

ハードウェアマニュアル
オプション

HIDIC
S10 シリーズ

ET.NET
(LWE550)

2α
シリーズ

対象機種

HIDIC-S10/2		NESP-S25E
HIDIC-S10/2	E	NESP-2 E
HIDIC-S10/2	H	NESP-2 H
HIDIC-S10/2	Hf	NESP-2 Hf

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問合わせください。

1997年 6月 (第1版) SAJ-2-124(A) (廃版)
1998年 3月 (第2版) SAJ-2-124(B) (廃版)
1999年 4月 (第3版) SAJ-2-124(C) (廃版)
2001年11月 (第4版) SAJ-2-124(D)

このマニュアルの一部、または全部を無断で転写したり複写することは、固くお断りいたします。
このマニュアルの内容を、改良のため予告なしに変更することがあります。

安全上のご注意

取付、運転、保守・点検の前に必ずこのマニュアルとその他の付属書類をすべて熟読し、正しくご使用ください。機器の知識、安全の情報そして注意事項のすべてについて熟読してご使用ください。また、このマニュアルは最終保守責任者のお手元に必ず届くようにしてください。

このマニュアルでは、安全上の注意事項のランクを「危険」「注意」として区分してあります。



：取り扱いを誤った場合に、危険な状況が起こりえて、死亡または重傷を受ける可能性が想定される場合。



：取り扱いを誤った場合に、危険な状況が起こりえて、中程度の障害や軽傷を受ける可能性が想定される場合および物的障害だけの発生が想定される場合。

なお、




に記載した事項でも、状況によっては重大な結果に結びつく可能性があります。

いずれも重要な内容を記載していますので必ず守ってください。

禁止、強制の絵表示の説明を次に示します。

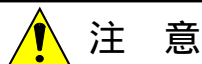


：禁止（してはいけないこと）を示します。例えば分解禁止の場合は  となります。



：強制（必ずしなければならないこと）を示します。例えば接地の場合は  となります。

1. 取付について



カタログ、マニュアルに記載の環境で使用してください。

高温、多湿、じんあい、腐食性ガス、振動、衝撃がある環境で使用すると感電、火災、誤動作の原因となることがあります。

マニュアルにしたがって取り付けてください。

取り付けに不備があると落下、故障、誤動作の原因となることがあります。

電線くずなどの異物を入れないでください。

火災、故障、誤動作の原因となることがあります。

2. 配線について

強 制

必ず接地（FG）を行ってください。
接地しない場合は、感電、誤動作のおそれがあります。

注 意

定格にあった電源を接続してください。
定格と異なった電源を接続すると火災の原因になることがあります。
配線作業は、資格のある専門家が行ってください。
配線を誤ると火災、故障、感電のおそれがあります。
トランシーバケーブルおよび同軸ケーブルは強電機器からの配線と同一付線しないでください。強電機器と同一付線した場合、誤動作の原因となることがあります。

3. 使用上の注意

危 険

通電中は端子に触れないでください。
感電のおそれがあります。
非常停止回路、インタロック回路等はプログラマブルコントローラの外部で構成してください。
プログラマブルコントローラの故障により、機械の破損や事故のおそれがあります。



注 意

運転中のプログラム変更、強制出力、RUN、STOP等の操作は十分安全を確認して行ってください。

操作ミスにより、機械の破損や事故のおそれがあります。

電源投入順序にしたがって投入してください。

誤動作により、機械の破損や事故のおそれがあります。

このモジュールの近くではトランシーバ、携帯電話などを使用しないでください。近くでトランシーバ、携帯電話などをご使用になりますとノイズにより、誤動作、モジュールダウンとなるおそれがあります。

4. 保守について



禁 止

分解、改造はしないでください。

火災、故障、誤動作の原因となります。



注 意

モジュール/ユニットの脱着は電源をOFFしてから行ってください。

感電、誤動作、故障の原因となることがあります。

保証・サービス

特別な保証契約がない場合において、この製品の保証は次の通りです。

1. 保証期間と保証範囲

【保証期間】

この製品の保証期間は、ご注文のご指定場所に納入後1年といたします。

【保証範囲】

上記保証期間中に、このマニュアルに従った製品仕様範囲内の正常な使用状態で故障を生じた場合は、その機器の故障部分をお買上げの販売店または（株）日立エンジニアリングサービスにお渡しください。交換または修理を無償で行います。ただし、郵送いただく場合は、郵送料金、梱包費用はご注文主のご負担となります。

次のいずれかに該当する場合は、この保証の対象範囲から除外いたします。

製品仕様範囲外の取扱い、ならびに使用により故障した場合。

納入品以外の事由により故障した場合。

納入者以外の改造、または修理により故障した場合。

リレーなどの消耗部品の寿命により故障した場合。

上記以外の天災、災害など、納入者側の責任にあらざる事由により故障した場合。

ここでいう保証とは、納入した製品単体の保証を意味します。したがって、当社ではこの製品の運用および故障を理由とする損失、逸失利益等の請求につきましては、いかなる責任も負いかねますのであらかじめご了承ください。また、この保証は日本国内でのみ有効であり、ご注文主に対して行うものです。

2. サービスの範囲

納入した製品の価格には技術者派遣などのサービス費用は含まれておりません。次に該当する場合は別個に費用を申し受けます。

取付け調整指導および試運転立ち会い。

保守点検および調整。

技術指導、技術教育、およびトレーニングスクール。

保証期間後の調査および修理。

保証期間中においても、上記保証範囲外の事由による故障原因の調査。

はじめに

このたびは、CPUオプション ET.NETモジュールをご利用いただきましてありがとうございます。

この「ハードウェアマニュアル オプションET.NET」は、ET.NETモジュールの取扱いについて述べたものです。このマニュアルをお読みいただき、正しくご使用いただくようお願いいたします。

このモジュールを使用する場合は、下記バージョンのシステムをご使用ください。

下記バージョンより古いシステムでは、ET.NETモジュールが正常に動作しません。

<システム>

対象ツール	システムF/D名称	バージョン
PSE	LADDER SYSTEM	Ver 5.0 Rev 5.0 以降
	Compact PMS SYS	Ver 5.0 Rev 5.0 以降
PC98	ラダーOS ロードシステム	Ver 4.3 Rev 5.0 以降
	CPMS ロードシステム	Ver 4.3 Rev 5.0 以降
	CPMSE ロードシステム	Ver 2.3 Rev 5.0 以降

<ツールソフト>

ET.NETモジュールのREV **B**(フロントケース左下に **B**シール貼付)以降の製品を使用される場合は、PSE 用“ET.NET SUPPORT”F/DのV1.0, R1.0以降のバージョン、レビジョンF/Dと組合せて使用してください。V1.0, R0.0を使用されますと、ET.NETモジュールの物理アドレスが正しく表示されません。

ET.NETモジュールとMicrosoft® Windows® 95 operating system版プログラミングツールとを接続する場合、ET.NETモジュールのREV (フロントケース左下にシール貼付)によって、パソコン上で同時に開けるS10/2 シリーズCPUのデータもしくはプログラムのモニタ画面は下記となります。

REV **B**以前：1画面のみ開けます。

REV **C**以降：4画面まで開けます。

S10/2 とWindows®版プログラミングツールをET.NETモジュールを介して接続する場合、下記条件にてS10/2 に同時に4台までのプログラミングツール(ラダー図もしくはHI-FLOWシステム)を接続できます。下記条件以外では、同時に1台までしか接続できませんので、注意してください。

- (1) ET.NETモジュール(LWE550)のREVが **H** 以降(ケース左下に貼ってあるシールが **H** 以降か、CPUインディケータ表示が“ETM 4.1”もしくは“ETS 4.1”以降であることを確認してください。)
- (2) ラダー図システムまたはHI-FLOWシステムのバージョン、レビジョンが07-00以降。
(ただし、ラダー図システムは1台のみとし、その他はモニタ専用ラダー図システムであること。
HI-FLOWの場合も、HI-FLOWシステムは1台のみでそれ以外はモニタ専用HI-FLOWであること。)

NESP (Nissan Electronic Sequence Processor) シリーズをご使用のユーザは下記対応表を参照の上ご使用ください。

【HIDIC - S10 シリーズ】		【NESP - S25シリーズ】
HIDIC - S10 / 2	NESP - S25 E
HIDIC - S10 / 2 E	NESP - 2 E
HIDIC - S10 / 2 H	NESP - 2 H
HIDIC - S10 / 2 H f	NESP - 2 H f

< 記憶容量の計算値についての注意 >

2ⁿ計算値の場合 (メモリ容量・所要量、ファイル容量・所要量など)

1KB (キロバイト) = 1,024バイトの計算値です。

1MB (メガバイト) = 1,048,576バイトの計算値です。

1GB (ギガバイト) = 1,073,741,824バイトの計算値です。

10ⁿ計算値の場合 (ディスク容量など)

1KB (キロバイト) = 1,000バイトの計算値です。

1MB (メガバイト) = 1,000²バイトの計算値です。

1GB (ギガバイト) = 1,000³バイトの計算値です。

* AVE - TCPはACCESS CO., LTDの商標です。

目 次

1	ご使用にあたり	1
1.1	CPUマウントベース	2
1.2	オプションモジュールの実装	2
1.3	アース配線	4
2	仕 様	5
2.1	用 途	6
2.2	仕 様	6
2.2.1	システム仕様	6
2.2.2	回線仕様	6
3	各部の名称と機能、配線	7
3.1	各部の名称と機能	8
3.2	配 線	9
4	利用の手引き	11
4.1	10BASE-5のシステム構成	12
4.2	S10/2 によるシステム構成例	18
4.3	システム定義情報	19
4.3.1	物理アドレス	19
4.3.2	IPアドレス	19
4.3.3	サブネットマスク	21
4.4	ET.NETのソフトウェア構成	22
4.5	ET.NETのシステムプログラム	23
4.5.1	ソケットハンドラ	23
4.5.2	ソケットドライバ	23
4.5.3	TCPプログラム	23
4.5.4	UDPプログラム	24
4.5.5	IPプログラム	24
4.5.6	ドライバ	24
4.6	ユーザの作成するプログラム	25
4.6.1	ユーザプログラム	25
4.7	ソケットハンドラ	26
4.7.1	ソケットハンドラ一覧	27
4.8	ソケットハンドラ発行手順例	51

4.8.1	TCP/IPプログラム使用例	51
4.8.2	UDP/IPプログラム使用例	52
5	プログラム例	55
5.1	ソケットハンドラによるCPU間通信プログラム例	56
5.1.1	システム構成	56
5.1.2	プログラム構成	57
5.1.3	CPU01側プログラムのフローチャート	58
5.1.4	CPU01側のC言語プログラム例	60
5.1.5	CPU02側プログラムのフローチャート	62
5.1.6	CPU02側のC言語プログラム例	63
5.2	ソケットハンドラによるCPU間連続通信プログラム例	65
5.2.1	システム構成	65
5.2.2	プログラム構成	66
5.2.3	CPU01側プログラムのフローチャート	67
5.2.4	CPU01側のC言語プログラム例	69
5.2.5	CPU02側プログラムのフローチャート	71
5.2.6	CPU02側のC言語プログラム例	73
6	オペレーション	75
6.1	立上げ手順	76
6.2	PSEシステム立上げ	77
6.2.1	PSEシステム立上げ手順	77
6.2.2	PSEシステム基本オペレーション	79
6.3	モジュールのセットアップ	80
6.3.1	機能概要	80
6.3.2	オペレーション	80
7	保守	83
7.1	保守点検	84
7.1.1	定期点検	84
7.2	トラブルシューティング	85
7.2.1	手順	85
7.2.2	故障かな!?と思う前に	86
7.3	エラーと対策	88
7.3.1	PSEエラーコード表	88
7.3.2	CPU LED表示メッセージ表	89

7.3.3	ハードウェアエラー	90
7.3.4	ソケットハンドラ検出のエラーコード表	93
8	付 録	95
8.1	ネットワーク構成部品	96
8.1.1	LWE550とイーサネット*との接続の問題点	96
8.1.2	構成品一覧表	96
8.1.3	トランシーバ(タップ形) HLT-200TB	98
8.1.4	トランシーバ(コネクタ形) HLT-200	98
8.1.5	マルチポートトランシーバ H-7612-64	99
8.1.6	リピータ HLR-200H	99
8.1.7	同軸ケーブル(屋内用) HBN-CX-100	100
8.1.8	同軸コネクタ HBN-N-PC	101
8.1.9	中継コネクタ HBN-N-AJJ	102
8.1.10	ターミネータ(J形) HBN-T-NJ	102
8.1.11	ターミネータ(P形) HBN-T-NP	102
8.1.12	アース端子 HBN-G-TM	103
8.1.13	トランシーバケーブル HDC4360	103
8.1.14	変換器 HSN-9010	105
8.2	施工分担	106
8.3	同軸ケーブルの配線	107
8.3.1	ケーブルセグメントの布設	107
8.4	トランシーバ(コネクタ形)の設置・取付け	108
8.5	トランシーバ(タップ形)の設置・取付け	112
8.6	同軸コネクタの取付け	112
8.7	タップコネクタの取付け	114
8.8	トランシーバケーブルの取付け	116
8.9	ターミネータの取付け	116
8.10	リピータの設置・取付け	117
8.11	システムの接地	118
8.12	アース端子取付け方法	118
8.13	シングルポートトランシーバの設定	119
8.14	マルチポートトランシーバの設定および表示	120
8.15	CPUのメモリマップ	122
8.16	ET.NETモジュールのメモリマップ	123
8.17	トラブル調査書	124

目 次

図4 - 1	最小構成（リピータなし、セグメント長 最長500m）	13
図4 - 2	中規模構成（リピータ使用、トランシーバ間最長1,500m）	13
図4 - 3	大規模構成（リピータ、リンクセグメント使用、トランシーバ間最長2,500m）	14
図8 - 1	同軸ケーブルの構造	100
図8 - 2	同軸ケーブルの構造外観	101
図8 - 3	トランシーバケーブルのコネクタ仕様	104
図8 - 4	ケーブル断面図	104
図8 - 5	壁面設置例	109
図8 - 6	壁面設置例	110
図8 - 7	壁面設置例	110
図8 - 8	壁面設置例	110
図8 - 9	BOX内設置例	111
図8 - 10	BOX内設置例	111
図8 - 11	タップコネクタ組立図	114
図8 - 12	コネクタ、トランシーバ接続図	115

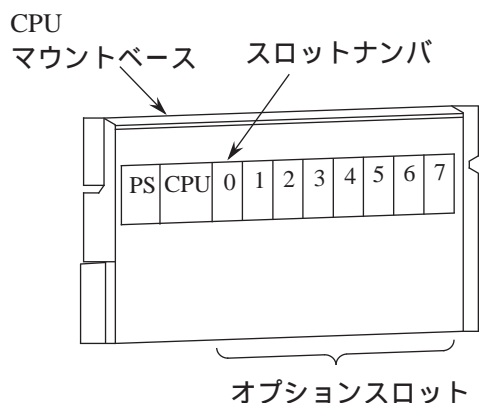
表 目 次

表 8 - 1	同軸ケーブルの構造	100
表 8 - 2	電気的特性	101
表 8 - 3	トランシーバケーブルの構造	103
表 8 - 4	トランシーバケーブルのピン配置	105
表 8 - 5	切替えスイッチの設定	121

1 ご使用にあたり

1 ご使用にあたり

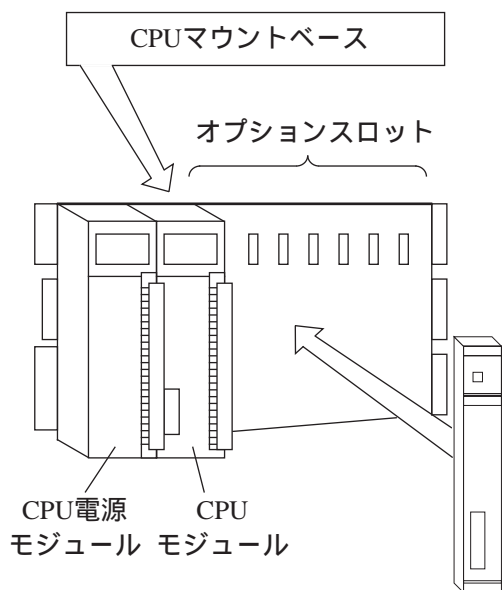
1.1 CPUマウントベース



CPUマウントベースには、以下の2種類があります。

- ・ 4スロットマウントベース（型式：HPC-1002）
 - ・ 8スロットマウントベース（型式：HPC-1000）
- 例えば、8スロットマウントベースの場合は、電源、CPUモジュール以外に1スロットタイプのモジュールを8モジュール、2スロットタイプのモジュールを4モジュールまで実装することができます。

1.2 オプションモジュールの実装



CPUマウントベース：HPC-1000

PSスロット：CPU電源モジュール（LWV000）を実装します。

CPUスロット：CPUモジュール（LWP000，040，070，075）を実装します。

LWP000	: 2
LWP040	: 2 E
LWP070	: 2 H
LWP075	: 2 Hf

スロット0~7：CPUオプションモジュールを実装します。

ET.NETモジュール
(スロット1, 3, 5, 7に左詰め実装)

⚠ 注意

ET.NETモジュールが実装できるスロットナンバは、1, 3, 5, 7の4スロットです。

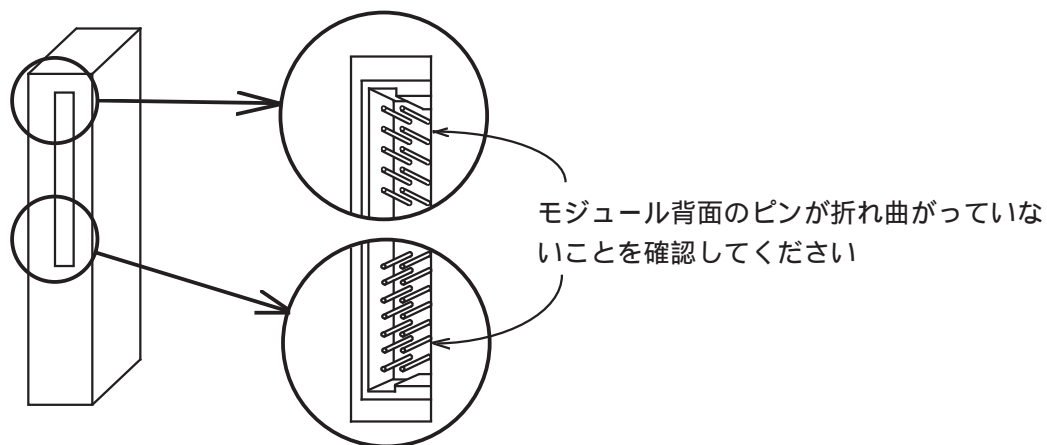
1, 3, 5, 7の空きスロットに左詰め実装してください。

ET.NETモジュールを1枚実装する場合は、必ずメインモジュールの設定にして使用してください。

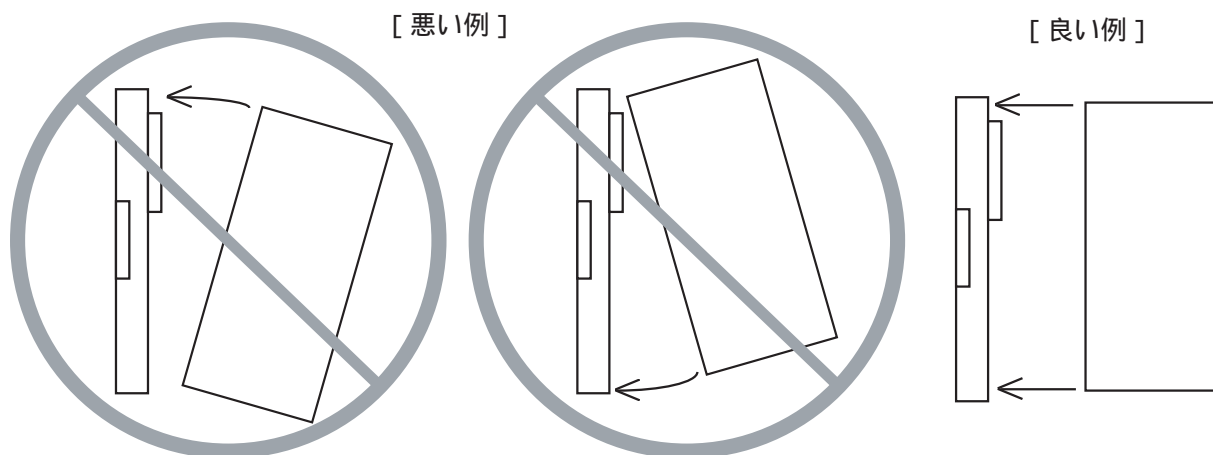
ET.NETモジュールを2枚使用する場合、どちらか一方は電源内蔵型トランシーバを接続してください。


オプションモジュール実装時は、以下のことに注意してください。

コネクタのピンが曲がっていないことを確認してください。



下図のように、オプションモジュールはCPUマウントベースに対して、正面からまっすぐ実装してください。[悪い例]のように斜めに実装すると、コネクタが破損しオプションモジュールが誤動作することがあります。

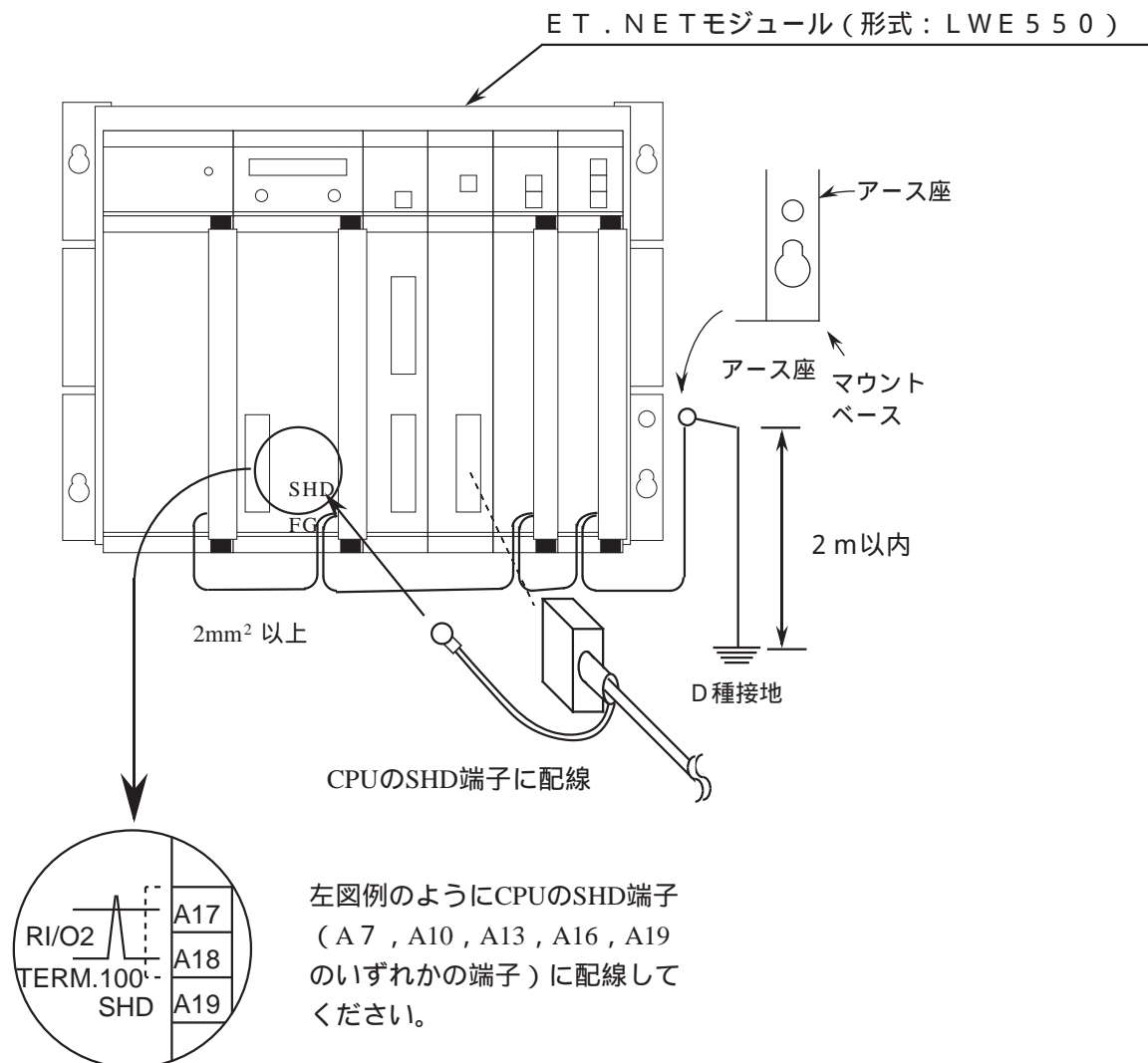


 注 意

キャビネットの構造上、CPUマウントベースが頭上の実装されている場合、モジュールは脚立などを使用してまっすぐに実装してください。

1 ご使用にあたり

1.3 アース配線



強制

FG (フレームグラウンド) のアース配線は、外部端子のある各モジュールのFG端子を、マウントベースのアース座に接続してください。アースの配線距離は2 m以内とし、マウントベースのアース座からD種接地してください。

アース線は、線径2mm²以上のものを用いてください。

トランシーバケーブルのシールド線をCPUモジュールのSHD端子 (A7, A10, A13, A16, A19のいずれかの端子) に配線してください。

2 仕 様

2 仕 様

2.1 用 途

ET.NETモジュール（型式：LWE550）は、IEEE802.3仕様に準拠したローカルエリアネットワークに接続し、TCP/IPまたは、UDP/IPプロトコルによるデータ通信を行います。

2.2 仕 様

2.2.1 システム仕様

項 目	仕 様
型式	LWE550
ET.NETモジュール最大実装枚数	2モジュール / CPU } 第1, 3, 5, 7の空き スロットにのみ左詰めで 実装可
モジュールスロット幅	1スロット幅モジュール
質量	380g

強 制

ET.NETモジュールを2枚使用する場合、どちらか一方は電源内蔵型トランシーバを接続してください。

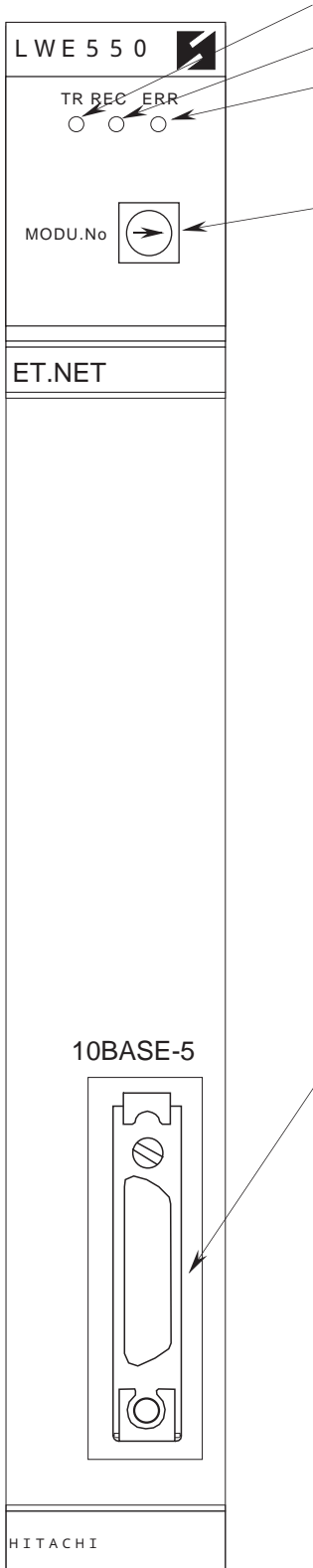
2.2.2 回線仕様

項 目	仕 様
伝送方式	直列伝送（ビットシリアル伝送）
電氣的インタフェース	IEEE802.3準拠（CSMA/CD準拠）
符号化方式	マンチェスタ符号方式
プロトコル	TCP/IP, UDP/IP
接続台数	10BASE-5：最大100台 / セグメント
ステーション台数	最大1024台 / ネットワーク
接続ケーブル	10BASE-5同軸ケーブル 最長500m / セグメント 10BASE-5トランシーバケーブル 最長 15m
データ伝送速度	10Mbps

3 各部の名称と機能、配線

3 各部の名称と機能、配線

3.1 各部の名称と機能



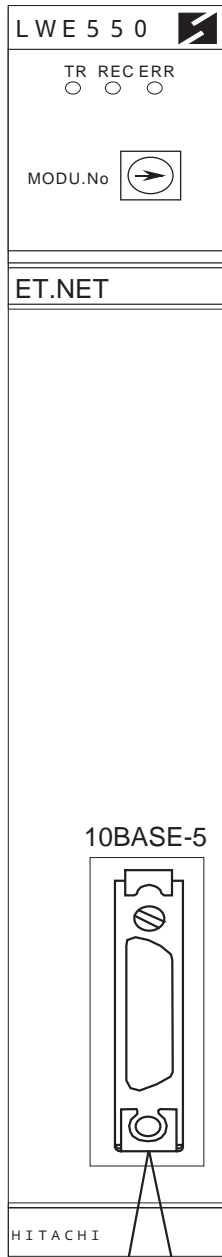
No.	名 称	機 能																	
	TR LED表示	データを送信しているときに点灯します。																	
	REC LED表示	伝送路上にデータが流れているとき（キャリア検出時）に点灯します。																	
	ERR LED表示	ハードウェア異常時点灯します。																	
	モジュールNo. 設定スイッチ	<p>下表メイン/サブモジュールの指定をします。この設定はCPUリセットまたは停復電後有効となります。</p> <table border="1"> <thead> <tr> <th colspan="2">モジュールNo.</th> <th rowspan="2">内 容</th> </tr> <tr> <th>メイン</th> <th>サブ</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>10BASE-5の通信</td> </tr> <tr> <td>2</td> <td>3</td> <td>設定しないでください</td> </tr> <tr> <td>4</td> <td>5</td> <td>パソコンとの通信</td> </tr> <tr> <td colspan="2">6 ~ F</td> <td>設定しないでください</td> </tr> </tbody> </table> <p>・モジュールNo. 2, 3, 6 ~ F は、設定禁止です。 ・モジュールNo. 4, 5 設定時IPアドレスは、下記設定値となります。 IPアドレス：192.192.192.001 （注）詳細設定方法は「ソフトウェアマニュアル オプション ET.NET For Windows（マニュアル番号 SAJ-3-148）」を参照してください。</p>	モジュールNo.		内 容	メイン	サブ	0	1	10BASE-5の通信	2	3	設定しないでください	4	5	パソコンとの通信	6 ~ F		設定しないでください
モジュールNo.		内 容																	
メイン	サブ																		
0	1	10BASE-5の通信																	
2	3	設定しないでください																	
4	5	パソコンとの通信																	
6 ~ F		設定しないでください																	
	10BASE-5 I/Fコネクタ	S10/2 および他のコントローラとの通信用コネクタです。																	

❗ 強 制

モジュールNo.設定スイッチを変更するときは、必ず電源OFFの状態、またはCPUリセット状態で行ってください。それ以外の状態での設定変更は、誤動作の原因となります。

ET.NETモジュールを2枚使用する場合、どちらか一方は電源内蔵型トランシーバを接続してください。

3.2 配線



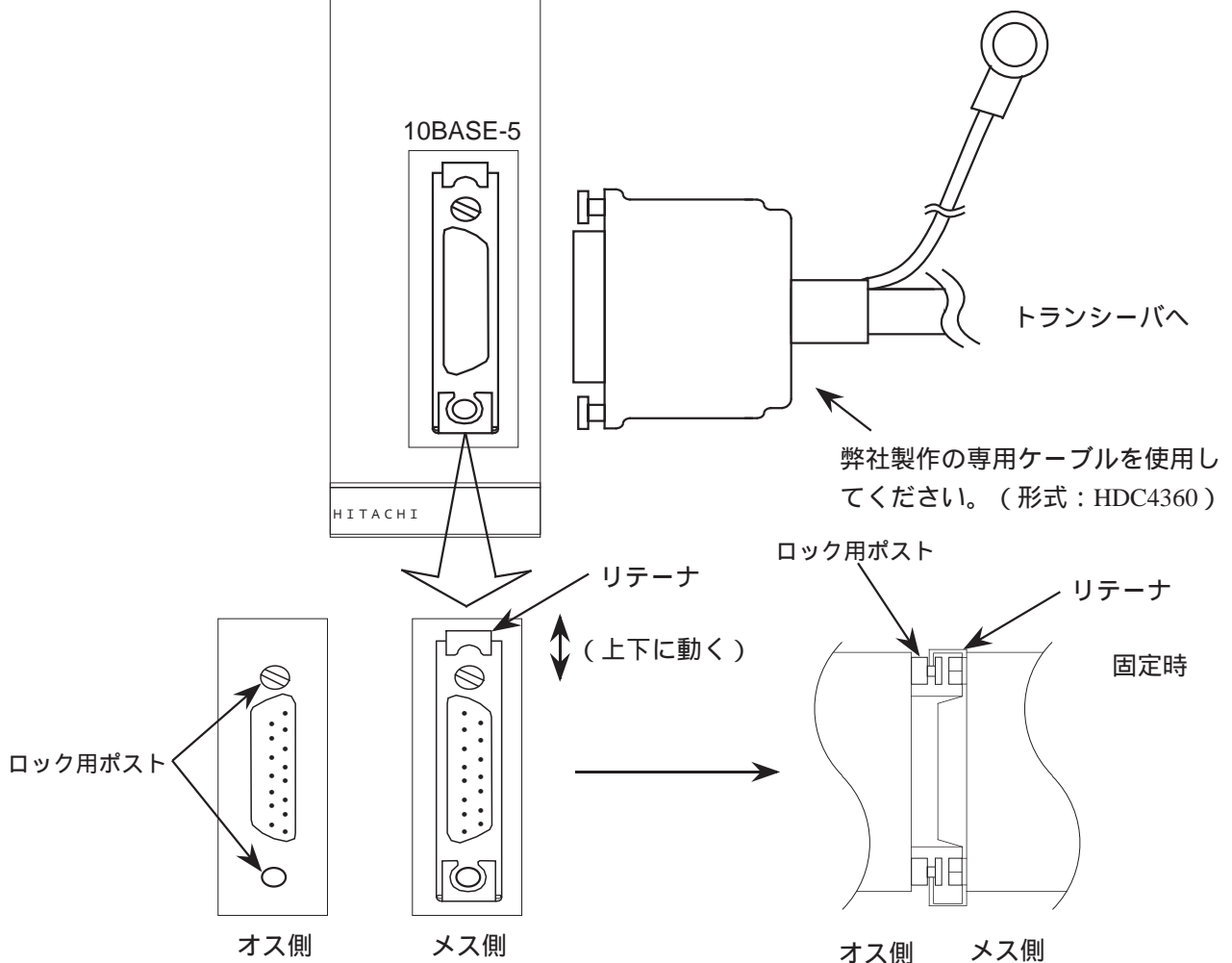
! 強制

10BASE-5のコネクタ（リテーナなど）に触れる際には、必ず人体に帯電している静電気を放電してから作業を行ってください。人体に帯電している静電気により感電、誤動作、故障の原因となります。

