DENSO
RAC 通信仕様書

Version 1.1.12

February 9, 2012

【備考】

【改版履歴】

日付	版数	内容
2012-02-09	1.0.0	初版

目次

1. はじめに

本書は，RC8 に実装されている DENSO RAC のユーザーズガイドです.

RAC(Robot Action Command)とは，ロボットを動かすための統一的なロボット言語を目指して開発されたものです.

クライアントアプリケーションからソケット通信(TCP/IP)を用いてロボットに RAC メッセージを送信します. DENSO RAC では，ソケット通信を用いて RC8 に RAC メッセージを送信します. RC8 は送信されてきた RAC メッセージを解析し，メッセージの内容に応じた変数の取得，設定をおこないます.

2. RAC 仕様

RAC では, クライアントからサーバへ要求を出すための“要求文字列”と, サーバがクライアントからの要求に応答するための“応答文字列”の 2 種類が既定されています.

本章では, これらの文字列の仕様について解説をおこないます.

2.1. RAC 文字列

2.1.1. 共通仕様

RAC 文字列には, 要求文字列と応答文字列の 2 種類がありますが, それらの文字列の共通仕様を以下に示します.

- データはテキスト表記とします
- 1 コマンド(ターミネータ(CR)を含む)は 256byte 以内とします
- 行頭につく Space や Tab 文字(インデント等)は無視します

2.1.2. 要求文字列

RAC の要求文字列は以下のように記述します. ここで[]は省略可能なことを示しています.

<トップレベルコマンド>:[<部位名>]:[<部位番号>]:[<サブコマンド>]:[<パラメータ>]<ターミネータ>

<トップレベルコマンド>	:	DENSO RAC サーバでは, “PUT”, “GET”のみサポートしています. “PUT”コマンドでは変数の設定を行います. “GET”コマンドでは変数の取得を行います.
<部位名>	:	コントローラ名(“RC8”固定)
<部位番号>	:	変数インデックス番号
<サブコマンド>	:	変数タイプ 以下の変数タイプを指定することができます. “I”, “F”, “D”, “S”, “V”, “P”, “J”, “T”, “IO”
<パラメータ>	:	データ “PUT”コマンド指定時に変数に対して設定する値を指定します. “GET”コマンド指定時には使用しません. データの記述方式については, 2.2 に詳細を示します.
<ターミネータ>	:	ターミネータ CR(0x0D)固定

2.1.3. 応答文字列

RAC の応答文字列は以下のように記述します. ここで[]は省略可能なことを示しています.

<処理結果>[,<応答データ>]<ターミネータ>

<処理結果>	:	RAC コマンドの実行結果を HRESULT 型の整数値で示します.

<応答データ> : “GET”コマンドで指定した変数の値を取得します.
“PUT”コマンドのときは, 使用しません.
データの記述方式については, 2.2 に詳細を示します. .

<ターミネータ> : ターミネータ
CR(0x0D)固定

2.2. データの記述方式

要求文字列のパラメータ及び応答文字列の応答データは共通して ORiN2 の VARIANT 型を文字列で表現する方法に従っています. その書式は, 以下に示すようにデータ型とデータ列をカンマ区切りで表現します.

<データ型>, <データ列>

ここで, <データ型>には VARTYPE 型で表記される整数値を示します. 表 2-1 に使用できるデータ型とその値を示します.

表 2-1 使用可能なデータ型

データ型	値	説明
VT_EMPTY	0	空データ
VT_NULL	1	NULL 値
VT_ERROR	10	エラーコード
VT_UI1	17	バイト型
VT_I2	2	短整数
VT_UI2	18	符号なし短整数
VT_I4	3	長整数
VT_UI4	19	符号なし長整数
VT_R4	4	単精度浮動小数点
VT_R8	5	倍精度浮動小数点
VT_CY	6	通貨型
VT_DATE	7	日付型
VT_BOOL	11	ブール型
VT_BSTR	8	文字列型
VT_VARIANT	12	Variant 型 データに引数部と同じものが入ります. VT_ARRAY のときのみ使用が可能です.
VT_ARRAY	0x2000	配列 他のデータ型との論理和で指定します. データ列にはデータを文字列で表記します. 配列データの表記は

		“, ”(カンマ)で区切って表記します.

