ADVANTEST®

株式会社アドバンテスト

D51130

シミュレーション機能モジュール

取扱説明書

MANUAL NUMBER FOJ-8311225C00

初版 1998 年 3 月 1 日
Printed in Japan

本器を安全に取り扱うための注意事項

本器の機能を十分にご理解いただき、より効果的にご利用いただくために、必ずご使用前に取扱説明書をお読み下さい。また、本器の誤った使用、不適切な使用等に起因する運用結果につきましては、当社は責任を負いかねますのでご了承下さい。

本器の操作・保守等の作業を行う場合、誤った方法で使用すると本器の保護機能がそこなわれることがあります。常に安全に心がけてご使用頂くようお願い致します。

■危険警告ラベル

アドバンテストの製品には、特有の危険が存在する場所に危険警告ラベルが貼られています。取り扱いには十分注意して下さい。また、これらのラベルを破いたり、傷つけたりしないで下さい。また、日本国内で製品を購入し海外で使用する場合は、必要に応じて英語版の危険警告ラベルをお貼り下さい。危険警告ラベルについてのお問い合わせは、当社の最寄りの営業所までお願いします。所在地および電話番号は巻末に記載してあります。

危険警告ラベルのシグナル・ワードとその定義は、以下のとおりです。

危険： 死または重度の障害が差し迫っている。
警告： 死または重度の障害が起こる可能性がある。
注意： 軽度の人身障害あるいは物損が起こる可能性がある。

■基本的注意事項

火災、火傷、感電、怪我などの防止のため、以下の注意事項をお守り下さい。

●電源電圧に応じた電源ケーブルを使用して下さい。ただし、海外で使用する場合は、それぞれの国の安全規格に適合した電源ケーブルを使用して下さい。また、電源ケーブルの上には重いものをのせないで下さい。

●電源プラグをコンセントに差し込むときは、電源スイッチを OFF にしてから奥までしっかり差し込んで下さい。

●電源プラグをコンセントから抜くときは、電源スイッチを OFF にしてから、電源ケーブルを引っぱらずにプラグを持って抜いて下さい。このとき、濡れた手で抜かないで下さい。

●電源投入前に、本器の電源電圧が供給電源電圧と一致していることを確認して下さい。

●電源ケーブルは、保護導体端子を備えた電源コンセントに接続して下さい。保護導体端子を備えていない延長コードを使用すると、保護接地が無効になります。

● 3 ピン - 2 ピン変換アダプタ（弊社の製品には添付していません）を使用する場合は、アダプタから出ている接地ピンをコンセントのアース端子に接続し、大地接地して下さい。また、アダプタの接地ピンの短絡に注意して下さい。

●電源電圧に適合した規格のヒューズを使用して下さい。

●ケースを開けたままで本器を使用しないで下さい。

●規定の周囲環境で本器を使用して下さい。

●製品の上に物をのせたり、製品の上から力を加えたりしないで下さい。また、花瓶や薬品などの液体の入った容器を製品のそばに置かないで下さい。

●通気孔のある製品については、通気孔に金属類や燃えやすい物などを差し込んだり、落としたりしないで下さい。

●台車に載せて使用する場合は、ベルト等によって落下防止を行って下さい。

●周辺機器を接続する場合は、本器の電源を切ってから接続して下さい。

■取扱説明書中での注意表記

取扱説明書中で使用している注意事項に関するシグナル・ワードとその定義は以下のとおりです。

危険： 重度の人身障害（死亡や重傷）の恐れがある注意事項
警告： 人身の安全／健康に関する注意事項
注意： 製品／設備の損傷に関する注意事項または使用上の制限事項

■製品上の安全マーク

アドバンテストの製品には、以下の安全マークが付いています。


： 取扱い注意を示しています。人体および製品を保護するため、取扱説明書を参照する必要のある場所に付いています。


： アース記号を示しています。感電防止のため機器を使用する前に、接地が必要なフィールド・ワイヤリング端子を示しています。


： 高電圧危険を示しています。1000V 以上の電圧が入力または出力される場所に付いています。


： 感電注意を示しています。

■寿命部品の交換について

計測器に使用されている主な寿命部品は以下のとおりです。
製品の性能、機能を維持するために、寿命を目安に早めに交換して下さい。
ただし、製品の使用環境、使用頻度および保存環境により記載の寿命より交換時期が早くなる場合がありますので、ご了承下さい。
なお、ユーザによる交換はできません。交換が必要な場合は、当社または代理店へご連絡下さい。

製品ごとに個別の寿命部品を使用している場合があります。
本書、寿命部品に関する記載項を参照して下さい。

主な寿命部品と寿命

部品名称	寿命
ユニット電源	5 年
ファン・モータ	5 年
電解コンデンサ	5 年
液晶ディスプレイ	6 年
液晶ディスプレイ用バックライト	2.5 年
フロッピー・ディスク・ドライブ	5 年
メモリ・バックアップ用電池	5 年

■ハード・ディスク搭載製品について

使用上の留意事項を以下に示します。

●本器は、電源が入った状態で持ち運んだり、衝撃や振動を与えないで下さい。
ハード・ディスクの内部は、情報を記録するディスクが高速に回転しながら、情報の読み書きを行っているため、非常にデリケートです。

●本器は、以下の条件に合う場所で使用および保管をして下さい。
極端な温度変化のない場所
衝撃や振動のない場所
湿気や埃・粉塵の少ない場所
磁石や強い磁界の発生する装置から離れた場所

●重要なデータは、必ずバックアップを取っておいて下さい。
取扱方法によっては、ディスク内のデータが破壊される場合があります。また、使用条件によりますが、ハード・ディスクには、その構造上、寿命があります。
なお、消失したデータ等の保証は、いたしかねますのでご了承下さい。

■本器の廃棄時の注意

製品を廃棄する場合、有害物質は、その国の法律に従って適正に処理して下さい。
有害物質： (1) PCB（ポリ塩化ビフェニール）
(2) 水銀
(3) Ni-Cd（ニッケル - カドミウム）
(4) その他
シアン、有機リン、六価クロムを有する物およびカドミウム、鉛、砒素を溶出する恐れのある物（半田付けの鉛は除く）
例： 蛍光管、バッテリ

■使用環境

本器は、以下の条件に合う場所に設置して下さい。
●腐食性ガスの発生しない場所
●直射日光の当たらない場所
●埃の少ない場所
●振動のない場所
●最大高度 2000 m

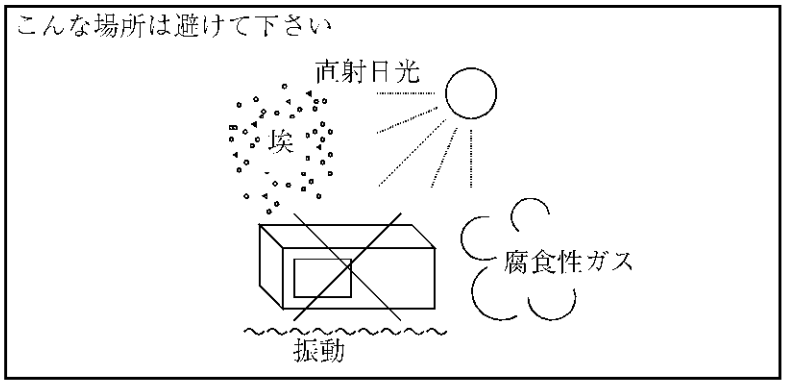


図 -1 使用環境

●設置姿勢
本器は、必ず水平状態で使用して下さい。
本器は内部温度上昇をおさえるため、強制空冷用のファンを搭載しております。
ファンの吐き出し口、通気孔をふさがないで下さい。

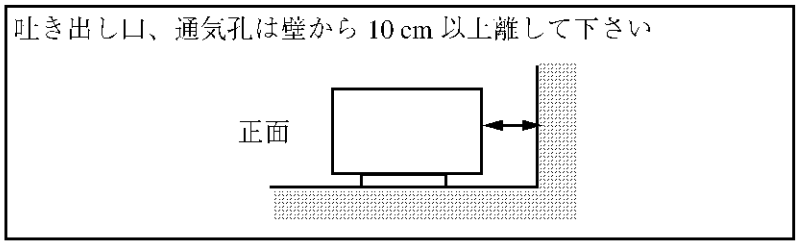


図 -2 設置

●保管姿勢
本器は、なるべく水平状態で保管して下さい。
本器を立てた状態で保管する場合、または運搬時、一時的に立てた状態で置く場合、転倒しないよう注意して下さい。衝撃・振動により転倒する恐れがあります。

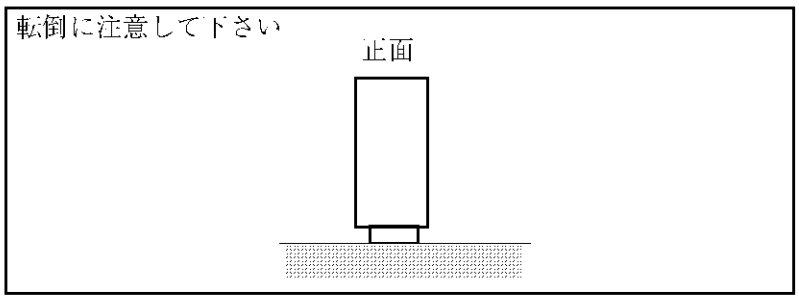
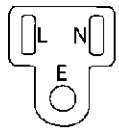
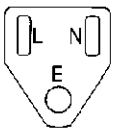
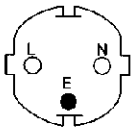
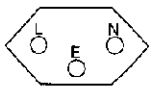
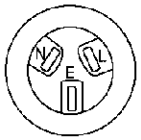
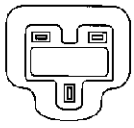
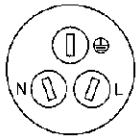


図 -3 保管

● IEC61010-1 で定義される、主電源に典型的に存在する過渡過電圧および汚染度の分類は、以下のとおりです。
IEC60364-4-443 の耐インパルス（過電圧）カテゴリ II
汚染度 2