トランシーバケーブルの接続は、モジュール側のロック用リテーナをスライドさせ、ケーブルのロック用ポストに完全にロックするように取付けてください。

トランシーバケーブルは、下図のように必ずシールドアース配線のできるケーブルを使用してください。アース用配線のないケーブルでは、ノイズなどの影響によりエラーが発生する場合があります。

CPUモジュールのSHD端子に配線



4 利用の手引き

4 利用の手引き

4.1 10BASE-5のシステム構成

基本構成は、図4-1のように最長500mの同軸ケーブルとそれに接続されるステーションからなります。ステーションは、トランシーバケーブルとトランシーバを介して同軸ケーブルに接続されます。

この基本構成をセグメントといい、1セグメントのステーション数は、最大100台です。

ステーション間距離が500m以上となる場合は、図4-2に示すようにリピータを使用して分岐状にセグメントの数を増やすことになります。

図4-2は、最大ステーションの距離が1,500m以内のシステム例であり、どの2つのステーション間の経路を取っても通過するリピータの数が2台以下となるように構成してください。

図4-3は、ステーション間の最大距離を2,500mとした例であり、リピータにリンクケーブル（最長500m）を付けたものを1台のリピータとして数え、リンクセグメントと呼びます。

システム構成上のパラメータを以下に示します。

項目	仕様
セグメント最長	500m
セグメント内トランシーバ取付け最大数	100台
ステーション間最大距離	2,500m以下（トランシーバケーブルを除く）
システム最大ステーション数	1024台
トランシーバケーブル最長	15m
ステーション間経路内リピータ最大数	2台

注 意

リピータは、トランシーバケーブルとトランシーバを介して同軸ケーブルに接続してください。

リピータは、同軸セグメント中のどの位置のトランシーバにも取付けられます。

リンクケーブルには、ステーションを取付けないでください。

トランシーバの取付け間隔は、2.5mの整数倍としてください。

パソコンツールと接続しMCSなどの画面を開く場合は、4画面までしか開きません。

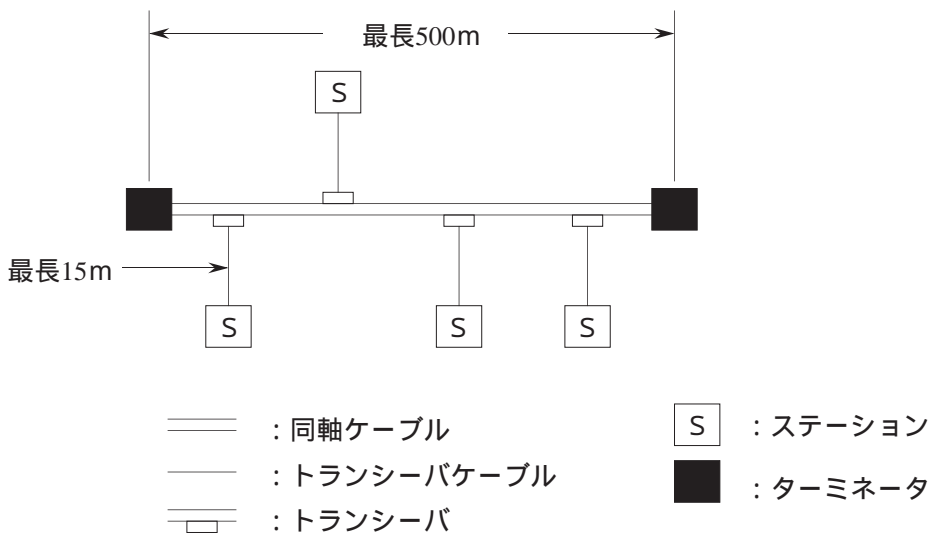


図4 - 1 最小構成 (リピータなし、セグメント長 最長500m)

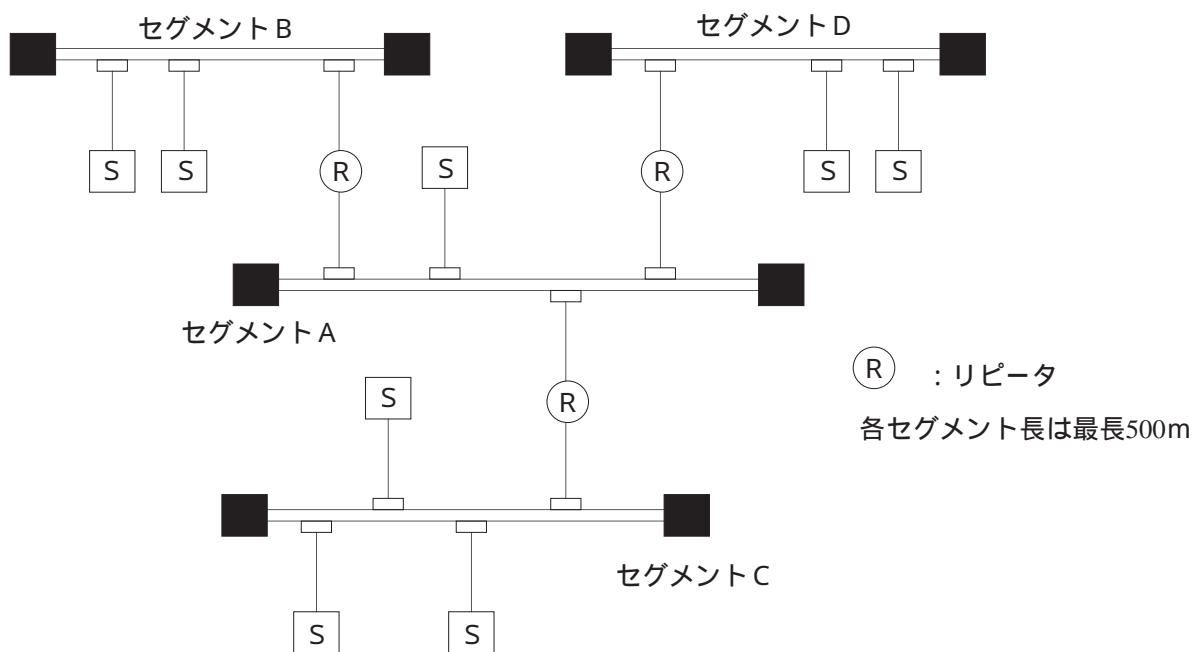


図4 - 2 中規模構成 (リピータ使用、トランシーバ間最長1,500m)

注 意

任意のステーション間のリピータは、2個以下にしてください。
 リピータが2個以上接続できるセグメントは、1つのみとしてください。

4 利用の手引き

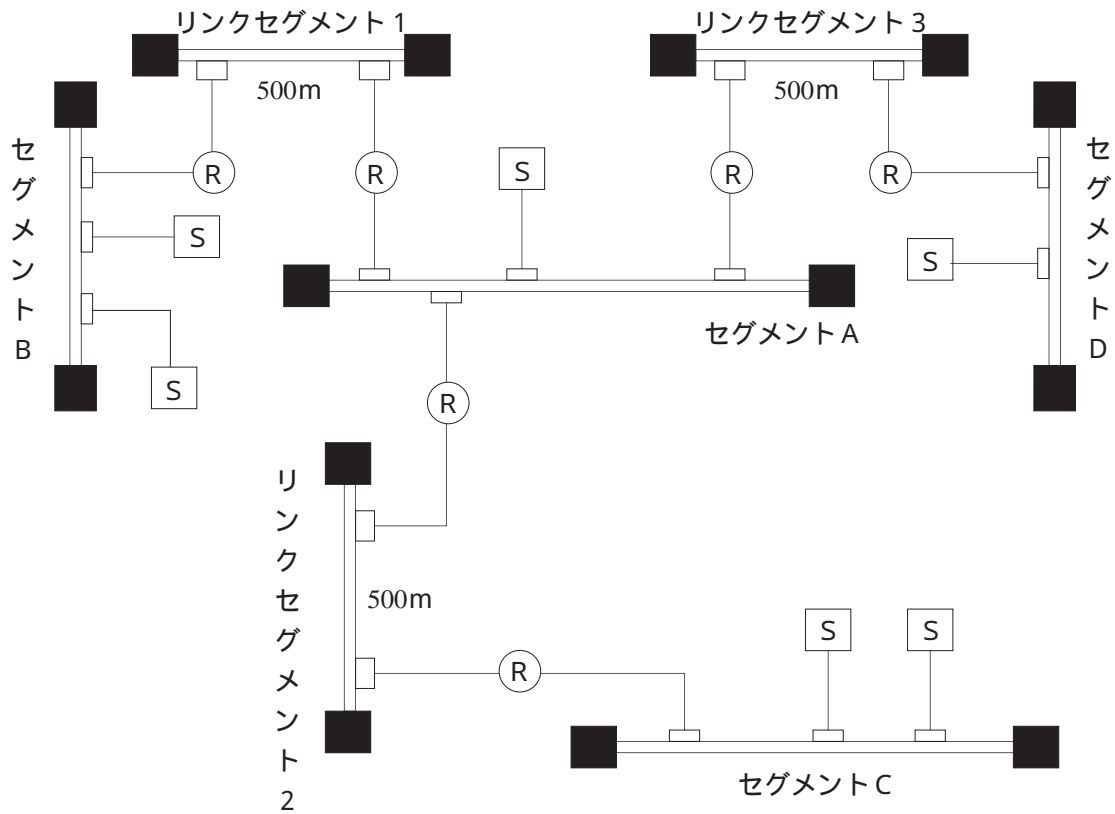


図4 - 3 大規模構成（リピータ、リンクセグメント使用、トランシーバ間最長2,500m）

注 意

リンクセグメントは、最長500mです。

リンクセグメントには、ステーションを取付けないでください。

任意のステーション間のリピータは、2個以下にしてください。

リピータが2個以上接続できるセグメントは、1つのみとしてください。

リンクセグメントは、両端のリピータを含めてリピータ1個とみなします。

注 意

マルチポートトランシーバの設置位置の制限

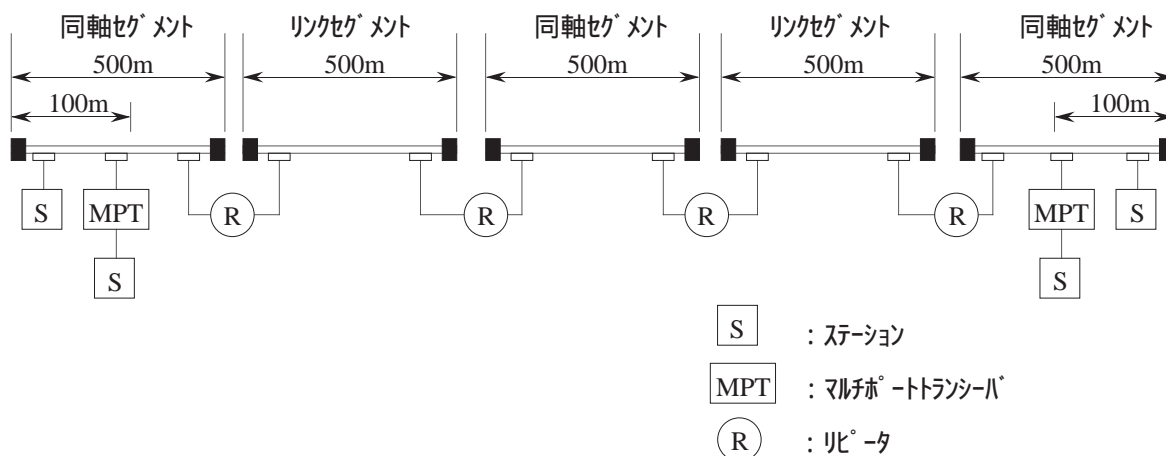
同軸ケーブル長として最長2,500m（5セグメント）で構成するシステムにおいて、最遠端の同軸ケーブルセグメント上にマルチポートトランシーバを設置する場合には、設置によりデータの遅延時間が増加するため、マルチポートトランシーバの設置位置に制限が生じます。

マルチポートトランシーバを経由したステーション間の最大距離は、マルチポートトランシーバ1台を通過することにより、同軸ケーブル長に換算して100m減少します。したがって、あるステーションから他のステーションに至る経路の同軸ケーブル長 L [m] には次のような制限があります。

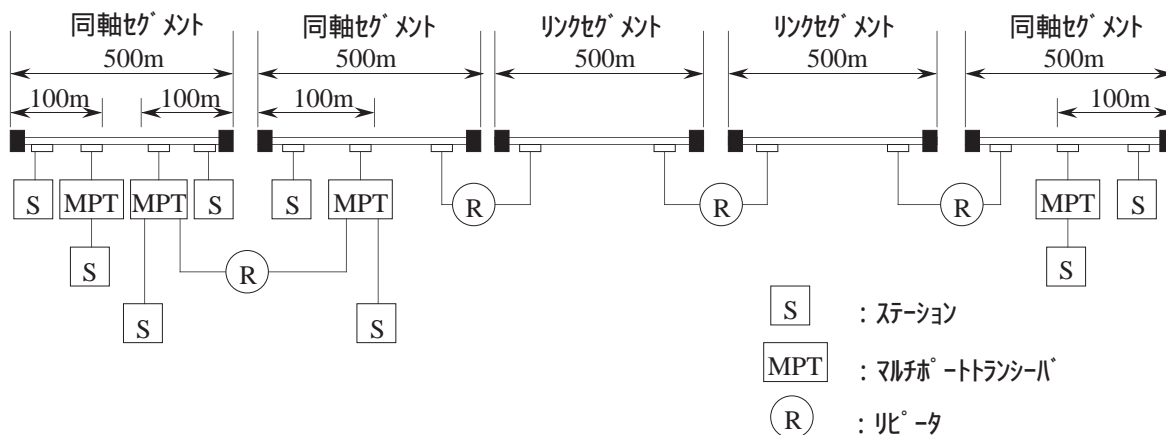
$$L \text{ [m]} \leq 2,500 \text{ [m]} - 100 \times N \text{ [m]}$$

N : 経由するマルチポートトランシーバの総数

2,500mの同軸ケーブルで構成されるシステムにおいては、マルチポートトランシーバは最遠端の同軸ケーブルターミネータから100m以上内側（ステーション間の距離を減少させる位置）に設定してください。

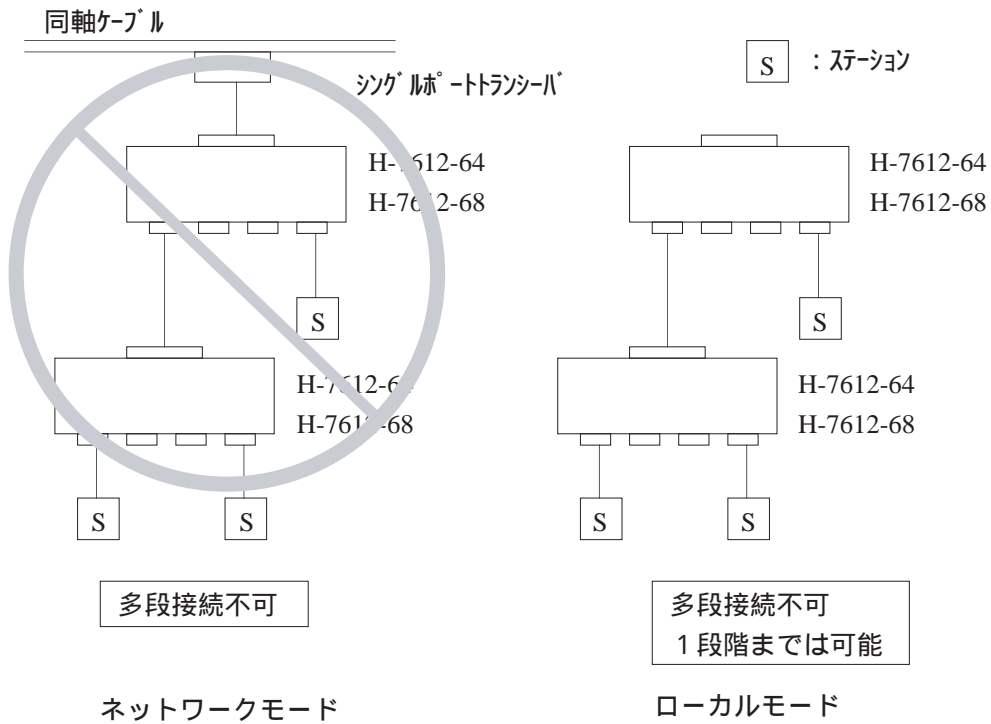


同様にマルチポートトランシーバを使用して、セグメント間のリピータを接続する場合もマルチポートトランシーバ1台を通過することにより、最遠端のステーション間の距離を100m減少させる位置にマルチポートトランシーバを設定する必要があります。



注 意

マルチポートトランシーバ（H-7612-64/68）は、ネットワークモードで使用する場合、伝送特性上の制約から多段接続はできません。



ネットワークモードにおいてはマルチポートトランシーバの上位に接続されるシングルポートトランシーバは、マルチポートトランシーバから給電されるDC12Vで動作する条件を保証するため、下記の指定機種を使用してください。

- ・ HLT - 200TB (メーカー: 日立電線 (株))
- ・ HLT - 200 (メーカー: 日立電線 (株))
- ・ HBN - 200TZ (メーカー: 日立電線 (株))
- ・ HLT - 200TD (メーカー: 日立電線 (株))

注 意

- ルータに関する制限 -

ルータ経由で通信する場合、下記制限があります。

経路情報として登録できるアドレスはIPアドレス、ネットワークアドレス合わせて15個までです。

登録できるアドレスはIPアドレスとネットワークアドレスで、サブネットアドレスは登録できません。

これは、ET.NETモジュールが経路情報をIPアドレスもしくはネットワークアドレスとして認識し、サブネットアドレスとして認識しないためです。仮にサブネットアドレスを登録したとしてもIPアドレスとして認識するため、通信できません。

下図のようなネットワーク構成時における経路情報の登録例を示します。

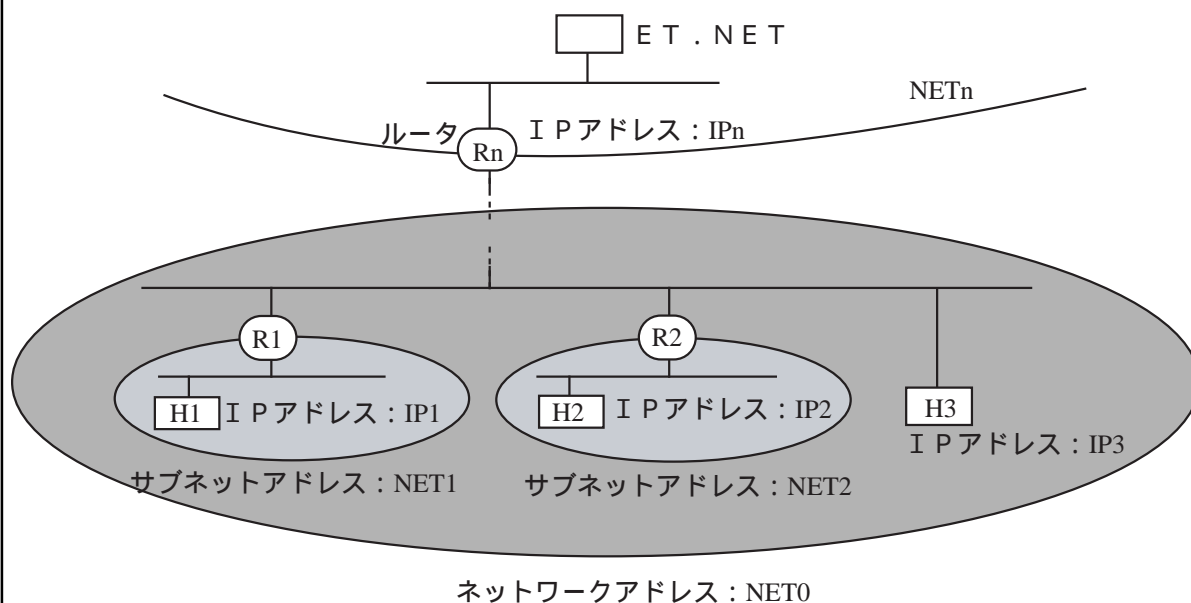
- 経路情報登録例 -

H1と通信する場合に登録する経路情報

- ・ルータRnのIPアドレス IPn
- ・ホストH1のIPアドレス IP1

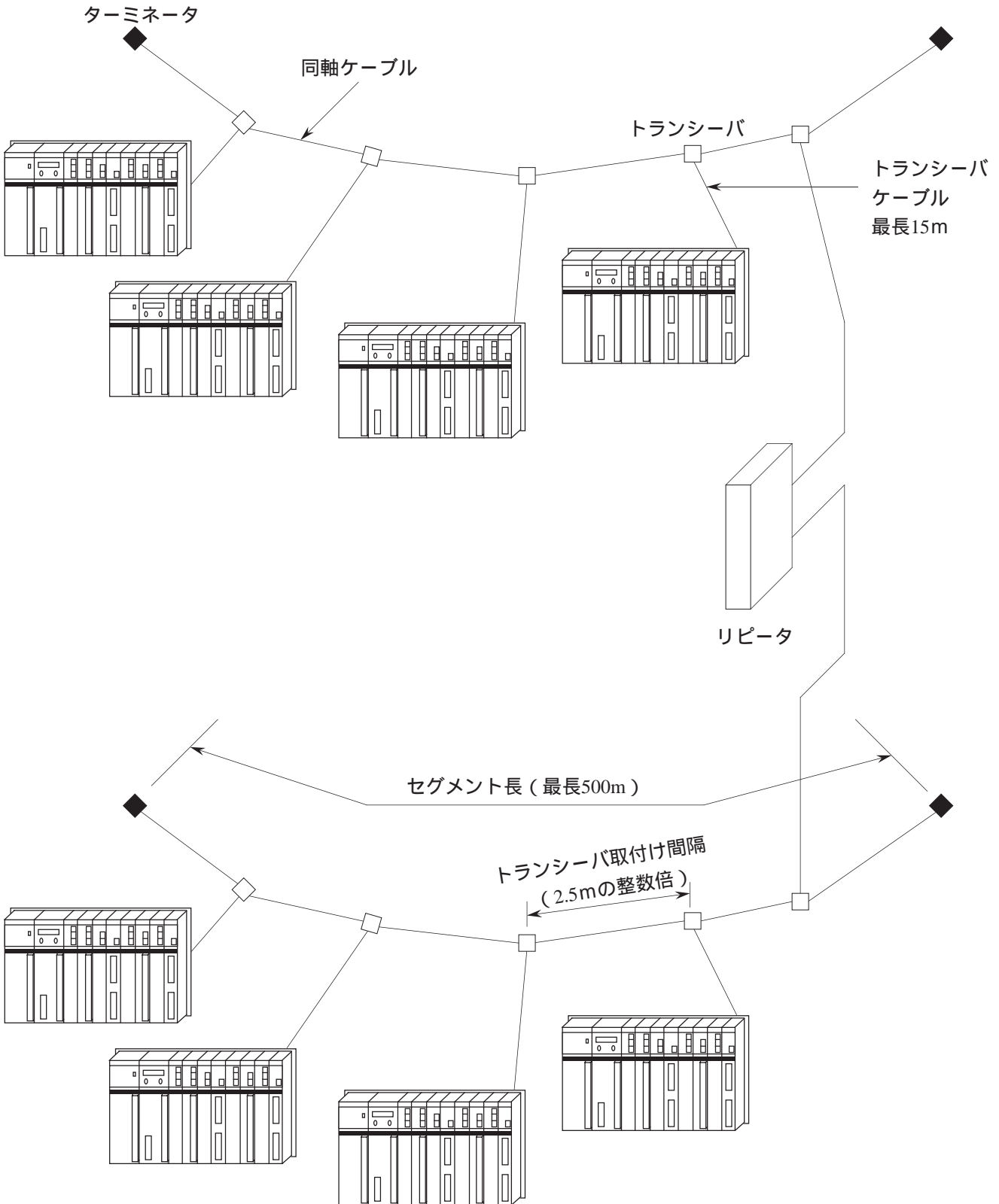
H3と通信する場合に登録する経路情報

- ・ルータRnのIPアドレス IPn
- ・ホストH3のIPアドレス IP3 またはネットワークアドレス NET0



4 利用の手引き

4.2 S10/2 によるシステム構成例



4.3 システム定義情報

各ステーションには必ず下記、の情報を定義してください。はネットワーク内で、ユニークな値でなければなりません。は同一サブネット内で同じ値としてください。

物理アドレス ————— ET.NETのROM 1台ごとにユニークなナンバが設定されています。
 IPアドレス }
 サブネットマスク } — ET.NET 1台ごとに、を定義してください。

4.3.1 物理アドレス

1台のET.NETには、48ビットの物理的なアドレスを割付けてあります。

このアドレスは全世界に1つのユニークなアドレスであり、ROM化されていますので、ユーザが変更することはできません。例えば、物理的アドレスは16進で以下のように記述します。

(例)

00008700B001

4.3.2 IPアドレス

TCP/IPとUDP/IPはIPアドレスという32ビットの論理アドレスを使用します。

IPアドレスはネットワーク番号とホスト番号からなり、そのアドレスの割付けはホストの台数によって、次の3通りが使用できます。

(i) クラスA (ネットワーク番号の上位1ビットを0とします。)

ネットワーク番号 (8ビット)	ホスト番号(24ビット)
--------------------	--------------

(ii) クラスB (ネットワーク番号の上位2ビットを10とします。)

ネットワーク番号 (16ビット)	ホスト番号(16ビット)
---------------------	--------------

(iii) クラスC (ネットワーク番号の上位3ビットを110とします。)

ネットワーク番号 (24ビット)	ホスト番号(8ビット)
---------------------	-------------

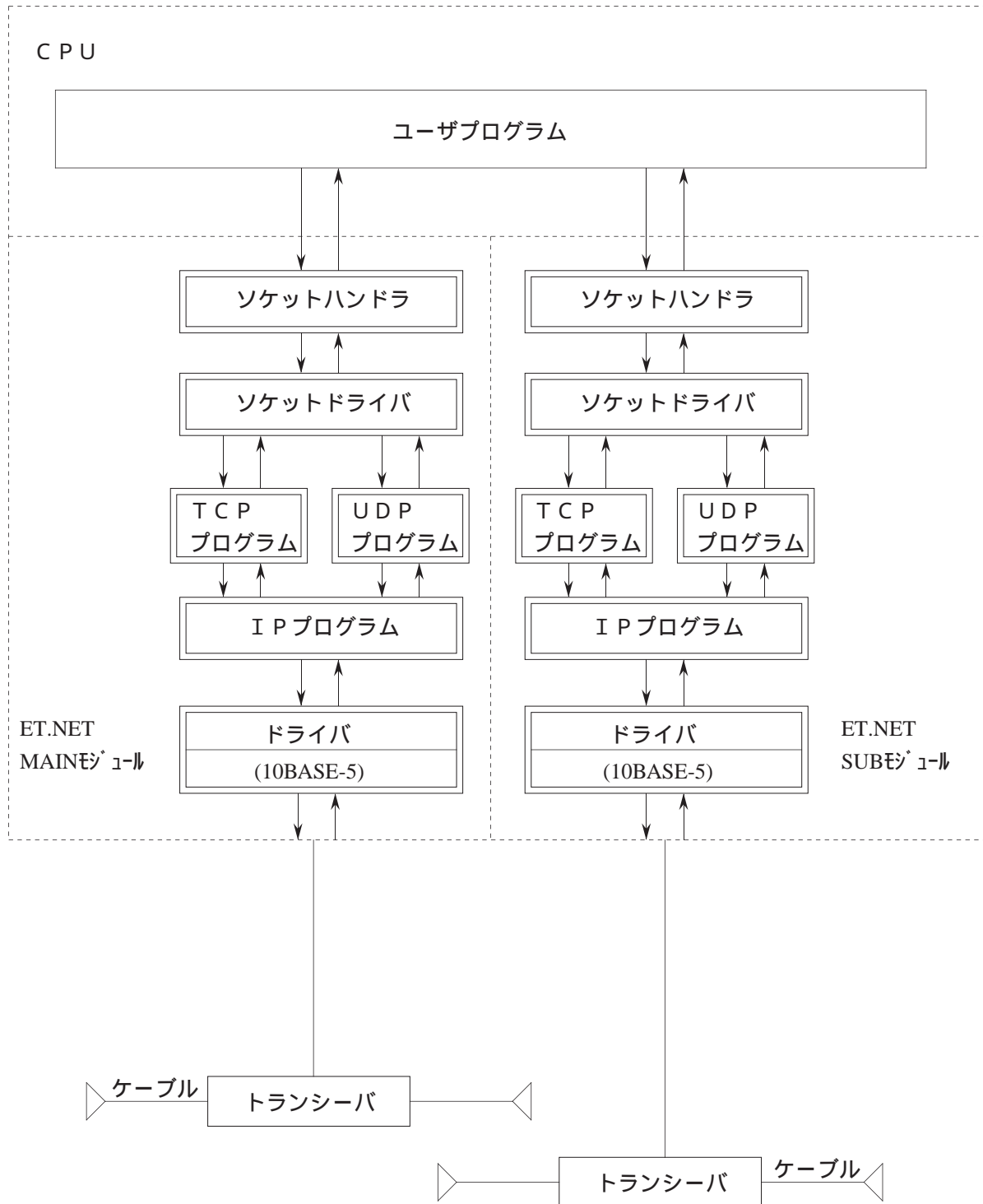
4.3.3 サブネットマスク

IPアドレスをサブネットに分割する場合、サブネットワーク番号とローカルホスト番号の境界をサブネットマスクによって定義します。サブネットマスクをデフォルト値以外で使用する場合、下記例のようなブロードキャストアドレスになることを前提にサブネットマスクを使用してください。

(例) クラスBの場合

IPアドレス	サブネットマスク	ブロードキャストアドレス
128.123.000.001	255.255.000.000	128.123.255.255
128.123.001.001	255.255.255.000	128.123.001.255

4.4 ET.NETのソフトウェア構成



4.5 ET.NETのシステムプログラム

「4.4 ET.NETのソフトウェア構成」で示したシステムプログラムの説明をします。

システムプログラムは、次の6種類に大別でき、CPUまたはET.NETモジュール上で動作します。

- ・ソケットハンドラ
- ・ソケットドライバ
- ・TCPプログラム
- ・UDPプログラム
- ・IPプログラム
- ・ドライバ

4.5.1 ソケットハンドラ

ソケットハンドラはC言語の関数として呼出され、ユーザプログラムの代わりにET.NETモジュールを制御します。ユーザはソケットハンドラを利用することにより、ハードウェア仕様ならびに通信プロトコルを意識することなくプログラミングできます。

4.5.2 ソケットドライバ

ソケットドライバはソケットハンドラからのコマンドをTCPプログラムまたは、UDPプログラムにメモリインタフェースで受渡し、処理を行います。

4.5.3 TCPプログラム

上位のプロトコルとして、高信頼性のデータ送受信管理を行います。

TCPプログラムの機能を以下に示します。

- ・信頼性チェック
 - ・受信応答信号 (ACK) の確認
 - ・シーケンス番号による順序チェック
 - ・データのチェックサム
- ・データ再送 (信頼性チェックにてエラー発生時)
- ・受信可能データ量のフロー制御
- ・複数プロセスの同時通信 (多重化)
- ・接続の確立による論理接続
- ・データのセキュリティと優先順位管理

4 利用の手引き

4.5.4 UDPプログラム

上位のプロトコルとして、高速かつ大量のデータ送受信管理を行います。

UDPプログラムの機能を以下に示します。

- ・コネクションレス型の通信
- ・同時通信
- ・パケットに基づいたデータ伝送

4.5.5 IPプログラム

下位のプロトコルとして、通信路の論理的な接続を行います。

IPプログラムの機能を以下に示します。

- ・パケットの最大長に応じたデータの分割と再組立て
- ・IPアドレスと物理アドレスの交換

4.5.6 ドライバ

通信回路を制御し、回線（トランシーバ）へのデータ送受信を行います。

ドライバの機能を以下に示します。

- ・送受信データのCRC（Cyclic Redundancy Check：巡回冗長検査）
- ・送受信時のデータ衝突検出と再送

4.6 ユーザの作成するプログラム

「4.5 ET.NETのシステムプログラム」では、システム提供のプログラムを説明しましたが、この節では、ユーザが作成する必要があるソフトウェアについて説明します。

4.6.1 ユーザプログラム

ユーザプログラムは、ソケットハンドラを起動し、データの送信、受信を実施します。

ユーザプログラムは、Cモードプログラムで作成し、H - S10 / 2 シリーズにローディングします。

Cモードプログラム	コンピュータ言語（C言語，アセンブラなど）で作成され、タスク、Pコイルの形で実行することができます。
	OSは、CPMS（Compact Process Monitor System）を使用してください。また、Cモードプログラムには拡張メモリが必要です。

