V016002-703

# CANTOOL-A シリーズ

# **CANTOOL A1**

# [CANTOOL Script-Manual]

Rev. 01.02 2022-08-31

製品名	型番	対応全体バージョン
	CTA101A-DOEOT	Ver01.05.00
CANTOOL AT	CTA101A-DOE0	



#### 改定履歴

Revision	日付	変更内容
01.00	2018-03-26	初版(対応全体バージョン:Ver01.03)
01.01	2019-11-05	対応全体バージョン: Ver01.04
		・MultiPacket 送受信 API 追加
		・Python3.7.3 に対応(python_env_ver01.00.03)
01.02	2022-08-31	対応全体バージョン: Ver01.05
		・UART 送信 API 追加



(2/42)

#### <u>商標</u>

Microsoft、Windows、Visual Studio、Visual Basic、Visual C#、Visual C++は、米国 Microsoft Corporation(ある いは米国マイクロソフト・コーポレーション)の米国およびその他の国における登録商標です。(Windows の正式名称は Microsoft Windows Operating System です)

その他、記載されている会社名、製品名は、各社の商標または登録商標です。



(3/42)

# 目次

1. 概	要	5
1.1	システム概要	6
1.2	稼働条件	8
2. CA	ANTOOL Script について	9
2.1	CANTOOL Script の構成	9
2.2	CANTOOL Script で扱うインターフェース情報ついて	9
2.3	CANTOOL Script 利用時の作業流れ	10
3. 使	用方法	11
3.1	CANTOOL Script 使用準備	11
3.1.	.1 Python のインストール	11
3.1.	.2 Python 用 CANTOOL 制御ライブラリのインストール	
3.2	Python スクリプト作成手順	16
3.2.	.1 Python スクリプト作成	
3.2.	.2 Windows 上での Python スクリプトデバッグ	
3.3	Python スクリプト実行手順	
3.3.	.1 事前準備	
3.3.	.2 Python スクリプトファイルを本体へ設定する	
3.3.	.3 本体での実行	
3.3.	.4 実行結果の確認	
3.4	注意事項	
4. ライ	イブラリインターフェース仕様	



(4/42)

# 1. 概要

"CANTOOL Script"は CANTOOL A1 が提供する機能です。

CANTOOL A1 の CAN、AD/DA、GPIO などを Python によって制御することができます。

CANTOOL Script は、Windows 上で実行することも、CANTOOL A1 に Python スクリプトを設定してスタンドアロンで実行することもできます。

本書では、以降 CANTOOL A1 を"CANTOOL" と称します。

本書は、「V016002-700\_CANTOOL\_A1-Manual」「V016002-702\_CANTOOL\_A1\_SDK-Manual」を参照し CANTOOLの基本的な利用方法をご理解されている方向けの記載となっています。



(5/42)

## 1.1 システム概要

CANTOOL Script は、CANTOOL の I/O などを制御するための制御用関数ライブラリを提供します。 提供する制御用関数ライブラリを使用することにより、制御用アプリケーションと内部通信を行い Python スクリプトに従った制御を行います。

以下にシステム構成を示します。



図 1-1 CANTOOL システム構成図

表 1-1 システム構成要素一覧

名称	説明	
Python スクリプト	ユーザーにて作成される CANTOOL 制御用 Python のスクリプトコードです。	
RmtDll ライブラリ	ANTOOL を制御するための API が含まれたライブラリです。 Python スクリプトで Import して利用します。	
CANTOOL PANEL	ANTOOL 基本ソフトウェアです。	
CANTOOL アプリ		
Remote I/F	RmtDll ライブラリとのインターフェースです。	
USB Driver	CANTOOL PANEL と CANTOOL アプリが通信するためのドライバです。	

(6/42)



CANTOOL および Script のソフトウェア構成を以下に示します。

図 1-2 CANTOOL ソフトウェア構成図

表 1-2 ソフトウェア構成要素一覧

名称	説明	
スクリプトファイル	ユーザーが作成する Python スクリプトです。	
Python 実行環境(IDLE)	Python をインストールすることで利用可能なソフトウェアです。	
PC 用制御ライブラリ	CANTOOL を制御するための API が含まれたライブラリです。	
(RmtDll.pyd)	Windows 上で Python から利用されます。	
CANTOOL PANEL	CANTOOL 基本ソフトウェアです。	
Python 実行エンジン	CANTOOL 上で Python スクリプトを実行するソフトウェアです。	
制御ライブラリ(RmtDll)	CANTOOL を制御するための API が含まれたライブラリです。	
	CANTOOL 上で Python 実行エンジンから利用されます。	

(7/42)

# 1.2 稼働条件

#### 対象オペレーティングシステム

CANTOOL Script は、以下の Microsoft Windows オペレーティングシステムに対応しています。

- Windows 8.1 32bit/64bit
- Windows 10 32bit/64bit

#### 動作確認済み開発環境

CANTOOL Script は、以下の開発環境を使用した動作を確認済みです。

•Python 3.5.4



(8/42)

# 2. CANTOOL Script について

# 2.1 CANTOOL Script の構成

CANTOOL Script として以下のものが用意されています。 なお、最新データについては、取扱説明書の「製品サポートページ」の章を参照して、入手してください。

Script-root

-	Man	ual	CANTOC	)L Script マニュアル (本書)
-	Libra	ary	ライブラリ	
-	-	RmtDll.pyd	Window	s 上での動作用
-	Head	der	ヘッダーファ	ッイル
-	-	ctdef	スクリプト関	國数用定義値
-	Sam	ple		各種サンプルプログラム
	-	readme_001R_sample.pdf	サンプル 1	の説明
	-	:		
	-	001R_sample	サンプル 1	(Python) * 1
	-	:		

\*1:サンプルプログラム名について

XXXY\_ZZZ
 XXX:サンプル番号 001~
 Y :開発環境 R=Python用
 ZZZ:サンプルプログラム名

\*Library、Hederのファイルは、実行環境に合わせてコピーして使用してください。

### 2.2 CANTOOL Script で扱うインターフェース情報ついて

「V016002-703\_CANTOOL\_A1\_Script-ReferenceManual-API」を参照してください。



(9/42)

### 2.3 CANTOOL Script 利用時の作業流れ

- Pythonスクリプトファイル作成する。 テキストエディタ等により、スクリプトファイルを作成する。 処理結果等を print 命令で、実行ログとして出力する。
- (2) Pythonスクリプトファイルのデバッグを行う。 Windows PC にて、Python スクリプトファイルをデバッグ実行して動作確認を行う。 デバッグ実行は Python に標準で入っている、IDEL ツール等でステップ実行や連続実行して確認する。
- (3) PythonスクリプトファイルをCANTOOLに反映させる。

CANTOOL PANELを使用し、ユーザスイッチに割り当てるスクリプトを設定し、反映を行う。

(4)CANTOOLのユーザスイッチ操作でスクリプトファイルを実行する。

Python スクリプトファイルを割り当てたユーザスイッチを押して、Python スクリプトファイルを実行する。

(5)実行状態は、CANTOOL3のLED点灯で確認する。 CANTOOL本体のLEDの点灯状態により、実行中/正常終了/異常終了を確認する。

(6)実行結果をログファイルで確認する。

実行終了後、CANTOOL PANELで本体ログを取得し、実行結果をログファイルにより確認する。



(10/42)

# 3. 使用方法

### 3.1 CANTOOL Script 使用準備

ここでは Windows 上で Python スクリプトを作成・デバッグするための準備を行います。

# 3.1.1 Python のインストール

#### 使用する Python のバージョンは「3.7.3」です。(Python 環境: Ver01.00.03 時)

- (1) Python のインストールモジュールをダウンロードする。
- 以下のダウンロード用 URL ヘブラウザでアクセスする。



図 3-1 Python3.5.4 ダウンロード(1/4)



(11/42)

Python	PSF	Docs	РуРІ	Jobs	Community
🥏 pyt	hon™		Q Search		GO Socialize
About	Downloads	ocumentation Co	mmunity Succes	s Stories News	Events
Download	All releases Source code	Download	for Windows		
Download Python	Windows	Note that P	ython 3.5+ cannot be use	d on Windows XP	
2 and 3.	Mac OS X	or earlier.	ou are looking for? Putho	a can be used on	
Looking for Python	Other Platforms	many operat	ting systems and environr	nents.	
Other	License	View the full	list of downloads.		
Want to help test de	Alternative Implementa	tions			

「Downloads」の選択から、「Windows」を選択してバージョン一覧を表示する。

図 3-2 Python3.5.4 ダウンロード(2/4)

「Python 3.5.4 - 2017-08-08」の、「Download Windows x86 executable installer」(32ビット版)を選択する。



#### 図 3-3 Python3.5.4 ダウンロード(3/4)

(12/42)



「ファイルを保存」を選択にしてファイルに保存する。



図 3-4 Python3.5.4 ダウンロード(4/4)

(2)Pythonをインストールする。

ダウンロードした「python-3.5.4.exe」を開き、「実行」を選択してインストールを実行する。



図 3-5 Python3.5.4 インストール(1/6)

「Add Python 3.5 to PATH」をチェックして、「Install Now」を選択してインストールを開始する。



図 3-6 Python3.5.4 インストール(2/6)

(13/42)



インストール実行中になる。	,
---------------	---

	Setup Progress	
ę	Installing: Python 3.5.4 Core Interpreter (32-bit)	
python		
windows	Cance	ł

図 3-7 Python3.5.4 インストール(3/6)

インストール成功で、「Close」を選択してインストールを終了する。

Python 3.5.4 (32-bit) Setu	ip 📃 🔤 💌
	Setup was successful
-	Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.
	New to Python? Start with the <u>online tutorial</u> and <u>documentation</u> .
marker -	
python	
windows	Close