■電源ケーブルの種類

「電源ケーブルの種類」の記述が本文中にある場合には、以下の表に置き替えてお読みください。

プラグ	適用規格	定格・色・長さ	型名（オプション No.）
	PSE: 日本 電気用品安全法	125V/7A 黒、2m	ストレート・タイプ A01402 アングル・タイプ A01412
	UL: アメリカ CSA: カナダ	125V/7A 黒、2m	ストレート・タイプ A01403 (オプション 95) アングル・タイプ A01413
	CEE: ヨーロッパ DEMKO: デンマーク NEMKO: ノルウェー VDE: ドイツ KEMA: オランダ CEBEC: ベルギー OVE: オーストリア FIMKO: フィンランド SEMKO: スウェーデン	250V/6A 灰、2m	ストレート・タイプ A01404 (オプション 96) アングル・タイプ A01414
	SEV: スイス	250V/6A 灰、2m	ストレート・タイプ A01405（オプション 97） アングル・タイプ A01415
	SAA: オーストラリア ニュージーランド	250V/6A 灰、2m	ストレート・タイプ A01406（オプション 98） アングル・タイプ ----
	BS: イギリス	250V/6A 黒、2m	ストレート・タイプ A01407（オプション99） アングル・タイプ A01417
	CCC: 中国	250V/10A 黒、2m	ストレート・タイプ A114009（オプション94） アングル・タイプ A114109

緒言

1. この取扱説明書は、D5115 のシミュレーション機能について説明しています。

2. この取扱説明書は、シミュレーション機能モジュール (D51130) に標準付属品として添付されます。

3. 本書の構成
   本書の章構成は、以下のとおりです。

1. はじめに	安全に使用するため、使用開始の前に必ずお読み下さい。
2. シミュレーション言語 3. 簡単なプログラムの作成 4. メッセージ・ビルダ 5. 保存 / 読み出し (セーブ / ロード) ・メニュー 6. シミュレーション言語 (PSL51) の構文について	シミュレーション機能およびシミュレーション言語について説明します。 プログラミングに必要なコマンド一覧を示します。
7. 機能別関数一覧 8. D チャネル・シミュレーション 9. B チャネル・シミュレーション 10. 性能諸元 付録　シミュレーションを使いこなすために	

4. 本書内での表記ルール
   本書ではキーボードのキーとメニュー項目を以下のように表記してあります。

   キーボードのキーの表記　：ボールド　　例：**Ctrl**, **Shift**, **A**, **X**
   メニュー項目の表記　　：ボールド・イタリック　例：***OK***, ***Exit***

   キー操作
   Ctrl -A　：**Ctrl** を押しながら **A** を押すことを示します。
   Shift - X　：**Shift** を押しながら **X** を押すことを示します。
   Spc　　：スペース・キーを示します。

目次

図一覧

表一覧

1. はじめに

この章では本器の概要、付属品、パネルおよび挿入方法などを説明しています。本器を使用する前に必ずお読み下さい。

1.1 製品概要

本器は D5115 マルチメディア・プロトコル・アナライザにより、端末側または網側をシミュレーションするために使用します。同線に接続するためには、D51101 基本インタフェース・モジュールあるいは D51102 U 点インタフェース・モジュールが必要です。
本器は、ISDN における D チャネル・プロトコルである LAPD および B チャネル上で使用されるプロトコルである LAPB をシュミレーションすることができます。下表に標準機能およびオプションで拡張される機能を示します。

構成 / 機能	シミュレーション機能モジュールのみ	シミュレーション機能モジュール＋シミュレーション機能追加オプション
LAPD シミュレーションの実行	1 チャネル	同時に 2 チャネル
LAPB シミュレーションの実行	1 チャネル	同時に 2 チャネル
音声シミュレーション 1	1 チャネル (B チャネル)	1 チャネル (B チャネル)
外部入出力機能	1 入出力	1 入出力
BERT 機能	1 チャネル (B チャネル)	1 チャネル (B チャネル)

注意　インタフェース・モジュールについては、「D5115 取扱説明書 A1 本器で搭載可能な機能モジュールとインタフェース・モジュール」を参照して下さい。

1.2 付属品

シミュレーション機能モジュールには、以下のものが標準で付属しています。

品名	型名	数量	備考
本取扱説明書	JD51130	1	
ヘッド・セット	AAA-HBH0030-2	1	

1.3 パネルの説明

シミュレーション機能モジュールのパネルには、以下の端子があります。

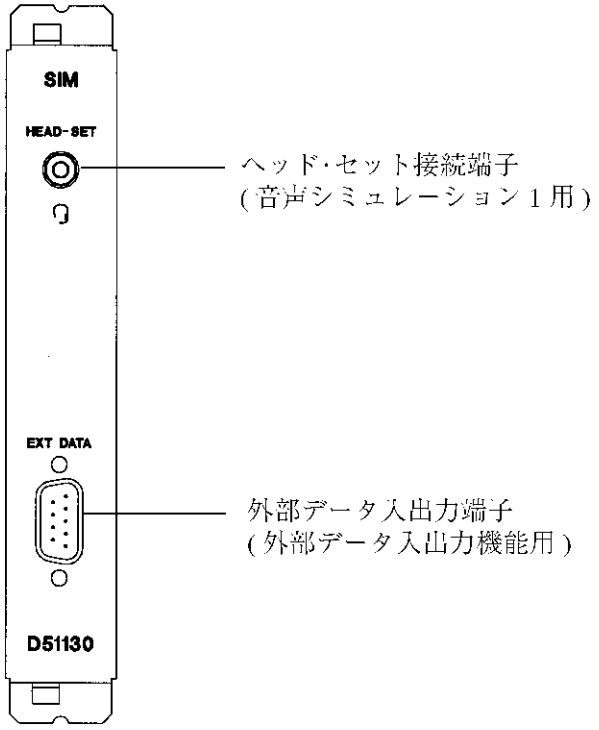
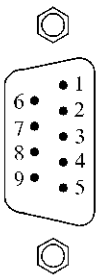


図 1-1 シミュレーション機能モジュールのパネル

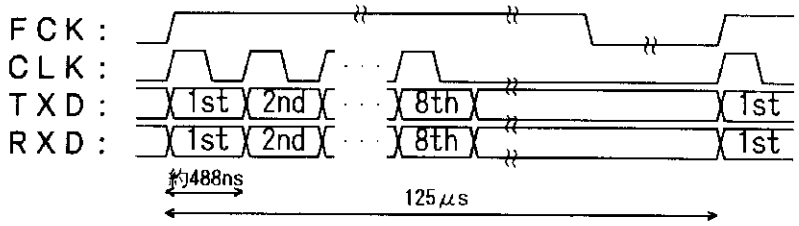


Dsub9 ピン male 端子
(DDK 製 17LE-23090-27(D4CC))

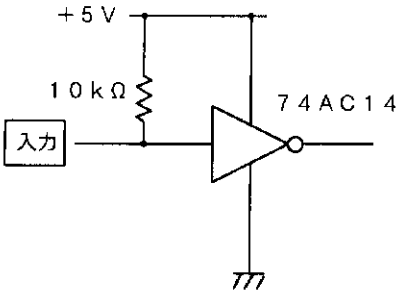
1: GND
2: CLK、クロック (64kHz) 出力
3: RXD、データ入力
4: ——
5: ——
6: FCK、フレームクロック (8kHz) 出力
7: TXD、データ出力
8: ——
9: ——

注意：端子４、５、８、９には何も接続しないで下さい。

・信号タイミング



・入力部 等価回路



・出力部 等価回路

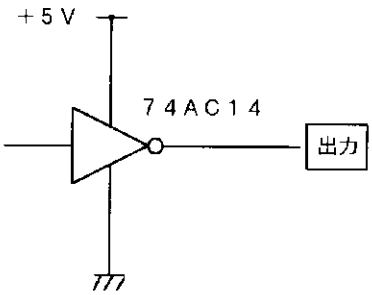


図 1-2 外部データ入出力端子のピン番号と機能

1.4 モジュールの挿入方法

本器にシミュレーション機能モジュールを挿入する手順は、「D5115 取扱説明書 3.2 機能モジュールとインタフェース・モジュールの挿入」を参照して下さい。モジュールの挿入は、必ず電源を切った状態で行って下さい。

1.5 システム・コンフィグレーション画面

本器の起動が終了すると画面にシステム・コンフィグレーション画面が表示されます。このとき、シミュレーション機能モジュール (D51130) が搭載されているスロット・フォルダに SIM と表示されます。(図 1-3 を参照)

このシミュレーション機能のスロット・フォルダには、シミュレータのログ・ファイル設定領域、シミュレータのオプション設定領域が表示されています。

また、機能モジュール選択メニューに以下の機能モジュールが追加されます。

・LAPD シミュレータ 1
・LAPB シミュレータ 1
・LAPD メッセージ・ビルダ
・LAPB メッセージ・ビルダ

シミュレーション機能モジュール (D51130) をさらに追加するか、シミュレーション機能追加オプション (OPT D51130+01) を追加すると、以下の各機能モジュールが追加されます。(図 1-4 を参照)

・LAPD シミュレータ 2
・LAPB シミュレータ 2

注意　以下、特別な場合を除いてすべて LAPD シミュレータ 1 および LAPB シミュレータ 1 を使用して操作を説明しますが、LAPD シミュレータ 2、3、4 および LAPB シミュレータ 2、3、4 でも同じように機能を実行することが出来ます。