ソケットハンドラについては、「4.7 ソケットハンドラ」で説明します。

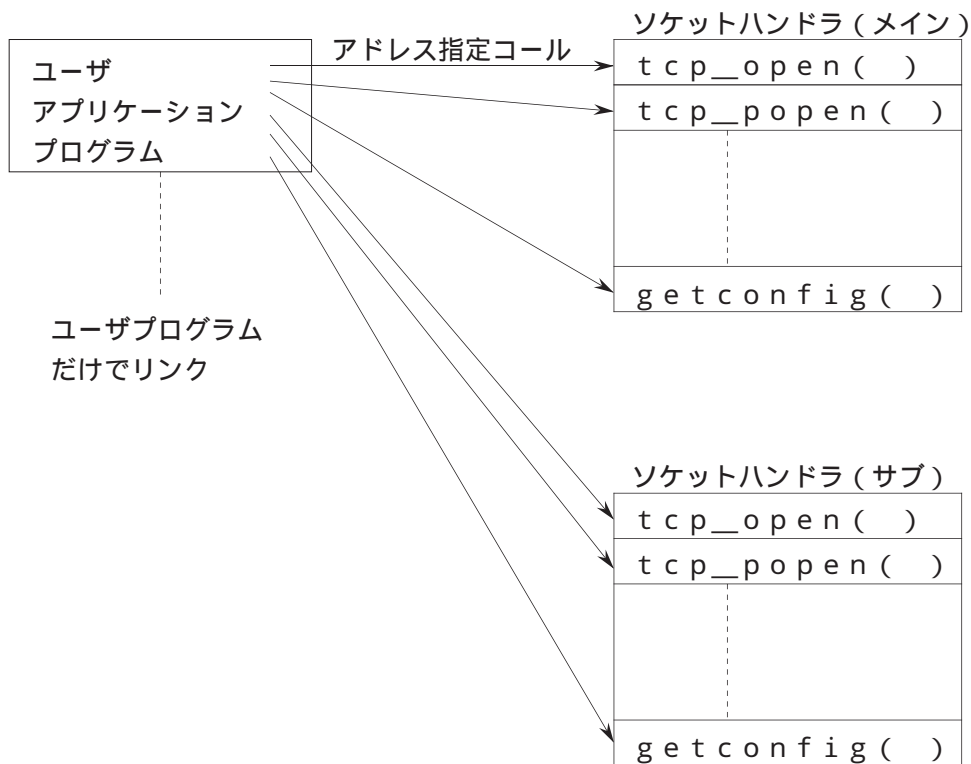
ソケットハンドラによるプログラムについては、「第5章 プログラム例」を参照してください。

4 利用の手引き

4.7 ソケットハンドラ

ソケットハンドラはC言語の関数として呼出され、ユーザプログラムの代わりにET.NETモジュールを制御し、データの送受信を実施します。ソケットハンドラは20種類の関数群で構成されます。

ソケットハンドラは、アドレス指定で呼出してください。ユーザプログラムは、ソケットハンドラを含めた形では作成（リンク）できません。



4.7.1 ソケットハンドラー一覧

以下にソケットハンドラー一覧を示します。

名 称	サブ-リコール アドレス		機 能	対応プログラム
	メイン	サブ		
tcp_open()	/874100	/8F4100	T C P 能動的オープン	T C P / I P
tcp_popen()	/874106	/8F4106	T C P 受動的オープン	T C P / I P
tcp_accept()	/87410C	/8F410C	T C P コネクション要求受け	T C P / I P
tcp_close()	/874112	/8F4112	T C P コネクション終了	T C P / I P
tcp_abort()	/87411E	/8F411E	T C P コネクション強制終了	T C P / I P
tcp_getaddr()	/874124	/8F4124	T C P ソケット情報読出し	T C P / I P
tcp_stat()	/87412A	/8F412A	T C P コネクション状態読出し	T C P / I P
tcp_send()	/874130	/8F4130	T C P データ送信	T C P / I P
tcp_receive()	/874136	/8F4136	T C P データ受信	T C P / I P
udp_open()	/874160	/8F4160	U D P オープン	U D P / I P
udp_close()	/874166	/8F4166	U D P クローズ	U D P / I P
udp_send()	/87416C	/8F416C	U D P データ送信	U D P / I P
udp_receive()	/874172	/8F4172	U D P データ受信	U D P / I P
route_list()	/874178	/8F4178	経路情報読出し	T C P / I P および U D P / I P
route_del()	/87417E	/8F417E	経路情報削除	T C P / I P および U D P / I P
route_add()	/874184	/8F4184	経路情報登録	T C P / I P および U D P / I P
arp_list()	/87418A	/8F418A	A R P 情報読出し	T C P / I P および U D P / I P
arp_del()	/874190	/8F4190	A R P 情報削除	T C P / I P および U D P / I P
arp_add()	/874196	/8F4196	A R P 情報登録	T C P / I P および U D P / I P
getconfig()	/87419C	/8F419C	コンフィグレーション情報読出し	T C P / I P および U D P / I P

注 意

E T . N E Tモジュールはフラッシュメモリを採用していて、フラッシュメモリに記憶しているプログラムデータの信頼性向上のために約1ヶ月周期でフラッシュメモリのデータを更新しています。しかし、更新中は約3秒間ソケットハンドラの応答が待たされます。`tcp_receive()`、`udp_receive()`の受信待ち時間を3秒より小さく設定した場合には、タイムアウトエラーが発生する可能性があるため、この場合はリトライしてください。

1つのモジュールで、同時に使用可能なソケット数は、TCPが10個でUDPが8個までです。

0~9999のポート番号はシステムで占有していますので、ユーザは10000~65535を使用してください。

データ送受信のデータ長は、1回の関数発行でTCPが1~4096バイトでUDPが1~1472バイトです。

IPアドレス、サブネットマスクはCPU内のOSテーブルに設定されます。CPUの交換OSの再ローディングを実施した場合、再設定が必要です。

- タスクの強制終了 -

ソケットハンドラを利用しているタスクが強制終了されると、ソケットが登録状態のまま残ってしまいます（そのタスクが自分で使用しているソケットを`tcp_close()`または`udp_close()`した後ならばこの限りではありません）。

つまり、タスクが強制終了されたときのソケットの状態が、タスクが終了したにもかかわらず残ってしまうことです。以下、そういう状態のソケットを「浮いたソケット」と呼ぶことにします。

浮いたソケットは、他のタスクで使用できません。したがって、浮いたソケットまたはモジュールに対して、下記1~3のいずれかの処理を行ってください。

1. 他のタスクまたは組込みサブルーチンから浮いたソケットを`tcp_close()`または`udp_close()`する。
2. CPUをリセットする。
3. 電源を一度遮断し復電する。

`tcp_open()`

[機能] この関数は、TCP/IPプログラムのソケットの登録、ポートの確保、相手局に対してのコネクションの要求を発行する関数です。リターン値には登録されたソケットIDまたは、エラーコードを返します。この関数はSYNを送信し、コネクションの確立（相手局からのSYN受信）を待ちます。相手局からの応答がない場合、75秒後にポート解放エラー（エラーコード：`0xF0FF`）でリターンしますので、`tcp_open()`を再発行してください。

[リンク手順]

C言語	
メイン	サブ
<pre> struct open_p { long dst_ip; short dst_port; short src_port; char notuse; char ttl; }; } short (*tcp_open)(); short rtn; struct open_p *padr; } tcp_open=(short(*)())0x874100; } rtn=(*tcp_open)(padr); } </pre>	<pre> struct open_p { long dst_ip; short dst_port; short src_port; char notuse; char ttl; }; } short (*tcp_open)(); short rtn; struct open_p *padr; } tcp_open=(short(*)())0x8F4100; } rtn=(*tcp_open)(padr); } </pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス

`p a d r - > d s t _ i p` : 相手局のIPアドレス

`p a d r - > d s t _ p o r t` : 相手局のポート番号

`p a d r - > s r c _ p o r t` : 自局のポート番号

`p a d r - > n o t u s e` : 0固定（未使用）

`p a d r - > t t l` : Time to live

`t t l`を0とした場合、デフォルト値30となります。

<出力パラメータ詳細>

リターン値 : 登録されたソケットIDまたは、エラーコードが返ります。

(`0~0x000F`) 登録されたソケットID

(`0xF000~0xFFFF`) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

tcp_popen ()

[機能] この関数は、TCP/IPプログラムのソケットの登録、そのソケットを受動状態にさせる関数です。リターン値には登録されたソケットIDまたは、エラーコードを返します。この関数は、UNIXにおけるsocket+bind+listenに相当します。dst_ip、dst_portを0に設定すると任意の相手局からの接続要求を受け付けることができます。また、src_portを0に設定すると1024～2047までの任意のポートが確保されます。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct popen_p { long dst_ip; short dst_port; short src_port; char listennum; char ttl; }; { short (*tcp_popen)(); short rtn; struct popen_p *padr; { tcp_popen = (short (*)())0x874106; } rtn = (*tcp_popen)(padr); } </pre>	<pre> struct popen_p { long dst_ip; short dst_port; short src_port; char listennum; char ttl; }; { short (*tcp_popen)(); short rtn; struct popen_p *padr; { tcp_popen = (short (*)())0x8F4106; } rtn = (*tcp_popen)(padr); } </pre>

[パラメータ]

< 入力パラメータ詳細 >

padr : 入力パラメータの先頭アドレス

padr->dst_ip : 相手局のIPアドレス

padr->dst_port : 相手局のポート番号

padr->src_port : 自局のポート番号

padr->listennum : ACCEPTされていない接続の最大数
(将来用のため0固定)

padr->ttl : Time to live

相手局未指定の場合は、dst_ip, dst_portを0とします。
ttlを0とした場合、デフォルト値30となります。

< 出力パラメータ詳細 >

リターン値 : 登録されたソケットIDまたは、エラーコードが返ります。
(0～0x000F) 登録されたソケットID
(0xF000～0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

<code>tcp_accept()</code>

[機能] この関数は、TCP/IPプログラムで`tcp_open()`関数により、受動状態になったソケットIDに対するコネクションの要求(SYNの受信)を待ち、コネクションの確立を受け付ける関数です。リターン値にはコネクション確立後の登録されたソケットIDまたは、エラーコードを返します。入力パラメータのソケットIDとコネクション確立後の登録されたソケットIDは同じ値となります。この関数は相手局と接続されるまで待ち続けます。

[リンク手順]

C言語	
メイン	サブ
<pre> struct accept_p { short s_id; }; { short (*tcp_accept)(); short rtn; struct accept_p *padr; { tcp_accept=(short(*)())0x84710C; { rtn=(*tcp_accept)(padr); { </pre>	<pre> struct accept_p { short s_id; }; { short (*tcp_accept)(); short rtn; struct accept_p *padr; { tcp_accept=(short(*)())0x8F410C; { rtn=(*tcp_accept)(padr); { </pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > s _ i d` : ソケットID

<出力パラメータ詳細>

リターン値 : 登録されたソケットIDまたは、エラーコードが返ります。
 (0~0x000F) 登録されたソケットID
 (0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

`tcp_close()`

[機能] この関数は、ソケットIDに対応したコネクションを終了させソケットを削除する関数です。リターン値に処理結果を返します。この関数は、FINを送信し、コネクションの終了（相手局からのFIN受信）を待ちます。相手局からの応答がない場合、30秒後にソケットドライバタイムアウトエラー（エラーコード：0xF012）でリターンしますので、`tcp_abort()`を発行してください。

[リンク手順]

C 言語	
メイン	サブ
<pre>struct close_p { short s_id; }; { short (*tcp_close)(); short rtn; struct close_p *padr; { tcp_close = (short (*) ())0x874112; } rtn = (*tcp_close)(padr); }</pre>	<pre>struct close_p { short s_id; }; { short (*tcp_close)(); short rtn; struct close_p *padr; { tcp_close = (short (*) ())0x8F4112; } rtn = (*tcp_close)(padr); }</pre>

[パラメータ]

< 入力パラメータ詳細 >

`padr` : 入力パラメータの先頭アドレス
`padr -> s_id` : ソケットID

< 出力パラメータ詳細 >

リターン値 : 処理結果が返ります。
(0) 正常終了
(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

```
tcp_abort( )
```

[機能] この関数は、ソケットIDに対応したコネクションを強制終了(RSTを送信)させソケットを削除する関数です。リターン値に処理結果を返します。

[リンク手順]

C言語	
メイン	サブ
<pre>struct sid_p { short s_id; }; { short (*tcp_abort)(); short rtn; struct sid_p *padr; { tcp_abort = (short(*)())0x87411E; { rtn = (*tcp_abort)(padr); {</pre>	<pre>struct sid_p { short s_id; }; { short (*tcp_abort)(); short rtn; struct sid_p *padr; { tcp_abort = (short(*)())0x8F411E; { rtn = (*tcp_abort)(padr); {</pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > s _ i d` : ソケットID

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

```
tcp_getaddr( )
```

[機能] この関数は、ソケットIDに対応したコネクション相手局のIPアドレス、自局ポート番号、相手局ポート番号を取得する関数です。リターン値に処理結果を返します。処理結果が正常終了の場合、outinfの取得情報が有効となります。

[リンク手順]

C 言語	
メイン	サブ
<pre>struct sid_p { short s_id; }; struct getaddr_p { long ipaddr; short src_port; short dst_port; }; { short (*tcp_getaddr)(); short rtn; struct sid_p *padr; struct getaddr_p *outinf; { tcp_getaddr = (short(*) ())0x874124; { rtn = (*tcp_getaddr)(padr, outinf); {</pre>	<pre>struct sid_p { short s_id; }; struct getaddr_p { long ipaddr; short src_port; short dst_port; }; { short (*tcp_getaddr)(); short rtn; struct sid_p *padr; struct getaddr_p *outinf; { tcp_getaddr = (short(*) ())0x8F4124; { rtn = (*tcp_getaddr)(padr, outinf); {</pre>

[パラメータ]

< 入力パラメータ詳細 >

padr : 入力パラメータの先頭アドレス
padr -> s_id : ソケットID

< 出力パラメータ詳細 >

outinf : 出力パラメータの先頭アドレス
outinf -> ipaddr : 相手局のIPアドレス
outinf -> src_port : 自局のポート番号
outinf -> dst_port : 相手局のポート番号

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

```
tcp_stat( )
```

[機能] この関数は、ソケットIDに対応したコネクションのステータスを取得する関数です。リターン値に処理結果を返します。処理結果が正常終了の場合、`outinf`の取得情報が有効となります。

[リンク手順]

C言語	
メイン	サブ
<pre>struct sid_p { short s_id; }; struct stat_p { unsigned short stat; unsigned short urg; unsigned short sendwin; unsigned short recvwin; }; { short (*tcp_stat)(); short rtn; struct sid_p *pdr; struct stat_p *outinf; { tcp_stat=(short(*)())0x87412A; { rtn=(*tcp_stat)(pdr, outinf); {</pre>	<pre>struct sid_p { short s_id; }; struct stat_p { unsigned short stat; unsigned short urg; unsigned short sendwin; unsigned short recvwin; }; { short (*tcp_stat)(); short rtn; struct sid_p *pdr; struct stat_p *outinf; { tcp_stat=(short(*)())0x8F412A; { rtn=(*tcp_stat)(pdr, outinf); {</pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス

`p a d r - > s _ i d` : ソケットID

4 利用の手引き

<出力パラメータ詳細>

`outinf` : 出力パラメータの先頭アドレス

```
outinf->stat      : コネクション状態
                   0 : CLOSED
                   1 : LISTEN
                   2 : SYN_SENT
                   3 : SYN_RECEIVED
                   4 : ESTABLISHED
                   5 : CLOSE_WAIT
                   6 : FIN_WAIT_1
                   7 : CLOSING
                   8 : LAST_ACK
                   9 : FIN_WAIT_2
                  10 : TIME_WAIT
outinf->urg       : urgent data あり/なし
                   0 : urgent data なし
                   0以外 : urgent data 数
outinf->sendwin   : 送信ウィンドウの残量
outinf->recvwin   : 到着済み受信データ量
```

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

<code>tcp_send()</code>

[機能] この関数は、ソケットIDに対応したコネクションにパラメータのbufからlen分のデータを送信する関数です。リターン値に処理結果を返します。処理結果に0xF012が返ってきた場合、`tcp_stat()`でコネクション状態および送信ウィンドウ残量により送信リトライ中を確認してください。この関数は、送信ウィンドウにデータが格納された時点でリターンします。データの送信状態は、`tcp_stat`の送信ウィンドウ残量により確認してください。

[リンク手順]

C言語	
メイン	サブ
<pre> struct send_p { short s_id; short len; char *buf; }; { short (*tcp_send)(); short rtn; struct send_p *padr; { tcp_send = (short(*) ())0x874130; } rtn = (*tcp_send)(padr); } </pre>	<pre> struct send_p { short s_id; short len; char *buf; }; { short (*tcp_send)(); short rtn; struct send_p *padr; { tcp_send = (short(*) ())0x8F4130; } rtn = (*tcp_send)(padr); } </pre>

[パラメータ]

<入力パラメータ詳細>

`padr` : 入力パラメータの先頭アドレス
`padr -> s_id` : ソケットID
`padr -> len` : 送信するデータ長 (バイト数: 1 ~ 4096)
`padr -> buf` : 送信するデータの先頭アドレス

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。
(0) 正常終了
(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

`tcp_receive()`

[機能] この関数は、ソケットIDに対応したコネクションからパラメータのlen分のデータをbufに受信する関数です。リターン値に処理結果を返します。この関数はパラメータのtimに受信待ち時間の指定が可能ですが、受信待ち時間以内であってもデータを受信した時点でリターンします。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct receive_p { short s_id; short len; char *buf; long tim; }; } short (*tcp_receive)(); short rtn; struct receive_p *pdr; } tcp_receive =(short(*) ()) 0x874136; } rtn = (*tcp_receive)(pdr); } </pre>	<pre> struct receive_p { short s_id; short len; char *buf; long tim; }; } short (*tcp_receive)(); short rtn; struct receive_p *pdr; } tcp_receive =(short(*) ()) 0x8F4136; } rtn = (*tcp_receive)(pdr); } </pre>

[パラメータ]

<入力パラメータ詳細>

p a d r : 入力パラメータの先頭アドレス
 p a d r - > s _ i d : ソケットID
 p a d r - > l e n : 受信バッファ長 (バイト数: 1 ~ 4096)
 p a d r - > b u f : 受信バッファの先頭アドレス
 p a d r - > t i m : 受信待ち時間 (m s : 0 ~ 86400000 (24時間))

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。
 (0) 正常終了 (受信データなし)
 (0x0001 ~ 0x1000) 正常終了 (受信したバイト数)
 (0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

udp_open ()

[機能] この関数は、UDP / IPプログラムのソケットの登録、ポートの確保を行う関数です。リターン値には登録されたソケットIDまたは、エラーコードを返します。

パラメータのdst_ipに0を指定すると任意のホストからパケットを受信できます。

パラメータのdst_portに0を指定すると任意のポートからデータを受信できます。

パラメータのsrc_portに0を指定すると1024~2048までの使用していないポートが確保されます。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct uopen_p { long dst_ip; short dst_port; short src_port; char pktmode; char ttl; }; { short (*udp_open)(); short rtn; struct uopen_p *padr; { udp_open =(short(*) ()) 0x874160; } rtn = (*udp_open) (padr); { </pre>	<pre> struct uopen_p { long dst_ip; short dst_port; short src_port; char pktmode; char ttl; }; { short (*udp_open)(); short rtn; struct uopen_p *padr; { udp_open =(short(*) ()) 0x8F4160; } rtn = (*udp_open) (padr); { </pre>

[パラメータ]

< 入力パラメータ詳細 >

p a d r : 入力パラメータの先頭アドレス

p a d r - > d s t _ i p : 相手局のIPアドレス

p a d r - > d s t _ p o r t : 相手局のポート番号

p a d r - > s r c _ p o r t : 自局のポート番号

p a d r - > p k t m o d e : パケットモード (0 固定)

p a d r - > t t l : Time to live

t t l を 0 とした場合、デフォルト値30となります。

< 出力パラメータ詳細 >

リターン値 : 登録されたソケットIDまたは、エラーコードが返ります。

(0x0020 ~ 0x0027) 登録されたソケットID

(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

`udp_close ()`

[機能] この関数は、ソケットIDに対応したソケットを削除する関数です。リターン値に処理結果を返します。

[リンク手順]

C 言語	
メイン	サブ
<pre>struct uclose_p { short s_id; }; { short (*udp_close)(); short rtn; struct uclose_p *padr; { udp_close =(short(*) ()) 0x874166; } rtn = (*udp_close)(padr); }</pre>	<pre>struct uclose_p { short s_id; }; { short (*udp_close)(); short rtn; struct uclose_p *padr; { udp_close =(short(*) ()) 0x8F4166; } rtn = (*udp_close)(padr); }</pre>

[パラメータ]

< 入力パラメータ詳細 >

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > s _ i d` : ソケットID

< 出力パラメータ詳細 >

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

udp_send ()

[機能] この関数は、ソケットIDに対応したソケットにパラメータのbufからlen分のデータを送信する関数です。リターン値に処理結果を返します。dst_ip, dst_portの指定については、udp_open ()で指定されたものが優先されます。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct usend_p { short s_id; short notuse; long dst_ip; short dst_port; short len; char *buf; }; } short (*udp_send)(); short rtn; struct usend_p *padr; } udp_send=(short(*) ()) 0x87416C; } rtn = (*udp_send)(padr); } </pre>	<pre> struct usend_p { short s_id; short notuse; long dst_ip; short dst_port; short len; char *buf; }; } short (*udp_send)(); short rtn; struct usend_p *padr; } udp_send=(short(*) ()) 0x8F416C; } rtn = (*udp_send)(padr); } </pre>

[パラメータ]

< 入力パラメータ詳細 >

padr : 入力パラメータの先頭アドレス

padr->s_id : ソケットID

padr->notuse : 0固定(未使用)

padr->dst_ip : 相手局のIPアドレス

padr->dst_port : 相手局のポート番号

padr->len : 送信するデータ長(バイト数; 1~1472)

padr->buf : 送信するデータの先頭アドレス


udp_open ()で0以外を指定した場合、udp_open ()のdst_ip, dst_portを使用します。

< 出力パラメータ詳細 >

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

 注 意

`dst_ip`, `dst_port`の指定について

`udp_open()`で0以外を指定した場合、`udp_open()`で指定したパラメータを使用します。

`udp_open()`で0を指定した場合、`udp_send()`で指定したパラメータを使用します。

`udp_open()`で0を指定し、`udp_send()`でも0を指定した場合、アドレス不正エラー（エラーコード：0xFFFF0）でリターンしますのでユーザプログラムを修正してください。

`udp_receive ()`

[機能] この関数は、ソケットIDに対応したソケットからパラメータの `buf` にデータを受信する関数です。リターン値に処理結果を返します。この関数はパラメータの `tim` に受信待ち時間の指定が可能ですが、受信待ち時間以内であってもデータを受信した時点でリターンします。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct ureceive_p { short s_id; short notuse; char *buf; long tim; }; { short (*udp_receive)(); short rtn; struct ureceive_p *padr; { udp_receive =(short(*) ()) 0x874172; { rtn = (*udp_receive) (padr); } } </pre>	<pre> struct ureceive_p { short s_id; short notuse; char *buf; long tim; }; { short (*udp_receive)(); short rtn; struct ureceive_p *padr; { udp_receive =(short(*) ()) 0x8F4172; { rtn = (*udp_receive) (padr); } } </pre>

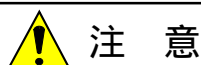
[パラメータ]

< 入力パラメータ詳細 >

`padr` : 入力パラメータの先頭アドレス
`padr -> s_id` : ソケットID
`padr -> notuse` : 0 固定 (未使用)
`padr -> buf` : 受信バッファの先頭アドレス
`padr -> tim` : 受信待ち時間 (ms : 0 ~ 86400000 (24時間))

< 出力パラメータ詳細 >

リターン値 : 処理結果または、エラーコードが返ります。
(0) 正常終了 (受信データなし)
(0x0001 ~ 0x05C0) 正常終了 (受信したバイト数)
(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。



注 意

`udp_receive ()` 関数は、パケットごとの受信を行いますので、バッファエリアを1472バイト確保してください。

4 利用の手引き

`route_list()`

[機能] この関数は、経路情報（経路情報テーブルサイズは最大16）を取得する関数です。リターン値には取得したエントリ数を返します。パラメータの `len` に 0 を指定すると取得エントリ数のみ返します。 `len` は16バイトの倍数を指定してください。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct lstrt_p { short len; short notues; void *buf; }; { short (*route_list)(); short rtn; struct lstrt_p *padr; { route_list = (short(*) ())0x874178; { rtn = (*route_list)(padr); { </pre>	<pre> struct lstrt_p { short len; short notues; void *buf; }; { short (*route_list)(); short rtn; struct lstrt_p *padr; { route_list = (short(*) ())0x8F4178; { rtn = (*route_list)(padr); { </pre>

[パラメータ]

< 入力パラメータ詳細 >

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > l e n` : データ長（バイト数：16の倍数）
`p a d r - > n o t u e s` : 0 固定（未使用）
`p a d r - > b u f` : データの先頭アドレス

< 出力パラメータ詳細 >

リターン値 : 取得したエントリ数が返ります。
(0) エントリなし
(0x0001 ~ 0x0010) 取得エントリ数

取得データ構造 (`b u f` の内容)

```

typedef struct {
    unsigned long dstaddr      : 相手局の I P アドレス
    unsigned long gatewayaddr  : ゲートウェイの I P アドレス
    unsigned short metric      : メトリック ( ゲートウェイの経由数 )
    unsigned short rt_types    : タイプ
    unsigned short refcnt      : 参照カウンタ
    unsigned short notuse      : ( 未使用 )
}routeentry

```



```
route_del ( )
```

[機能] この関数は、経路情報テーブルから経路情報を削除する関数です。リターン値に処理結果を返します。

[リンク手順]

C言語	
メイン	サブ
<pre>struct delrt_p { long dstaddr; long gatewayaddr; }; \ short (*route_del)(); short rtn; struct delrt_p *paddr; \ route_del =(short(*) ()) 0x87417E; \ rtn = (*route_del)(paddr); \</pre>	<pre>struct delrt_p { long dstaddr; long gatewayaddr; }; \ short (*route_del)(); short rtn; struct delrt_p *paddr; \ route_del =(short(*) ()) 0x8F417E; \ rtn = (*route_del)(paddr); \</pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > d s t a d d r` : 相手局のIPアドレス
`p a d r - > g t w a y a d d r` : ゲートウェイIPアドレス

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。
 (0) 正常終了
 (0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

`route_add()`

[機能] この関数は、経路情報テーブルに経路情報を登録する関数です。リターン値に処理結果を返します。

[リンク手順]

C 言語	
メイン	サブ
<pre>struct addrt_p { long dstaddr; long gatewayaddr; short metric; }; { short (*route_add)(); short rtn; struct addrt_p *padr; { route_add = (short(*) ())0x874184; { rtn = (*route_add)(padr); {</pre>	<pre>struct addrt_p { long dstaddr; long gatewayaddr; short metric; }; { short (*route_add)(); short rtn; struct addrt_p *padr; { route_add = (short(*) ())0x8F4184; { rtn = (*route_add)(padr); {</pre>

[パラメータ]

<入力パラメータ詳細>

`padr` : 入力パラメータの先頭アドレス
`padr ->dstaddr` : 相手局のIPアドレス
`padr ->gatewayaddr` : ゲートウェイのIPアドレス
`padr ->metric` : メトリック(ゲートウェイの経由数)

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。
(0) 正常終了
(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

arp_list ()

[機能] この関数は、ARP情報（ARP情報テーブルサイズは最大32）を取得する関数です。リターン値には取得したエントリ数を返します。パラメータのlenに0を指定すると取得エントリ数のみ返します。lenは12バイトの倍数を指定してください。

[リンク手順]

C言語	
メイン	サブ
<pre> struct lstart_p { short len; short notuse; void *buf; }; { short (*arp_list)(); short rtn; struct lstart_p *pdr; { arp_list=(short (*)())0x87418A; } rtn = (*arp_list)(pdr); } </pre>	<pre> struct lstart_p { short len; short notuse; void *buf; }; { short (*arp_list)(); short rtn; struct lstart_p *pdr; { arp_list=(short (*)())0x8F418A; } rtn = (*arp_list)(pdr); } </pre>

[パラメータ]

< 入力パラメータ詳細 >

p a d r : 入力パラメータの先頭アドレス
p a d r - > l e n : データ長（バイト数：12の倍数）
p a d r - > n o t u s e : 0固定（未使用）
p a d r - > b u f : データの先頭アドレス

< 出力パラメータ詳細 >

リターン値 : 取得したエントリ数が返ります。
(0) エントリなし
(0x0001 ~ 0x0020) 取得エントリ数

取得データ構造 (b u f の内容)

```

typedef struct {
    unsigned long ip_addr : 相手局のIPアドレス
    unsigned char et_addr(6) : 相手局の物理アドレス
    unsigned char ar_timer : タイマ
    unsigned char ar_flags : フラグ
}arpt-t

```

4 利用の手引き

arp_del ()

[機能] この関数は、ARP情報テーブルからARP情報を削除する関数です。リターン値に処理結果を返します。

[リンク手順]

C 言語	
メイン	サブ
<pre>struct delarp_p { unsigned long ipaddr; unsigned char etaddr[6]; }; } short (*arp_del)(); short rtn; struct delarp_p *pdr; } arp_del =(short(*) ()) 0x874190; } rtn = (*arp_del)(pdr); }</pre>	<pre>struct delarp_p { unsigned long ipaddr; unsigned char etaddr[6]; }; } short (*arp_del)(); short rtn; struct delarp_p *pdr; } arp_del =(short(*) ()) 0x8F4190; } rtn = (*arp_del)(pdr); }</pre>

[パラメータ]

< 入力パラメータ詳細 >

p a d r : 入力パラメータの先頭アドレス
p a d r - > i p a d d r : 相手局のIPアドレス
p a d r - > e t a d d r [6] : 相手局の物理アドレス

< 出力パラメータ詳細 >

リターン値 : 処理結果が返ります。
(0) 正常終了
(0xF000 ~ 0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

arp_add ()

[機能] この関数は、ARP情報テーブルにARP情報を登録する関数です。リターン値に処理結果を返します。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct addarp_p { long ipaddr; char etaddr[6]; short flag; }; { short (*arp_add)(); short rtn; struct addarp_p *padr; { arp_add =(short(*) ()) 0x874196; { rtn = (*arp_add)(padr); } } </pre>	<pre> struct addarp_p { long ipaddr; char etaddr[6]; short flag; }; { short (*arp_add)(); short rtn; struct addarp_p *padr; { arp_add =(short(*) ()) 0x8F4196; { rtn = (*arp_add)(padr); } } </pre>

[パラメータ]

< 入力パラメータ詳細 >

padr : 入力パラメータの先頭アドレス

padr -> ipaddr : 相手局のIPアドレス

padr -> etaddr[6] : 相手局の物理アドレス

padr -> flag : フラグ(0固定)

< 出力パラメータ詳細 >

リターン値 : 処理結果が返ります。

(0) 正常終了

(0xF000~0xFFFF) エラーコードは、「7.3 エラーと対策」を参照してください。

4 利用の手引き

`getconfig()`

[機能] この関数は、コンフィグレーションブロックを取得する関数です。リターン値に処理結果を返します。

[リンク手順]

C 言語	
メイン	サブ
<pre> struct config_p { void *config_ptr; }; { short (*getconfig)(); short rtn; struct config_p *padr; { getconfig = (short(*) ())0x87419C; } rtn = (*getconfig)(padr); } </pre>	<pre> struct config_p { void *config_ptr; }; { short (*getconfig)(); short rtn; struct config_p *padr; { getconfig = (short(*) ())0x8F419C; } rtn = (*getconfig)(padr); } </pre>

[パラメータ]

<入力パラメータ詳細>

`p a d r` : 入力パラメータの先頭アドレス
`p a d r - > c o n f i g _ p t r` : コンフィグレーションブロックの先頭アドレス

<出力パラメータ詳細>

リターン値 : 処理結果が返ります。
 (0) 正常終了

<コンフィグレーションブロックの詳細>

コンフィグレーションブロックは、下記データ構造となります。

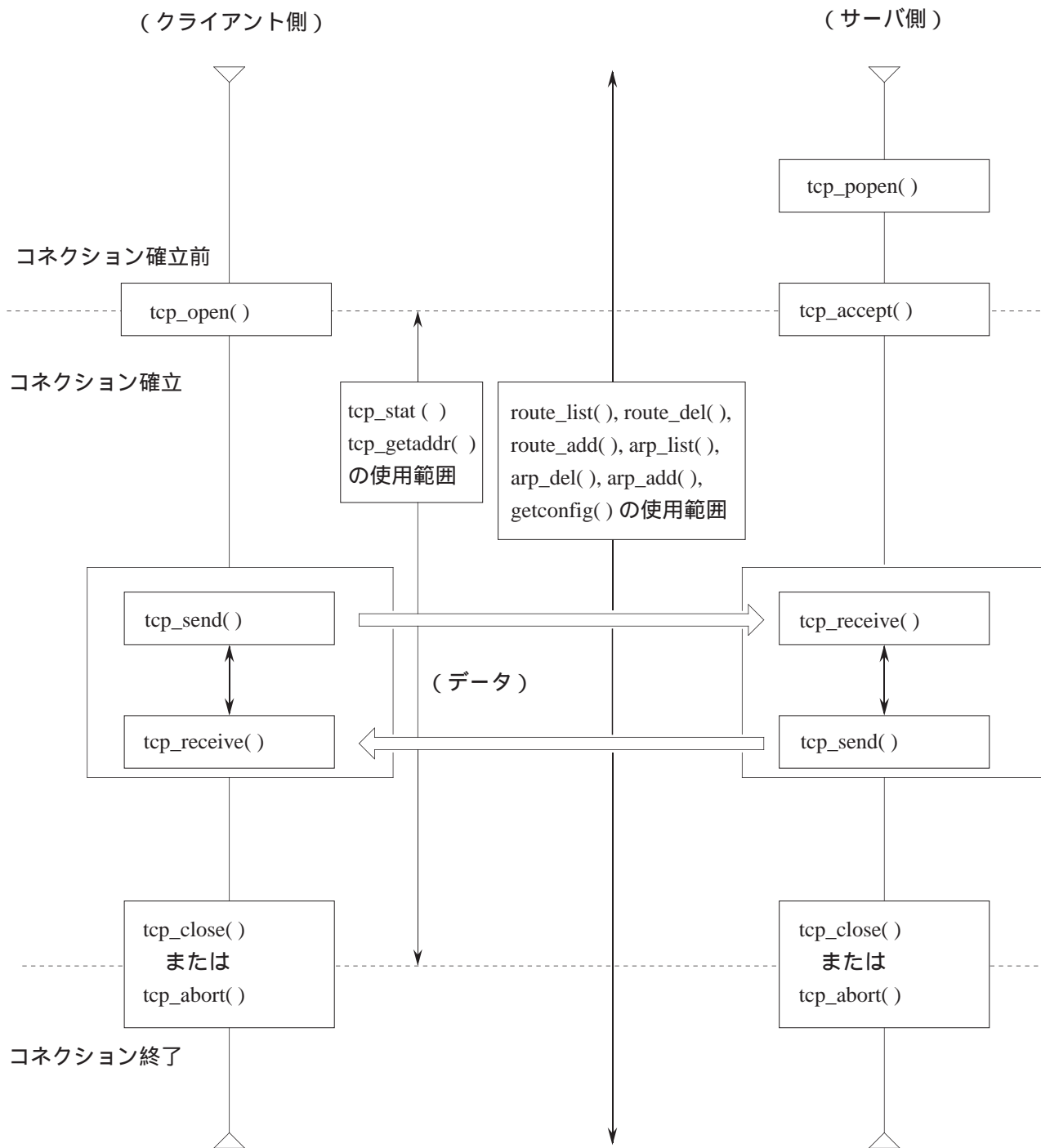
```

struct config_ptr {
    long ip_addr;      自局のIPアドレス(ネットワークオーダ)(任意)
    long netmask;     サブネットマスク (任意)
    long broadcast;  ブロードキャストアドレス (任意)
    char tcp_num;    最大TCPソケット数 (16)
    char udp_num;    最大UDPソケット数 (8)
    char rt_num;     経路情報テーブルサイズ (16)
    char arp_num;    ARP情報テーブルサイズ (32)
    short tcp_win;   TCPの送受信ウィンドウサイズ (1024)
};

