b-CAP スレーブモードの使い方

DENSO WAVE. 2009

注意

- このドキュメントはb-CAPに対する基礎的な知識を持ち，b-CAPを使ったことがある方を対象としています．
  - 初めての方は，“b-CAP ユーザーズガイド”を先にお読みください．
- スレーブモードを使用するには各種設定のためティーチングペンダントが必要です．

Contents

スレーブモードとは

- スレーブモードは短い時間間隔で位置・姿勢データを送信する事でロボットをコントロールする機能です.
- b-CAPには3つの関数が実装されています.
  - slvChangeMode
    - スレーブモードの設定を変更します.
  - slvGetMode
    - 現在のスレーブモードの設定を取得します.
  - slvMove
    - 指定した位置・姿勢にロボットを移動させます.
- バージョン3.0以降のRC7コントローラでは，TCPとUDP接続に対応しています.
  - UDP接続の場合，ユーザーはリトライ手順の実装が必要です.

スレーブモード関数(1)

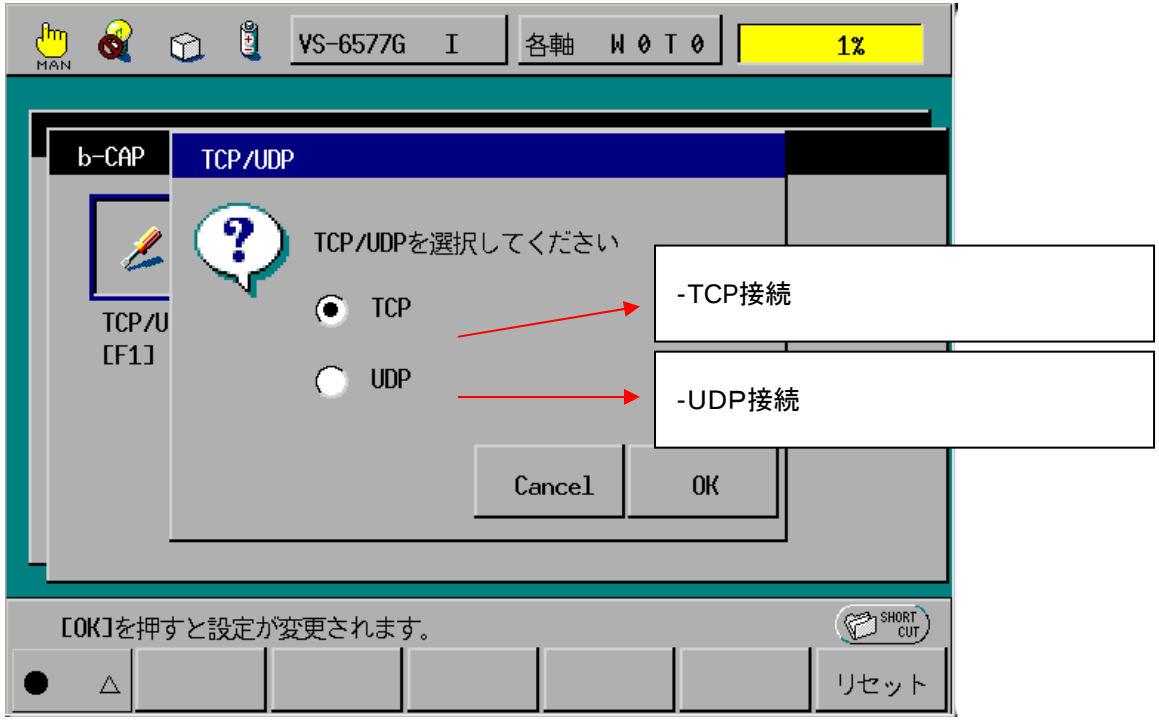
- slvChangeMode
  - この関数は位置・姿勢データ型の選択，および同期・非同期モードの選択をし，スレーブモードを開始します．
  - 同期モードを選択すると，8ミリ秒毎にタイムアウトを検出します．
  - 非同期モードを選択すると，ティーチングペンダントの設定によってタイムアウト検出の有効・無効を切り替えることが出来ます．
  - パラメータ0x0を設定すると，スレーブモードを解除します．

slvChageModeのパラメータ

パラメータ	位置形式	同期/非同期モード
0x000	−	（スレーブモード解除）
0x001	P	同期
0x002	J	同期
0x003	T	同期
0x101	P	非同期
0x102	J	非同期
0x103	T	非同期

  - スレーブモードに切り替える時に，slvChangeModeはロボット動作の停止完了を待ちます．．
  - ただし，この待ち時間が500msecを超えるとエラー600B [ロボット動作中]が発生します．
  - そのため，スレーブモードに切り替える前にロボット動作命令を使用する場合には，ロボット動作命令のオプションとして@Eオプションを使用する事をお勧めします．

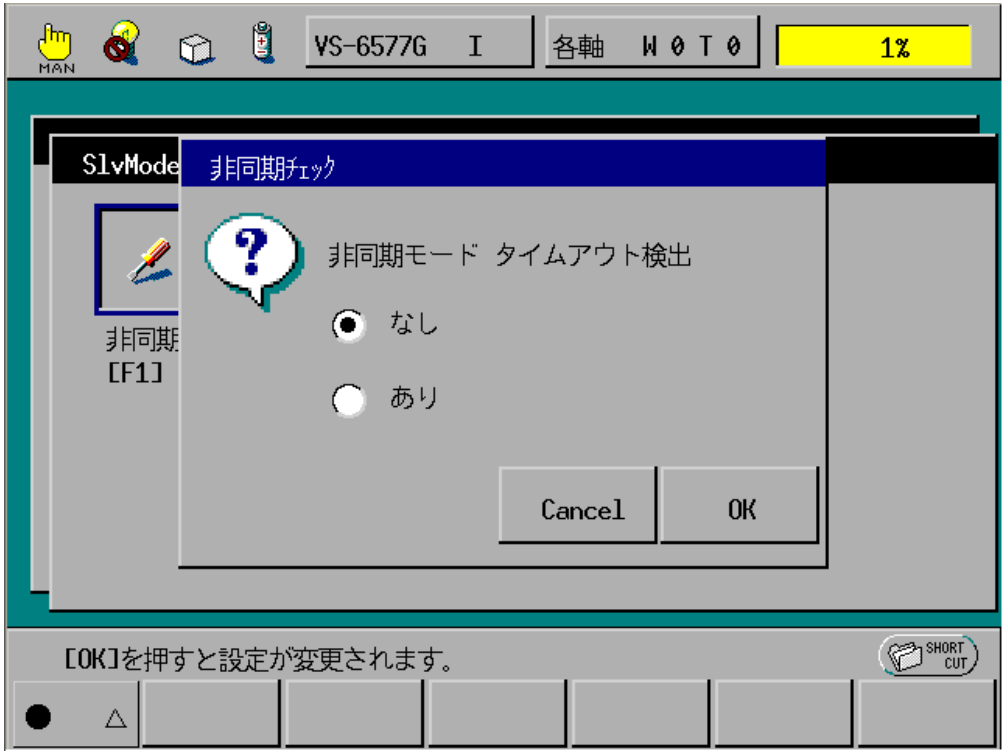
- ティーチングペンダントでの通信プロトコル設定



ティーチングペンダントでの通信設定画面

- UDP接続を行う場合は，リトライシーケンスを実装する必要があります．
  詳しくは”UDP接続時のリトライ手順”の項目を参照してください．

- ティーチングペンダントでの非同期モードタイムアウト検出設定



ティーチングペンダントでの設定画面

スレーブモード関数(2)

- slvGetMode
  - この関数は設定されているスレーブモードの値を返します.
  - 戻り値は,下記になります.
    - 0x000
    - 0x001-0x003
    - 0x101-0x103

スレーブモード関数(3)

- slvMove
  - この関数はslvChangeModeで指定した型の軌道データを送信して，ロボットを動作させます.
  - slvChangeModeの項目も参照してください.

