Windows 用ターミナル ソフト

あじゃた～む

マクロ 内部関数 リファレンス

(1.6.2.2 版)

※ あじゃた～む・マクロ 内部関数のの記述形式は、以下の通り。

1）[*XXXXX*]で囲まれた引数は省略可能。但し、後続のパラメタを指定する場合は省略できない。
ex. _Func(a [,b] [,c]) ------ 「a」は必ず指定しなければならない。
「b」「c」は省略可能だが、「c」を指定する場合は「b」も指定しなければならない。

2）「・・・」は、直前のパラメタを繰り返して指定可能
ex. _Func(a [,b]・・・) ----- 「b」は（何回でも）繰り返して指定可能

3）{XXXXX | YYYYY | ZZZZZ}は、いずれかの項目(XXXXX, YYYYY or ZZZZZ)を指定することを意味します。

1. オプション

#	関数定義 内容	戻り値／※備考	サンプル
1	**option {signed | unsigned | double}** 数値型変数のタイプを指定します。 この命令は、マクロテキストの先頭に記述しなければなりません。 signed　- マクロテキスト中の数値型変数を「符号付き整数」として扱います unsigned - マクロテキスト中の数値型変数を「符号なし整数」として扱います double　- マクロテキスト中の数値型変数を「実数」として扱います	－ 整数は64Bit 実数は倍精度（64Bit）	M010
2	**_EnableStdRClk(fEnable)** 右クリックによる標準のポップアップメニューを許可／禁止します。 許可した場合は、右クリックにより標準のポップアップメニューが表示されます。 禁止した場合は、右クリックイベントが有効となり、右クリックにより_OnRClick()が実行されます。 fEnable - _TRUE ：標準のポップアップメニュー許可 （デフォルト） _FALSE：標準のポップアップメニュー禁止（_OnRClick()実行可） ※Shist/Ctrl キーを押さないで右クリックしたときの挙動を設定します。 Shift/Ctrl キーを押しながら右クリックした場合は、本設定とは関係なく常に右クリックイベントが発生します。	－ ※標準のポップアップメニュー  	M010 M070
3	**_EnableAutoScroll(fEnable)** オートスクロール（描画時に画面最下行へスクロールする）の許可／禁止 fEnable - _TRUE： オートスクロール許可 （デフォルト） _FALSE：オートスクロール禁止	－	M020

2. スクリーン情報

#	関数定義 内容	戻り値／※備考	サンプル
1	**_SetVramSize(width, height)** ＶＲＡＭのサイズを設定します。表示内容はクリアーされます。 width ： VRAM の幅 （半角文字数，デフォルト＝512 文字） height： VRAM の高さ（行数，デフォルト＝128 行） ※本コントロールを使用するアプリケーションで設定する VRAM サイズに依存します。（デフォルトでは最大、512 桁，128 行）	－ ※_locate()により VRAM 内の任意の位置へカーソルを移動できます。	M020 M030
2	**_SetBufLines(lines)** スクロールバッファの行数を設定します。表示内容はクリアーされます。 lines ： スクロール可能な行数（デフォルトは 20,000 行）	－ ※バッファリングし、何行までスクロール可能とするかを設定します。	M030
3	**_GetVRamWidth()** VRAM の幅（半角文字数）を取得します	VRAM の幅（半角文字数） ※デフォルトは 512 桁	M030
4	**_GetVRamHeight()** VRAM の高さ（行数）を取得します	VRAM の高さ（行数） ※デフォルトは 128 行	M030
5	**_GetBufLines()** スクロールバッファの行数（VRAM の行数を含む）を取得します	スクロールバッファの行数 ※デフォルトは 20,000 行	M030

3. 表示出力

#	関数定義 内容	戻り値／※備考	サンプル
1	**_PrintF($fmt [,arg]・・・)** 書式化した文字列を表示します。 $fmt - 書式文字列（C言語の printf()とほぼ同じ） arg - 書式(%・・)に対応した引数	－	M010 M020 M050
2	**_Print({item1 | $item1} [,{item2 | $item12] ・・・)** カーソル位置へ多項目表示出力 item / $item - 数値／文字列 （引数で指定された項目を、数値は 10 進で、文字列はそのまま出力）	－	M020
3	**_Pd({item | $item} [,len] [,lup])** カーソル位置へ１項目表示出力（引数で指定された項目を、数値は10進で、文字列はそのまま出力） item / $item - 数値／文字列 len - 全表示桁数（負数の場合は左詰め，省略時は有効文字長と同じ） lup - 小 数点以下の表示桁数（実数モード時のみ有効）	－ ※実数モード時、「lup」を省略した場合、小数部は常に６桁で表示	M061 M070 M101
4	**_Ph({item | $item} [,len])** カーソル位置へ１項目表示出力 item / $item - 数値／文字列 len - 全表示桁数（負数の場合は左詰め） （引数で指定された項目を、数値は 16 進で、文字列はそのまま出力）	－	M051 M052 M070
5	**_Locate(x, y)** 描画位置設定 x: 桁位置（０～VRAM の幅-1） y: 行位置（０～VRAM の行数-1）	－ ※位置を設定して描画する場合通常_EnableAutoScroll(_FALSE)によりオートスクロールを禁止します。	M020
6	**_Cls( )** 画面クリアーと描画色リセット（描画色＝黒，背景色＝白）		M020
7	**_SetTextColor(color)** テキストの描画色を設定します。 color: テキスト描画色（0:黒，１:赤，２:緑，３:黄，４:青，５:紫，６:水色，７:白）		M020
8	**_SetBackColor(color)** テキストの背景色を設定します。 color: テキスト背景色（0:黒，１:赤，２:緑，３:黄，４:青，５:紫，６:水色，７:白）		M020
9	**_ResetColor()** テキストの描画色＝黒，背景色＝白を設定します。		M020

4. 配列操作

#	関数定義 内容	戻り値／※備考	サンプル
1	**_Elements(arr[])** 配列変数の要素数を得る	配列の要素数	M040 M063
2	**_MakeArray(n)** ｎで指定された要素数の数値型配列を返します。(ex. arr[] = _MakeArray(10);) ※ ex.　n[] = _MakeArray(10)　--- n[]を要素数１０の配列に変更する	※戻り値を受ける変数を要素数ｎの数値型配列に変更する	M040
3	**$_MakeArray(n)** ｎで指定された要素数の文字型配列を返します。(ex. $arr[] = $_MakeArray(10);) ※ ex.　$s[] = $_MakeArray(10); -- $s[] を要素数＝１０の配列に変更する	※戻り値を受ける変数を要素数ｎの文字型配列に変更する	M040

5. 整数演算

#	関数定義 内容	戻り値／※備考	サンプル
1	**_CalcCrc(knd, bno, loc, len [, ini])** １６ビットＣＲＣの算出 knd - 演算種別　_CRC_ITL : CCITT X.25 左送り _CRC_ITR : CCITT X.25 右送り _CRC_16L : ANSI-CRC 左送り _CRC_16R : ANSI_CRC 右送り bno - メモリブロック番号 loc - メモリブロックの先頭ロケーション len - バイト数 ini - 初期値(0x0000 or 0xFFFF)	ＣＲＣ値 ※CCITT X.25 の生成多項式 $X^{16}+X^{12}+X^{5}+1$ ※ANSI-CRC の生成多項式 $X^{16}+X^{15}+X^{2}+1$	M021
2	**_SRand(n)** 擬似乱数の乱数系列初期化 n ＝1: 乱数ジェネレータの再初期化 n ≠1: 乱数系列の初期化	－	M020
3	**_Rand()** 擬似乱数値取得	擬似乱数値（0～32767）	M020

