RN700 通信仕様書

Ver. 1.6Draft_e

2020/09/07

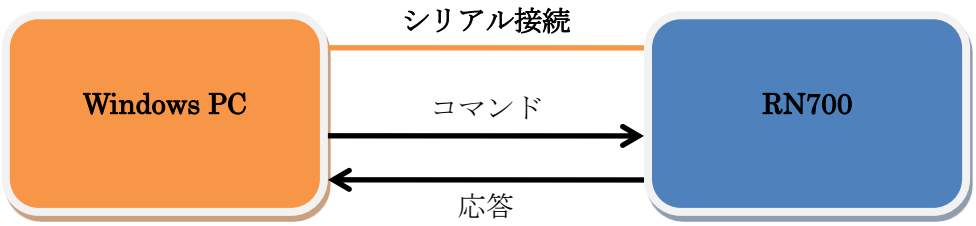
株式会社イーアールアイ

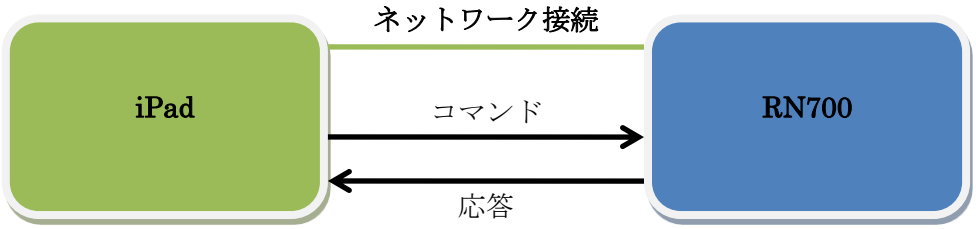
目次

1 概要

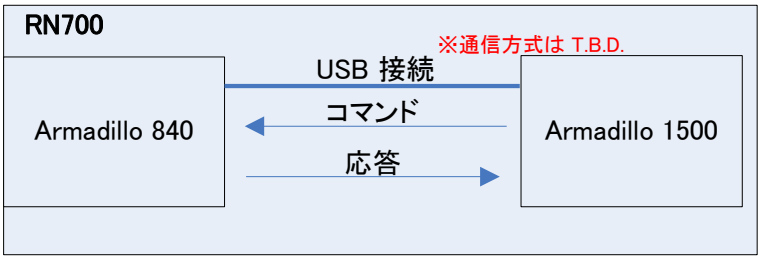
本書は RN700 を制御するコマンド仕様を示した仕様書です。

ネットワーク接続またはシリアル接続を使って、本書に定義されたコマンドを送信することで RN700 をコントロールすることができます。





また、Armadillo 1500 を使用し画像処理を行う際、Armadillo 840 – Armadillo 1500 間でコマンドを使用します。使用するコマンドに関しては、RN700FW 仕様書の動作シーケンス項目を参照してください。



2 関連資料

本書の関連資料を下記に記載します。

資料名
RN700 FW 仕様書

3 コマンド一覧

コマンド一覧を記載します。

分類	コマンド	説明
読出コマンド	getVersion	バージョン取得
	getApiList	コマンド一覧取得
	getStatus	ステータス取得
	getOperatingStatus	動作状態取得
	getTemperature	温度取得
	getIllumination	光量取得
	getAcceleration	加速度取得
	getLimitSwitch	リミットスイッチ状態取得
	getCameraShutterWidth	Shutter Width 取得
	getCameraGain	カメラ Gain 取得
	getLed1RGB	LED1 Red, Green, Blue 取得
	getLed2RGB	LED2 Red, Green, Blue 取得
	getLcdPwm	LCD Pwm 取得
	getLcdBitmap	LCD 表示ビットマップ取得
	getImage	画像取得
	getAnalysisResults	測定結果取得
	getAnalysisType	測定粒種取得
	getAnalysisLevelVal	選別レベル取得
	getAnalysisLevel	選択中の選別レベル取得
	getSettingFile	設定ファイルの取得
	getTimeout	通信タイムアウト時間取得
	getProfileData	プロファイルデータ取得（デバッグ用）
	getCaptureImage	連続キャプチャ画像取得
	getOperatingMode	動作モード取得
	getDate	日時取得
	getBinaryFileEx	バイナリデータ受信（ブロック転送）
	getPrintPattern	印字設定取得
	getLotNumber	ロット番号取得
	getSaveData	データ保存設定取得
設定コマンド	setCameraShutterWidth	Shutter Width 設定
	setCameraGain	カメラ Gain 設定
	setLed1RGB	LED1 Red, Green, Blue 設定
	setLed2RGB	LED2 Red, Green, Blue 設定
	setLedPower	LED1, LED2 の点灯設定
	setLcdPower	LCD の電源設定
	setLcdPwm	LCD Pwm 設定

	setLcdBitmap	LCD 表示ビットマップ設定
	setAnalysisType	測定粒種設定
	setAnalysisLevel	選別レベル選択
	setSettingFile	設定ファイルの設定
	setDate	日時設定
	setTimeout	通信タイムアウト時間設定
	setOperatingMode	動作モード設定
	setUserInfo	ユーザー情報設定（デバッグ用）
	setBinaryFile	バイナリデータ設定
	setBinaryFileEx	バイナリデータ送信（ブロック転送）
	setPrintPattern	印字設定
	setLotNumber	ロット番号設定
	setSaveData	データ保存設定
操作コマンド	writeFlash	設定データ保存
	readFlash	設定データ読み込み
	calibrateRGB	LED 及び LCD 発光時のカラーセンサ取得 (デバッグ用)
	initialCalibration	歪み補正、LED 及び LCD の光量調整
	captureImage	撮影
	capAnalysisImg	照明連動撮影（デバッグ用）
	deleteImage	画像を全消去
	actImageProcessing	画像処理
	startAnalysis	測定開始（測定後に応答メッセージ送信）
	startAnalysisAsync	測定開始（測定前に応答メッセージ送信）
	printResult	測定結果印字
	dispLcdBmp	LCD に BMP 画像を表示
	dispLcdRGB	LCD 単色塗りつぶし
	dispOledText	OLED に文字列表示
	dispOledMsg	指定した ID のメッセージを OLED に表示
	soundBuzzer	ブザー鳴動
	startAccMonitoring	加速度センサ値計測開始（デバッグ用）
	stopAccMonitoring	加速度センサ値計測終了（デバッグ用）
	readAccelerationReg	加速度センサレジスタ値取得（デバッグ用）
	writeAccelerationReg	加速度センサレジスタ値書込み（デバッグ用）
	startCaptureLoop	連続キャプチャの開始
	stopCaptureLoop	連続キャプチャの終了
	rawToImage	RAW 画像変換
	chkTray	トレイ確認
アップデート コマンド	writeFirmware	ファームウエア更新
	sendFirmware	ファームウエア送信

	getFirmwareStatus	ファームウエア更新状態取得
Armadillo 840 – Armadillo 1500 通信用コマンド	setImgProocBoardStatus	Armadillo 1500 状態通知
	setImgProcRequestNotify	画像処理リクエスト通知設定
	setResultData	結果ファイル送信

コマンド形式

コマンドの発行及び応答は下記のような形式となります。

{“項目 1”: 値 1, “項目 2” : 値 2, … “項目 n” : 値 n}

コマンドの文字列は全て文字コード UTF-8 で送信します。

項目 1, 項目 2, …, 項目 n は、それぞれ値 1, 値 2, …, 値 n に対応しています。

値は各項目が使用するデータ種にあわせて下記のように記述します。

データ種	指定例
文字列	“string”
数値	1
複数指定	[1, “string”, …]
指定なし	[ ]

また、ダブルコーテーション(") で囲まれた部分以外のコマンドに含まれる半角スペースは無視されます。

3.1 コマンド発行形式

コマンド発行形式を下記に記載します。

{	“	M	e	t	h	o	d	“	:	“	コマンド名	“	,
“	p	A	r	a	m	s	“	:	パラメーター				,
“	i	D	“	:	番号		}						

項目	内容
{	開始文字
“method” :	値に発行するコマンド名を指定します。
“コマンド名”	発行するコマンド名を記述します。コマンド名は半角英数 32 文字以内で指定します。
,	区切り文字
“params” :	値に発行するコマンドのパラメーターを指定します。パラメーターがない場合、省略することが可能です。
パラメーター	パラメーターを記述します。
,	区切り文字
id	値に id 番号を指定します。
番号	id 番号を指定します。id 番号はコマンド発行ごとにインクリメントします。値の範囲は 0 ～ 65535 で、id が最大値になった場合、次の id は 0 に戻ります。
}	終了文字

例:

```
{“method”: “getVersion”, “params”: [], “id”: 1}
```

コマンド	getVersion
パラメーター	なし
id 番号	1

{“method”: “getVersion”, “id”: 1}

コマンド	getVersion
パラメーター	なし
id 番号	1

バイナリデータを送信するコマンドの場合は、コマンド発行に続いてバイナリデータを送信します。
RN700 がバイナリデータ受信中にエラーが発生した場合は残りのデータを受け捨て、その後エラー応答します。
また、受信中にタイムアウトが発生した場合は、発生時にエラー応答し受信を終了します。タイムアウトの時間は、setTimeout コマンドで設定された時間となり最後のデータ受信からの時間になります。

{	“	m	e	t	h	o	d	“	:	“	コマンド名	“	,
“	p	a	r	a	m	s	“	:	パラメーター				,
“	i	d	“	:	番号		}	バイナリデータ					

バイナリデータは下記フォーマットで送信します。

データ数	内容
4 バイト	データサイズ
n バイト (データサイズで指定したサイズ)	データ ・ ・
4 バイト	チェックサム (データサイズ及びデータ部をバイト単位に加算した値)

バイナリデータはリトルエンディアンとなります。

3.2 コマンド応答形式

コマンド応答形式を下記に記載します。

正常時

{	“	r	e	s	u	l	t	“	:	値	,	“	i	d	“	:	番号	}

項目	内容
{	開始文字
“result”:	値に発行コマンドに対する戻り値を指定します。
値	コマンドの戻り値
,	区切り文字
“id”:	値に id 番号を指定します。
番号	発行コマンドと同じ id 値を返します。
}	終了文字

例：{“result”: “1.0”, “id”: 1}

戻り値	“1.0”
id 番号	1

正常時(バイナリデータ送信時)

コマンド応答時にバイナリデータを送信するコマンドの場合は、コマンド応答に続いてバイナリデータを送信します。RN700 がバイナリデータ送信中にエラーが発生した場合は、送信を中断します。

{	“	r	e	s	u	l	t	“	:	“	b	i	n	a	r	y	“	,
“	i	d	“	:	番号	}	バイナリデータ											

項目	内容
{	開始文字
“result”:	値に発行コマンドに対応する戻り値を指定します。
“binary”	バイナリデータ送信時は必ず戻り値を “binary” とします。
,	区切り文字
“id”:	値に id 番号を指定します。
番号	発行コマンドと同じ id 値を返します。
}	終了文字
バイナリデータ	送信するバイナリデータ。 バイナリデータのフォーマットは、コマンド発行形式を参照。

異常時

{	“	e	r	r	o	r	“	:	[	エラー番号	,	“	エラーメッセージ	“	]	,
“	i	d	“	:	番号	}										

項目	内容
{	開始文字
“error”:	値にエラー番号とエラーメッセージを返します。
[	値を複数返す場合の開始文字です。
エラー番号	エラー番号を指定します。
,	区切り文字
“エラーメッセージ”	エラーメッセージを指定します。
]	値を複数返す場合の終了文字です。
,	区切り文字
“id”:	値に id 番号を指定します。
番号	発行コマンドと同じ id 値を返します。発行コマンドの id が不正値だった場合 null を返します。
}	構文の終了を表します。

例：

{“error”: [1, “Error Message”], “id”: 1}

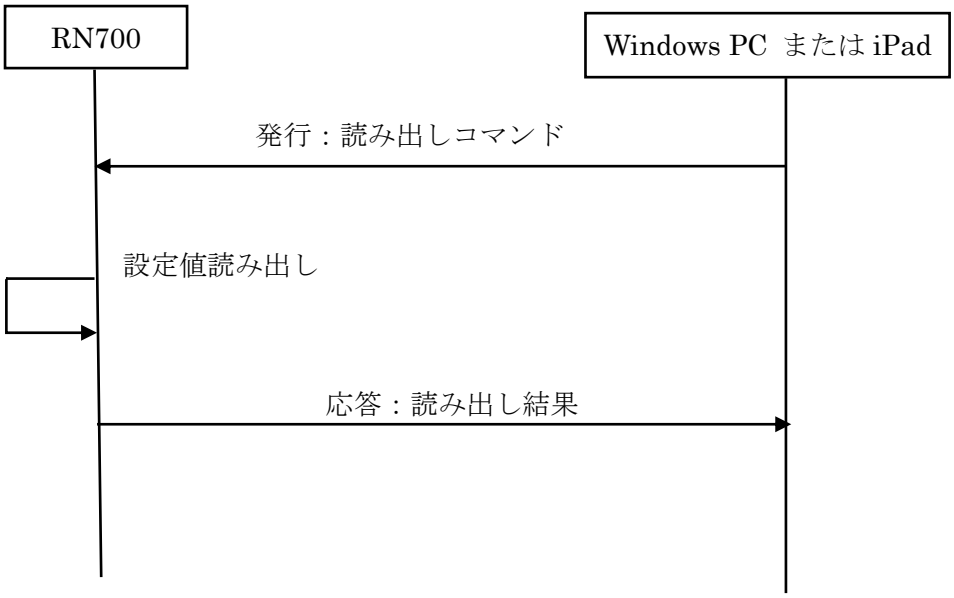
エラー番号	1
エラーメッセージ	“Error Message”
id 番号	1