スレーブモードの推奨設定

位置・座標データが
8msec周期で送信される場合

TCP/UDP	同期/非同期	slvMove関数の リトライとタイムアウト設定	
UDP	Sync	Timeout = 2msec Retry count = 7	Timeout = 1000msec Retry count = 4

すなわち，slvChangeModeの値は 0x001 - 0x003となります．

位置・座標データが
8msecより速い周期で送信される場合

TCP/UDP	同期/非同期	slvMove関数の リトライとタイムアウト設定	それ以外の関数の リトライとタイムアウト設定
UDP	Async	Timeout = 2msec Retry count = 0 (この場合，リトライパケットを送信しないで下さい．送信するとリトライパケットによってネットワーク帯域が占拠されます． この場合には，新たなパケットを送信する事でリトライパケットの代用とすることが出来ます．	Timeout = 1000msec Retry count = 3

すなわち，slvChangeModeの値は 0x101 - 0x103となります．

スレーブモード支援機能

- スレーブモードをサポートする便利な機能として，下記のものがあります．
  - StartLog
  - StopLog
- これらの関数はロボットの軌道データのログを取得します．
  - スレーブモードを使って動作させたデータを，WINCAPS3などで確認することが出来ます．

RC7とアプリケーションの設定手順

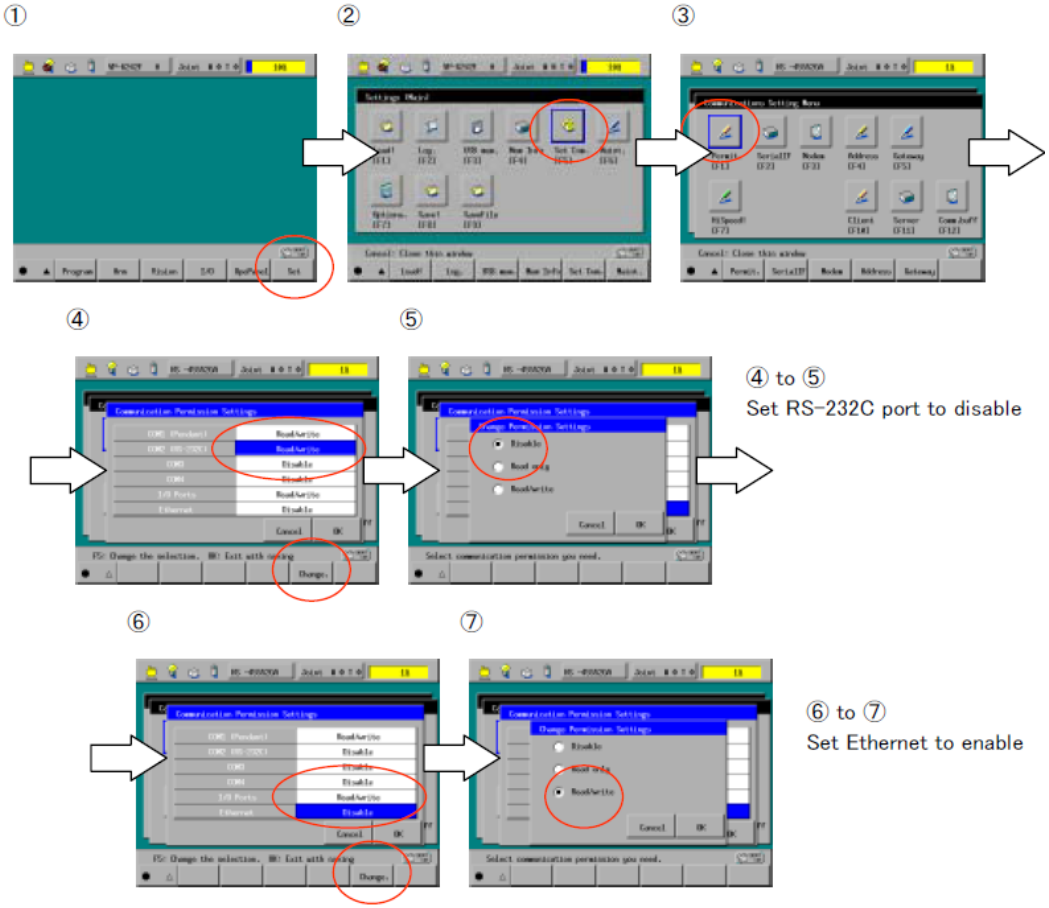
- RC7の設定
  - これらの設定は一度だけ必要です.
- コントローラの準備
  - ティーチングペンダントで外部自動モードに設定してください.
- アプリケーションの動作手順
  - 「RobSlave.pac」をb-CAPで起動してください.
    - 「RobSlave.pac」はスレーブモード際に起動している必要があります.
  - ロボットをスレーブモードで動作させる初期位置に移動させてください.
  - slvChangeMode関数でスレーブモードを開始してください.
  - slvMoveで位置・姿勢データを送信してください.
  - slvChangeModeでスレーブモードを終了してください.
- エラーのクリア
  - 上記関数がエラーを返した場合は，エラークリアを行ってください.

以降のページで，それぞれの手順について詳細に記述します.

RC7の設定 (1)

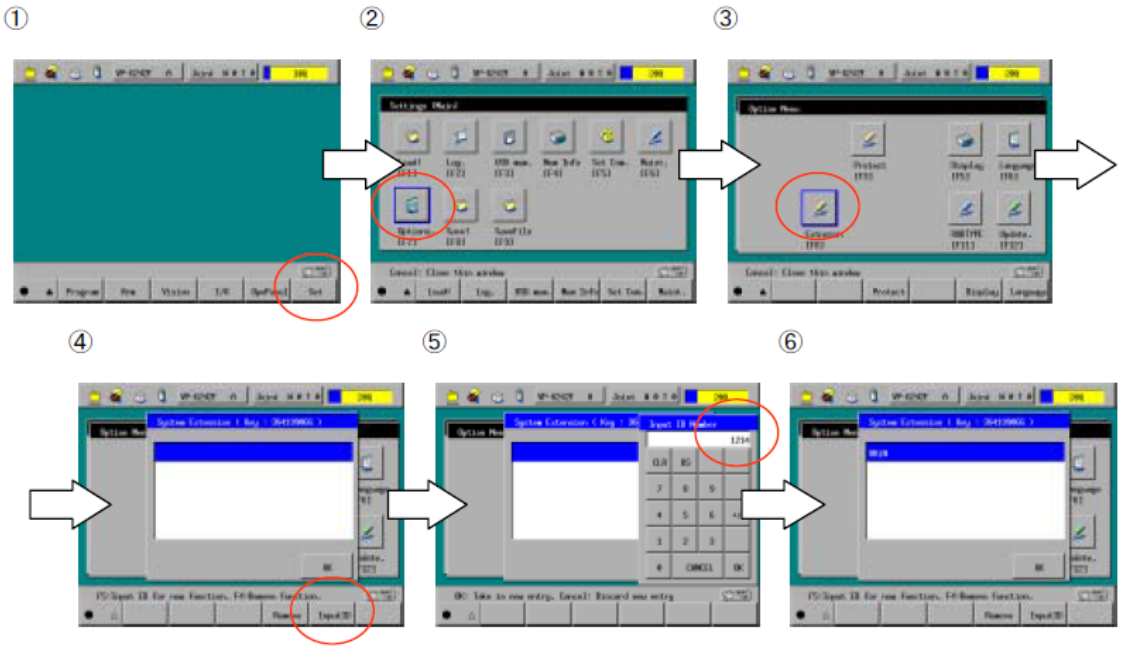
この章の設定は，「4)b-CAPの通信設定」を除きb-CAPユーザーマニュアルの2章と同じ内容です．