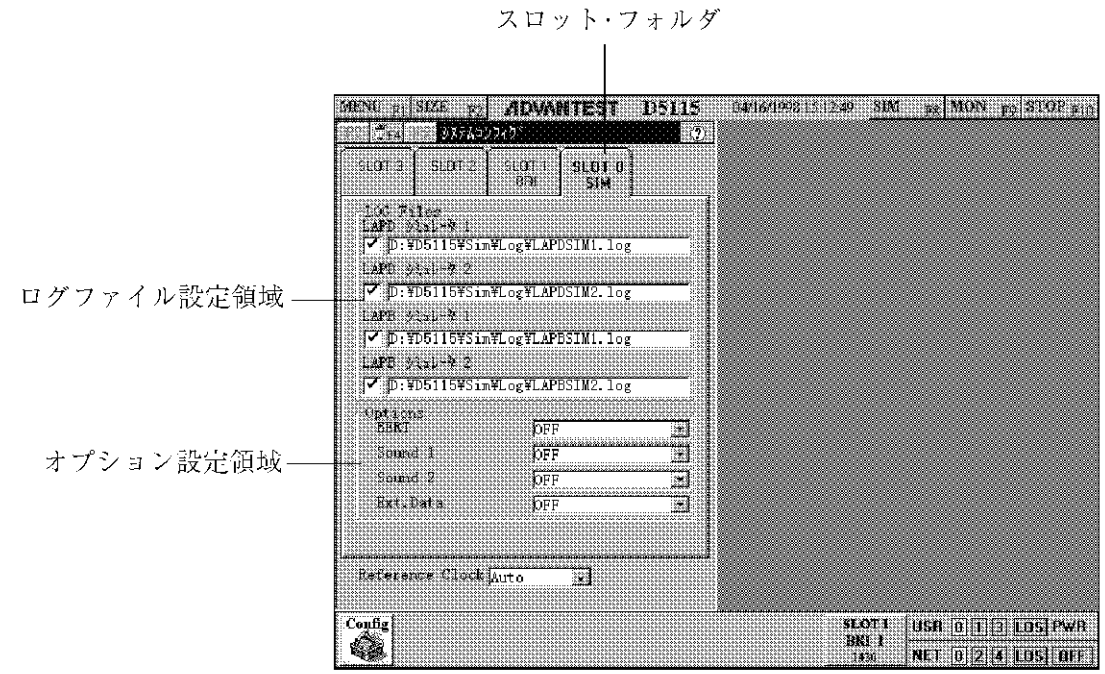


図 1-3 システム・コンフィグレーション画面

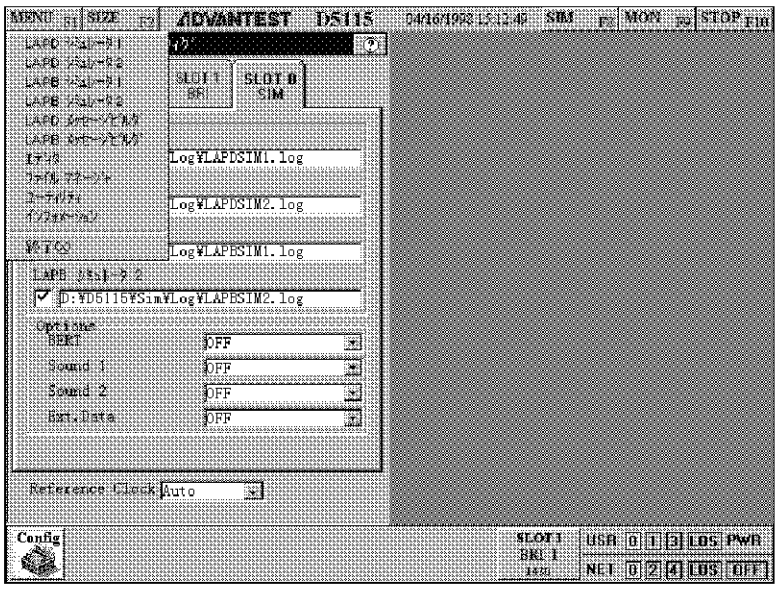


図 1-4 機能モジュール選択メニュー

1.5.1 シミュレータのログ・ファイル設定

LAPD/LAPB シミュレータは、シミュレーション中の画面表示結果をログ・ファイルとして記録することが出来ます。
該当するシミュレータの左欄にチェックを入れると、その右欄に指定したファイル名でログファイルが作成されます。
このログ・ファイルは、ファイルマネージャで表示したり、エディタで編集したりする事が出来ます。

1.5.2 シミュレータのオプション設定

LAPB シミュレータの機能オプションとして、以下の 4 つがあります。
これらの機能オプションを各シミュレータで使用可能にするために、シミュレータのスロット・フォルダにあるオプション項目を設定して下さい。
標準の設定では「OFF」(使用しない) となっています。

- ・BERT : ビット・エラーレート測定、基準正弦波発生
- ・Sound1 : ヘッド・セットによる音声通話
- ・Sound2 (注) : 600Ω 系アナログ音声の入出力
- ・Ext.Data : 外部データ入出力ポート

割り当てられていない機能オプションをシミュレータで使用すると、エラー（指定したシミュレータは使用できない）となります。詳細は「9. B チャネル・シミュレーション」を参照して下さい。

注　Sound2 は、現バージョンでは対応していません。

---

2. シミュレーション言語

2.1 PSL51

シミュレーション言語 (PSL51 : Program Simulation Language for D51 シリーズ、以下 PSL51 と記します。) には、以下の機能があります。

(1) ユーザにより作成されたフレームの送出

(フレームは、LAPD メッセージ・ビルダ、または LAPB メッセージ・ビルダ画面により作成します。)

(2) 受信したフレーム内容の判定

表 2-1 PSL51 言語仕様

データ型	整数型	符号付 32 ビット (–2147483648 ~ +2147483647)
配列	整数型	1 次元配列
算術演算子	+ – * /	加算 減算 乗算 除算
関係演算子	>= <= == !=	より大きい より小さい より大きいか、等しい より小さいか、等しい 等しい 等しくない
論理演算子	!	否定
ビット毎の論理演算子	& |	ビット毎の論理積 ビット毎の論理和
関数	FUNC	整数型の値を持つ
文	代入文 IF 文 FOR 文 WHILE 文 CASE 文 EXIT 文 RETURN 文	VAL = A1 + BB * XYZI IF A==1 THEN X=0 ELSE X=1 END FOR I=0 TO 100 DO ..... END WHILE X=1 ..... END CASE XY + 1 OF '1"2' ..... END WHILE 文のループから抜ける 関数からの復帰

2.2 プログラミング構造

2.2.1 PSL51 のプログラミング構造例

PSL51 の基本的プログラミング構造例を以下に示します。

```
CHANNEL   D
LAYER     2
SIMMODE   TE
PFEED     OFF

ARRAY  A[8], XYZ[10]
ARRAY  X[3] = [0,1,2]
ARRAY  Y[10] = [0,1,2,3[7]]

FUNC  MAIN( )
          ⋮
      SUB1( )
          :
RETURN

FUNC  SUB1( )
        :
      RT =SUB2( )
RETURN

FUNC  SUB2( )
      BB = 3
        :
RETURN(BB)
```

シミュレーションの宣言文

配列の宣言文

主プログラム関数

関数

関数

プログラムは MAIN( ) という名の主プログラム関数から実行します。したがって、プログラム中に MAIN( ) という名の関数が必ず存在しなくてはなりません。
関数以外では、宣言文のみ認識されます。

2.3 シミュレーションの宣言文

シミュレーションを行う場合、宣言文によりあらかじめ使用モードを明記しておく必要があります。宣言文は、プログラムの先頭に記述する必要があります。
宣言文により定義するものは、以下の 7 種類です。

(1) CHANNEL D/B

D : D チャネル上でシミュレーションを行います。
B : B チャネル上でシミュレーションを行います。

(2) LAYER 2/3

2 : トランスペアレント・モードによる実行です。
3 : レイヤ 2 自動実行モードによる実行です。

(3) SIMMODE

TE : TE モードによる実行です。
NT : NT モードによる実行です。

注意 (7) の宣言文が U 点インタフェース時は、NT モードのみ有効です。

(4) PFEED OFF/NORM/RVS

OFF : 給電なし。
NORM : ノーマル給電を行います。
RVS : リバース給電を行います。

注意 (7) の宣言文が、基本インタフェース時のみ有効で、その他では無効です。
(3)、(4) の宣言文は、D チャネルと B チャネル・プログラムの両方で宣言される場合、D チャネル・プログラムで宣言される方が有効です。(4) の宣言文は、本器が (3) で NT に宣言されたときのみ有効です。

(5) ADDRESS DTE/DCE

DTE : 自局のアドレスを DTE に指定します。
DCE : 自局のアドレスを DCE に指定します。

(6) MODULO 8/128

8 : モジュロ 8 にてシミュレーションを行います。
128 : モジュロ 128 にてシミュレーションを行います。

注意 (5)、(6) の宣言文は本器が (1) で B チャネル・シミュレーションに宣言されたときのみ有効です。

(7) INTERFACE BRI/UI/PRI/PTI

BRI : 基本インタフェースでシミュレーションを行います。
UI : U 点インタフェースでシミュレーションを行います。
PRI : PTI（一次群 1.5M インタフェース）でシミュレーションを行います。
PTI : PTI（一次群 1.5M インタフェース）でシミュレーションを行います。

注意 この宣言を省略すると基本インタフェースでのシミュレーションになります。

〈宣言文による宣言例〉

```
INTERFACE    BRI ..... 基本インタフェース
CHANNEL      D ..... D チャネル・シミュレーション
LAYER        3..... レイヤ 2 自動モード（レイヤ 3 モード）
SIMMODE      NT ..... NT モード
PFEED        RVS ..... リバース給電
```

2.4 基本、一次群およびＵ点インタフェースにおける相違点

基本、一次群とＵ点のシミュレーション・プログラムでは、以下の相違があります。

(1) 宣言文 INTERFACE

基本 I/F	INTERFACE BRI
一次群 I/F	INTERFACE PRI または INTERFACE PTI
U 点 I/F	INTERFACE UI

(2) レイヤ 1 の起動関数

基本 I/F	PH_ACT()
一次群 I/F	－
U 点 I/F	PH_ACT()

一次群シミュレーションでは、レイヤ 1 は常時起動になっています。従ってレイヤ 1 を起動する関数（PH_ACT 関数）を実行する必要はありません。

(3) 使用チャネルの指定関数

- D チャネルの指定

基本 I/F	－	
一次群 I/F	SET_CHANN(24)	(B24 チャネルを指定する場合)
U 点 I/F	－	

- B チャネルの指定

基本 I/F	SET_CHANN(1)	(B1 チャネルを指定する場合)
一次群 I/F	SET_CHANN(24)	(B24 チャネルを指定する場合)
U 点 I/F	SET_CHANN(2)	(B2 チャネルを指定する場合)

基本やＵ点の D チャネル・シミュレーションでは、チャネルの指定を省略できますが、一次群では B1 ～ B24 の範囲で指定する必要があります。
また、B チャネル・シミュレーションでは、必ずチャネル指定が必要です。

(4) レイヤ 2 自動モードで TE シミュレーションを行うときのリンク設定値 (初期化)

レイヤ 2 リンクを設定する場合、以下のように LINKON( ) 関数を使用します。

- 基本インタフェース
  TEI 割当手順を起動したあと、網から割り当てられた TEI 値でリンクの設定を行います。
- 一次群インタフェース
  TEI 値 0 でリンクの設定を行ないます。
- U 点インタフェース
  U 点インタフェースでは、NT シミュレーションのみ有効です。

ただし、REG_TEI( ), ACT_TEI( ), REQ_TEI( ), ACT_SAPI( ) などの関数を使用することにより、初期値以外の SAPI, TEI 値でレイヤ 2 リンクの設定をすることができます。
また、初期値の SAPI 値は 0 です。SAPI 値を 16 で使用したい場合は、ACT_SAPI(16) を実行したあと、LINKON( ) 関数を実行します。