図 3-8 Python3.5.4 インストール(4/6)



(14/42)

(3)正しくインストールされたかを確認する。

DOSプロンプトを開き、インストール確認する。

"python --version"を入力して、"Python 3.5.4"の表示を確認する。

"python"を入力して、"Python 3.5.4"が実行される事を確認する。



図 3-9 Python3.5.4 インストール(5/6)

このインストールで Python 3.5.4 自身と付属のデバッガとして IDEL がインストールされます。 以下はスタートメニュー抜粋のインストール状態です。



図 3-10 Python3.5.4 インストール(6/6)

以上で、Python 3.5.4 のインストールが完了です。

# 3.1.2 Python 用 CANTOOL 制御ライブラリのインストール

Python で CANTOOL を制御するための制御ライブラリのインストール手順を説明します。

CANTOOL リリースイメージの Script¥Library 内の RmtDll.pyd を Python の実行するカレントフォルダまたは Python の実行環境にコピーします。 "C:¥Users¥(ユーザ名)¥AppData¥Local¥Programs¥Python¥Python35-32¥DLLs"フォルダに RmtDll.pyd をコピーすれば、カレントフォ ルダに関係なく使用できます。

(15/42)

# 3.2 Python スクリプト作成手順

ここでは、Windows 上でのスクリプトの作成とデバッグの手順を説明します。

# 3.2.1 Python スクリプト作成

Python スクリプトはテキストエディタもしくは、Python 関連のツールで作成します。 ここでは、Python 付属のデバッガである IDEL を使用した例を説明します。 IDLE は命令強調表示したり、インデントを自動で行ったりする機能があり、Python スクリプトの作成を助けてくれます。

※IDELの操作詳細については、下記の「25.5. IDLE」を参照ください。※URL: <u>https://docs.python.jp/3/library/idle.html</u>

#### (1)IDLEの起動

スタートメニューの「Python 3.5.4」のIDELを選択してIDELを起動します。



図 3-11 Python スクリプト作成手順(1/8)

以下が「Python 3.5.4 Shell」画面です。



図 3-12 Python スクリプト作成手順(2/8)



(16/42)

(2)編集画面を開く

「File」メニューの「New File」を選択して、編集画面を表示します。

ne out snell	Debug Op	uons window Help
New File	Otrl+N	8, Aug 8 2017, 02:07:06) [MSC v.1900 32 bit (Intel)]
Open	Ctrl+O	or "license()" for more information.
Open Module	Alt+M	
Recent Files	•	
Class Browser	Alt+C	
Path Browser		
Save	Ctrl+S	-
Save As	Ctrl+Shift+S	
Save Copy As	Alt+Shift+S	
Print Window	Ctrl+P	
Close	Alt+F4	
Exit	Ctrl+Q	

#### 図 3-13 Python スクリプト作成手順(3/8)

#### 以下が編集画面です。

#### 図 3-14 Python スクリプト作成手順(4/8)



#### (3)Pythonスクリプトコードの入力

CANTOOL 制御用 Python スクリプトの基本的な記載方法として、本体バージョンを取得する Python スクリプトの例を以下に示します。 ※より実践的な利用方法については、CANTOOL リリースイメージ内の Script¥Sample 内を参照ください。



図 3-15 Python スクリプト作成手順(5/8)



(18/42)

#### (4) Pythonスクリプトファイルの保存

編集終了で、「File」メニューの「Save As...」を選択しファイルに保存します。



#### 図 3-16 Python スクリプト作成手順(6/8)

ファイル名を指定して、保存します。

各前を付けて保存	
	Python      マ      チ     Pythonの検索     P
整理 ▼ 新しいフォルダー	8== 👻 🔞
<ul> <li>☆ お気に入り</li> <li>♪ ダウンロード</li> <li>型 最近表示した場所</li> <li>■ デスクトップ</li> <li>▲ OneDrive</li> <li>■ ライブラリ</li> <li>▲ ドキュメント</li> <li>■ ピクチャ</li> <li>■ ピグチャ</li> <li>■ ビデオ</li> <li>→ ミュージック</li> </ul>	ドキュメント ライブラリ 並べ替え: フォルダー・ Python 名前 ▲ Python (C:¥ユーザー¥shinmura¥マイドキュメント)(1) ■ Sample.py
<ul> <li>■ コンピューター</li> <li>▲ ローカル ディスク (C:</li> <li></li></ul>	・ (*.py;*.pyw) (保存(S)) キャンセル

図 3-17 Python スクリプト作成手順(7/8)



(19/42)

ファイル保存でファイル名が表示されます。

Version.py - C:/Users/	
File Edit Format Run Options Window Help	
#from struct import * Import RmtDII Import sys	*
<pre>#script start ret = RmtDll.Scr_Start() print('Scr_Start()={0}'.format(ret)) sys.stdout.flush()</pre>	
<pre>#get status ret,stat = RmtDII.Scr_GetStatus(); print('Scr_GetStatus()={0}'.format(ret)) print('Version={0}'.format(stat[70]))</pre>	

#### 図 3-18 Python スクリプト作成手順(8/8)

(5)終了時は、各画面を閉じてください。



(20/42)

# 3.2.2 Windows 上での Python スクリプトデバッグ

ここでは Windows 上での Python スクリプトのデバッグ手順を説明します。 Python インストールでインストールされている IDEL デバッガを使用して説明します。

#### (1)CANTOOL PANEL起動

Windows上でCANTOOLの制御ライブラリを使用する場合は、CANTOOL PANELをあらかじめ起動しておく必要があります。 CANTOOL PANELを起動せずに実行すると。後述のScr\_Start関数でエラーとなります。

#### (2)IDLE起動

スタートメニューの「Python 3.5.4」のIDELを選択してIDELを起動します。



#### 図 3-19 Python スクリプトデバッグ手順(1/21)

以下が「Python 3.5.4 Shell」画面です。

le Edit Shell Debug Options Window F	lelp						
vthon 3.5.4 (v3.5.4:3f56838, Aug 8 2017 on win32 /pe "copyright", "credits" or "license() >>	, 02:07:06) ″for more	[MSC )	7.1900 ation.	32 bi	t (I	ntel)]	

図 3-20 Python スクリプトデバッグ手順(2/21)



#### (3)デバッグ対象ファイルを開く

デバッグするPythonスクリプトファイルを「File」メニューの「Open..」を選択して開きます。



#### 図 3-21 スクリプトデバッグ手順(3/21)

ファイ名を選択して、Python スクリプトファイルを開きます。

整理▼ 新しいフォルダー	i
<ul> <li>☆ お気に入り</li> <li>♪ ダウンロード</li> <li>型 最近表示した場所</li> <li>■ デスクトップ</li> </ul>	ドキュメント ラ… 並べ替え: フォルダー▼ Python 名前
ConeDrive	Version.py
📑 ドキュメント	
E ピクチャ	
📙 ビテオ 🎝 ミュージック	
<del>.</del> (	

図 3-22 Python スクリプトデバッグ手順(4/21)



(22/42)

編集画面に Python スクリプトファイル内容が読み込まれ、表示されます。



図 3-23 Python スクリプトデバッグ手順(5/21)



(23/42)

(4)デバッグ画面を開く

ステップ実行確認をする場合は、「Debug」メニューの「Debugger」を選択してデバッグ設定を行います。

File Edit Sher	Debug Options Window	/ Help
Python 3.5.4 (v: on win32 Sype ″copyriget	Go to File/Line Debugger	7, 02:07:06) [MSC v.1900 32 bit (Intel)]
»»>	Stack Viewer Auto-open Stack Viewer	

#### 図 3-24 Python スクリプトデバッグ手順(6/21)

以下のように、4つ全てにチェックしてデバッグ設定を行います。

🛓 Debug Control	
Go Step Over Out Quit 🔽 Stack 🖾 Sour	ce als
(None)	*

#### 図 3-25 Python スクリプトデバッグ手順(7/21)

上記チェックで Shell 画面には「DEBUG ON」が表示されます。

Python 3.5.4 Shell			23	
File Edit Shell Debug Options Window Help				
Python 3.5.4 (v3.5.4:3f56838, Aug 8 2017, 02:07:06) [MSC v.1900 32 b on win32 Type "copyright", "credits" or "license()" for more information. [DEBUG ON]	oit (	(Intel)	ני	

#### 図 3-26 Python スクリプトデバッグ手順(8/21)

#### (5)デバッグの開始

デバッグの開始を、編集画面の「Run」メニューの「Run Module」を選択して行います。

File Edit Forma	Run Options Window Help	
from struct impor	Python Shell	
mport sys	Check Module Alt+X	
lscript start 🛛 🐧	Run Module F5	

図 3-27 Python スクリプトデバッグ手順(9/21)



		23
Go Step C	Over Out Quit V Stack V Source	
Version.py:2	:: <module>()</module>	
bdb'.run(), l	ine 431: exec(cmd, globals, locals)	-
> 'main	. <module>(Cline 2: import RmtDl)</module>	
		-
	Locals	
None	Locals	 *
None	Locals	^ +
None	Locals Globals	*
None	Locals Globals <module 'builtins'="" (built-in)=""></module>	 *
None builtins doc	Locals Globals <module 'builtins'="" (built-in)=""> None</module>	 *
None builtins doc file	Locals Globals <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥</module>	*
None builtins doc file loader	Locals Globals Clobals	 *
None builtins doc file loader name	Locals Globals <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ Guiltinitinition:py' <class '_frozen_importlib.builtinimporter'=""> 'main'</class></module>	•
None builtins doc file loader name package_	Locals         Globals            'C:¥¥¥¥Users¥¥¥¥         ¥¥¥nts¥¥¥¥Python¥¥¥¥Version.py' <class '_frozen_importlib.builtinimporter'="">         'main'         None</class>	*