```

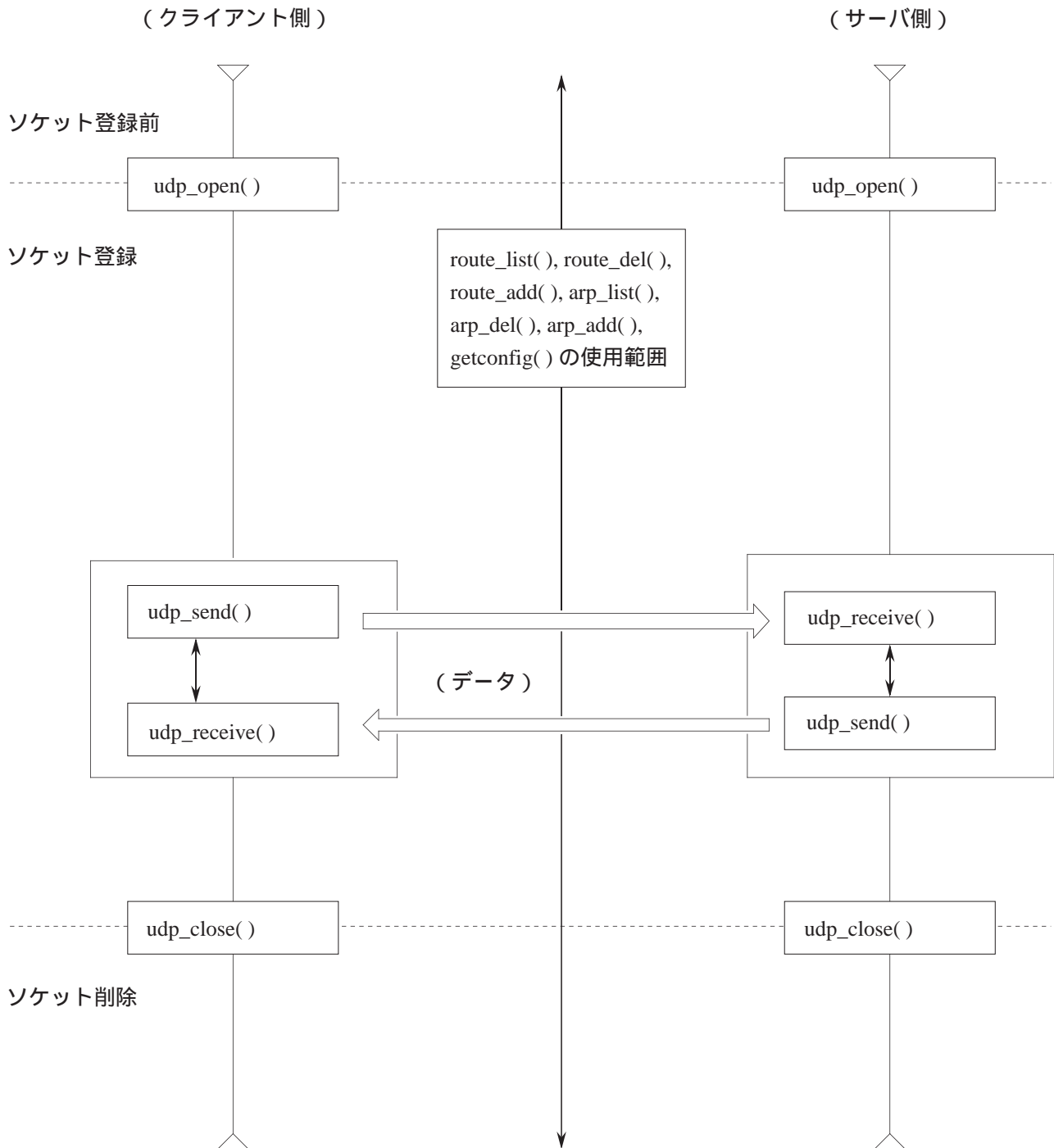
4.8 ソケットハンドラ発行手順例

4.8.1 TCP/IPプログラム使用例



4 利用の手引き

4.8.2 UDP/IPプログラム使用例

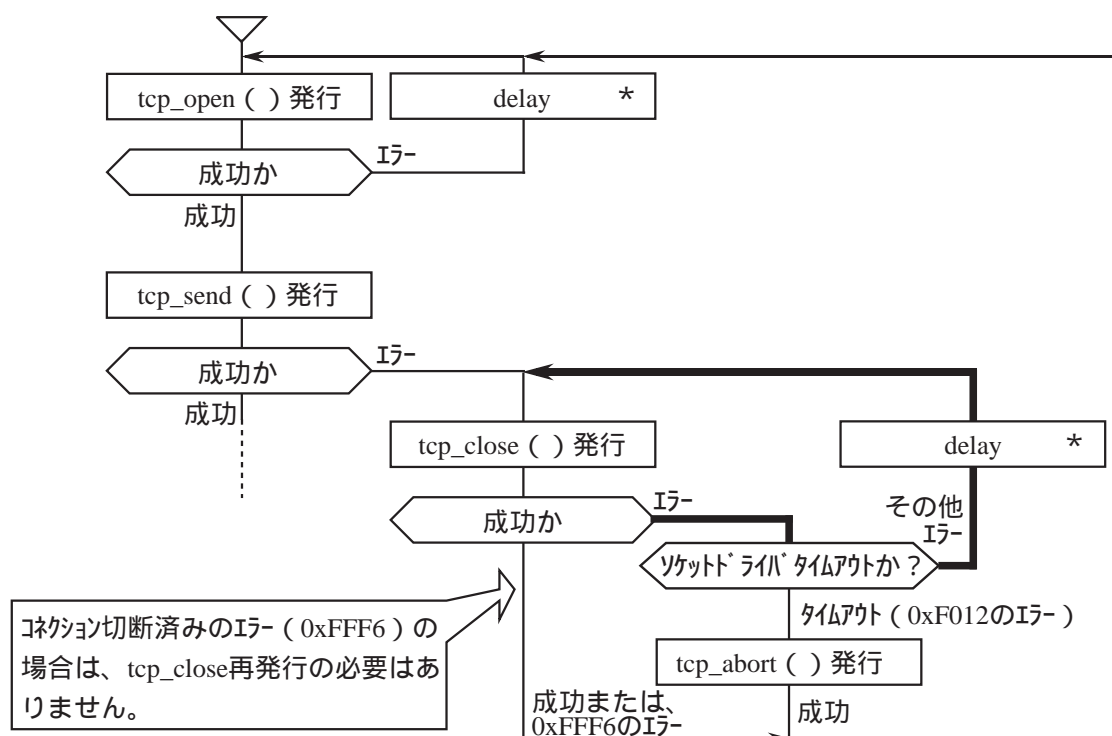


注 意

S10/2 のイーサネットモジュール（型式：LWE550）を使用する場合は、下記の内容に注意してください。

1. tcp_closeのエラー処理

ソケットハンドラ関数のリターンコードがエラーとなったためにtcp_closeを発行する場合、tcp_closeのリターンコードもチェックしエラーの場合には「ソケットハンドラ検出のエラーコード表」に従って再発行してください。エラーのままにしてtcp_closeを再発行しないと再コネクションできなかつたり、浮いたソケットが発生する可能性があります。下図にソケットハンドラの発行例を示します。



（注）上記はudp_closeのエラー処理のときも同じです。

2. 同一ソケットに対する非同期アクセスの禁止

1つのソケットに対し、非同期に複数のソケットライブラリ関数を実行すると関数の実行結果がエラーとなる場合があります。複数のタスクで同じソケットに対してソケットライブラリ関数を実行するとこのような現象が発生しやすいので、1つのソケットに対して1つのタスクで処理してください。

* : delayマクロ命令に関しては、「ソフトウェアマニュアル 概説&マクロ仕様 コンパクトPMS V5（マニュアル番号 SAJ-3-201）」を参照してください。

4 利用の手引き

注 意

3. 送信タイムアウト検出時間

LWE550でソケットライブラリ関数を発行し、通信異常や相手装置のダウンなどによりACKパケットのタイムアウトが発生した場合、タイムアウト検出時間は下表となります。したがって、ソケットハンドラのタイムアウトを検出し再発行もしくは再コネクションしても下表の時間がかかります。システム設計時には必ず通信エラーが発生することを前提として、下表タイムアウト時間に問題がないかを確認してください。

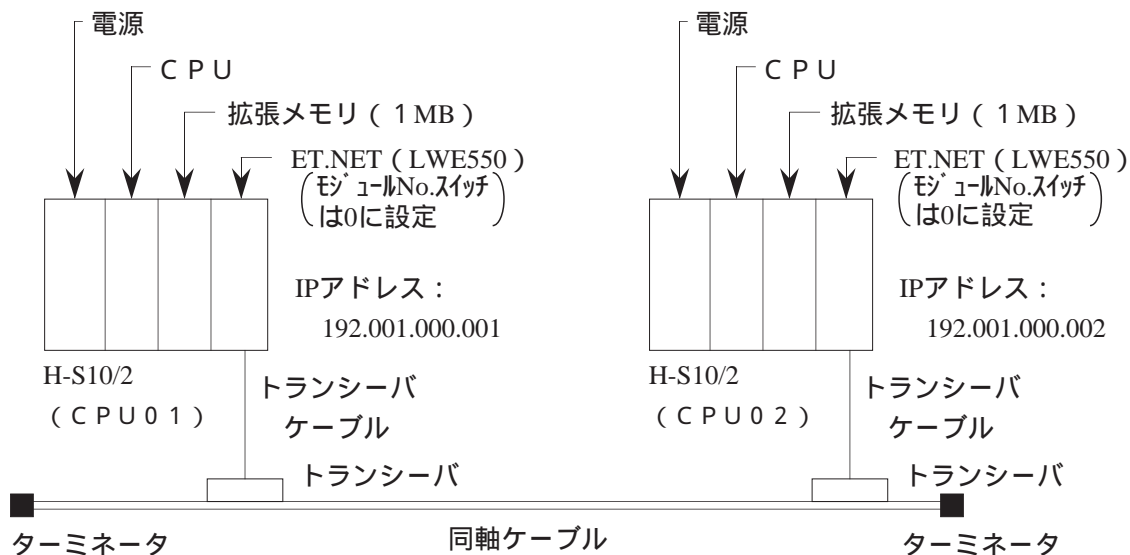
項 目	検出時間	内 容
tcp_openタイムアウト検出時間 (SYNのリトライ間隔)	75秒	相手装置からの応答がない場合、下記間隔でSYNのリトライを行います。 6秒, 12秒, 24秒, 33秒
tcp_sendタイムアウト検出時間 (SENDのリトライ間隔)	30秒	相手装置からの応答がない場合、下記間隔で送信リトライを行います。 1秒, 2秒, 4秒, 8秒, 16秒 ただし、tcp_send発行から30秒でソケットドライバタイムアウト (リターンコード0xF012) を検出します。
tcp_closeタイムアウト検出時間 (FINのリトライ間隔)	30秒	相手装置からFINを受信し、正常にコネクションが切断された場合は、すぐに終了します。 LWE550からFINを送信してコネクション切断する場合も、すぐに終了します。 相手装置からの応答がない場合、下記間隔でFINのリトライを行います。 1秒, 2秒, 4秒, 8秒, 16秒 ただし、tcp_close発行から30秒でソケットドライバタイムアウト (リターンコード0xF012) を検出するので、tcp_abortを発行して、コネクションを切断してください。
リスンス タイムアウト 検出時間	tcp_close, tcp_send, udp_close	30秒
	tcp_abort, route_list, route_del, route_add, arp_list, arp_del, arp_add, getconfig, udp_send, tcp_getaddr, tcp_stat	10秒
		ソケットハンドラがマイクロプログラム に対してコマンド発行後、無応答を検出 する時間

5 プログラム例

5 プログラム例

5.1 ソケットハンドラによるCPU間通信プログラム例

5.1.1 システム構成



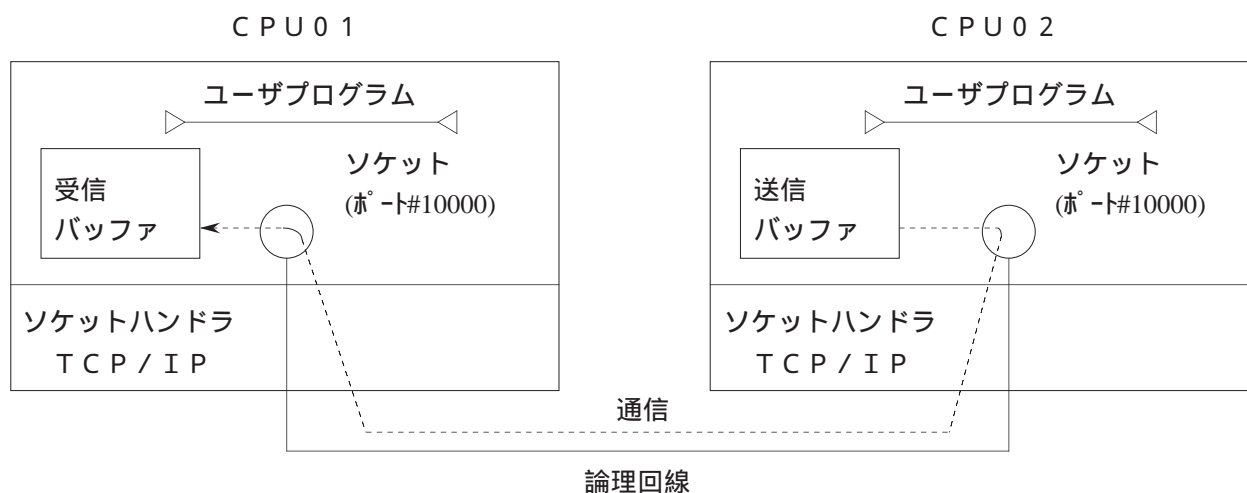
システム構成品一覧

品名	形式	数量	備考
電源	LWV000	2	
CPU	LWP000	2	
拡張メモリ	LWM414	2	
ET.NET	LWE550	2	
トランシーバケーブル	HDC4360	2	メーカー：(株)日立製作所
トランシーバ	HLT-200TB	2	メーカー：日立電線(株)
同軸ケーブル	HBN-CX-100	1	メーカー：日立電線(株)
ターミネータ	HBN-T-NJ	2	メーカー：日立電線(株)

5.1.2 プログラム構成

プログラム構成を以下に示します。CPU 01のET.NETモジュールとCPU 02のET.NETモジュールを論理回線で接続し、CPU 02のET.NETモジュールは1024バイトのデータを送信し、CPU 01のET.NETモジュールは1024バイトのデータを受信するプログラムです。

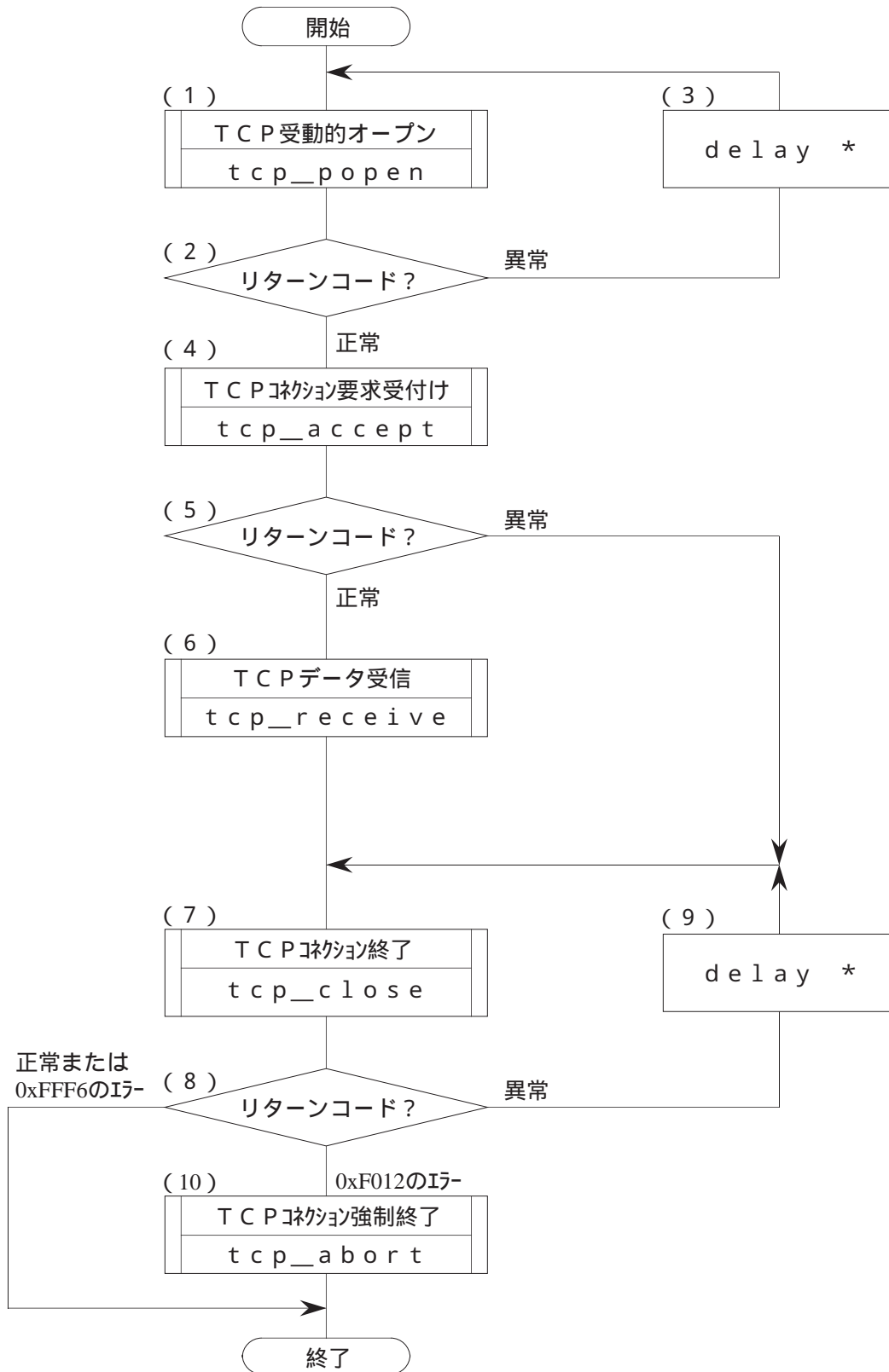
このプログラムを動作させる場合、必ずCPU 01からユーザプログラムを起動してください。



項目		CPU	
		CPU 01	CPU 02
機能		受信	送信
送信バッファ		_____	アドレス: 0x1E6000 バイト数: 1024
受信バッファ		アドレス: 0x1E6000 バイト数: 1024	_____
ポート番号		10000	10000
ソケット ハンドラの 先頭アドレス	tcp_open()	0x874100	0x874100
	tcp_popen()	0x874106	0x874106
	tcp_accept()	0x87410C	0x87410C
	tcp_close()	0x874112	0x874112
	tcp_abort()	0x87411E	0x87411E
	tcp_getaddr()	0x874124	0x874124
	tcp_stat()	0x87412A	0x87412A
	tcp_send()	0x874130	0x874130
	tcp_receive()	0x874136	0x874136
	udp_open()	0x874160	0x874160
	udp_close()	0x874166	0x874166
	udp_send()	0x87416C	0x87416C
	udp_receive()	0x874172	0x874172
	route_list()	0x874178	0x874178
	route_del()	0x87417E	0x87417E
	route_add()	0x874184	0x874184
	arp_list()	0x87418A	0x87418A
arp_del()	0x874190	0x874190	
arp_add()	0x874196	0x874196	
getconfig()	0x87419C	0x87419C	

5 プログラム例

5.1.3 CPU01側プログラムのフローチャート



- (1) ポート番号を10000としてソケットの登録を行い、そのソケットを受動状態にします。
- (2) 登録されたソケットIDはリターンコードで返されますので、リターンコードが正のときは正常に登録されたものと見なします。
- (3) `delay` マクロを発行し、(1), (2)を繰り返します。
- (4) CPU02側からのコネクション要求に対してコネクション要求を受付けます。
- (5) リターンコードにより、正常か異常かを判定します。
- (6) CPU02側から送信されたデータを受信バッファに取込みます。
- (7) 確立したコネクションを終了させます。
- (8) リターンコードにより、正常か異常かを判定します。ただし、0xFFFF6のエラーの場合、正常と同様に終了し、0xF012のエラーの場合、(10)へ進みます。
- (9) `delay` マクロを発行し、(7), (8)を繰り返します。
- (10) 相手局からの応答が返らないので、コネクションを強制終了します。

* : `delay` マクロ命令に関しては、「ソフトウェアマニュアル 概説&マクロ仕様 コンパクトPMS V5 (マニュアル番号 SAJ-3-201)」を参照してください。

5 プログラム例

5.1.4 CPU01側のC言語プログラム例

```
#define TCP_POPEN 0x874106L /* tcp_popen( ) 先頭アドレス(メイン) */
#define TCP_ACCEPT 0x87410CL /* tcp_accept( ) 先頭アドレス(メイン) */
#define TCP_CLOSE 0x874112L /* tcp_close( ) 先頭アドレス(メイン) */
#define TCP_RECEIVE 0x874136L /* tcp_receive( ) 先頭アドレス(メイン) */
#define TCP_ABORT 0x87411EL /* tcp_abort( ) 先頭アドレス */
#define IPADDR 0xC0010002L /* 相手局IPアドレス */
#define RBUFADDR 0x1E6000L /* 受信バッファ先頭アドレス */
#define PARADDR 0x1E5000L /* パラメータ先頭アドレス */

struct popen_p{
    long dst_ip; /* 相手局のIPアドレス */
    short dst_port; /* 相手局のポート番号 */
    short src_port; /* 自局のポート番号 */
    char listennum; /* ACCEPTされていない接続の最大数 */
    char ttl; /* Time to live */
};

struct accept_p{
    short s_id; /* ソケットID */
};

struct receive_p{
    short s_id; /* ソケットID */
    short len; /* バッファ長 */
    char *buf; /* バッファ先頭アドレス */
    long tim; /* 受信待ち時間(ms) */
};

struct close_p{
    short s_id; /* ソケットID */
};

struct abort_p{
    short s_id; /* ソケットID */
};

/*****
/* task2:サ-バ (CPU01) */
*****/
main()
{
    register short ( *tcp_popen )( );
    register short ( *tcp_accept )( );
    register short ( *tcp_receive )( );
    register short ( *tcp_close )( );
    register short ( *tcp_abort )( );
    long time;
    short rtn;
    char *rbuf;
    struct popen_p *popen;
    struct accept_p *accpt;
    struct receive_p *recv;
    struct close_p *close;
    struct abort_p *abort;
```



```

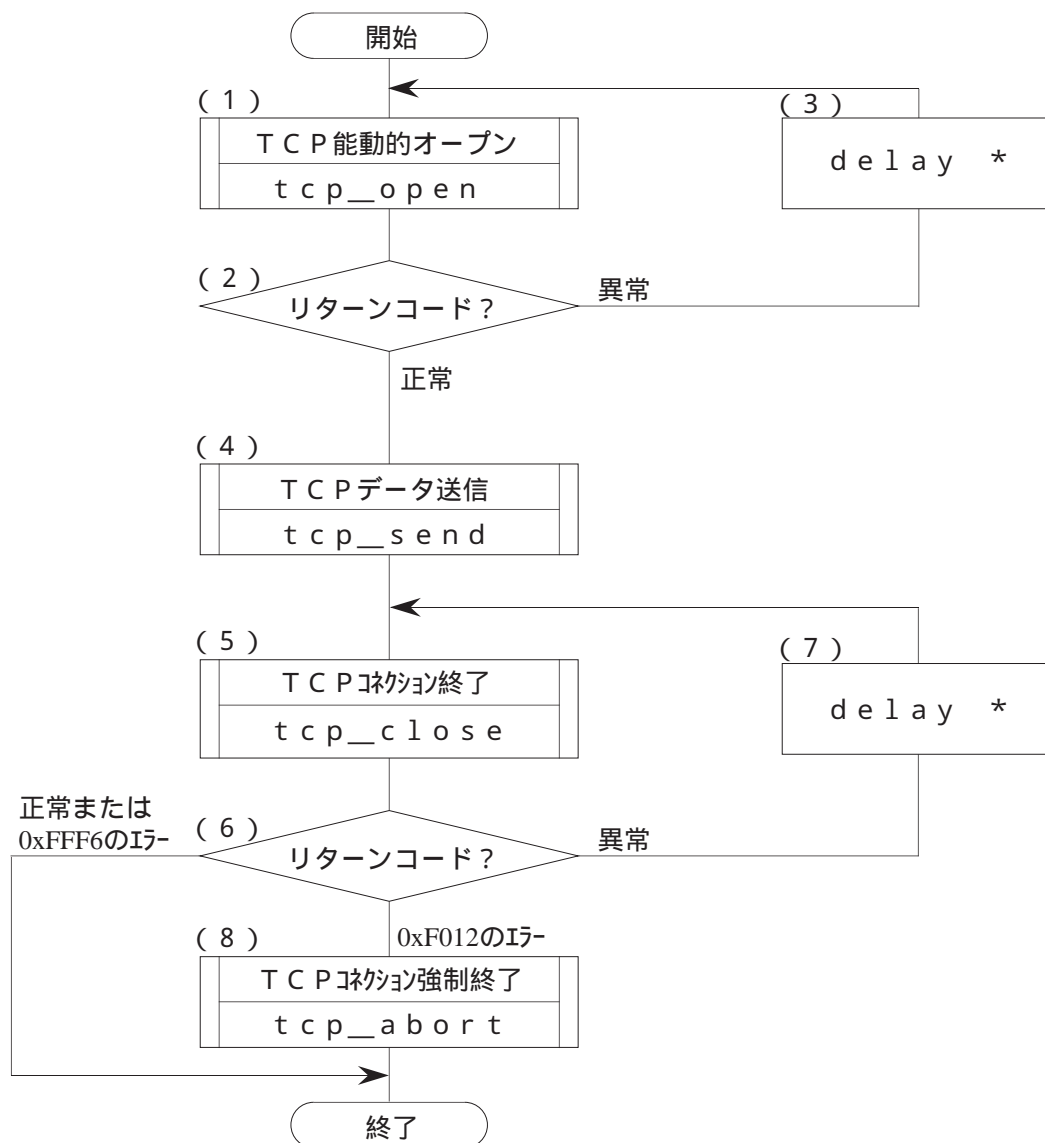
popen = (struct popen_p *)PARADDR;          /* 入力パラメータ先頭アドレス*/
acctp = (struct accept_p *)(popen + 1);
recv = (struct receive_p *)(acctp + 1);
close = (struct close_p *)(recv + 1);
abort = (struct abort_p *)(close + 1);

while( 1 ){
    popen->dst_ip    = IPADDR;                /* 相手局のIPアドレス */
    popen->dst_port  = 10000;                /* 相手局のポート番号 */
    popen->src_port  = 10000;                /* 自局のポート番号 */
    popen->listennum = 0;                    /* ACCEPTされていない */
                                           /* 接続の最大数 */
    popen->tll      = 0;                      /* Time to live */
    tcp_popen      = ( short (*)())TCP_POPE;
    rtn            = (tcp_popen)(popen);     /* TCP受動的オープン */
    if( rtn > 0 ){                            /* リターンコード 正常? */
        break;
    }
    time = 100;                               /* 100ms Delay発行 */
    delay( &time);
}
acctp->s_id = rtn;                            /* ソケットID */
tcp_accept = ( short (*)())TCP_ACCEPT;
rtn        = (tcp_accept)(acctp);           /* TCPコネクション要求受付 */
recv->s_id  = rtn;                            /* ソケットID */
if( rtn > 0 ){                                /* リターンコード 正常? */
    recv->len = 1024;                          /* 受信バッファバイト長 */
    recv->buf = ( char *)RBUFADDR;            /* 受信バッファ先頭アドレス */
    recv->tim = 60000;                          /* 受信待ち時間(ms) */
    tcp_receive = ( short (*)())TCP_RECEIVE;
    rtn         = (tcp_receive)(recv);        /* TCP受信 */
    close->s_id = recv->s_id;                  /* ソケットID */
} else {
    close->s_id = acctp->s_id;                  /* ソケットID */
}
while( 1 ){
    tcp_close = ( short (*)())TCP_CLOSE;
    rtn = (tcp_close)(close);                 /* TCPコネクション終了 */
    if( rtn == 0 || rtn == ( short )0xFFFF6 ){
        break;
    } else if( rtn == ( short )0xF012 ){
        tcp_abort = ( short (*)())TCP_ABORT;
        rtn = (tcp_abort)(abort);            /* TCPコネクション強制終了 */
        break;
    }
    time = 100;                               /* 100ms Delay発行 */
    delay( &time);
}
return;
}

```

5 プログラム例

5.1.5 CPU02側プログラムのフローチャート



- (1) ポート番号を10000としてソケットの登録を行い、そのソケットを能動状態にします。
- (2) 登録されたソケットIDはリターンコードで返されますので、リターンコードが正のときは正常に登録されたものと見なします。
- (3) delayマクロを発行し、(1)、(2)を繰り返します。
- (4) 送信バッファのデータをCPU01に送信します。
- (5) 確立したコネクションを終了させます。
- (6) リターンコードにより、正常か異常かを判定します。ただし、0xFFFF6のエラーの場合、正常と同様に終了し、0xF012のエラーの場合、(8)へ進みます。
- (7) delayマクロを発行し、(5)、(6)を繰り返します。
- (8) 相手局からの応答が返らないので、コネクションを強制終了します。

* : delayマクロ命令に関しては、「ソフトウェアマニュアル 概説&マクロ仕様 コンパクトPMS V5 (マニュアル番号 SAI-3-201)」を参照してください。

5.1.6 CPU 0 2 側の C 言語プログラム例

```

#define TCP_OPEN    0x874100L /* tcp_open( ) 先頭アドレス */
#define TCP_CLOSE   0x874112L /* tcp_close( ) 先頭アドレス */
#define TCP_SEND    0x874130L /* tcp_send( ) 先頭アドレス */
#define TCP_ABORT   0x87411EL /* tcp_abort ( ) 先頭アドレス */
#define IPADDR      0xC0010001L /* 相手局のIPアドレス */
#define SBUFADDR    0x1E6000L /* 送信バッファ先頭アドレス */
#define PARADDR     0x1E5000L /* パラメータ先頭アドレス */

struct open_p{
    long   dst_ip;      /* 相手局のIPアドレス */
    short  dst_port;   /* 相手局のポート番号 */
    short  src_port;   /* 自局のポート番号 */
    char   notuse;     /* 未使用 (0) */
    char   ttl;        /* Time to live */
};

struct send_p{
    short  s_id;       /* ソケットID */
    short  len;        /* 送信データバイト長 */
    char   *buf;      /* 送信データ先頭アドレス */
};

struct close_p{
    short  s_id;       /* ソケットID */
};

struct abort_p{
    short  s_id;       /* ソケットID */
};

/*****
/* task3:クライアント(CPU02) */
*****/
main()
{
    register short ( *tcp_open )( );
    register short ( *tcp_send )( );
    register short ( *tcp_close )( );
    register short ( *tcp_abort )( );
    long   time;
    short  rtn;
    struct open_p   *open;
    struct send_p   *send;
    struct close_p  *close;
    struct abort_p  *abort;

    open = (struct open_p *)PARADDR; /* 入力パラメータ先頭アドレス*/
    send = (struct send_p *) (open + 1);
    close = (struct close_p *) (send + 1);
    abort = (struct abort_p *) (close + 1);

    while( 1 ){
        open->dst_ip = IPADDR; /* 相手局のIPアドレス */
        open->dst_port = 10000; /* 相手局のポート番号 */
        open->src_port = 10000; /* 自局のポート番号 */

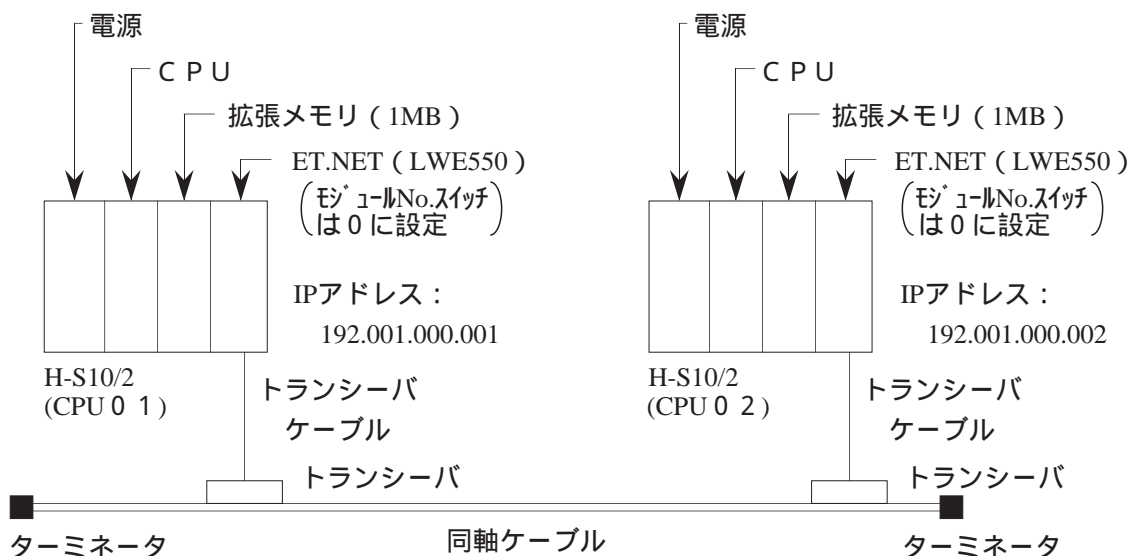
```