3. 簡単なプログラムの作成

3.1 画面に文字表示

プログラムを用いて、画面に文字を表示させる手順を説明します。
プログラムの実行は、以下の手順で行います。

(1) エディタ機能を使用してソース・プログラムを作成
(2) エディタ機能の「コンパイル機能」を使用して、オブジェクト・プログラムを作成
(3) LPAD シミュレータ 1 画面で実行するオブジェクト・プログラムを指定
(4) オブジェクト・プログラムを実行

プログラムは以下の手順により作成および実行します。

(1) エディタの使用法

1. **F1** を押して、機能モジュール選択メニューを表示させます。↓ (または ↑) でカーソルを「エディタ」に移動して **Enter** を押します。(エディタ機能モジュールの起動)
2. 以下のプログラムをキーボードから入力します。(エディタ機能の詳細は「D5115 取扱説明書 5. エディタ機能」を参照して下さい。)

```
FUNC    MAIN ( )
        PRINT ("Welcome to D5115World")
RETURN
```

(2) コンパイル

1. **F3** を押して、機能選択メニューを表示させます。↓ (または ↑) で「コンパイル」を選択して、**Enter** を押します。
2. 作成した内容をファイルに保存するかしないかを確認するワーニング・メッセージが表示されるので、↓でカーソルを ***Yes*** に移動して、**Enter** を押します。
3. 作成した内容を保存するファイル名を入力するポップアップ・メニューが表示されるので、ファイル名入力領域に

   D:¥D5115¥Sim¥Prg¥MOJI. PRG

   と入力します。

   ↓によりカーソルを ***OK*** に移動して、**Enter** を押します。
   これにより、ファイル MOJI. PRG をディレクトリ D5115¥Sim¥Prg¥ に作成することができます。
4. コンパイルを開始し、画面が上下に分割されます。下部画面に

   コンパイル　　– D:¥D5115¥Sim¥Prg¥MOJI. PRG
   オブジェクト　　– D:¥D5115¥Sim¥Prg¥MOJI. OBJ を作成
   コンパイル成功

   と表示され、コンパイルが終了したことを示します。コンパイルが正常に終了すると、ソース・プログラムを作成したディレクトリ (D5115¥Sim¥Prg¥) 内に

   MOJI. OBJ

   というオブジェクト・プログラムが自動的に作成されます。また、エラー・メッセージが表示されるときは、ソース・プログラムの入力に誤りがあるので、エラー・メッ

セージにしたがってソース・プログラムを修正して再度コンパイルを実行して下さい。

(3) 実行するオブジェクト・プログラムを指定

1. **F1** を押して、↓ (または ↑) で「LAPD シミュレータ 1」を選択し、**Enter** を押して、LAPD シミュレータ 1 画面を表示させます。
2. **F4** を押すと Load/Save メニューが表示されます (図 3-1)。オブジェクト・プログラム指定領域で MOJI. OBJ を指定します。ファイル名の指定方法は、「D5115 取扱説明書 3.7.8 (3) ファイル名入力領域」を参照して下さい。***OK*** にカーソルを移動して、**Enter** を押すと、オブジェクト・プログラム表示領域に MOJI. OBJ が表示されます。

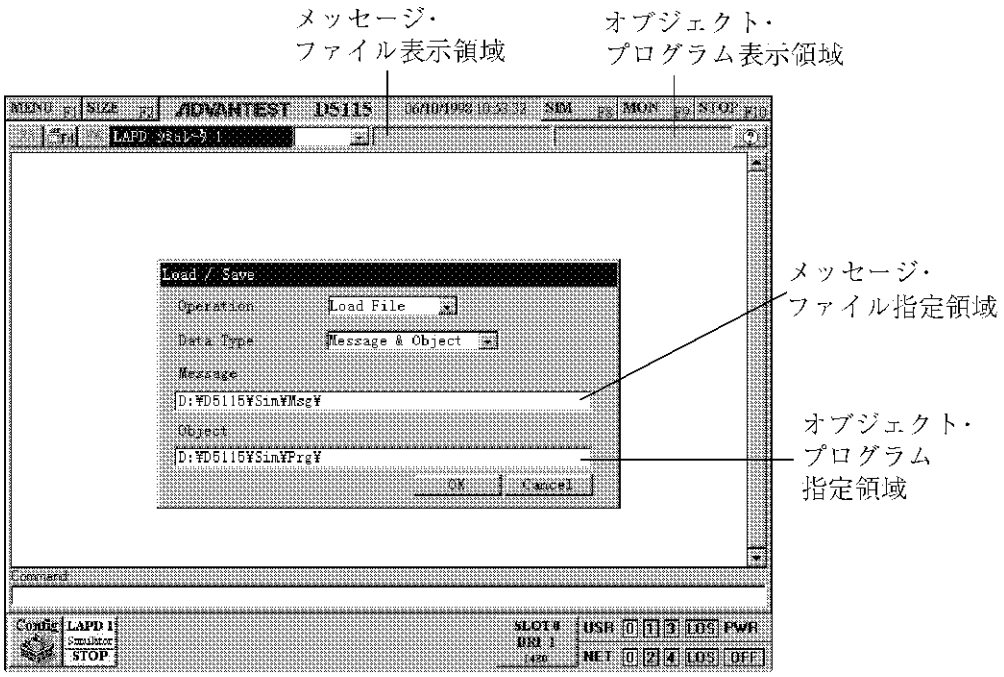


図 3-1 Load/Save メニュー

(4) プログラムの実行

1. **F8** (シミュレーション起動キー) を押すと、画面上に

   Welcome to D5115World

   と、表示されます。また、このとき機能モジュール・アイコン表示領域 (画面最下段) の「LAPD1 Simulator」アイコン表示が

   STOP → RUN → STOP

   と変化します。オブジェクト・プログラムが実行されている間はアイコンが RUN 表示となります。
   また、**F10** (モニタ / シミュレーション機能停止キー) を押すと、オブジェクト・プログラムの実行が停止します。上記のオブジェクト・プログラムは自動的に停止するので、**F10** を押す必要はありません。

3.2 プログラムを用いたシミュレーション

プログラムを用いて、シミュレーションを行う場合、以下に示すフローに従い実行する必要があります。

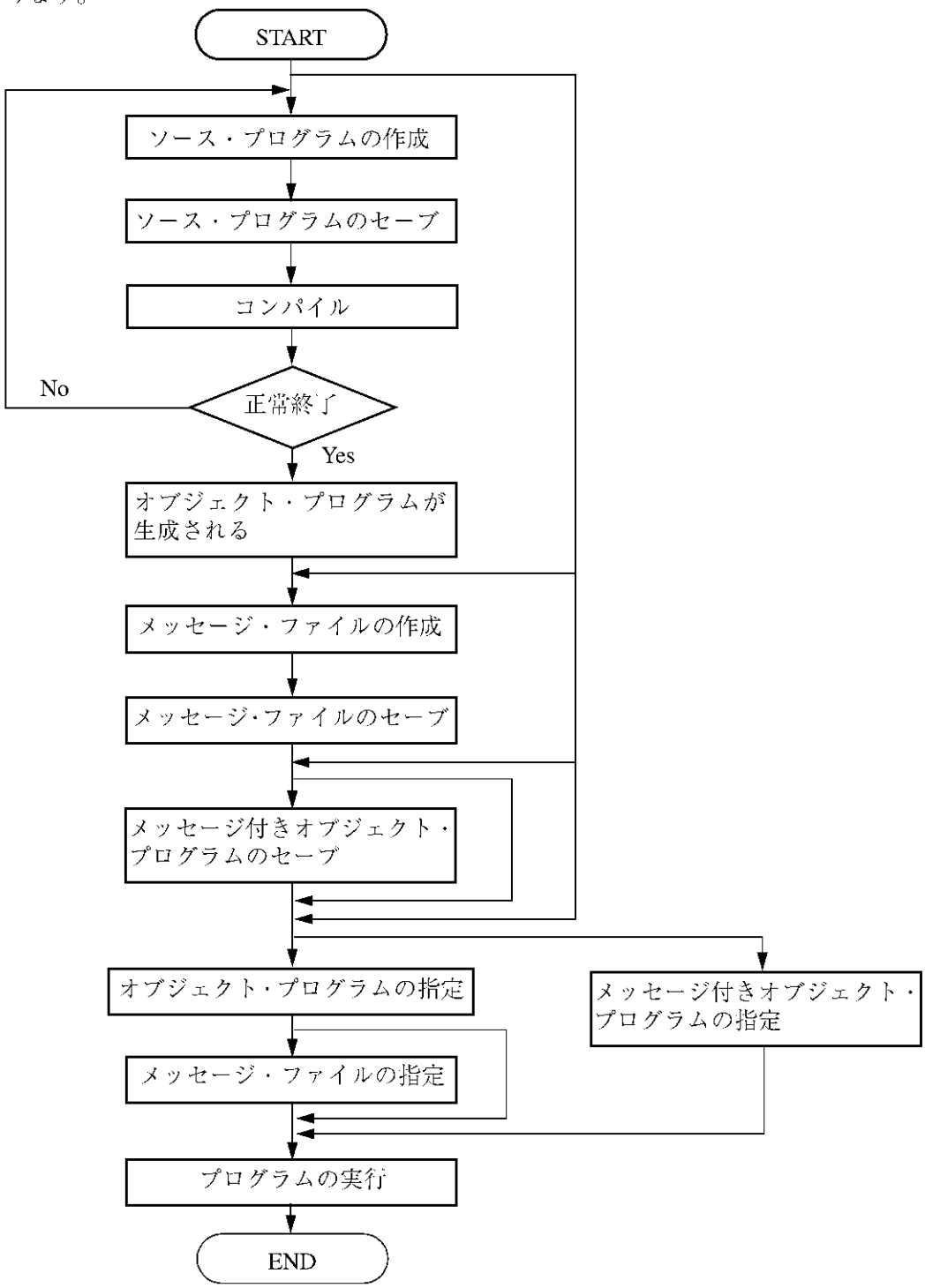


図 3-2 シミュレーションの実行手順

(1) ソース・プログラムの作成

プログラムの作成は、エディタ機能により行います。

1. **F1** を押し、機能モジュール選択メニューを表示させます。↓ (または↑) を押して「エディタ」を選択し、**Enter** を押します。

2 エディタ画面が表示されたら、ソース・プログラムを作成します。エディタの使用法は、「D5115 取扱説明書 5. エディタ機能」を参照して下さい。

(2) コンパイルの実行 (オブジェクト・プログラムの作成)

ソース・プログラムの作成が終了したら、コンパイルを実行してソース・プログラムからオブジェクト・プログラムを作成します。

1. **F1** を押して、機能選択メニューを表示させます。↓ (または↑) により「コンパイル実行」を選択して **Enter** を押します。(コンパイル機能の実行)