6. 算術演算

#	関数定義 内容	戻り値／※備考	サンプル
1	**_Sqrt(n)** ｎの平方根取得	ｎの平方根	M100
2	**_Sin(degree)** 正弦値取得（dgree は度単位の角度）	正弦値	M100 M120
3	**_Cos(degree)** 余弦値取得（dgree は度単位の角度）	余弦値	M100 M120
4	**_Tan(degree)** 正接値取得（dgree は度単位の角度）	正接値	M100 M120

つづく

つづき

5	**_ASin(n)** n の逆正弦値取得	度単位の角度(-90～+90)	M100
6	**_ACos(n)** n の逆余弦値取得	度単位の角度(0～180)	M100
7	**_ATan(n)** n の逆正接値取得	度単位の角度(-90～+90)	M100
8	**_Atan2(y, x)** y/x の逆正接値取得	度単位の角度(-180～+180)	M100
9	**_Abs(n)** n の絶対値取得	絶対値	M100 M120
10	**_Min(a, b)** a, b の小さい方の値を返します	小さいほうの値	M120
11	**_Max(a, b)** a, b の大きい方の値を返します	大きいほうの値	M120

7. 時間／ウェイト

#	関数定義 内容	戻り値／※備考	サンプル
1	**_CurTime()** 現在時刻（ローカルタイム）を 1970/1/1 00:00:00 からの通算秒数で取得	通算秒数	M020 M090
2	**_GetTime(year, month, day, hour, minute, second)** 指定日時を 1970/1/1 00:00:00 からの通算秒数で取得	1970/1/1 00:00:00 からの通算秒数	M090
3	**_GetDateAndTime([seconds])** 指定日時(1970/1/1 00:00:00 からの通算秒数)がら日付と時刻を得る seconds を省略した場合は、現在の日時を得る （ex. Today[] = _DateAndTime(); ）	日時を格納した配列 [0]: 西暦年　[3]: 時 [1]: 月　[4]: 分 [2]: 日　[5]: 秒 [6]: ミリ秒	M020
4	**$_Date([seconds])** 指定日時(1970/1/1 00:00:00 からの通算秒数)がら日付の文字列を得る seconsd を省略した場合は、現在の日付を得る	日付の文字列 （“yyyy/mm/dd”）	M020 M090
5	**$_Time([seconds])** 指定日時(1970/1/1 00:00:00 からの通算秒数)がら時刻の文字列を得る seconds を省略した場合は、現在の時刻を得る	時刻の文字列 （“hh:mm:dd”）	M090
6	**$_DateAndTime([seconds])** 指定日時(1970/1/1 00:00:00 からの通算秒数)がら日付と時刻の文字列を得る seconds を省略した場合は、現在の日時を得る	日付と時刻の文字列 （“yyyy/mm/dd hh:mm:ss”）	M020 M090
7	**_msWait(msTime)** ミリ秒単位でウェイト (Sleep(1)のループで時間経過をチェック。解像度はWindows タイマ解像度 (10～16ms 程度)に依存)	－	M020 M120
8	**_usWait(usTime)** マイクロ秒単位でウェイト （ウェイト中はSleep()せずにループし、ＣＰＵを使い続けます）	－	M090

8. タイマ起動／停止

#	関数定義 内容	戻り値／※備考	サンプル
1	**_TimerStart(tid, period)** 一定周期でタイマイベント「_OnTimer()」を発生します。 tid　- タイマ識別ＩＤ(１～31) period - タイマイベント起動周期[ms]	－	M062
2	**_TimerStop([tid])** タイマイベントの発生を停止します。 tid　- タイマ識別ＩＤ(１～31) ※「tid」を省略した場合は、全てのタイマを停止します。	－	M062

9. 文字列

#	関数定義 内容	戻り値／※備考	サンプル
1	**_StrLen($str)** 文字列の長さを取得します。	文字列の長さ	M061
2	**_StrICmp($str1 , $str2 [,nEqual]);** 文字列の比較（英大文字，小文字を区別しないで比較　**※１**） nEqual:文字列が一致した時の戻り値（nEqual 未指定時は０を仮定）	＝ nEqual: 等しい ＞ nEqual：$str1 ＞ $str2 ＜ nEqual: $str1 ＜ $str2	M130
3	**_StrStr($str1 , $str2)** 文字列１($str1)中の部分文字列($str2)検索（大文字／小文字を区別して比較）	≠-1:部分文字列の位置(0～) ＝-1:文字列が見つからない	M061
4	**_StrIStr($str1 , $str2)** 文字列１($str1)中の部分文字列($str2)検索（大文字／小文字を区別しないで比較）	≠-1:部分文字列の位置(0～) ＝-1:文字列が見つからない	M130
5	**_ToValue($str)** 数値表現の文字列を数値に変換する（先頭が”0x”の場合は 16 進整数とします） $str: 数値表現の文字列（ex. “123”, “-3.14”, “0xFF”）	変換した数値	M130
6	**$_StrCat([$str1] [,$str2] ・・・)** 指定した文字列を全て連結した文字列を取得	連結した文字列	M060 M070
7	**$_PathCat($str1, $str2)** ２つの文字列の間に、必ず１つの「¥」が挿入された文字列を作成する	連結した文字列	M130
8	**$_StrPart($str , loc, len)** 文字列中の部分文字列を取得 loc: $str 中の部分文字列のバイト位置（負数も可） len: $str 中の部分文字列の長さ（負数の場合は左方向への長さ ex. $_StrMid($s, 3, -3); 先頭 3 文字)	部分文字列 ※loc,len が文字列の範囲外の場合は、マッピングされる文字列の右端／左端を仮定する。但し、文字列外をマッピングしている場合は空文字列	M070
9	**$_StrRep($str1 , loc, len, $Str2)** 文字列($str1)中の部分文字列を、$str2 で置換 loc: $str1 中の部分文字列のバイト位置（0～$str1 の長さ） len: $str1 中の len:部分文字列の長さ（0 の場合は文字列($Str2)を挿入）	部分文字列を置換した文字列 ※loc が負数の場合は 0 に、$str を超える場合は$str1 の長さに補正します	M072