5 プログラム例

```
open->notuse = 0; /* 未使用 */
open->tll = 0; /* Time to live */
tcp_open = (short (*)())TCP_OPEN;
rtn = (tcp_open)(open); /* TCP能動的オープン */
if( rtn > 0 ){ /* リターンコード 正常? */
    break;
}
time = 100; /* 100ms Delay発行 */
delay( &time);
}
send->s_id = rtn; /* ソケットID */
send->len = 1024; /* 送信データバイト長 */
send->buf = ( char *)SBUFADDR; /* 送信データ先頭アドレス */
tcp_send = ( short (*)())TCP_SEND;
rtn = (tcp_send)(send); /* TCPデータ送信 */
close->s_id = send->s_id; /* ソケットID */
while( 1 ){
    tcp_close = ( short (*)())TCP_CLOSE;
    rtn = (tcp_close)(close); /* TCP接続終了 */
    if( rtn == 0 || rtn == ( short)0xFFFF6 ){
        break;
    } else if( rtn == ( short )0xF012 ){
        tcp_abort = ( short (*)())TCP_ABORT;
        rtn = (tcp_abort)(abort); /* TCP接続強制終了 */
        break;
    }
    time = 100; /* 100ms Delay発行 */
    delay( &time);
}
return;
}
```

5.2 ソケットハンドラによるCPU間連続通信プログラム例

5.2.1 システム構成



システム構成品一覧

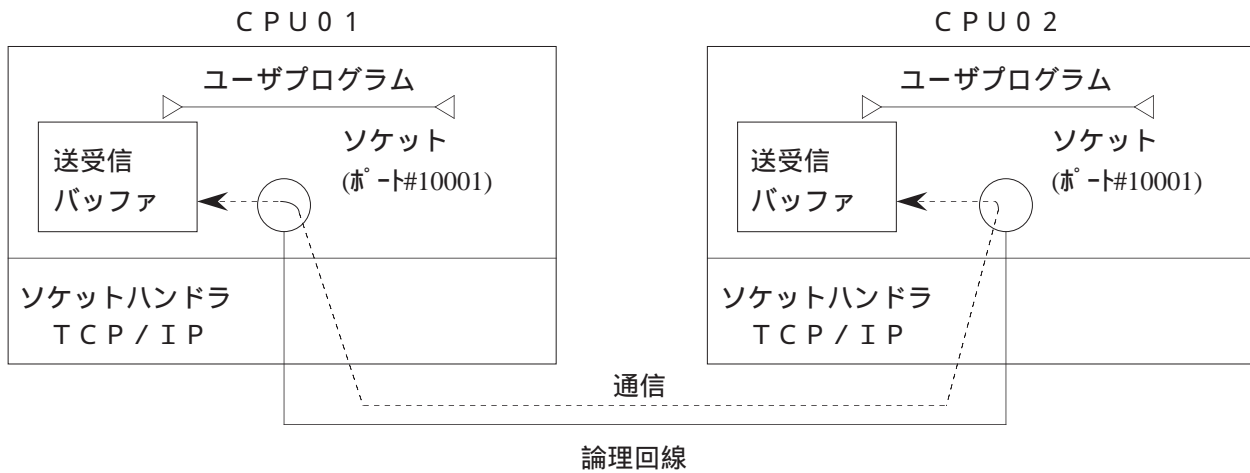
品名	形式	数量	備考
電源	LWV000	2	
CPU	LWP000	2	
拡張メモリ	LWM414	2	
ET.NET	LWE550	2	
トランシーバケーブル	HDC4360	2	メーカー：(株)日立製作所
トランシーバ	HLT-200TB	2	メーカー：日立電線(株)
同軸ケーブル	HBN-CX-100	1	メーカー：日立電線(株)
ターミネータ	HBN-T-NJ	2	メーカー：日立電線(株)

5 プログラム例

5.2.2 プログラム構成

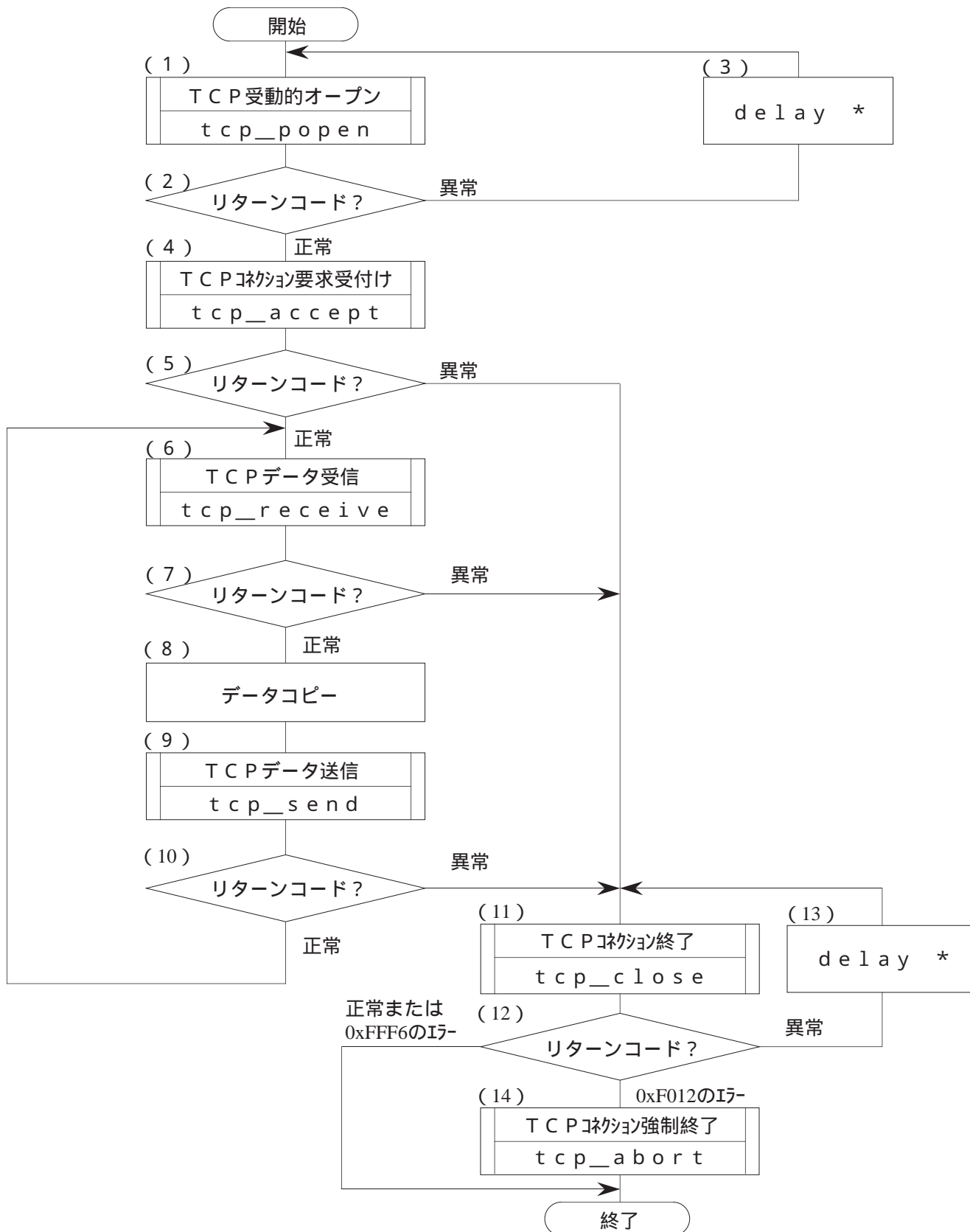
プログラム構成を以下に示します。CPU01のET.NETモジュールとCPU02のET.NETモジュールを論理回線で接続し、CPU02のET.NETモジュールとCPU01のET.NETモジュールの間で1024バイトのデータを送受信するプログラムです。

このプログラムを動作させる場合、必ずCPU01からユーザプログラムを起動してください。



CPU		CPU01	CPU02
項目			
機能		送信/受信	送信/受信/比較
送信バッファ		アドレス: 0x1E1000 バイト数: 1024	アドレス: 0x1E1000 バイト数: 1024
受信バッファ		アドレス: 0x1E2000 バイト数: 1024	アドレス: 0x1E2000 バイト数: 1024
ポート番号		10001	10001
ソケット ハンドラの 先頭アドレス	tcp_open()	0x874100	0x874100
	tcp_popen()	0x874106	0x874106
	tcp_accept()	0x87410C	0x87410C
	tcp_close()	0x874112	0x874112
	tcp_abort()	0x87411E	0x87411E
	tcp_getaddr()	0x874124	0x874124
	tcp_stat()	0x87412A	0x87412A
	tcp_send()	0x874130	0x874130
	tcp_receive()	0x874136	0x874136
	udp_open()	0x874160	0x874160
	udp_close()	0x874166	0x874166
	udp_send()	0x87416C	0x87416C
	udp_receive()	0x874172	0x874172
	route_list()	0x874178	0x874178
	route_del()	0x87417E	0x87417E
	route_add()	0x874184	0x874184
	arp_list()	0x87418A	0x87418A
arp_del()	0x874190	0x874190	
arp_add()	0x874196	0x874196	
getconfig()	0x87419C	0x87419C	

5.2.3 CPU01側プログラムのフローチャート



5 プログラム例

- (1) ポート番号を10001としてソケットの登録を行い、そのソケットを受動状態にします。
- (2) 登録されたソケットIDはリターンコードで返されますので、リターンコードが正のときは正常に登録されたものと見なします。
- (3) `delay` マクロを発行し、(1), (2)を繰り返します。
- (4) CPU02側からの接続要求に対して接続要求を受付けます。
- (5) リターンコードにより、正常か異常かを判定します。
- (6) CPU02側から送信されたデータを受信バッファに取込みます。
- (7) リターンコードがエラーまたは、取込みデータなしの場合(11)を実行します。
- (8) 受信バッファのデータを送信バッファへコピーします。
- (9) 送信バッファのデータをCPU02に送信します。
- (10) リターンコードにより正常か異常を判定し、正常な場合は(6)~(10)を繰り返します。
- (11) 確立した接続を終了させます。
- (12) リターンコードにより、正常か異常かを判定します。ただし、0xFFF6のエラーの場合、正常と同様に終了し、0xF012のエラーの場合、(14)へ進みます。
- (13) `delay` マクロを発行し、(11), (12)を繰り返します。
- (14) 相手局からの応答が返らないので、接続を強制終了します。

* : `delay` マクロ命令に関しては、「ソフトウェアマニュアル 概説&マクロ仕様 コンパクト PMS V5 (マニュアル番号 SAJ-3-201)」を参照してください。

5.2.4 CPU01側のC言語プログラム例

```

#define TCP_POPEN 0x874106L /* tcp_popen( ) 先頭アドレス(メイン) */
#define TCP_ACCEPT 0x87410CL /* tcp_accept( ) 先頭アドレス(メイン) */
#define TCP_RECEIVE 0x874136L /* tcp_receive( ) 先頭アドレス(メイン) */
#define TCP_SEND 0x874130L /* tcp_send( ) 先頭アドレス(メイン) */
#define TCP_CLOSE 0x874112L /* tcp_close( ) 先頭アドレス(メイン) */
#define TCP_ABORT 0x87411EL /* tcp_abort( ) 先頭アドレス */
#define IPADDR 0xC0010002L /* 相手局IPアドレス */
#define SBUFADDR 0x1E1000L /* 送信バッファ先頭アドレス */
#define RBUFADDR 0x1E2000L /* 受信バッファ先頭アドレス */
#define PARADDR 0x1E5000L /* パラメータ先頭アドレス */

struct popen_p{
    long dst_ip; /* 相手局のIPアドレス */
    short dst_port; /* 相手局のポート番号 */
    short src_port; /* 自局のポート番号 */
    char listennum; /* ACCEPTされていない接続の最大数 */
    char ttl; /* Time to live */
};

struct accept_p{
    short s_id; /* ソケットID */
};

struct receive_p{
    short s_id; /* ソケットID */
    short len; /* バッファ長 */
    char *buf; /* バッファ先頭アドレス */
    long tim; /* 受信待ち時間(ms) */
};

struct send_p{
    short s_id; /* ソケットID */
    short len; /* 送信データバイト長 */
    char *buf; /* 送信データ先頭アドレス */
};

struct close_p{
    short s_id; /* ソケットID */
};
struct abort_p{
    short s_id; /* ソケットID */
};
/*****/
/* task2:サーバ(CPU01) */
/*****/
main()
{
    register short ( *tcp_popen )( );
    register short ( *tcp_accept )( );
    register short ( *tcp_receive )( );
    register short ( *tcp_send )( );
    register short ( *tcp_close )( );
    register short ( *tcp_abort )( );
    long time;
    short rtn, i;
    char *sbuf, *rbuf;
    struct popen_p *popen;
    struct accept_p *acctp;
    struct receive_p *recv;
    struct send_p *send;
    struct close_p *close;
    struct abort_p *abort;

    popen = (struct popen_p *)PARADDR; /* 入力パラメータ先頭アドレス */
    acctp = (struct accept_p *) (popen + 1);
    recv = (struct receive_p *) (acctp + 1);
    send = (struct send_p *) (recv + 1);
    close = (struct close_p *) (send + 1);
    abort = (struct abort_p *) (close + 1);

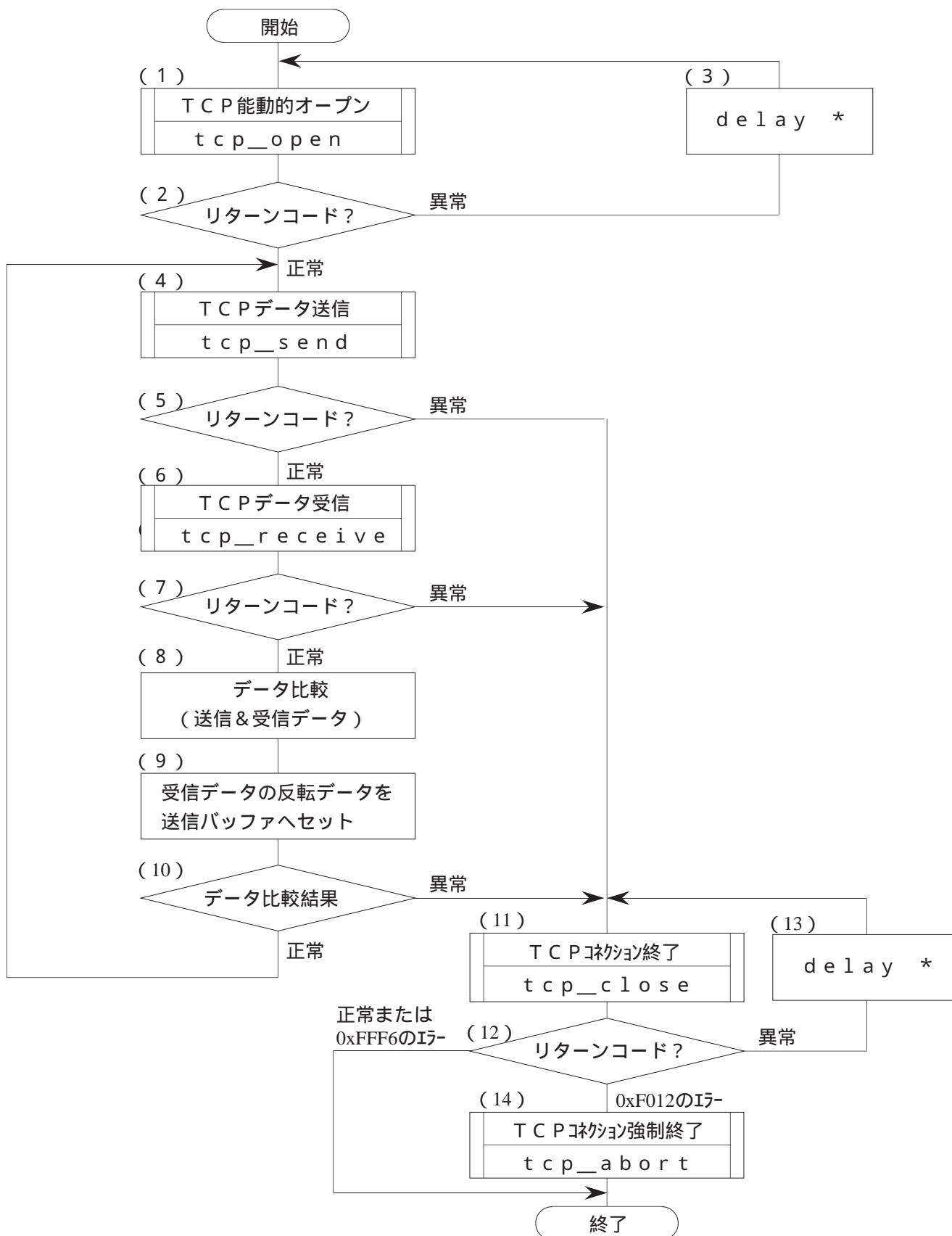
    while( 1 ){

```

5 プログラム例

```
popen->dst_ip   = IPADDR;           /* 相手局のIPアドレス */
popen->dst_port = 10001;           /* 相手局のポート番号 */
popen->src_port = 10001;           /* 自局のポート番号 */
popen->listennum = 0;              /* ACCEPTされていない */
                                   /* 接続の最大数 */
popen->ttl      = 0;               /* Time to live */
tcp_popen      = ( short (*)())TCP_POPEX;
rtn            = (tcp_popen)(popen); /* TCP受動的オープン */
if( rtn > 0 ){ /* リターンコード 正常? */
    break;
}
time = 100; /* 100ms Delay発行 */
delay( &time);
}
acct->s_id     = rtn; /* ソケットID */
tcp_accept    = ( short (*)())TCP_ACCEPT;
rtn           = (tcp_accept)(acct); /* TCPコネクション要求受付 */
if( rtn > 0 ){ /* リターンコード 正常? */
    rcv->s_id   = rtn; /* ソケットID */
    while( 1 ){
        rcv->len = 1024; /* 受信バッファのバイト長 */
        rcv->buf = ( char *)RBUFADDR; /* 受信バッファ先頭アドレス */
        rcv->tim = 60000; /* 受信待ち時間(ms) */
        tcp_receive = ( short (*)())TCP_RECEIVE;
        rtn         = (tcp_receive)(rcv); /* TCPデータ受信 */
        if( rtn < 0 ){ /* リターンコード 異常? */
            break;
        }
        sbuf = ( char *)SBUFADDR; /* 送信バッファ先頭アドレス */
        rbuf = ( char *)RBUFADDR; /* 受信バッファ先頭アドレス */
        for( i = 0 ; i < 1024 ; i++ ){
            sbuf[i] = rbuf[i];
        }
        send->s_id = rcv->s_id; /* ソケットID */
        send->len  = 1024; /* 送信データのバイト長 */
        send->buf  = ( char *)SBUFADDR; /* 送信データ先頭アドレス */
        tcp_send  = ( short (*)())TCP_SEND;
        rtn       = (*tcp_send)(send); /* TCPデータ送信 */
        if( rtn < 0 ){ /* リターンコード 異常? */
            break;
        }
    }
    close->s_id = rcv->s_id; /* ソケットID */
} else {
    close->s_id = acct->s_id; /* ソケットID */
}
while( 1 ){
    tcp_close = ( short (*)())TCP_CLOSE;
    rtn       = (tcp_close)(close); /* TCPコネクション終了 */
    if( rtn == 0 || rtn == ( short )0xFFFF ){
        break;
    } else if( rtn == ( short )0xF012 ){
        tcp_abort = ( short (*)())TCP_ABORT;
        rtn       = (tcp_abort)(abort); /* TCPコネクション強制終了 */
        break;
    }
    time = 100; /* 100ms Delay発行 */
    delay( &time);
}
return;
}
```

5.2.5 CPU02側プログラムのフローチャート



5 プログラム例

- (1) ポート番号を10001としてソケットの登録を行い、そのソケットを能動状態にします。
- (2) 登録されたソケットIDはリターンコードで返されますので、リターンコードが正のときは正常に登録されたものと見なします。
- (3) `delay`マクロを発行し、(1)、(2)を繰り返します。
- ➔ (4) 送信バッファのデータをCPU01側に送信します。
- (5) リターンコードにより、正常か異常かを判定します。
- (6) CPU01から送信されたデータを受信バッファへ取込みます。
- (7) リターンコードにより、正常か異常かを判定します。
- (8) 自局の送信バッファと受信バッファのデータの比較を行います。
- (9) 受信データの反転データを送信バッファへコピーします。
- (10) 比較結果の判定を行い、正常な場合は(4)~(10)を繰り返します。
- (11) 確立した接続を終了させます。
- (12) リターンコードにより、正常か異常かを判定します。ただし、0xFFF6のエラーの場合、正常と同様に終了し、0xF012のエラーの場合、(14)へ進みます。
- (13) `delay`マクロを発行し、(11)、(12)を繰り返します。
- (14) 相手局からの応答が返らないので、接続を強制終了します。

* : `delay`マクロ命令に関しては、「ソフトウェアマニュアル 概説&マクロ仕様 コンパクト PMS V5 (マニュアル番号 SAJ-3-201)」を参照してください。

5.2.6 CPU02側のC言語プログラム例

```

#define TCP_OPEN    0x874100L /* tcp_open( ) 先頭アドレス(メイン) */
#define TCP_CLOSE  0x874112L /* tcp_close( ) 先頭アドレス(メイン) */
#define TCP_SEND   0x874130L /* tcp_send( ) 先頭アドレス(メイン) */
#define TCP_RECEIVE 0x874136L /* tcp_receive( ) 先頭アドレス(メイン) */
#define TCP_ABORT  0x87411EL /* tcp_abort( ) 先頭アドレス */
#define IPADDR     0xC0010001L /* 相手局のIPアドレス */
#define SBUFADDR   0x1E1000L /* 送信バッファ先頭アドレス */
#define RBUFADDR   0x1E2000L /* 受信バッファ先頭アドレス */
#define PARADDR    0x1E5000L /* パラメータ先頭アドレス */

struct open_p{
    long  dst_ip;          /* 相手局のIPアドレス */
    short dst_port;       /* 相手局のポート番号 */
    short src_port;       /* 自局のポート番号 */
    char  notuse;         /* 未使用 (0) */
    char  ttl;            /* Time to live */
};

struct send_p{
    short s_id;           /* ソケットID */
    short len;            /* 送信データバイト長 */
    char  *buf;           /* 送信データ先頭アドレス */
};

struct receive_p{
    short s_id;           /* ソケットID */
    short len;            /* バッファ長 */
    char  *buf;           /* バッファ先頭アドレス */
    long  tim;           /* 受信待ち時間(ms) */
};

struct close_p{
    short s_id;           /* ソケットID */
};

struct abort_p{
    short s_id;           /* ソケットID */
};

/*****
/* task3:タライント(CPU02) */
*****/
main()
{
    register short ( *tcp_open )( );
    register short ( *tcp_send )( );
    register short ( *tcp_receive )( );
    register short ( *tcp_close )( );
    register short ( *tcp_abort )( );
    long  time;
    short rtn, i, cerr_flg;
    char  *sbuf, *rbuf;
    struct open_p  *open;
    struct send_p  *send;
    struct receive_p *recv;
    struct close_p *close;
    struct abort_p *abort;

    open = (struct open_p *)PARADDR;          /* 入力パラメータ先頭アドレス*/
    send = (struct send_p *) (open + 1);
    recv = (struct receive_p *) (send + 1);
    close = (struct close_p *) (recv + 1);
    abort = (struct abort_p *) (close + 1);

    sbuf = ( char *)SBUFADDR;                /* 送信バッファ先頭アドレス */
    for( i = 0 ; i < 1024 ; i++ ){
        sbuf[i] = 0x55;
    }
    while( 1 ){
        open->dst_ip    = IPADDR;             /* 相手局のIPアドレス */
        open->dst_port  = 10001;             /* 相手局のポート番号 */
        open->src_port  = 10001;             /* 自局のポート番号 */
        open->notuse    = 0;                 /* 未使用 */

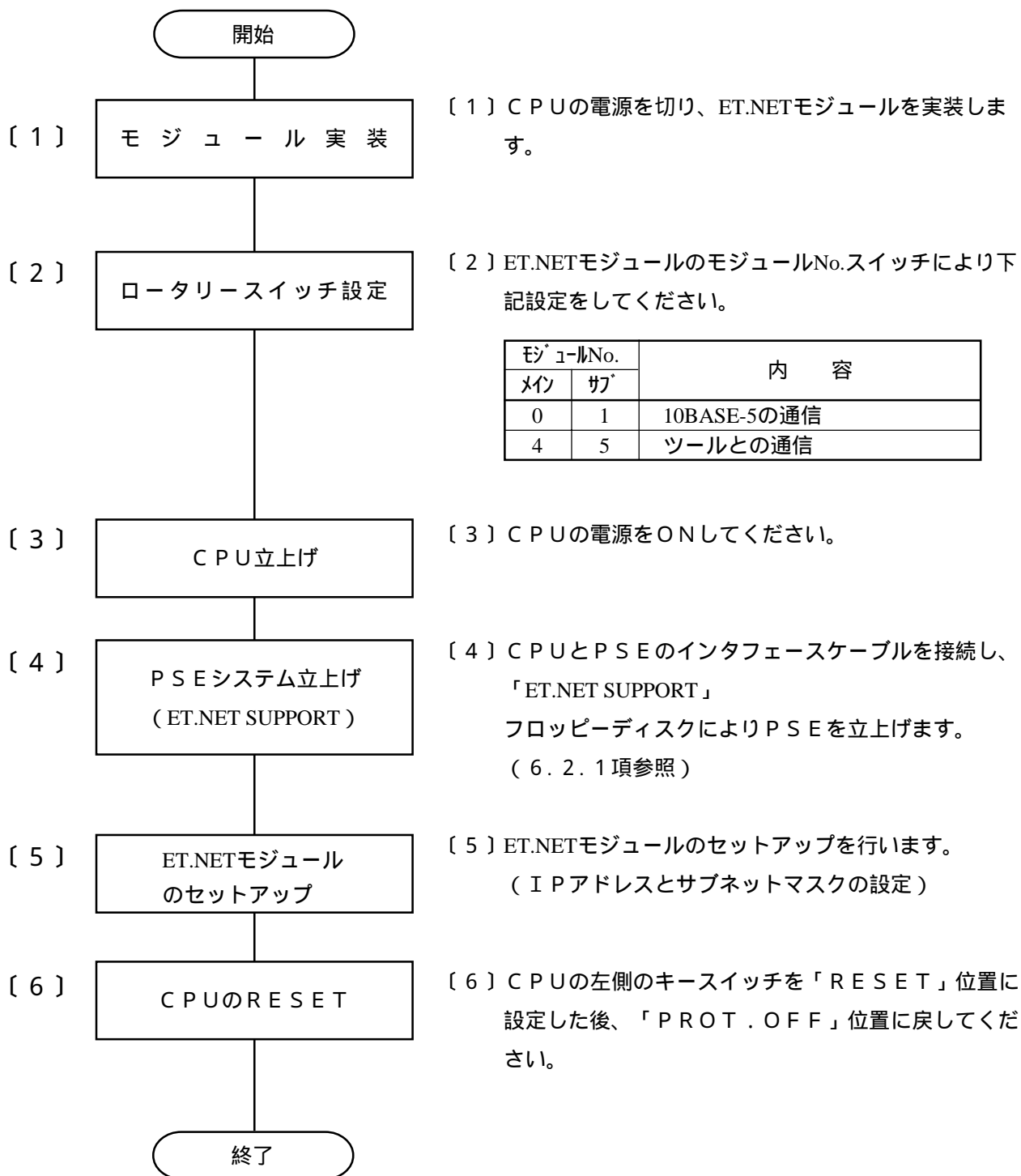
```

5 プログラム例

```
open->tll = 0; /* Time to live */
tcp_open = (short (*)())TCP_OPEN;
rtn = (tcp_open)(open); /* TCP能動的オープン */
if( rtn > 0 ){ /* リターンコード 正常? */
    break;
}
time = 100; /* 100ms Delay発行 */
delay( &time);
}
send->s_id = rtn; /* ソケットID */
recv->s_id = rtn; /* ソケットID */
while( 1 ){
    send->len = 1024; /* 送信データバイト長 */
    send->buf = ( char *)SBUFADDR; /* 送信データ先頭アドレス */
    tcp_send = ( short (*)())TCP_SEND;
    rtn = (tcp_send)(send); /* TCPデータ送信 */
    if( rtn < 0 ){ /* リターンコード 異常? */
        break;
    }
    recv->len = 1024; /* 受信バッファバイト長 */
    recv->buf = ( char *)RBUFADDR; /* 受信バッファ先頭アドレス */
    recv->tim = 60000; /* 受信待ち時間(ms) */
    tcp_receive = ( short (*)())TCP_RECEIVE;
    rtn = (tcp_receive)(recv); /* TCPデータ受信 */
    if( rtn < 0 ){ /* リターンコード 異常? */
        break;
    }
    cerr_flg = 0; /* コンパアエラーフラグクリア */
    sbuf = ( char *)SBUFADDR; /* 送信バッファ先頭アドレス */
    rbuf = ( char *)RBUFADDR; /* 受信バッファ先頭アドレス */
    for( i = 0 ; i < 1024 ; i++){
        if( sbuf[i] != rbuf[i]){
            cerr_flg = 1; /* コンパアエラーフラグセット */
            break;
        }
        sbuf[i] = ~rbuf[i]; /* 反転データセット */
    }
    if( cerr_flg == 1 ){ /* コンパアエラー? */
        break;
    }
}
close->s_id = send->s_id; /* ソケットID */
while( 1 ){
    tcp_close = ( short (*)())TCP_CLOSE;
    rtn = (tcp_close)(close); /* TCPコネクション終了 */
    if( rtn == 0 || rtn == ( short )0xFFFF ){
        break;
    } else if( rtn == ( short )0xF012 ){
        tcp_abort = ( short (*)())TCP_ABORT;
        rtn = (tcp_abort)(abort); /* TCPコネクション強制終了 */
        break;
    }
    time = 100; /* 100ms Delay発行 */
    delay( &time);
}
return;
}
```

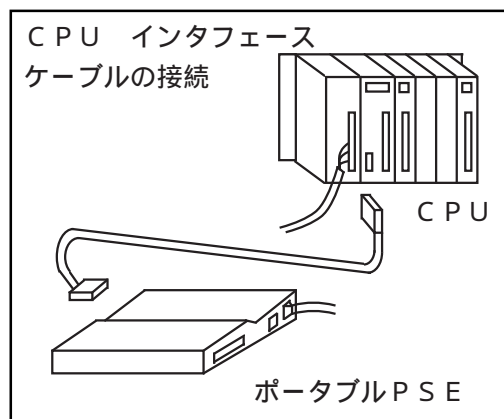
6 オペレーション

6.1 立上げ手順

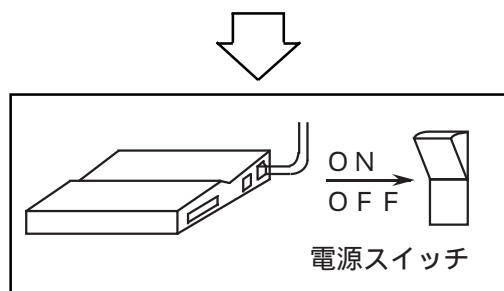


6.2 PSEシステム立上げ

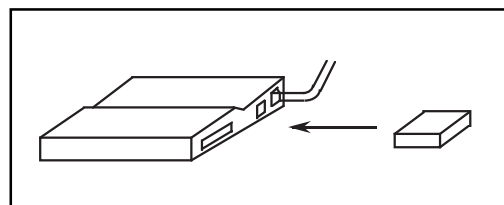
6.2.1 PSEシステム立上げ手順



〔1〕PSEの電源がOFFの状態、CPUと正しくインタフェースケーブルを接続します。このとき、CPUのコンソールスイッチはストップ（STOP）とし、メモリプロテクトスイッチはプロテクトOFF（PROT.OFF）に設定してください。



〔2〕PSEの電源をONしてください。



〔3〕システムフロッピーディスク「ET.NET SUPPORT」フロッピーディスクをPSEにセットしてください。

STRIKE ANY KEY

〔4〕PSEの画面上に、左図のメッセージを表示します。任意のキーを入力してください。

SYSTEM LOADING

〔5〕PSEは、「SYSTEM LOADING」と表示し、フロッピーディスクからシステムプログラムをローディングします。

ERROR MESSAGE KEY IN

〔6〕エラーメッセージが「日本語」か「英文」かを設定してください。

次ページへ

6 オペレーション

〔 7 〕 ET.NET SUPPORTのメニュー画面が表示されます。

初期画面

```
ET.NET SUPPORT   A PCSno=0000 Nno=000 MODE=STP Rno=04 KBD=NORM
***   ET.NET SUPPORT SYSTEM   ( Ver 1.0 Rev 0.0 )   ***
KEYIN MENU NO. =

-----
                        MENU
-----
1 : ET.NET MAIN MODULE SETUP
2 : ET.NET SUB  MODULE SETUP
-----
```

6.2.2 PSEシステム基本オペレーション

オペレーションは、画面に表示されたカーソルにそって入力することにより、簡単に操作できます。

選択する基本的なオペレーションには、次の3種類があります。

- ・ 選択項目のナンバを入力する。
- ・ 設定キーまたは修正キーを選択して押す。
- ・ 数値データを入力する。

設定キーまたは修正キーを押す場合の操作

画面に〔SET/RTY/CLS〕のように選択キーが表示される場合、それらのキーの意味は、次のようになっています。

表示画面名称	対応するキー	意味
SET	設定 キー	OKのとき
CLS	終了 キー	1つまたはそれ以上前の画面に戻る
RTY	再設定 キー	データの再設定をするとき
CNT	続行 キー	処理を繰返し行うとき
DEL	削除 キー	ファイル等の削除を行うとき

(注) CLS : CLOSE

RTY : RETRY

CNT : CONTINUE

DEL : DELETE

6 オペレーション

6.3 モジュールのセットアップ

6.3.1 機能概要

ET.NETモジュールのIPアドレス、サブネットマスク、物理アドレスの表示とIPアドレス、サブネットマスクの設定を行います。

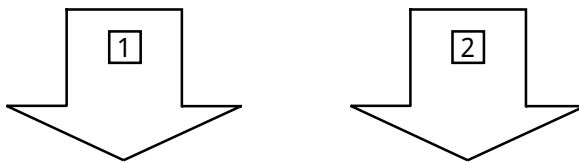
6.3.2 オペレーション

[1] モジュールセットアップメニュー画面

```
ET.NET MAIN   A PCSno=0000 Nno=000 MODE=STP Rno=04 KBD=NORM
KEYIN MENU NO. = [CLS]

*** ET.NET MAIN MODULE SETUP MENU ***
    1 : IP ADDRESS      EDITION
    2 : SUB NET MASK   EDITION
-----
          MODULE SETUP DATA
-----
IP ADDRESS      : *.*.*.*.*.*.*.*.*.*
SUB NET MASK    : *.*.*.*.*.*.*.*.*.*
PHYSICAL ADDRESS : /*****
-----
```