#### 図 3-28 Python スクリプトデバッグ手順(10/21)

Shell 画面には Python スクリプト開始メッセージが表示されます。



図 3-29 Python スクリプトデバッグ手順(11/21)



(25/42)

#### (6)デバッグ実行

デバッグ画面操作で、デバッグを開始します。

「Over」でプログラムを進め、6 行目に停止した状態です。

🎍 Debug Control	23
Go Step Over Out Quit Go Step Over Out Quit Go Locals Version.py:6: <module>()</module>	
'bdb'.run(), line 431: exec(cmd, globale, lecals) > 'main'. <module>(, line 6: ret = RmtDll.Scr_Start()</module>	

#### 図 3-30 Python スクリプトデバッグ手順(12/21)

編集画面の6行目で停止状態表示。



#### 図 3-31 Python スクリプトデバッグ手順(13/21)



(26/42)

デバッグ操作は以下の操作で行います。

「Go」ボタン

プログラムをブレークポイントの直前まで実行。ブレークポイントがない場合は、最後まで実行される。

「Step」ボタン

現在の行を実行して、次の行で停止します。現在の行に関数呼び出しがあるとデバッガーはその関数の中に入ります。

「Over」ボタン

現在の行を実行して、次の行で停止します。現在の行に関数呼び出しがあるとその関数を実行してから次の行で停止します。 「Out」ボタン

現在の関数を最後まで実行して、関数の外に移ります。Step で関数の中に入った場合、Out で関数の外に出ることができます。 「Quit」ボタン

デバッグを終了します。残りのコードは実行されません。もう一度、デバッグを開始するには、最初と同じように、 Debug > Debugger を選択します。

Step と Over の違いは、現在の行に関数呼び出しがある場合に、関数の中に入るかどうかです。通常は、関数の中に入らない Over の方をよく用います。



(27/42)

「Over」でプログラムを進め6行目の関数実行後、7行目に停止した状態です。

「Globals」欄で、変数等の使用した状態が確認できます。下図では、6 行目の関数の戻り値が 0 であることがわかります。

	ontrol	
Go Step (	Over Out Quit	
Version pv: 7		
version.py./		
'bdb'.run(), l	ne 431: exec(cmd, globals, locals)	-
> 'main	'. <module>(), line 7: print('Scr_Start()={0}'.format(ret))</module>	
		-
	Locals	1.100
None	Locals	
None	Locals	
None	Locals Globals	, A.,
None	Locals Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""></module>	
None RmtDll builtins	Locals Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""></module></module>	
None RmtDll builtins	Locals Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None</module></module>	
None RmtDll builtins file	Locals  Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥</module></module>	
None RmtDll builtins doc file loader	Locals Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ {<class '_frozen_importlib.builtinimporter'=""></class></module></module>	
None RmtDll builtins doc file loader name	Locals  Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ SY¥¥*Sers*Y¥¥*Sers*Y¥¥*Sersion.py' <class '_frozen_importlib.builtinimporter'=""> 'main'</class></module></module>	
None RmtDll builtins doc file loader name package_	Locals  Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None  'C:¥¥¥¥Users¥¥¥4 ¥¥¥nts¥¥¥¥Python¥¥¥¥Version.py' <class '_frozen_importlib.builtinimporter'=""> 'main' None</class></module></module>	
None RmtDllbuiltinsdocfileloadernamepackagespec	Locals  Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥4 ¥¥¥nts¥¥¥¥Python¥¥¥¥Version.py' <class '_frozen_importlib.builtinimporter'=""> 'main' None None</class></module></module>	
None RmtDll builtins file loader name package_ spec ret	Locals  Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥# ¥¥¥nts¥¥¥¥Python¥¥¥¥Version.py' <class '_frozen_importlib.builtinimporter'=""> 'main' None None 0</class></module></module>	

#### 図 3-32 Python スクリプトデバッグ手順(14/21)

#### 現在の停止位置です。

File Edit Format Run	Options Window Help	
#from struct import * import RmtDll import sys		
#script start		

#### 図 3-33 Python スクリプトデバッグ手順(15/21)



「Step」操作により、"print('Scr\_Start()={0}'.format(ret))"行で標準ライブラリ内にステップインして、 別の編集画面で表示された状態です。 抜ける時は、「Out」操作で元の Python スクリプト編集画面に戻ります。

🍃 PyShell.py - C:¥Users¥ 🔤 ¥AppData¥Local¥Programs¥Python¥Python... 🗖 🔍 🔀 File Edit Format Run Options Window Help **Oproperty** def encoding(self): return self.\_encoding @property def name(self): return '<%s>' % self.tags def isatty(self): return True class PseudoOutputFile(PseudoFile): def writable(self): return True def write(self, s):
 if self.closed: ir self.closed:
 raise ValueError("write to closed file")
if type(s) is not str:
 if not isinstance(s, str):
 raise TypeError('must be str, not ' + type(s).\_\_name\_\_)
 # See issue #19481
 s = str.\_str\_\_(s)
 return colf chell write(c, colf tere) s = str.\_str\_(s) return self.shell.write(s, self.tags) class PseudoInputFile(PseudoFile): \_\_init\_\_(self, shell, tags, encoding=None): PseudoFile.\_\_init\_\_(self, shell, tags, encoding) self.\_line\_buffer = def def readable(self): return True def read(self, size=-1): if self.closed: Ln: 1 Col: 0

#### 図 3-34 Python スクリプトデバッグ手順(16/21)

7行目の"print('Scr\_Start()={0}'.format(ret))"行によるログ表示内容です。



図 3-35 Python スクリプトデバッグ手順(17/21)



(29/42)

上記以外に、ブレーク行を指定して実行を停止する事もできます。

ブレークしたい行で、右クリックによりポップアップ表示で指定できる。

ブレーク行設定で、選択行のバックが黄色表示になります。



図 3-36 Python スクリプトデバッグ手順(18/21)

「Go」選択で最後まで実行する。

最後まで実行すると、下記の確認画面が表示されます。

「OK」を選択すると、実行ログを表示した Shell 画面とデバッグ操作画面が消えます。

「キャンセル」を選択すると、実行ログを表示した Shell 画面とデバッグ操作画面は残ります。



図 3-37 Python スクリプトデバッグ手順(19/21)



	ontrol	23
Go Step C	Over Out Quit V Stack V Source	
halk' mus() i	ne 421 ( superformation labele)	-
>' main	<pre>i=431: exec(criid, globals, locals) '.<module>(), line 18: RmtDll.Scr Finish( RmtDll.OK(), RmtDll.ORG_ON(), 'Script</module></pre>	c
	s services (1) interesting and the services ⊂ minute (1) services and (1) interesting of (1).	
ŝ.	Locals	1
s Ene		_
None		
None		
None	Globals	
RmtDll	Globals	
None RmtDll	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""></module>	
None RmtDll builtins	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""></module></module>	
None RmtDll builtins doc	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None</module></module>	
None RmtDll builtins doc file	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ 'C:¥¥¥¥Users¥¥¥¥ output former interation of the public terms that</module></module>	
None RmtDll builtins doc file loader	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ <class '_frozen_importlib.builtinimporter'=""></class></module></module>	
None RmtDll builtins doc file loader name	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ <class '_frozen_importlib.builtinimporter'=""> 'main'</class></module></module>	
None RmtDllbuiltinsdocfileloadernamepackage	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ <class '_frozen_importlib.builtinimporter'=""> 'main' _None</class></module></module>	
None RmtDllbuiltinsdocfileloadernamepackagespec	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ <module 'builtins'="" (built-in)=""> C:¥¥¥¥Users¥¥¥¥ <module 'builtins'="" (built-in)=""> None None None</module></module></module></module>	
None RmtDllbuiltinsdocfileloadernamepackagespec ret	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ class '_frozen_importlib.BuiltinImporter'&gt; 'main' _None None 0</module></module>	
None RmtDllbuiltinsdocfileloadernamepackagespec ret stat	Globals <module 'c:¥¥5-32¥¥¥¥dlls¥¥¥¥rmtdll.pyd'="" 'rmtdll'="" from=""> <module 'builtins'="" (built-in)=""> None 'C:¥¥¥¥Users¥¥¥¥ 'C:¥¥¥¥Users¥¥¥¥ (class '_frozen_importlib.BuiltinImporter'&gt; 'main' None None 0 [131672063842900000, '01.03751632910000000, '01.00.01']</module></module>	

#### 実行終了で、「キャンセル」を選択した時の Shell 画面です。

図 3-38 Python スクリプトデバッグ手順(20/21)



(31/42)

実行終了時の操作画面です。Python バージョンがログに出力されています。 Scr\_Finish 関数の結果はログ出力されません。



図 3-39 Python スクリプトデバッグ手順(21/21)



(32/42)

# 3.3 Python スクリプト実行手順

CANTOOL 本体での Python スクリプト実行手順を説明します。 なお、Python スクリプトの実行には SD カードが必要となります。

### 3.3.1 事前準備

(1)CANTOOLにPythonスクリプト実行環境を構築する

CANTOOL PANELを起動します。

「Setting」を選択します。



図 3-40 Python スクリプト実行手順(1/9)

「プロジェクト設定」で「スタンドアロン設定」を選択します。

CANTOOLにSDカードを挿入し、「スクリプト実行環境」「構築」ボタンを選択します。

プロジェクト設定 インターフェース設計 スタンドアロン設定 環境設定 🏜 💽 主
実行データ操作 SW1 スクリプト · C:¥Users¥shinmura¥Documents¥Python¥Version.py … □連続再生 再生データ ZAUプト · C:¥Users¥shinmura¥Documents¥Python¥Sample2.py … □連続再生 SW3 スクリプト · C:¥Users¥shinmura¥Documents¥Python¥Sample3.py … □連続再生 本体実行データ操作 取得 スクリプト実行環境 構築
ログ ログ設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥project¥Standalor … ログ分割サイズ 1~1024MB 3 フィルク設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥f … フィルク設定 本体ログ操作 取得 削除
r設定送信 スタンドアロン設定を外部メディアへ転送 転送