例 1) 2,100 型: VT_I2 値: 100
例 2) 8,Sample 型: VT_BSTR 値: Sample
例 3) 12,(8,Sample) 型: VT_VARIANT 値: “Sample”(文字列型)
例 4) 8194,100,200,300 型: VT_I2 | VTARRAY 値: 100, 200, 300
例 5) 8200,Sample,Test 型: VT_BSTR | VTARRAY 値: “Sample”, “Test”
例 6) 8,Sample,Test 型: VT_BSTR 値: “Sample,Test”
例 7) 8204,(8,Sample),(2,100) 型: VT_VARIANT | VTARRAY 値: “Sample”, 100

2.3. リターンコード

b-CAP ではリターンコードは以下のように割り当てられています.

表 2-2 リターンコードの割り当て

リターンコード	説明
0x00000000～0x8000FFFF 0x80010000～0x800101FF 0x80070000～0x8007FFFF	既定リターンコード, 予約領域
0x80040200～0x8004FFFF	ユーザ定義エラー

以下に示す「既定リターンコード」以外のリターンコードを作成するときは,「ユーザ定義エラー」の値の範囲内で任意のコードを割り当てることが可能です.

表 2-3 既定のリターンコード一覧

リターンコード	エラー	説明
0x00000000	S_OK	正常終了
0x80004001	E_NOTIMPL	未実装の機能です.
0x80004004	E_ABORT	関数が中断されました.
0x80004005	E_FAIL	関数が失敗しました.
0x8000FFFF	E_UNEXPECTED	致命的エラーが発生しました.
0x80010001	E_INVALIDRCVPACKET	ロボットコントローラが受信した n パケットが不正です. このエラーが発生した際には、ロボットコントローラは自動的に接続を切断します.

		ロボットコントローラへ送信したパケットを確認してください.
0x80010002	E_INVALIDSNDPACKET	送信パケットが不正です.
0x80010003	E_INVALIDARGTYPE	受信パケット内の引数型が不正です.
0x80010004	E_ROBOTISBUSY	ロボットが動作中に新たな動作命令を受信しました. 動作できません.
0x80010005	E_INVALIDCOMMAND	不正なコマンド文字列を受信しました. 実行できません.
0x80010011	E_PACKETSIZEOVER	受信パケットサイズが不正です. (>16Mbytes)
0x80010012	E_ARGSIZEOVER	受信パケットの引数サイズが不正です. (>16Mbytes)
0x80070005	E_ACCESSDENIED	アクセスできません.
0x80070006	E_HANDLE	ハンドルが不正です.
0x8007000E	E_OUTOFMEMORY	メモリが不足しています.
0x80070057	E_INVALIDARG	引数が不正です.

3. 通信手順

3.1. 通信シーケンス

RAC のシーケンスは，必ずクライアントからの要求パケットの送信によって始まります．サーバ側は，要求パケットの関数を実行し， 応答パケットをクライアントに返します．

1つのセッションでは，要求メッセージ送信後は応答メッセージの受信を待ち，常に同期をとる必要があります．もし，複数の要求メッセージを使用する場合には，複数セッションで行うことを推奨します．

サーバ側では要求パケットを受信してから回答を返信するまでの時間に関して特に規定がありません．このため，サーバ側の処理時間が長いときにクライアントのタイムアウト検出時間が短い場合，クライアント側でタイムアウトが発生することになりますので注意してください．