- 1) ペンダントで通信設定を行います
  - “Ethernet”を有効にしてください．



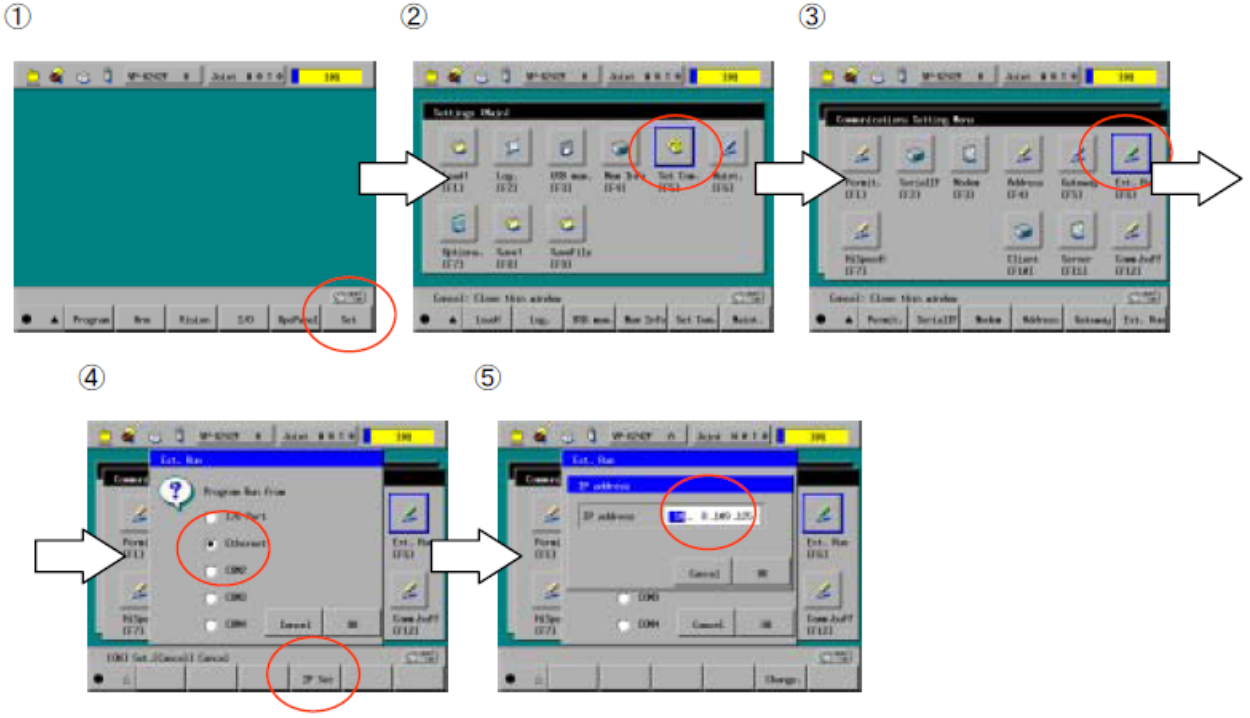
RC7の設定(2)

- 2) オプションの設定
  - “1214”と入力して，“ORiN”オプションを有効にしてください．



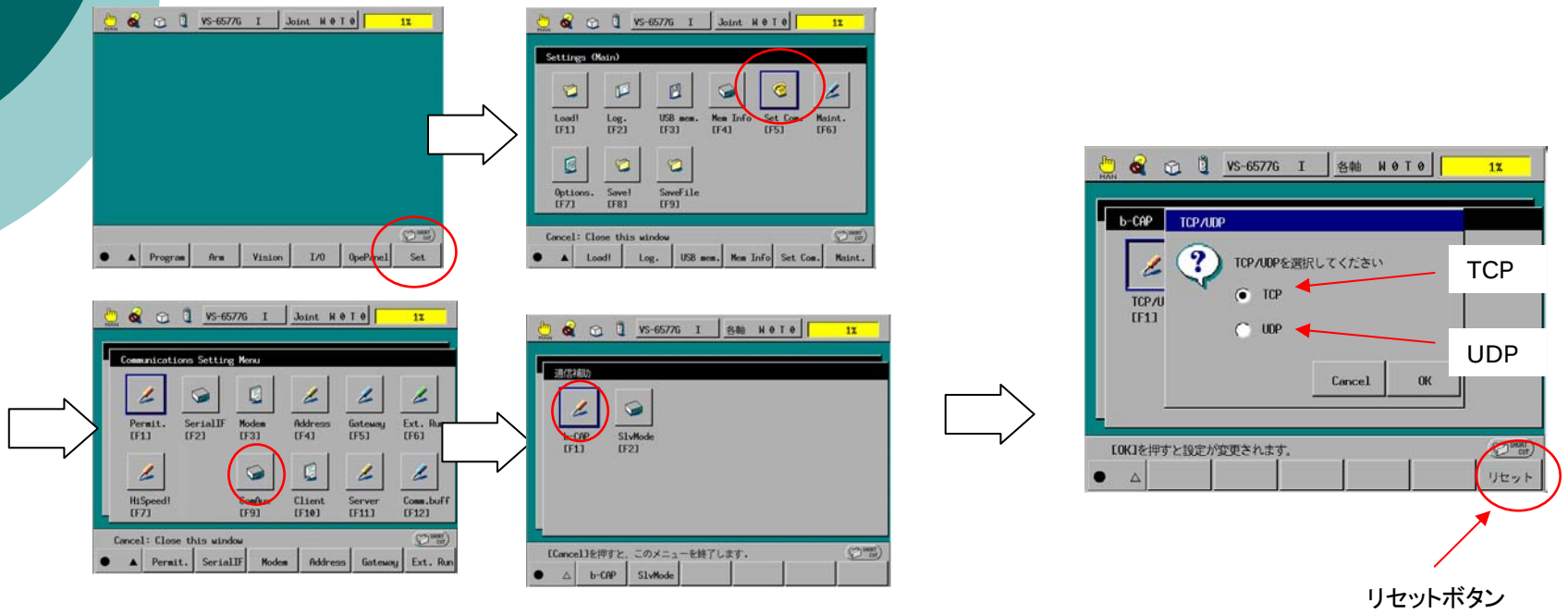
RC7の設定(3)

- 3) 起動権の設定
  - “Ethernet”を選択します.
  - IPアドレスとしてユーザアプリケーションが動作するデバイスのIPアドレスを入力します.



RC7の設定(4)

- 4) b-CAPの通信設定
  - b-CAPの通信設定を行います.（TCP/UDP）
  - 詳しくはslvChangeModeの項目を参照してください.