図 3-41 Python スクリプト実行手順(2/9)

(33/42)



Python実行環境のインストールファイルを指定します。

CANTOOL の¥Support¥ScriptImage¥内にある python\_env\_verXX.XX.XX.zip を選択し、「開く」ボタンを選択します。 ※XX はバージョンを表す数字



図 3-42 Python スクリプト実行手順(3/9)

構築が完了すると以下のメッセージが表示されます。



図 3-43 Python スクリプト実行手順(4/9)



# 3.3.2 Python スクリプトファイルを本体へ設定する

(1) CANTOOL PANELでPythonスクリプトファイルを設定する。

「プロジェクト設定」で「スタンドアロン設定」を選択します。

各ユーザスイッチに Python スクリプトファイルを設定します。

「実行データ操作」で「スクリプト」を選択し、実行する Python スクリプトファイルを指定します。

画面例ではユーザスイッチ1に4章で作成したスクリプトファイルを指定しています。

プロジェクト設定 インターフェース設定 スタンドアロン設定 環境設定  ユタンドアロン設定  環境設定  ユタンドアロン設定  ロショース
第二日     1001000000000000000000000000000000000
スタンドアロン設定を反映 反映
スタンドアロン設定を外部メディアへ転送転送

#### 図 3-44 Python スクリプト実行手順(5/9)



(35/42)

(2)設定したPythonスクリプトファイルを本体に設定します。

「スタンドアロン設定を反映」「反映」を選択します。

プロジェクト設定 インターフェース設定 スタンドアロン設定 環境設定 12ターフェース設定 スタンドアロン設定 環境設定
実行データ操作 SW1 スカリプト・ C:¥Users¥shinmura¥Documents¥Python¥Version.py … □ 連続再生 SW2 スカリプト・ C:¥Users¥shinmura¥Documents¥Python¥Sample2.py … □ 連続再生 SW3 スカリプト・ C:¥Users¥shinmura¥Documents¥Python¥Sample3.py … □ 連続再生
本体実行データ操作取得スカリプト実行環境構築
ログ ログ設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥project¥Standalor … ログ分割サイズ 1~1024MB 3 フィルク設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥r … フィルク設定 本体ログ操作 取得 削除
設定送信 スタンドアロン設定を外部メディアへ転送 <b>転送</b>

図 3-45 Python スクリプト実行手順(6/9)

反映が完了すると以下のメッセージが表示されます。

CANTOOLのSWに対応するLEDが緑点灯し、Pythonスクリプトが実行可能状態であることを示します。

反映完了	
	ОК

図 3-46 Python スクリプト実行手順(7/9)

# 3.3.3 本体での実行

実行したい Python スクリプトを登録した CANTOOL の SW を押下します。

実行開始されると対応する LED が緑点滅します。

緑点滅中に SW を押下した場合は、スクリプトを強制終了します。

Python スクリプトの実行が終了すると、Scr\_Finish()関数で指定した LED 状態になります。

システムやスクリプトの問題で Python スクリプトが停止した場合は LED が赤点灯状態になります。



(36/42)

## 3.3.4 実行結果の確認

#### (1) Pythonスクリプトの実行ログ取得

「スタンドアロン設定」の「本体ログ操作」>「取得」を選択する。

プロジェクト設定 インターフェース設定 スタンドアロン設定 環境設定  立
<ul> <li>実行データ操作</li> <li>SW1 スクリプト・ C:¥Users¥shinmura¥Documents¥Python¥Version.py</li> <li>ご ■連続再生</li> <li>SW2 スクリプト・ C:¥Users¥shinmura¥Documents¥Python¥Sample2.py</li> <li>ご ■連続再生</li> <li>SW3 スクリプト・ C:¥Users¥shinmura¥Documents¥Python¥Sample3.py</li> <li>ご ■連続再生</li> <li>本体実行データ操作</li> <li>取得</li> <li>スクリプト実行環境</li> <li>構築</li> </ul>
ログ設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥project¥Standalor … ログ分割サイズ 1~1024MB 3 フィルタ設定ファイル C:¥Users¥shinmura¥Desktop¥連続試験用¥連続試験用¥CANTOOL_PANEL¥CANTOOL_PANEL¥; … フィルク設定 本体ログ操作 取得 削除
設定送信 スタンドアロン設定を外部メディアへ転送 転送

図 3-47 Python スクリプト実行手順(8/9)

#### ログの保存先を選択(任意)し、「OK」ボタンを選択します。

保存先フォルダ内に"Log\_YYYYMMDDhhmmss¥ScriptLog¥SW1¥0001~N¥"(※1)内に拡張子"txt"と"result"のログファイルが出来ます。

※1 YYYYMMDDhhmmssはログ取得した年月日時分秒、Nは0001から実行の度連番となる。

> 🕌 お気に入り	*
▷ 🚵 スタート メニュー	
▷ 🚺 ダウンロード	
-  -  -  -  -  -  -  -  -  -  -  -  -	1000
> 퉲 Android Workspace	E
A 🎽 CANTOOL	
<ul> <li>Image: project</li> </ul>	
👂 🌆 DataFromCantool	
Log_20180403143451	
Play_20180403164525	
< III	

#### 図 3-48 Python スクリプト実行手順(9/9)

(37/42)



拡張子"txt"のファイルには、Python スクリプトで print 出力した内容がログ出力されます。

Scr\_Start 01.03.00 2018/04/02 : script\_RmtDll\_Version Scr\_Start()=0 Scr\_GetStatus()=0 Version=01.00.01 Scr\_GetStatus() : OK RmtDll.Scr\_End=0

拡張子"result"のファイルには、後述の Scr\_Finish 関数で指定した出力がログ出力されます。

OK,ORG\_ON,Script OK



(38/42)

# 3.4 注意事項

特になし



(39/42)

# 4. ライブラリインターフェース仕様

CANTOOL Script が提供する API の一覧を以下に示します。各 API の詳細は

「V016002-703\_CANTOOL\_A1\_Script-ReferenceManual-API」を参照してください。

### 表 4-1 ライブラリインターフェース一覧

カテゴリ	説明	名称	
スタート	スクリプト使用開始通知処理	Scr_Start	
エンド	スクリプト使用終了処理	Scr_End	
ステータス	各種状態を取得する。	Scr_GetStatus	
	指定の LED を制御する。	Scr_SetLED	
	スクリプト終了時に、スクリプトを登録した SW に対応した LED を	Scr_Finish	
	制御する。		
	また、実行ログとは別に結果ログに文字列を出力する。		
	結果 OK を判定する文字列を返す。Scr_Finish の引数①用。	ОК	
	結果 NG を判定する文字列を返す。Scr_Finish の引数①用。	NG	
	LED を OFF に制御する文字列を返す。Scr_Finish の引数②	LGT_OFF	
	用。		
	LEDを緑点灯に制御する文字列を返す。Scr_Finishの引数②	GRN_ON	
	用。		
	LEDを緑点滅に制御する文字列を返す。Scr_Finishの引数②	GRN_FLS	
	用。		
	LED を橙点灯に制御する文字列を返す。 Scr_Finish の引数②   _	ORG_ON	
	LEDを橙点滅に制御する文字列を返す。Scr_Finishの引数②	ORG_FLS	
	LED を亦点滅に制御する文字列を返す。SCr_Finish の引数(2)	RED_FLS	
	Frameに割り目(られている Signal のリストを取得する。	Scr_ConvertSignalListFromFrame	
	BUSID と XID から FrameID を取得する。	Scr_ConvertXID2FrameID	
	BUSID と XID と Signal Symbol から Signal ID を取得する。	Scr_ConvertXID2SignalID	
	本体のUARTの設定の変更を行う。	Scr_SetUartSetting	
		Scr_GetUartSetting	
	指定したノレームIDのテータサイス(ハイト)を取得する。	Scr_GetFrameSize	
	指正しにングナルIDのテーダザイス(ハイト)を取得する。	Scr_GetSignalSize	
テータ达信	指定ノレームのテータを达信する。周期达信の個に反映される。	Scr_SendFrameData	
	指定シリナルのテータを达信する。周期达信の他に反映される。	Scr_SendSignalData	
		Scr_SendDaData	
		Scr_SendGpoData	
		Scr_SendAnyCANFrame	
	」 达信テーダを Frame サイムに分割し、指正周期で送信を行う。	Scr_SendCyclicData2	
	MultiPacket データを送信する。	Scr_SendMultiPacketData2	
	MultiPacketへの応答内容設定。	Scr_ResponseMultiPacketData	

(40/42)

カテゴリ	説明	名称	
	UART へのデータを送信す。	Scr_SendUartData	
データ受信	指定フレームの最新データを取得する。	Scr_GetFrameData	
	指定シグナルの最新データを取得する。	Scr_GetSignalData	
	指定ポートの AD の値を取得する。	Scr_GetAdData	
	指定ポートの GPI の値を取得する。	Scr_GetGpiData	
	MultiPacket 状態を取得する。	Scr_GetMultiPacketStatus	
	MultiPacket の応答結果を取得する。	Scr_GetResponseMultiPacketDataS	
		tatus	



(41/42)

# アイテック阪急阪神株式会社

Copyright (C) 2017 ITEC HANKYU HANSHIN CO., LTD. All Rights Reserved.