[1] モジュールセットアップメニュー画面より、**[1]** または、**[2]** を選択します。



[3] ~ [5] へ

[2] IPアドレス設定画面

```
KEYIN IP ADDRESS = . . . [SET/CLS/RTY]
```

[2] IPアドレスを10進数（000～255）で3桁ごとに設定します。

（例） IPアドレスを

「192.001.000.001」

とするときは **[1] [9] [2]**

[0] [0] [1] [0] [0] [0] [0]

[0] [1] [設定] と入力します。

IPアドレス設定後、自動的に下記デフォルト値がサブネットマスクへ設定されます。

IPアドレス	サブネットマスクのデフォルト値
クラスA	255.000.000.000
クラスB	255.255.000.000
クラスC	255.255.255.000

注意

IPアドレスの設定にて、ホストアドレスがオール/0または、オール/F設定の場合は、入力エラーとなります。

[3] IPアドレスがクラスAの場合のサブネットマスク設定画面

KEYIN SUB NET MASK = 255. . . . [SET/CLS/RTY]

[3] サブネットマスクを10進数（000～255）で3桁ごとに設定します。

(例) サブネットマスクを「255.255.255.000」とするときは

[設定] と入力します。

[4] IPアドレスがクラスBの場合のサブネットマスク設定画面

KEYIN SUB NET MASK = 255.255. . . [SET/CLS/RTY]

[4] サブネットマスクを10進数（000～255）で3桁ごとに設定します。

(例) サブネットマスクを「255.255.255.000」とするときは
 [設定] と入力します。

[5] IPアドレスがクラスCの場合のサブネットマスク設定画面

KEYIN SUB NET MASK = 255.255.255. . [SET/CLS/RTY]

[5] サブネットマスクを10進数（000～255）で3桁設定します。

(例) サブネットマスクを「255.255.255.240」とするときは
[設定] と入力します。



注 意

ET.NETモジュールを未実装状態でPSE のセットアップメニュー画面に切替えると物理アドレス表示は/FFFFFFFFFとなります。物理アドレスを参照する場合は、ET.NETモジュールを実装してください。なお、IPアドレスおよびサブネットマスクはET.NETモジュール未実装時でも設定、参照できます。

IPアドレスが未設定、またはOSロード時のメモリクリアなどによりクリアされた場合は、ET.NETモジュールのERR LEDが点灯しCPUインディケータに下記を表示し通信が停止します。

メインモジュールのIPアドレスが未設定時：“ETM IPNG”

サブモジュールのIPアドレスが未設定時：“ETS IPNG”

7 保 守

7 保 守

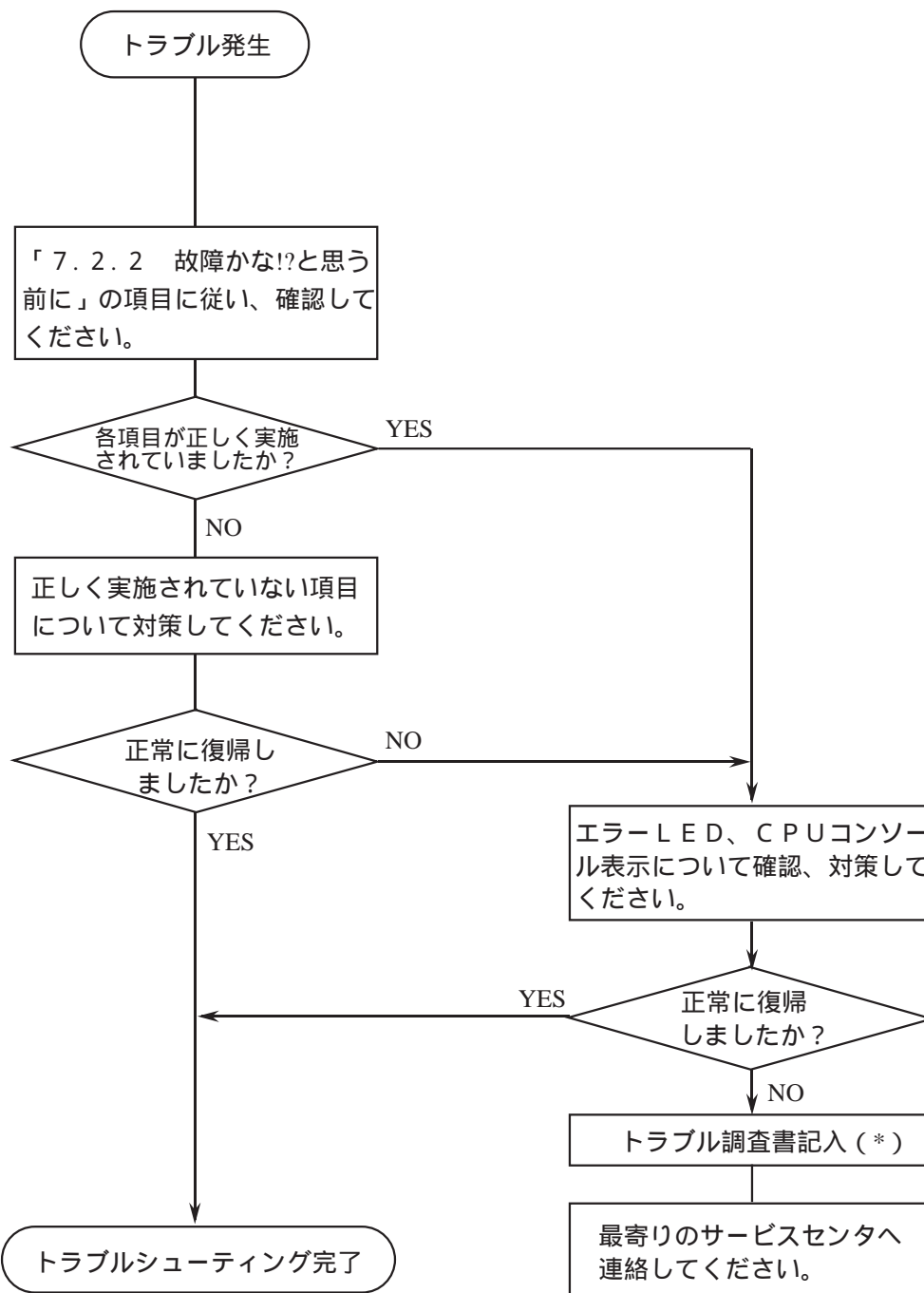
7.1 保守点検

7.1.1 定期点検

項目	点 検 内 容	頻度
ユニット清掃	電源をすべてOFFし、ET.NETモジュールのケースのすきまから、真空掃除器でほこりをたてないように清掃してください。	1回/年
機構チェック	ET.NETモジュールの取付けネジ、TB取付けネジ、通信ケーブル取付けネジのゆるみ、損傷の有無を点検してください。 ゆるみのあるものは締付けを行ってください。損傷箇所は交換してください。	1回/年

7.2 トラブルシューティング

7.2.1 手 順



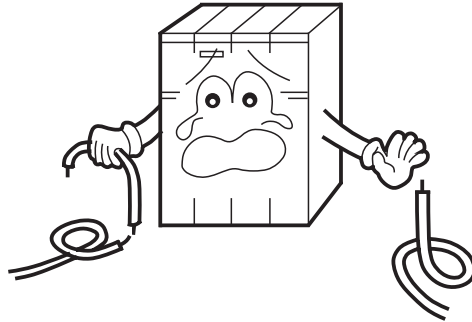
(*) 「8.17 トラブル調査書」を利用してください。

7 保 守

7.2.2 故障かな!?と思う前に

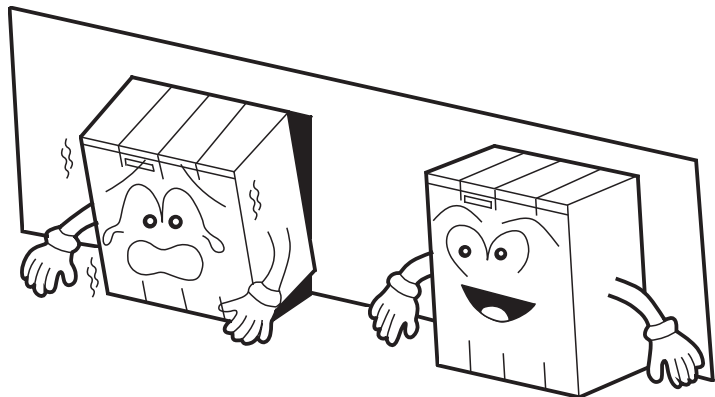
配線は正常ですか？

- ・ケーブルの断線、接続誤りがないか調べてください。
- ・トランシーバケーブルはシールドアース線付きのケーブルを使用しているか調べてください。



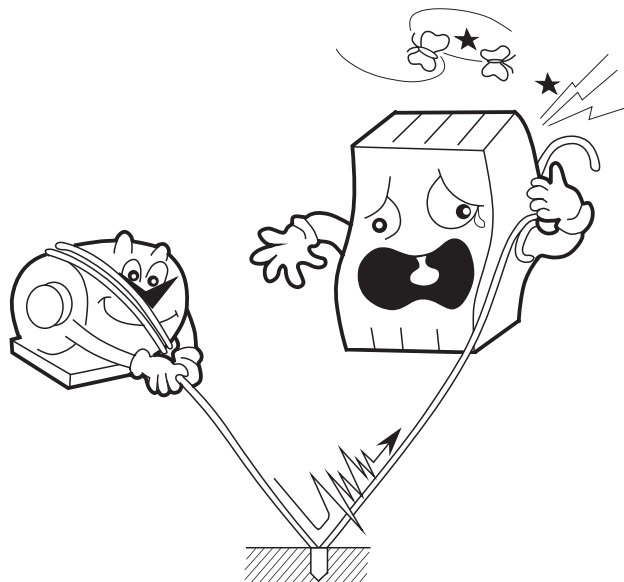
モジュールは正しく実装されていますか？

- ・ET.NETモジュールの実装位置は、奇数スロットに左詰めで実装されているか調べてください。
- ・取付けネジのゆるみがないか調べてください。



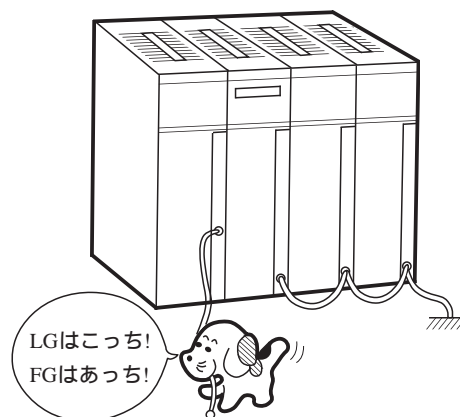
正しく接地されていますか？

- ・ 強電機器と同一点での接地は避け、分離してください。
- ・ D種接地以上の接地工事を行ってください。



L GとF Gは分離されていますか？

- ・ 電源からのノイズがL Gを介してF Gへ入り込み、誤動作の原因となるため、必ず分離してください。
- ・ L Gは電源供給側で接地してください。



7.3 エラーと対策

7.3.1 PSEエラーコード表

ET.NET SUPPORTシステム（システムF / D）におけるPSEエラーコード表を示します。

エラーコード (HEX)	内容および原因	対 策
03	PSEとCPU間が正しく接続されていません。	PSEとCPU間を正しく接続してください。
15	オプションモジュールが実装されていません。	オプションモジュールを実装してください。
81	入力が範囲外か不正データ	確認の上、再入力してください。
AA	<u>PSEシステムタイプ不一致エラー</u> 使用しているPSEシステムとPCsの機種が一致しない。	対象となるPCs用のPSEシステムフロッピーディスクを使用してください。
E9	プリンタと正常に交信できない。	PSEとプリンタのケーブルは正しく接続されているか、プリンタの電源はONかチェックしてください。

7.3.2 CPU LED表示メッセージ表

CPU LED表示は、下記表に示すようにメイン、サブモジュールで区別します。

モジュール	表示内容	内容および説明	対 策
メイン	ETM @. @	ET.NETモジュール(メイン)が正常に立ち上がった。	エラーではありません。
	ETM	ET.NETモジュール(メイン)のボードでハードウェアエラーを検出。	「7.3.3 ハードウェアエラー」を参照してください。
	EXD2 PTY	ET.NETモジュール(メイン)のメモリをCPUが読込んだとき、パリティエラーが発生。	CPUキースイッチを一度リセットし、元に戻しても表示が消えない場合、ET.NETモジュールを交換してください。
サブ	ETS @. @	ET.NETモジュール(サブ)が正常に立ち上がった。	エラーではありません。
	ETS	ET.NETモジュール(サブ)のボードでハードウェアエラーを検出。	「7.3.3 ハードウェアエラー」を参照してください。
	EXD3 PTY	ET.NETモジュール(サブ)のメモリをCPUが読込んだとき、パリティエラーが発生。	CPUキースイッチを一度リセットし、元に戻しても表示が消えない場合、ET.NETモジュールを交換してください。

- ・@. @は、ET.NETモジュールのバージョン、レビジョンを表します。
- ・ は、「7.3.3 ハードウェアエラー」のエラー表示データを表します。

7 保 守

7.3.3 ハードウェアエラー

ET.NETモジュールがハードウェアエラーを検出した場合は、CPU LEDに下表のエラーメッセージを表示します。また、エラ - LEDを点灯し、エラーフリーズ情報の収集を行います。

ET.NETモジュールの動作は停止します。

表示 メッセージ	エラー内容	対 策
BUS	バスエラー	ET.NETモジュールが故障している可能性があります。モジュールを交換してください。
ADDR	アドレスエラー	
ILLG	不当命令	
ZERO	ゼロ除算	
PRIV	特権違反	
FMT	フォーマットエラー	
SINT	スプリアス割込み	
EXCP	未使用例外	
PTY	パリティエラー	
MDSW	モジュールスイッチ設定ミス	モジュールスイッチ設定を確認してください。
ROM1	ROM1サムエラー	ET.NETモジュールが故障している可能性があります。モジュールを交換してください。
RAM1	RAM1コンペアエラー	
RAM2	RAM2コンペアエラー	
ROM3	ROM3サムエラー	
IPNG	IPアドレス未登録	IPアドレスを登録してください。
MAC	MACアドレス未登録	ET.NETモジュールが故障している可能性があります。モジュールを交換してください。
PRG	マイクロプログラムエラー	い。

ET.NETモジュールがハードウェアエラーを検出した場合は、エラーLEDを点灯しエラーフリーズ情報の登録を行います。ET.NETモジュールの動作は停止します。

メインモジュール	サブモジュール	2 ³¹ ——— 2 ¹⁶ 2 ¹⁵ ——— 2 ⁰
/840400	/8C0400	エラーコード
/840404	/8C0404	——
/840410	/8C0410	D0レジスタ
/840414	/8C0414	D1レジスタ
/840418	/8C0418	D2レジスタ
/84041C	/8C041C	D3レジスタ
/840420	/8C0420	D4レジスタ
/840424	/8C0424	D5レジスタ
/840428	/8C0428	D6レジスタ
/84042C	/8C042C	D7レジスタ
/840430	/8C0430	A0レジスタ
/840434	/8C0434	A1レジスタ
/840438	/8C0438	A2レジスタ
/84043C	/8C043C	A3レジスタ
/840440	/8C0440	A4レジスタ
/840444	/8C0444	A5レジスタ
/840448	/8C0448	A6レジスタ
/84044C	/8C044C	A7レジスタ
/840450	/8C0450	スタックフレーム (4ワード, 6ワード, バスエラー)
/8404FC	/8C04FC	

No.	コード	内 容
1	0010H	バスエラー
2	0011H	アドレスエラー
3	0012H	不当命令
4	0013H	ゼロ除算
5	0014H	特権違反
6	0016H	フォーマットエラー
7	0017H	スプリアス割込み
8	0018H	未サポート例外 (CHK, TRAPV, L1010など)
9	0019H	パリティエラー
10	001AH	停電予告
11	0100H	モジュール・スイッチの設定ミス
12	0102H	ROM1のサムエラー
13	0103H	RAM1のコンペアエラー
14	0105H	RAM2のコンペアエラー
15	010BH	ROM3のサムエラー
16	0113H	I Pアドレス未登録
17	0114H	M A Cアドレスエラー

(注) スタックフレームについては、次ページに詳細を示します。

エラーリーク情報テーブル内スタックフレームの詳細を以下にします。

メイン レジスタ	サブ レジスタ	フォーマット \$0 (4ワードスタックフレーム)	フォーマット \$2 (6ワードスタックフレーム)	フォーマット \$C (プリフェッチおよびオペランドの バスエラースタック)	フォーマット \$C (MOVEMオペランドの バスエラースタック)	フォーマット \$C (4ワードおよび6ワード バスエラースタック)
		2 ¹⁵ ————— 2 ⁰	2 ¹⁵ ————— 2 ⁰	2 ¹⁵ ————— 2 ⁰	2 ¹⁵ ————— 2 ⁰	2 ¹⁵ ————— 2 ⁰
/840450	/8C0450	ステータスレジスタ	ステータスレジスタ	ステータスレジスタ	ステータスレジスタ	ステータスレジスタ
/840452	/8C0452	プログラム カウンタ	次命令プログラム カウンタ	リターンプログラム カウンタ	リターンプログラム カウンタ	次命令プログラム カウンタ
/840454	/8C0454					
/840456	/8C0456	/0 ベクタオフセット	/2 ベクタオフセット	/C ベクタオフセット	/C ベクタオフセット	/C ベクタオフセット
/840458	/8C0458		フォールトを起こした 命令のプログラムカウンタ	フォールトを起こした アドレス	フォールトを起こした アドレス	フォールトを起こした アドレス
/84045A	/8C045A					
/84045C	/8C045C			DBUF	DBUF	例外発生前のステータスレジスタ
/84045E	/8C045E					フォールトを起こしたベクタオフセット
/840460	/8C0460			現在命令 プログラムカウンタ	現在命令 プログラムカウンタ	フォールトを起こした命令の プログラムカウンタ
/840462	/8C0462					
/840464	/8C0464			内部転送カウンタレジスタ	内部転送カウンタレジスタ	内部転送カウンタレジスタ
/840466	/8C0466			0 0 特殊ステータスワード	0 1 特殊ステータスワード	1 0 特殊ステータスワード

7.3.4 ソケットハンドラ検出のエラーコード表

ソケットハンドラのエラーコードと対策について、以下に示します。

エラーコード	内 容	原 因	対 策
0xF000	コネクション未接続	ハンドラ起動時、未接続またはポート解放されました。	tcp_open、またはtcp_popenを発行しコネクション確立後にハンドラを再発行してください。
0xF002	FIN受信	ハンドラ起動時、FINを受信しました。	tcp_closeを発行しコネクション切断後tcp_open、またはtcp_popenから再コネクションしてください。
0xF010	ソケットID不正	<ul style="list-style-type: none"> ソケットIDが範囲外(TCP:1 ID 15, UDP:0×20 ID 0×27) 使用していないソケットID、または解放済みのソケットIDを指定しました。 未接続、または接続が確立していません。(tcp_acceptのみ) 	ユーザプログラム(ソケットIDにtcp_open、またはtcp_popenのリターン値を指定しているかなど)を見直してください。
0xF011	ソケット数オーバ	ソケットを制限数以上登録しています。(TCP:12個, UDP:8個)	未使用ソケットをクローズ後(tcp_close/udp_close) tcp_open、またはtcp_popenから再コネクションしてください。
0xF012	ソケットドライバタイムアウト	一定時間経過してもソケットドライバから応答がありません。	tcp_closeを発行しコネクション切断後tcp_open、またはtcp_popenから再コネクションしてください。再コネクションを繰り返しても通信が復旧しない場合は、コネクタ、ケーブル、相手局に異常がないか確認してください。 tcp_closeにて発生した場合は、tcp_abortを発行し、コネクションを切断後、tcp_openまたは、tcp_popenから再コネクションをしてください。
0xF013	モジュール停止	ハンドラ起動時、100秒経過してもソケットドライバの初期化が終了できない場合	アプリケーションの許容範囲内でtcp_closeを発行後、tcp_openまたは、tcp_popenから再コネクションしてください。
0xF020	送信データ長不正	送信データ長が制限値を満足していません。(TCP:1 データ長 4096, UDP:1 データ長 1472)	ユーザプログラム(送信データ長の指定値)を見直してください。
0xF021	受信バッファ長不正	受信データ長が制限値を満足していません。(1 データ長 4096)	ユーザプログラム(受信データ長の指定値)を見直してください。
0xF0FF	ポート解放	<ul style="list-style-type: none"> ハンドラ起動後、ポート解放状態(RST受信)になりました。(tcp_open) ハンドラ起動時、ポート解放状態でした。(tcp_send/tcp_receive) 	<ul style="list-style-type: none"> tcp_open、またはtcp_popenから再コネクションしてください。 tcp_closeを発行しコネクション切断後tcp_open、またはtcp_popenから再コネクションしてください。

7 保 守

エラーコード	内 容	原 因	対 策
0xFFFF0	アドレス不正	<ul style="list-style-type: none"> udp_open, udp_sendともに相手局のIPアドレス, ポート番号に0を設定していません。 udp_sendでイーサネットレベルのエラー（コリジョンなど）が発生しました。 	<ul style="list-style-type: none"> ユーザプログラムを見直してください。 トラフィックが下がった時点でudp_sendをリトライしてください。
0xFFFF3	引数不正	不正なパラメータを指定しました。	ユーザプログラムを見直してください。
0xFFFF5	接続タイムアウト	相手局からの応答がありません。	tcp_closeを発行し接続切断後tcp_open、またはtcp_popenから再接続してください。再接続を繰り返しても通信が復旧しない場合は、コネクタ, ケーブル, 相手局に異常がないか確認してください。
0xFFFF6	知らず済み	接続が終了した（closeまたはabortされた）ソケットIDに対し、コマンドが発行されました。	tcp_open、またはtcp_popenから再接続してください。
0xFFFF8	FIN受信	相手局からFINを受信しました。	tcp_closeを発行しソケットをクローズしてください。
0xFFFFA	接続強制終了	相手局から強制終了（RST受信）されました。（RST受信後にtcp_receiveを発行した。）	tcp_closeを発行し接続切断後tcp_open、またはtcp_popenから再接続してください。
0xFFFFC	ソケットハンドル不正	TCP/UDPでオープンしていないハンドル番号を使用して送受信を行おうとしました。RST受信で発生する可能性があります。（tcp_receiveで受信待ちのときRST受信しました。）	tcp_closeを発行しソケットをクローズ後tcp_open、またはtcp_popenから再接続してください。
0xFFFFD	2重ソケットエラー	同じソケット（相手局のIPアドレス, 相手局ポート番号, 自局ポート番号）がすでに存在しています。	ユーザプログラムを見直してください。
0xFFFFE	コントロールロック不正	制限を超えてソケットを使用しています。	未使用ソケットをクローズ後(tcp_close/udp_close)tcp_open、またはtcp_popenから再接続してください。

8 付 録

8.1 ネットワーク構成部品

8.1.1 LWE550とイーサネット*との接続の問題点

LWE550は、国際標準であるIEEE802.3規格に準拠している標準仕様品です。

しかし同一規格に準拠した異社間のトランシーバおよびリピータなどを組合せた場合、相性によって正常に動作しない場合があります。

したがって、LWE550では、トランシーバ、リピータ、同軸ケーブル、コネクタおよびターミネータはすべて当社の推奨する機器を使用してください。

イーサネットの使用にはIEEE802.3規格とオリジナルイーサネット仕様とがあります。

LWE550にオリジナルイーサネット仕様の機器を接続することはできませんので注意してください。

*イーサネットは米国ゼロックス社の登録商標です。

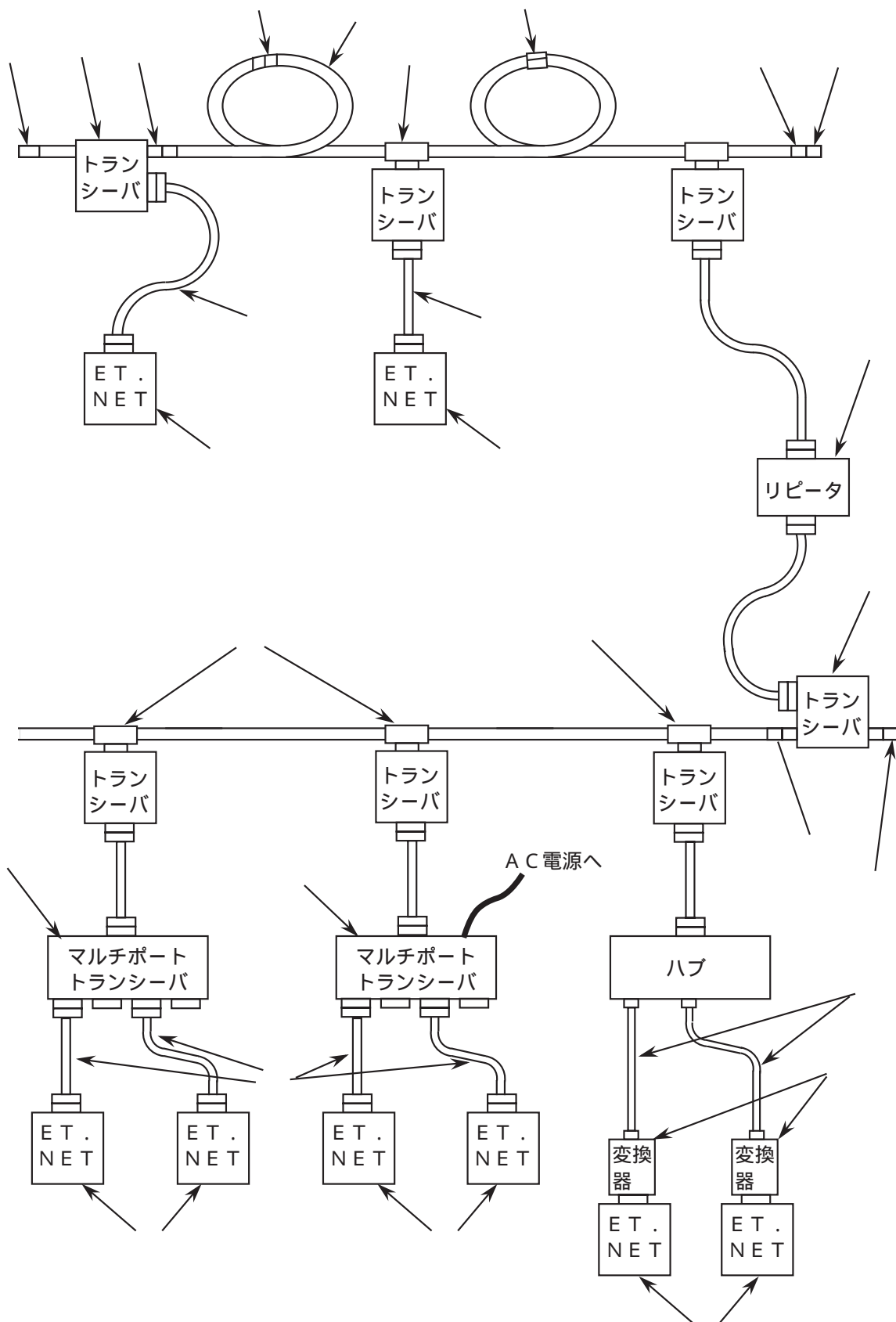
8.1.2 構成品一覧表

No.	品名	メーカー	型式	備考
	ET.NET	(株)日立製作所	LWE550	H-S10/2 に実装されるIEEE802.3準拠LAN制御装置
	トランシーバ	日立電線(株)	HLT-200TB HBN200TZ HBN200TD	タップ形トランシーバ
	トランシーバ	日立電線(株)	HLT-200	コネクタ形トランシーバ
	リピータ	日立電線(株)	HLR-200H	同軸ケーブルの伝送距離延長用リピータ装置
	マルチポートトランシーバ	日立電線(株)	H-7612-64 H-7612-68	4ポート/8ポートトランシーバ AC電源内蔵
	同軸ケーブル	日立電線(株)	HBN-CX-100	屋内用、ケーブル長指定(最長500m)
	同軸コネクタ	日立電線(株)	HBN-N-PC	同軸ケーブル用コネクタ
	中継コネクタ	日立電線(株)	HBN-N-AJJ	同軸ケーブル用中継コネクタ
	ターミネータ	日立電線(株)	HBN-T-NJ	J形
	ターミネータ	日立電線(株)	HBN-T-NP	P形
	アース端子	日立電線(株)	HBN-G-TM	同軸ケーブル用アース端子
	トランシーバケーブル	(株)日立製作所	HDC4360	オス、メスD-sub15ピンコネクタ付 最長15m
	変換器	日立電線(株)	HSN-9010	10BASE-5/T変換器
	ツイストペアケーブル	日立電線(株)	HUTP-CAT5 4P	ツイストペアケーブル
	マルチポートトランシーバ	日立電線(株)	HBM-400TZ	4ポートトランシーバ

上表のNo. ~ の寸法などは参考として「8.1.3 トランシーバ(タップ形)」~「8.1.14 変換器」に記載していますが、メーカー都合により変更となる場合がありますので、詳細は必ずメーカーに問い合せてください。

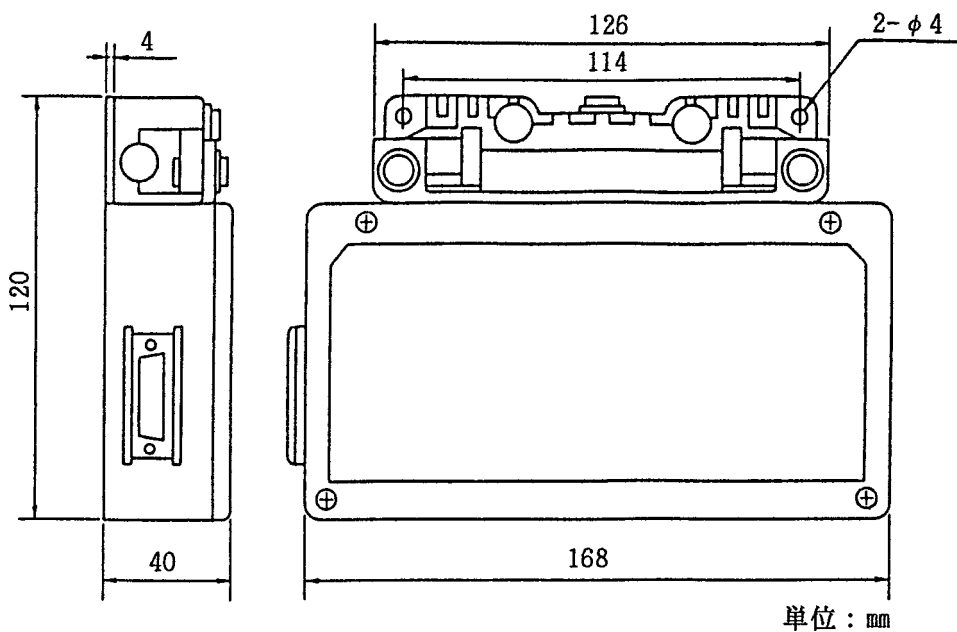
強制

10BASE-5/T変換器は、必ず弊社指定の変換器(型式:HSN-9010,メーカー:日立電線(株))を使用してください。他の変換器を使用するとノイズなどの影響により極端に性能が落ちる可能性があります。