つづく

10	**$_StrLTrim($str)** **$_StrLTrim($str[])** 文字列の前部空白を除去 $str ： 前部空白を除去した文字列を作成するソース文字列 $str[]: 前部空白を除去する文字列型の配列（配列の全要素ををを書換える）	前部空白を除去した文字列 ※配列（$str[]）を指定した場合、戻り値はありません。	M130
11	**$_StrRTrim($str)** **$_StrRTrim($str[])** 文字列の後部空白を除去 $str ： 後部空白を除去した文字列を作成するソース文字列 $str[]: 後部空白を除去する文字列型の配列（配列の全要素ををを書換える）	後部空白を除去した文字列 ※配列（$str[]）を指定した場合、戻り値はありません。	M130
12	**$_StrTrim($str)** **$_StrTrim($str[])** 文字列の前後の空白を除去 $str ： 前後の空白を除去した文字列を作成するソース文字列 $str[]: 前後の空白を除去する文字列型配列	前後の空白を除去した文字列 ※配列（$str[]）を指定した場合、戻り値はありません。	M063
13	**$_StrSplit($str [,$delimiter])** 文字列($str)をデリミタ($delimiter)で分解した文字列の配列を返す $str ： デリミタで分解した配列を作成するソース文字列 $delimiter: デリミタ文字列（未指定時はカンマ(“,”)）	デリミタで分解した文字列の配列	M063
14	**$_StrJoin($str[] [,$delimiter])** 文字列配列($str[])の各要素をデリミタ($delimiter)で繋ぎ１つの文字列にする $str[] ： 要素をデリミタで繋ぎ結合する文字列配列 $delimiter: デリミタ文字列（未指定時はカンマ(“,”)）	デリミタで繋いだ文字列	M130
15	**$_SPrintF($fmt [,arg]・・・)** 書式文字列作成（数値項目を**16進**文字列に変換） $fmt - 書式文字列（C言語の printf()とほぼ同じ） arg - 書式に対応した引数	書式化した文字列	M130
16	**$_Input([$Prompt] [,$DefStr])** ダイアログによる文字列の入力（最大 511 バイト） $Prompt - タイトル（入力ダイアログのタイトルバーに表示する文字列） $DefStr - 初期文字列（省略時は空文字列）	入力／選択した文字列 ※文字列の入力は、コンボボックスから項目を選択するか、直接文字列を入力します。	M061 M063 M070 M072
17	**_HexStrToBinary($HexStr, MemBlkNo, loc)** 空白で区切られた２桁の 16 進数をバイナリデータに変換しメモリブロックに格納する。 $HexStr - 16 進文字列（Ex. “41 42 0D 0A”） MemBlkNo - メモリブロック番号 loc - バイト位置（０～） ※以下の場合は、バイナリデータへの変換を中止します。 ・16 進の値が 0x00～0xFF 以外 ・16 進文字以外を検出 ・メモリブロックをオーバー	バイナリデータのバイト数	M070
18	**_GetValueByKey($str, $key [,fExact])** 文字列($str)中のキー($key)に後続する数値取得 $str - 文字列 $key - キー fExact - キーの比較方法（_TRUE:大文字と小文字を区別する, _FALSE：区別しない） ※ ex. _GetValueByKey(“-- X=123 --”, “X=”) → 数値(123)を返す	キーに後続する数値 ※キーが無い場合や、後続する数値が無い場合は０を返します。	－
19	**_GetValues($str)** 文字列中の字句から、数値を収集した配列を返します。 $str - 数値を抽出する文字列 ※ ex. v[] = _GetValues(“-- x = 123 (r is 10, q is -20)”) → v[0] = 123, v[1] = 10, v[2] = -20 を返す ※ 字句の単位はC言語と同じ（コメント「/*・・*/」や「//・・」も有効）	収集した数値の配列 ※数値が１つも無い場合は、要素数＝１の配列で、値＝０を返します。	－

※1：英大文字，小文字を区別する場合は、文字列の比較式（ex. $txt == “ABC”）で記述可能

10. 問い合わせ

#	関数定義 内容	戻り値／※備考	サンプル
1	**_MsgBox($Message, $Title, BoxType)** 問い合わせのメッセージボックスを表示します。 $Message - 問い合わせメッセージ文字列 $Title - メッセージボックスのタイトルバーに表示する文字列 BoxType - メッセージボックスのタイプ _MB_OK - [OK] プッシュボタンだけを表示 _MB_OKCANCEL - [OK]、[キャンセル] の各プッシュボタンを表示 _MB_RETRYCANCEL - [再試行]、[キャンセル] の各プッシュボタンを表示 _MB_YESNO - [はい]、[いいえ] の各プッシュボタンを表示 _MB_YESNOCANCEL - [はい]、[いいえ]、[キャンセル] の各プッシュボタンを表示 _MB_ICONERROR - 停止マークアイコンを表示します。	_IDOK - 「OK」を選択 _IDCANCEL - 「CANCEL」を選択 _IDNO - 「NO」を選択 _IDYEAS - 「YES」を選択	M070
2	**_PopupMenu($menu[], x, y [,$Func])** ポップアップメニューを表示し、選択した項目を返します。 $menu[] - メニュー項目（文字型配列） x, y - メニューの表示位置（マクロウインドの左上が原点(0,0)） $Func - メニューが選択された時に実行するイベントファンクション	・$Func 未指定時 ≧0:選択されたメニュー項目 (項目のインデクス値) -1:メニューはキャンセルされた ・$Func 指定時は返り値なし	M010 M070 M071 M072

11. メモリブロック

#	関数定義 内容	戻り値／※備考	サンプル
1	**_MemAlloc(BlockSize)** メモリブロックを割り当てます BlockSize - 割り当てるメモリブロックのバイト数(１～0xFFFFFFFF, 最大 4GB-1)	メモリブロック番号	M050
2	**_MemFree(MemBlkNo)** メモリブロックを解放します。 MemBlkNo - メモリブロック番号	ー	M050 M060 M070
3	**_GetMemBlk(MemBlkNo, loc, len [,fChangeEndian])** メモリブロック内のデータを読み出します。 (指定位置から指定バイト数の整数値(8～64Bit,リトルエンディアン)を読み出します) MemBlkNo - メモリブロック番号 loc - バイト位置（０～） len - バイト数 fChangeEndian - エンディアン形式(BigEndian/LittleEndian)の変更指定フラグ _FALSE : エンディアン形式を変更しない _TRUE : エンディアン形式を変更する	メモリブロック内から読み出したデータ値 （実数モードの場合は、読み出した整数値(8～64Bit)を実数値に変換します。）	M050 M051 M052 M070
4	**_PutMemBlk(MemBlkNo, loc, len, value [,fChangeEndian])** メモリブロック内へデータを書き込みます （データの整数値（下位8～64Bit）を指定バイト位置, バイト数へ書き込みます） MemBlkNo - メモリブロック番号 loc - バイト位置（０～） len - バイト数（１～８） value - メモリブロック内へ書き込むデータ値 実数モードの場合は、実数値を整数値(8～64 Bit)に変換して書き込みます fChangeEndian - エンディアン形式(BigEndian/LittleEndian)の変更指定フラグ _FALSE : エンディアン形式を変更しない _TRUE : エンディアン形式を変更する	ー	M050 M051 M052

つづく

つづき

5	**_ClrMemBlk(MemBlkNo, ByteValue)** メモリブロックをバイト値(Value)でクリアーします。 MemBlkNo - メモリブロック番号 Value - バイト値(0～255)	－	M050 M051 M052
6	**_DumpMemBlk(MemBlkNo [,loc] [,len])** メモリブロックの内容を、１６進ダンプ表示します。 MemBlkNo - メモリブロック番号 loc - バイト位置（省略時は０を仮定） len - バイト数 （省略時はメモリブロック末尾までのバイト数を仮定）	－	M050 M051 M052