(42/42)



(※)各エラーや定義、	_構造体については、同梱の「c	tdef.py」を参照してください。				Rev. 01.02 最終更新日: 2022/08/31
種別 Connection	定義 ISor Start	概要 コーザーアプロをCANTOOL本体	引数	設定可能範囲	取得内容(戻り値) ①教教刊・CT result	説明 ・ 木APIにとりCANTOOL木体アプリに接続すること
		アプリと接続する。			ctdef.pyのCT_resultを参 照	で、スクリプトからAP1でCANTOOL本体が制御可能 となる。複数のスクリプト(最大4個)を接続するこ とが可能。 ・すでにCANTOOL本体に読み込んでいるプロジェク
						トファイルを使用する。
	Scr_End	ユーサーアフリをCANIOOL- PANELから切断する。	無し	-	①整数型:GI_result ctdef.pyのCT_resultを参	・本APTICよりCANTOOL本体アフリからスクリフト を切断し、接続していたスクリプトの情報を破棄
					泯	する。 ・Sor_End()を呼び出さずにスクリプトが終了した 場合、その後のスクリプト動作は保証しない。 CANTOOL本体を再起動すること。ただし、本体のSW を押して終了した場合は再起動の必要はない。
Status	Scr_GetStatus	各種状態を取得する	無し (無し)	-	①整数型:CT_result ctdef.pyのCT_resultを参 照	・statusのリストの項目はCT_Configを使って取得 する。
					②リスト型:status 取得できる情報は ctdef.pyのCT_Configを参照	
	Scr_SetLED	指定のLEDを制御する	①整数型 LED port [1] LED番号 ②整数型 LED color [1] 色 ③整数型 LED Status [1] 動作 (点灯、点滅、消灯)	T SCR_LED_PORT_USER1 SCR_LED_PORT_USER2 2 SCR_LED_COLOR_OFF SCR_LED_COLOR_RED SCR_LED_COLOR_GREEN SCR_LED_COLOR_ORANGE 3 SCR_LED_STATUS_OFF SCR_LED_STATUS_ON SCR_LED_STATUS_BRINK	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定したLED番号の色と動作を制御する。 ・色と動作のどちらかを「消灯」とした場合、LED は消灯状態となる。
	Scr_Finish	スクリプト終了時に、スクリ プトを登録したSWに対応した LEDを制御する。 また、実行ログとは別に結果 ログに文字列を出力する。	①文字列型:OK/NG判定文字列 ②文字列型:LED制御文字列 ③文字列型:ユーザーメッセージ 用	① OK() NG() (2) LGT_OFF() GRN_FLS() ORG_ON() ORG_FLS() 罷D_FLS() ③任意の文字列。最大1000文字。	①整数型:CT_result ctdef.pyのCT_resultを参 照	・本APIをコールすると実行ログとは別に 「script_log_XXX、result」と言うファイルが結 果ログとして出力される。 ①と③を利用してスクリプトの実行結果を判断に 利用できる。 ・コールせずスクリプトを終了させたり、スクリ プトやシステムの問題で途中で実行が停止した場 合は結果ログは出力されず、対応するLEDは赤点灯 となる。 ・スクリプト終了後、スクリプトを登録したSWに 対応するLEDが、②で指定した点灯/消灯状態とな る。ログを確認せずとも、本体LEDで実行結果判断 に利用できる。 ・③に1000文字以上の文字列を設定した場合の動 作は保証しない。
	ОК	結果OKを判定する文字列を返	 無し	-	①文字列型	※Windows上での美行は非りホード。 ・「OK」の文字列を返す。
	NC	す。Scr_Finishの引数①用。 結果NGも制定する立定列も返	Am. 1		結果OKを判定する文字列	
	NG	結果NGを判定する文字列を返 す。Scr_Finishの引数①用。	無し	-	①又子列型 結果NGを判定する文字列	・「NG」の文字列を返す。
	LGT_OFF	LEDをOFFに制御する文字列を 返す。Scr_Finishの引数② 用。	無し 	-	①文字列型 LEDをOFFに制御する文字列	・「LGT_0FF」の文字列を返す。
	IGRN_ON	LEDを緑点灯に制御する文字列 を返す。Scr_Finishの引数② 用。	無 し	-	①文字列型 LEDを緑点灯に制御する文字 列	・「GRN_ON」の文字列を返す。
	GRN_FLS	LEDを緑点滅に制御する文字列 を返す。Scr_Finishの引数② 用。	無し	-	①文字列型 LEDを緑点滅に制御する文字 列	・「GRN_FLS」の文字列を返す。
	ORG_ON	LEDを橙点灯に制御する文字列 を返す。Scr_Finishの引数② 用。	無し	-	①文字列型 LEDを橙点灯に制御する文字 列	・「ORG_ON」の文字列を返す。
	ORG_FLS	LEDを橙点滅に制御する文字列 を返す。Scr_Finishの引数② 田	無し	-	<ol> <li>①文字列型</li> <li>LEDを橙点滅に制御する文字</li> <li>제</li> </ol>	・「ORG_FLS」の文字列を返す。
	RED_FLS	n.。 LEDを赤点滅に制御する文字列 を返す。Scr_Finishの引数② 甲	無し し	-	77 ①文字列型 LEDを赤点滅に制御する文字 <sup>列</sup>	・「RED_FLS」の文字列を返す。
Control	Scr_ConvertSignalListFromFr ame	Frameに割り当てられている Signalのリストを取得する	①WORD FrameId [I] Frame ID ②DWORD FrameSize [I] 1Frame データのサイズ(バイト) ③BYTE* FrameData [I] 1Frame のデータ ④DWORD SignalSize [I] SignalDataの格納サイズ(バイト) ⑤cfg_signal_data_t_rmt* SignalData [0] Signalデータ	<ul> <li>①0~4999 内部管理IDのため</li> <li>Scr_ConvertXID2FrameID()で取得した値を使うこと。</li> <li>②0~255</li> <li>③20の数分セットすること</li> <li>④SignalDataサイズは、 「sizeof(cfg_signal_data_max_t_rm t)」を固定でセットすること。</li> <li>⑤cfg_signal_data_t_rmt構造体のメモリを確保する。</li> </ul>	小整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>SignalDataの領域は、 cfg_signal_data_max_t_rmt構造体サイズ分のメモ リを確保すること。</li> <li>指定されたフレームのデータを登録されている シグナルデータに変換して返す。</li> <li>シグナルデータは、cfg_signal_data_t_rmt構造 体の情報が取得できる。</li> <li>戻り値のSignalData件数分、SignalDataに情報 が格納される。</li> <li>構造体は、[cfg_signal_data_t_rmt構造体]定義を 参照。</li> </ul>
	Scr_ConvertXID2FrameID	BusIdとXIDからFrameIDを取得 する	<ol> <li>①整数型 BusId [1] バスID</li> <li>②整数型 SidOnly [1] 拡張ID使用有無</li> <li>③整数型 Sid [1] 標準フォーマットのベースID 11ビット</li> <li>④整数型 Eid [1] 拡張フォーマットの拡張ID 18ビット</li> <li>⑤リスト型 FrameId [0] フレームID(フレームを一意に識別するID番号) リストの先頭に整数型の値が格納される。</li> </ol>	① SCR_BUS_ID_CAN0 SCR_BUS_ID_CAN1 SCR_BUS_ID_CAN2 SCR_BUS_ID_CAN3 SCR_BUS_ID_LN ② 1: ベースIDのみ 0: ベースID+拡張ID ③ CAN: 0x00~0x7FF LIN: 0x00~0x3FF ④CAN: 0x00~0x3FFF ⑤0~4999	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたバス情報に登録されたSid(標準 フォーマットのベースID 11ビット)とEid(拡張 フォーマットの拡張ID 18ビット)から、</li> <li>FrameId(フレームを一意に識別するID番号)を取得 する。</li> <li>・標準フォーマットのベースID 11ビットのみを指 定する場合は、SidOnlyをTRUEで指定すること。</li> <li>・標準フォーマットのベースID 10と拡張IDの29ビットを指定する場合は、SidOnlyをFALSEで指定する こと。</li> <li>・本I/Fで取得したFrameIdを、SendFrameData、 GetFrameDataなどの引数に使用する。</li> </ul>
	Scr_ConvertXID2SignalID	BusIdとXIDとSignalSymbolか らSignalIDを取得する	<ol> <li>①整数型 BusId [I] バスID</li> <li>②整数型 SidOnly [I] 拡張ID使用有無(TRUE:ベースID+拡張ID)</li> <li>③整数型 Sid [I] 標準フォーマットのベースID 11ビット</li> <li>④整数型 Eid [I] 標準フォーマットの拡張ID 18ビット</li> <li>⑤文字列 SignalSymbol [I] シグ ナル名</li> <li>⑤リスト型 SignalId [0] シグナルID(シグナルを一意に識別する ID番号)</li> <li>リストの先頭に整数型の値が格納される。</li> </ol>	① SCR_BUS_ID_CAN0 SCR_BUS_ID_CAN1 SCR_BUS_ID_CAN3 SCR_BUS_ID_CAN3 SCR_BUS_ID_LIN ② 1:ペースIDのみ 0:ペースID+拡張ID ③ CAN: 0x00~0x3FF LIN: 0x00~0x3FFF ⑤1~63文字 ⑥0~29999	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>指定されたバス情報に登録されたSid(標準 フォーマットのベースID 11ビット)ととid(拡張 フォーマットの拡張ID 18ビット)ととうグナル名称 から、SignalId(シグナルを一意に識別するID番 号)を取得する。</li> <li>標準フォーマットのベースID 11ビットのみを指 定する場合は、SidOnlyをTRUEで指定すること。</li> <li>標準フォーマットのベースIDと拡張IDの29ビットを指定する場合は、SidOnlyをTALSEで指定する こと。</li> <li>本1/Fで取得したSignalIdを、SendSignalData、 GetSignalDataなどの引数に使用する。</li> </ul>