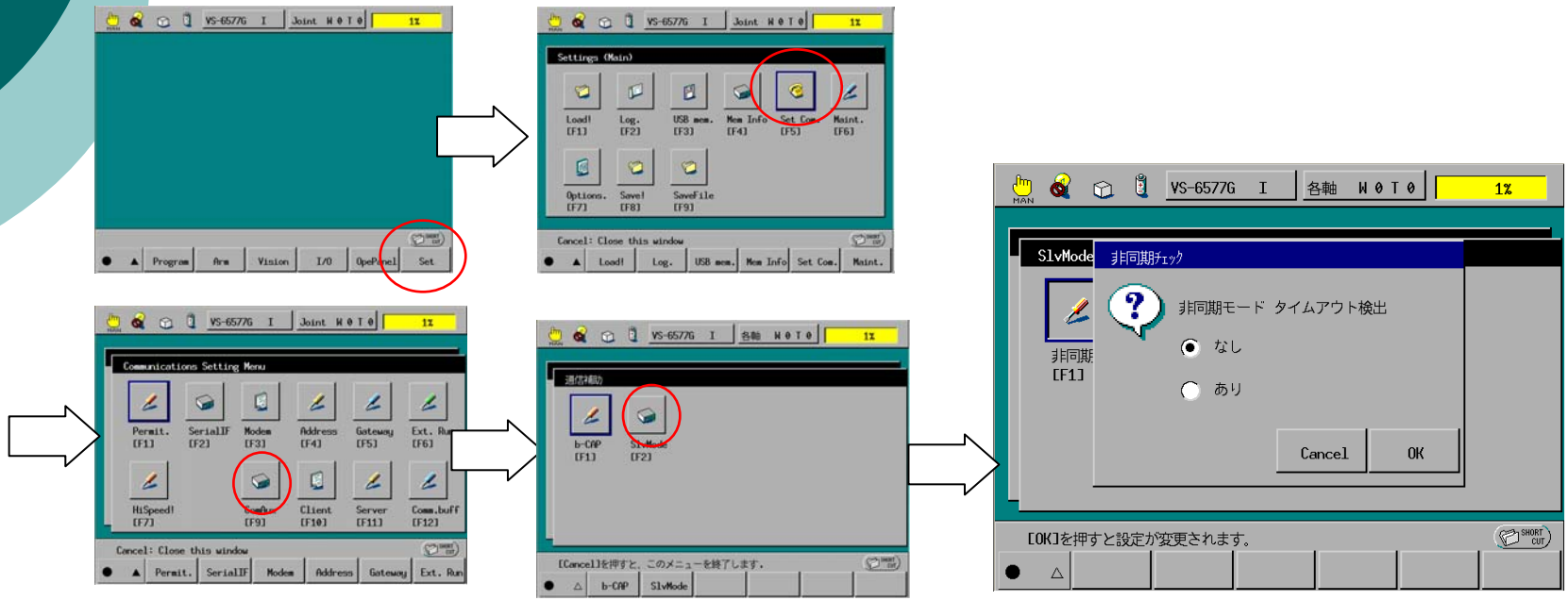


注意:

- ユーザアプリケーションがTCP/UDP接続を正しい手順で切断しないまま終了すると，RC7に再接続する際に不安定になる恐れがあります.
- 上記の場合はコントローラを再起動するか，リセットボタンを押してb-CAPの通信をリセットしてください.

RC7の設定(5)

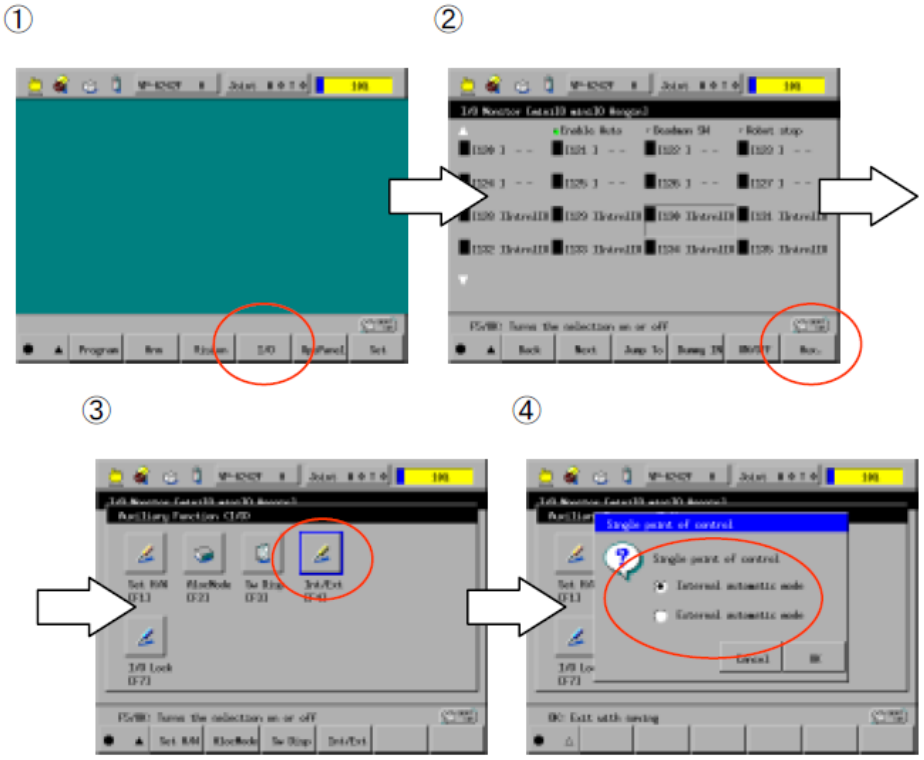
- 非同期モードのタイムアウト検出設定
  - 詳しくはslvChangeModeの項目を参照してください.



RC7の設定(6)

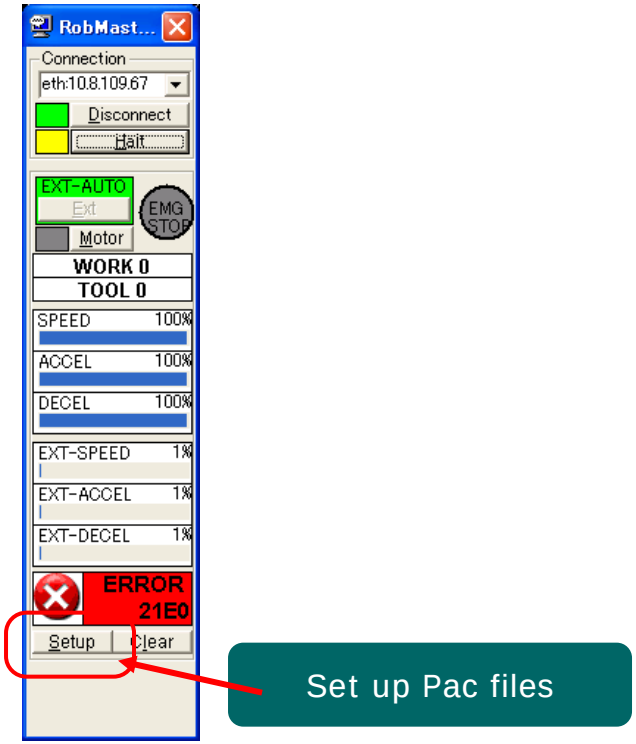
- 6) I/O設定で，単一制御を外部自動モードに設定します．
  - 外部のPCからスレーブモードでロボットを制御するには，外部自動モードにする必要があります．

TIP
グローバルタイプコントローラのみこの設定が必要です．



RC7の設定(7)

- RobSlaveの更新
  - Slave modeに対応しているRobSlave.pacとUserExtension.pacとRobSlave.hをRoboMasterでコントローラに送ります.



コントローラの準備

1. ペンダントでモーターON　もしくはマシンロック状態にしてください.