12. ファイル／フォルダ

#	関数定義 内容	戻り値／※備考	サンプル
1	**_FOpen($FilePath, $Access)** ファイルをオープンします。 $FilePath - ファイルパス名 $Access - ファイルアクセス指定(“rt”,“wt”, at”, “rb”, “wb” or “ab”) “rt”:テキストファイル読出 “rb”:バイナリファイル読出 “wt”:テキストファイル書込 “wb”:バイナリファイル書込 “at”:テキストファイル追記 “ab”:バイナリファイル追記	ファイル番号（１～３１）	M060 M061 M062
2	**_FClose(fno)** ファイルをクローズします。 fno - ファイル番号(_FOpen の戻り値)	－ ※マクロ終了時、クローズされていないファイルは全て自動的にクローズします。	M060 M061 M062
3	**_FPutS(fno, $str)** ファイルに文字列を書き込みます。 fno - ファイル番号(_FOpen の戻り値) $str - ファイルに書き込む文字列 ※ _FOpen のファイルアクセス指定は”wt” or “at”でなければなりません。 ※ ファイルから文字列の読み出しは、文字型関数($_FGetS)参照	－	M060
4	**_FRead(fno, bno, loc, len)** ファイルからバイナリデータを読み出します。 fno - ファイル番号(_FOpen の戻り値) bno - メモリブロック番号(_MemAlloc の戻り値) loc - 読みだしたデータを格納するバイト位置（メモリブロック内ロケーション） len - 読み出すバイト数 ※ _FOpen のファイルアクセス指定は”rb”でなければなりません。	読み出したバイト数	M060
5	**_FWrite(fno, bno, loc, len)** ファイルへバイナリデータを書き込みます。 fno - ファイル番号(_FOpen の戻り値) bno - メモリブロック番号(_MemAlloc の戻り値) loc - 書き込むデータのバイト位置（メモリブロック内ロケーション） len - 書き込むバイト数 ※ _FOpen のファイルアクセス指定は”wb” or “ab”でなければなりません。	書き込んだバイト数	M060
6	**_FGetSeek(fno)** 現在のファイルポインタ（ファイルのアクセスバイト位置）を取得します。 fno - ファイル番号(_FOpen の戻り値)	現在のファイルポインタ	M060
7	**_FSetSeek(fno, loc)** ファイルポインタ（ファイルのアクセスバイト位置）を設定します。 fno - ファイル番号(_FOpen の戻り値) loc - ファイルポインタ	－	M060

つづく

つづき

8	**$_FGetS(fno)** ファイルから文字列を読み出します。(１行分，最大 1023 バイト) fno － ファイル番号(_FOpen の戻り値) ※ _FOpen のファイルアクセス指定は”rt”でなければなりません。	読み出した文字列	M060 M061 M062
9	**_FDelete($FilePath)** 指定されたファイルをを削除します。 $FilePath - 削除するファイルのパス名(ex. ‘d:¥work¥sub¥temp¥Sample.txt’);	－	M060
10	**_MakeFolder($FolderPath)** 絶対パス名で指定されたフォルダを作成します。(パス名の末尾は「¥」であること) $FolderPath - 作成するフォルダの絶対パス名(ex. ‘d:¥work¥sub¥temp¥’);	－	M140
11	**_PathExists($Path)** パスが存在するかチェックします $Path - 存在をチェックするパス名	_TRUE ： 存在する _FALSE ： 存在しない	M140
12	**_PathIsFolder($Path** 指定したパスが、実在するフォルダかチェックします $Path - 実在するフォルダがをチェックするパス名	_TRUE ：実在するフォルダ _FALSE ： 当該フォルダは存在しない	M140

13. ファイル名／フォルダ名取得

#	関数定義 内容	戻り値／※備考	サンプル
1	**$_GetOpenFile([$Title] [,$Filter[[,$DefExt] [,$DefFile])** ダイアログにより、オープンするファイルを選択します。 $Title － ダイアログタイトル (未指定時は空文字列) $Filter - ファイル選択フィルタ(未指定時は “All(*.*)/*.*)--- ※１ $DefExt - デフォルト拡張子 (未指定時は“txt”) $DefFile - デフォルトのファイル・パス名(未指定時は空文字列)	選択したオープンファイルのパス名 （キャンセルした場合は空文字列(“”))	M070
2	**$_GetSaveFile([$Title] [,$Filter] [,$DefExt] [,$DefFile])** ダイアログにより、セーブするするファイルを選択／指定します。 $Title － ダイアログタイトル (未指定時は空文字列) $Filter - ファイル選択フィルタ(未指定時は “All(*.*)/*.*) --- ※１ $DefExt - デフォルト拡張子 (未指定時は“txt”) $DefFile - デフォルトのファイル・パス名(未指定時は空文字列)	選択/指定したセーブファイルのパス名 （キャンセルした場合は空文字列(“”))	M061 M062 M071 M072
3	**$_GetFolder([$WndTitle] [,$BoxTitle] [,DefFolder])** ダイアログにより、フォルダを選択します。 $WndTitle － ウインド・タイトルバーに表示するタイトル(未指定時は空文字列) $BoxTitle － ダイアログボックス内に表示するタイトル(未指定時は空文字列) $DefFolder - デフォルトのフォルダ・パス名(未指定時は空文字列)	選択したフォルダのパス名 （キャンセルした場合は空文字列(“”))	M063 M072
4	**$_GetMacFilePath()** 現在実行中のマクロファイル・パス名を取得します。	マクロファイルのパス名	M064
5	**$_SearchFiles($_Folder, fSubDir [{, $Wild | $Wild[] }]…)** ファイルを検索し、見つかったファイルパス名の配列を返します。 $_Folder - 検索するフォルダパス名 fSubDir －サブフォルダ検索フラグ (_TRUE:サブフォルダも検索する、_FALSE:検索しない) $Wild － 検索するワイルドカード $Wild[] － 検索するワイルドカード (配列による複数指定) ex. $files[] = _SearchFiles(‘d:¥work’, _TRUE, “*.c”, “*.h”);	見つかったファイルパス名の配列 ※マクロテキスト中に「_OnFileSearch()」関数があれば、これを呼び出して、検索中のフォルダ名を通知します。（→イベント処理関数を参照）	M063

つづく

つづき

6	**$_SplitPath($PathName)** パス名分解（絶対パス名を分解し、要素数＝４の文字型配列を返します） ex. $s[] =$_SplitPath(‘c:¥sub_dir¥sample.txt’);	以下の内容の文字型配列 [0]:ドライブ (ex. ‘c:’) [1]:フォルダパス (ex. ¥sub_dir¥’) [2]:ファイル名 (ex. ‘sample’) [3]:拡張子 (ex. ‘.txt’)	M064
7	**$_MakePath($drive [, $dir] [, $fname] [, $ext])** **$_MakePath($parts[])** パス名作成（ドライブ，フォルダ名，ファイル名，拡張子を連結したパス名の作成） $drive: ドライブ (ex. ‘c:’) $dir : フォルダ名 (ex. ‘¥sub_dir¥’) $fname: ファイル名 (ex. ‘sample’) $ext : 拡張子 $parts[] - ４項目の文字列配列で、[0] = ドライブ，[1] = フォルダ名，[2] = ファイル名，[3] =拡張子	連結した絶対パス名 ex. ‘c:¥sub_dir¥sample.txt’	M064

※１：「タイトル」と「ワイルドカード」のペアをスラッシュ(/)で区切って指定します。
例えば、テキストファイルと全ファイルを指定する場合は、” TextFile(*.txt)/*.txt/All Files(*.*)/*.* ” のように指定します

