RICOH

RICOH SC-20

ソケット通信機能使用説明書

本書の読み方

マークについて

本書で使われているマークには次のような意味があります。

重要

操作するときに注意していただきたいことや、制限事項などを説明しています。必ずお読みください。

補足

知っておくと便利な情報や、補足的な操作方法などを説明しています。

参照／（→P. ##）

参照先を示します。

［ ］

画面上の項目やボタンの名称を示します。

RICOH

目次

1. 概要

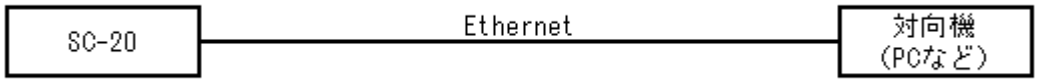
SC-20 は、TCP/IP のソケット通信機能を利用して外部機器と接続することができます。
本書は、ソケット通信の接続手順とソケット通信時に設定するデータフォーマットを説明します。

接続構成

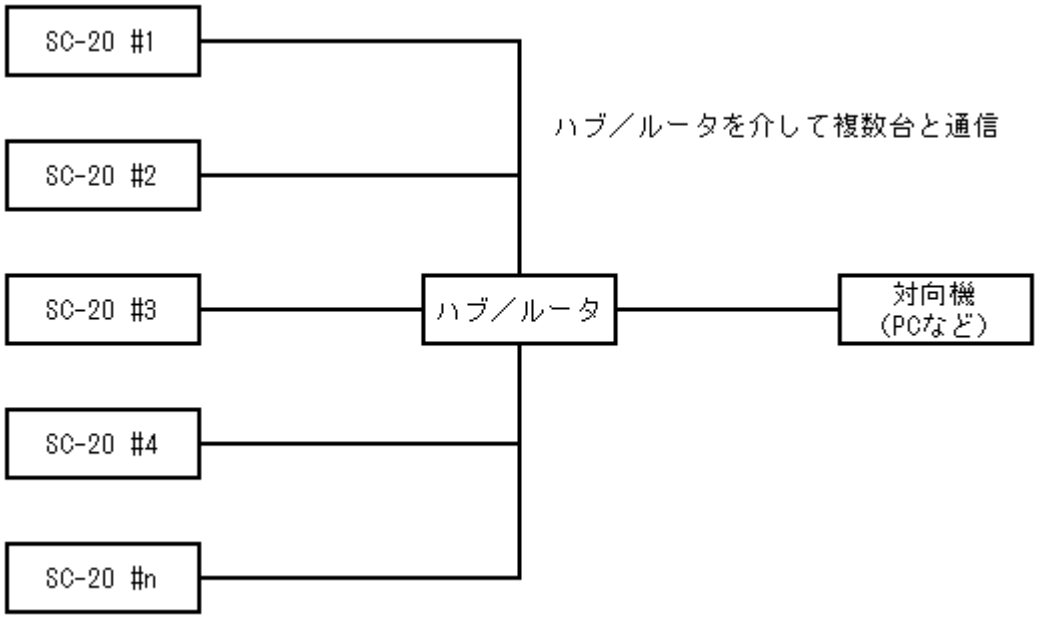
接続構成

Ethernet 接続を利用すると、下図に示すように複数台の作業支援カメラシステムを接続できます。
SC-20 の接続方式に応じて、対向機側で操作用のアプリケーションソフトを構築する必要があります。

例 1：



例 2：



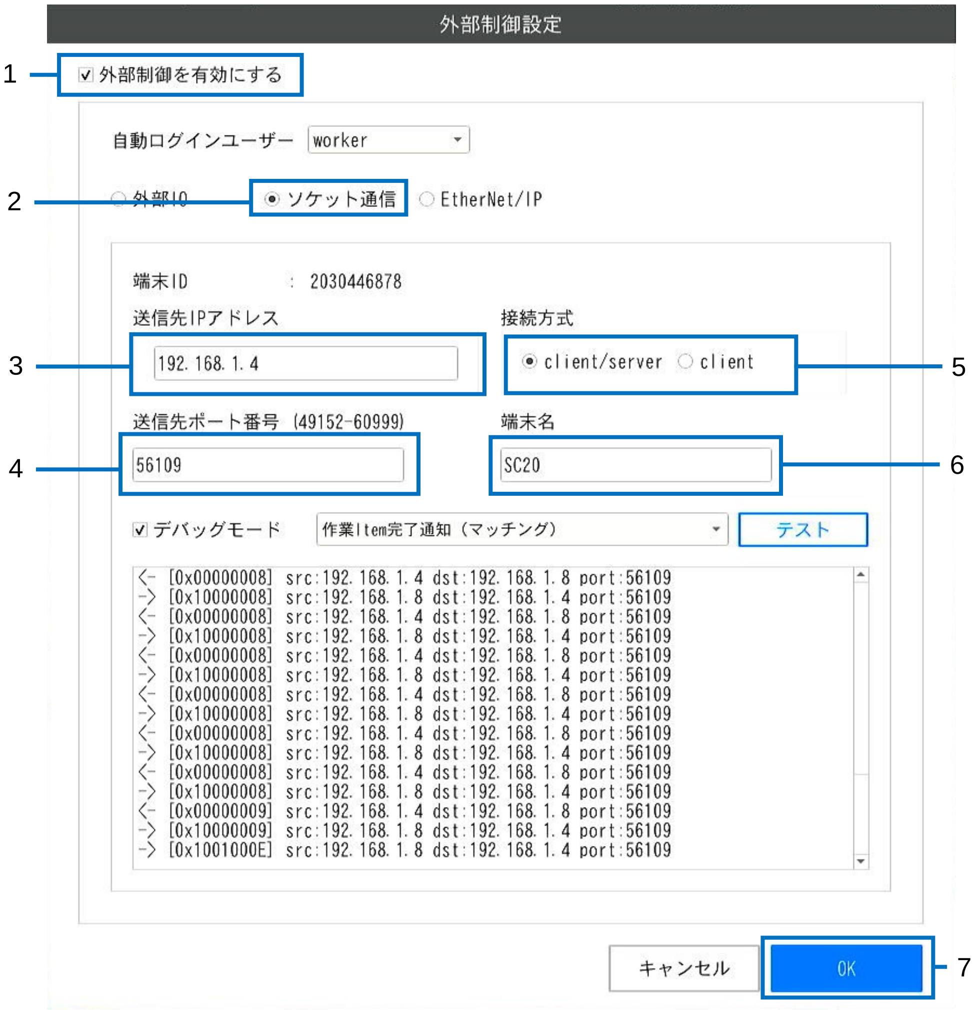
2. ソケット通信

ソケット通信制御機能を有効にする

SC-20 に［管理者モード］でログインし、［システム設定］メニューの［外部制御設定...］を選択して、下図画面を表示します。

参照

- SC-20 の操作の詳細については、SC-20 シリーズ 使用説明書を参照してください。



1. ［外部制御を有効にする］をチェックします。
2. ［ソケット通信機能］を選択します。
3. ［送信先 IP アドレス］に、対向機（ソケット通信の相手）の IP アドレスを設定します。
4. ［送信先 IP ポート番号］に、対向機のポート番号を設定します。
5. ［接続方式］に[client/server]または[client]のいずれかを設定します。
6. ［端末名］に、機器の名称を入力します。
   1～50 文字の半角英数字で任意の名称を入力してください。
7. ［OK］をクリックします。
   設定が保存されます、機能は再起動後反映されます。