2. コンパイル機能を実行すると、エディタ画面が上下に分割され、コンパイルが正常終了すると

   コンパイル　　– D:¥D5115¥Sim¥Prg¥********.PRG
   オブジェクト　– D:¥D5115¥Sim¥Prg¥********.OBJ を作成
   コンパイル成功

   と表示され、ソース・プログラムと同じディレクトリにオブジェクト・プログラム

   ********.OBJ
   (******** は、ソース・プログラム名と同じ)

   が作成されます。ただし、ソース・プログラム内に文法的な誤りがあると、下段 (エラー・メッセージ表示領域) にエラー・メッセージが表示されます。エラー・メッセージが表示された場合は、「3.3 コンパイラによるエラー・コード」と「D5115 取扱説明書 5.4.1 編集領域とエラーメッセージ領域間の連動」を参照して、ソース・プログラムを修正して再度コンパイルを実行します。エラーが表示されなくなるまで、この操作を繰り返します。

(3) オブジェクト・プログラムの指定

オブジェクト・プログラムの作成が終了したら、オブジェクト・プログラムの種類により LAPD シミュレータ 1 または、LAPB シミュレータ 1 画面においてオブジェクト・プログラムを実行します。ここでは、LAPD 用オブジェクト・プログラムを LAPD シミュレータで実行させる場合を示します。

注意　LAPD シミュレータ 2、3、4 (または LAPB シミュレータ 2、3、4) でも同様に LAPD 用オブジェクト・プログラム (または LAPB 用オブジェクト・プログラム) を実行することができます。

---

1. **F1** を押して、機能モジュール選択メニューを表示し、↓ (または↑) により LAPD シミュレータ 1 を選択して **Enter** を押します。

2. **F4** を押すと、Load/Save メニューが表示されます（図 3-3）。カーソル移動 (↑、↓、→、←) を押してカーソルをオブジェクト・プログラム指定領域に移動して **Enter** を押します。「D5115 取扱説明書 3.7.8 (3) ファイル名入力領域」を参照にして、実行するオブジェクト・プログラム名を指定します。

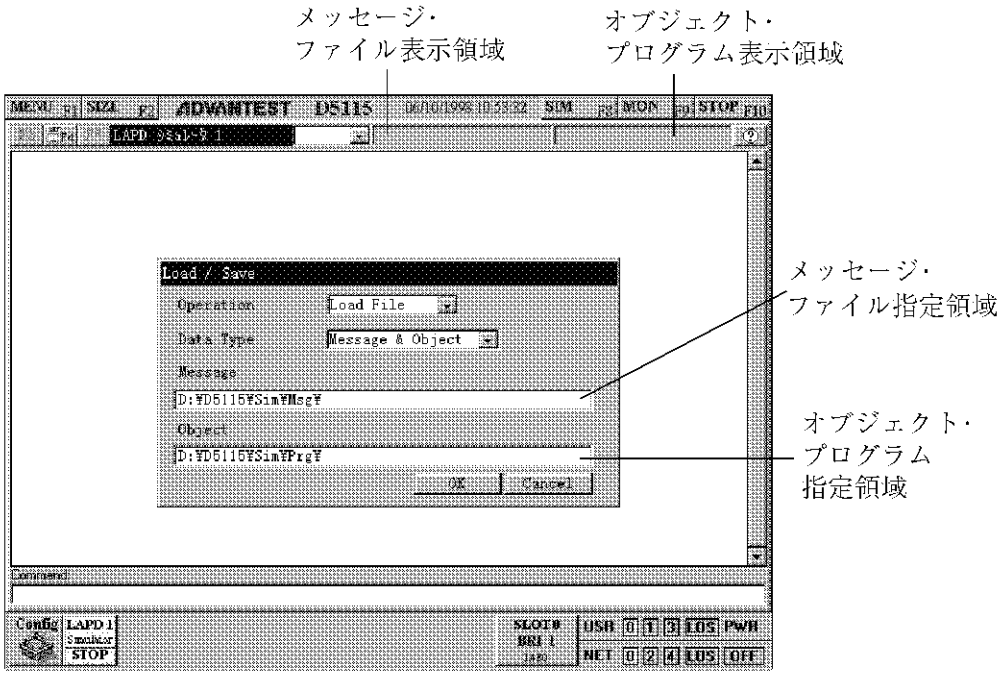


図 3-3 Load/Save メニュー

(4) メッセージ・ファイルの指定

ユーザが作成したメッセージを送信する関数 (SENDT、SENDF、SENDI 関数等) を使用したオブジェクト・プログラムを実行する場合は、使用するメッセージ・ファイル名を指定する必要があります。
メッセージ・ファイル名を指定しないと正常にメッセージを送信することができません。

カーソル移動 (↑、↓、→、←) により、メッセージ・ファイル指定領域にカーソルを移動して **Enter** を押し、使用するメッセージ・ファイル名を指定します。

OK にカーソルを移動して、**Enter** を押すと、オブジェクト・プログラム表示領域、メッセージ・ファイル表示領域に、それぞれ指定したファイルが表示されます。

(5) オブジェクト・プログラムの実行

F8 を押して、オブジェクト・プログラムを実行します。オブジェクト・プログラムの実行を停止する場合は、**F10** を押します。

3.3 コンパイラによるエラー・コード

表 3-1 エラー・メッセージ (1/4)

エラー・メッセージ	概要
カッコが、正常なバランスでない missing (,) or illegal character for operator	カッコが、正常なバランスでない。 また作用素として不適切な文字 〔 〕がある。
予約語を名前として使用している illegal use for RESERVED word	予約語〔 〕を名前として使用している。
作用素が間違っている illegal operator	被作用素が連続している。被作用素があるべき位置に作用素〔 〕がある。
関数名、名札等を変数名として使用している illegal function name or label	関数名、名札等を変数名〔 〕として使用している。
PRINT 文の表記法が不適切である PRINT format error	PRINT 文の表記法が不適切である。
構文エラー SYNTAX error	構文エラー
値が不適切である invalid value	値が不適切である。
関数名として不適切な名前を関数呼出しの形で使用している illegal function name	関数名として不適切な名前〔 〕を関数呼出しの形で使用している。
関数名は既に宣言されている redeclared function name	関数名〔 〕は概に宣言されている
代入の左辺が正しくない the left side is illegal	代入の左辺が正しくない。
配列名に添字がついていない ARRAY needs suffix	配列名に添字がついていない。
配列名に対して、演算が行われている illegal ARRAY use	配列名に対して、演算が行われている。添字として配列名を使用している。IF 文や WHILE 文の条件部分に、配列名だけの式が、使用されている。
配列の添字が正しくない illegal ARRAY suffix	配列の添字が正しくない。“)”があるべき位置に〔 〕がある。

(注) 文中の〔 〕の内の文字はディスプレイ上に表示されます。

表 3-1 エラー・メッセージ (2/4)

エラー・メッセージ	概要
配列の大きさの指定が不適切である illegal ARRAY size	配列の大きさの指定が不適切である。
配列の初期値の並びの区切り記号が間違っている illegal separator	配列の初期値の並びの区切りに記号〔 〕が使用されている。
配列名として不適切である illegal ARRAY name	名前〔 〕は配列名として不適切である。
配列の初期値が少なすぎる needs more initial values	配列の初期値が少なすぎる。
配列の初期値が多すぎる too many initial values	配列の初期値が多すぎる。
引数の形が正しくない illegal arguments	引数の形が正しくない。")" がない。","や")"の代わりに文字〔 〕が使用されている。
引数の区切り記号が間違っている illegal separator	引数の区切りに記号〔 〕が使用されている。
引数として正しくない illegal argument	名前〔 〕は、引数として正しくない。
関数名として不適切である illegal function name	名前〔 〕は、関数名として不適切である。
引数の個数が不適切である illegal argument number	関数〔 〕の引数の個数が不適切である。
関数が未定義である undefined function	関数〔 〕が、未定義である。
IF 文の中に構文エラーがある SYNTAX error for IF	IF の次の式は、形が不適切である。THEN の位置に〔 〕がある。IF 文の中に構文エラーがある。END または ELSE の位置に〔 〕がある。
FOR 文の制御変数が不適切である illegal control variable for FOR	FOR 文の制御変数が不適切である。
FOR 文の中に構文エラーがある SYNTAX error for FOR	FOR 文中で、= の位置が不適切である。= がない。FOR 文中で、増文の式が不適切である。TO がない。FOR 文中で、終値の式が不適切である。DO がない。FOR 文中で、文の並びが不適切である。

(注) 文中の〔 〕の内の文字はディスプレイ上に表示されます。

表 3-1 エラー・メッセージ (3/4)

エラー・メッセージ	概要
WHILE 文の中に構文エラーがある SYNTAX error for WHILE	WHILE の次の式の形が不適切である。 WHILE 文中で、文の並びが不適切である。
名札が不適切である illegal label	WHILE 文でないもの、〔 〕に名札がついている。名札〔 〕が正しく書かれていない。 〔 〕は名札としては不適切である。
名札が 2 回以上定義されている redefined label	名札〔 〕が 2 回以上定義されている。
名札が未定義である undefined label	名札〔 〕が未定義である。
EXIT 文のあとに名札でない物が記述されている needs label after EXIT	EXIT の "(" のあとに名札でない〔 〕がある。
EXIT 文の名札が正しくない illegal label	EXIT の名札〔 〕が正しく書かれていない。
CASE 文で、条件が不適切である illegal use for CASE	CASE 文で、条件〔' '〕が不適切である。
CASE 文の条件式が不適切である SYNTAX error for CASE	CASE 文の条件式が不適切である。OF がない。CASE 文で条件が "'" で始まっていない。
RETURN 文で、式が不適切である illegal RETURN format	RETURN 文で、式が不適切である。")" がない。
MAIN 関数がない MAIN not found	主プログラム関数 MAIN( ) がない。

(注) 文中の〔 〕の内の文字はディスプレイ上に表示されます。

表 3-1 エラー・メッセージ (4/4)

エラー・メッセージ	概要
変数及び配列が、許容量を超えた variable & array memory overflow	変数および配列が許容量を超えてしまった。
プログラムが、許容量を超えた program memory overflow	プログラム容量がメモリ容量を超えてしまった。
テンポラリ容量が、許容量を超えた temporary memory overflow	テンポラリ容量が、メモリ容量を超えてしまった。
PRINT 文の容量が、許容量を超えた print statement area overflow	PRINT 文の容量が、メモリ容量を超えてしまった。
引数が見つからない parameter not found	引数が見つからない。
RETURN 文の数が正しくない illegal return	RETURN 文の数が不適切である。
PRINT 文の長さが、255 文字を超えた print string length more than 255	1つのPRINT 文の文字列の長さが255文字を超えてしまった。
オブジェクトの書き込みに失敗した Fail to create .OBJ File	オブジェクト・プログラムの書き込みに失敗した。
コメント処理中に EOF を検出した unexpected EOF found in comment	コメント処理中に */ (コメントの終わり) が、ファイルの最後まで見つからなかった。
文字列処理中に改行または EOF を検出した unexpected EOF or LF found in string	文字列処理中に、改行またはファイルの最後が検出された。