14. プロファイル・アクセス

#	関数定義 内容	戻り値／※備考	サンプル
1	**_SetProfileSect($sect)** アクセスするプロファイルのセクションを設定します $sect - セクション名 本関数を実行しない場合のデフォルトセクション名は「DefSect」です。	－	M061
2	**_SetProfileVal($key, value)** プロファイル（プロファイルセクション）へ、値を記録します $key - キー名称 value - プロファイルへ記録する値	－	M150
3	**_SetProfileStr($key, $string)** プロファイル（プロファイルセクション）へ、文字列を記録します $key - キー名称 $string - プロファイルへ記録する文字列	－	M061 M063
4	**_GetProfileVal($key, DefVal)** プロファイル（プロファイルセクション）から値を読み出します $key - キー名称 DefVal - デフォルト値	プロファイルから読み出した値 当該キーが存在しない場合は DefVal を返します。	M150
5	**$_GetProfileStr($key, $DefStr)** プロファイル（プロファイルセクション）から文字列を読み出します $key - キー名称 $DefStr - デフォルト文字列	プロファイルから読み出した文字列 当該キーが存在しない場合は $DefStr を返します。	M061 M063

15. ＣＯＭポート／メールスロットアクセス

#	関数定義 内容	戻り値／※備考	サンプル
1	**_ComCreate([{PortNo | $MySlotName}])** ＣＯＭポートの初期化を行います。（まだ、オープンはしません） PortNo　　- ＣＯＭポート番号　（ＣＯＭ１～２５５生成） $MySlotName - 自メールスロット名（メールスロット生成） ※未指定のポート情報はプロファイルから読み出されます。 ※引数の指定がない場合は、前回設定値を採用します。	ポート番号（1～256）	M062 M070 M071 M073 M074 M080
2	**_ComCreateByDialog([OldPort])** ダイアログにより、ポートのパラメタを設定し、ポートの初期化を行います。 OldPort - 旧ポート番号 ※オープン状態で同一ポートを指定しＯＫボタンを押した場合は、当該ポートを再オープンします。 ※新たなポートを生成した場合、「OldPort」の指定がありれば、（新旧のポート番号が異なる場合）この旧ポートを破棄します。（但し、OldPort=0 の場合は何もしない） ※「OldPort」を指定しないで異なるポートを選択した場合、旧ポートは破棄されずにそのまま残ります。	1～256 - 生成したポート番号 0　　- キャンセル／エラー	M070 M071 M072 M080
3	**_ComDelete(PortNo)** _ComCreate/_ComCreateByDialog で初期化したポートを破棄します。 ポートがオープン状態の場合は、クローズしてから破棄します。 PortNo　- ポート番号（1～256） ※ポート情報はプロファイルへ記録されます。	－ ※マクロ実行終了時、破棄されていないポートは自動的に破棄されます。	M070 M071 M072 M080
4	**_ComOpen (PortNo [, Rate] [, DataLen] [, Parity] [, StopBit])** **_ComOpen ([$RemoteSlotName] [,$RemoteHostName])** **_ComOpen(256)** １）ＣＯＭポートをオープンします。 PortNo　- ＣＯＭポート番号（1～255） Rate　　- 通信レート[bps] DataLen　- データ長（７～８） Paritely - パリティビット(0:なし、1:奇数, 2:偶数) StopBit　- ストップビット長（１～２） ２）メールスロットをオープンします $RemoteSlotName - 相手のメールスロット名 $RemoteHostName - 相手のコンピュータ名（空文字(“”)指定時は自コンピュータ） ３）現在値でメールスロットをオープン（port = 256 を指定）	－ ※当該ポートは、_ComCreate/_ComCreateByDialog により初期化されていなければなりません。 ※引数未指定時は、前回値を採用します。	M062 M070 M071 M072 M080
5	**_ComIsOpenPossible (PortNo)** ポートがオープン可能か、チェックします。 PortNo　- ポート番号（1～256）	_TRUE　- オープン可能 _FALSE - オープン不可能	M070 M071 M080
6	**_ComIsOpened(PortNo)** ポートがオープン状態か、チェックします。 PortNo　- ポート番号（1～256）	_TRUE　- オープン状態 _FALSE - クローズ状態	M070
7	**_ComClose(PortNo)** ポートをクローズします。 ポートが既にクローズされている場合は、何もしません。 PortNo　- ＣＯＭポート番号（1～256）	－	M062 M070 M071 M072 M080

つづく

つづき

8	**_ComSendText(PortNo, $Text1 [, $Text2)] ...)** ポートへテキストデータを送信します。 PortNo - ポート番号（1～256） Text1～ - 送信するテキスト群	－	M062 M070
9	**_ComSendBinary(PortNo, MemBlkNo, loc, len)** ポートへバイナリデータを送信します。 PortNo - ポート番号（1～256） MemBlkNo - メモリブロック番号 loc - 送信データのバイト位置（メモリブロック内ロケーション） len - 送信データのバイト数	－	M070
10	**_ComSendBytes(PortNo, b1 [,b2]・・・)** ポートへバイトデータ（複数バイト可）を送信します。 PortNo - ＣＯＭポート番号（1～256） b1 [,b2]・・・ - 送信バイトデータ群	－	M070
11	**_ComSendPacket(PortNo, MemBlkNo, loc, len)** ポートへパケットデータを送信します。 PortNo - ポート番号（1～256） MemBlkNo - メモリブロック番号 loc - 送信データのバイト位置（メモリブロック内ロケーション） len - 送信データのバイト数	－ ※送信データブロックを DLE・STX～DLE・ETX でサンドイッチし、データ中の DLE(0x10) を２バイトの DLE に拡張します	M070
12	**_ComGetRate(PortNo)** ＣＯＭポートの設定値（転送速度[bps]）を取得します PortNo - ＣＯＭポート番号（1～255）	転送速度（１～）	M070
13	**_ComGetDataBits(PortNo)** ＣＯＭポートの設定値（データビット数）を取得します PortNo - ＣＯＭポート番号（1～255）	７～８	M070
14	**_ComGetParitely(PortNo)** ＣＯＭポートの設定値（パリティビット）を取得します PortNo - ＣＯＭポート番号（1～255）	０：なし １：奇数パリティ ２：偶数パリティ	M070
15	**_ComGetStopBits(PortNo)** ＣＯＭポートの設定値（ストップビット数）を取得します PortNo - ＣＯＭポート番号（1～255）	１～２	M070
16	**$_ComGetParam(PortNo)** ポートの設定値を文字列で取得します PortNo - ポート番号（1～256） ※メールスロット(PortNo=256)の場合は、接続先スロットのパス名を返します。	“COMn, <r> , <d>, <p>, <s>” <r>:転送レート[bps] <d>:データビット数（7 or 8） <p>:パリティ(No, Odd or Even) <s>:ストップビット長(1 or 2)	M070 M071 M080
17	**_ComSetDTR(PortNo, fDTR)** ＣＯＭポートのＤＴＲ信号を設定します PortNo - ＣＯＭポート番号（1～255） fDTR - ＤＴＲ信号設定値(_TRUE ： 有効状態, _FALSE ： 無効状態)	－	M080
18	**_ComSetRTS(PortNo, fRTS)** ＣＯＭポートのＲＴＳ信号を設定します PortNo - ＣＯＭポート番号（1～255） fRTS - ＲＴＳ信号設定値(_TRUE ： 有効状態, _FALSE ： 無効状態)	－	M080
19	**_ComGetSignal(PortNo)** ＣＯＭポートの信号状態を取得します。 PortNo - ＣＯＭポート番号（1～255）	Bit4(0x10) - CTS 信号状態 Bit5(0x20) - DSR 信号状態 Bit6(0x40) - RING 信号状態 Bit7(0x80) - RLSD 信号状態 いずれも「1 」で有効状態	M080