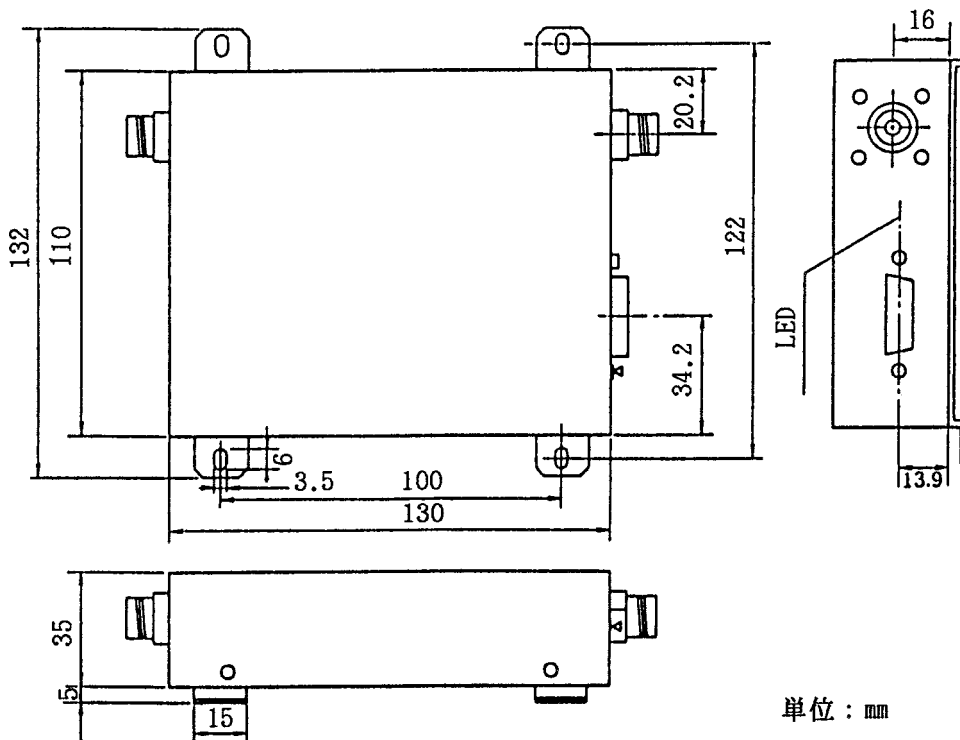
8. 1. 3 トランシーバ (タップ形) HLT - 200TB

トランシーバ (タップ形) の外観を示します。



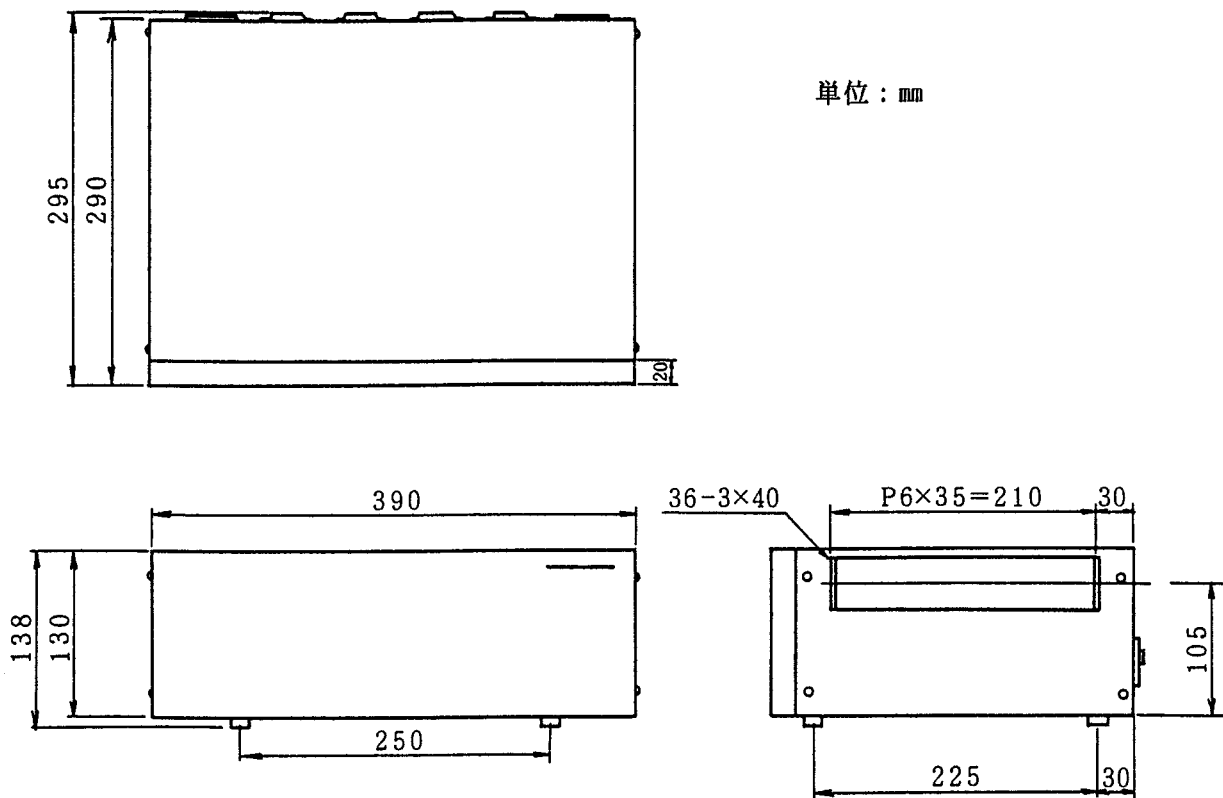
8. 1. 4 トランシーバ (コネクタ形) HLT - 200

トランシーバ (コネクタ形) の外観を示します。



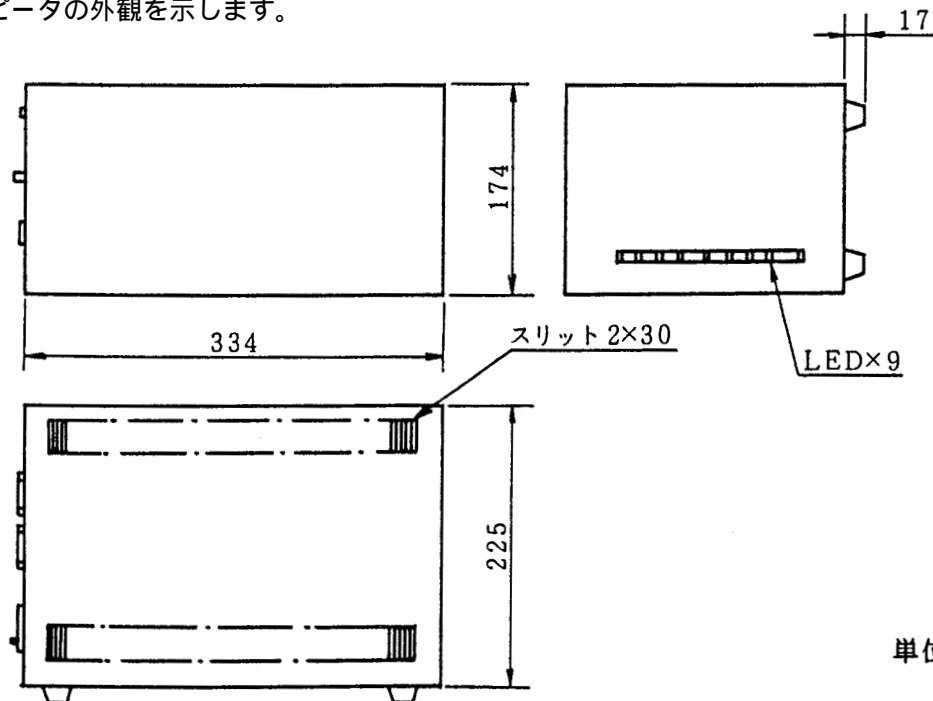
8. 1. 5 マルチポートトランシーバ H - 7 6 1 2 - 6 4

マルチポートトランシーバの外観を示します。



8. 1. 6 リピータ HLR - 2 0 0 H

リピータの外観を示します。



8. 1. 7 同軸ケーブル(屋内用) HBN-CX-100

同軸セグメント用ケーブルの構造および特性を以下に示します。

表 8 - 1 同軸ケーブルの構造

線 心 種 類		50 高周波発泡PE同軸ケーブル
錫メッキ軟銅線導体	断面積	3.7mm ² (12AWG)
	構成	1本 / 2.17mm
	外径	2.17mm
絶縁体	材質	発泡PE
	厚さ	2.015mm
	外径	6.2mm
	色別	自然色
アルミ箔テープ巻	外径	6.35mm 1
シールド	構成	0.18Tmm × 7持 × 24打
	外径	7.25mm
アルミ箔テープ巻	外径	7.4mm 1
シールド	構成	0.18Tmm × 7持 × 24打
	外径	8.3mm
PVCシース	厚さ	1.0mm
	外径	10.0 ~ 10.8mm
	色別	黄

1 アルミ箔テープは、縦添えとします。 T : スズメッキ軟銅線

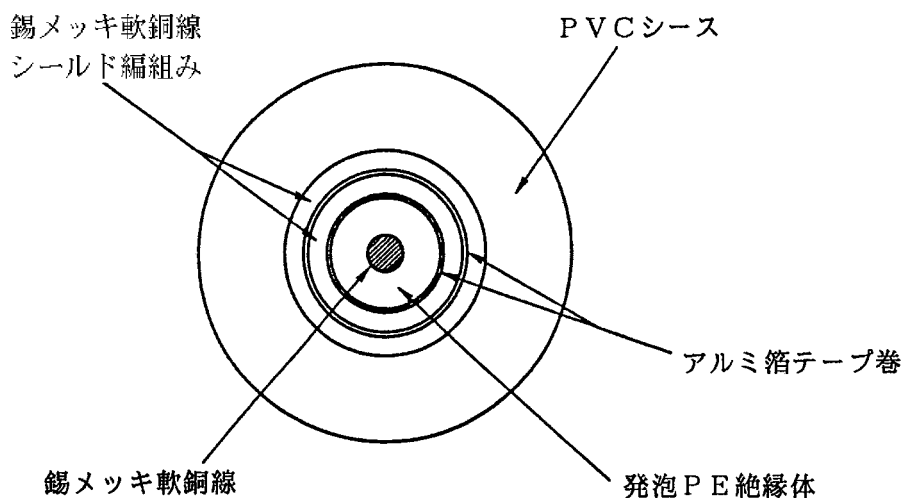


図 8 - 1 同軸ケーブルの構造

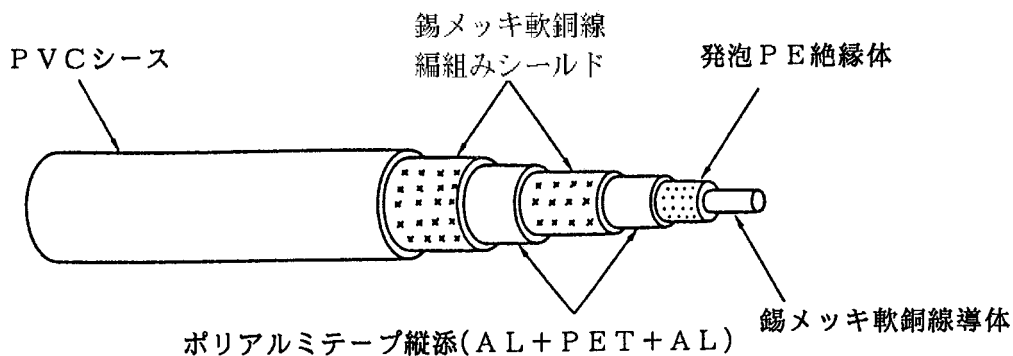


図 8 - 2 同軸ケーブルの構造外観

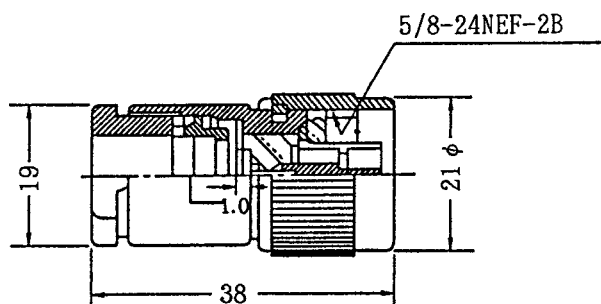
表 8 - 2 電気的特性

項目		特性値
特性インピーダンス		50 ± 2
伝播速度		0.77C* 以上
伝播インピーダンス		IEEE 802.3仕様に適合
減衰量	5 MHz	6.0 dB / 500m以下
	10 MHz	8.5 dB / 500m以下

* Cは光速を示します。

8.1.8 同軸コネクタ HBN - N - PC

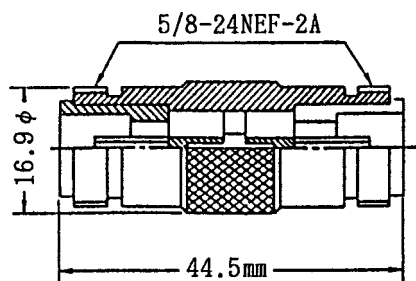
同軸ケーブル用のコネクタの外観を示します。



単位：mm

8. 1. 9 中継コネクタ HBN - N - A J J

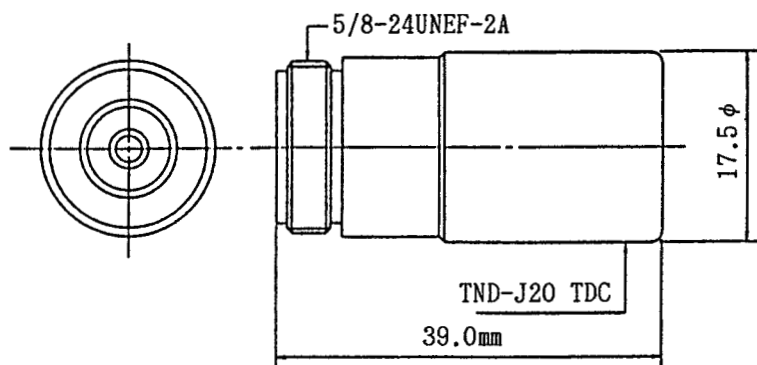
同軸ケーブル用の中継コネクタの外観を示します。



8. 1. 10 ターミネータ (J 形) HBN - T - N J

同軸セグメントの端末を終端するためのターミネータ (J 形) の外観を示します。

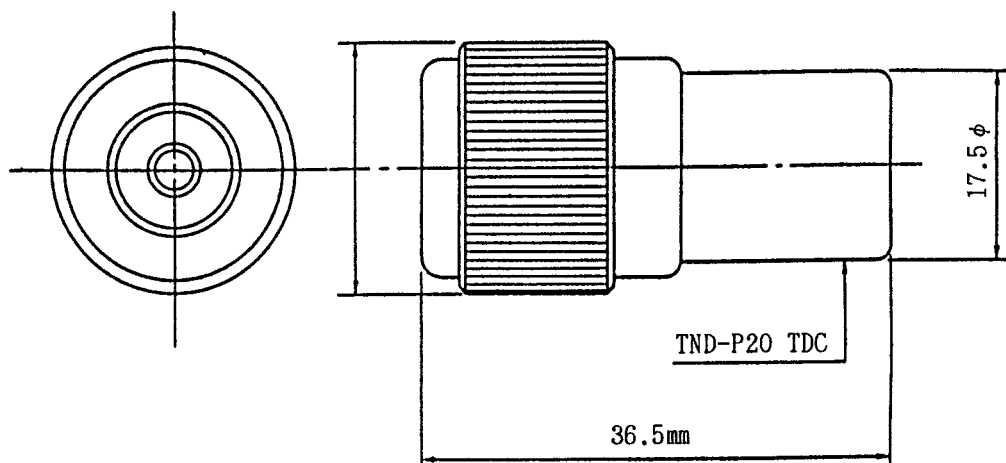
(同軸ケーブルの終端に使用します。)



8. 1. 11 ターミネータ (P 形) HBN - T - N P

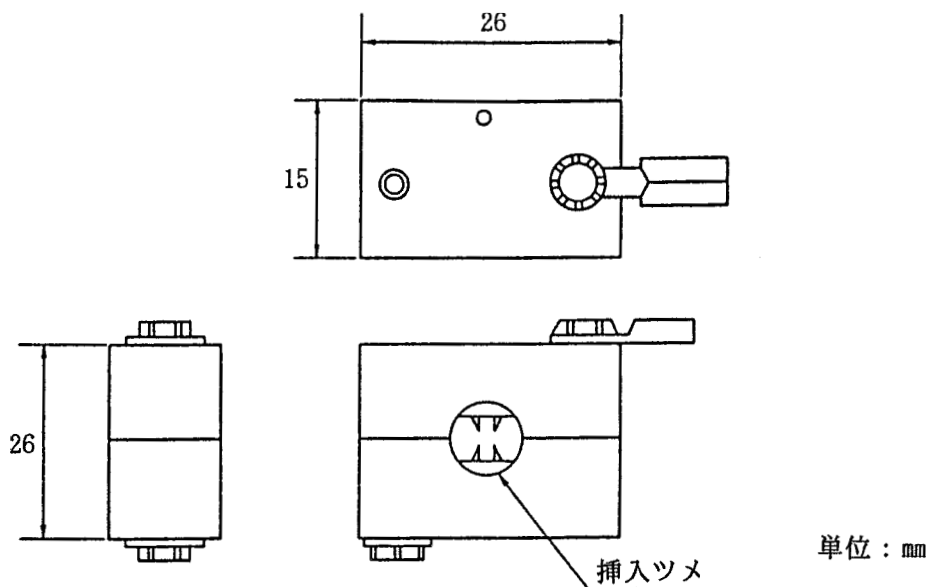
同軸セグメントの端末を終端するためのターミネータ (P 形) の外観を示します。

(トランシーバ コネクタ形の終端に使用します。)



8. 1. 12 アース端子 HBN - G - TM

同軸セグメントを接地するためのアース端子の外観を示します。



8. 1. 13 トランシーバケーブル HDC 4 3 6 0

トランシーバとコントローラおよびトランシーバとリピータを接続するトランシーバケーブルの外観およびピン配置を示します。

表 8 - 3 トランシーバケーブルの構造

線 心 種 類	信 号 線	電 源 線
線 心 番 号	1 ~ 6	7 ~ 8
錫メッキ軟銅線導体	断面積	22 AWG
	構成	7本 / 0.26mm
	外径	0.78mm
絶縁体	材質	架橋発泡 PE 半硬質 PVC
	厚さ	0.41mm
	外径	1.6mm
撚り	外径	3.2mm
シールド	構成	ALテープ巻
	外径	3.4mm
撚り合わせ	外径	6.6
錫メッキ軟銅線シールド	構成	0.14 T mm × 9 持 × 24 打
	外径	7.3mm
PVCシース	厚さ	1.0mm
	外径	9.3mm ± 0.5mm
	色別	灰

T：スズメッキ軟銅線

コネクタ

標準長トランシーバケーブルのコネクタ仕様を下図に示します。

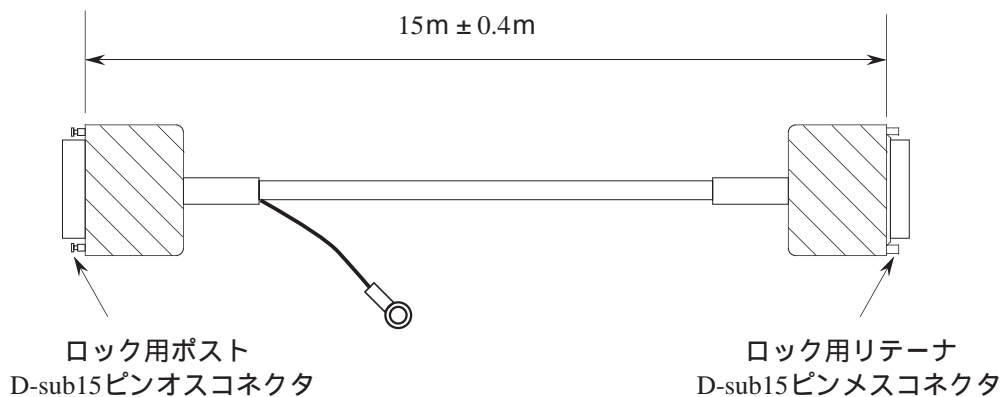


図8 - 3 トランシーバケーブルのコネクタ仕様

ケーブル断面図

ケーブル断面図を下図に示します。

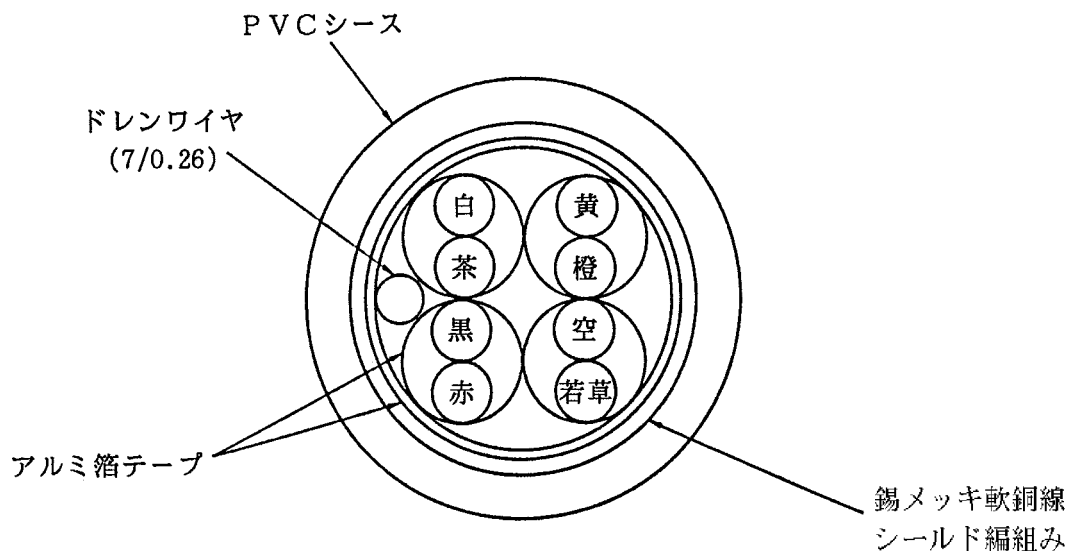


図8 - 4 ケーブル断面図

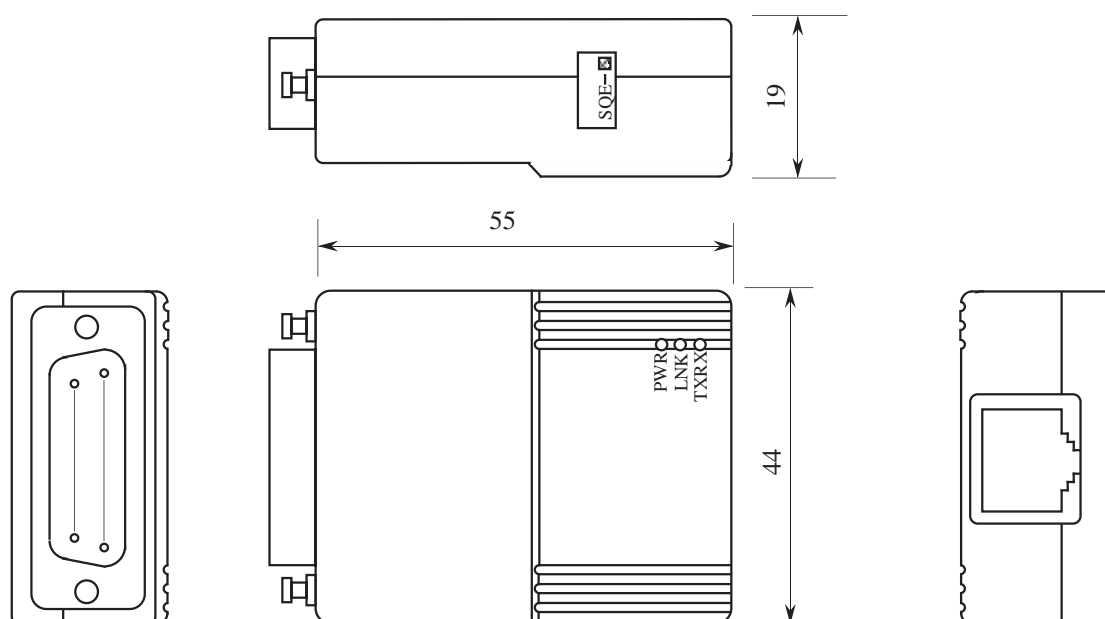
コネクタのピン配置を以下に示します。

表 8 - 4 トランシーバケーブルのピン配置

ピンNo.	信号名	心線名	ピンNo.	信号名	心線名
1	シールド	(ドレインワイヤ)			
2	衝突検出 +	茶	9	衝突検出 -	白
3	送信 +	空	10	送信 -	若草
4	——	——	11	——	——
5	受信 +	橙	12	受信 -	黄
6	電源リターン	黒	13	電源	赤
7	——	——	14	——	——
8	——	——	15	——	——

8. 1. 14 変換器 HSN - 9010

変換器の外観図を示します。



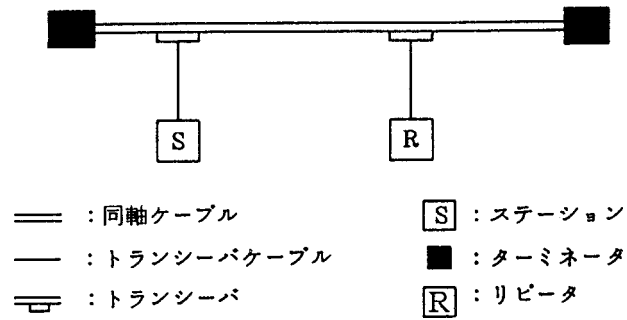
強制

ET.NETに取付けの際は、SQEスイッチの設定は“ON”にしてください。



8.2 施工分担

下図の施工は、すべてユーザが実施してください。



8.3 同軸ケーブルの配線

同軸ケーブルは、屋内の配線ダクトに布設、配線を行い、100V以上の配線とは区別してください。
また、ケーブル布設前には必ず短絡、断線がないかどうかチェックしてください。

8.3.1 ケーブルセグメントの布設

(1) ケーブルの布設配線方法は、配線される場所によりいろいろなケーブルの取付方法が考えられます。

その主なものは以下のとおりです。

- ・天井内コロガシ配線
- ・ケーブルラック内配線
- ・壁面露出配線
- ・フリーアクセス、床ビット内配線
- ・電線管内配線

(2) 布設配線工事上の留意事項は以下のとおりです。

- ・このケーブルは、原則として屋内に布設、配線してください。
- ・ケーブルの重量は、約1.9kg / 10mです。
- ・ケーブルの布設中、ケーブル本体に25kg・f以上の張力を加えないでください。
- ・ケーブルの曲げ半径は布設時、最終固定時共に250R（やむを得ないとき150R）以上としてください。
- ・壁面、天井などへの固定はサドルを用いて行い、特殊な場合を除き固定間隔は1mを標準とします。
その際、サドルの締付けなどによりケーブルが変形しないようにしてください。
- ・ケーブルラックにケーブルを固定する場合の固定間隔は2mを標準とします。
- ・管路内配線の際に使用する電線管は、防火壁貫通部に使用される場合などを除き、通常の配管の場合は、内径22mm以上の管路を使用してください。
- ・使用する電線管の曲げ半径は、300mm以上としてください。
- ・床上または床際にケーブルを配線する場合は、歩行または器物によりケーブルに変形、損失を受けやすいので結びなどにより保護を行ってください。
- ・ケーブルの外部導体は保安上、接地を行ってください。接地を行う場合は、1セグメントの1点で接地を行いD種接地以上としてください。接地点以外のケーブルの金属の露出部分が大地や他の金属部分に接触しないようコネクタ、ターミネータは付属のブーツを被せるか、絶縁テープを巻き絶縁してください。

8.4 トランシーバ（コネクタ形）の設置・取付け

(1) トランシーバの設置場所および取付け方法は、現場の状況によりいろいろ考えられます。主な設置場所は次のような所が考えられます。

- ・壁面に設置
- ・ステーションのそばに設置

図8 - 5 から図8 - 10に設置例を示します。

(2) トランシーバを取付ける上での留意事項は以下のとおりです。

- ・トランシーバを取付け金具を介して木ネジなどで固定してください。
- ・トランシーバの取付け間隔は、2.5m以上としてください。

(3) トランシーバの取付け

同軸ケーブル側のコネクタは、8.1.8項に示したものを使用します。トランシーバはケーブルに無理な力がかからないよう4箇所のネジ穴で固定してください。トランシーバ寸法図は8.1.4項を参照してください。同軸ケーブルの外部導体はアース電位から浮いていますので、同軸コネクタは他の金属に触れないようゴムブーツまたはビニールテープなどで絶縁してください（トランシーバ本体のケースはトランシーバケーブルの接続によってアース電位に保たれていますが、多点アースとならないようトランシーバ本体のケースは取付け時に絶縁してください）。