補足

- ［端末 ID］は、システムが自動で設定します。

デバッグモードを行う

デバッグモードを設定できます。対向機のアプリケーション開発時に、デバッグモードを利用して疎通確認や動作検証できます。

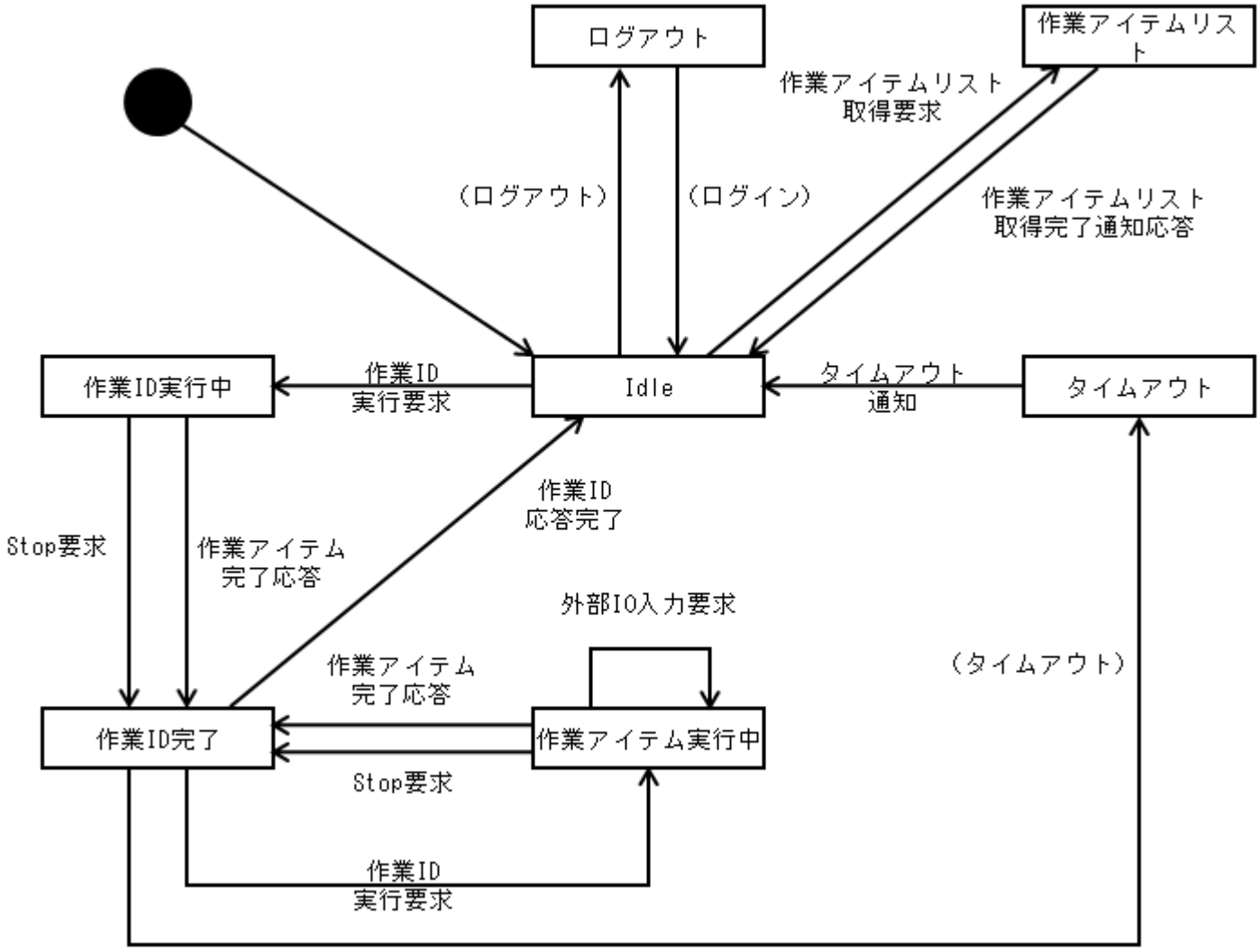
補足

- デバッグモードを設定するには、あらかじめソケット通信機能を有効に設定してください（→P. 6）。

1. ［デバッグモード］にチェックをします。
2. **左のプルダウンから通知メッセージを選択して［テスト］をクリックします。**
   プルダウンメニューに応じたダミーデータが入った通知メッセージを送信できます。
3. **SC-20 と対向機のメッセージログが表示されます。**

コマンドとシステムの状態

ソケット通信でやり取りされるコマンドと、システムの状態遷移を以下に記します。
括弧で囲まれた処理はユーザーによる手動、或いはシステム内で自動に行っている操作になります。



シーケンス

ソケット通信制御機能で、想定されるシーケンスを説明します。

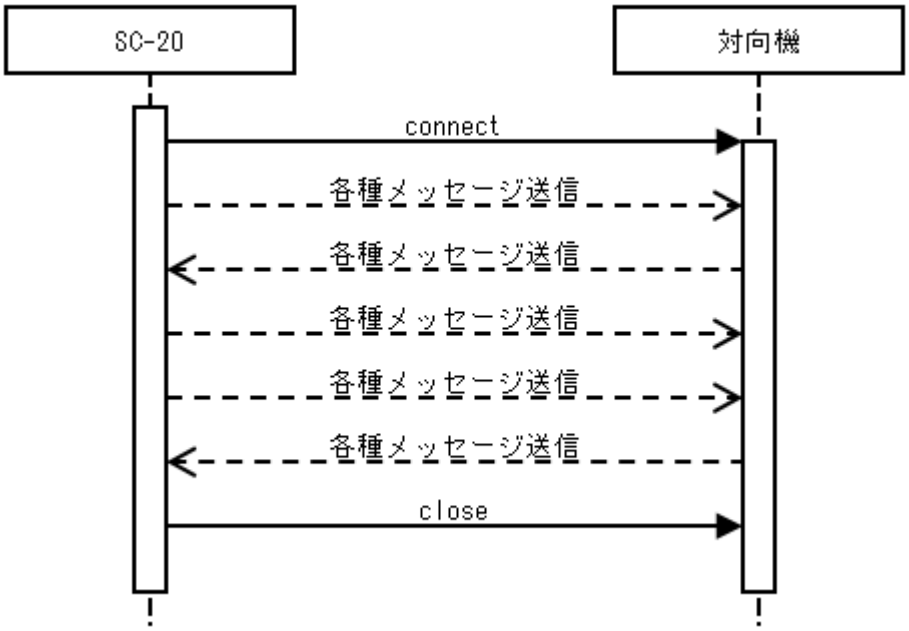
重要

・以下に示すシーケンス外の通信を行った場合、正しい処理ができなくなります。
・他コマンドのシーケンス中に要求コマンド送信を行わないようにしてください。

接続方式

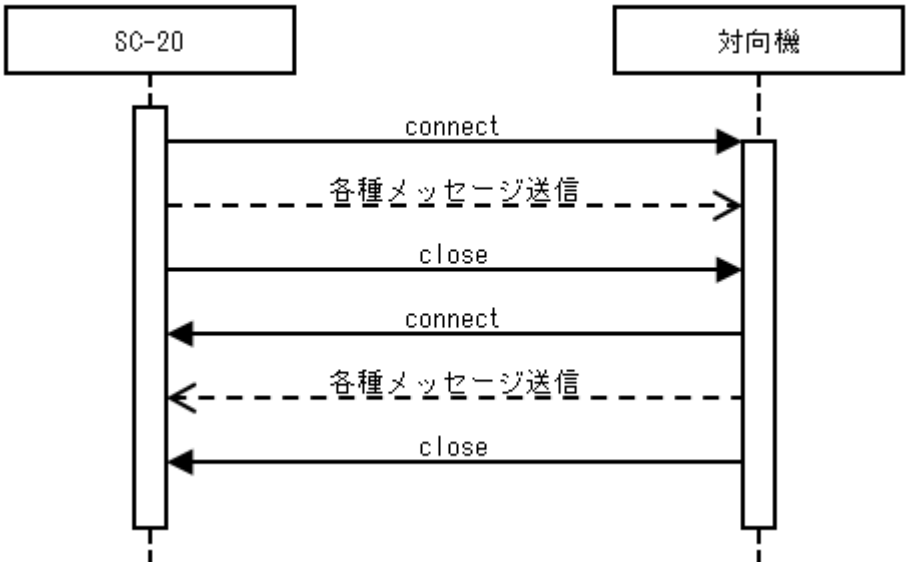
【client モード】

connect を実施しシステム停止終了まで維持し続ける接続方式となります。



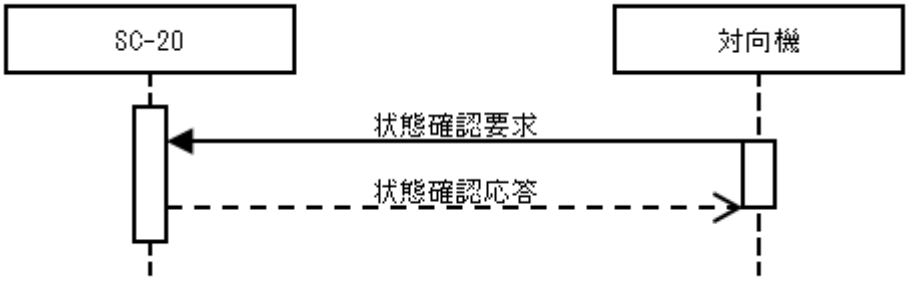
【client/server モード】

送信する側が connect を実施しメッセージ送信後に close を実施する方式となります。
※対向機から SC-20 方向の IP ポートは 56109 固定になります。



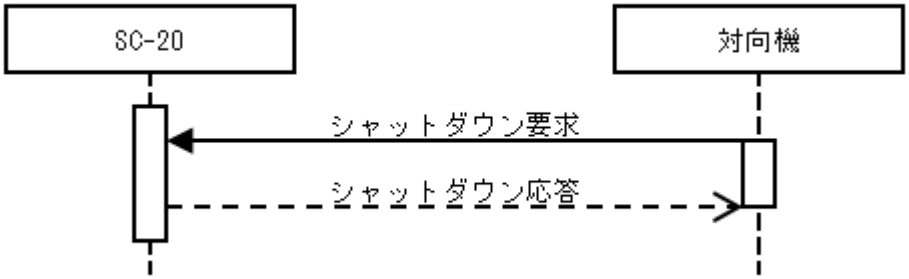
状態確認

対向機から SC-20 の状態を確認する場合は、対向機から「状態確認要求」を送信します。
任意のタイミングで、カメラの状態が「状態確認応答」として送信されます。



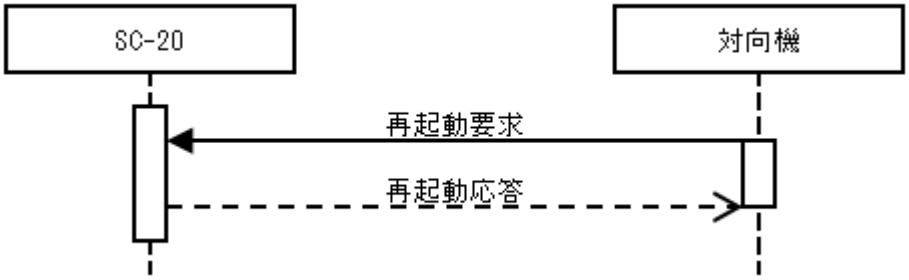
シャットダウン

対向機から SC-20 のシャットダウンを行う場合は、対向機から「シャットダウン要求」を送信します。
任意のタイミングで、カメラの状態が「シャットダウン応答」として送信されます。