3.4 プログラムの実行

プログラムを実行するためには、あらかじめオブジェクト・プログラムまたはメッセージ付きオブジェクト・プログラムを作成しておく必要があります。

コンパイルしたオブジェクト･プログラムを LPAD シミュレータ上で実行するか LPAB シミュレータ上で実行するかは､宣言文“CHANNEL”で D を指定したか B を指定したかによって決まります。

以下にオブジェクト・プログラムを実行する場合について説明します。

① D チャネル・シミュレーションの場合は“LAPD シミュレータ x”を、B チャネル・シミュレーションの場合は“LAPB シミュレータ x”を選択します。
(x : 本器に搭載されているシミュレーション機能モジュール数により表示される数値が異なり、1 ~ 4 までが表示されます。)
選択は **F1** を押して表示される機能モジュール選択メニューで行うか、すでにアプリケーションをロードしてある場合は、**ALT** ＋ファンクション・キーで行います。

② **F4** を押して、Load/Save メニューを表示させ、**"Operation"** を **"Load"** に設定します。
オブジェクト･プログラムを指定するときは､**"Data Type"** を **"Message & Object File"** に設定します (図 3-4)。
また､メッセージ付きオブジェクト･プログラムを指定するときは､**"Message+Object File"** に設定します (図 3-5)。
ファイル名の指定方法は「D5115 取扱説明書 3.7.8(3) ファイル名入力領域」を参照して下さい。

③ オブジェクト・プログラムの実行にメッセージ・ファイルが必要な場合、メッセージ・ファイル指定領域にメッセージ・ファイルを指定します。

④ ***OK*** にカーソルを移動し、**Enter** を押します。
指定したファイルが、それぞれの表示領域に表示されます。

⑤ **F8** (シミュレーション起動キー) を押すとオブジェクト・プログラムが実行されます。このとき､機能モジュールアイコン表示領域 (画面最下段) の「LAPDx Simulator」アイコンまたは「LAPBx Simulator」アイコンの表示が
STOP → RUN
と変化します。オブジェクト・プログラムが実行されている間はアイコンが RUN 表示となります。

⑥ オブジェクト・プログラムが自動的に停止するようになっている場合、オブジェクト・プログラムが自動的に停止します。また、**F10** (モニタ / シミュレータ機能停止キー) を押すことにより、実行中のプログラムを停止することができます。
このとき、機能モジュールアイコン表示領域 (画面最下段) のアイコン表示が
RUN → STOP
と変化します。

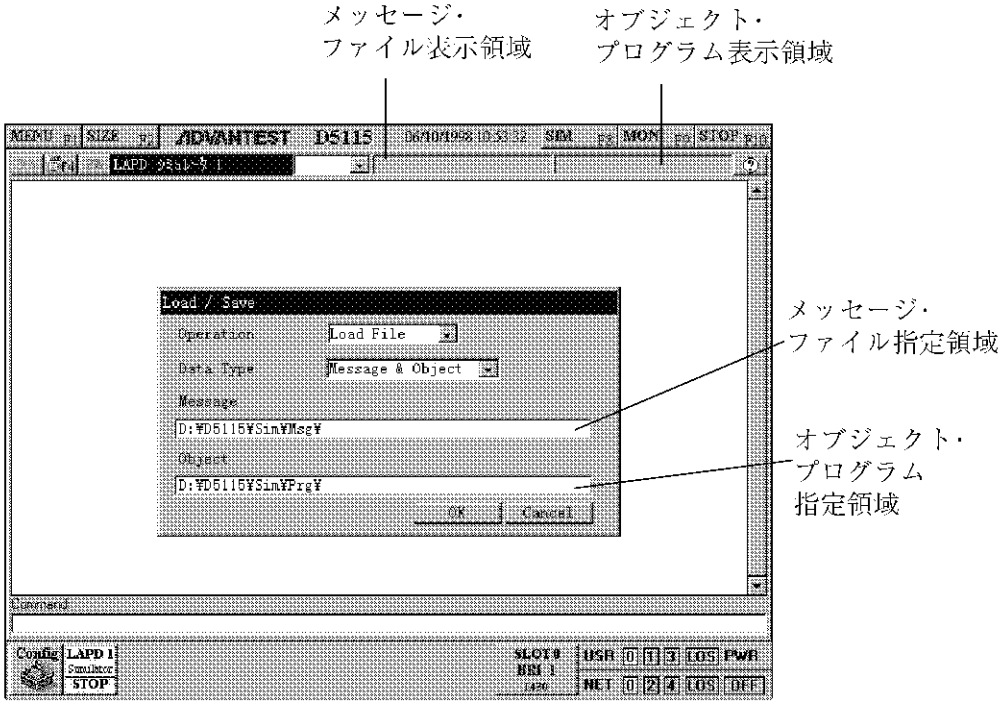


図 3-4 Load/Save メニュー (Data Type :Message&Object の場合)

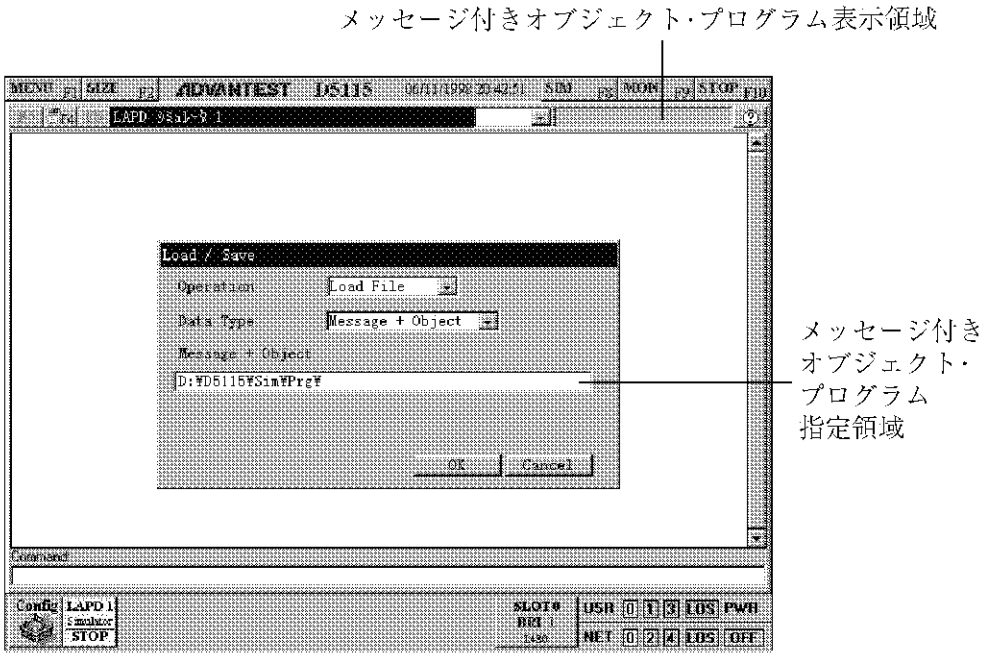


図 3-5 Load/Save メニュー (Data Type :Message+Object の場合)

3.5 プログラムの実行における注意事項

3.5.1 D5112 シリーズのプログラム・ファイルの互換性

本器で PSL51 を使用する際、以下の制限があります。

項目	D5112 での機能	D5115 での制限事項
BEEP( )	ビープ音を鳴らす。	本機能は、未対応です。 コンパイル実行時はエラーとなりませんが、シミュレーション実行時にこの命令は実行されません。
SOUND_ON( )	ヘッドホン・ブザーより音を発生させる。	本機能は、引数 TYPE = 20(B チャネル音声) のみ対応しています。 それ以外の引数は、引数エラー (関数値 -60) となります。

3.5.2 プログラム作成上の注意

以下に示すようなプログラムを実行すると、シミュレーション機能が正常に実行されなくなることがあります。動作を確認しながら、プログラムを作成して下さい。

① 配列を使用する際、宣言文により確保した領域外へ不正なアクセスを行うと、正常な動作をしないことがあります。
〔例〕

```
ARRAY A[100]
FUNC MAIN( )
        C = 10000
        A[C] = 1
RETURN
```

② 関数の再帰呼出しなどによりメモリを消費するプログラムを実行すると、正常な動作をしないことがあります
〔例〕

```
FUNC MAIN( )
        FUNC1( )
RETURN

FUNC FUNC1( )
        FUNC1( )
RETURN
```

3.5.3 シミュレーション実行時のワーニング表示

シミュレーションを実行するために **F8** を押したとき、ワーニングが表示されることがあります。表示およびその内容は以下のようになっています。

ワーニング表示	説明
シミュレーション機能は、ビジー状態です。	すでにシミュレーションが実行されている状態で、**F8** を押したときに表示されます。
オブジェクト (.OBJ) ファイルを設定し直して下さい。	起動しているシミュレーション・モジュールに、オブジェクト・ファイルを指定していないモジュールがある場合に表示されます。 オブジェクト・ファイルを指定する、または、使用しないシミュレーション機能モジュールを終了してから、シミュレーションを実行して下さい。
シミュレーション機能は現在使用できない状態です。	**F8** を押したときこのワーニングが頻繁に表示される場合、プログラムの記述または製品の不具合が原因と考えられます。 本器を再起動して、プログラムを確認して下さい。 (「3.5.2 プログラム作成上の注意」を参照) 不明な点がございましたら、最寄りのアドバンテスト営業所または代理店までお問い合わせ下さい。所在地および電話番号は巻末に記載してあります。
レイヤ 1 の指定が正しくありません。	指定されたレイヤ 1 インタフェースでシミュレーションが実行できない場合に表示されます。 例えば、同じレイヤ 1 インタフェースに対して、TE モードのシミュレーションと NT モードのシミュレーションを同時に実行した場合等に表示されます。 レイヤ 1 インタフェースの指定を確認する、または、シミュレーション・プログラムの宣言文 "SIMMODE" を確認して下さい。