1	Sar SatllartSatting		①教教刊 anab la [1] IMPT 古林/毎		①教教刊,CT rooult	・ や ウ キャ も た は た IIADT の 汎 ウ に ら い ナ て
		j₀	3 ②整数型 baudrate [I] ボーレート ③整数型 datalength [I] データ 長 ④整数型 parity [I] パリティ ⑤整数型 stopbit [I] ストップ ビット	(1)         (2)         (3)         (2)         (3)         (2)         (3)         (2)         (3)         (3)         (4)         (3)         (3)         (4)         (3)         (4)         (5)         (6)         (7)         (8)         (9)         (9)         (9)         (9)         (9)         (	(J. 単成上: O rodult ctdef. pyのCT_resultを参 照	
	Scr_GetUartSetting	本体のUARTの設定を取得す る。	無 し	UART_SETTING_STPB2	<ol> <li></li></ol>	・現在のUARTの設定内容を返信する。
	Scr_GetFrameSize	指定したフレームIDのデータ サイズ (パイト) を取得する。	<ol> <li>①整数型 FrameId [I] フレーム ID</li> <li>②リスト型 size [0] フレームの データサイズ(バイト) リストの先頭に整数型の値が格 納される。</li> </ol>	①0~4999 内部管理IDのため Scr_ConvertXID2FrameID()で取得し た値を使うこと。 ②0~255 ③②の数分セットすること	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定したフレームIDのデータサイズ(バイト)を 取得する。
	Scr_GetSigna Size	指定したシグナル1Dのデータ サイズ(バイト)を取得する。	<ol> <li>①整数型 SignalId [I] シグナル ID</li> <li>②リスト型 size [0] シグナルの データサイズ(バイト)</li> <li>リストの先頭に整数型の値が格 納される。</li> </ol>	①0~29999 内部管理IDのため Sor_ConvertXID2SignalID()で取得し た値を使うこと。 ②0~255	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定したシグナルIDのデータサイズ(バイト)を 取得する。
DataSend	Scr_SendFrameData	指定Frameのデータを送信する (周期送信の値も更新される)	10 整数型 FrameId [I] フレーム ID ②整数型 size [I] 送信データサ イズ (バイト) ③bytearray data [I] 送信デー タ	①0~4999 内部管理IDのため Sor_ConvertXID2FrameID()で取得し た値を使うこと。 ②0~255 ③②の数分セットすること	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定されたFrameIDでデータを送信する。周期送 信の値に反映される。
	Scr_SendSignalData	指定Signalのデータを送信す る(周期送信の値も更新され る)	①整数型 SignalId []] シグナル ID ②整数型 size [1] 送信データサ イズ (バイト) ③bytearry data [I] 送信データ	<ul> <li>①0~29999 内部管理IDのため Sor_ConvertXID2SignalID()で取得した値を使うこと。</li> <li>②20~255</li> <li>③2の数分セットすること</li> </ul>	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたSignalIDでデータを送信する。周期</li> <li>送信の値に反映される。</li> <li>・送信データへのデータ格納について</li> <li>下詰めでStart位置に上から送信データに格納される。</li> <li>(例)Width:9の場合</li> <li>上 data[0] bit7~bit1:未使用</li> <li>data[0] bit0→sig data0(Start位置)</li> <li>data[1] bit0→sig data1</li> <li>data[1] bit6→sig data2</li> <li>data[1] bit6→sig data3</li> <li>data[1] bit3→sig data3</li> <li>data[1] bit3→sig data5</li> <li>data[1] bit3→sig data5</li> <li>data[1] bit3→sig data6</li> <li>data[1] bit1→sig data7</li> <li>下 data[1] bit0→sig data8</li> </ul>
	Scr_SendDaData	指定ボートのDAの値を変更す る	①整数型 PortNum [I] DAポート 番号 ②整数型 value [I] 値	① SCR_DA_CHANNEL_0 SCR_DA_CHANNEL_1 ②値 = ( 0.0000~5.0000(V) ) * 10000	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定されたポートのDAの値を変更する。
	Scr_SendGpoData	指定ポートのGPOの値を変更す る	①整数型 PortNum [I] GPOポート 番号 ②整数型 value [I] 値	T SCR_GPI0_CHANNEL_0 SCR_GPI0_CHANNEL_1 SCR_GPI0_CHANNEL_2 SCR_GPI0_CHANNEL_3 (2) SCR_GPI0_BIT_VALUE_LOW SCR_GPI0_BIT_VALUE_HIGH	①整数型:CT_result ctdef.pyのCT_resultを参 照	・指定されたポートのGPOの値を変更する。 ・値は、HI(1)、LO(0)を設定すること。
	Scr_SendAnyCANFrame	任意のCAN Frameデータを送信 する	<ul> <li>①整数型 BusId [I] バスID</li> <li>②整数型 SidOnly [I] 拡張ID使用有無</li> <li>③整数型 Sid [I] 標準フォーマットのベースID 11ビット</li> <li>④整数型 Eid [I] 拡張フォーマットの拡張ID 18ビット</li> <li>⑤整数型 isCANFD [I] CAN-FDフラグ</li> <li>⑤整数型 BitRateSwitch [I] CAN</li> <li>FDデータフェーズ高速化</li> <li>⑦整数型 DataLength [I] 送信 データサイズ</li> <li>⑧bytearray Data [I] 送信データ</li> </ul>	30K-BHT2-KLUE_INIGH ① SCR_BUS_ID_CAN0 SCR_BUS_ID_CAN1 SCR_BUS_ID_CAN2 SCR_BUS_ID_CAN3 ② 1: ペースIDのみ 0: ペースID+拡張ID ③ CAN: 0x00~0x3F ④CAN:0x00~0x3F 「 1: CAN FDフレーム 0: CANフレーム ⑥1: 有効、0: 無効(⑤がCAN FDフ レームの場合のみ利用可能) ⑦0~255 ⑧⑦の数分セットすること	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>指定されたバスにFrameデータを送信する。</li> <li>単発送信とし周期送信のFrameであっても周期送 信データには反映されない。</li> <li>(周期送信データを更新する場合は、 Sor_SendFrameData/Sor_SendSignalDataを使用する)</li> </ul>
	Scr_SendCyclicData2	送信データをFrameサイズに分 割し、指定周期で送信を行う	<ul> <li>①整数型 number CycleData番号</li> <li>②整数型 enable CycleData送信</li> <li>有効/無効</li> <li>③整数型 Busld []] バスID</li> <li>⑤整数型 SidOnly [I] が張ID使</li> <li>用有無(TRUE: ベースIDのみ、</li> <li>FALSE: ベースID+拡張ID)</li> <li>⑥整数型 Sid [I] 標準フォーマットのベースID 11ビット</li> <li>⑦整数型 Sid [I] 標準フォーマットのベースID 18ビット</li> <li>⑧整数型 Sid Sid [I] ホ張フォーマットの低張ID 18ビット</li> <li>⑧整数型 BitRateSwitch [I] 転</li> <li>送速度高速化(1: 有効、0: 無効)</li> <li>⑩整数型 DataLength [I] 送信</li> <li>データサイズ(バイト)</li> <li>⑪bytearray Data [I] 送信データ</li> </ul>	①0~14 ② TRUE: CycleData送信を行う。 FALSE: CycleData送信を停止する。 ③ 0~10000 (ms) ④ RMT_BUS_ID_CAN0 RMT_BUS_ID_CAN1 RMT_BUS_ID_CAN2 RMT_BUS_ID_CAN3 ⑤ TRUE: ペースIDのみ FALSE: ペースID+拡張ID ⑥0x00~0x3FFF ⑧ TRUE: cAN-FD FALSE: CAN ⑤ TRUE: 有効 FALSE: 無効 引数®か <sup>c</sup> CAN-FDの場合のみ有効 ⑩ 0~1024Byte ① 「⑪」で指定したデータ領域を設定	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたCycleData番号毎に、指定されたパス、CAN 1D等の設定及び、その周期送信の有効/無効を制御する。</li> <li>・引数の情報はCANTOOL本体に保存され、次回のCANTOOL本体起動時も設定に従った動作を行う。</li> <li>・設定したデータはFrameサイズに分割され、設定した周期で順番に送信される。</li> <li>・データの最終まで送信すると、再びデータの先頭から送信を行う。</li> <li>・summary_t_rmt構造体の「USB通信状態(本体-PC間)」が「USB_STATUS_DISCONNECT」の場合は利用不可。</li> </ul>