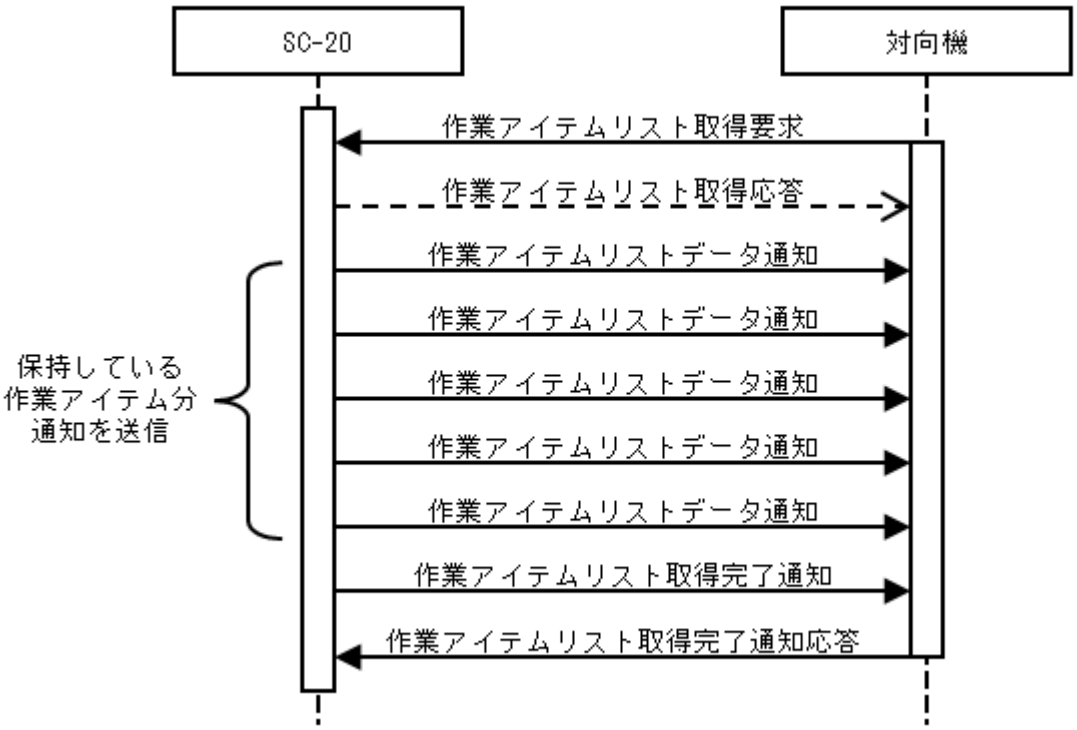
再起動

対向機から SC-20 の再起動を行う場合は、対向機から「再起動要求」を送信します。
任意のタイミングで、カメラの状態が「再起動応答」として送信されます。



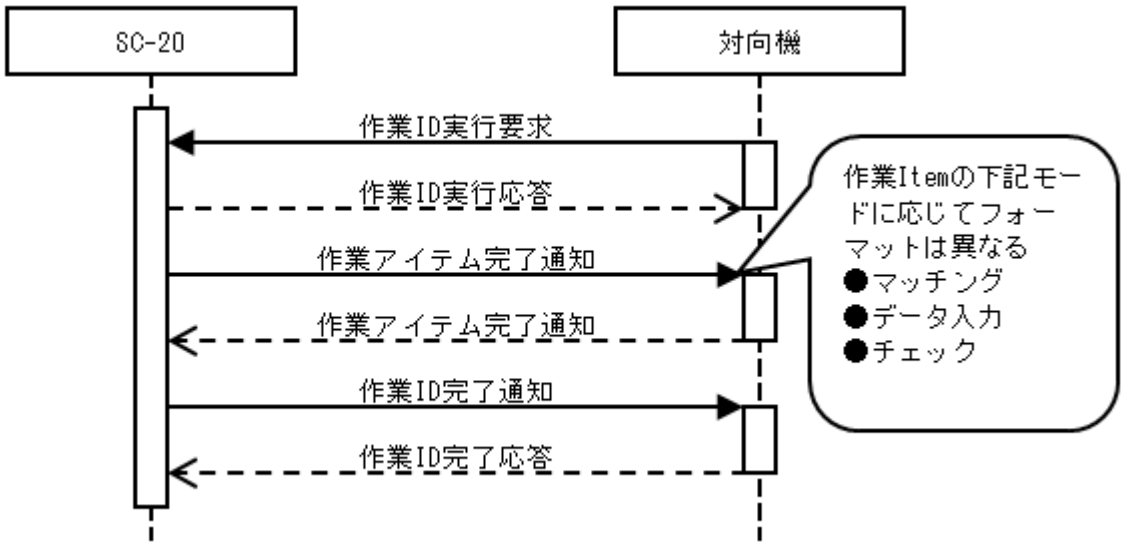
作業アイテムリスト取得

SC-20 に登録されている作業アイテムを取得する場合は、対向機から「作業アイテムリスト取得要求」を送信します。SC-20 は、「作業アイテムリスト応答」を送信した後に、「作業アイテムリストデータ通知」を送信します。
「作業アイテムリストデータ通知」は、SC-20 に登録されている作業アイテム１件に対して１回の通知になります。全作業アイテム分の「作業アイテムリストデータ通知」の送信が完了したら、「作業アイテムリスト取得完了通知」を送信します。対向機が「作業アイテムリスト取得完了応答」を送信し、シーケンスは終了となります。



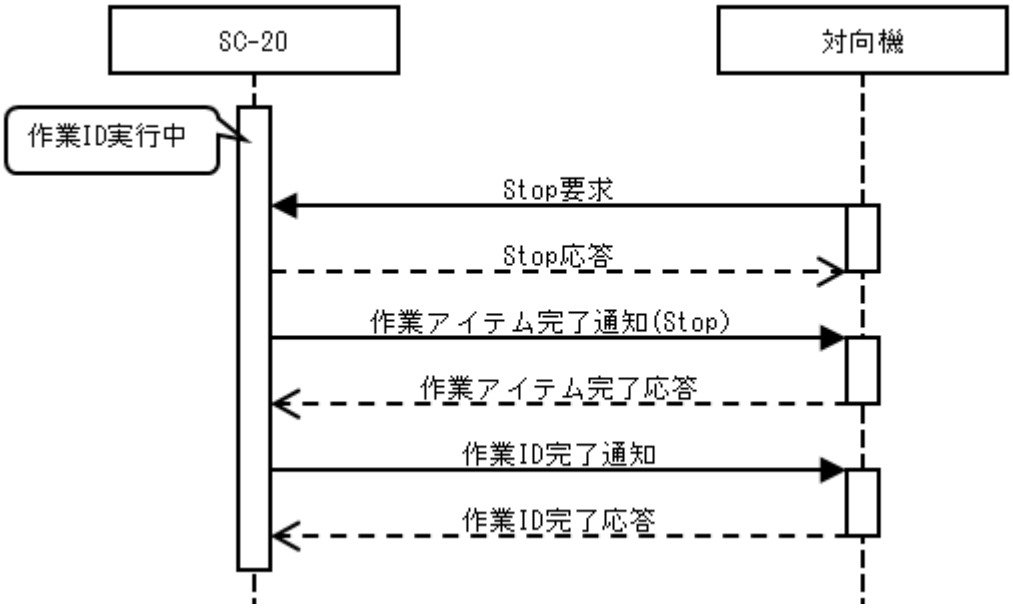
作業 ID 実行処理

対向機から SC-20 の作業 ID を切り替えて、作業 ID に登録されている先頭の作業アイテムからシーケンスを実行させる場合は、対向機から「作業 ID 実行要求」を送信します。作業 ID に登録されている先頭の作業アイテムから順次実行されます。実行結果は、SC-20 から「作業アイテム完了通知」で送信します。
作業 ID に登録されているすべての作業アイテムの実行が完了したら、SC-20 が「作業 ID 完了通知」を送信します。対向機が「作業 ID 完了応答」を SC-20 に送信し、シーケンスは終了となります。



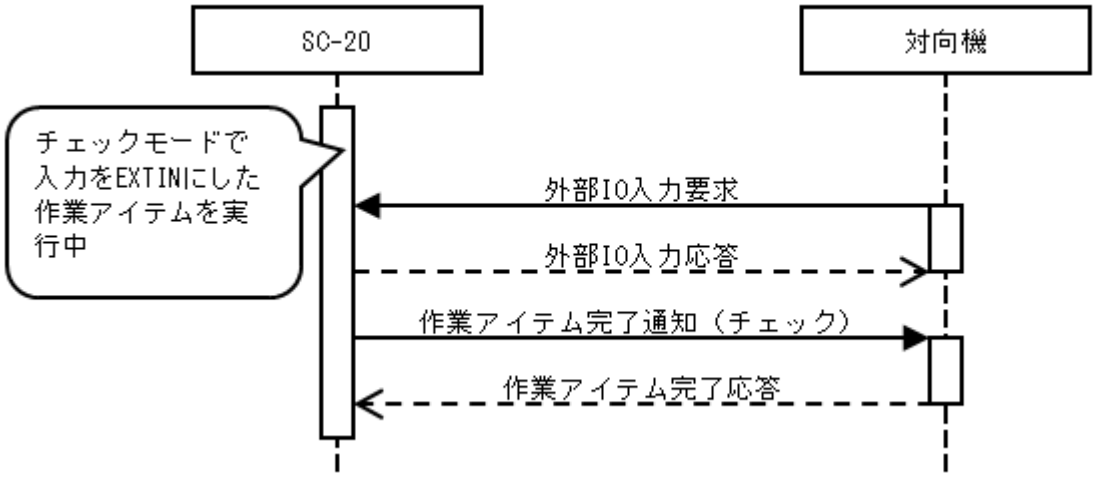
作業アイテム停止

「作業アイテム実行中」の間に処理を停止する場合は、SC-20 に「Stop 要求」を送信すると、作業を停止します。



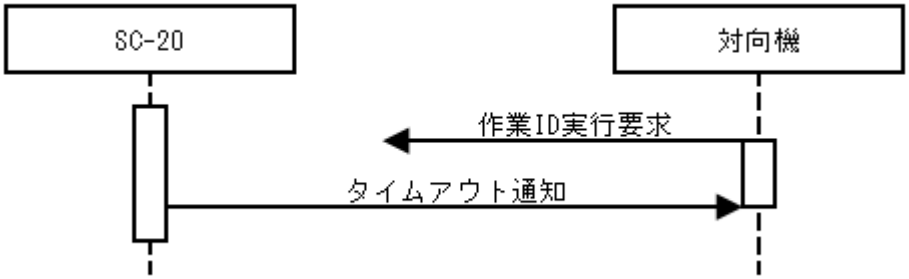
外部 IO 入力

あらかじめチェックモードの EXTIN を設定している作業アイテム実行中に、「外部 IO 入力要求」を送信すると、コマンドパラメータに応じた処理を実行し、「外部 IO 入力応答」、「作業アイテム完了通知」を続けて SC-20 から送信します。