{“error”: [1, “Error Message”], “id”: null}

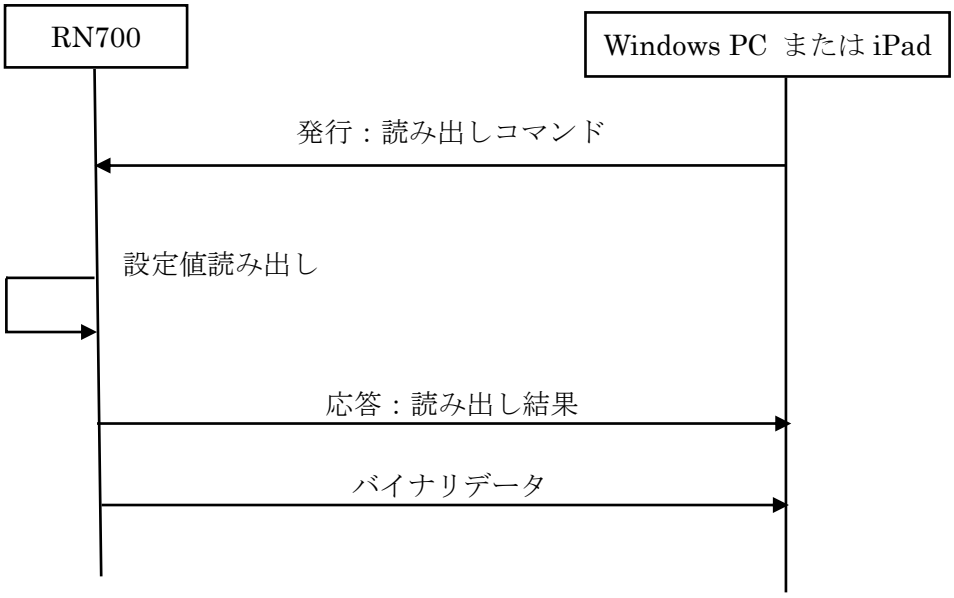
エラー番号	1
エラーメッセージ	“Error Message”
id 番号	null

4 シーケンス

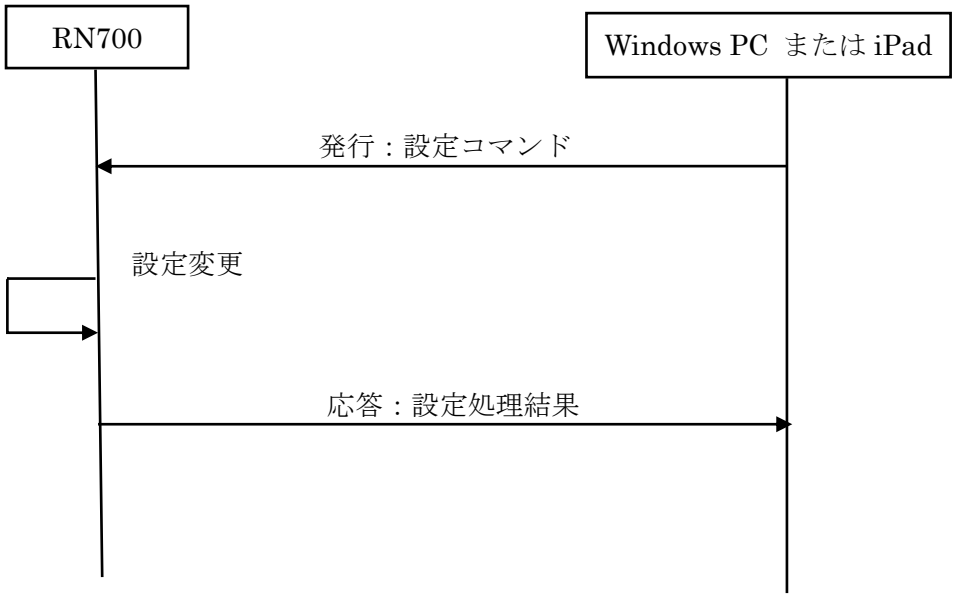
4.1 読み出しコマンドシーケンス



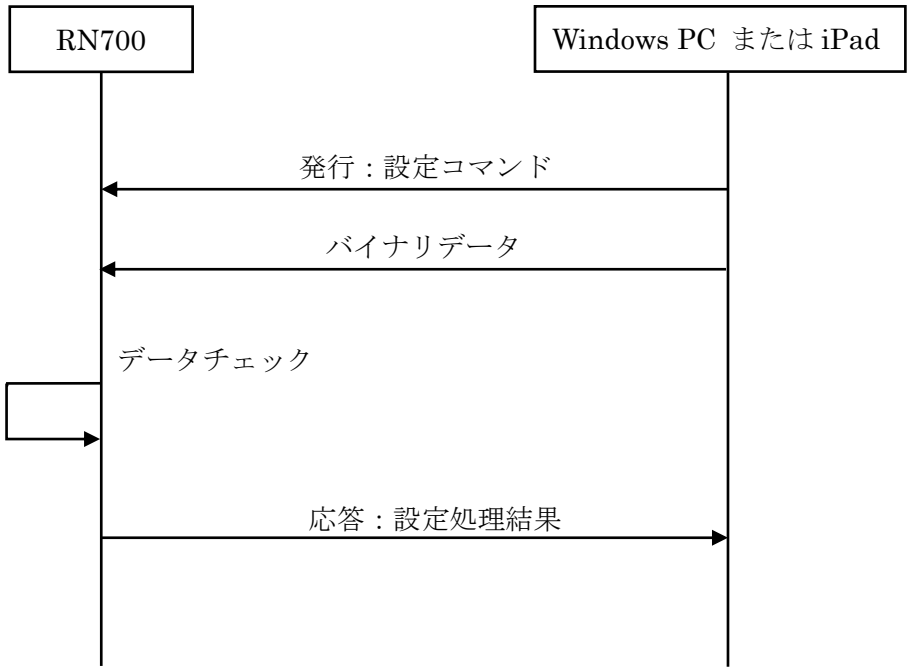
4.2 読み出しコマンドシーケンス(バイナリデータ送信時)



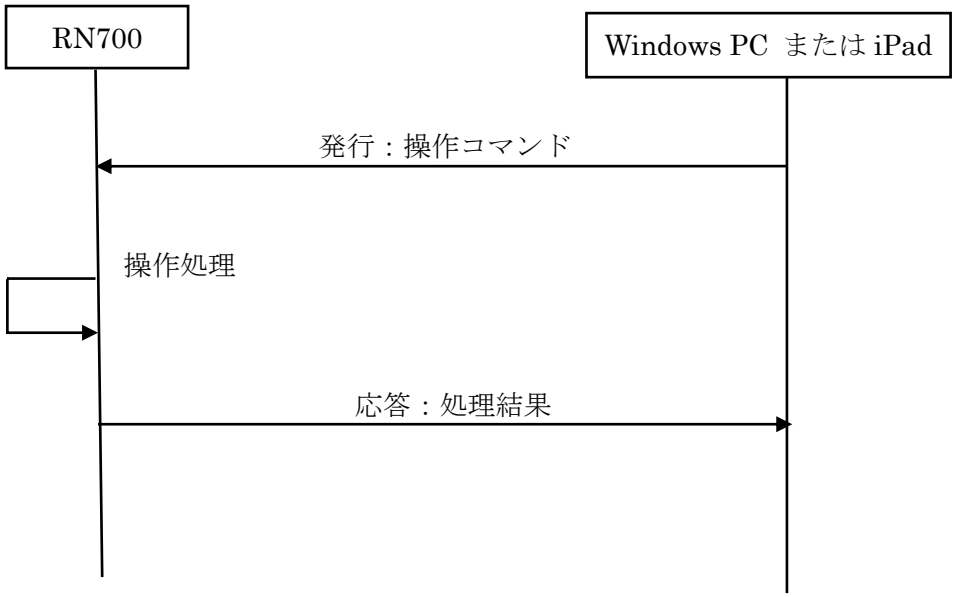
4.3 設定コマンドシーケンス



4.4 設定コマンドシーケンス(バイナリデータ送信時)

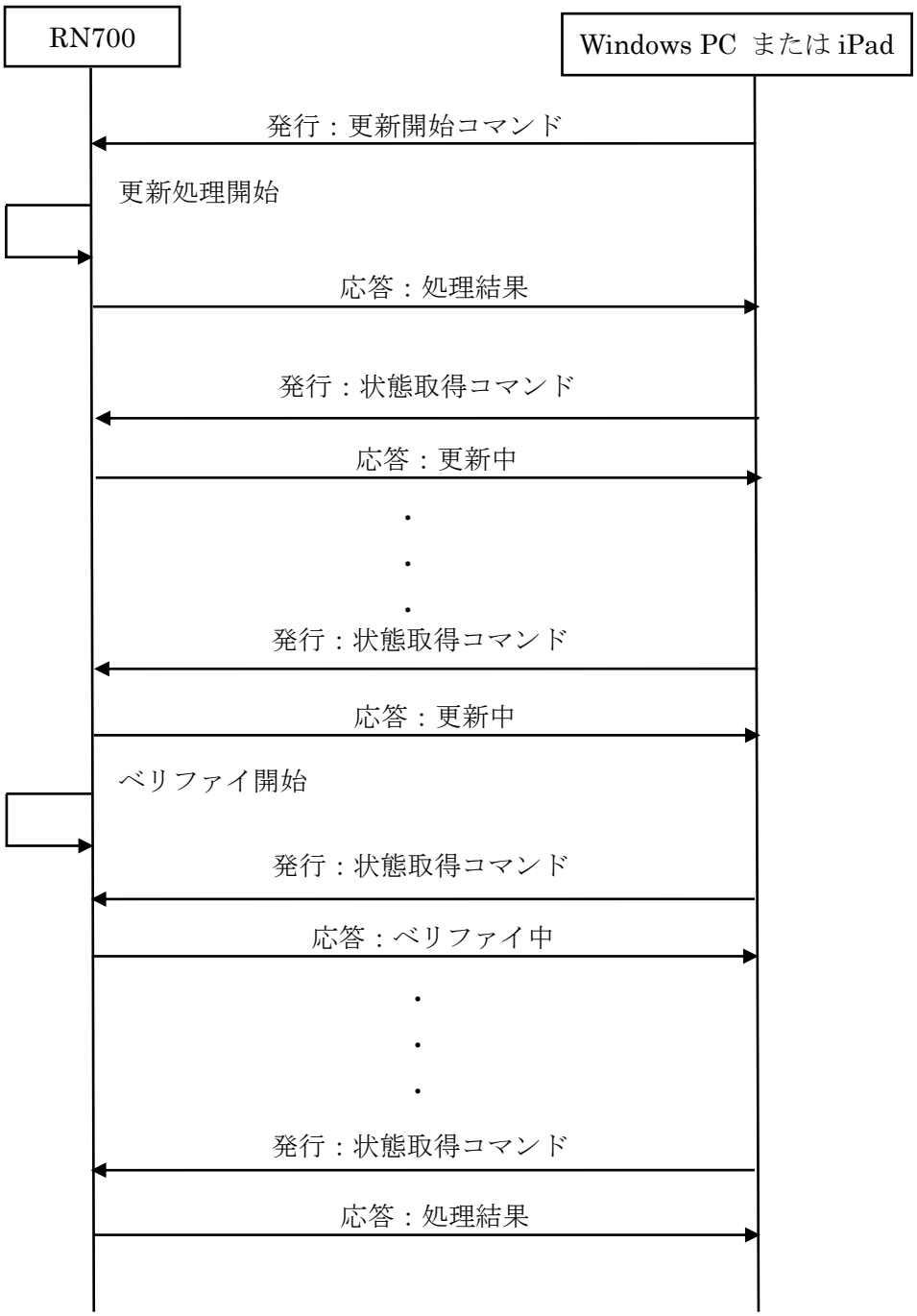


4.5 操作コマンドシーケンス



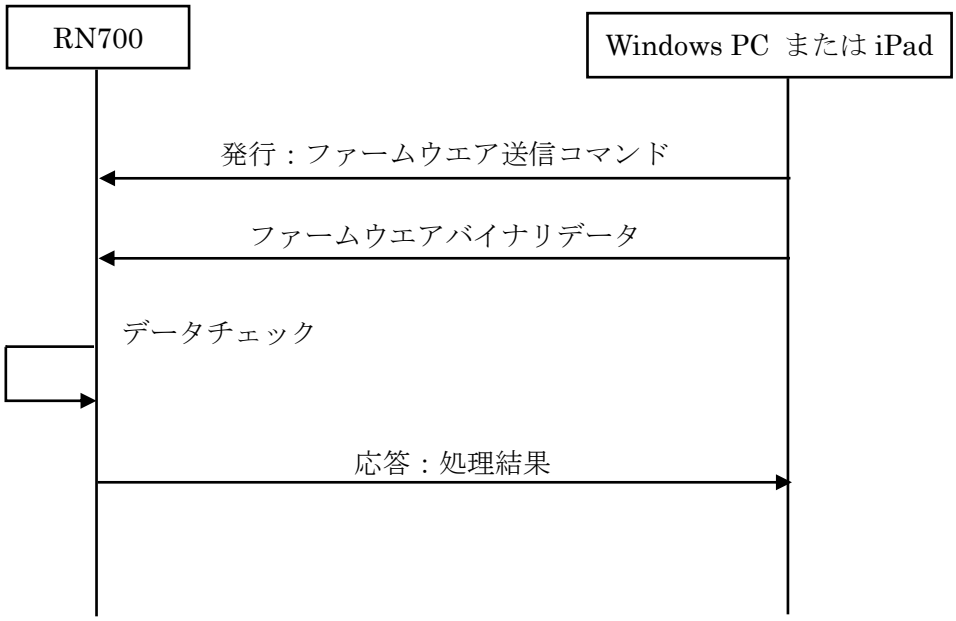
4.6 アップデートコマンドシーケンス

アップデートコマンドを使用し、ファームウエアを更新するシーケンスを記載します。



4.7 アップデートコマンドシーケンス(ファームウエア送信時)

