskanji(constant chr_in : in character)
```

【機能】

文字列(str_in)の先頭の文字、または文字(chr_in)が S-JIS 漢字コードの１バイト目かどうかの判定をする。

判定は単に対象となる文字の ASCII コードの値のみで行い、x"81"～x"AF"または x"E0"～x"FC"の範囲内の場合は漢字、それ以外の場合は非漢字とする。

【返却値】

判定結果(boolean)。

漢字の場合 TRUE, 非漢字の場合 FALSE。

【使用例】

```
chr1 := char(128);
chr2 := char(129);
strcpy(str1, chr1 & chr2);
bool0 := iskanji(str1);
bool1 := iskanji(chr2);
```

bool0 に FALSE、bool1 に TRUE がセットされる。

4.30 char

```
char(constant int_in : in integer)
```

【機能】

ASCII コード(int_in)を１文字の文字列または文字に変換する。

【返却値】

文字列(string(1 to 1))または文字(character)。

【使用例】

```
str1(1) := char(49);
chr1 := char(97);
```

str1(1)に"1"、chr1 に'a'がセットされる。

4.31 strpack_ver

strpack_ver

【機能】

本パッケージのバージョンを返す。

【返却値】

メジャーバージョン(2 桁) : マイナーバージョン(2 桁)(integer)。

【使用例】

```
int1 := strpack_ver;
```

バージョン 01.04 の場合、int1 に 104 がセットされる。

4.32 readstr

```
readstr(FILE f_id : text; variable str_out : out string; variable ret_cd : out boolean)
readstr(FILE f_id : text; variable str_out : out string)
```

【機能】

f_id でリードオープンされているファイルから１行リードして内容を str_out にセットする。

str_out の領域が足りない場合 ret_cd が TRUE となる。

【返却値】

なし。

【使用例】

```
file_open(Fid0, "infile.txt", READ_MODE);
readstr(Fid0, str_buf);
```

infile.txt ファイルから１行リードされ、内容が str_buf にセットされる。

4.33 writestr

```
writestr(FILE f_id : text; constant str_in : in string)
writestr(FILE f_id : text; constant chr_in : in character)
writestr(FILE f_id1 : text; FILE f_id2 : text; constant str_in : in string)
writestr(FILE f_id1 : text; FILE f_id2 : text; constant chr_in : in character)
```

【機能】

f_id, f_id1, f_id2 でライトオープンまたはアペンドオープンされているファイルに str_in または chr_in の内容をライトする。最後に改行コード(CR+LF)が付加される。

【返却値】

なし。

【使用例】

```
file_open(Fid1, "outfile.txt", WRITE_MODE);
writestr(Fid1, str_buf);
```

str_buf の内容が outfile.txt ファイルにライトされる。

4.34 writestr0

```
writestr0(FILE f_id : text; constant str_in : in string)
writestr0(FILE f_id : text; constant chr_in : in character)
writestr0(FILE f_id1 : text; FILE f_id2 : text; constant str_in : in string)
writestr0(FILE f_id1 : text; FILE f_id2 : text; constant chr_in : in character)
```

【機能】

f_id, f_id1, f_id2 でライトオープンまたはアペンドオープンされているファイルに str_in または chr_in の内容をライトする。改行コード(CR+LF)は付加されない。

【返却値】

なし。

【使用例】

```
file_open(Fid1, "outfile.txt", WRITE_MODE);
writestr0(Fid1, str_buf);
```

str_buf の内容が outfile.txt ファイルにライトされる。

4.35 readchr

readchr(FILE f_id : FCHR; variable chr_out : out character)

【機能】

f_id でリードオープンされているファイルから１文字バイナリーリードして内容を chr_out にセットする。

【返却値】

なし。

【使用例】

```
file_open(Fid2, "infile.bin", READ_MODE);
readchr(Fid2, chr1);
```

infile.bin ファイルから１文字リードされ、内容が chr1 にセットされる。

4.36 writechr

writechr(FILE f_id : FCHR; constant chr_in : in character)

【機能】

f_id でライトオープンまたはアペンドオープンされているファイルに chr_in の内容をバイナリーライトする。

【返却値】

なし。

【使用例】

```
file_open(Fid3, "outfile.bin", WRITE_MODE);
writestr(Fid3, chr2);
```

chr2 の内容が outfile.bin ファイルにバイナリーライトされる。

4.37 get_cmd_param

get_cmd_param(FILE f_id : text; variable eof_flg : out boolean; variable param_num : out integer;
variable param : out PARAM_TYPE)

【機能】

f_id でリードオープンされているファイルから１行リードして空白または TAB で区切られた単語を切り出す。

切り出された個数を param_num に、単語を param(0)～param(10)にセットする。

ただし１行内に１つも単語がない場合は次行をリードして単語の切り出しを続ける。

見つかった単語の個数が n 個(n=1～11)の場合、param_num に n-1、param(0)～param(n)に切り出された単語がセットされる。

１行内の単語が１１個を越える場合は param_num は 10 となり、最初の 11 個の単語が param(0)～param(10)にセットされ、以降の単語は廃棄される。

単語が見つかる前に EOF を検出した場合は eof_flg が TRUE となり、param_num は 0、param(0)～param(10)は全て nul 文字がセットされる。

【返却値】

なし。

【使用例】

```
file_open(Fid0, "infile.txt", READ_MODE);
get_cmd_param(Fid0, eof_flg, param_num, param);
```

infile.txt ファイルから単語を切り出し param_num に単語個数－１、param(0)～param(10)に単語がセットされる。

単語が見つかる前に EOF を検出した場合は eof_flg が TRUE となる。