また、コネクタの同軸ケーブルへの取付けは、「8.6 同軸コネクタの取付け」を参照してください。

(4) 設置場所を選択する際には、下記事項を厳守してください。

- ・コネクタ・ターミネータのゆるみを確認できる。
- ・トランシーバケーブルコネクタのゆるみを確認できる。
- ・付帯のLEDを確認できる。

トランシーバ、トランシーバケーブル設置例

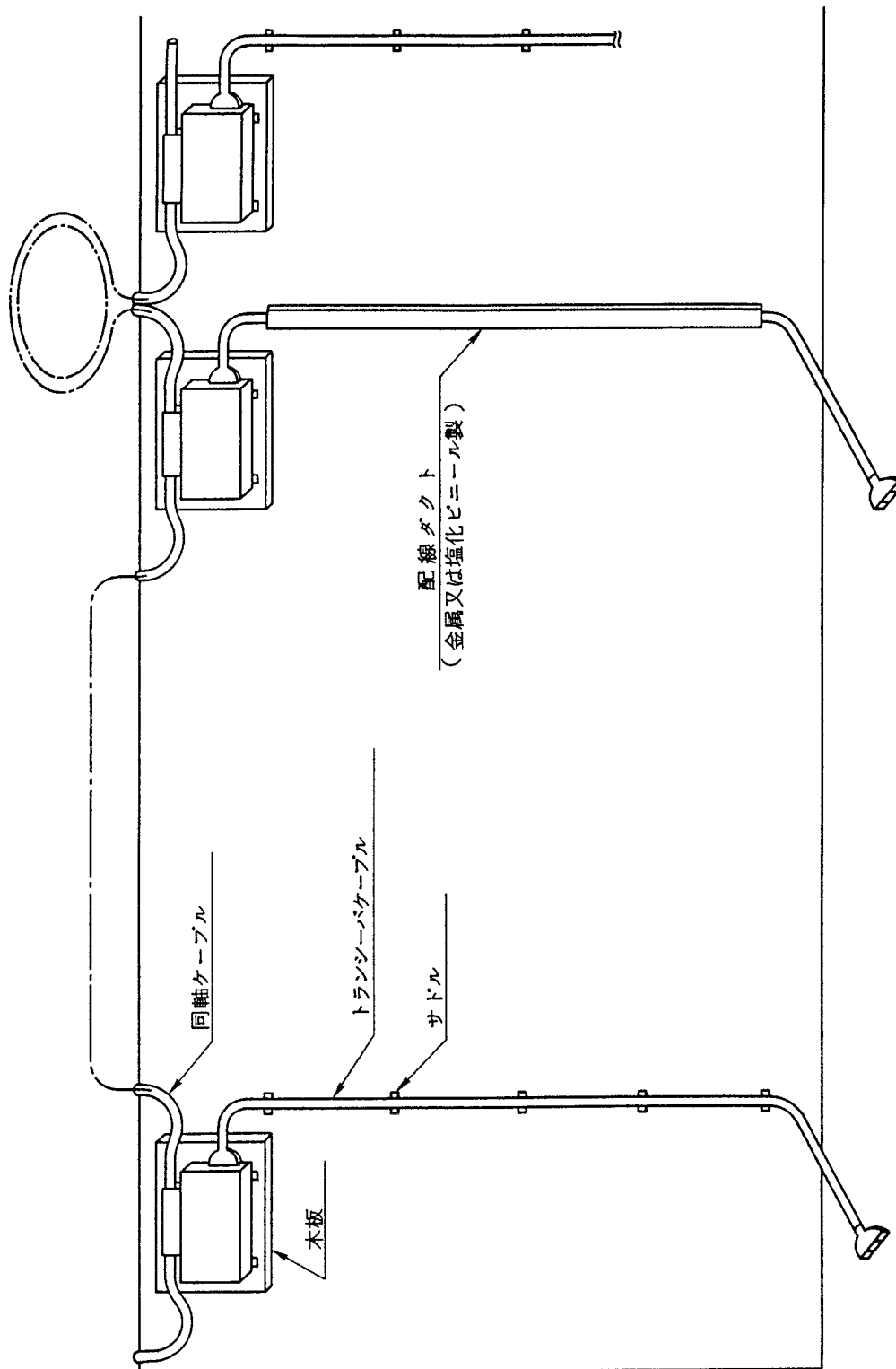


図 8 - 5 壁面設置例

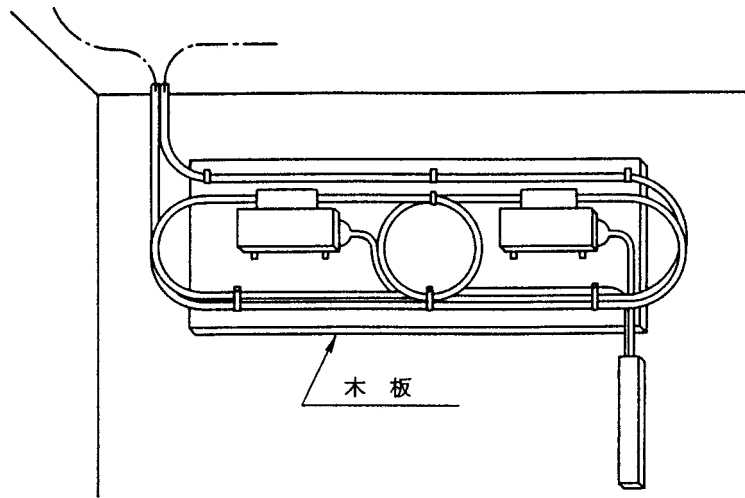


図 8 - 6 壁面設置例

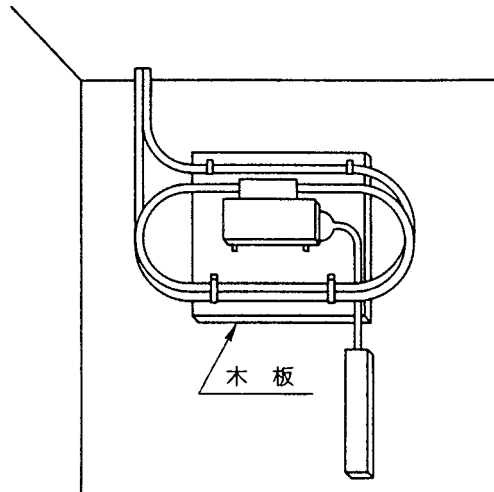


図 8 - 7 壁面設置例

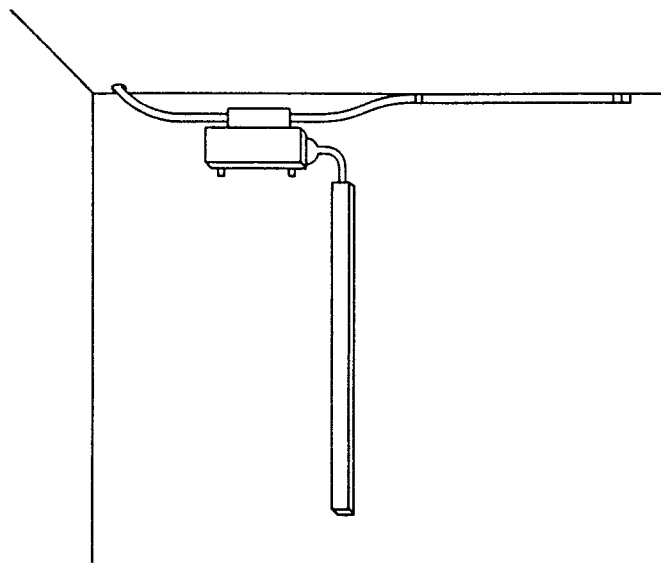


図 8 - 8 壁面設置例

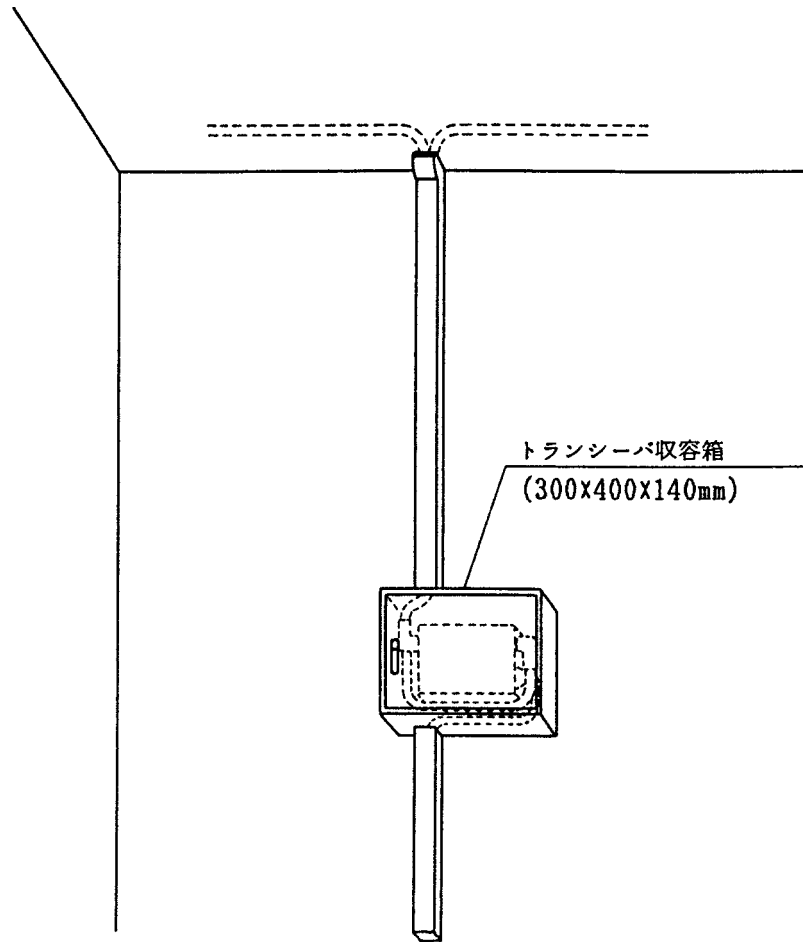


図8 - 9 BOX内設置例

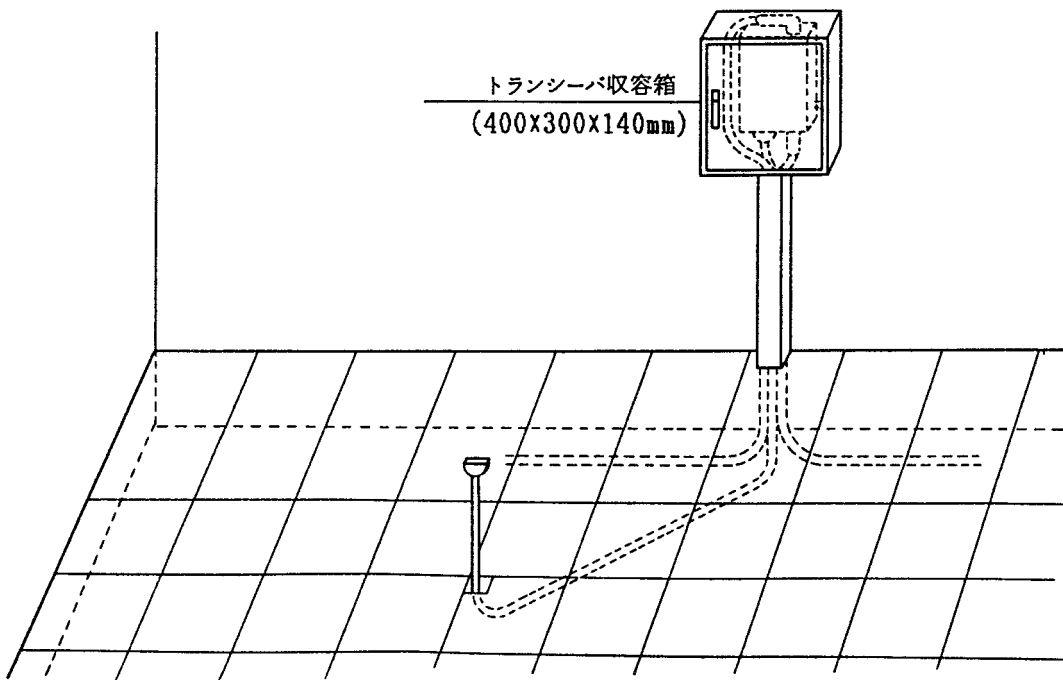


図8 - 10 BOX内設置例

8.5 トランシーバ（タップ形）の設置・取付け

トランシーバの設置場所、取付け方法および留意事項は8.4節と同様です。

なお、タップコネクタの同軸ケーブルの取付けは、「8.7 タップコネクタの取付け」を参考にして行ってください。

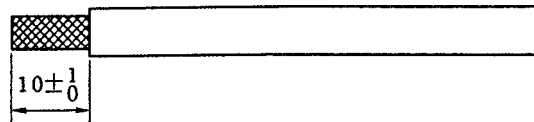
タップ形トランシーバの寸法図は8.1.3項を参照してください。

8.6 同軸コネクタの取付け

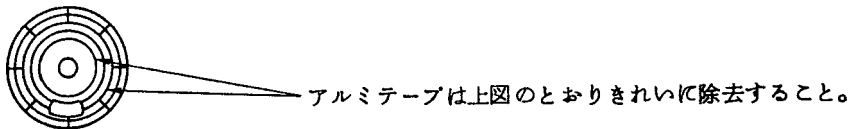
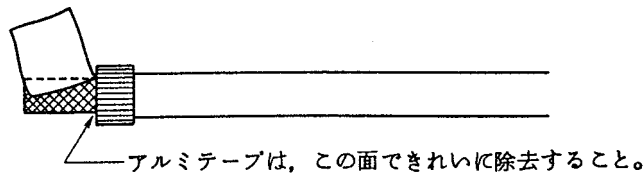
(1) コネクタの取付け手順

同軸コネクタの取付け作業手順を以下に示します。

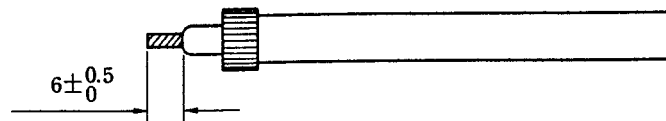
PVCシースムキ



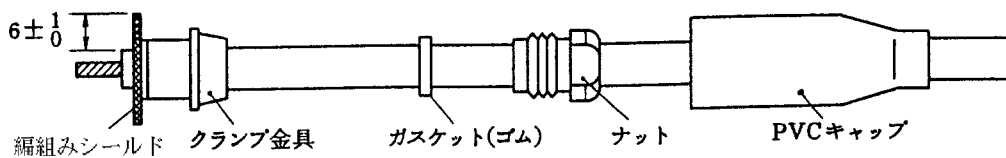
アルミテープ除去



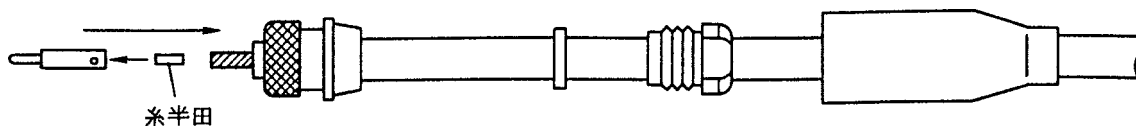
絶縁体ムキ



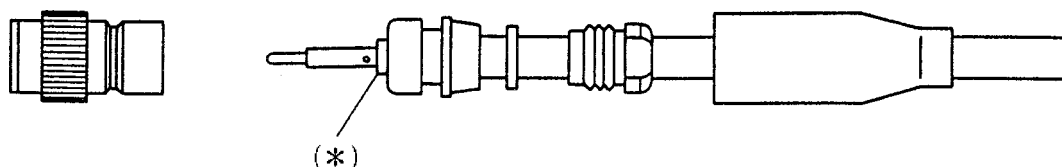
部品組み込みおよびシールド処理



シールド処理およびピンコンタクト半田付け



組 立

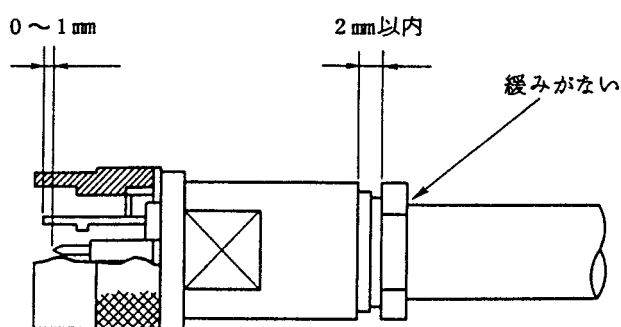


(*) ピンコンタクト絶縁体に 1 mm 以上のすき間および絶縁体内に食い込みのないこと。

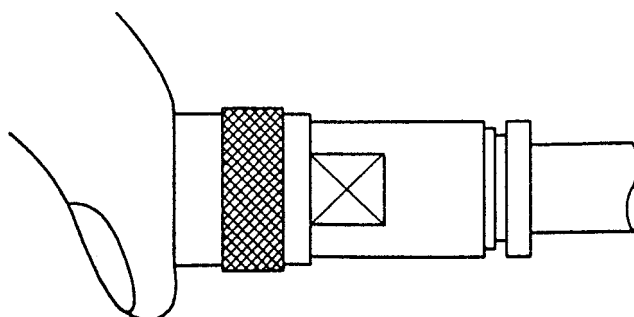
(2) コネクタ取付け後のチェック方法

(a) コネクタ開口部の寸法

- ・コネクタ先端の外部導体と内部のコンタクトの差が 0 ~ 1 mm 以内。異常な内部コンタクトの突出し、または引込みがないこと。



- ・一般にコネクタ開口部に親指を当ててその腹に内部コンタクトの先端が軽く触れる程度



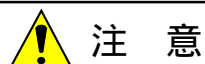
- ・目視により中心導体に異常な偏心がないこと。

(b) ゆるみの確認

- ・取付後、コネクタのボディと同軸ケーブルを手でつかんでひねり、ゆるみがないこと。
- ・締付け後、締付けナットと本体のすき間は約 2 mm 以内であること。

(c) 絶縁抵抗 (ターミネータを外すこと)

- ・トランシーバが付いていないとき
内 - 外導体間 1000M / km 以上 (DC 500V)
- ・トランシーバが付いているとき
一般の回線テスターで外部導体側を内部電池の + 極にして測定し、表示であること。



注 意

試験後は必ず放電してください。放電を行わないと感電します。

8.7 タップコネクタの取付け

タップ形トランシーバのタップコネクタと同軸ケーブルの接続は次のとおりです。

- (1) 同軸ケーブル を、タップコネクタの本体 の溝に挿入し、さらに上部からカバー を取付けることによって、同軸ケーブル を固定します。
- (2) 六角ボルト を、ボックスドライバを使用して規定されたトルクに従ってネジ締付けをして、同軸ケーブル の外部導体と接続させます。
六角ボルトネジ ネジ締付けトルク：30～40 [kg・cm]
- (3) バックアッププローブ 、信号用プローブ の順に、両側から同時にボックスドライバを使い、規定されたトルクでゆっくりネジ締付けをして、同軸ケーブル の中心導体と接続させます。
信号用プローブ
バックアッププローブ } ネジ締付けトルク：20～30 [kg・cm]
- (4) バックアッププローブ の上に、添付されているキャップ を取付けます。
以上によってタップコネクタと同軸ケーブルの接続を完了します。
なお、信号用プローブ およびバックアッププローブ の先端とネジ山はこわれやすいので取扱いに十分注意してください。

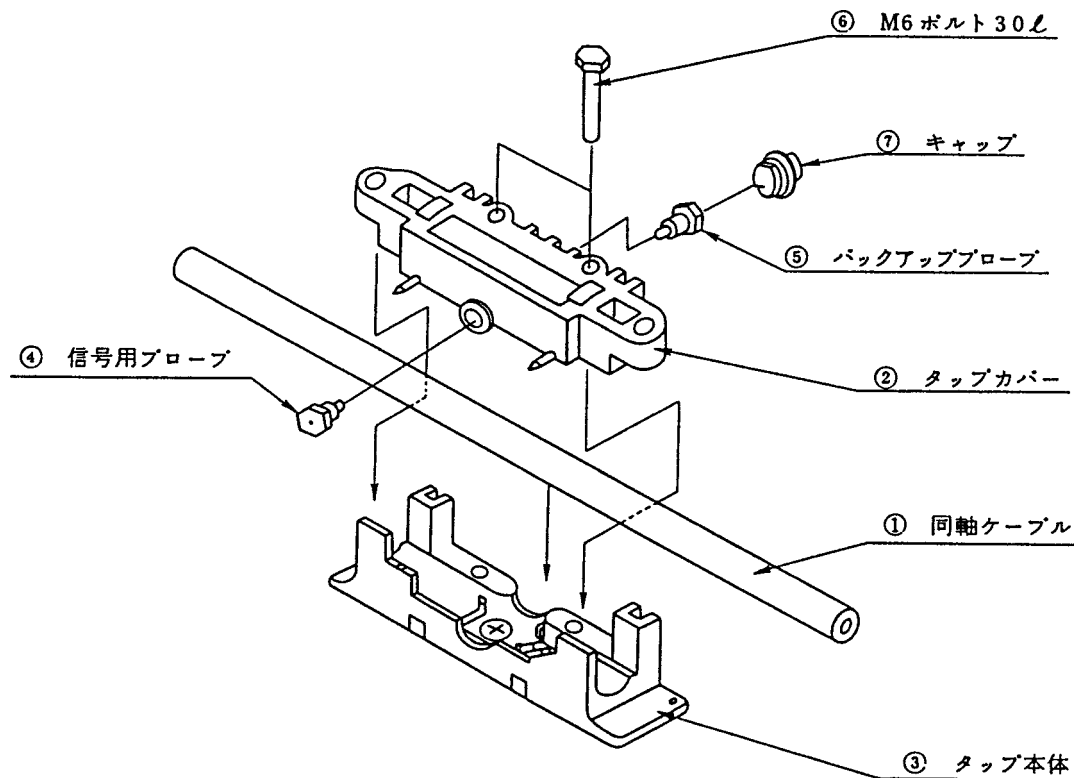


図 8 - 11 タップコネクタ組立図

⚠ 注意

同軸ケーブルへの接続手順は、上記順序で行ってください。
プローブ を取付後、同軸ケーブルを取付けますと、プローブ が破壊されます。それを防ぐためにプローブ を完全にはずした状態で、同軸ケーブルを装着してください。
プローブ の締付の作業後、ボルト の増し締めは行わないでください。プローブ に力が加わり破壊の原因になります。

タップコネクタとトランシーバの接続は、次の手順に従って実施します。

- (1) タップコネクタ をトランシーバ の側面に接着することによって、タップコネクタ のプローブおよびグランド端子がトランシーバ の取付穴に挿入されて、接続されます。
- (2) 六角ボルト をボックスドライバを使用して、規定されたトルクに従ってネジを締付けることによって、トランシーバ とタップコネクタ が完全に固定されます。

六角ボルト ネジ締付けトルク：30～40〔kg・cm〕

以上によってタップコネクタとトランシーバの接続を完了します。

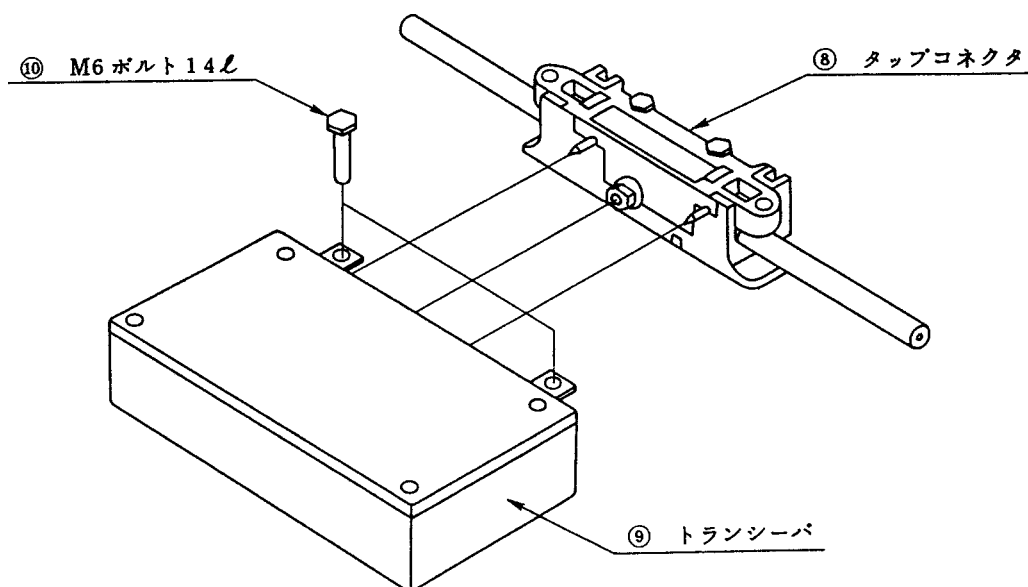


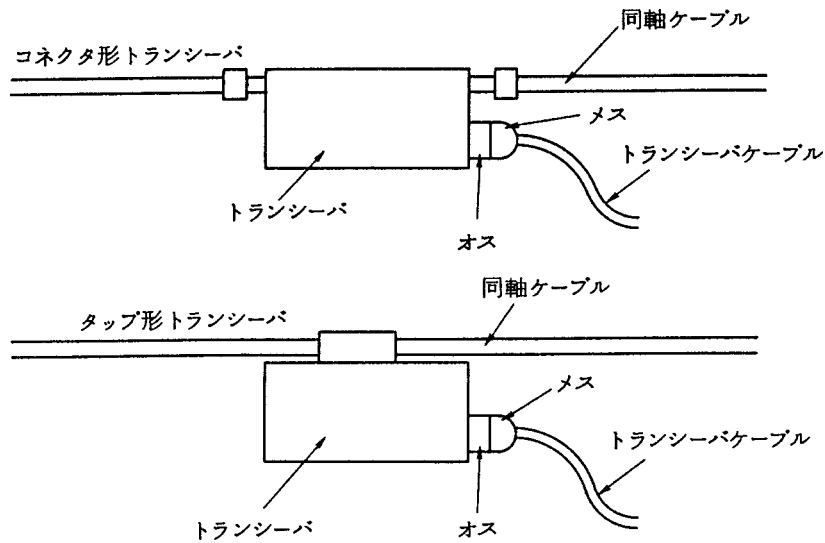
図 8 - 12 コネクタ、トランシーバ接続図

8.8 トランシーバケーブルの取付け

トランシーバケーブルは、最長15mとします。

<トランシーバケーブルの取付け>

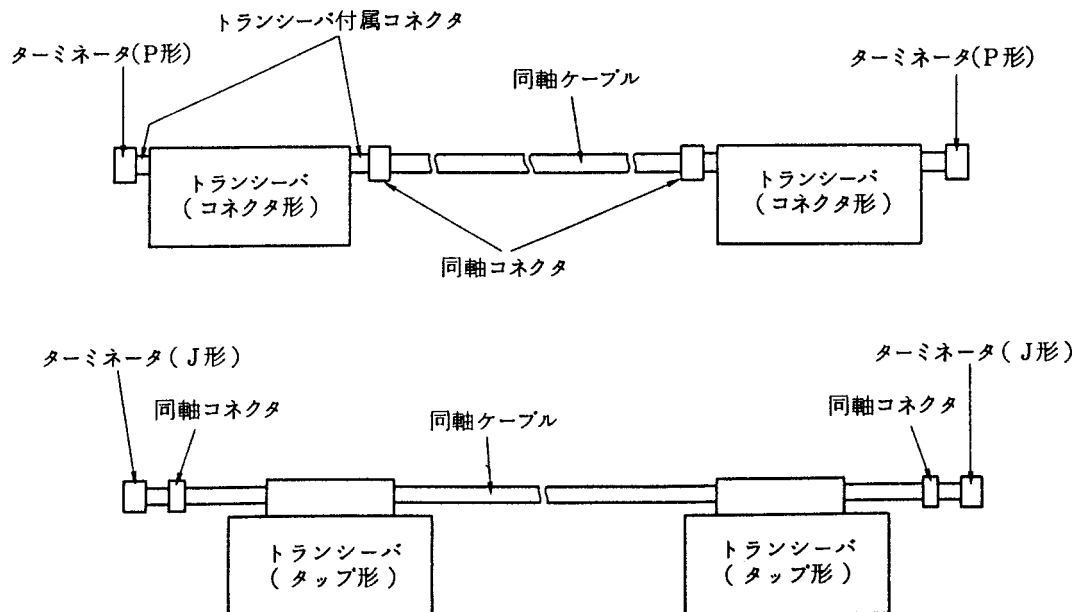
トランシーバ本体へのトランシーバケーブルの接続は、ケーブルのロック用リテーナをスライドさせ、トランシーバ本体のロック用ポストに完全にロックするように取付けてください。トランシーバ本体がオス、トランシーバケーブルがメスとなります。



8.9 ターミネータの取付け

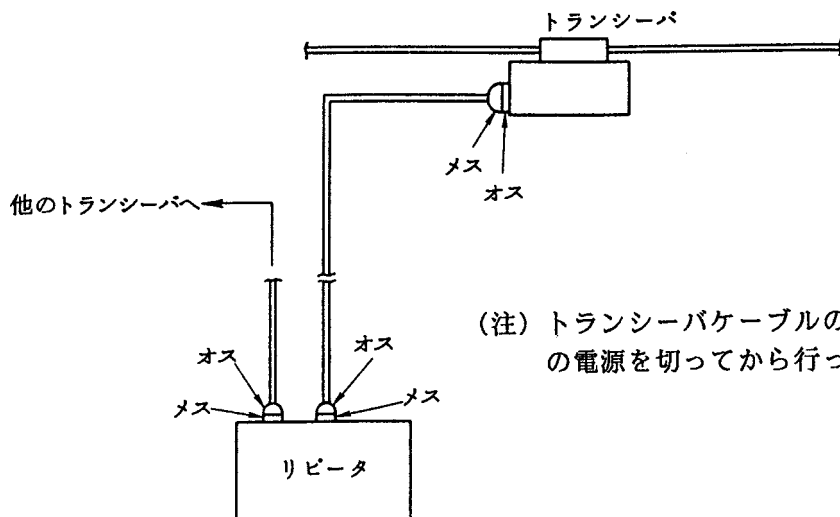
<ターミネータの取付け>

ターミネータは同軸セグメントの最端部（両端）に必ず接続してください。



8.10 リピータの設置・取付け

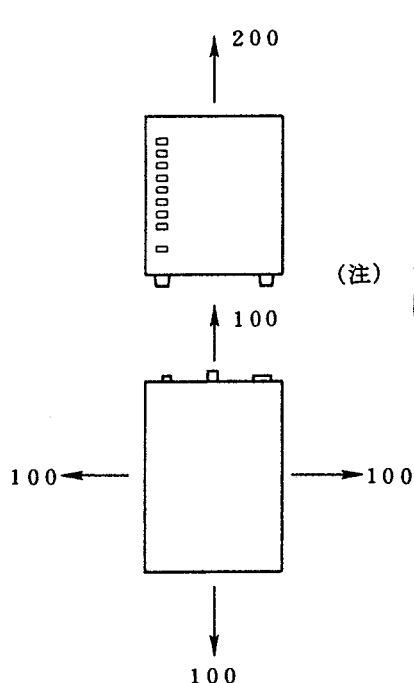
(1) 接続方法



(注) トランシーバケーブルの着脱は、必ずリピータの電源を切ってから行ってください。

(2) 設置場所とスペースの確保

リピータを設置する場所は、ワークステーション(サーバ)付近で、容易に保守できる場所(一般事務室内で天井裏、地下などは不可)を選び、前後、左右、上方に少なくとも以下のスペースを確保してください。なお、リピータはAC電源を必要としますので、接地付コンセントを準備してください。)



定格 AC100V ±10% 平常時 : 0.07 kVA
突入時 : 10A

(注) フロントパネル面は開放できるだけのスペースを確保してください。



2極接地極付プラグ形状
15A125V
(JIS C8303)

単位 : mm

ちりやほこりの多いところでは使用しないでください。
底面に空気の入取れ口、上面に吹出し口がありますのでふさがないでください。
リピータの設置場所付近には保守を考慮し、電話を取付けることを推奨します。
誤って電源を切ることのないよう、独立した電源を使用してください。リピータの電源が切れま
すと、伝送機能が停止します。

8.11 システムの接地

リピータの接地

リピータは、必ず3線式電源を使用するか、または接地端子で接地を行ってください。

各ステーションの接地

LAN制御機構に接続されているすべての装置はD種以上の接地を行ってください。

システム内に接地されていない装置がある場合、接地されている装置との間で感電の恐れがあります。

また、データエラー（CRCエラー）の原因にもなります。

同軸ケーブルの接地

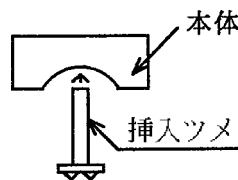
各セグメントごとに同軸ケーブルの1点接地を行ってください。

これは保安上の目的と同時に、不完全な大地との接触による雑音の発生を防止するためです。

接地にはアース端子を使用してください。

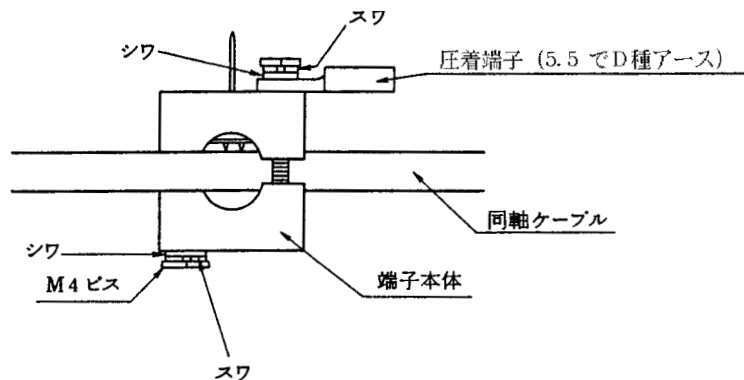
8.12 アース端子取付け方法

- (1) 挿入ツメを本体に挿入します。

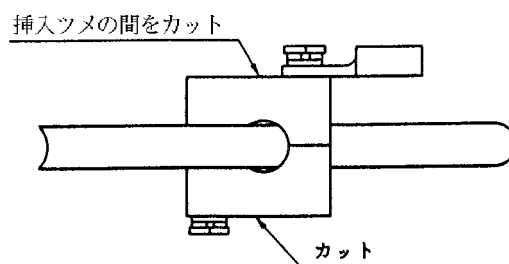


- (2) 同軸に取付けて、M4のビスを交互に締付けます。このとき圧着端子はどちらかのビスに取付けます。

同軸セグメント上の位置はアースが取付けやすい任意の1箇所のみとしてください。



- (3) 締付け後、挿入ツメの間をカットします。



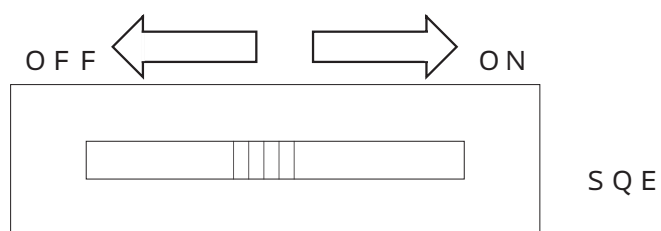
8.13 シングルポートトランシーバの設定

(1) シングルポートトランシーバのSQEスイッチの設定

シングルポートトランシーバのSQEスイッチは、接続先により下表の設定変更が必要となりますので注意してください。

接続先 SQEスイッチ	ET.NET コントローラ	マルチポート トランシーバ	リピータ
設 定	ON	OFF	OFF

なお、シングルトランシーバHLT - 200、HLT - 200TBのSQEスイッチはケース内部にあります。設定を変更するときはケースを開いて実施してください（ボード上のシルク印刷“SQE”側にスイッチを倒すとONになります）。



8.14 マルチポートトランシーバの設定および表示

(1) 動作モードの設定

マルチポートトランシーバは、ネットワークモードとローカルモードの2種類の動作モードで使用できます。モードの設定は、裏面パネル上の切替えスイッチの操作により行います。

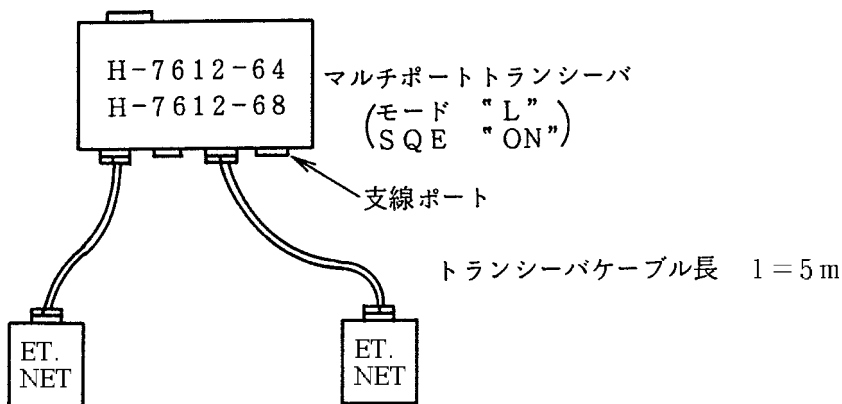
ローカルモード

同軸ケーブルから切離し、単独で使用するモードです。

中継ポートへのトランシーバケーブル接続は行わないでください。

モード切替えスイッチを ' L ' (ローカルモード) に設定して使用します。

また、このとき支援ポートの S Q E スイッチは ' ON ' に設定します。

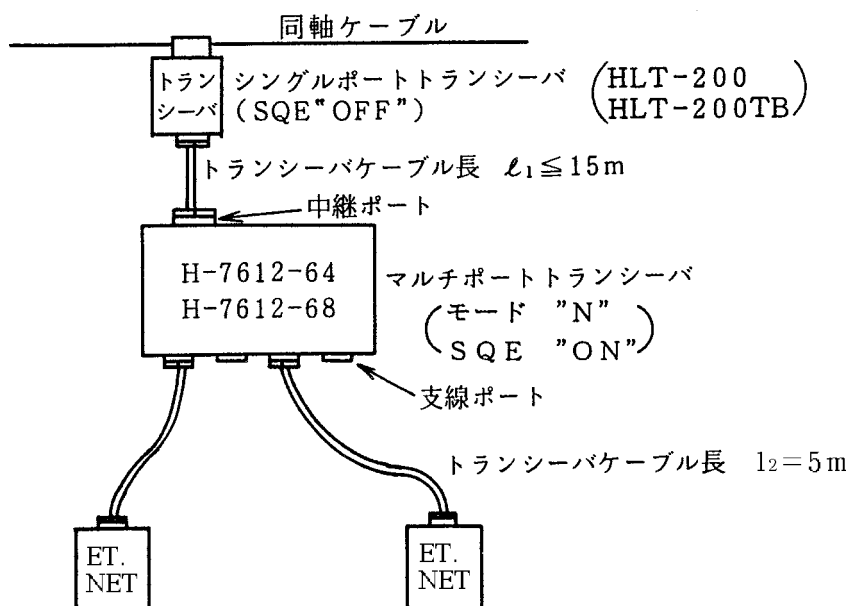


ネットワークモード

下図のように同軸ケーブルと接続して使用するモードです。

モード切替えスイッチを ' N ' (ネットワークモード) に設定して使用します。

また、このとき中継ポートに接続されたシングルポートトランシーバの S Q E スイッチは ' OFF ' に設定します。



(2) 切替えスイッチの設定

マルチポートトランシーバには、2つの切替えスイッチがあります。それぞれの機能を表8 - 5に示します。

表8 - 5 切替えスイッチの設定

スイッチの種類	スイッチの位置	機 能	製品出荷時の設定
SQE切替えスイッチ	裏面パネル	SQE機能のON/OFF	‘ ON ’
動作モード切替えスイッチ	裏面パネル	動作モードの切替え	‘ N ’ (ネットワークモード)

(3) リピータ接続時のSQEスイッチの設定

リピータをマルチポートトランシーバに接続する場合、マルチポートトランシーバの当該支線ポートのSQEスイッチを‘ OFF ’に設定してください。

(4) 電源スイッチ

裏面パネルのスイッチを‘ I ’側に倒すとマルチポートトランシーバの電源が‘ ON ’されます。

(5) LEDの表示

筐体の正面パネル上には、“ POWER ” LEDおよび各支線ポートごとに“ LINK ” LEDがあります。

“ POWER ” LED：電源スイッチが‘ ON ’のときに点灯します。

“ LINK ” LED：情報ステーションがマルチポートトランシーバの支線ポートに接続されているとき（情報ステーションよりDC12Vが給電されているとき）に点灯します。

8.16 ET.NETモジュールのメモリマップ

メインモジュール	サブモジュール	
/ 8 4 0 0 0 0	/ 8 C 0 0 0 0	モジュール情報テーブル
/ 8 4 0 4 0 0	/ 8 C 0 4 0 0	エラーフリーズテーブル
/ 8 4 0 C 0 0	/ 8 C 0 C 0 0	WORKテーブル
/ 8 4 3 0 0 0	/ 8 C 3 0 0 0	T C P情報テーブル
/ 8 4 4 0 0 0	/ 8 C 4 0 0 0	T C P送信バッファ
/ 8 5 4 0 0 0	/ 8 D 4 0 0 0	T C P受信バッファ
/ 8 6 4 0 8 0	/ 8 E 4 0 8 0	U D P情報テーブル
/ 8 6 4 8 8 0	/ 8 E 4 8 8 0	U D P送信バッファ
/ 8 6 7 8 8 0	/ 8 E 7 8 8 0	U D P受信バッファ
/ 8 7 3 8 8 0	/ 8 F 3 8 8 0	

RAM (共有メモリ)

8.17 トラブル調査書

トラブル調査書

貴会社名		担当者		発生日時	月	日	時	分
ご連絡先	ご住所							
	T E L							
	F A X							
不具合モジュール形式				CPU形式				
OS Ver. Rev.	プログラム名：						Ver.	Rev.
サポートプログラム	プログラム名：						Ver.	Rev.
不具合現象								
接続負荷	種 類							
	形 式							
	配線状態							
システム構成およびスイッチ設定								
通 信 欄								

ご利用者各位

〒101-8010

東京都千代田区神田駿河台4丁目6番地
株式会社日立製作所

お 願 い

各位にはますますご清栄のことと存じます。

さて、この資料をより良くするために、お気付きの点はどんなことでも結構ですので、
下欄にご記入の上、当社営業担当または当社所員に、お渡しくださいますようお願い申
しあげます。なお、製品開発、サービス、その他についてもご意見を併記して頂ければ
幸甚に存じます。

ご住所 〒	_____
貴会社名 (団体名)	_____
芳名	_____
製品名	
ご意見欄	_____ _____