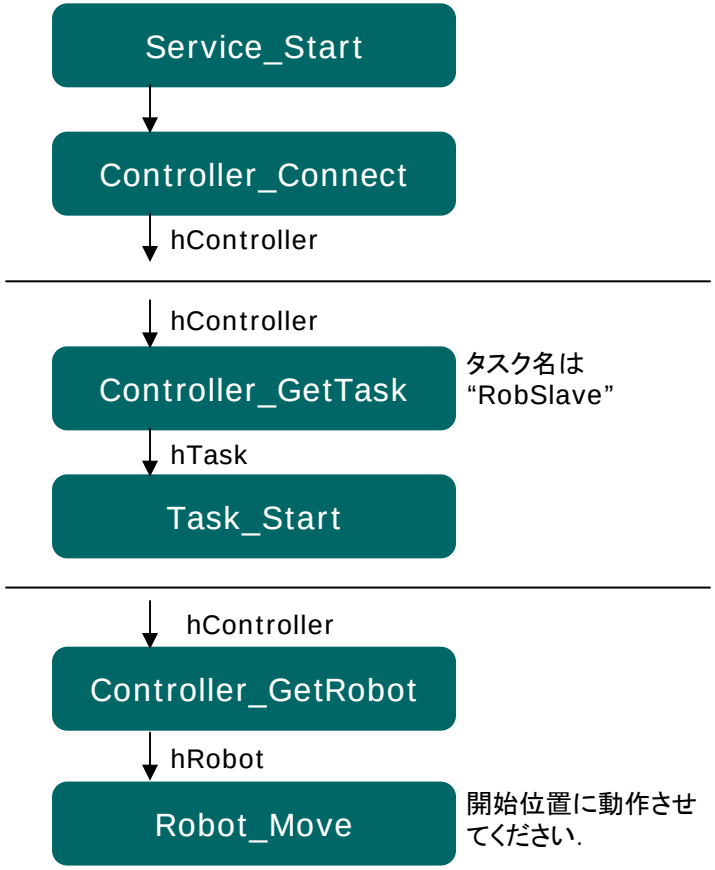
2. ペンダントで自動モードにしてください.

Machine-Lock

Auto mode

アプリケーションの動作手順 (1)

- 1) b-CAPの開始
  - b-CAPでコントローラに接続します.
- 2) “RobSlave.pac”をb-CAPで起動します.
  - Slave modeを使用する場合は“RobSlave.pac”を起動してください.
- 3)Slave modeで動かす始めのポイントにロボットを移動します.

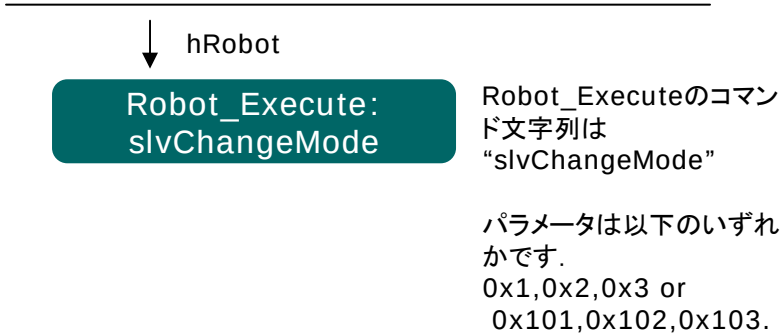


TIPS,

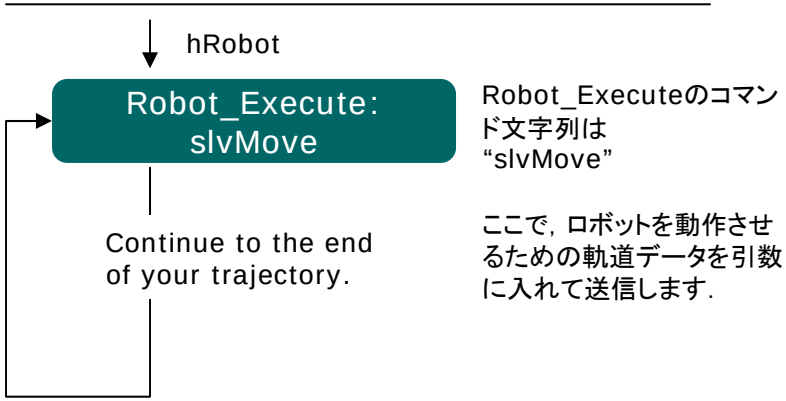
- 安全上，ロボットを動作させる前にロボットタイプの確認を行ってください.
  ロボットクラスのシステム変数@TYPEで確認することができます.

アプリケーションの動作手順(2)

- 4) slvChageModeでSlave modeを変更してください.

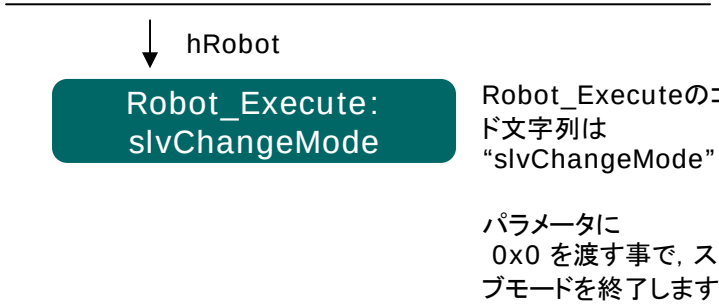


- 5)slvMoveで軌道データを送信してください.



アプリケーションの動作手順(3)

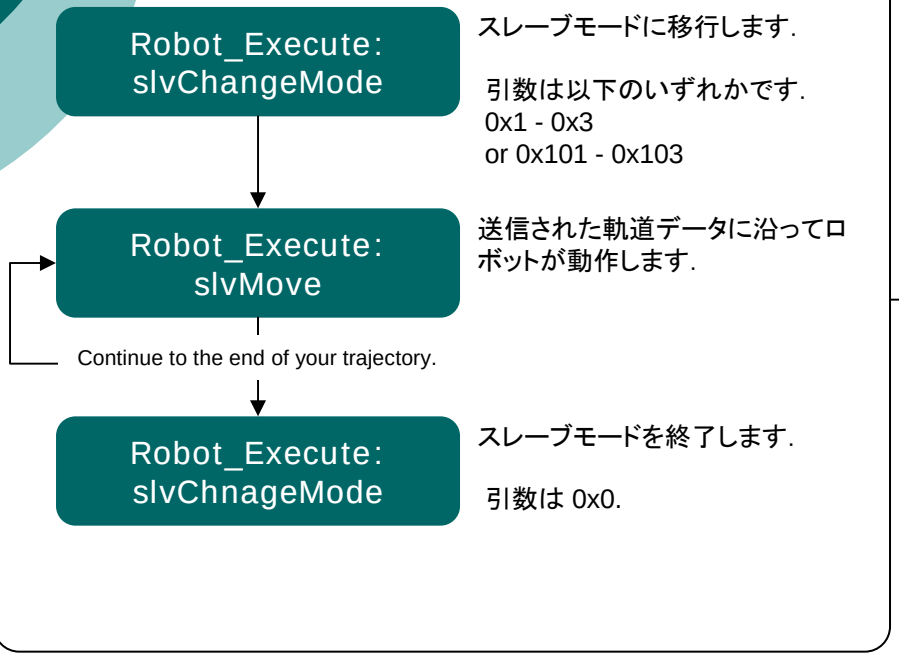
- 6)アプリケーションを終了・停止する前にslvChangeModeでノーマルモードに変更してください.



エラークリア

- 関数がエラーを返した時は，フォローシーケンスでエラーのクリアを行ってください．

スレーブモードによるロボット制御の代表例：



エラー発生後のエラークリア手順：

Robot_Executeの戻り値としてS_OK以外が返された場合は，エラーが発生しています．

エラーが発生すると，
・RobSlave.PACは実行を停止します．
・また，スレーブモードは解除され，ノーマルモードに移行します．
・ティーチングペンダントに該当するエラーが表示されています．

そのため，以下の手順でエラーをクリアーし，再度スレーブモードに復帰する必要があります．

・b-CAPでエラークリアを実行
・RobSlave.PACの再起動
・ロボットを初期位置に戻す
・スレーブモードに再設定

関数リファレンス

- Slave modeの関数はb-CAPのRobot_Executeの関数の1つとして実装されています.