ファームウエアを RN700 へ送信する際のシーケンスを記載します。



5 バイナリデータブロック転送

getBinaryFileEx / setBinaryFileEx コマンドを使用することでバイナリデータを小分けに転送（ブロック転送）を開始することができます。

ブロック転送中は、メッセージと応答の送受信を繰り返しバイナリデータを転送します。

※ getBinaryFileEx / setBinaryFileEx コマンドについては、”getBinaryFileEx” 項目及び ”setBinaryFileEx” 項目を参照。

5.1 メッセージ及び応答のデータ転送形式

ブロック転送時のメッセージ及び応答のデータ転送形式を記載します。各データの転送形式内のチェックサムは、ヘッダ、ブロック番号、データサイズ、データの全てをバイト単位で加算した値が格納されます。ブロック番号は、メッセージ毎に 0x00 ～ 0xFF までインクリメントし、0xFF の次は 0x00 に戻ります。（メッセージの転送完了後、インクリメントします）

5.1.1 データ送信メッセージ

バイナリデータを送信するメッセージです。

データサイズは次に続くデータのサイズを表し、データサイズの上限値は getBinaryFileEx / setBinaryFileEx コマンドのパラメータによって決まります。

ヘッダ	ブロック番号	データサイズ(byte)			データ	チェックサム			
STX(0x02)	0xXX	0xXX	0xXX	0xXX	0xXX ・・・	0xXX	0xXX	0xXX	0xXX

5.1.2 データ送信終了通知メッセージ

バイナリデータの送信が終了したことを示すメッセージです。

ヘッダ	ブロック番号	チェックサム			
EOT(0x04)	0xXX	0xXX	0xXX	0xXX	0xXX

5.1.3 データ送信中断通知メッセージ

中断時のブロック番号の値はチェックされません。

ヘッダ	ブロック番号	チェックサム			
CAN(0x18)	0xXX	0xXX	0xXX	0xXX	0xXX

5.1.4 応答

ブロック番号は対応するメッセージのブロック番号を格納します。

ACK 応答

ヘッダ	ブロック番号	チェックサム			
ACK(0x06)	0xXX	0xXX	0xXX	0xXX	0xXX

NAK 応答

ヘッダ	ブロック番号	チェックサム			
NAK(0x15)	0xXX	0xXX	0xXX	0xXX	0xXX

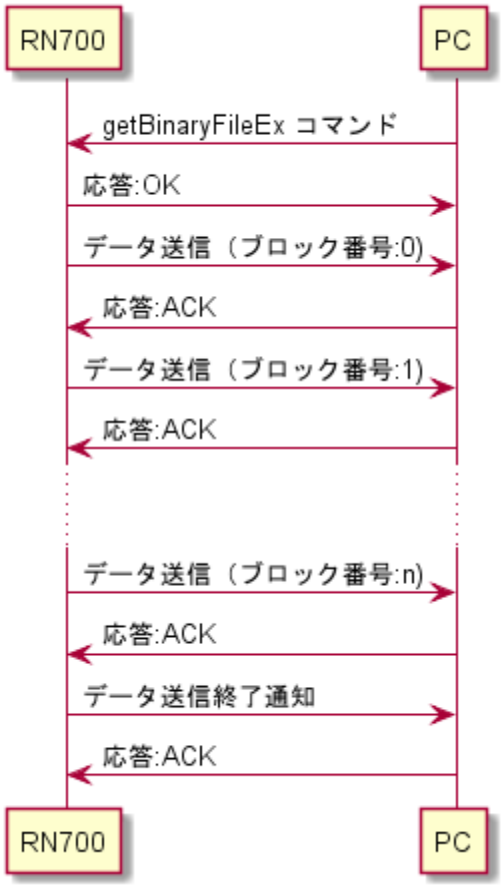
5.2 シーケンス

ブロック転送を行う場合のシーケンスを記載します。

図中の送信側、受信側はバイナリデータの送受信側を示しており、RN700 または PC のいづれかになります。

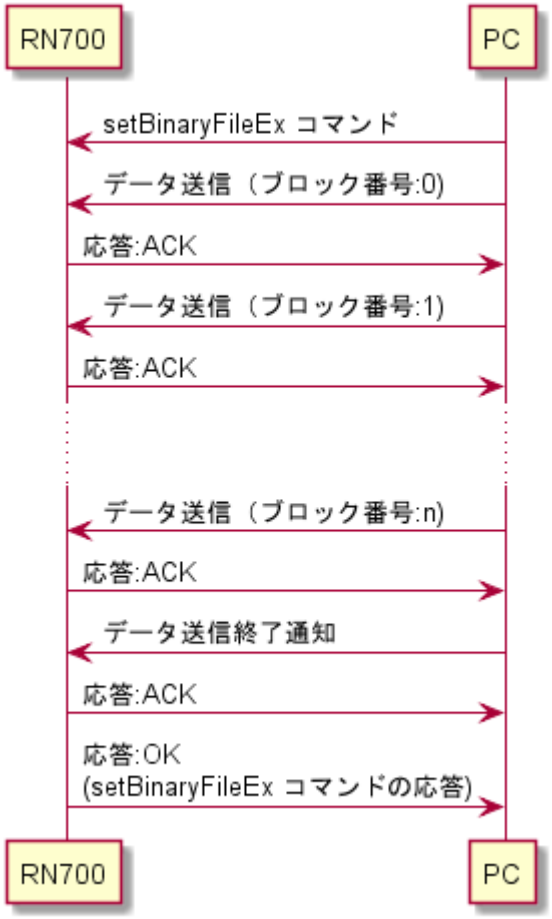
5.2.1 バイナリデータを RN700 から PC へ送信する場合

getBinaryFileEx コマンドを発行した場合、バイナリデータは RN700 から PC へ送信します。

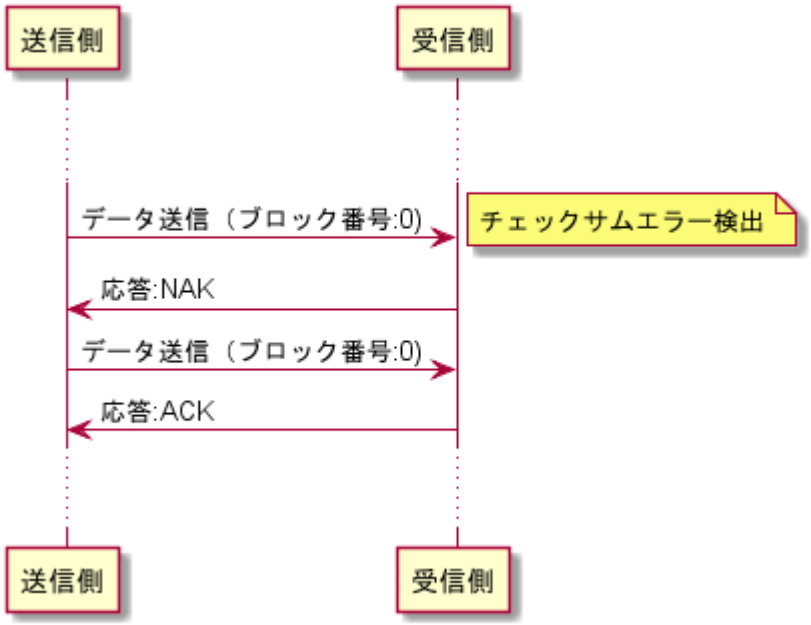


5.2.2 バイナリデータを PC から RN700 へ送信する場合

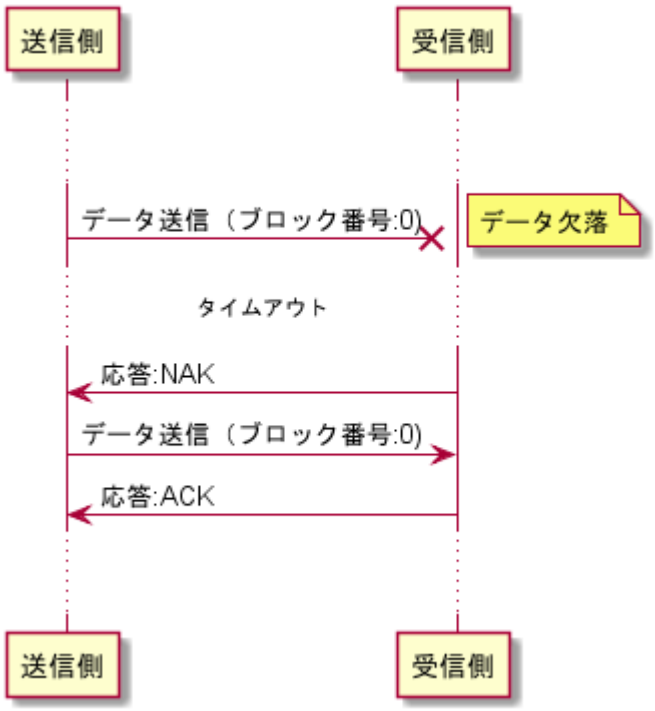
setBinaryFileEx コマンドを発行した場合、バイナリデータは PC から RN700 へ送信します。