タイムアウト

対向機からの「作業 ID 実行要求」がなんらかの原因で SC-20 に届かなかった場合は、SC-20 内部で設定していたタイマーが満了するとシステム内部で「タイムアウト通知」送信後対向機へ「タイムアウト通知」を送信します。ネットワーク経路上の問題で「タイムアウト通知」が到達しない場合もあるので対向機側で、ネットワーク状況を確認し、対応してください。



3. メッセージ ID

ここでは、ソケット通信で使用するメッセージ ID を説明します。

ソケット通信メッセージ ID

要求系

メッセージ名		メッセージ ID	
		対向機→SC-20	SC-20→対向機
		要求	応答
	Stop	0x00000003	0x10000003
	作業アイテムリスト取得	0x00000004	0x10000004
	作業 ID 実行	0x00000005	0x10000005
	外部 IO 入力	0x00000007	0x10000007
	状態確認	0x00000008	0x10000008
	シャットダウン実行	0x00000009	0x10000009
	再起動実行	0x0000000A	0x1000000A

通知系

メッセージ名		メッセージ ID	
		SC-20→対向機	対向機→SC-20
		通知	通知応答
	作業アイテム完了（マッチング）	0x10010002	0x00010007
	作業アイテム完了（データ入力）	0x10010003	0x00010007
	作業アイテム完了（チェックモード）	0x10010004	0x00010007
	作業アイテム完了（Stop）	0x10010005	0x00010007
	作業 ID 完了	0x10010008	0x00010008
	作業アイテムリストデータ	0x10010009	-
	作業アイテムリスト取得完了	0x1001000B	0x0001000B
	システム停止	0x1001000E	-
	タイムアウト	0x1001000F	-

メッセージヘッダー

以下は、ソケット通信のデータ領域に設定する共通メッセージヘッダーの構成図です。
端末 ID および端末名は、SC-20 の起動時に送信される「起動通知」内のデータを使用してください。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

メッセージ ID（要求メッセージ）

Stop 要求

メッセージ ID	メッセージ名	説明
0x00000003	Stop 要求	「作業アイテム完了通知」の前に「Stop 要求」を送信すると、認識中の処理を中断する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

Stop 応答

メッセージ ID	メッセージ名	説明
0x10000003	Stop 応答	対向機から送信される「Stop 要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0：OK -1：NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

作業アイテムリスト取得要求

メッセージ ID	メッセージ名	説明
0x00000004	作業アイテムリスト取得要求	SC-20 に登録されている作業アイテムリストを取得する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

作業アイテムリスト取得応答

メッセージ ID	メッセージ名	説明
0x10000004	作業アイテムリスト取得応答	対向機から送信される「作業アイテムリスト取得要求」に対する応答メッセージ。 結果には SC-20 に登録されている作業アイテムの件数が挿入される（最大 32767 件）。 NG 時は-1 が返却されエラーコードに事由を示すコードが付与される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	1-32767：総 Item 件数 -1：NG
0x0052	2 Byte	uint16	エラーコード	4. エラーコード（P.44）参照

作業 ID 実行要求

メッセージ ID	メッセージ名	説明
0x00000005	作業 ID 実行要求	対向機から、作業 ID の実行を開始する。 外部 IO の OUTO に「RUN」が設定されている場合は ON 状態に遷移する。 補足 • ON/OFF は、ユーザー定義によります。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	作業 ID			
0x0088	作業指示			
0x00c8	作業アイテム			
0x0108	ユーザーID			
0x0148	作業番号			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048 - 0x0087	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0088 - 0x00c7	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00c8 - 0x0107	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで
0x0108 - 0x0147	64 Byte	char	ユーザーID	半角英数字で、最大 50 文字まで
0x0148 - 0x0187	64 Byte	char	作業番号	半角英数字で、最大 50 文字まで

作業 ID 実行応答

メッセージ ID	メッセージ名	説明
0x10000005	作業 ID 実行応答	対向機から送信される「作業 ID 実行要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0：OK -1：NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

外部 IO 入力要求

メッセージ ID	メッセージ名	説明
0x00000007	外部 IO 入力要求	作業アイテムがチェックモードで待機中のときに、SC-20 に対して送信するメッセージ。 外部 IO の EXTIN に相当する動作を行う。

メッセージフォーマット

アドレス	bit										
	31	16	15								0
0x0000	メッセージ ID										
0x0004	端末 ID										
0x0008	端末名										
0x0048	作業 ID										
0x0088	reserve		EXTIN9	EXTIN8	EXTIN7	EXTIN6	EXTIN5	EXTIN4	EXTIN3	EXTIN2	EXTIN1	EXTIN0

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048 - 0x0087	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0088	4 Byte	uint32	外部 IO	外部 IO の入力論理値をビットフィールドで設定する。 [0]EXTIN0 : 1/0 [1]EXTIN1 : 1/0 [2]EXTIN2 : 1/0 [3]EXTIN3 : 1/0 [4]EXTIN4 : 1/0 [5]EXTIN5 : 1/0 [6]EXTIN6 : 1/0 [7]EXTIN7 : 1/0 [8]EXTIN8 : 1/0 [9]EXTIN9 : 1/0 [10-31] : reserve

外部 IO 入力応答

メッセージ ID	メッセージ名	説明
0x10000007	外部 IO 入力応答	「外部 IO 入力要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0：OK -1：NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

状態確認要求

メッセージ ID	メッセージ名	説明
0x00000008	状態確認要求	ソケット通信の現状態を確認するためのメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

状態確認応答

メッセージ ID	メッセージ名	説明
0x10000008	状態確認応答	「状態確認要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。 結果の値によって現在のカメラ状態を示しているものとなる。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	コマンドとシステムの状態（→P.8）の各状態を、以下の番号で示す。 1：ログアウト 2：Idle 3,4：作業アイテム転送中 5,6：作業 ID 開始 7,13：作業アイテム実行中 8,9,14：作業 ID 実行中 10,11,12：作業 ID 完了 15：タイムアウト -1: NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード（P.44）参照

シャットダウン実行要求

メッセージ ID	メッセージ名	説明
0x00000009	シャットダウン実行要求	SC-20 のシャットダウンを実施したい場合に使用するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

補足

・シャットダウン実行要求はログオフ時には作用しません。

シャットダウン実行応答

メッセージ ID	メッセージ名	説明
0x10000009	シャットダウン実行応答	「シャットダウン実行要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0: OK -1: NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

再起動実行要求

メッセージ ID	メッセージ名	説明
0x0000000A	再起動実行要求	SC-20 の再起動を実施したい場合に使用するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

補足

・再起動実行要求はログオフ時には作用しません。

再起動実行応答

メッセージ ID	メッセージ名	説明
0x1000000A	再起動実行応答	「再起動実行要求」に対する応答メッセージ。 応答メッセージに結果とエラーコードが入力される。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0: OK -1: NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

メッセージ ID（通知メッセージ）

作業アイテム完了通知（マッチング）

メッセージ ID	メッセージ名	説明
0x10010002	作業アイテム完了通知(マッチング)	マッチング処理終了時に、対向機側へ送信するメッセージ。 「チェックポイント ID_X」以降のデータは、「チェックポイント数」の設定数に応じてデータの入力を行う。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			
0x0090	作業指示			
0x00d0	作業アイテム			
0x0110	ユーザーID			
0x01d8	作業番号			
0x02a0	経過時間		作業アイテム最終結果	
0x02a4	基準ポイント類似度			
0x02ac	チェックポイント数		基準ポイント回転角度	
0x02b0	reserve	判定結果	モード(マッチング)	チェックポイント ID_1
0x02b4	マッチング時間[msec]		回転角度	
0x02b8	類似度			
0x02c0	reserve	判定結果	モード(マッチング)	チェックポイント ID_2
0x02c4	マッチング時間[msec]		回転角度	
0x02c8	類似度			
∫	∫			
0x04a0	reserve	判定結果	モード(マッチング)	チェックポイント ID_20
0x04a4	マッチング時間[msec]		回転角度	
0x04a8	類似度			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0090 - 0x00cf	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00d0 - 0x010f	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで
0x0110 - 0x01d7	200 Byte	char	ユーザーID	半角英数字で、最大 198 文字まで
0x01d8 - 0x029f	200 Byte	char	作業番号	半角英数字で、最大 198 文字まで
0x02a0	2 Byte	int16	作業アイテム 最終結果	0：OK -1：NG -2：基準ポイント NG
0x02a2	2 Byte	uint16	経過時間(秒)	作業経過時間を秒単位で設定
0x02a4	8 Byte	double	基準ポイント 類似度	0.00000～1.00000 で設定
0x02ac	2 Byte	int16	基準ポイント 回転角度	180～-180 で設定する。
0x02ae	2 Byte	uint16	チェックポイント 数	チェックポイント数を 0～20 で設定
0x02b0	1 Byte	uchar	チェックポイント ID_1	チェックポイント ID を 1～20 で設定
0x02b1	1 Byte	uchar	モード	0：形状 1：色認識 2：質感
0x02b2	1 Byte	char	判定結果	0：OK -1：NG
0x02b3	1 Byte	uchar	reserve	未使用領域
0x02b4	2 Byte	int16	回転角度	180～-180 で設定 ※色認識・質感は 0 固定
0x02b6	2 Byte	uint16	マッチング時間 [msec]	0～65535 で設定
0x02b8	8 Byte	double	類似度	0.00000～1.00000 で設定