Command	Parameters	Return value	Function
slvChangeMode	VT_I2 0x0:Disable Slave mode 0x1:Slave mode(P type、**Sync**) 0x2:Slave mode (J type、**Sync**) 0x3:Slave mode (Ttype、**Sync**) 0x101: Slave mode ド(P type、**Async**) 0x102:Slave mode (J type、**Async**) 0x103:Slave mode (Ttype、**Async**)	None	Enable/Disable the Slave mode
slvGetMode	None	VT_I2 0x0:Disable Slave mode 0x1:Slave mode(P type、**Sync**) 0x2:Slave mode (J type、**Sync**) 0x3:Slave mode (Ttype、**Sync**) 0x101: Slave mode ド(P type、**Async**) 0x102:Slave mode (J type、**Async**) 0x103:Slave mode (Ttype、**Async**)	Returns current mode
slvMove	<Ptype:VT_R4|VT_ARRAY>	<Jtype:VT_R4|VT_ARRAY> Current joint angle	Move to a destination position

関数リファレンス

- slvChangeMode

Communication sample - SlvChangeMode				
Activate/release SlaveMode				
Transmitted packet	Client→server: 01 54 00 00 00 09 1D 00 00 40 00 00 00 03 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 24 00 00 00 08 00 01 00 00 00 1A 00 00 00 73 00 6C 00 76 00 43 00 68 00 61 00 6E 00 67 00 65 00 4D 00 6F 00 64 00 65 00 0A 00 00 00 03 00 01 00 00 00 02 00 00 00 04			
	Name	Description	Type	Value
		Binary		
	hRobot	Handle of controller (Ref:Controller_GetRobot)	VT_I4	0x00000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrCommand	Command strings	VT_BSTR	"slvChangeMode"
		24 00 00 00 08 00 01 00 00 00 1A 00 00 00 73 00 6C 00 76 00 43 00 68 00 61 00 6E 00 67 00 65 00 4D 00 6F 00 64 00 65 00		
	vntParam	Command parameter	VT_I2	0x0002
		0A 00 00 00 03 00 01 00 00 00 02 00 00 00		
Received packet	server → Client: 01 10 00 00 00 09 1D 00 00 00 00 00 00 00 00 04			
	None	-	-	-
		-		

関数リファレンス

- slvGetMode

Communication sample - SlvGetMode				
Send robot move position (periodical)				
Transmitted packet	Client →server: 01 4A 00 00 00 0B 00 00 00 40 00 00 00 03 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 1E 00 00 00 08 00 01 00 00 00 14 00 00 00 73 00 6C 00 76 00 47 00 65 00 74 00 4D 00 6F 00 64 00 65 00 06 00 00 00 00 00 01 00 00 00 04			
	Name	Description	Type	Value
		Binary		
	hRobot	Handle of controller (Ref:Controller_GetRobot)	VT_I4	0x0000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrCommand	Command strings	VT_BSTR	"slvGetMode"
		1E 00 00 00 08 00 01 00 00 00 14 00 00 00 73 00 6C 00 76 00 47 00 65 00 74 00 4D 00 6F 00 64 00 65 00		
	vntParam	Option String	VT_EMPTY	Empty
		06 00 00 00 00 00 01 00 00 00		
	server → Client: 01 1C 00 00 00 0B 00 00 00 00 00 00 01 00 08 00 00 00 02 00 01 00 00 00 02 00 04			
		Return Parameter	VT_I2	2
		08 00 00 00 02 00 01 00 00 00 02 00		

関数リファレンス

- slvMove

Communication sample - SlvMove

Send robot move position (periodical)

Transmitted packet

Client →server:

```
01 5C 00 00 00 0A 1D 00   00 40 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 18 00 00
00 08 00 01 00 00 00 0E   00 00 00 73 00 6C 00 76
00 4D 00 6F 00 76 00 65   00 1E 00 00 00 04 20 06
00 00 00 FF 32 46 41 BF   71 71 42 C0 67 8C 41 E0
41 0F 42 A1 6D A3 41 AF   9A BF 41 04
```

Name	Description	Type	Value
	Binary		
hRobot	Handle of controller (Ref.:Controller_GetRobot)	VT_I4	0x0000001
	00 00 00 03 00 01 00 00 00 01 00 00 00 0A		
bstrCommand	Command parameter	VT_BSTR	"slvMove"
	00 08 00 01 00 00 00 0E 00 00 00 73 00 6C 00 76 00 4D 00 6F 00 76 00 65 00 18 00 00		
vntParam	Command parameter	VT_R4|ARRAY	12.38745 60.36108 17.55066 35.81433 20.42853 23.95053
	00 00 00 FF 32 46 41 BF 71 71 42 C0 67 8C 41 E0 41 0F 42 A1 6D A3 41 AF 9A BF 41 1E 00 00 00 04 20 06		

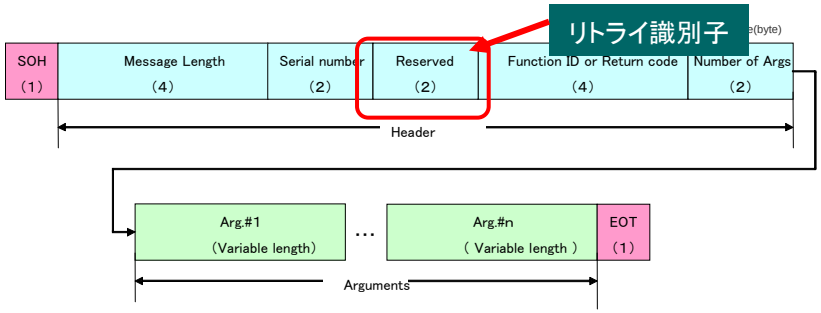
server → Client:

```
01 3A 00 00 00 0A 1D 00   00 00 00 00 00 01 00 26
00 00 00 04 20 08 00 00   00 00 A3 45 41 C1 71 71
42 80 C0 8B 41 E0 41 0F   42 A0 6D A3 41 7D 9C BF
41 00 00 00 00 00 00 00   00 04
```

Return Parameter	VT_R4|ARRAY	12.35229 60.36109 17.46899 35.81433 20.42853 23.95141 0 0
00 00 00 04 20 08 00 00 00 00 A3 45 41 C1 71 71 42 80 C0 8B 41 E0 41 0F 42 A0 6D A3 41 7D 9C BF 41 00 00 00 00 00 00 00 00 26		

UDP接続時のリトライ手順 (1)

- UDP接続を行うためには，パケットをリトライする手順の実装が必要です．
- b-CAPパケットのヘッダに有る“予約”領域をリトライパケットの識別子として使用します．



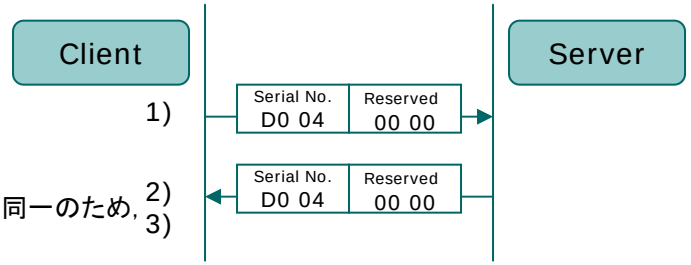
UDP接続時のリトライ手順(2)