5.2.3 送信データに異常があった場合



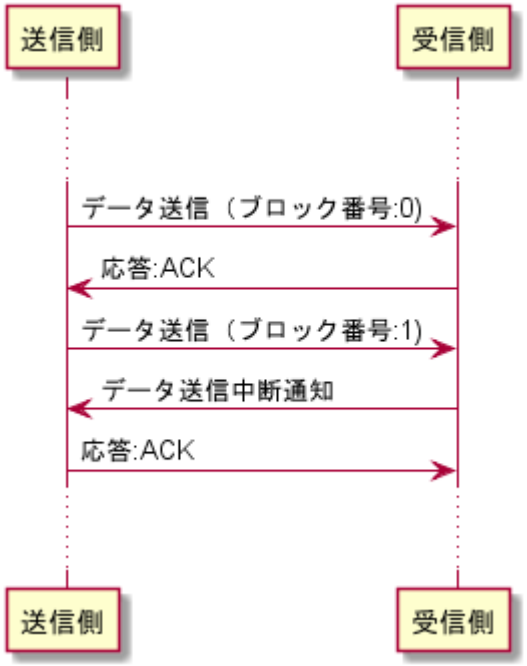
5.2.4 送信データが欠落した場合



5.2.5 送信データを中断する場合

データ送信中断通知は、送信側、受信側のどちらからでも送信可能です。

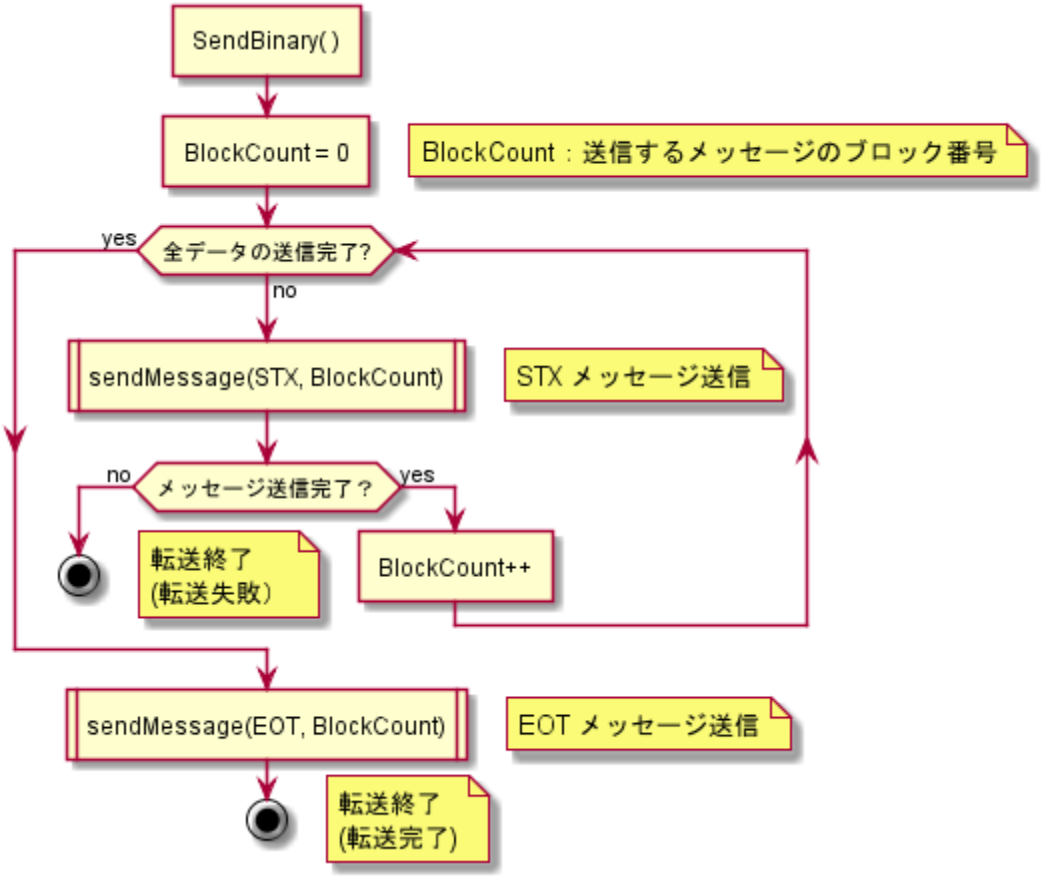
下記は受信側からデータ送信中断通知を送信した場合の例です。

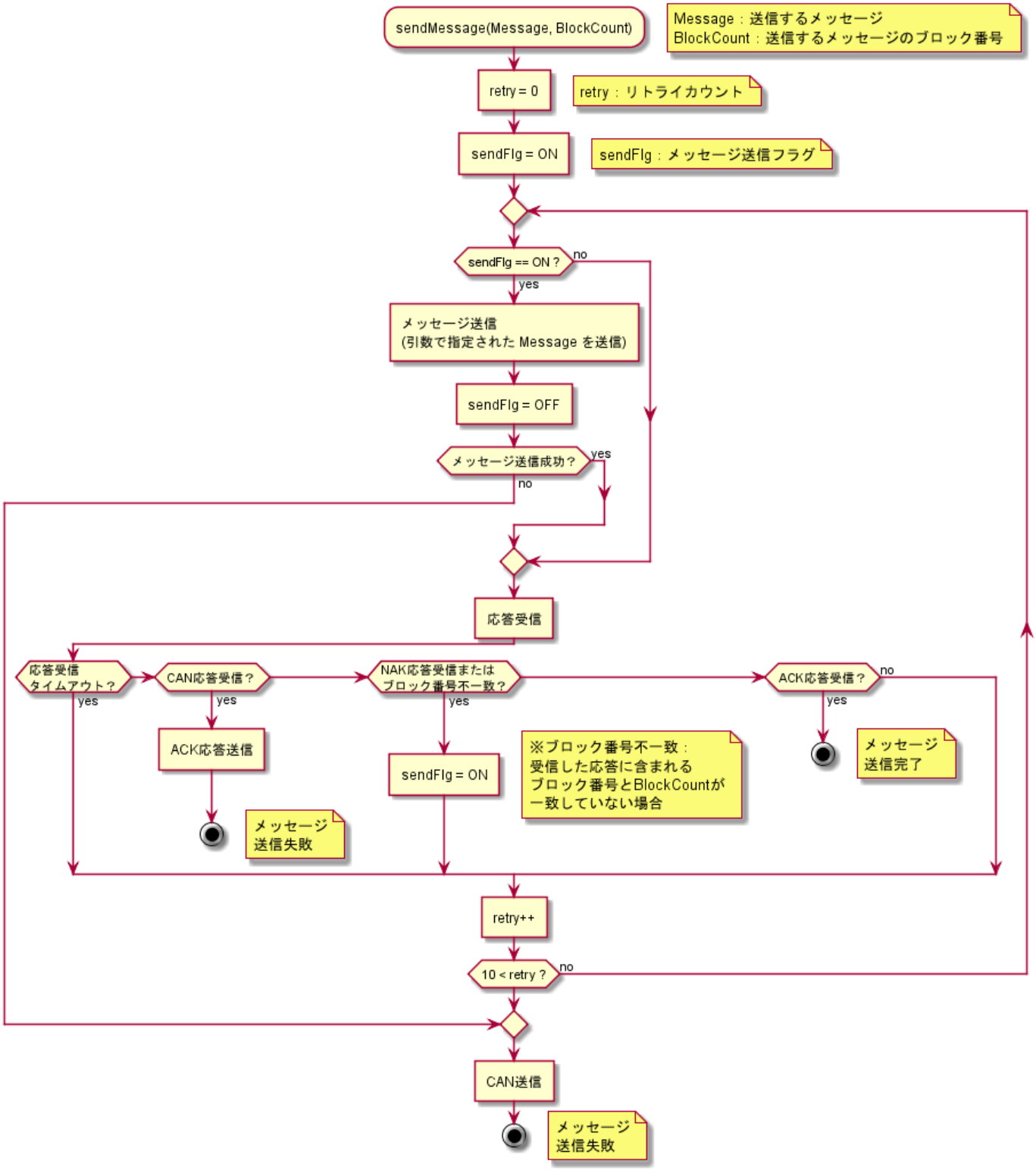


5.3 フロー

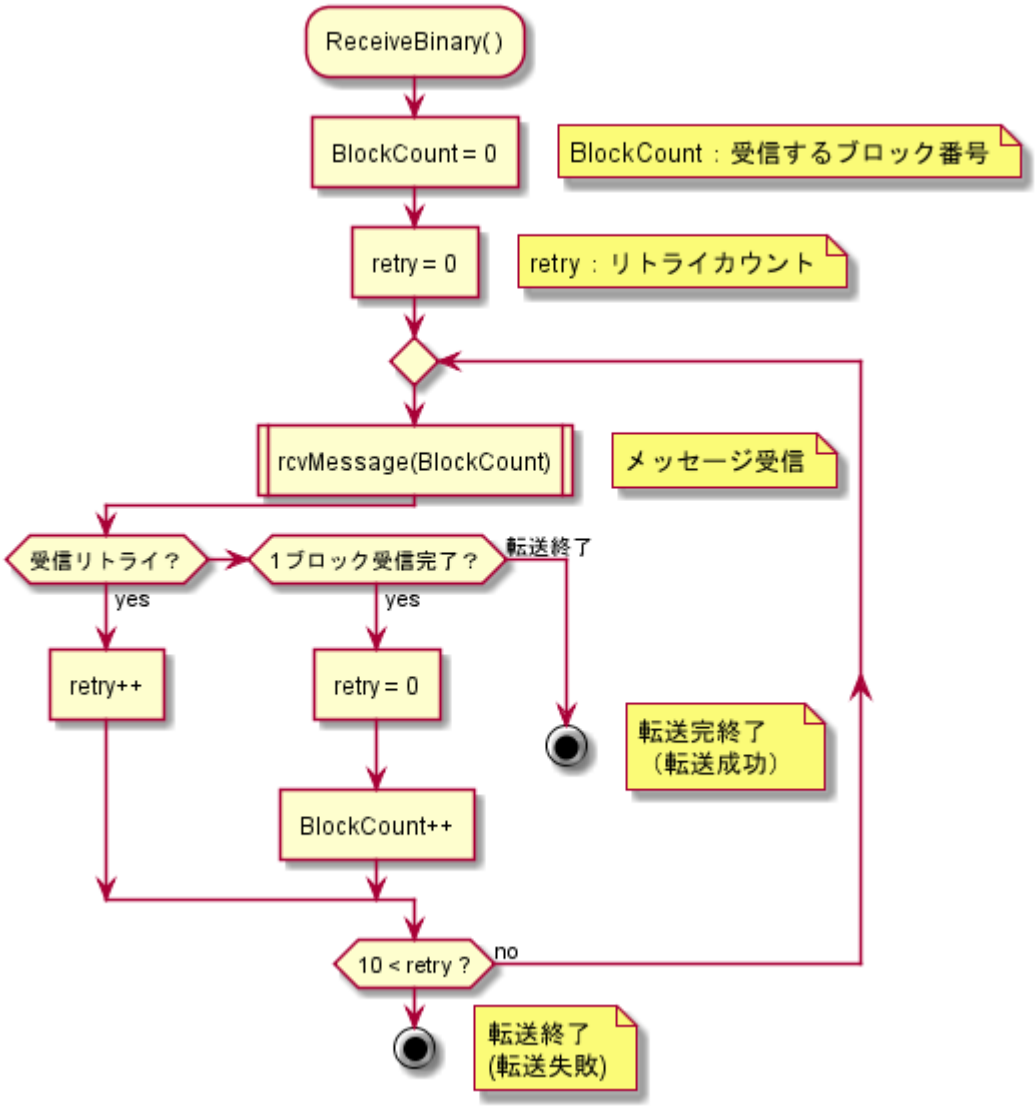
バイナリデータ送信処理、受信処理時のメッセージ及び応答の発行タイミングを示すフローを記載します。

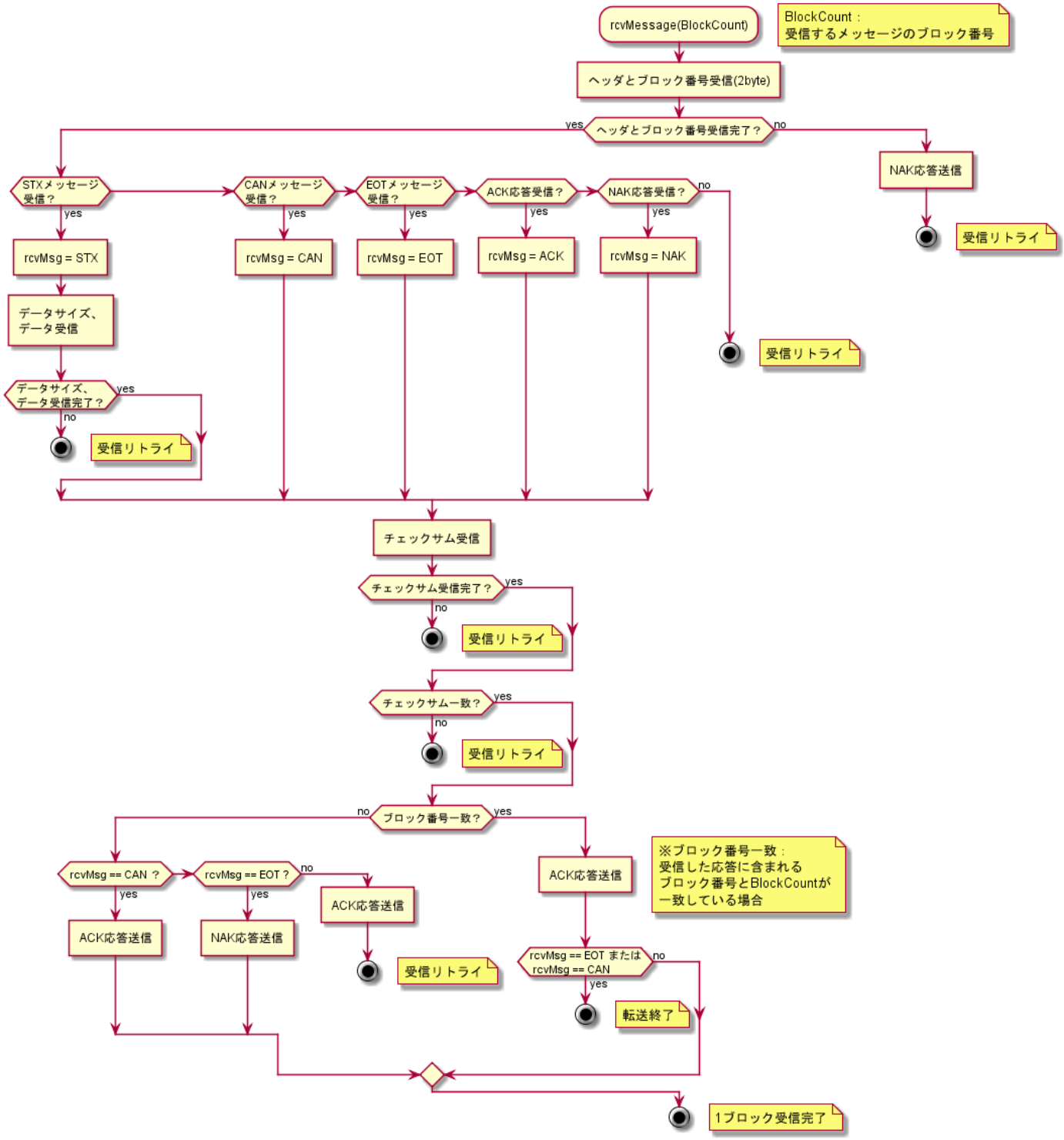
5.3.1 バイナリデータ送信処理




sendMessage(Message, BlockCount)
Message：送信するメッセージ
BlockCount：送信するメッセージのブロック番号
retry = 0
retry：リトライカウント
sendFlg = ON
sendFlg：メッセージ送信フラグ
sendFlg == ON ?
no
yes
メッセージ送信
(引数で指定された Message を送信)
sendFlg = OFF
メッセージ送信成功？
yes
no
応答受信
応答受信
タイムアウト？
yes
CAN応答受信？
yes
NAK応答受信または
ブロック番号不一致？
yes
ACK応答受信？
no
yes
ACK応答送信
メッセージ
送信失敗
sendFlg = ON
※ブロック番号不一致：
受信した応答に含まれる
ブロック番号とBlockCountが
一致していない場合
メッセージ
送信完了
retry++
10 < retry ?
no
CAN送信
メッセージ
送信失敗