1 [	Scr_SendMultiPacketData2	MultiPacketデータを送信する	①整数型 fs [I] FS値	①FS值	①整数型:CT_result	・本APIで送受信ポートの情報を設定する。設定し
			<ul> <li>②整数型 bs [1] BS値</li> <li>③整数型 STmin [1] 最低送信間 隔</li> <li>④整数型 timeout [1] タイムア ウト時間</li> <li>⑤整数型 busid [1] バス1D</li> <li>⑥整数型 SidOnly [1] 拡張ID使 用有無(TRUE:ベースIDのみ、 FALSE:ベースID+拡張ID)</li> <li>⑦整数型 Sid [1] 標準フォー マットのベースID 11ビット</li> <li>⑧整数型 isid [1] 拡張フォー マットのベースID 11ビット</li> <li>⑨整数型 isidANFD [1] CAN-FDフ ラグ(1: CAN-FD、0: CAN)</li> <li>⑩整数型 RecvBusid [1] バスID</li> <li>⑩整数型 RecvBusid [1] バスID</li> <li>⑩整数型 RecvSidOnly [1] 拡張 D使用有無(TRUE:ベースID-拡張ID)</li> <li>⑩整数型 RecvSidOnly [1] 拡張 フォーマットのベースID 11ビット</li> <li>⑩整数型 RecvSidOnly [1] 拡張 D体用有無(TRUE:ベースID-拡張ID)</li> <li>⑩整数型 RecvSid [1] 標準 フォーマットのベースID 11ビット</li> <li>⑩整数型 RecvSid [1] 拡張 フォーマットのベースID 11ビット</li> <li>⑩整数型 RecvSid [1] 拡張</li> <li>フォーマットのボ法[ID 18ビット</li> <li>⑩整数型 RecvSithateSwitch [1] 転送速度高速化(1:有効、0:無効)</li> <li>⑪を数型 DataLength [1] 送信 データサイズ(バイト)</li> <li>⑩を数型 AddressFormatType [1]</li> <li>⑩を数型 N_TA</li> </ul>	NULTI_PACKET_FS_CONTINUOUS_RECEIV E:連続受信可、 MULTI_PACKET_FS_0VERFLOW:Overflow (2)0~FF MULTI_PACKET_BS_ALL_RECEIVE_POSSI BLE: 全て受信可 (3)0~FF(ms) (4)0~10000(ms) (5)SCR_BUS_ID_CAN0 SCR_BUS_ID_CAN1 SCR_BUS_ID_CAN2 SCR_BUS_ID_CAN3 (6)TRUE: ペースIDのみ FALSE: ペースIDのみ GRUUE: ADD / FALSE: 無効 GI (1) SCR_BUS_ID_CAN3 (1) SCR_BUS_ID_CAN3 (2)TRUE: ペースIDのみ FALSE: ペースIDのみ FALSE: ペースIDのみ FALSE: ペースIDのみ FALSE: ペースIDのみ FALSE: ペースIDのみ GI (3) T(0) - CAN3 (2) TRUE: ADD / FALSE: CAN (5) TRUE: ADD / FALSE: MD GI TUL: ADD / FALSE: MD GI TULI - PACKET_ADD MESS_FORMAT_TYPE_ EXTENDED : extended C0 0x00~0xFF	〜 ctdef.pyのGT_resultを参照	なければScr_GetMultiPacketStatus() での情報取 得はできない。 ・ 4095Byteより大きい送受信データサイズでの動 作は保証しない。 ・ マルチパケットのシーケンス完了前に本APIを コールした場合の動作は保証しない。 ・ 引数①の値ごとの挙動は以下となる。 0:正常ケースであり、APIの動作を継続する。 1:待機状態であることをFCICて送信する。以降は通 常のAPIの動作を継続する。 2:相受信に失敗したことをFCICて送信する。本 来、TP上はマルチパケット送受信処理は終了する が、APIとしてはデータを受信できる状態でタイム アウトまで待つ。 ・ 引数①203は対象へFCパケットで通知する値で あり、FC情報は相手から受け取った情報に従う。 ・ 複数コネクションには対応せず、本APIコール毎 に情報は上書きされる。 ・ を彼して他の情報に一致するデータを待ちIP受信 する。 ●想定利用方法 ・ Scr_GetMultiPacketData2()で送受信情報を設 定し、データを送信する。 ・ Scr_GetMultiPacketStatus()で100ms程度の間隔 でポーリングして受信したデータを取得する。 ・ Scr_GetMultiPacketStatus()で送受信の失敗や ターゲットの応答タイムアウトを取得する。
	Scr_ResponseMultiPacketData	MultiPacketへの応答内容設定	<ol> <li>①整数型 fs [1] FS値</li> <li>②整数型 STmin [1] 最低送信間 隔</li> <li>④整数型 STmin [1] 最低送信間 隔</li> <li>④整数型 timeout [1] タイムア ウト時間</li> <li>⑤整数型 SidOnly [1] 拡張ID使 用有無 (IRUE: ベース1Dのみ、 FALSE: ベース1D+拡張ID)</li> <li>⑦整数型 SidOnly [1] 拡張Up+ イース1D+拡張ID)</li> <li>⑦整数型 Sid [1] 標準フォー マットのベースID 11ビット</li> <li>⑧整数型 Fid [1] 拡張フォー マットのが売スID 11ビット</li> <li>⑨整数型 Sid [1] 拡張フォー マットのが売スID 11ビット</li> <li>⑨整数型 Sid [1] 拡張フォー マットのが一スID 11ビット</li> <li>⑨整数型 RecvSidOnly [1] 拡張 Di使用有無 (TRUE: ベースID 11ビット</li> <li>③整数型 RecvSidOnly [1] 拡張 Di使用有無 (TRUE: ベースID み、FALSE: ベースID+拡張ID)</li> <li>③整数型 RecvSid [1] 標準 フォーマットのベースID 11ビット</li> <li>⑤整数型 RecvSid [1] 拡張 フォーマットのが伝張ID 18ビット</li> <li>⑤整数型 RecvSid [1] 拡張 フォーマットの気法(II) 18 (20)</li> <li>「動数型 RecvSid [1] 拡張</li> <li>フォーマットの太張ID 18ビット</li> <li>⑤整数型 RecvSid [1] 拡張</li> <li>フォーマットの太張ID 18ビット</li> <li>⑤整数型 RecvSid [1] 拡張</li> <li>フォーマットの太張ID 18ビット</li> <li>⑤整数型 RecvSid [1] 拡張</li> <li>アーターケズ(バイト)</li> <li>⑧bytearray Data [1] 送信 デー タサイズ(バイト)</li> <li>⑧を数型 AddressFormatType [1]</li> <li>⑩ 整数型 N_TA [1]</li> </ol>	<ul> <li>①FS値 MULTI_PACKET_FS_CONTINUOUS_RECEIV E:連続受信可、 MULTI_PACKET_FS_WAIT:待機、 MULTI_PACKET_FS_OVERFLOW:Overflow ②0~FF</li> <li>MULTI_PACKET_BS_ALL_RECEIVE_POSSI BLE:全て受信可</li> <li>③0~(10000 (ms)</li> <li>⑤SCR_BUS_ID_CAN0</li> <li>SCR_BUS_ID_CAN1</li> <li>SCR_BUS_ID_CAN2</li> <li>SCR_BUS_ID_CAN2</li> <li>SCR_BUS_ID_CAN3</li> <li>⑥TRUE:ベースIDのみ</li> <li>FALSE:ベースID+拡張ID</li> <li>⑦0×00~0x7FF</li> <li>⑧TRUE:CAN-FD/FALSE:CAN</li> <li>⑩TRUE:AD / FALSE: KD</li> <li>⑪TRUE:AD / FALSE: KD</li> <li>⑪TRUE:AD / FALSE: KD</li> <li>⑪TRUE:AD / FALSE:KD</li> <li>⑪TRUE:AD / FALSE:KD</li> <li>⑪TRUE:AD / FALSE:KD</li> <li>⑪TRUE:AD / FALSE:KD</li> <li>⑪SCR_BUS_ID_CAN3</li> <li>⑫TRUE:AD / FALSE:KD</li> <li>⑪SCR_BUS_ID_CAN3</li> <li>⑫TRUE:AD / FALSE:KD</li> <li>⑫AN0~0x3FFFF</li> <li>⑬AN00~0x3FFFF</li> <li>⑭AN200~0x3FFFF</li> <li>⑭AN200~0x3FFFF</li> <li>⑭TRUE:AD / FALSE:KD</li> <li>⑭TRUE:AD / FALSE:KD</li> <li>⑭TAUSBBYte.0M場合のみ有効</li> <li>⑪T-4095Byte.00場合のみ有効</li> <li>⑪T-4095Byte.00場合のみ有効</li> <li>⑪T-4095Byte.00場合のみ有効</li> <li>⑫AN0~0x3FFFF</li> <li>⑭TRUE:AD / FALSE:FM</li> <li>⑭TRUE:AD / FALSE:SE</li> <li>𝔅</li> <li>𝔅<th>①整数型:CT_result ctdef.pyのCT_resultを参 照</th><th><ul> <li>・本APIでマルチパケットへの応答内容を設定する。</li> <li>・4095Byteより大きい送受信データサイズでの動作は保証しない。</li> <li>・マルチパケットのシーケンス完了前に本APIをコールした場合の動作は保証しない。</li> <li>・マルチパケットのシーケンス完了前に本APIをコールした場合の動作は保証しない。</li> <li>・引数①の値ごとの挙動は以下となる。</li> <li>の正常ケースであり、APIの動作を継続する。</li> <li>1:待機状態であることをFCICて送信する。以降は通常のAPIの動作を継続する。</li> <li>2:相受信に失敗したことをFCICて送信する。本来、TP上はマルチパケット送受信処理は終了するが、APIとしてはデータを受信できる状態でタイムアウトまで待つ。</li> <li>・引数①2(3は対象へFCパケットで通知する値であり、FCF報は相手から受け取った情報に従う。</li> <li>・引数①2(3は対象へFCパケットで通知する値であり、FCF報は相手から受け取った情報に従う。</li> <li>・割数①2(3は対象へFCパケットで通知する値であり、FC情報は相手から受け取った情報に従う。</li> <li>・割数①2(3は対象へFCパケットで通知する値であり、FC情報は上書きされる。</li> <li>・1000情報に一致するデータをTP受信し、その後(5)~(1000情報で①~2000データをFP送信する。</li> <li>●想定利用方法</li> <li>・Scr_ResponseMultiPacketData()で送受信情報を設定する。</li> <li>・Scr_GetResponseMultiPacketStatus()で100ms程度の間隔でボーリングして受信したデータを取得する。</li> <li>・Scr_GetResponseMultiPacketStatus()で送受信の失敗やターゲットの応答タイムアウトを取得する。</li> </ul></th></li></ul>	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・本APIでマルチパケットへの応答内容を設定する。</li> <li>・4095Byteより大きい送受信データサイズでの動作は保証しない。</li> <li>・マルチパケットのシーケンス完了前に本APIをコールした場合の動作は保証しない。</li> <li>・マルチパケットのシーケンス完了前に本APIをコールした場合の動作は保証しない。</li> <li>・引数①の値ごとの挙動は以下となる。</li> <li>の正常ケースであり、APIの動作を継続する。</li> <li>1:待機状態であることをFCICて送信する。以降は通常のAPIの動作を継続する。</li> <li>2:相受信に失敗したことをFCICて送信する。本来、TP上はマルチパケット送受信処理は終了するが、APIとしてはデータを受信できる状態でタイムアウトまで待つ。</li> <li>・引数①2(3は対象へFCパケットで通知する値であり、FCF報は相手から受け取った情報に従う。</li> <li>・引数①2(3は対象へFCパケットで通知する値であり、FCF報は相手から受け取った情報に従う。</li> <li>・割数①2(3は対象へFCパケットで通知する値であり、FC情報は相手から受け取った情報に従う。</li> <li>・割数①2(3は対象へFCパケットで通知する値であり、FC情報は上書きされる。</li> <li>・1000情報に一致するデータをTP受信し、その後(5)~(1000情報で①~2000データをFP送信する。</li> <li>●想定利用方法</li> <li>・Scr_ResponseMultiPacketData()で送受信情報を設定する。</li> <li>・Scr_GetResponseMultiPacketStatus()で100ms程度の間隔でボーリングして受信したデータを取得する。</li> <li>・Scr_GetResponseMultiPacketStatus()で送受信の失敗やターゲットの応答タイムアウトを取得する。</li> </ul>
	Scr_SendUartData	UARTへのデータを送信する	①DWORD size [I] 送信データサ	①最大データ長:UARTDATA_SIZE_MAX	①整数型:CT_result	・送信データをUARTに送信する。
DataRecieve	Scr_GetFrameData	指定Frameの最新データを取得 する	<ul> <li>(2)BYTE+ data [1] 送信データ</li> <li>(2)BYTE+ data [1] フレーム</li> <li>(1) フレーム</li> <li>(1) フレーム</li> <li>(2) 整数型 size [1] 受信フレーム</li> <li>データサイズ (パイト)</li> <li>(3) bytearray data [0] 受信フレームデータ</li> <li>(4) リスト型 Time [0] 最終更新時 タイムスタンプ((単位:100ns</li> <li>(1601-01-01:00:00を基準とした経過時間))</li> <li>リストの先頭に整数型の値が格 約される。</li> </ul>	①の~4999 内部管理IDのため Scr_ConvertXID2FrameID()で取得し た値を使うこと。 ②の~255 ④0~0xFFFFFFFFFFFFFFF	照 ①整数型:CT_result ctdef.pyのCT_resultを参 照	UARTDAT_SIZE_MAX(単位:Bytes) ・指定されたFrameIDの最新データを取得する。 ・一度も受信していない場合は、戻り値が RET_NOTFOUND、受信フレームデータはNULL、最終 更新時タイムスタンプは0となる。
	Scr_GetSignalData	指定Signalの最新データを取 得する	<ol> <li>①整数型 SignalId [I] シグナル ID</li> <li>②整数型 size [I] 受信シグナル データサイズ(バイト)</li> <li>③bytearray data [0] 受信シグ ナルデータ</li> <li>④リスト型 Time [0] 最終更新時 タイムスタンプ(単位:100ns 1601-01-01:00:00を基準とし た経過時間))</li> <li>リストの先頭に整数型の値が格 納される。</li> </ol>	①0~29999 内部管理IDのため Scr_ConvertXID2SignalID()で取得し た値を使うこと。 ②0~255 ④0~0xFFFFFFFFFFFFFFFF	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたSignal IDの最新データを取得する。</li> <li>・一度も受信していない場合は、戻り値が</li> <li>RET_NOTFOUND、受信フレームデータはNULL、最終 更新時タイムスタンプは0となる。</li> <li>・data格納について</li> <li>下詰めで上からStart位置よりdataに格納される (例)Widh: 9の場合</li> <li>上 data[0] bit0~sig data0(Start位置) data[1] bit7~sig data1 data[1] bit16~sig data2 data[1] bit5~sig data3 data[1] bit5~sig data3 data[1] bit4~sig data5 data[1] bit2~sig data5 data[1] bit1←sig data7</li> <li>下 data[1] bit0~sig data8</li> </ul>
	Scr_GetAdData	指定ポートのADの値を取得す る	<ol> <li>①整数型 PortNum [1] ADポート 番号</li> <li>②リスト型 value [0] 値 リストの先頭に整数型の値が格 納される。</li> <li>③リスト型 Time [0] 最終更新時 タイムスタンプ((単位:100ns 1601-01-01 00:00:00を基準とし た経過時間)) リストの先頭に整数型の値が格 納される。</li> </ol>	① SCR_AD_CHANNEL_0 SCR_AD_CHANNEL_1 ②値 = (0.0000~50.0000(V)) * ③0~0xFFFFFFFFFFFFFFF	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたADポートの最新データを取得する。</li> <li>・値は、AD電圧値0V~50Vを1万倍した値が取得できる。</li> <li>・一度も受信していない場合は、戻り値が RET_NOTFOUND、値はNULL、最終更新時タイムスタンプは0となる。</li> </ul>