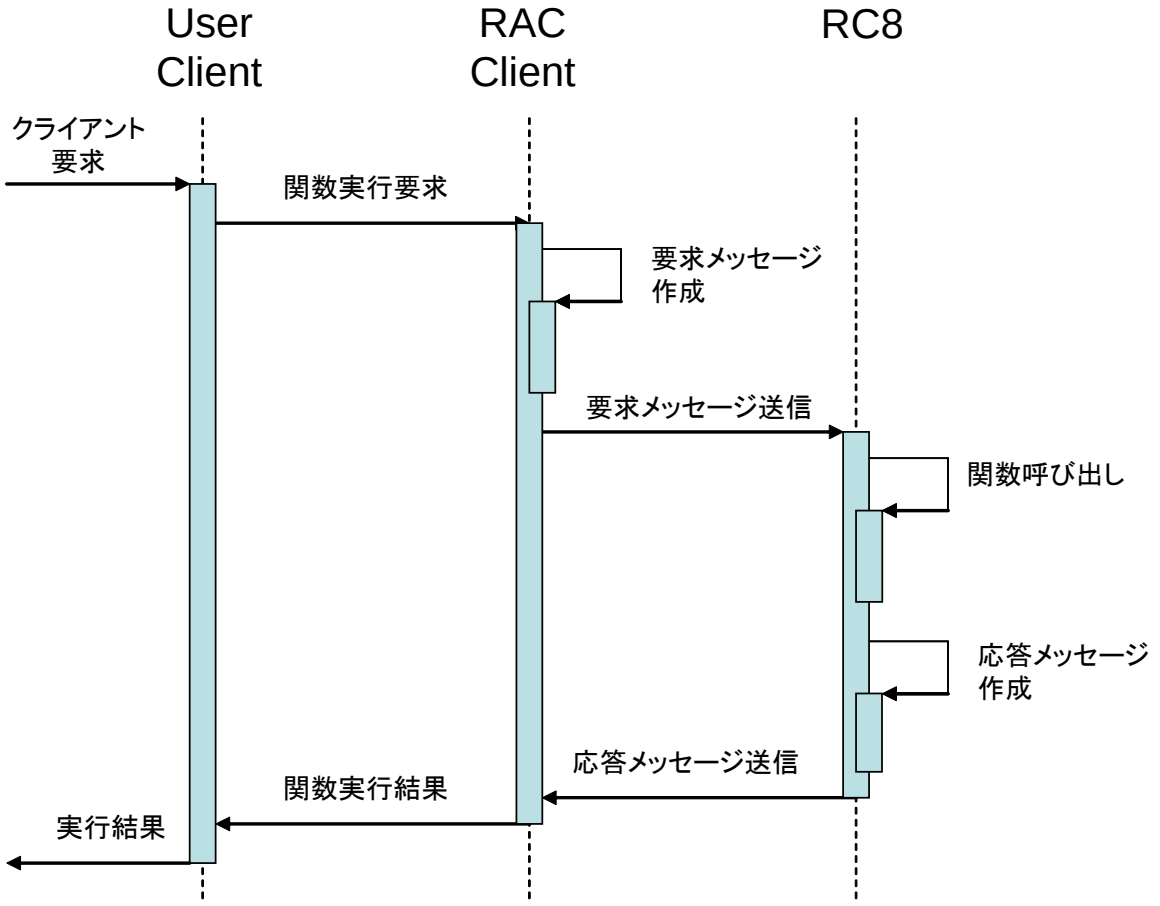

図 3-1 通信シーケンス

3.2. クライアントの通信手順

以下にクライアントの通信手順の概要を示します.

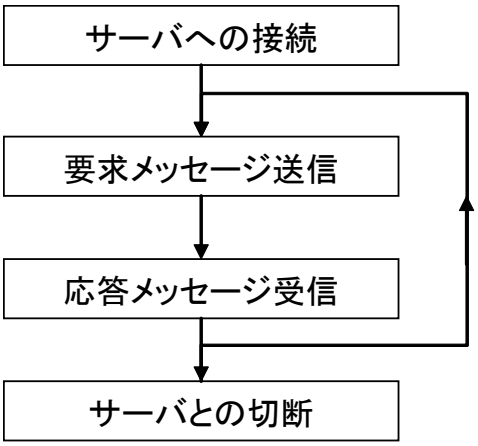

図 3-2 クライアントの通信手順

クライアントは最初にサーバへの接続をし, セッションを確立します. このとき, サーバの TCP の 5006 番ポートへ接続します.

接続後, 実行した関数の要求メッセージを送信し, サーバからの実行結果を待ちます.

クライアントは, サーバからの応答がないときにタイムアウト処理を行う必要があります. しかし, サーバの応答時間は処理内容によって異なるため, タイムアウトの設定時間には注意が必要です.