5.3.2 バイナリデータ受信処理




rcvMessage(BlockCount)
BlockCount：
受信するメッセージのブロック番号
ヘッダとブロック番号受信(2byte)
ヘッダとブロック番号受信完了？
yes
no
NAK応答送信
受信リトライ
STXメッセージ受信？
CANメッセージ受信？
EOTメッセージ受信？
ACK応答受信？
NAK応答受信？
rcvMsg = STX
rcvMsg = CAN
rcvMsg = EOT
rcvMsg = ACK
rcvMsg = NAK
データサイズ、データ受信
データサイズ、データ受信完了？
受信リトライ
チェックサム受信
チェックサム受信完了？
受信リトライ
チェックサム一致？
受信リトライ
ブロック番号一致？
rcvMsg == CAN ？
rcvMsg == EOT ？
ACK応答送信
NAK応答送信
ACK応答送信
受信リトライ
ACK応答送信
※ブロック番号一致：
受信した応答に含まれる
ブロック番号とBlockCountが
一致している場合
rcvMsg == EOT または
rcvMsg == CAN
転送終了
1ブロック受信完了

6 コマンド実行可能範囲

各コマンドの実行可能な状態の一覧を記載します。コマンドは「○」が記載された状態でのみ実行可能です。実行不可能な場合は、下記のようなエラー応答が返ってきます。

```
{“error”: [103, “Command Executed”], “id”: 1}
```

また、状態については RN700FW 仕様書の「状態遷移」項目を参照。

6.1 読み出しコマンド

	状態								
コマンド名	起動時処理	初期動作確認	測定可能	測定	簡易調整	エラー発生	コマンド受付	デバッグコマンド処理	遷移待機
getApiList		○	○	○	○	○	○	○	
getVersion		○	○	○	○	○	○	○	
getStatus		○	○	○	○	○	○	○	
getOperatingStatus		○	○	○	○	○	○	○	○
getTemperature		○	○	○	○	○	○	○	
getIllumination		○	○	○	○	○	○	○	
getAcceleration		○	○	○	○	○	○	○	
getLimitSwitch		○	○	○	○	○	○	○	
getCameraShutterWidth		○	○	○	○	○	○	○	
getCameraGain		○	○	○	○	○	○	○	
getLed1RGB		○	○	○	○	○	○	○	
getLed2RGB		○	○	○	○	○	○	○	
getLcdPwm		○	○	○	○	○	○	○	
getLcdBitmap		○	○	○	○	○	○	○	
getImage		○	○	○	○	○	○	○	
getAnalysisResults		○	○	○	○	○	○	○	
getAnalysisType		○	○	○	○	○	○	○	
getAnalysisLevelVal		○	○	○	○	○	○	○	
getAnalysisLevel		○	○	○	○	○	○	○	
getSettingFile		○	○	○	○	○	○	○	
getTimeout		○	○	○	○	○	○	○	
getProfileData		○	○	○	○	○	○	○	
getCaptureImage		○	○	○	○	○	○	○	
getOperatingMode		○	○	○	○	○	○	○	
getDate		○	○	○	○	○	○	○	
getCalcType		○	○	○	○	○	○	○	
getClassType		○	○	○	○	○	○	○	
getOutputMode		○	○	○	○	○	○	○	
getBinaryFileEx		○	○	○	○	○	○	○	
getPrintPattern		○	○	○	○	○	○	○	

getLotNumber		○	○	○	○	○	○	○	
getSaveData		○	○	○	○	○	○	○	

6.2 設定コマンド

コマンド名	状態								
	起動時処理	初期動作確認	測定可能	測定	簡易調整	エラー発生	コマンド受付	デバッグコマンド処理	遷移待機
setCameraShutterWidth						○	○		
setCameraGain						○	○		
setLed1RGB						○	○		
setLed2RGB						○	○		
setLedPower						○	○		
setLcdPower						○	○		
setLcdPwm						○	○		
setLcdBitmap		○		○		○	○		
setAnalysisType			○			○	○		
setAnalysisLevel			○			○	○		
setSettingFile			○	○		○	○		
setDate			○			○	○		
setTimeout			○			○	○	○	
setOperatingMode			○			○	○		○
setUserInfo			○	○			○		
setBinaryFile		○	○	○	○	○	○	○	
setCalcType			○			○	○		
setClassType			○			○	○		
setOutputMode			○			○	○		
setBinaryFileEx		○	○	○	○	○	○	○	
setPrintPattern			○			○	○		
setLotNumber			○			○	○		
setSaveData			○			○	○		

6.3 操作コマンド

コマンド名	状態								
	起動時処理	初期動作確認	測定可能	測定	簡易調整	エラー発生	コマンド受付	デバッグコマンド処理	遷移待機
writeFlash			○			○	○		
readFlash			○			○	○		
writeRamData			○			○	○		
calibrateRGB			○						
initialCalibration			○						
captureImage						○	○		
capAnalysisImg						○	○		
deleteImage			○			○	○		
actImageProcessing						○	○		
startAnalysis			○						
startAnalysisAsync			○						
printResult			○			○	○		
dispLcdBmp						○	○		
dispLcdRGB						○	○		
dispOledText							○		
dispOledMsg							○		
soundBuzzer			○			○	○		
startAccMonitoring						○	○		
stopAccMonitoring						○	○		
readAccelerationReg						○	○		
writeAccelerationReg						○	○		
startCaptureLoop						○	○		
stopCaptureLoop						○	○		
rawToImage			○			○	○		
chkTray							○		

6.4 アップデートコマンド

コマンド名	状態								
	起動時処理	初期動作確認	測定可能	測定	簡易調整	エラー発生	コマンド受付	デバッグコマンド処理	遷移待機
setImgProcBoardStatus			○	○					
setImgProcRequestNotify			○	○					
setResultData			○	○					

6.5 Armadillo 840 – Armadillo 1500 通信用コマンド

コマンド名	状態								
	起動時処理	初期動作確認	測定可能	測定	簡易調整	エラー発生	コマンド受付	デバッグコマンド処理	遷移待機
writeFirmware							○		
sendFirmware							○		
getFirmwareStatus							○		

7 コマンド詳細

7.1 読み出しコマンド

7.1.1 getApiList

説明

コマンド一覧を取得します。

コマンド

{“method”: “getApiList”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [“Param1”, “Param2”, . . .], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1,2, …, n	String	コマンド一覧

例：{“result”: [“getApiList”, “getVersion”, “getStatus”, ...], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.2 getVersion

説明

バージョンを取得します。

コマンド

{“method”: “getVersion”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [“Param1”, “Param2”], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	ファームウエアバージョン
Param2	String	画像処理バージョン
Param3	String	RN-700 ボードタイプ(設定ファイルで指定したボードタイプ)

例：{“result”: [“1.0”, “1.0”, “0”], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.3 getStatus

説明

ステータスを取得します。

コマンド

{“method”: “getStatus”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: “Param1”, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	動作モード(16 進数文字列 “0000” ～ “FFFF”)

動作モードパラメータの詳細は下記の通りです。(T.B.D.)

ビット	内容	設定値	初期値
1 bit	ステータス値有効ビット 無効の場合、その他のステータスビットは無効な値です。	0：有効 1：無効	1
2 bit	カメラ温度センサ	0： 正常 1： 異常	0
3 bit	ボード温度センサ	0： 正常 1： 異常	0
4 bit	光量センサ	0： 正常 1： 異常	0
5 bit	加速度センサ	0： 正常 1： 異常	0
6 bit	トレイスイッチ(SW1)	0： OFF 1： ON	0
7 bit	トレイスイッチ(SW2)	0： OFF 1： ON	0
8 bit	入口シャッター(SW3)	0： OFF 1： ON	0

9 bit	リザーブ	-	0
10 bit	リザーブ	-	0
11 bit	リザーブ	-	0
12 bit	リザーブ	-	0
13 bit	リザーブ	-	0
14 bit	リザーブ	-	0
15 bit	リザーブ	-	0
16 bit	リザーブ	-	0

例：{“result”: “0001”, “id”: 1}

エラー

コマンド応答形式を参照。

7.1.4 getOperatingStatus

説明

動作状態を取得します。

コマンド

{“method”: “getOperatingStatus”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2] “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	動作状態 0 ：測定可能 1 ：初回動作確認 1 2 ：初回動作確認 2,3,4,5 3 ：測定可能状態 4 ：測定 5 ：簡易調整 6 ：エラー発生状態 7 ：コマンド受付 8 :メニュー表示 9 ：遷移待機 90：デバッグコマンド処理
Param2	int	詳細状態（T.B.D.）

例：{“result”: [0,1], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.5 getTemperature

説明

カメラ温度及びボード温度を取得します。

コマンド

{“method”: “a”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	double	カメラ温度(-55 ～ 125) 単位：℃
Param2	double	ボード温度(-55 ～ 125) 単位：℃

例：{“result”: [30, 25], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.6 getIllumination

説明

カラーセンサで計測した光量を取得します。

コマンド

{“method”: “getIllumination”, “params”: Params1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	ゲート時間(ms) (0 ～ 999)

例：{“method”: “getIllumination”, “params”: 20, “id”: 1}

応答

正常

{“result”: [Param1, Param2, Param3], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	Red 光量　(0 ～ 4095)
Param2	int	Green 光量 (0 ～ 4095)
Param3	int	Blue 光量　(0 ～ 4095)

例：{“result”: [4095, 4095. 4095], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.7 getAcceleration

説明

加速度を取得します。

コマンド

{“method”: “getAcceleration”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2, Param3], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	double	x 軸加速度（-2 ～ 2)
Param2	double	y 軸加速度（-2 ～ 2)
Param3	double	z 軸加速度（-2 ～ 2)

例：{“result”: [1,0,0], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.8 getLimitSwitch

説明

トレイスイッチ(SW1. SW2) 及び 入口シャッター(SW3) のリミットスイッチの状態を取得します。

コマンド

{“method”: “getLimitSwitch”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	トレイスイッチ SW1 (0：OFF 1：ON)
Param2	int	トレイスイッチ SW2 (0：OFF 1：ON)
Param3	int	入口シャッター SW3 (0：OFF 1：ON)

例：{“result”: [0, 1, 0], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.9 getCameraShutterWidth

説明

カメラの Shutter Width を取得します。

コマンド

{“method”: “getCameraShutterWidth”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	Shutter Width Upper (0 ～ 65535)
Param2	int	Shutter Width Lower (0 ～ 65535)

例：{“result”: [0, 1943], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.10 getCameraGain

説明

カメラの Gain を取得します。取得項目で指定した項目の Gain 値を取得します。

コマンド

{“method”: “getCameraGain”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	取得項目(Green1, Blue, Red, Green2)

例：{“method”: “getCameraGain”, “params”: “Green1”, “id”: 1}

応答

正常

{“result”: [Param1, Param2, Param3], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	Digital Gain (0 ～ 120)
Param2	int	Analog Multiplier (0 ～ 1)
Param3	int	Analog Gain (8 ～ 63)

例: {“result”: [0, 1, 8], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.11 getLed1RGB

説明

LED1 Red、LED1 Green、LED1 Blue を取得します。

コマンド

{“method”: “getLed1RGB”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2, Param3], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	LED1 Red (0 ～ 4095)
Param2	int	LED1 Green (0 ～ 4095)
Param3	int	LED1 Blue (0 ～ 4095)

例：{“result”: [0, 0, 0], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.12 getLed2RGB

説明

LED2 Red、LED2 Green、LED2 Blue を取得します。

コマンド

{“method”: “getLed2RGB”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2, Param3], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	LED2 Red (0 ～ 4095)
Param2	int	LED2 Green (0 ～ 4095)
Param3	int	LED2 Blue (0 ～ 4095)

例：{“result”: [0, 0, 0], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.13 getLcdPwm

説明

LCD PWM を取得します。

コマンド

{“method”: “getLcdPwm”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	LCD PWM (0 ～ 255)

例：{“result”: 0, “id”: 1}

エラー

コマンド応答形式を参照。

7.1.14 getLcdBitmap

説明

LCD 表示ビットマップを取得します。

コマンド

{“method”: “getLcdBitmap”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、LCD 表示ビットマップデータ(バイナリデータ)を送信する。

エラー

コマンド応答形式を参照。

7.1.15 getImage

説明

パラメーターで指定した画像を取得します。

コマンド

{“method”: “getImage”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	画像 ID

例：{“method”: “getImage”, “params”: “D20140507105640.jpg”, “id”: 1}

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、指定された画像データ(バイナリデータ)を送信する。

エラー

コマンド応答形式を参照。

7.1.16 getAnalysisResults

説明

測定結果を取得します。

コマンド

{“method”: “getAnalysisResults”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	測定結果 ID

例：{“method”: “getAnalysisResults”, “params”: “result.csv”, “id”: 1}

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、指定された測定結果データを送信する。

エラー

コマンド応答形式を参照。

7.1.17 getAnalysisType

説明

設定されている測定粒種 ID 及び測定粒種名を取得します。

コマンド

{“method”: “getAnalysisType”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, “Param2”], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	測定粒種 ID
Param2	String	測定粒種名