正常時の通信

1) パケットを送信

2) パケットを受信

3) 送信したパケットのシリアル番号と受信したパケットのシリアル番号が同一のため，正常に通信が完了したことがわかります．



リトライ発生時

1) パケットを送信

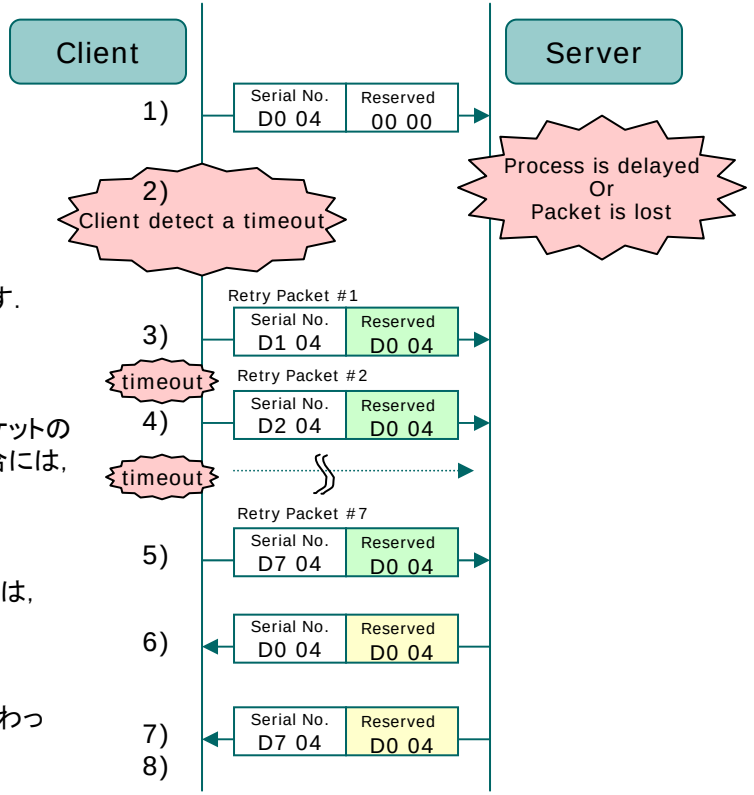
2) パケットを受信

3) - 5) タイムアウトを検出したため，リトライパケットを作成し，送信します．
- シリアル番号を＋1します．
- 正常パケットの予約領域には，通常0が入っています．

リトライパケットの場合，予約領域には，前回送信されたオリジナルパケットのシリアル番号が格納されています．したがって，予約領域が0以外の場合には，そのパケットがリトライパケットであることが解ります．

6) パケットを受信
クライアントが送信した最後のパケットはが異なるシリアル番号の場合には，パケットを捨て去る必要があります．

7) パケットを受信．
8) 受信したパケットのシリアル番号が同じであるため，通信が正常に終わったことがわかります．



UDP接続時のリトライ手順(3)

- 正常なパケットの例：.
  - パケットのシリアル番号は “D0 04”.
  - 予約領域は “00 00”.

Packet TX(Client->Server)

01 64 00 00 00 **D0 04 00 00** 40 00 00 00 03 00 0A 00
00 00 03 00 01 00 00 00 01 00 00 00 18 00 00 00 08 00
01 00 00 00 0E 00 00 00 73 00 6C 00 76 00 4D 00 6F 00
76 00 65 00 26 00 00 00 04 20 08 00 00 00 B3 A3 5F C2
66 DB CB 41 86 09 33 43 60 EF 07 42 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 04

Packet RX(Server->Client)

01 3A 00 00 00 **D0 04 00 00** 00 00 00 00 01 00 26 00
00 00 04 20 08 00 00 00 67 B2 65 C2 00 28 CE 41 81 77
32 43 13 76 09 42 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 04

- リトライ時のパケットの例：.

Packet TX(Client->Server)

01 64 00 00 00 **D0 04 00 00** 40 00 00 00 03 00 0A 00

Packet TX(Client->Server) Retry #1

01 64 00 00 00 **D1 04 D0 04** 40 00 00 00 03 00 0A 00

Packet TX(Client->Server) Retry #2

01 64 00 00 00 **D2 04 D0 04** 40 00 00 00 03 00 0A 00

Packet TX(Client->Server) Retry #3

01 64 00 00 00 **D3 04 D0 04** 40 00 00 00 03 00 0A 00
00 00 03 00 01 00 00 00 01 00 00 00 18 00 00 00 08 00
01 00 00 00 0E 00 00 00 73 00 6C 00 76 00 4D 00 6F 00
76 00 65 00 26 00 00 00 04 20 08 00 00 00 B3 A3 5F C2
66 DB CB 41 86 09 33 43 60 EF 07 42 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 04

Packet RX(Server->Client) #1

01 3A 00 00 00 **D1 04 D0 04** 00 00 00 00 01 00 26 00

Packet RX(Server->Client) #2

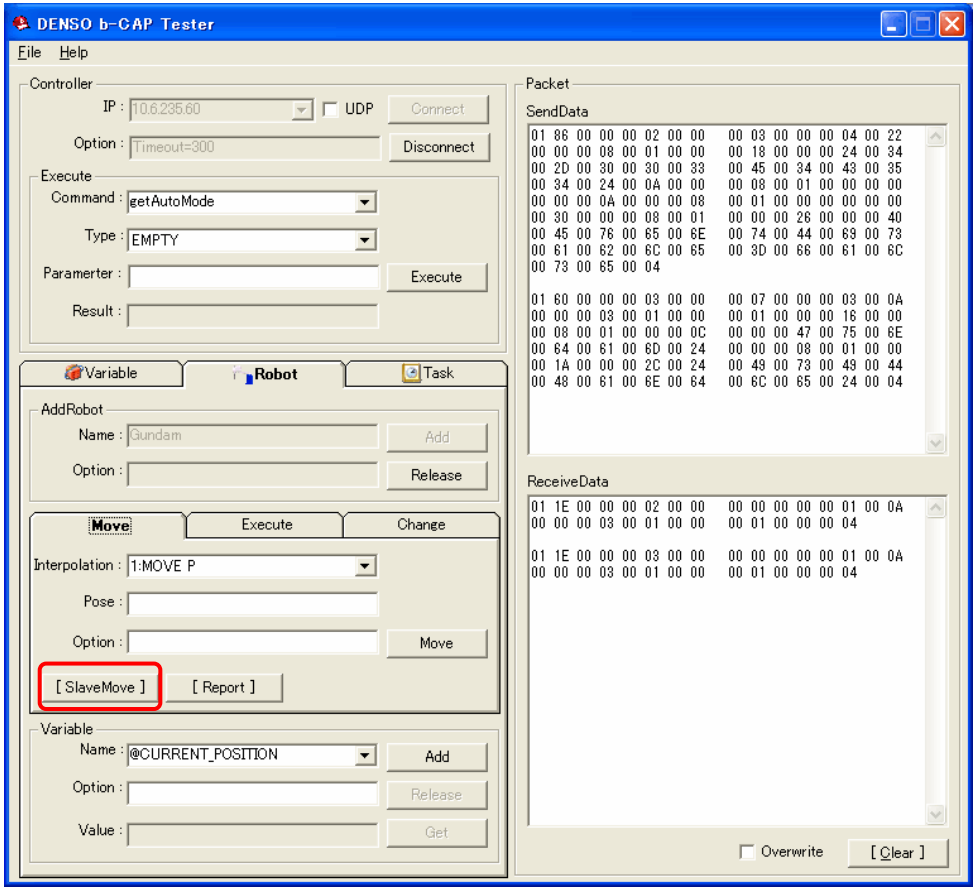
01 3A 00 00 00 **D3 04 D0 04** 00 00 00 00 01 00 26 00
00 00 04 20 08 00 00 00 67 B2 65 C2 00 28 CE 41 81 77
32 43 13 76 09 42 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 04

b-CAP Testerの使い方(1)

- b-CAP Testerはb-CAPの動作を確認するためのツールです.
  - ORiN2/CAP/b-CAP/CapLib/DENSO/Bin/b-CAPTester.exe
  - 送受信されるパケットがコマンドごとに表示されます.
- Slave Mode時にb-CAP Testerを用いてロボットを動かすためには，下記の準備を行ってください.
  - 制御ログを取得したWINCAPS3のプロジェクトファイルを準備してください.
    slvMoveは制御ログの指令値を使用してロボットを動作させます.
    ロボットの姿勢は制御ログを取り始めた場所にあわせてください.
  - コントローラの準備を行ってください.
    詳しくは，Procedures of setting - Setting up your RC7を参照してください.
  - コントローラ内にあるRoboSlave.pacを起動してください.
    b-CAP Testerからも起動することができます.