4. メッセージ・ビルダ

4.1 メッセージ・ビルダの概略

送信するフレームのデータはメッセージ・ビルダにより作成します。以下に、シミュレーション用メッセージの作成について説明します。

① 機能モジュールを選択します。

D チャネル・シミュレーションの場合は“LAPD メッセージ・ビルダ”を、B チャネル・シミュレーションの場合は“LAPB メッセージ・ビルダ”を選択します。選択は、**F1** を押して表示される機能モジュール選択メニューで行います。すでにアプリケーションをロードしてある場合は、**ALT** ＋ファンクション・キーで行います。

② メッセージは、No.1 ~ No.100 までの 100 種類を作成できます。

メッセージ・ビルダを使用する際に注意しなければならないことがあります。まずメッセージ・ビルダで作成した LAYER2 の情報は、レイヤ 2 自動モードでは無視されます。また、トランスペアレント・モードで使用される N(S) 、N(R) はプログラムで設定された V(S)、V(R) 値を参照し、自動設定されます。P/F ビット値は、SENDF 関数において引数により指定した値が有効です。

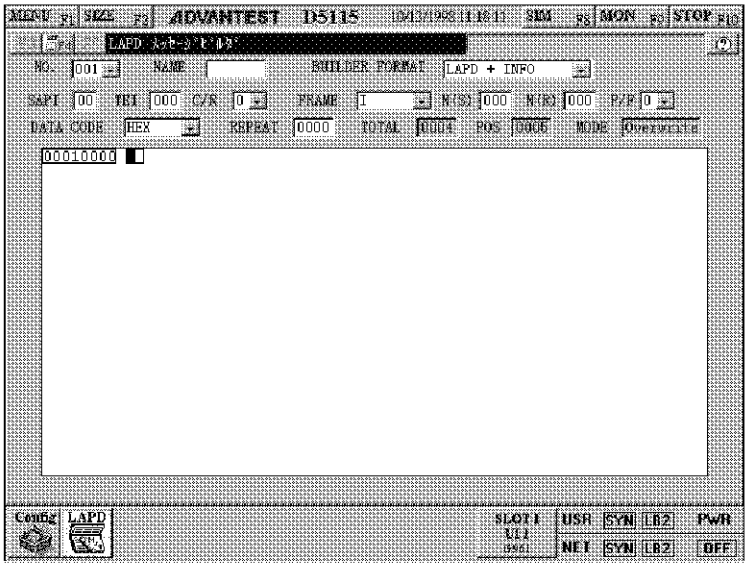
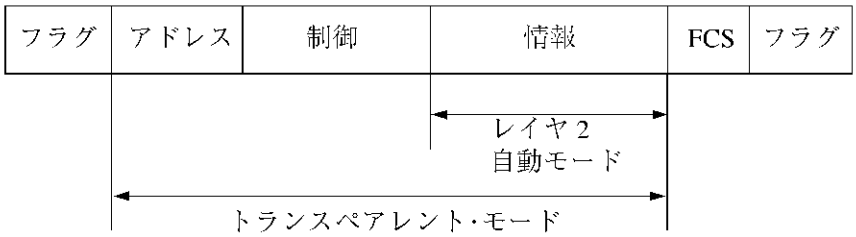


図 4-1 メッセージの作成 (メッセージ・ビルダ)

4.2 LAPD 用メッセージ・ビルダ

メッセージ・ビルダにより下図に示す部分のフレームを作成できます。



メッセージ・ビルダにより、NO.1 ~ NO.100 の 100 種類のフレームを作成できます。各フレームの最大長は、512 オクテットです。
作成したフレームには名前をつける必要があります。名前 (6 文字まで入力可能) を付けることにより作成したレームが有効になります。アドレスと制御フィールドは、SAPI、TEI 、C/R ビット、フレーム・タイプが設定可能です。N(R)、N(S) 値は、シミュレーション関数 (INCVS, INCVR, SETVS, SETVR) により設定された V(S)、V(R) 値を参照して自動的に付加されます。また、P/F ビットは、SENDF 関数実行時に引数で与えられます。
メッセージ作成には、7 通りのフォーマットがあります。
作成フォーマットの切り換えは、"BUILDER FORMAT" のポップアップ・メニューで行います。"BUILDER FORMAT" の位置で **Spc** または、**Enter** を押すと以下の 7 種類のフォーマットが選択できるポップアップ・メニューが表示されます。

(1) LAYER2 INFO : レイヤ 2 レベルからの直接入力
(2) LAPD + INFO : レイヤ 2 は LAPD メニューによる入力でレイヤ 3 は直接入力
(3) LAPD + Q.931 : レイヤ 2 は LAPD メニューによる入力でレイヤ 3 は Q.931 メニューによる入力
(4) LAPD + X.25 : レイヤ 2 は LAPD メニューによる入力でレイヤ 3 は X.25 メニューによる入力
(5) LAYER3 INFO : レイヤ 3 は直接入力
(6) Q.931 : レイヤ 3 は Q.931 メニューによる入力
(7) X.25 : レイヤ 3 は X.25 メニューによる入力

希望するフォーマットの位置に ↑、↓ でカーソルを移動させます。**Enter** を押すことにより、作成フォーマットを変更することができます。
レイヤ 2 自動モードでシミュレーションを行う場合、メッセージのレイヤ 2 部分は自動的に付加されます。そこで、この部分を画面から消去したものが、(5) ~ (7) のフォーマットです。
内部的には、(3) の "LAPD + Q.931" と (6) の "Q.931" のフォーマットは、同じ動作をするので、レイヤ 2 自動モードで使用するメッセージを作成するときは、どちらのフォーマットを使用しても構いません。(4) の "LAPD + X.25" と (7) の "X.25" についても同様のことが言えます。

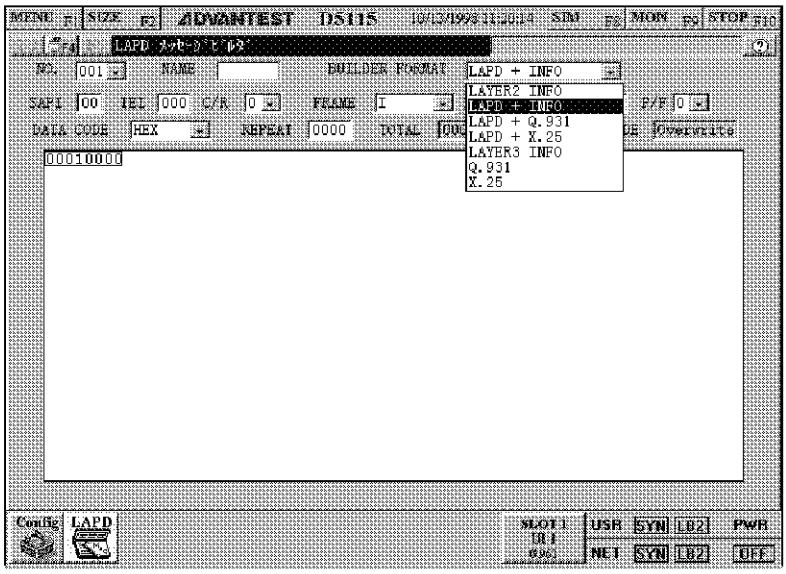
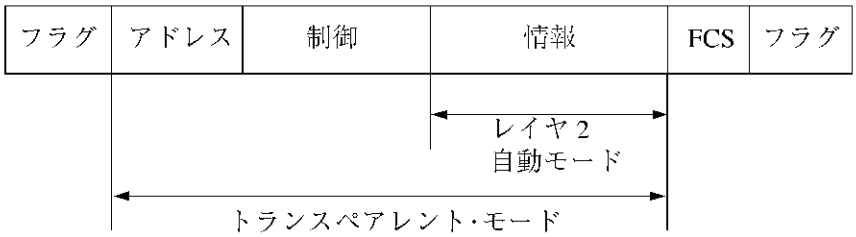


図 4-2 D チャネル用メッセージ・ビルダのフォーマット選択

4.3 LAPB 用メッセージ・ビルダ

メッセージ・ビルダにより下図に示す部分のフレームを作成できます。



メッセージ・ビルダにより、NO.1 ~ NO.100 の 100 種類のフレームを作成できます。各フレームの最大長は、4200 オクテットです。
作成したフレームには名前をつける必要があります。名前 (6 文字まで入力可能) を付けることにより作成したフレームが有効になります。アドレスと制御フィールド内のフレーム・タイプが設定可能です。N(R)、N(S) 値は、シミュレーション関数 (INCVS, INCVR, SETVS, SETVR) により設定された V(R)、V(S) 値を参照して SENDF 関数使用時に自動的に付加されます。また、P/F ビットは、SENDF 関数の引数で与えられます。
メッセージ作成には、7 通りのフォーマットがあります。BUILDER FORMAT の位置で **Spc** または、**Enter** を押すと以下の 7 種類のフォーマットが選択できるポップアップ・メニューが表示されます。7 種類のフォーマットは以下の通りです。

(1) LAYER2 INFO: レイヤ 2 レベルからの直接入力

(2) LAPB8 + INFO: レイヤ 2 は LAPB8 メニューによる入力でレイヤ 3 は直接入力

(3) LAPB8 + X.25: レイヤ 2 は LAPB8 メニューによる入力でレイヤ 3 は X.25 メニューによる入力

(4) LAPB128 + INFO: レイヤ 2 は LAPB128 メニューによる入力でレイヤ 3 は直接入力

(5) LAPB128 + X.25: レイヤ 2 は LAPB128 メニューによる入力でレイヤ 3 は X.25 による入力

(6) LAYER3 INFO: レイヤ 3 は直接入力

(7) X.25: レイヤ 3 は X.25 メニューによる入力

希望するフォーマットの位置に↑、↓でカーソルを移動させます。**Enter** を押すことにより、作成フォーマットを変更することができます。
レイヤ 2 自動モードでシミュレーションを行う場合、メッセージのレイヤ 2 部分は自動的に付加されます。そこで、この部分を画面から消去したものが、(6)、(7) のフォーマットです。
(3) の "LAPB8 + X.25" や (5) の "LAPB128 + X.25" のフォーマットで作成したメッセージと (7) の "X.25" フォーマットで作成したメッセージはレイヤ 2 自動モードで使用する場合は同じです。レイヤ 2 自動モードのときのレイヤ 2 モジュロの選択は、宣言文で行います。