4. RAC コマンド

4.1. GET コマンド

・ 要求文字列

GET:RC8:<インデックス番号>:<変数タイプ>:<CR>

・応答文字列

<処理結果>,<応答データ型>,<応答データ><CR>

通信サンプル 1		
I 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:I:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	I
受信 コマンド	サーバ→クライアント: 0,3,123<CR>	
	引数	値
	処理結果	0（成功）
	応答データ型	VT_I4
	応答データ	123

通信サンプル 2		
F 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:F:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	F
受信 コマンド	サーバ→クライアント: 0,4,123.01<CR>	
	引数	値
	処理結果	0（成功）
	応答データ型	VT_R4

	応答データ	123.01

通信サンプル 3		
D 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:D:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	D
受信 コマンド	サーバ→クライアント: 0,5,123.01<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_R8
	応答データ	123.01

通信サンプル 4		
S 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:S:	
	パラメータ	値
	インデックス番号	10
	変数タイプ	S
受信 コマンド	サーバ→クライアント: 0,8,Test<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_BSTR
	応答データ	“Test”

通信サンプル 5		
V 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:V:<CR>	
	パラメータ	値

	インデックス番号	10
	変数タイプ	V
受信 コマンド	サーバ→クライアント: 0,8196,1,2,3<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_ARRAY | VT_R4
	応答データ	(1, 2, 3)

通信サンプル 6		
P 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:P:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	P
受信 コマンド	サーバ→クライアント: 0,8196,1,2,3,4,5,6,-1<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_ARRAY | VT_R4
	応答データ	(1, 2, 3, 4, 5, 6, -1)

通信サンプル 7		
J 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:J:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	J
受信 コマンド	サーバ→クライアント: 0,8196,1,2,3,4,5,6,7,8<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_ARRAY | VT_R4

	応答データ	(1, 2, 3, 4, 5, 6, 7, 8)

通信サンプル 8		
T 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:T:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	J
受信 コマンド	サーバ→クライアント: 0,8196,1,2,3,4,5,6,7,8,9,-1<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_ARRAY | VT_R4
	応答データ	(1, 2, 3, 4, 5, 6, 7, 8, 9, -1)

通信サンプル 9		
IO 型変数(インデックス番号:10)の値を取得します.		
送信 コマンド	クライアント→サーバ: GET:RC8:10:IO:<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	J
受信 コマンド	サーバ→クライアント: 0,11,0<CR>	
	引数	値
	処理結果	0 (成功)
	応答データ型	VT_BOOL
	応答データ	FALSE

4.2. PUT コマンド

・ 要求文字列

PUT:RC8:<インデックス番号>:<変数タイプ>:<送信データ型>,<送信データ><CR>

・応答文字列

<処理結果><CR>

通信サンプル 1		
I 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:I:3,123<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	I
	送信データ型	VT_I4
	送信データ	123
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0 (成功)

通信サンプル 2		
F 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:F:4,123.01<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	F
	送信データ型	VT_R4
	送信データ	123.01
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0 (成功)

通信サンプル 3		
D 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:D:5,123.01<CR>	
	パラメータ	値

	インデックス番号	10
	変数タイプ	D
	送信データ型	VT_R8
	送信データ	123.01
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）

通信サンプル 4		
S 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:S:8,Test<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	S
	送信データ型	VT_BSTR
	送信データ	“Test”
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）

通信サンプル 5		
V 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:V: 8196,1,2,3<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	V
	送信データ型	VT_ARRAY | VT_R4
	送信データ	(1, 2, 3)
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値

	処理結果	0（成功）

通信サンプル 6		
P 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:P:8196,1,2,3,4,5,6,-1<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	P
	送信データ型	VT_ARRAY | VT_R4
	送信データ	(1, 2, 3, 4, 5, 6, -1)
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）

通信サンプル 7		
J 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:J:8196,1,2,3,4,5,6,7,8<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	J
	送信データ型	VT_ARRAY | VT_R4
	送信データ	(1, 2, 3, 4, 5, 6, 7, 8)
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）

通信サンプル 8		
T 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:T:8196,1,2,3,4,5,6,7,8,9,-1<CR>	
	パラメータ	値

	インデックス番号	10
	変数タイプ	J
	送信データ型	VT_ARRAY | VT_R4
	送信データ	(1, 2, 3, 4, 5, 6, 7, 8, 9, -1)
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）

通信サンプル 9		
IO 型変数(インデックス番号:10)の値を設定します.		
送信 コマンド	クライアント→サーバ: PUT:RC8:10:IO:11,0<CR>	
	パラメータ	値
	インデックス番号	10
	変数タイプ	J
	送信データ型	VT_BOOL
	送信データ	FALSE
受信 コマンド	サーバ→クライアント: 0<CR>	
	引数	値
	処理結果	0（成功）