作業アイテム完了通知（データ入力）

メッセージ ID	メッセージ名	説明
0x10010003	作業アイテム完了通知(データ入力)	データ入力終了時に、対向機側へ送信するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			
0x0090	作業指示			
0x00d0	作業アイテム			
0x0110	ユーザーID			
0x01d8	作業番号			
0x02a0	経過時間(秒)		作業アイテム最終結果	
0x02a4	部品番号(設定済みの番号)			
0x0324	入力データ(ユーザーが入力した値)			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0090 - 0x00cf	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00d0 - 0x010f	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで
0x0110 - 0x01d7	200 Byte	char	ユーザーID	半角英数字で、最大 198 文字まで
0x01d8 - 0x029f	200 Byte	char	作業番号	半角英数字で、最大 198 文字まで
0x02a0	2 Byte	int16	作業アイテム 最終結果	0：OK -1：NG
0x02a2	2 Byte	uint16	経過時間(秒)	作業経過時間を秒単位で設定

アドレス	サイズ	属性	フィールド名称	説明
0x02a4 - 0x0323	128 Byte	char	部品番号	半角英数字
0x0324 - 0x0523	512 Byte	char	入力データ	半角英数字

作業アイテム完了通知（チェックモード）

メッセージ ID	メッセージ名	説明
0x10010004	作業アイテム完了通知(チェックモード)	チェックモード終了時に、対向機側へ送信するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			
0x0090	作業指示			
0x00d0	作業アイテム			
0x0110	ユーザーID			
0x01d8	作業番号			
0x02a0	経過時間(秒)		作業アイテム最終結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0090 - 0x00cf	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00d0 - 0x010f	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで
0x0110 - 0x01d7	200 Byte	char	ユーザーID	半角英数字で、最大 198 文字まで
0x01d8 - 0x029f	200 Byte	char	作業番号	半角英数字で、最大 198 文字まで
0x02a0	2 Byte	int16	作業アイテム最終結果	0：OK -1：NG
0x02a2	2 Byte	uint16	経過時間(秒)	作業経過時間を秒単位で設定

作業アイテム完了通知（Stop）

メッセージ ID	メッセージ名	説明
0x10010005	作業アイテム完了通知(Stop)	UI、外部 IO およびソケット通信からの Stop 実行後に SC-20 から送信するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			
0x0090	作業指示			
0x00d0	作業アイテム			
0x0110	経過時間(秒)		Stop 要因	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0090 - 0x00cf	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00d0 - 0x010f	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで
0x0110	2 Byte	int16	Stop 要因	0：UI からの停止 1：外部 IO からの停止 2：ソケット通信からの停止
0x0112	2 Byte	uint16	経過時間(秒)	作業経過時間を秒単位で設定

作業アイテム完了通知応答

メッセージ ID	メッセージ名	説明
0x00010007	作業アイテム完了通知応答	SC-20 から送信される「作業アイテム完了通知」に対する応答メッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	reserve			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	4 Byte	Uint32	reserve	未使用領域

作業 ID 完了通知

メッセージ ID	メッセージ名	説明
0x10010008	作業 ID 完了通知	作業 ID で管理している作業アイテムの実行がすべて完了した後、SC-20 から送信するメッセージ。 「作業アイテム完了通知」の「結果」フィールドに"2"が設定されていた場合にも「作業 ID 完了通知」を送信する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで

作業 ID 完了通知応答

メッセージ ID	メッセージ名	説明
0x00010008	作業 ID 完了通知応答	SC-20 から送信される「作業 ID 完了通知」に対する応答メッセージ。 「作業 ID 完了通知応答」の受信で、一連の動作が完了する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

作業アイテムリストデータ通知

メッセージ ID	メッセージ名	説明
0x10010009	作業アイテムリストデータ通知	「作業アイテムリスト取得要求」に対して、作業 ID、作業指示、作業アイテム情報を通知するメッセージ。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	作業 ID			
0x0090	作業指示			
0x00d0	作業アイテム			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050 - 0x008f	64 Byte	char	作業 ID	半角英数字で、最大 50 文字まで
0x0090 - 0x00cf	64 Byte	char	作業指示	半角英数字で、最大 50 文字まで
0x00d0 - 0x010f	64 Byte	char	作業アイテム	半角英数字で、最大 50 文字まで

作業アイテムリスト取得完了通知

メッセージ ID	メッセージ名	説明
0x1001000B	作業アイテムリスト取得完了通知	すべての「作業アイテムリストデータ通知」の送信完了後に、対向機へ送信するメッセージ。 転送件数フィールドに送信したアイテムの件数を入力する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		転送件数	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	転送件数	1-32767：総 Item 件数 -1：NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード（P.44）参照

作業アイテムリスト取得完了通知応答

メッセージ ID	メッセージ名	説明
0x0001000B	作業アイテムリスト取得完了通知応答	SC-20 から送信される「作業アイテムリスト取得完了通知」に対する応答メッセージ。「作業アイテムリスト取得完了通知応答」で、一連の動作が完了する。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで

システム停止通知

メッセージ ID	メッセージ名	説明
0x1001000E	システム停止通知	SC-20 をシャットダウンまたは再起動した後に、対向機側に送信するメッセージ。 このメッセージに対する応答はしない。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	停止モード			

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	4 Byte	uint32	停止モード	0：シャットダウン 1：再起動

タイムアウト通知

メッセージ ID	メッセージ名	説明
0x1001000F	タイムアウト通知	SC-20 内部で何らかの原因でメッセージが処理しきれなかった場合に対向機側に送信するメッセージ。 このメッセージに対する応答はしない。

メッセージフォーマット

アドレス	bit			
	31	16	15	0
0x0000	メッセージ ID			
0x0004	端末 ID			
0x0008	端末名			
0x0048	日	月	年	
0x004c	reserve	秒	分	時
0x0050	エラーコード		結果	

アドレス	サイズ	属性	フィールド名称	説明
0x0000	4 Byte	uint32	メッセージ ID	メッセージ固有の ID
0x0004	4 Byte	uint32	端末 ID	機体固有の ID
0x0008 - 0x0047	64 Byte	char	端末名	半角英数字で、最大 50 文字まで
0x0048	2 Byte	uint16	年	SC-20 の時間を設定
0x004a	1 Byte	uchar	月	SC-20 の時間を 1～12 で設定
0x004b	1 Byte	uchar	日	SC-20 の時間を 1～31 で設定
0x004c	1 Byte	uchar	時	SC-20 の時間を 0～23 で設定
0x004d	1 Byte	uchar	分	SC-20 の時間を 0～59 で設定
0x004e	1 Byte	uchar	秒	SC-20 の時間を 0～59 で設定
0x004f	1 Byte	uchar	reserve	未使用領域
0x0050	2 Byte	int16	結果	0：OK -1：NG
0x0052	2 Byte	uint16	エラーコード	4.エラーコード (P.44) 参照

4. エラーコード

エラーコード	エラー名	原因	対象メッセージ	対処方法
1	不明な端末 ID	SC-20 に送られた端末 ID と設定の端末 ID が不一致	要求メッセージ全般	SC-20 で、送信したメッセージの端末 ID、端末名と一致しているか確認してください。 端末名の大文字小文字が間違っていないか確認してください。
2	不明な端末名	SC-20 に送られた端末名と設定の端末名が不一致	要求メッセージ全般	
101	ステータス移行失敗	待機状態以外で作業 ID 開始要求を受信	作業 ID 開始要求	「作業 ID 開始要求」、「作業 ID 実行要求」はログイン後、または作業 ID 完了後のみ受け付けます。 マッチング中や「作業 ID 完了応答」前に送信している可能性があります。
102		待機状態以外で作業 ID 実行要求を受信	作業 ID 実行要求	
103	ステータス移行失敗	準備状態以外で開始要求を受信	開始要求	開始要求は「作業 ID 開始要求」後または「作業アイテム完了応答」後のみ受け付けます。 作業 ID 実行中は受け付けません。
104	ステータス移行失敗	実行状態以外で停止要求を受信	停止要求	停止要求はマッチング中のみ受け付けます。停止要求よりも先にマッチングが完了した場合も、このエラーコードが表示されます。
105	ステータス移行失敗	待機状態以外でアイテム取得要求を受信	作業アイテム取得要求	アイテム取得要求はログイン後、または作業 ID 完了後のみ受け付けます。 マッチング中や「作業 ID 完了応答」前に送信している可能性があります。
106	ステータス移行失敗	作業者モードでアイテム取得要求を受信	作業アイテム取得要求	アイテム取得要求は管理者モードでのみ要求可能です。管理者モードでログインしてください。
107	ステータス移行失敗	待機状態以外で作業 ID 変更要求を受信	作業 ID 変更要求	アイテム取得要求はログイン後、または作業 ID 完了後のみ受け付けます。 マッチング中や「作業 ID 完了応答」前に送信している可能性があります。
108	EXTIN 入力	マッチング中以外で EXTIN 入力要求を受信	EXTIN 入力要求	EXTIN 入力要求はマッチング中のみ受け付けます。