例：{“result”: [1, “うるち短粒種玄米”], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.18 getAnalysisLevelVal

説明

指定した選別レベル ID の選別レベル名を取得します。

コマンド

{“method”: “getAnalysisLevelVal”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型（json_type）	内容
Param1	int	選別レベル ID

例：{“method”: “getAnalysisLevelVal”, “params”: 1, “id”: 1}

応答

正常

{“result”: [“Param1”], “id”: id 番号}

項目	JSON の型（json_type）	内容
Param1	String	選別レベル名

例：{“result”: [“選別レベル 1”], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.19 getAnalysisLevel

説明

選択されている選別レベル ID 及び選別レベル名を取得します。

コマンド

{“method”: “getAnalysisLevel”, “params”: [Param1, “Param2”] , “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, “Param2”], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	選別レベル ID
Param2	String	選別レベル名

例：{“result”: [0, “選別レベル 1 ”], “id”: 1}

エラー

コマンド応答形式を参照。

7.1.20 getSettingFile

説明

RN700 の /tmp/config フォルダ配下から指定したファイル名の設定ファイルを取得します。

コマンド

{“method”: “getSettingFile”, “params”: “Param1” , “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	設定ファイル名

例：{“method”: “getSettingFile”, “params”: “rn700.conf”, “id”: 1}

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、指定された設定ファイルデータ(バイナリデータ)を送信する。

エラー

コマンド応答形式を参照。

7.1.21 getTimeout

説明

通信のタイムアウト時間を取得します。

コマンド

{“method”: “getTimeout”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	タイムアウト時間(秒) (1 ～ 10)

例：{“result”: 1, “id”: 1}

7.1.22 getProfileData

説明

デバッグ用コマンド。RN-700 の /tmp/ フォルダ配下に保存されているファイルを取得します。

コマンド

{“method”: “getProfileData”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ファイル名

例：{“method”: “getProfileData”, “params”: “accLog.csv”, “id”: 1}

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、指定されたファイルデータを送信する。

エラー

コマンド応答形式を参照。

7.1.23 getCaptureImage

説明

startCaptureLoop コマンドで連続キャプチャしている raw 画像データを取得します。画像は連続キャプチャ中のみに取得可能です。また、取得できる画像データがない場合はエラーとなります。

コマンド

{“method”: “getCaptureImage”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: “binary”, “id”: id 番号}

上記応答に続き、指定された画像データ(バイナリデータ)を送信する。

エラー

コマンド応答形式を参照。

7.1.24 getOperatingMode

説明

動作モードを取得します。

コマンド

{“method”: “getOperatingMode”, “params”:[], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: “Param1”, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	動作モード(16 進数文字列 “00” ～ “FF”)

動作モードパラメータの詳細は下記の通りです。

ビット	内容	設定値	初期値
1 bit	デバッグモード ON/OFF 切替。ON の場合、下記のファイルを出力します。 ・画像処理時の中間ファイル ・画像撮影時の BIN, BMP, JPG 画像の保存	0： OFF 1： ON	0
2 bit	トレイ検出モード ON/OFF 切替。OFF の場合、トレイ挿入時に測定を開始しません、	0： OFF 1： ON	1
3 bit	コマンドモード ON/OFF 切替。ON の場合、測定可能状態からコマンド受付状態に遷移します。コマンド受付状態時は、状態遷移するコマンドの使用はできません。	0： OFF 1： ON	0
4 bit	リザーブ	-	0
5 bit	リザーブ	-	0
6 bit	リザーブ	-	0
7 bit	リザーブ	-	0
8 bit	リザーブ	-	0

例：{“result”: “02”, “id”: 1}

エラー

コマンド応答形式を参照。

7.1.25 getDate

説明

日時を取得します。

コマンド

{“method”: “getDate”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String YYYYMMDDhhmm.ss	設定する日時 YYYY：年 MM　：月 DD　：日 hh　：時間（24 時間形式） mm　：分 .　：固定文字 ss　：秒

例：{“result”: “201405150910.00”, “id”: 1}

7.1.26 getCalcType

説明

測定結果の計算方法設定を取得します。

コマンド

{“method”: “getCalcType”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	0：粒数％ 1：質量換算％

例：{“result”: 1, “id”: 1}

7.1.27 getClassType

説明

測定結果の分類方法設定を取得します。

コマンド

{“method”: “getClassType”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	0：詳細分類 1：標準分類

例：{“result”: 1, “id”: 1}

7.1.28 getOutputMode

説明

測定結果出力方法の設定を取得します。

コマンド

{“method”: “getOutputMode”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	0：通常 1：「異物」「その他」項目を出力しない

例：{“result”: 1, “id”: 1}

7.1.29 getBinaryFileEx

説明

RN-700 の /tmp/ フォルダ配下に保存されているファイルを取得します。ブロック転送でバイナリデータを転送します。

コマンド

{“method”: “getBinaryFileEx”, “params”: [“Param1”, “Param2” , “Param3”] “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ファイル名
Param2	int	ブロックサイズ(byte)
Param3	int	タイムアウト時間(ms)

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

例：{“result”: 1, “id”: 1}

7.1.30 getPrintPattern

説明

印字設定を取得します。

コマンド

{“method”: “getPrintPattern”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	0：文字大・粒数有 1：文字大・粒数無 2：文字小・粒数有 3：印字しない

エラー

コマンド応答形式を参照。

例：{“result”: 1, “id”: 1}

7.1.31 getLotNumber

説明

ロット番号を取得します。

コマンド

{“method”: “getLotNumber”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	ロット番号(0 ～ 99999999)
Param2	int	ロット番号自動増加 0:自動増加しない 1:自動増加する

エラー

コマンド応答形式を参照。

例：{“result”: [100, 1], “id”: 1}

7.1.32 getSaveData

説明

データ保存設定の取得をします。

コマンド

{“method”: “getSaveData”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [Param1, Param2, Param3, Param4], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	測定結果（0：保存しない 1：保存する）
Param2	int	画像・大（0：保存しない 1：保存する）
Param3	int	画像・小（0：保存しない 1：保存する）
Param4	int	結果一覧（0：保存しない 1：保存する）

エラー

コマンド応答形式を参照。

例：{“result”: [1, 1, 0, 0], “id”: 1}

7.2 設定コマンド

7.2.1 setCameraShutterWidth

説明

カメラの Shutter Width を設定します。

コマンド

{“method”: “setCameraShutterWidth”, “params”: [Param1, Param2], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	Shutter Width Upper (0 ～ 15)
Param2	int	Shutter Width Lower (1 ～ 65535)

例：{“method”: “setCameraShutterWidth”, “params”: [0, 1943], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.2 setCameraGain

説明

カメラの Gain を設定します。

設定項目で指定した項目の Gain 値を設定します。設定項目に Global を指定した場合は、Green1, Blue, Red, Green2 の全てに対して Gain 値が設定されます。

コマンド

{“method”: “setCameraGain”, “params”: [“Param1”, Param2, Param3, Param4], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	設定項目(Global, Green1, Blue, Red, Green2)
Param2	int	Digital Gain (0 ～ 120)
Param3	int	Analog Multiplier (0 ～ 1)
Param4	int	Analog Gain (8 ～ 63)

例：{“method”: “setCameraGlobalGain”, “params”: [“Global”, 0, 1, 8], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.3 setLed1RGB

説明

LED1 Red、LED1 Green、LED1 Blue を設定します。

コマンド

{“method”: “setLed1RGB”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	LED1 Red　(0 ～ 4095)
Param2	int	LED1 Green (0 ～ 4095)
Param3	int	LED1 Blue　(0 ～ 4095)

例：{“method”: “setLed1RGB”, “params”: [0, 0, 0], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.4 setLed2RGB

説明

LED2 Red、LED2 Green、LED2 Blue を設定します。

コマンド

{“method”: “setLed2RGB”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	LED2 Red　(0 ～ 4095)
Param2	int	LED2 Green (0 ～ 4095)
Param3	int	LEd2 Blue　(0 ～ 4095)

例：{“method”: “setLed2RGB”, “params”: [0, 0, 0], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.5 setLedPower

説明

LED1 及び LED2 の点灯設定を行います。

コマンド

{“method”: “setLedPower”, “params”: [“Param1”, “Param2”], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ON：点灯　OFF：消灯
Param2	String	ON：点灯　OFF：消灯

例：{“method”: “setLedPower”, “params”: [“ON”, “OFF”], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.6 setLcdPower

説明

LCD の電源を設定します。

コマンド

{“method”: “setLcdPower”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ON：電源オン　OFF：電源オフ

例：{“method”: “setLcdPower”, “params”: “ON”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.7 setLcdPwm

説明

LCD PWM を設定します。

コマンド

{“method”: “setLcdPwm”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	LCD PWM (0 ～ 255)

例：{“method”: “setLcdPwm”, “params”: 0, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.8 setLcdBitmap

説明

LCD 表示ビットマップを設定します。

コマンド

{“method”: “setLcdBitmap”, “params”: [], “id”: id 番号}

上記コマンドに続いて、LCD 表示ビットマップ画像データ(バイナリデータ)を送信します。

パラメーター

なし

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.9 setAnalysisType

説明

測定粒種を設定します。測定粒種は、測定粒種 ID を指定し設定をします。

指定された測定粒種 ID が測定粒種ファイルに存在しないまたは、

指定された測定粒種 ID の選別レベルファイル(RN700_Parameter_xxxx_00.csv)のパラメーターが全て 0 の場合はエラー応答を返します。

コマンド

{“method”: “setAnalysisType”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	測定粒種 ID

例：{“method”: “setAnalysisType”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.10 setAnalysisLevel

説明

使用する選別レベルを選択します。

コマンド

{“method”: “setAnalysisLevel”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	選別レベル ID

例：{“method”: “setAnalysisLevel”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.11 setSettingFile

説明

指定したファイル名で設定ファイルを RN700 の /tmp/config フォルダ配下に保存します。既に同名のファイルがあった場合は上書きします。

コマンド

{“method”: “setSettingFile”, “params”: “Param1”, “id”: id 番号}

上記コマンドに続いて、設定ファイルデータ(バイナリデータ)を送信します。

パラメーター

項目	JSON の型 (json_type)	内容
Param1	Strings	設定ファイル名

例：{“method”: “setSettingFile”, “params”: “rn700.conf”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.12 setDate

説明

日時を設定する。

コマンド

{“method”: “setDate”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String YYYYMMDDhhmm.ss	設定する日時 YYYY：年 MM　：月 DD　：日 hh　：時間（24 時間形式） mm　：分 .　：固定文字 ss　：秒

例：{“method”: “setDate”, “params”: “201405150910.00”, “id”: 1}

応答

正常

{“result”: “Param1”, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	YYYYMMDDhhmm.ss	設定された日時。値のフォーマットは上記パラメーターと同様となります。

例：{“result”: “201405150910.00”, “id”: 1}

エラー

コマンド応答形式を参照。

7.2.13 setTimeout

説明

通信のタイムアウト時間を設定します。

コマンド

{“method”: “setTimeout”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	タイムアウト時間(秒) (1 ～ 10)

例：{“method”: “setTimeout”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.14 setOperatingMode

説明

動作モードを設定します。設定値は RN-700 の電源 ON 時に初期値に戻ります。設定値の初期値及び内容の詳細は getOperatingMode コマンド項目に記載してあります。

コマンド

{“method”: “setOperatingMode”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	動作モード (16 進数文字列 “00” ～ “FF”)

例：{“method”: “setOperatingMode”, “params”: “02”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.15 setUserInfo

説明

RN-700 が測定状態時に本コマンドを送信することでユーザー情報を設定できます。設定したユーザー情報は測定開始時にクリアされます。

コマンド

{“method”: “setUserInfo”, “params”: [“Param1”, “Param2” , “Param3”], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	生産者コード（最大全角 16 文字）
Param2	String	生産者名　　（最大全角 16 文字）
Param3	String	品種名　　（最大全角 16 文字）

例：{“method”: “setUserInfo”, “params”: [“０１２３”, “なまえ” , “ひんしゅ”], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.16 setBinaryFile

説明

バイナリファイルを /tmp/ 配下に保存します。

コマンド

{“method”: “setBinaryFile”, “params”: “Param1”, “id”: id 番号}

上記コマンドに続いて、バイナリデータを送信します。

パラメーター

項目	JSON の型 (json_type)	内容
Param1	Strings	バイナリファイル名

例：{“method”: “setBinaryFile”, “params”: “capImg/00.jpg”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.17 setCalcType

説明

測定結果の計算方法を設定します。

コマンド

{“method”: “setCalcType”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	0：粒数％ 1：質量換算％

例：{“method”: “setCalcType”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.18 setClassType

説明

測定結果の分類方法を設定します。

コマンド

{“method”: “setClassType”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	0：詳細分類 1：標準分類

例：{“method”: “setClassType”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.19 setOutputMode

説明

測定結果出力方法の設定をします。

コマンド

{“method”: “setOutputMode”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	0：通常 1：「異物」「その他」項目を出力しない

例：{“method”: “setOutputMode”, “params”: 1, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.2.20 setBinaryFileEx

説明

バイナリファイルを /tmp/ 配下に保存します。ブロック転送でバイナリデータを転送します。

コマンド

{“method”: “setBinaryFileEx”, “params”: [“Param1”, “Param2” , “Param3”] “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ファイル名
Param2	int	ブロックサイズ(byte)
Param3	int	タイムアウト時間(ms)