※ポート番号=256 はメールスロットを意味します

16. ＣＯＭポート／メールスロットイベント登録

#	関数定義 内容	戻り値／※備考	サンプル
1	**_ComSetOnRxText(PortNo, $func)** ポートからのテキスト受信時のイベント処理関数登録 PortNo － ポート番号（1～256） $func － テキスト受信時に実行する関数名	－	M070
2	**_ComSetOnRxCtrl(PortNo, $func)** ポートからの制御コード受信時のイベント処理関数登録 PortNo － ポート番号（1～256） $func － 制御コード受信時に実行する関数名	－	M070
3	**_ComSetOnRxEsc(PortNo, $func)** ポートからのＥＳＣシーケンス文字列受信時のイベント処理関数登録 PortNo － ポート番号（1～256） $func － ＥＳＣシーケンス受信時に実行する関数名	－	M070
4	**_ComSetOnRxPacket(PortNo, $func, MemBlkNo)** ポートからのパケットデータ受信時のイベント処理関数登録 PortNo － ポート番号（1～256） $func － パケットデータ受信時に実行する関数名 MemBlkNo － パケットデータを格納するメモリブロック番号	－	M070

17. ＣＯＭポート／メールスロット通信イベント処理関数

#	関数定義 内容	戻り値／※備考	サンプル
1	***OnRxText*($Text)** ポートからのテキスト受信時の処理を行います $Text － 受信したテキストデータ	－ ※実際の関数名は、_ComSetOnRxText()で登録した関数名となります。	M070
2	***OnRxCtrl*(Ctrl)** ポートからの制御コード受信時の処理を行います Ctrl － 受信した制御コード	－ ※実際の関数名は、_ComSetOnRxCtrl()で登録した関数名となります。	M070
3	***OnRxEsc*($Esc)** ポートからのＥＳＣシーケンス文字列受信時の処理を行います $Esc － 受信したＥＳＣシーケンス文字列	－ ※実際の関数名は、_ComSetOnRxEsc()で登録した関数名となります。	M070
4	***OnRxPacket*(len)** ポートからのパケットデータ受信時の処理を行います len － 受信したパケットデータのバイト数 ※受信パケットデータは、_ComSetOnRxPacket()で指定したメモリブロックの先頭から格納されます。	－ ※実際の関数名は、_ComSetOnRxPacket()で登録した関数名となります。	M070

※実際の関数名は、ＣＯＭポート／メールスロットイベント登録群「_ComSetOn*XXXXX*()」により指定します。
※上記イベントは、_DisComEvt()/_EnaComEvt()により禁止／許可できます。

18. ＸＭＯＤＥＭ／ＹＭＯＤＥＭ ファイル転送

#	関数定義 内容	戻り値／※備考	サンプル
1	**_ComSendXModemSum(PortNo, $File) - 128Bytes ブロック，チェックサム** **_ComSendXModemCrc(PortNo, $File) - 128Bytes ブロック，ＣＲＣ** **_ComSendXModem1K (PortNo, $File) - 1KBytes ブロック，ＣＲＣ** XMODEM プロトコルによりファイルを送信します。 PortNo - ポート番号（1～256） $File - 送信するファイルのパス名	－ ※送受信中は状況をログ表示 ：- 1024Bytes データ送受信 ．- 128Bytes データ送受信 R - リトライ発生 E - 送信終了 C[SUM/CRC/1K] - プロトコル変更	M071 M073
2	**_ComRecvXModemSum(PortNo, $File) - 128Bytes ブロック，チェックサム** **_ComRecvXModemCrc(PortNo, $File) - 128Bytes ブロック，ＣＲＣ** **_ComRecvXModem1K (PortNo, $File) - 1KBytes ブロック，ＣＲＣ** XMODEM プロトコルによりファイルを受信します。 PortNo - ポート番号（1～256） $File - 受信データを格納するファイルのパス名		M071 M074
3	**_ComSendYModem (PortNo, $Wild [,fSubDir] [,fFolder])** YMODEM プロトコルによりファイルを送信します。 PortNo - ポート番号（1～256） $Wild - 送信するファイルのワイルドパス名 fSubDir - サブディレクトリ検索フラグ(_TRUE:検索する,_FALSE:検索しない) fFolder - 送信ファイル名にパス情報を付加するか否かのフラグ _TRUE : 送信するファイル名にパス情報を付加する _FALSE : 送信するファイル名にパス情報を付加しない	－ ※送受信中は状況をログ表示 ：- 1024Bytes データ送受信 ．- 128Bytes データ送受信 R - リトライ発生 E - 送信終了 ※:ファイル名にフォルダパスを付加する場合は、先頭フォルダからの相対パスとなる （ex. .¥SubDir¥ sample.txt）	M072 M075 M076 M077
4	**_ComRecvYModem(PortNo, $folder, [,fSubDir])** YMODEM プロトコルによりファイルを受信します。 PortNo - ポート番号（1～256） $folder - 受信ファイルを格納する先頭フォルダのパス名 fSubDir - 受信ファイル名にパス情報が付加されている場合の動作 _TRUE : 先頭フォルダからの相対で、サブフォルダを作成する _FALSE : サブフォルダは作成せずに、全て先頭フォルダに格納		M072 M078 M079

19. 標準イベント処理関数

#	関数定義 内容	戻り値／※備考	サンプル
1	**_OnKeyIn(key)** キー入力時に実行されます。 key - 入力したキーコード	－ ※以下のキーは通知しません。 CTRL+A(0x01):全テキスト選択 CTRL+C(0x03):選択テキスト・コピー	M070
2	**_OnRClick(x, y, shift, ctrl)** マクロウインド上で右クリックした場合に実行されます。 x - クリックしたＸ位置 y - クリックしたＹ位置 shift - Shift キーの押下状態（１：押下，０：非押下） ctrl - Ctrl キーの押下状態（１：押下，０：非押下）	－ ※ x, y はマクロウインドの左上を原点(0, 0)としたピクセル位置	M010 M070 M071 M072
3	**_OnTmcClose()** タイムチャートウインドがクローズされたことを通知します。	－ TmcOpen()実行後ウインドが破棄された時にコールされます。	M120
4	**_OnButton(BtnNo)** ボタンが押されたことを通知します。 BtnNo - ボタン識別番号（０～７）	－	M020 M070 M110

つづく

つづき

5	**_OnChkBox(ChkBoxNo, staus)** チェックボックスの状態が設定されたことを通知します。 status - チェック状態（_TRUE:チェックしている, _FALSE:チェックしていない） ※チェック状態を設定時、チェック状態が変化しなくても実行されます。	ー ※この関数内から、_ChkBoxSetSts()は実行できません。	M110
6	**_OnFileSearch($folder)** ファイル検索中のフォルダ・パス名を通知します。 $folder - フォルダ・パス名	_TRUE ： ファイル検索を継続する _FALSE ： ファイル検索を中止する ※$_SearchFiles()の実行中にコールされます。	M063
7	**_OnTimer(tid)** タイマの経過を通知します tid - タイマ識別ＩＤ（１～３１）	ー	M062
8	***_OnMenu*(id)** ポップアップメニューで項目が選択されたことを通知します。 id - メニュー項目インデクス（０～）	ー ※実際の関数名は、_PopupMenu()の$Func 引数で指定します。	M551