エラーコード	エラー名	原因	対象メッセージ	対処方法
109	ログアウト中	ユーザーログアウト中に要求メッセージを受信	要求メッセージ全般	ログインを実施してください。
201	作業 ID 名不一致	指定した作業 ID が存在しない	開始要求 作業 ID 実行要求	要求に入力された作業 ID、指示リスト、アイテムが一致しませんでした。 SC-20 で、作業名が変更されていないか確認してください。 先頭や末尾に不要なスペースなどを入力していないか確認してください。
202	作業指示 リスト名不一致	指定した作業指示が存在しない	開始要求 作業 ID 実行要求	
203	作業アイテム名 不一致	指定した作業アイテムが存在しない	開始要求 作業 ID 実行要求	
204	作業 ID 名空欄	作業 ID 名が指定されていない	開始要求 作業 ID 実行要求	指定フォーマットに作業 ID 名を入力してください。
207	ビジー状態	カメラが処理中で開始に失敗	作業 ID 開始要求	時間を置いて再度要求して下さい。
208	ビジー状態	カメラが処理中で変更に失敗	作業 ID 変更要求	時間を置いて再度要求して下さい。
209	ビジー状態	カメラが連続した要求に対してタイムアウトが発生	作業 ID 開始要求、 作業 ID 実行要求	時間を置いて再度要求して下さい。
210	Extin Input	入力する IO が不正、もしくはマッチングが停止中	Extin 入力要求	マッチング画面で正常な範囲内の値を入力してください。
301	マッチング結果 生成失敗	マッチング結果の生成に失敗	作業アイテム完了通知	マッチングデータの生成に失敗した場合に表示します。
401	タイムアウト	通知に対しての応答が受信できない	タイムアウト通知	「作業 ID 開始応答」、「作業アイテム完了通知」、「作業 ID 完了通知」を送ってから 3 秒間応答が無い場合に表示します。
550	コネクションエラー NomachIPError	不明な IP アドレスからのデータを受信しました。	メッセージ全般	設定した IP アドレスから送信を行ってください。

5.サンプルコード

C 言語によるサンプルコードを以下に記載します、本コードは既に SC-20 本体に作業 ID「Default」、リスト「Work_1」、アイテム「Item_1」がマッチングモードとして登録されているものとして、作業 ID 実行 1 パス分のサンプル（P.12 作業 ID 実行処理）を記したものになります。
また、本コードは Windows ベースで winsock2 を使用したものになりますので、「ws2_32.lib」ライブラリのリンクが必要になります。当ライブラリは Visual Studio などをインストールすることで入手できます。

client モード

```
#include <stdio.h>
#include <winsock2.h>       // need ws2_32.lib
#include <ws2tcpip.h>

// char* -> short
short char2short(char* c)
{
    short ret = 0;
    ret |= ((short)c[0] & 0x00FF);
    ret |= (((short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> unsigned short
unsigned short char2ushort(char* c)
{
    unsigned short ret = 0;
    ret |= ((unsigned short)c[0] & 0x00FF);
    ret |= (((unsigned short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> double
double char2double(char* c)
{
    double ret = 0.0;
    memcpy(&ret, c, 8);
    return ret;
}

// main
int main()
{
    int ret = 0;
    unsigned short port = 56109; // IP Port Number (Default=56109)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[]= "SC20"; // Terminal Name (Default="SC20")
    SOCKET srcSocket;  // My PC Socket
    SOCKET dstSocket;  // SC20 Socket

    // sockaddr_in
    struct sockaddr_in srcAddr;
    struct sockaddr_in dstAddr;
    int dstAddrSize = sizeof(dstAddr);
```

```
// buffer
char rcvBuffer[4096];
char tmpBuffer[128];
char sndBuffer1[0x0188];
char sndBuffer2[0x004c];

// Windows Only Process
WSADATA data;
WSAStartup(MAKEWORD(2,0), &data);

// sockaddr_in Setting
memset(&srcAddr, 0, sizeof(srcAddr));
srcAddr.sin_port = htons(port);
srcAddr.sin_family = AF_INET;
srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);

// My PC Socket Create
srcSocket = socket(AF_INET, SOCK_STREAM, 0);

// My PC Socket Bind
bind(srcSocket, (struct sockaddr *) &srcAddr, sizeof(srcAddr));

// Connect Listen
listen(srcSocket, 1);

// Connect Accept <- SC20
printf("Waiting for connection ...¥n");
dstSocket = accept(srcSocket, (struct sockaddr *) &dstAddr, &dstAddrSize);
printf("Connected from %s¥n¥n", inet_ntoa(dstAddr.sin_addr));

////////////////////////////////////////
// Work ID Execute Command
////////////////////////////////////////

// 0x00000005 Request Event Buffer
memset(sndBuffer1, 0, 0x0188);
sndBuffer1[0x0000] = 0x05;
sndBuffer1[0x0001] = 0x00;
sndBuffer1[0x0002] = 0x00;
sndBuffer1[0x0003] = 0x00;
sndBuffer1[0x0004] = (char)((termID & 0x000000FF) >>  0);
sndBuffer1[0x0005] = (char)((termID & 0x0000FF00) >>  8);
sndBuffer1[0x0006] = (char)((termID & 0x00FF0000) >> 16);
sndBuffer1[0x0007] = (char)((termID & 0xFF000000) >> 24);
sprintf(&sndBuffer1[0x0008], termName);
sprintf(&sndBuffer1[0x0048], "Default");
sprintf(&sndBuffer1[0x0088], "Work_1");
sprintf(&sndBuffer1[0x00C8], "Item_1");
sprintf(&sndBuffer1[0x0108], "User");
sprintf(&sndBuffer1[0x0148], "1234567890");

// 0x00000005 Request Send -> SC-20
ret = send(dstSocket, sndBuffer1, 0x0188, 0);
if(0 > ret){
    printf("Socket Send Error¥n");
    return -1;
```

```
    }


    // 0x10000005 Response Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
    }

    // Response  Buffur Output
    printf("¥n----------------------------------------------------------¥n");
    printf("@ Work ID Execute Command Response¥n");
    printf("----------------------------------------------------------¥n¥n");
    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName  : %s¥n", tmpBuffer);
    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                        , rcvBuffer[0x4A]
                                                        , rcvBuffer[0x4B]
                                                        , rcvBuffer[0x4C]
                                                        , rcvBuffer[0x4D]
                                                        , rcvBuffer[0x4E]
    );
    short result = ((short)rcvBuffer[0x51] << 8) | (short)rcvBuffer[0x50];
    printf("Result    : %d¥n", result);
    printf("ErrorCode : %d¥n", (((unsigned short)rcvBuffer[0x53] & 0xFF) << 8) | ((unsigned short)rcvBuffer[0x52] &
0xFF));
    if(0 > result){
        printf("0x00000005 Command Error¥n");
        return -1;
    }

    ////////////////////////////////////
    // Work Item Done (Matching) Notify
    ////////////////////////////////////

    // 0x10010002 Notify Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
    }

    // Work Item Done (Matching) Buffer Output
    printf("¥n----------------------------------------------------------¥n");
    printf("@ Work Item Done (Matching) Notify¥n");
    printf("----------------------------------------------------------¥n¥n");
    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName  : %s¥n", tmpBuffer);
    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                        , rcvBuffer[0x4A]
                                                        , rcvBuffer[0x4B]
```

```
                                                    , rcvBuffer[0x4C]
                                                    , rcvBuffer[0x4D]
                                                    , rcvBuffer[0x4E]
    );
    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
    printf("WorkID    : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x0090]);
    printf("WorkList  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x00D0]);
    printf("WorkItem  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x0110]);
    printf("WorkerID  : %s¥n", tmpBuffer);
    strcpy(tmpBuffer, &rcvBuffer[0x01D8]);
    printf("WorkNum   : %s¥n", tmpBuffer);
    printf("Result    : %d¥n", char2short(&rcvBuffer[0x02A0]));
    printf("Time      : %d¥n", char2ushort(&rcvBuffer[0x02A2]));
    printf("BaseScore : %f¥n", char2double(&rcvBuffer[0x02A4]));
    printf("BaseAngle : %d¥n", char2short(&rcvBuffer[0x02AC]));
    int checkNum = (int)char2short(&rcvBuffer[0x02AE]);
    printf("CheckNum  : %d¥n", checkNum);
    for(int i = 0; i < checkNum; i++){
        int offset = i * 16;
        printf("CheckIndex[%d]¥n", rcvBuffer[0x02B0 + offset]);

        printf("  Mode    : %d¥n", rcvBuffer[0x02B1 + offset]);
        printf("  Result  : %d¥n", char2short(&rcvBuffer[0x02B2]));
        printf("  Angle   : %d¥n", char2short(&rcvBuffer[0x02B4]));
        printf("  Time    : %d¥n", char2ushort(&rcvBuffer[0x02B6]));
        printf("  Score   : %f¥n", char2double(&rcvBuffer[0x02B8]));
    }

    // 0x00010007 Work Item Done (Matching) Notify Response Send -> SC-20
    memset(sndBuffer2, 0, 0x004c);
    sndBuffer2[0x0000] = 0x07;
    sndBuffer2[0x0001] = 0x00;
    sndBuffer2[0x0002] = 0x01;
    sndBuffer2[0x0003] = 0x00;
    sndBuffer2[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer2[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer2[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer2[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer2[0x0008], termName);
    ret = send(dstSocket, sndBuffer2, 0x004c, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
        return -1;
    }

    ////////////////////////////////////////
    // Work ID Done Notify
    ////////////////////////////////////////

    // 0x10010008 Notify Recieve <- SC-20
    ret = recv(dstSocket, rcvBuffer, 4096, 0);
    if(0 >= ret){
        printf("Socket Recieve Error¥n");
        return -1;
```

```
    }
    int len = 0;
    printf("¥n----------------------------------------------------------¥n");
    printf("@ Work ID Done Notify¥n");
    printf("----------------------------------------------------------¥n¥n");
    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) | ((unsigned
int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
    strcpy(tmpBuffer, &rcvBuffer[8]);
    printf("TermName  : %s¥n", tmpBuffer);
    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                        , rcvBuffer[0x4A]
                                                        , rcvBuffer[0x4B]
                                                        , rcvBuffer[0x4C]
                                                        , rcvBuffer[0x4D]
                                                        , rcvBuffer[0x4E]
    );
    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
    printf("WorkID    : %s¥n", tmpBuffer);

    // 0x00010008 Work ID Done Notify Response Send -> SC-20
    memset(sndBuffer2, 0, 0x004C);
    sndBuffer2[0x0000] = 0x08;
    sndBuffer2[0x0001] = 0x00;
    sndBuffer2[0x0002] = 0x01;
    sndBuffer2[0x0003] = 0x00;
    sndBuffer2[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer2[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer2[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer2[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer2[0x0008], termName);
    ret = send(dstSocket, sndBuffer2, 0x0048, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
        return -1;
    }

    // Close Scocket
    closesocket(dstSocket);
    closesocket(srcSocket);

    // Windows Only
    WSACleanup();

    return 0;
}
```