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

例：{“method”: “setBinaryFileEx”, “params”: [“test.bin”, “1024” , “1000”] “id”: 1 }

7.2.21 setPrintPattern

説明

印字設定を設定します。

コマンド

{“method”: “setPrintPattern”, “params”:Param1 “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	0：文字大・粒数有 1：文字大・粒数無 2：文字小・粒数有 3：印字しない

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

例：{“method”: “setPrintPattern”, “params”:0 “id”: 1}

7.2.22 setLotNumber

説明

ロット番号の設定をします。

コマンド

{“method”: “setLotNumber”, “params”:[Param1, Param2] “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	ロット番号(0 ～ 99999999)
Param2	int	ロット番号自動増加 0:自動増加しない 1:自動増加する

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

例：{“method”: “setLotNumber”, “params”: [10, 1] “id”: 1}

7.2.23 setSaveData

説明

データ保存の設定をします。

コマンド

{“method”: “setSaveData”, “params”:[Param1, Param2, Param3, Param4] “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	測定結果（0：保存しない 1：保存する）
Param2	int	画像・大（0：保存しない 1：保存する）
Param3	int	画像・小（0：保存しない 1：保存する）
Param4	int	結果一覧（0：保存しない 1：保存する）

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

例：{“method”: “setSaveData”, “params”: [1, 1, 0, 0] “id”: 1}

7.3 操作コマンド

7.3.1 writeFlash

説明

本体 /tmp/ フォルダ内の下記設定ファイルを外部 Flash（RN700 アプリが SD ブートの場合は SD カード）に書込みます。

ファイル種別	ファイル名
測定粒種ファイル	analysis.conf
halcon ライセンスファイル	license.dat
機器共通設置ファイル	comm.conf
測定粒種別設定ファイル	analysisType_[測定粒種 ID]/内のファイル

コマンド

```
{“method”: “writeFlash”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.2 readFlash

説明

外部 Flash （RN700 アプリが SD ブートの場合は SD カード）から設定ファイルを本体 /tmp/ フォルダ内にコピーし、設定値を読込みます。

対象の設定ファイルは writeFlash コマンド時に保存されるファイル種別と同じです。

コマンド

```
{“method”: “readFlash”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.3 writeRamData

説明

本体 /tmp/ フォルダ内の Ram 上の下記ファイルに現在の設定データを書き出します。

rn700.conf
comm.conf
machine.conf

コマンド

```
{“method”: “writeRamData”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.4 illuminationTest

説明

デバッグ用コマンド。応答メッセージは試験実行前に返します。LED 及び LCD の 各 RGB 値を変化させ、そのときのカラーセンサの値をファイルに保存します。保存ファイルは RN-700 内の /tmp/ 配下にパラメータで指定されたファイル名で保存します。保存したファイルは getProfileData　コマンドで取得できます。
LED 及び LCD の点灯時の初期値、変動幅は設定ファイル(/tmp/config/calibSetting.ini) に記述します。本コマンドを実行前に設定ファイルを setSettingFile コマンドで RN-700 に送信する必要があります。

コマンド

{“method”: “illuminationTest”, “params”: [“Param1”, “Param2”, Param3, Param4, Param5], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	測定結果保存ファイル名
Param2	int	測定回数(1 ～ 4095)
Param3	int	点灯後待ち時間（ms）(0 ～ 3600000)
Param4	int	消灯後待ち時間（ms）(0 ～ 3600000)
Param5	int	カラーセンサのゲート時間(ms) (0 ～ 999)

例：{“method”: “illuminationTest”, “params”: [“LED_Right”, 10, 200, 200,10], “id”: 1}

設定ファイルのフォーマット
[デバイス名]
init_R=(初期値:赤色)
init_G=(初期値:緑色)
init_B=(初期値:青色)
range_R=(変動幅:赤色)
range_G=(変動幅:緑色)
range_B=(変動幅:青色)

例：
[LED_Right]
init_R=1000
init_G=0
init_B=0
range_R=10
range_G=0
range_B=0

※設定ファイルには全てのデバイス(LED_Right, LED_Left, LCD) に対しての定義が必要になります。

※初期値 + 変動幅値 *（測定回数 – 1）が初期値の指定可能範囲を超えている場合はエラーになります。

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.5 initialCalibration

説明

初期調整２の状態へ遷移し、歪み補正、LED 光量補正及び LCD 光量補正を行います。

コマンド

{“method”: “initialCalibration”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.6 captureImage

説明

画像を撮影します。保存するファイル名は ”00.bin” で既に同じファイルがあれば上書きします。また setOperatingMode コマンドでデバッグモードを ON にすることで下記命名規則で撮影した画像を保存することが可能です。

デバッグモード時の保存ファイル名形式

DYYYYMMDDHHmmSS

例： D20140507105640.bin

項目	説明
D	プレフィックス
YYYY	年
MM	月
DD	日
HH	時（24 時間形式）
mm	分
SS	秒

既に同名のファイルが存在していた場合は、下記命名規則に
従います。

DYYYYMMDDHHmmSS-n

例： D20140507105640-1.bin

項目	説明
n	1,2,3, … 255 の値。全てのパターンのファイル名が存在する場合はエラーとなる。

コマンド

```
{“method”: “captureImage”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: [“Param1”, “Param2”, “Param3”], “id”: id 番号}
```

項目	値	内容
Param1	String	RAW データ ID
Param2	String	24bit BMP 画像 ID
Param3	String	JPEG 画像 ID

※ BMP 画像及び JPEG 画像を出力しない場合の値は “-“ となります。

例：

```
{“result”: [“D20140507105640.bin”, “D20140507105640.bmp”, “D20140507105640.jpg”],
“id”: 1}
```

エラー

コマンド応答形式を参照。

7.3.7 capAnalysisImg

説明

デバッグ用コマンド。パラメータで指定した画像種に対応したカメラ及び照明設定を行い撮影を行います。撮影時の設定は設定ファイル（rn700.conf）を参照し、保存するファイル名の命名規則は captureImage コマンドと同様です。本コマンドは、コマンド受付状態時のみに実行可能でその他の場合エラーで終了します。また、LED / LCD の点灯 / 消灯待ち時間中に初期化スイッチを押下することで待ち時間をスキップすることができます。

コマンド

```
{“method”: “capAnalysisImg”, “params”: [Param1, Param2], “id”: id 番号}
```

パラメーター

項目	JSON の型（json_type）	内容
Param1	int	撮影する画像種 0：透過画像 1：反射画像 2：スポット照明画像 3：グリッド画像
Param2	int	光量センサ値取得 0：OFF 1：ON

応答

正常

```
{“result”: “Param1”, “id”: id 番号}
```

項目	値	内容
Param1	String	RAW データ ID
Param2	int	Red 光量　(0 ～ 4095)
Param3	int	Green 光量 (0 ～ 4095)
Param4	int	Blue 光量　(0 ～ 4095)

例：

```
{“result”: [“D20140507105640.bin”, 10, 10, 10], “id”: 1}
```

※ 光センサ値取得 OFF の場合、光量は全て 0 になります

エラー

コマンド応答形式を参照。

7.3.8 deleteImage

説明

SD カードに保存している captureImage コマンドで保存した画像を全て消去します。

コマンド

{“method”: “deleteImage”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.9 actImageProcessing

説明

指定された画像に対し指定された画像処理を実行し、処理結果を出力します。

処理結果は、SD カードの actImg フォルダ配下の画像処理 ID に対応したフォルダ内に保存されます。処理結果を出力する際は、一度保存フォルダ内のファイルを全削除してから保存します。

コマンド

{“method”: “actImageProcessing”, “params”: [Param1, “Param2”], “id”: id 番号}

パラメーター

項目	値	内容
Param1	int	画像処理 ID (1 ～ 8)
Param2	String Array	HALCON　ID

例：{“method”: “actImageProcessing”, “params”: [1, “D20140507105640.bmp”], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.10 startAnalysis

説明

測定を開始します。測定終了後に応答メッセージを返します。

コマンド

{“method”: “startAnalysis”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [“Param1”, “Param2”], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	画像 ID
Param2	String	測定結果 ID

例：{“result”: [“image.jpg”, “result.csv”], “id”: 1}

エラー

コマンド応答形式を参照。

7.3.11 startAnalysisAsync

説明

測定を開始します。測定の開始前に応答メッセージを返します。

コマンド

{“method”: “startAnalysisAsync”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: [“Param1”, “Param2”], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	画像 ID
Param2	String	測定結果 ID

例：{“result”: [“image.jpg”, “result.csv”], “id”: 1}

エラー

コマンド応答形式を参照。

7.3.12 printResult

説明

最後に測定した測定結果をパラメーターに応じて印字します。測定結果はファームウエア起動時に削除されます。

コマンド

{“method”: “printResult”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	印字種別 (0：詳細分類　1：標準分類)
Param2	int	印字計算方法 (0：粒%　1：質量換算%)
Param3	int	印字結果指定(-1： 平均なし時の結果 0：平均測定時の平均結果 1 以上： 平均測定時の N 回目の結果)

例：{“method”: “printResult”, “params”: [0, 1, -1], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.13 dispLcdBmp

説明

LCD の画像表示の表示 / 非表示を切り替えます。表示する画像は、setLcdBitmap コマンドで設定するか、または SD カードの lcd フォルダ内にファイル名 ”dispLcd.bmp” でビットマップファイルを格納しておく必要があります。

コマンド

{“method”: “dispLcdBmp”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	ON：表示　OFF：非表示

例：{“method”: “dispLcdBmp”, “params”: “ON”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.14 dispLcdRGB

説明

指定した RGB 値で LCD の全面を塗りつぶします。

コマンド

{“method”: “dispLcdRGB”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	Red 値　(0 ～ 255)
Param2	int	Green 値 (0 ～ 255)
Param3	int	Blue 値　(0 ～ 255)

例：{“method”: “dispLcdRGB”, “params”: [0,255, 128], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.15 dispOledText

説明

OLED に全角文字列を表示します。文字は、6 文字まで表示できます。

コマンド

{“method”: “dispOledText”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	OLED 表示文字

例：{“method”: “dispOledText”, “params”: “測定可能”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.16 dispOledMsg

説明

指定した ID に対応するメッセージを OLED に表示します。

コマンド

{“method”: “dispOledMsg”, “params”: Param1, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	OLED 表示メッセージ ID

例：{“method”: “dispOledMsg”, “params”: 1, “id”: 1}

表示メッセージ(T.B.D.)

ID	表示メッセージ
0	(表示なし)
1	初期化中
2	初期調整 1
3	初期調整 2
4	測定可能
5	測定中
6	標準版を入れて下さい
7	標準版を抜いて下さい
8	トレイが違います
9	トレイを抜いて下さい
10	トレイが抜かれました
11	温度範囲外
12	傾いています
13	光が漏れています
14	振動を検出しました

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.17 soundBuzzer

説明

ブザーを鳴らします。切り替えタイミング１、２はそれぞれ信号を ON 出力する時間、OFF する時間を指定します。

コマンド

{“method”: “soundBuzzer”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	鳴動時間(ms)
Param2	int	切り替えタイミング１　(us)
Param3	int	切り替えタイミング２　(us)

例：{“method”: “soundBuzzer”, “params”: [500, 0, 100], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.18 startAccMonitoring

説明

デバッグ用コマンド。加速度センサ値の計測を開始します。計測結果は、RN700 の /tmp/accLog.csv ファイルに保存され、getProfileData　コマンドで取得できます。

コマンド

```
{“method”: “startAccMonitoring”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.19 stopAccMonitoring

説明

デバッグ用コマンド。stopAccMonitoring コマンドで開始した加速度センサ値の計測を終了します。

コマンド

```
{“method”: “stopAccMonitoring”, “params”: [], “id”: id 番号}
```

パラメーター

なし

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.20 readAccelerationReg

説明

デバッグ用コマンド。加速度センサの指定アドレスのレジスタ値を取得します。

コマンド

{“method”: “readAccelerationReg”, “params”: “Param1”, “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	レジスタアドレス（16 進数文字列 “00” ～ “39”）

例：{“method”: “readAccelerationReg”, “params”: “00”, “id”: 1}

応答

正常

{“result”: “Param1”, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	加速度センサレジスタ値（16 進数文字列）

例：{“result”: “E5”, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.21 writeAccelerationReg

説明

デバッグ用コマンド。加速度センサの指定アドレスのレジスタ値に指定データを書込みます。

コマンド

{“method”: “writeAccelerationReg”, “params”: [“Param1”, “Param2”], “id”: id 番号}

パラメーター

項目	JSON の型（json_type）	内容
Param1	String	レジスタアドレス（16 進数文字列 “00” ～ “39”）
Param2	String	書込みデータ（16 進数文字列 “00” ～ “FF”）

例：{“method”: “writeAccelerationReg”, “params”: “00”, “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.22 startCaptureLoop

説明

カメラ画像の連続キャプチャを開始します。画像を取得する場合は getCaptureImage コマンドを使用し、連続キャプチャを停止する場合は stopCaptureLoop コマンドを使用します。

コマンド

```
{“method”: “startCaptureLoop”, “params”: [Param1, Param2], “id”: id 番号}
```

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	取得画像サイズ(幅) （1 ～ 2592）
Param2	int	取得画像サイズ(高さ)（1 ～ 1944）