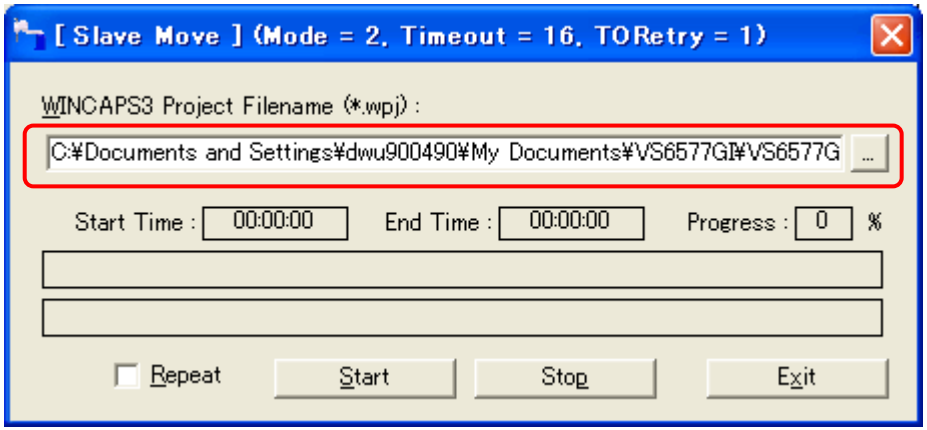
b-CAP Testerでの使い方(1)

- ロボットに接続した後，「Slave Mode」ボタンを押し，Slave Moveウィンドウを立ち上げます．

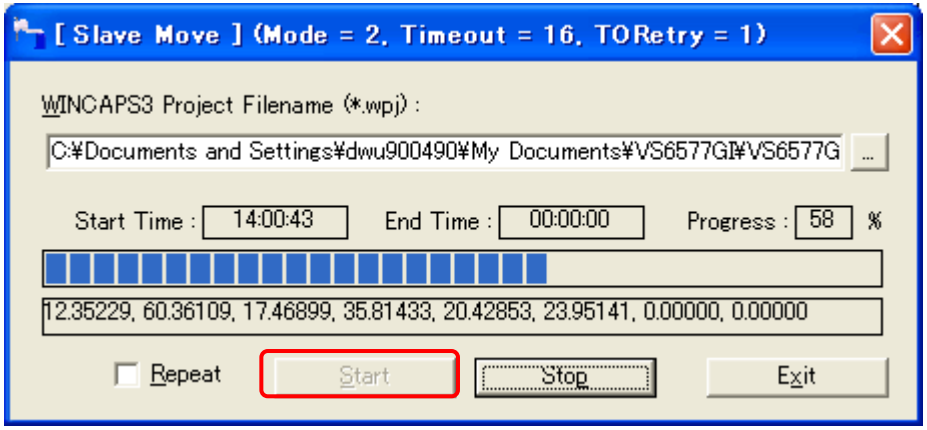


b-CAP Testerでの使い方(2)

- 制御ログを保存しているWINCAPS3プロジェクトを指定します.



- Startボタンを押すと動き始めます.

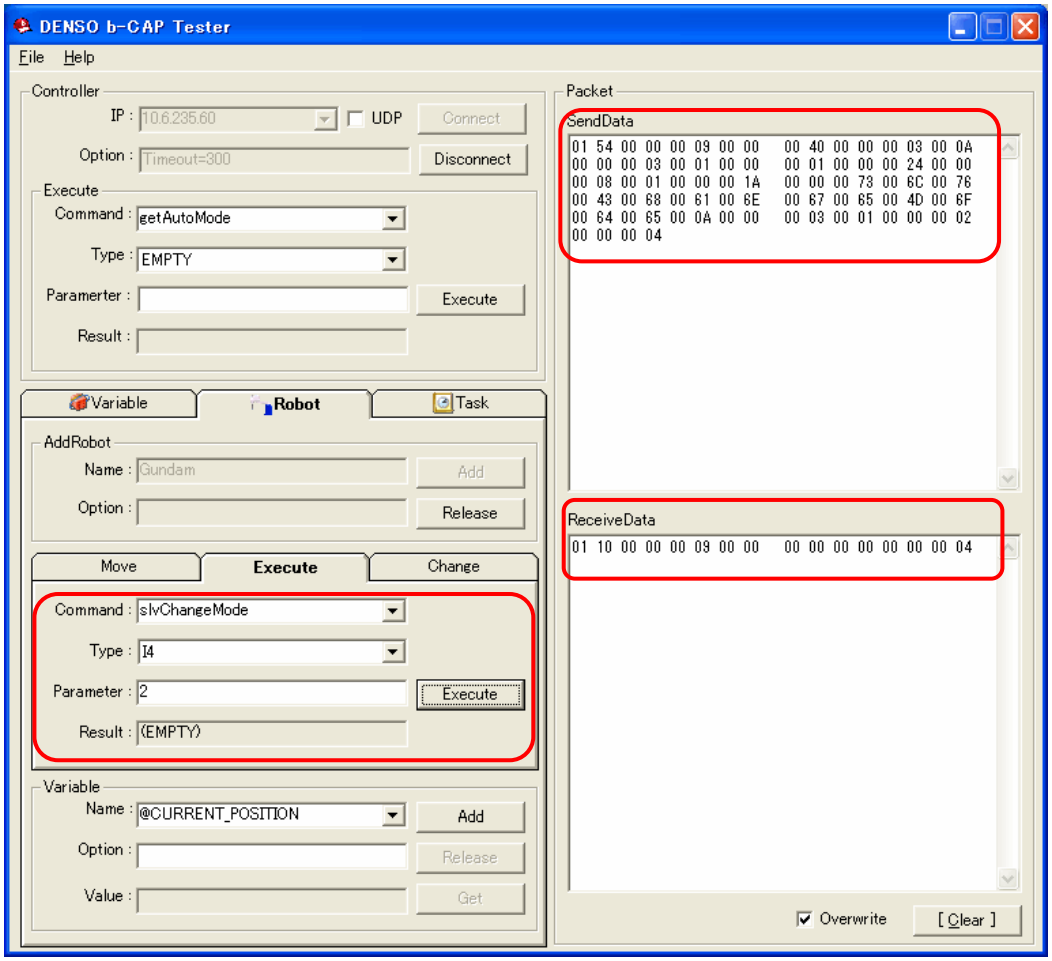


パケットの確認(1)

- Slave Moveウィンドウを使用してロボットを動かす場合，送受信されるパケットは表示されません．
- パケットを確認したい場合は，ロボットのExecuteからコマンドを発行して確認することができます．
  - まずslvChangeModeでSlaveModeを変更します．
  - 次にslvMoveで移動先を指定します．
  - Executeコマンドでは8ミリ秒毎に指令値を送れないので，コントローラ側でエラーが発生します．

パケットの確認(1)

- ExcuteタブのslvChangeModeでモードを変更します.



パケットの確認(2)

- slvMoveコマンドで動作させます.
  - コマンド発生後, コントローラで必ずエラーになります.