※上記関数は、ユーザがマクロテキスト内で、固定の関数名で定義します。
※上記イベントは、_DisStdEvt()/_EnaStdEvt()により禁止／許可できます。

20. 「あじゃたーむ」接続イベント処理関数

#	関数定義 内容	戻り値／※備考	サンプル
1	_OnTermLink($MySlot, $RmtSlot) 「あじゃた～む」との接続が可能であることを通知します。	ー	M200
2	_OnTermUnlink() 「あじゃた～む」が終了したことを通知します。	ー	M200
3	_OnTermOpened() 「あじゃた～む」上のシリアル回線が接続されたことを通知します。	ー	M200
4	_OnTermClosed() 「あじゃた～む」上のシリアル回線が切断されたことを通知します。	ー	M200

※１：「あじゃた～む」と接続するには、_OnTermLink()関数で、以下のように処理します。（サンプルマクロ「M200.txt」参照）

```
function _OnTermLink($MySlot, $RmtSlot)
{
    //  ターミナルとの接続用メールスロット・ポート生成
    _ComCreate($MySlot);
    //  受信イベント関数設定
    _ComSetOnRxText   (_SLOT_PORT, "OnText");
    _ComSetOnRxCtrl   (_SLOT_PORT, "OnCtrl");
    _ComSetOnRxEsc    (_SLOT_PORT, "OnEsc" );
    _ComSetOnRxPacket(_SLOT_PORT, "OnPacket", bno);
    //  ターミナルとの接続用メールスロット・ポートオープン
    _ComOpen($RmtSlot);
      ・・・・・
}
```

「あじゃた～む」と正常に接続した場合、ウインド下部に「AjrTerm*nnn*.ini と接続中」と表示されます。

（AjrTerm*nnn*.ini は、「あじゃた～む」のウインドタイトルに表示されている内容です）

21. イベントの禁止／許可

#	関数定義 内容	戻り値／※備考	サンプル
1	**_DisStdEvt()** 標準イベントの実行を禁止します。（※１）	－ ※この関数は、標準イベント関数(*_OnXXX*)からはコールできません。	－
2	**_EnaStdEvt()** 標準イベントの実行を許可します。（※１）		－
3	**_GetDisStdEvt()** 標準イベント禁止のネストレベルを取得します。	標準イベント禁止のネストレベル ＝０：イベント許可状態 ＞０：イベント禁止状態	－
4	**_DisComEvt()** ＣＯＭポート通信イベントの実行を禁止します。（※１）	－ ※この関数は、標準イベント関数(*_OnXXX*)からはコールできません。	－
5	**_EnaComEvt()** ＣＯＭポート通信イベントの実行を許可します。（※１）		－
6	**_GetDisComEvt()** ＣＯＭポート通信イベント禁止のネストレベルを取得します。	ＣＯＭポート通信イベント禁止のネストレベル ＝０：イベント許可状態 ＞０：イベント禁止状態	－

※１：イベントの禁止／許可は多重制御します。
つまり、_Dis*XXX*Evt()と同じ回数 _Ena*XXX*Evt()を実行した時に、イベントの実行が許可状態となります。

22. ボタン

#	関数定義 内容	戻り値／※備考	サンプル
1	**_BtnOpen(BtnNo, $Caption)** ボタンを表示し、使用可能にします。 BtnNo　- ボタン識別番号（０～７） $Caption - ボタンフェース文字列	－ ※ボタンは、コントロールウインドの上部に表示します。	M020
2	**_BtnClose(BtnNo)** ボタンを削除します BtnNo - ボタン識別番号（０～７）	－	M020
3	**_BtnEnable(BtnNo, fEnable)** ボタンを有効化／無効化します BtnNo　- ボタン識別番号（０～７） fEnable - _TRUE　: ボタンを有効化（押せる状態にする） _FALSE : ボタンを無効化（押せない状態にする）	－	M020

23. チェックボックス

#	関数定義 内容	戻り値／※備考	サンプル
1	**_ChkBoxOpen(ChkBoxNo, $Caption)** チェックボックスを表示し、使用可能にします。 ChkBoxNo - チェックボックス識別番号（０～７） $Caption - チェックボックス・キャプション文字列	－ ※チェックボックスは、コントロールウインドの上部に表示します。	M010 M070 M110
2	**_ChkBoxClose(ChkBoxNo)** チェックボックスを削除します ChkBoxNo - チェックボックス識別番号（０～７）	－	M160
3	**_ChkBoxGetSts(ChkBoxNo)** チェックボックスをのチェック状態を取得します ChkBoxNo - チェックボックス識別番号（０～７）	チェック状態 _TRUE　: チェックしている _FALSE : チェックしていない	M160
4	**_ChkBoxSetSts(ChkBoxNo, Status)** チェックボックスをのチェック状態を取得します ChkBoxNo - チェックボックス識別番号（０～７） Status　- 設定するチェックステータス（_TRUE:チェック. _FALSE:未チェック）	－	M110
5	**_ChkBoxEnable(ChkBoxNo, fEnable)** チェックボックスを有効化／無効化します ChkBoxNo - チェックボックス識別番号（０～７） fEnable　- _TRUE　: チェックボックスを有効化（チェックできる状態にする） _FALSE : チェックボックスを無効化（チェックできない状態にする）	－	M160

24. タイムチャート・グラフ

#	関数定義 内容	戻り値／※備考	サンプル
1	**_TmcOpen([$Title] [,$Poroparty] [,x] [,y] [,cx] [,cy])** タイムチャートウインドをオープンします $Title - ウインドタイトル $Property - プロパティ文字列（**※１**） x, y - ウインド表示位置 cx, cy - ウインドのサイズ	－ ※既にオープン済の場合は 一旦ウインドをクローズして から再度オープンする。	M120
2	**_TmcSetItems(NumOfDataItems)** タイムチャート・グラフの データ項目数設定 NumOfDataItems - データ項目数（１～８）	－	M120
3	**_TmcSetRange(Low, High)** タイムチャート・グラフのレンジ設定 Low - グラフ低位の値 High - グラフ高位の値 ※グラフのレンジを自動調整する場合は、_TmcSetRange(0, 0); としてください。	－ ※Low > High でも OK ※Low = High の場合はグラフ レンジを自動調整する	M120
4	**_TmcPutData(d1 [,d2]・・・[, d8])** **_TmcPutData(data[])** タイムチャート・グラフへ データ投与 d1・・・d8 - グラフへ投与するデータ data[] - グラフへ投与するデータの配列（[1]：d1 ・・・[8]:d8）	－ ※各データ d1～d8 の描画色は以下 d1: ―― d5: ―― d2: ―― d6: ―― d3: ―― d7: ―― d4: ―― d8: ——	M120
5	**_TmcClear()** タイムチャート・グラフの クリアー	－	M120
6	**_TmcWndMove(x, y)** タイムチャート ウインドの移動 x, y - 移動先のスクリーン座標	－	M120
7	**_TmcWndSize(cx, cy)** タイムチャート 。ウインドサイズ設定 cx, cy - 設定するウインドのサイズ	－	M120
8	**_TmcClose()** タイムチャートウインドを クローズします	－	M120