※ パラメータに指定する画像サイズの推奨値は下記のとおりです。推奨値以外を指定した場合は正しい画像が取得できな場合があります。

推奨画像サイズ(幅 x 高さ)
2592 x 1944
1296 x 972
648 x 486

```
{“method”: “startCaptureLooop”, “params”: [2592, 1944], “id”: id 番号}
```

応答

正常

```
{“result”: 0, “id”: id 番号}
```

エラー

コマンド応答形式を参照。

7.3.23 stopCaptureLoop

説明

startCaptureLoop コマンドで開始した連続キャプチャを終了します。

コマンド

{“method”: “stopCaptureLoop”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.24 rawToImage

説明

RAW 画像(bin ファイル)を指定した画像形式（jpg, bmp, tif ファイル） に変換します。

コマンド

{“method”: “rawToImage”, “params”: [“Param1”, “Param2”, ”Param3”, Param4], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	String	変換画像形式 jpg：jpg 画像に変換 bmp：bmp 画像に変換 tif：tif 画像に変換
Param2	String	変換画像および変換後画像の保存フォルダパス
Param3	String	変換画像および変換後画像のファイル名 拡張子なしのファイル名を指定します
Param4	int	スムージング設定 0：スムージングしない 1：スムージングする ※変換画像形式に jpg を指定した場合は無視します

例：{“method”: “rawToImage”, “params”: [“bmp”, “/tmp/capImg”, “image”, 0], “id”: 1}

/tmp/capImg/image.bin をスムージングなしで bmp 変換し、変換後画像を /tmp/capImg/image.bmp に保存します。

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.3.25 chkTray

説明

トレイ確認を行います。

コマンド

{“method”: “chkTray”, “id”: id 番号}

パラメーター

なし

応答

正常

{“result”:[Param1, Param2], “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	0 ：トレイ使用可 1 ：トレイ使用不可 2 ：測定用トレイ未挿入 3 ：光量調整の必要あり 10：トレイ確認異常終了
Param2	int	エラー番号 トレイ確認異常終了時に有効。エラー番号は RN700_FW 仕様書のエラー番号項目を参照

エラー

コマンド応答形式を参照。

7.4 アップデートコマンド

ファームウエアのアップデートに関連するコマンドを記載します。使用するファームウエアの種類とファイル名は下記の通りです。

種類	ファイル名
Linux カーネルイメージ	rn700kernel.bin
ユーザーランドイメージ	rn700userland.bin
ブートローダーイメージ	rn700bootloader.bin

7.4.1 writeFirmware

説明

ファームウエア更新を開始します。ファームウエア更新の完了は、 `getFirmwareStatus` コマンドを使用することで検出可能です。

本コマンドを実行する前に、SD カードまたは USB メモリの fw フォルダに更新イメージファイルを格納しておく必要があります。

更新したファームウエアは、更新完了後に再起動することで適用されます。

また、本コマンドで使用するファームウエアは下記フォーマットの 16 バイトバイナリデータのヘッダが付加されたイメージファイルである必要があります。

R	N	7	0	0	_	イメージタイプ	_	バージョン

イメージタイプ：1 バイト固定長

Linux カーネルイメージの場合 “ k ”

ユーザーランドイメージの場合 “ u “

ブートローダーの場合 “ b ”

バージョン ：8 バイト固定長（例：00000100）

コマンド

```
{“method”: “writeFirmware”, “params”: [“Param1”, “Param2”], “id”: id 番号}
```

パラメーター

項目	JSON の型（json_type）	内容
Param1	String	更新対象 k：Linux カーネル u：ユーザーランド b：ブートローダー
Param2	String	更新イメージファイル読み込み先 sd ：SD カード経由 usb：USB メモリ経由

例：`{“method”: “writeFirmware”, “params”: [“u”, “sd”], “id”: 1}`

応答

正常

{“result”: “Param1”, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	String	イメージファイルのヘッダ情報のバージョンを 8 文字の文字列で返します。

例：{“result”: “00000100”, “id”: 1}

エラー

コマンド応答形式を参照。

7.4.2 sendFirmware

説明

ファームウエアを RN700 に送信し、指定したファイル名で SD カードの fw フォルダに格納します。
既に同じファイル名のファイルがあった場合は上書きします。

コマンド

{“method”: “sendFirmware”, “params”: [], “id”: id 番号}
上記のコマンドに続き、ファームウエア (バイナリデータ)を送信します。

パラメーター

なし

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.4.3 getFirmwareStatus

説明

ファームウエア更新、ベリファイに関わるステータス情報を返します。

コマンド

{“method”: “getFirmwareStatus”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

{“result”: Param1, “id”: id 番号}

項目	JSON の型 (json_type)	内容
Param1	int	ステータス情報 0　：ファームウエア更新開始可能状態 1　：ファームウエア更新開始不可状態 100：ファームウエア更新中 101：ファームウエア更新エラー 200：ファームウエアベリファイ中 201：ファームウエアベリファイエラー 300：ファームウエア受信中 301：ファームウエア受信エラー

例：{“result”: 0, “id”: 1}

エラー

コマンド応答形式を参照。

7.5 Armadillo 840 - Armadillo 1500 通信用コマンド

7.5.1 setImgProcBoardStatus

説明

Armadillo 1500 の状態を Armadillo 840 に通知します。

コマンド

{“method”: “setImgProcBoardStatus”, “params”: [Param1, Param2, Param3], “id”: id 番号}

パラメーター

項目	JSON の型 (json_type)	内容
Param1	int	Armadillo 1500 の状態 0：起動処理中 1：A840 起動確認中 2：待受け状態 3：画像処理 1 処理中 4：画像処理 2 処理中 5：歪み補正確認中 6：歪み補正処理中 7：JPG 変換処理中 8：BMP 変換処理中 9：TIF 変換処理中
Param2	int	エラー状態 0：エラーなし 0 以外：エラーあり（エラーコード）
Param3	int	画像処理結果 最後に実行した画像処理の戻り値 初期値は 0

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

7.5.2 setImgProcRequestNotify

説明

Armadillo 840 から応答コマンドで画像処理リクエスト通知を受け取ります。本コマンドを Armadillo 840 に送信すると、コマンド受信応答（下記参照）が送信されます。続いて、画像処理を実行する際に画像処理リクエスト応答(下記参照)が送信されます。画像処理リクエスト応答は、setImgProcBoardStatus コマンドを発行するまで繰り返し送信されます。

コマンド

{“method”: “setImgProcRequestNotify”, “params”: [], “id”: id 番号}

パラメーター

なし

応答

正常

コマンド受信応答

{“result”: 0, “id”: id 番号}

画像処理リクエスト応答

{“result”: [Param1, “Param2”], “id”: id 番号}

項目	JSON の型（json_type）	内容
Param1	int	リクエストする画像処理番号 1：画像処理 1 処理 2：画像処理 2 処理 3：歪み補正確認 4：歪み補正処理 5：JPG 変換処理 6：BMP 変換処理 7：TIF 変換処理
Param2	String	各処理のパラメータ（T.B.D.）

エラー

コマンド応答形式を参照。

7.5.3 setResultData

説明

Armadillo 840 - Armadillo 1500 通信用コマンド。

Armadillo 840 の /tmp/actImg フォルダへ測定結果ファイルを送信します。既に同名のファイルがあった場合は上書きします。

コマンド

{“method”: “setResultData”, “params”: “Param1”, “id”: id 番号}

上記コマンドに続いて、ファイルデータ(バイナリデータ)を送信します。

パラメーター

項目	JSON の型 (json_type)	内容
Param1	Strings	ファイル名

例：{“method”: “setResultData”, “params”:[], “id”: 1}

応答

正常

{“result”: 0, “id”: id 番号}

エラー

コマンド応答形式を参照。

8 改定履歴

Ver. No.	日付	改定内容
1.0	2014/04/25	新規制定
1.1	2014/05/15	コマンド形式、シーケンス項目追加。コマンド詳細の追加変更。
1.2	2014/05/26	コマンド形式、シーケンスの小項目番号を修正。 設定コマンドシーケンス修正。 追加コマンド getOperatingStatus、deleteImage、writeFlash / readFlash、 printResult 変更コマンド getAnalysisLevel / setAnalysisLevel、actImageProcessing、 addAnalysisLevel / editAnalysisLevel、getVersion、 getStatus、getTemperature 削除コマンド getLcdFillColor / setLcdFillColor 、reset、 getAnalysisResultDisplayFlag / setAnalaysisResultDisplayFlag
1.3	2014/9/12	追加コマンド setCameraGain / getCameraGain, setTimeout / getTimeout dispLcdBmp、dispOledText、soundBuzzer 削除コマンド setCameraGlobalGain / getCameraGlobalGain , setCameraRGBGain / getCameraRGBGain 変更コマンド writeFirmware / getFirmwareStatus getIllumination setLed1RGB / getLed1RGB setLed2RGB / getLed2RGB getTemparature、getAcceleration 各コマンドのパラメーター表に値の型を追記
1.4Draft	2014/9/30	追加コマンド getAnalysisLevel 削除コマンド addAnalysisLevel 変更コマンド getAnalysisTypeList, getAnalysisType, getAnalysisLevelList, getAnalysisLevel, setAnalysisType, setAnalysisLevel, editAnalysisLevel, deleteAnalysisLevel

1.4	2014/10/06	追加コマンド getLimitSwitch, getSettingFile / setSettingFile, setLcdPower, dispLcdRGB, setLedPower 削除コマンド getAnalysisTypeList, getAnalysisLevelList, editAnalysisLevel, editAnalysisLevel 変更コマンド getAnalysisType, getAnalysisLevelVal, getAnalysisLevel,, setLcdPwm, actImageProcessing, printResult
1.5Draft	2014/12/19	変更コマンド getIllumination getAnalysisType setCameraShutterWidth getOperatingStatus
1.5Draft_a	2015/01/30	コマンド詳細の表記を統一。 追加コマンド startAnalysisAsync startAccMonitoring stopAccMonitoring getAccMonitoringData readAccelerationReg writeAccelerationReg 削除コマンド stopAnalysis
1.5Draft_b	2015/02/23	追加コマンド calibrateRGB getProfileData 削除コマンド getAccMonitoringData
1.5Draft_c	2015/02/27	変更コマンド calibrateRGB getProfileData
1.5Draft_d	2015/03/06	変更コマンド calibrateRGB 追加コマンド startCaptureLoop stopCaptureLoop getCaptureImage
1.5Draft_e	2015/03/12	変更コマンド

		captureImage 追加コマンド getOperatingMode setOperatingMode
1.5Draft_f	2015/03/25	変更コマンド calibrateRGB
1.5Draft_g	2015/04/03	変更コマンド getVersion getStatus getLimitSwitch
1.5Draft_h	2015/04/20	変更コマンド getOperatingMode dispOledText 追加コマンド dispOledMsg
1.5Draft_i	2015/04/28	変更コマンド calibrateCamera
1.5Draft_j	2015/05/19	追加コマンド capAnalysisImg
1.5Draft_k	2015/05/29	変更コマンド getSettingFile / settingFile calibrateRGB
1.5Draft_l	2015/06/16	関連資料、コマンド実行可能範囲 項目を追加 “calibrateRGB” を “illuminationTest” にコマンド名変更 変更コマンド capAnalysisImg
1.5Draft_m	2015/07/17	変更コマンド captureImage getOperatingStatus コマンド実行可能範囲に”デバッグ中”項目を追加

1.5Draft_n	2015/08/07	追加コマンド initialCalibration 削除コマンド calibrateLed calibrateCamera
1.5Draft_o	2015/08/07	追加コマンド setUserInfo
1.5Draft_p	2016/03/28	状態名を統一
1.5Draft_q	2016/06/30	概要に Armadillo 840 – Armadillo 1500 間通信について追記

		追加コマンド setImgProcBoardStatus setImgProcRequestNotify setResultData
1.5Draft_r	2016/09/30	変更コマンド setImgProcBoardStatus setResultData 削除コマンド getBattelyLevel
1.5Draft_s	2016/10/26	追加コマンド setBinaryFile
1.5Draft_t	2017/02/28	変更コマンド getAnalysisLevelVal getAnalysisLevel
1.5Draft_u	2017/03/15	追加コマンド getDate 変更コマンド setUserInfo
1.5Draft_v	2017/03/22	追加コマンド getCalcType, getClassType, getOutputMode, setCalcType, setClassType, setOutputMode
1.5Draft_w	2017/05/23	変更コマンド writeFlash, readFlash writeFirmware
1.5Draft_x	2018/08/28	見出しの書式を変更 バイナリデータブロック転送 項目追加 追加コマンド getBinaryFileEx, setBinaryFileEx
1.5Draft_y	2018/09/25	バイナリデータブロック転送 項目 説明文変更、フロー項目を追加
1.5Draft_z	2018/10/31	変更コマンド printResult
1.6Draft_a	2019/01/31	追加コマンド rawToImage, getPrintPattern, getLotNumber, getSaveData, setPrintPattern, setLotNumber, setSaveData, chkTray
1.6Draft_b	2019/06/28	コマンド実行可能範囲に遷移待機状態を追加 変更コマンド getOperatingStatus
1.6Draft_c	2020/06/17	追加コマンド writeRamData
1.6Draft_e	2020/09/07	変更コマンド

		setAnalysisType(測定粒種 ID の有効/無効チェックを追加)