Scr	′_GetGpiData	指定ポートのGPIの値を取得す る	<ol> <li>①整数型 PortNum [1] GP1ポート 番号</li> <li>②リスト型 value [0] 値 リストの先頭に整数型の値が格 納される。</li> <li>③リスト型 Time [0] 最終更新時 タイムスタンブ((単位:100ns 1601-01-01 00:00:00を基準とし た経過時間)) リストの先頭に整数型の値が格 納される。</li> </ol>	① SCR_GP10_CHANNEL_0 SCR_GP10_CHANNEL_1 SCR_GP10_CHANNEL_2 SCR_GP10_CHANNEL_3 (②HI (1) 、L0 (0) ③0~0xFFFFFFFFFFFFFFFF	①整数型:CT_result ctdef.pyのCT_resultを参 照	<ul> <li>・指定されたGP1ポートの最新データを取得する。</li> <li>・値は、HI(1)、L0(0)が取得できる。</li> <li>・一度も受信していない場合は、戻り値が RET_NOTFOUND、値はNULL、最終更新時タイムスタンプは0となる。</li> </ul>
Sor	r_GetMultiPacketStatus	MultiPacket状態を取得する	①bytearray Data [0] 受信デー タ	①4095Byteのメモリを確保すること。	<ol> <li>①整数型:CT_result ctdef.pyのCT_resultを参照</li> <li>②整数型データ有無状態 「MULTI_PACKET_STATE_~」 の定義を参照</li> <li>データがある</li> <li>データがある</li> <li>データがある</li> <li>ジークタがある</li> <li>ジークシンス</li> <li>ツークタがない</li> <li>ジークタがない</li> <li>ジークタがない</li> <li>ジークタがない</li> <li>ジークがない</li> <li>ジークがない</li> </ol>	<ul> <li>マルチパケット送受信の状態を取得することができる。</li> <li>先にSor_SendMultiPacketData2()で送受信ポートの設定を行っておく必要がある。</li> <li>送受信データサイズが4095Byte以上となる環境での動作は保証しない。</li> <li>状態はバッファのみ保持する。Get前に新しい状態に変化すると情報を上書きする。</li> <li>本和PIをコールすると情報を破棄し、取得内容②が「データがある」と「FC情報」の場合のみ取得内容③と引数①は有効。「30」の時は受信したFCのデータが格納される。</li> <li>18yte目:FS値、2Byte目:BS値、3Byte目:STmin値</li> <li>FCデータのFS値ごとの挙動は以下となる。</li> <li>ご市常な応答であり、APIの動作を継続する。</li> <li>1:相手の処理待ち状態であり、処理待ちが解除され次第APIの動作を継続する。</li> <li>1:相手側で受信に失敗した状態であり、TP上はマルチパケット送受信処理は終了する。ただしAPIとしてはデータを受信できる状態でタイムアウトまで待つ。</li> </ul>
Sor tat	r_GetResponseMultiPacketS tus	MultiPacketの応答結果を取得 する	①bytearray Data [0] 受信デー タ	①4095Byteのメモリを確保すること。	<ol> <li>①整数型:CT_result ctdef.pyのCT_resultを参照</li> <li>②整数型データ有無状態 「MULTI_PACKET_STATE_~」 の定義を参照</li> <li>データがある</li> <li>データがある</li> <li>データがある</li> <li>ジ信エラー(タイムアウト)</li> <li>注信エラー(ク部エラー)</li> <li>注信エラー(タイムアウト)</li> <li>空信エラー(タイムアウト)</li> <li>ジャンス</li> <li>単次的</li> <li>第二、第二、第二、第二、第二、第二、第二、第二、第二、第二、第二、第二、第二、第</li></ol>	・マルチパケットの応答結果状態を取得すること ができる。 ・先にScr_ResponseMultiPacketData()で送受信 ボートの設定を行っておく必要がある。 ・受信データサイズが4095Byte以上となる環境で の動作は保証しない。 ・状態にない。ファのみ保持する。Get前に新しい 状態に変化すると情報を上書きする。 ・本APIをコールすると情報を破棄し、取得内容② が「1」となる。 ・取得内容②が「データがある」と「FC情報」の 場合のみ取得内容③と引数①は有効。 「30」の時は受信したFCのデータが格納され る。 1Byte目:FS値、2Byte目:BS値、3Byte目: STmin値 +FCデータのFS値ごとの挙動は以下となる。 0:正常な応答であり、APIの動作を継続する。 1:相手の処理待ち状態であり、処理待ちが解除 され次第APIの動作を継続する。 2:相手側で受信に失敗した状態であり、TP上は マルチパケット送受信処理は終了する。ただしAPI としてはデータを受信できる状態でタイムアウト まで待つ。