※１：以下のパラメタ文字列を指定します。(何も指定しない場合は空文字列)

```
P: [L=fff], [H=fff], [B=nnn], [I=nnn], [A=nnn], [BC=nnn], [0=[nnn][/nnn]], ・・・ [7=[nnn][/nnn]]
```

文字列の先頭は「P:」でなければなりません。(「P:」の直後には空白を置けます)
「fff」は実数で、「nnn」は整数（１６進数の場合は先頭に'0x'を付加）で指定します。
各パラメタはカンマ（,）で区切ります。(カンマの前後には空白を置けます)
各パラメタの設定内容は、以下のとおりです。(各パラメタの指定順序は任意です)

キーワード	内　容
L	低位グラフレンジ
H	高位グラフレンジ
B	バッファ容量（バッファに格納するデータ数)
I	有効なデータ項目数（１～８）
A	平均化個数
BC	コントロール外枠の表示色
0～7	各データ項目の情報を「オフセット値／表示色」の形式で指定

表示色は、１６進数で「0xbbggrr」の形式で指定します（bb:青成分, gg:緑成分, rr:赤成分)

25. その他

#	関数定義 内容	戻り値／※備考	サンプル
1	**_ShowText($Text)** コントロールウインドの右上にテキストを表示します $Text - 表示するテキスト文字列	ー	M070 M071 M080
2	**_GetWndPosX()** コントロールウインドの左端位置（スクリーン座標）を取得します。	コントロールウインドの左端位置	M160
3	**_GetWndPosY()** コントロールウインドの上端位置（スクリーン座標）を取得します。	コントロールウインドの上端位置	M160
4	**_GetWndWidth()** コントロールウインドの幅を取得します	コントロールウインドの幅 （ピクセル数）	M160
5	**_GetWndHeight()** コントロールウインドの高さを取得します	コントロールウインドの高さ （ピクセル数）	M160
6	**_GetCharWidth()** 文字の幅を取得します	半角文字の幅 （平均ピクセル数）	M160
7	**_GetCharHeight()** 文字の高さを取得します	文字の高さ （ピクセル数）	M160
8	**_GetLineHeight()** 行の高さを取得します	行の高さ （ピクセル数）	M160
9	**_Exit([$Text])** マクロの実行を終了します $Text - このテキストは親ウインドから、AjcCipGetExitText()により取得できます。 ※_Exit()で指定するテキストは、言語上は何の効果もありません。 テキストの扱いは、このコントロールを使用するアプリケーションに依存します。 例えば、Exit()で、次に実行するマクロテキストファイルを指定したりできます。	ー	M071

26. デバッグ機能

#	関数定義 内容	戻り値／※備考	サンプル
1	**_VarDump([MaxArrNum])** トレース表示ウインドへ全ての変数情報を表示します MaxArrNum - 配列の最大表示項目数（未指定時は１０を仮定）	－	－
2	**_Trace ($fmt [,arg] ・・・)** トレース表示ウインドへ書式文字列を表示します。 $fmt - 書式文字列（C言語の ptintf()とほぼ同じ） arg - 書式（%・・）に対応した引数		－
3	**_Pause()** **※1** 全ての実行を一時停止します。 ※メイン処理およびイベント処理の全てが停止状態となります。 ※ステップトレースウインド未オープンの場合、メッセージボックスが表示され、 ＯＫボタンを押すと実行を再開します。 ※ステップトレースウインドがオープンされている場合は、停止した旨を表示し、 実行継続ボタン( )で再開します。	－	－
4	**_Stop()** **※1** メイン処理の実行を一時停止します。 ※メイン処理だけが停止状態となります。イベントは実行されます。 ※_Restart()を実行すると、メイン処理を再開します。	－	M010 M070 M080
5	**_Restart ()** _Stop()により中断していたメイン処理を再開します。	－	M010 M070 M080
6	**_StepTrace(fEnable)** ステップトレースの表示を許可／禁止します。 fEnable : TRUE - ステップトレースの表示を許可 FALSE - ステップトレースの表示を禁止	－ ※ステップトレース。ウインドが開かれていない場合は何もしません。	－

※1：_Pause()や_Stop()は、_Pause()/_Stop()を含むステップの次のステップ（_Stop()は次のメインコードステップ）で停止します。
（_Pause()/_Stop()も数値型関数（単に0を返すだけですが）なので、_Pause()/_Stop()を含む数式が実行された後に停止します）

27. サンプルマクロテキスト一覧

マクロテキスト	内容	備考
M010.txt	右クリックとポップアップメニュー，ユーザイベントの実行	
M020.txt	ロケート&プリント	
M030.txt	スクリーン情報	
M040.txt	配列操作	
M050.txt	メモリブロックの各操作	
M051.txt	メモリブロック（整数）アクセス	
M052.txt	メモリブロック（実数）アクセス	
M060.txt	ファイルのコピー&ベリファイ	
M061.txt	テキストファイルを読み出して 指定した文字列を含む行を表示する	
M062.txt	テキストファイルを読み出して 100ms／行の速度でＣＯＭポートへ送信	
M063.txt	指定フォルダ下のファイルを検索	
M064.txt	ファイルパスの分解，組み立て	
M070.txt	シリアル送受信とイベントの実行	
M071.txt	ＸＭＯＤＥＭ送受信	
M072.txt	ＹＭＯＤＥＭ送受信	
M073.txt	ＸＭＯＤＥＭ送信	
M074.txt	ＸＭＯＤＥＭ受信	
M075.txt	ＹＭＯＤＥＭ送信（フォルダ下の全ファイル）	
M076.txt	ＹＭＯＤＥＭ送信（フォルダ下の全「.C」ファイル	
M077.txt	ＹＭＯＤＥＭ送信（単一ファイル）	
M078.txt	ＹＭＯＤＥＭ受信（サブフォルダを作成しない）	
M079.txt	ＹＭＯＤＥＭ受信（サブフォルダを作成する）	
M080.txt	ＣＯＭポートの信号設定と読み出し	
M090,txt	時間，ウェイト	
M100.txt	算術演算	
M101.txt	３点から円の中心を求める	
M110.txt	ボタンとチェックボックス	
M120.txt	タイムチャート	
M130.txt	文字列操作	
M140.txt	フォルダ作成，パスチェック	
M150.txt	プロファイル（数値）の読み出し／書き込み	
M160.txt	チェックボックスの生成／消去	
M200.txt	「あじゃた～む」と接続して通信	
M500.txt	ボタンとチェックボックス，ＣＯＭ送信	AjmMacroJ.pdf で解説しているマクロです
M510.txt	シリアルポートを介して、テキストデータの送受信	
M520.txt	シリアルポートを介して、バイナリ・パケット・データの送受信	
M530.txt	ポート状態の永続化と、テキスト／バイナリの送受信	
M550.txt	右クリックとポップアップメニュー	
M551.txt	右クリックとポップアップメニュー（改良版）	
M560.txt	位相の異なる８つの正弦波を表示	
M570.txt	１秒毎に現在時刻のテキスト送信（イベントドリブン）	
M571.txt	１秒毎に現在時刻のテキスト送信（イベントドリブン未使用）	