client/server モード

```
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <thread>

#pragma warning(disable:4996)

bool commandReady = false;

// char* -> short
short char2short(char* c)
{
    short ret = 0;
    ret |= ((short)c[0] & 0x00FF);
    ret |= (((short)c[1] << 8) & 0xFF00);
    return ret;
}


// char* -> unsigned short
unsigned short char2ushort(char* c)
{
    unsigned short ret = 0;
    ret |= ((unsigned short)c[0] & 0x00FF);
    ret |= (((unsigned short)c[1] << 8) & 0xFF00);
    return ret;
}

// char* -> double
double char2double(char* c)
{
    double ret = 0.0;
    memcpy(&ret, c, 8);
    return ret;
}

// send function (Client)
int sendFunction(unsigned int reqID)
{
    int ret = 0;
    unsigned short port = 56109; // IP Port Number (Server/Client Mode SC20's IP Port = 56109(fixed))
    const char ipAddr[] = "192.168.1.8"; // IP Address String (depend SC20's Setting)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[] = "SC20"; // Terminal Name (Default="SC20")
    char sndBuffer[0x0188];
    int sndBufferSize;

    SOCKET mySocket;  // My PC Socket

    // My PC Socket Create
    mySocket = socket(AF_INET, SOCK_STREAM, 0);

    // My PC Client Setting
    struct sockaddr_in myAddr;
```

```
memset(&myAddr, 0, sizeof(myAddr));
myAddr.sin_port = htons(port);
myAddr.sin_family = AF_INET;
myAddr.sin_addr.s_addr = inet_addr(ipAddr);

// Connect To SC20
connect(mySocket, (struct sockaddr*) &myAddr, sizeof(myAddr));

// Send Buffer Setting (depend Reqest ID)
switch(reqID){

// Work ID Execute Command Buffer Sample
memset(sndBuffer, 0, 0x0188);
case 0x00000005:
    sndBufferSize = 0x0188;
    sndBuffer[0x0000] = 0x05;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x00;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
    sprintf(&sndBuffer[0x0048], "Default");
    sprintf(&sndBuffer[0x0088], "Work_1");
    sprintf(&sndBuffer[0x00C8], "Item_1");
    sprintf(&sndBuffer[0x0108], "User");
    sprintf(&sndBuffer[0x0148], "1234567890");
    break;

// Work Item Done (Matching) Notify Response Send -> SC-20 Sample
case 0x00010007:
    sndBufferSize = 0x004C;
    sndBuffer[0x0000] = 0x07;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x01;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
    break;

// 0x00010008 Work ID Done Notify Response Send -> SC-20 Sample
case 0x00010008:
    sndBufferSize = 0x004C;
    sndBuffer[0x0000] = 0x08;
    sndBuffer[0x0001] = 0x00;
    sndBuffer[0x0002] = 0x01;
    sndBuffer[0x0003] = 0x00;
    sndBuffer[0x0004] = (char)((termID & 0x000000FF) >>  0);
    sndBuffer[0x0005] = (char)((termID & 0x0000FF00) >>  8);
    sndBuffer[0x0006] = (char)((termID & 0x00FF0000) >> 16);
    sndBuffer[0x0007] = (char)((termID & 0xFF000000) >> 24);
    sprintf(&sndBuffer[0x0008], termName);
```

```
        break;
    }

    // Send Buffer -> SC20
    ret = send(mySocket, sndBuffer, sndBufferSize, 0);
    if(0 > ret){
        printf("Socket Send Error¥n");
    }

    closesocket(mySocket);
    return ret;
}

// receive thread (Server)
void receiveThread(void)
{
    unsigned short port = 56109; // IP Port Number (Default=56109)
    unsigned int termID = 2030446878; // Terminal ID (unique)
    char termName[] = "SC20"; // Terminal Name (Default="SC20")
    SOCKET srcSocket;  // My PC Socket
    SOCKET dstSocket;  // SC20 Socket

    // sockaddr_in
    struct sockaddr_in srcAddr;
    struct sockaddr_in dstAddr;
    int dstAddrSize = sizeof(dstAddr);

    // buffer
    char rcvBuffer[4096];
    char tmpBuffer[4096];

    // sockaddr_in Setting
    memset(&srcAddr, 0, sizeof(srcAddr));
    srcAddr.sin_port = htons(port);
    srcAddr.sin_family = AF_INET;
    srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    // My PC Socket Create
    srcSocket = socket(AF_INET, SOCK_STREAM, 0);

    // My PC Socket Bind
    bind(srcSocket, (struct sockaddr *) &srcAddr, sizeof(srcAddr));

    // Receive Loop
    while(1){

        // Connect Listen
        listen(srcSocket, 1);

        // Connect Accept <- SC20
        printf("Waiting for connection ...¥n");
        dstSocket = accept(srcSocket, (struct sockaddr *) &dstAddr, &dstAddrSize);
        printf("Connected from %s¥n¥n", inet_ntoa(dstAddr.sin_addr));

        // Receive Buffer <- SC20
        int ret = recv(dstSocket, rcvBuffer, 4096, 0);
        if(0 >= ret){
```

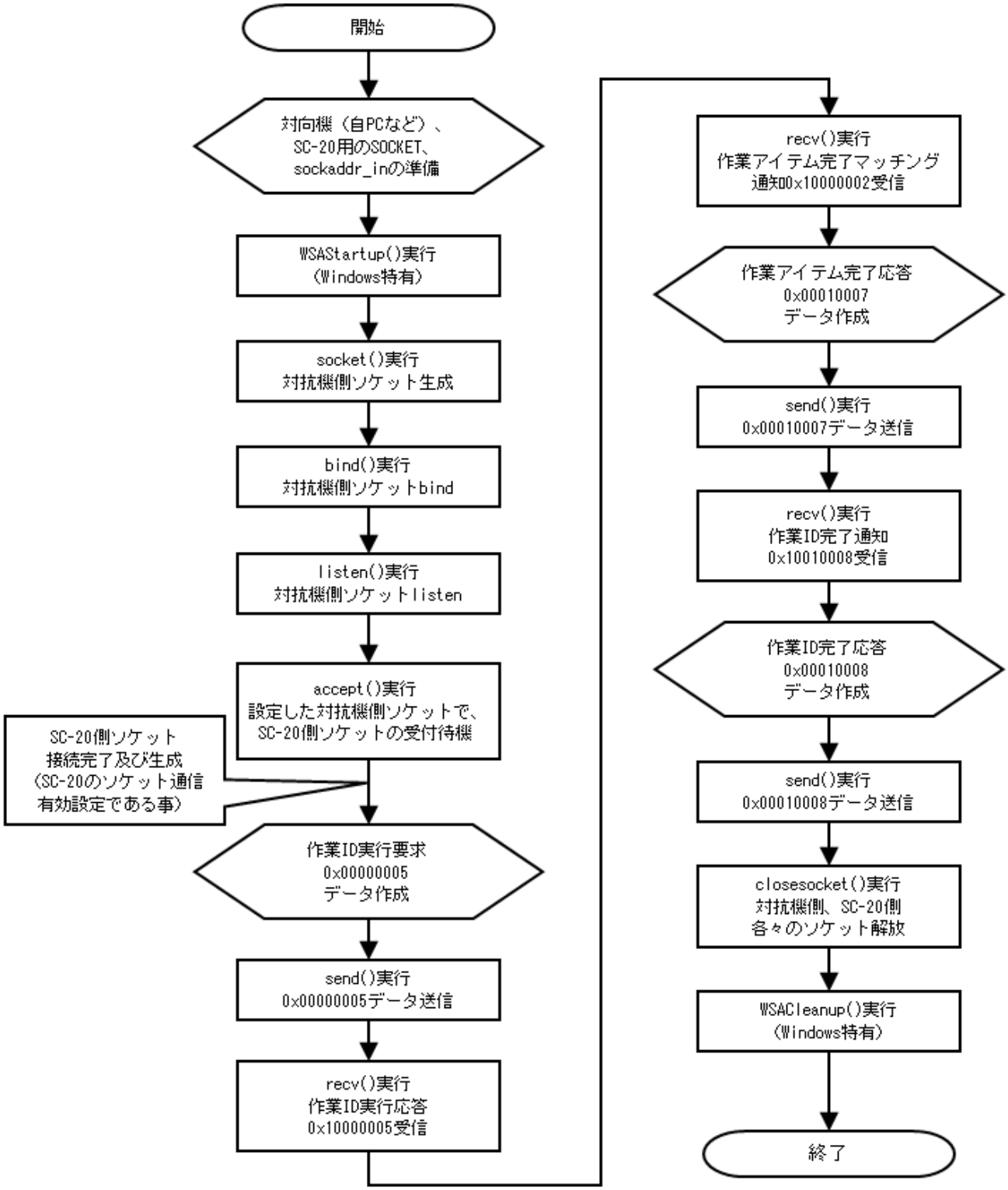
```
            printf("Socket Recieve Error¥n");
            break;
        }
        else{
            // resID Check
            unsigned int resID = ((unsigned int)rcvBuffer[3] << 24) | ((unsigned int)rcvBuffer[2] << 16) | ((unsigned
int)rcvBuffer[1] << 8) | ((unsigned int)rcvBuffer[0]);
            switch(resID){

            // Work ID Execute Command Response
            case 0x10000005:
                {
                    printf("¥n-----------------------------------------------------------¥n");
                    printf("@ Work ID Execute Command Response¥n");
                    printf("-----------------------------------------------------------¥n¥n");
                    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                    strcpy(tmpBuffer, &rcvBuffer[8]);
                    printf("TermName  : %s¥n", tmpBuffer);
                    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                        , rcvBuffer[0x4A]
                                                                        , rcvBuffer[0x4B]
                                                                        , rcvBuffer[0x4C]
                                                                        , rcvBuffer[0x4D]
                                                                        , rcvBuffer[0x4E]
                    );
                    short result = ((short)rcvBuffer[0x51] << 8) | (short)rcvBuffer[0x50];
                    printf("Result    : %d¥n", result);
                    printf("ErrorCode : %d¥n", (((unsigned short)rcvBuffer[0x53] & 0xFF) << 8) | ((unsigned
short)rcvBuffer[0x52] & 0xFF));
                    break;
                }

            // Work Item Done (Matching) Notify
            case 0x10010002:
                {
                    printf("¥n-----------------------------------------------------------¥n");
                    printf("@ Work Item Done (Matching) Notify¥n");
                    printf("-----------------------------------------------------------¥n¥n");
                    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                    strcpy(tmpBuffer, &rcvBuffer[8]);
                    printf("TermName  : %s¥n", tmpBuffer);
                    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                        , rcvBuffer[0x4A]
                                                                        , rcvBuffer[0x4B]
                                                                        , rcvBuffer[0x4C]
                                                                        , rcvBuffer[0x4D]
                                                                        , rcvBuffer[0x4E]
                    );
                    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
                    printf("WorkID    : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x0090]);
                    printf("WorkList  : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x00D0]);
```

```
                    printf("WorkItem  : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x0110]);
                    printf("WorkerID  : %s¥n", tmpBuffer);
                    strcpy(tmpBuffer, &rcvBuffer[0x01D8]);
                    printf("WorkNum   : %s¥n", tmpBuffer);
                    printf("Result    : %d¥n", char2short(&rcvBuffer[0x02A0]));
                    printf("Time      : %d¥n", char2ushort(&rcvBuffer[0x02A2]));
                    printf("BaseScore : %f¥n", char2double(&rcvBuffer[0x02A4]));
                    printf("BaseAngle : %d¥n", char2short(&rcvBuffer[0x02AC]));
                    int checkNum = (int)char2short(&rcvBuffer[0x02AE]);
                    printf("CheckNum  : %d¥n", checkNum);
                    for(int i = 0; i < checkNum; i++){
                        int offset = i * 16;
                        printf("CheckIndex[%d]¥n", rcvBuffer[0x02B0 + offset]);
                        printf("  Mode    : %d¥n", rcvBuffer[0x02B1 + offset]);
                        printf("  Result  : %d¥n", char2short(&rcvBuffer[0x02B2]));
                        printf("  Angle   : %d¥n", char2short(&rcvBuffer[0x02B4]));
                        printf("  Time    : %d¥n", char2ushort(&rcvBuffer[0x02B6]));
                        printf("  Score   : %f¥n", char2double(&rcvBuffer[0x02B8]));
                    }

                    // Work Item Done (Matching) Notify Response Send -> SC20
                    sendFunction(0x00010007);

                    break;
                }

            // Work ID Done Notify
            case 0x10010008:
                {
                    printf("¥n-------------------------------------------------------¥n");
                    printf("@ Work ID Done Notify¥n");
                    printf("-------------------------------------------------------¥n¥n");
                    printf("MsgID     : 0x%02X%02X%02X%02X¥n", rcvBuffer[3], rcvBuffer[2], rcvBuffer[1], rcvBuffer[0]);
                    printf("TermID    : %d¥n", ((unsigned int)rcvBuffer[7] << 24) | ((unsigned int)rcvBuffer[6] << 16) |
((unsigned int)rcvBuffer[5] << 8) | ((unsigned int)rcvBuffer[4]));
                    strcpy(tmpBuffer, &rcvBuffer[8]);
                    printf("TermName  : %s¥n", tmpBuffer);
                    printf("Date      : %04d/%02d/%02d %02d:%02d:%02d¥n", char2short(&rcvBuffer[0x48])
                                                                        , rcvBuffer[0x4A]
                                                                        , rcvBuffer[0x4B]
                                                                        , rcvBuffer[0x4C]
                                                                        , rcvBuffer[0x4D]
                                                                        , rcvBuffer[0x4E]
                    );
                    strcpy(tmpBuffer, &rcvBuffer[0x0050]);
                    printf("WorkID    : %s¥n", tmpBuffer);

                    // Work ID Done Notify Response Send -> SC20
                    sendFunction(0x00010008);

                    Sleep(3000);
                    commandReady = true;
                    break;
                }
            }
        }
```

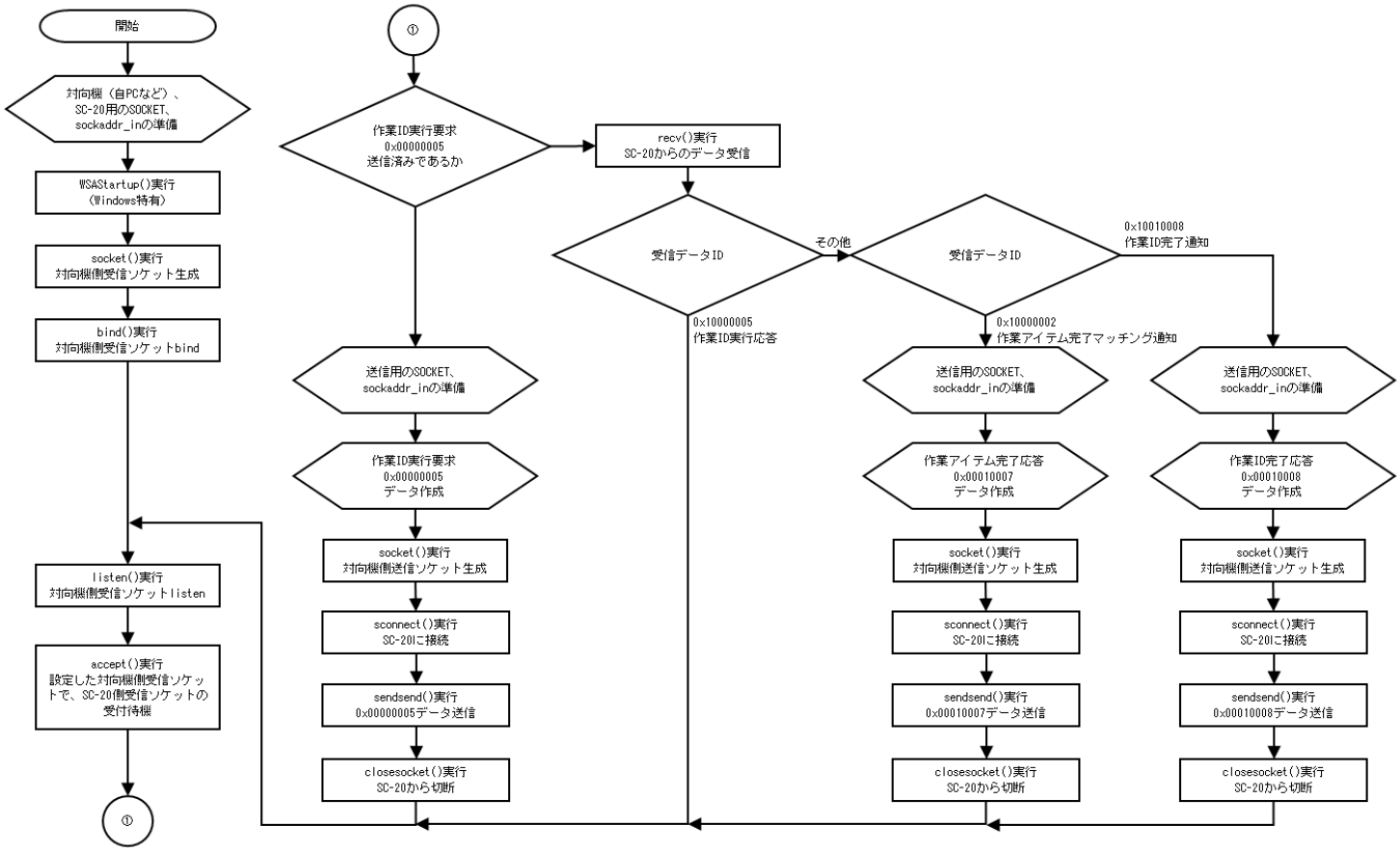
```
        // Close Scocket(dst)
        closesocket(dstSocket);
    }
    // Close Scocket(src)
    closesocket(srcSocket);
}

// main
int main()
{
    // Windows Only Process
    WSADATA data;
    WSAStartup(MAKEWORD(2,0), &data);

    // Receive Thread Start (My PC's Server)
    std::thread rcvThred(receiveThread);
    rcvThred.detach();

    // Thread Ready Wait
    Sleep(3000);
    commandReady = true;

    // Work ID Execute Command Trigger
    while(1){
        if(true == commandReady){
            printf("¥n¥n>>>> Put Enter¥n");
            getchar();
            sendFunction(0x00000005);
            commandReady = false;
        }
        else{
            Sleep(1000);
        }
    }

    // Windows Only
    WSACleanup();

    return 0;
}
```

6. フローチャート

client モード



client/server モード



改訂履歴

Ver.	作成年月日	改版記事	備考
1.0	2023/06/14	・新規発行	
2.0	2024/03/28	・クライアント／サーバーモード関連の追記 ・通知系に存在しない ID を記述していた為削除 ・その他誤記修正	