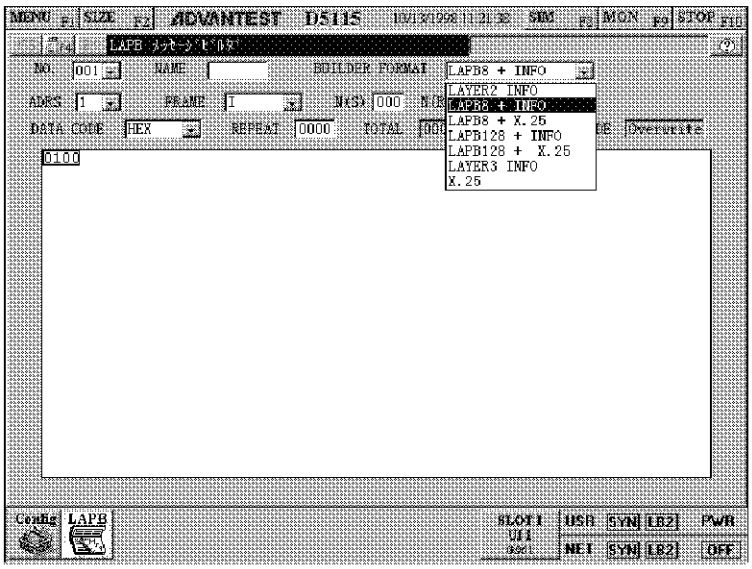


図 4-3 B チャネル用メッセージ・ビルダのフォーマット選択

4.4 メッセージ・ビルダの操作方法

(1) 直接入力部分

メニューを使用しないでメッセージを直接入力する場合について説明します。

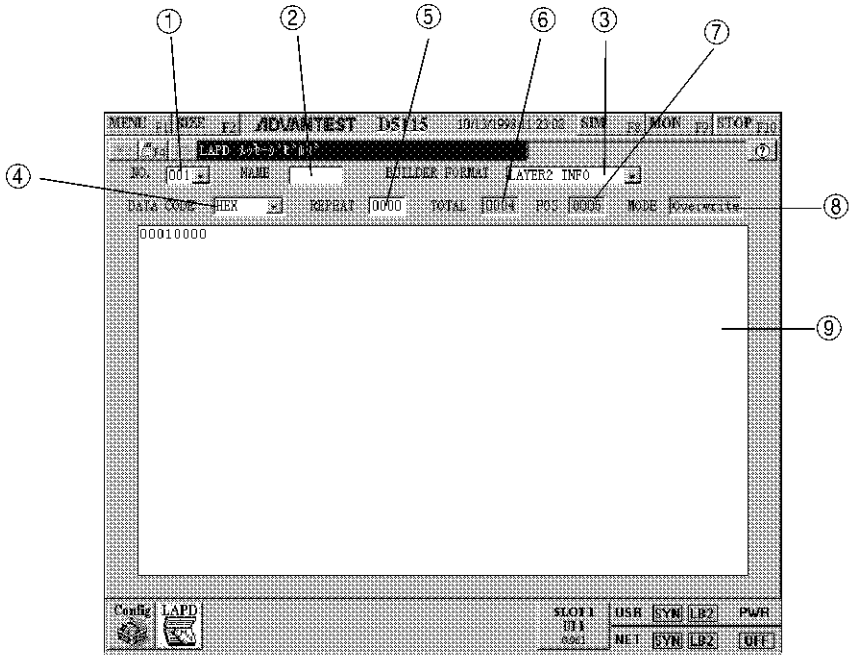


図 4-4 データの直接入力

上記の画面についての詳細を以下に示します。

① NO.

フレーム番号を表示します。NO. 1～NO. 100 までのメッセージを作成できます。この設定は NO. の位置で **Spc** または、**Enter** を押し、ポップアップ・メニューを表示させ希望の番号を選択するか、**Page Up**、**Page Down** で希望の数字に変更することができます。

② NAME

フレーム名を入力します。最大 6 文字まで入力可能です。
フレーム名として使用できる文字は、英数字および @ # _ です。
メッセージ・ファイル内に同じフレーム名が存在する場合、シミュレーション実行時は、NO. の小さいフレームが使用されます。

③ BUILDER FORMAT

メッセージ・ビルダの作成フォーマットを指定します。カーソルを "BUILDER FORMAT" の位置に移動させて、**Spc** または **Enter** を押します。以下のポップアップ・メニューが表示されますので、↑、↓で選択し、**Spc** または **Enter** で決定します。

i) D チャネルの場合

LAYER2 INFO
LAPD + INFO
LAPD + Q.931
LAPD + X.25
LAYER3 INFO
Q.931
X.25

ii) B チャネルの場合

LAYER2 INFO
LAPB8 + INFO
LAPB8 + INFO
LAPB128 + INFO
LAPB128 + X.25
LAYER3 INFO
X.25

"BUILDER FORMAT" を切り換えると、データ直接入力領域の中に下図のような枠が表示されることがあります。これはメニューにより作成する領域であることを示しています。⑨の領域では、この枠の中の数値を変えることはできません。この枠内のメッセージはメニューの値を変えると自動的に作り直されます。

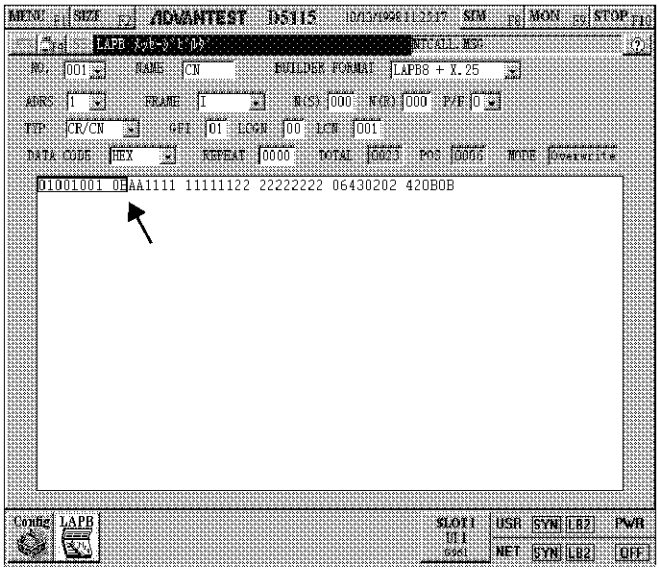


図 4-5 データ直接入力領域の枠

また、"BUILDER FORMAT" で "LAYER3 INFO" や " Q.931", "X.25" を選択したとき、下図のような網掛けが現れます。これは、レイヤ 2 自動モードでフレームを送出するときに使われない領域であることを示しています。

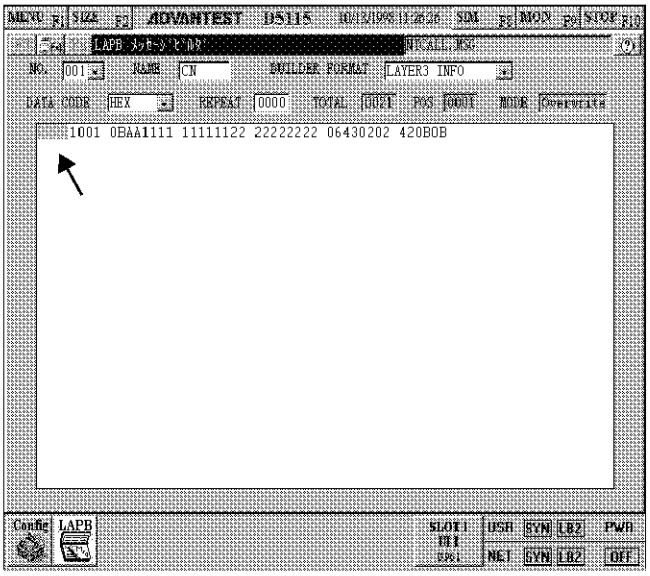


図 4-6 レイヤ 2 領域の網掛け

④ DATA CODE

⑨の領域に送信データを直接作成しますが、そのときの入力モードを指定します。入力モードは、HEX 、ASCII 、JIS8、EBCDIC の 4 種類です。入力モードの設定は以下の手順で行います。
カーソルを "DATACODE" の位置に移動させます。ここで、**Spc** または **Enter** を押すと下のポップアップ・メニューが表示されます。

HEX
ASCII
JIS8
EBCDIC

↑、↓で、自分の希望する入力モードの位置にカーソルを移動させ、**Spc** または **Enter** を押します。ポップアップ・メニューが閉じて、④ の位置には設定された入力モードが表示されます。

⑤ REPEAT

REPEAT の領域にリピート回数を設定します。設定された REPEAT 領域をリピート回数だけ繰り返します。0 が設定されるとリピートは行いません。0 以外が設定された場合は、⑨の領域に設定されたREPEAT領域のフォントの色が青色に変わります。

⑥ TOTAL

作成したデータからフレームを組み立てた場合の総オクテット数を表示します。ただし、フラグと FCS はカウントされません。また、"BUILDER FORMAT" で "LAYER3 INFO" 、"Q.931" または "X.25" が選択されている場合は、レイヤ 2 部分 (▒▒ 部分) もカウントされません。

⑦ POS

カーソル位置を先頭からのオクテット値で表示します。
レイヤ 2 からメッセージを作成した場合、レイヤ 2 の先頭から数えたオクテット値を表示します。また、レイヤ 3 からメッセージを作成した場合、レイヤ 3 の先頭から数えたオクテット値を表示します。

⑧ MODE

データの入力モードが何に設定されているか表示します。入力モードの選択は、⑨の領域で **Ctrl-I** (**Ctrl** と **I** を同時に押します。) を行うことによりできます。
Overwright ： 置換モード
Insert ： 挿入モード

⑨ データの直接入力領域

データを直接入力します。入力モードには、HEX 、ASCII 、JIS8、EBCDIC の 4 種類があります。この設定は④の領域で行うことができます。また、入力データはリピート領域を設定すれば、その部分のデータがリピート回数だけ繰り返して送信されます。
⑨の領域では、以下のキーを使用することができます。

リピート位置のリセット: **Ctrl-R**
リピート開始位置指定: **Ctrl-B**
リピート終了位置指定: **Ctrl-E**
入力モード (Overwright ⇔ Insert) の切り換え: **Ctrl-I**
マークのセット: **Ctrl-Spc**
指定領域の削除: **Ctrl-W**
カーソルの位置から最後まで削除: **Ctrl-K**
削除バッファの内容を挿入: **Ctrl-Y**
データを初期化: **Ctrl-C**
情報要素識別子の前方検索 (Q.931 のときのみ有効): **Ctrl-P**
情報要素識別子の後方検索 (Q.931 のときのみ有効): **Ctrl-N**

注意 "**Ctrl-**" はコントロール・キーを押しながら "**Ctrl-**" 以降のキーを押すという意味です。
"**Spc**" はスペース・キーです。

(2) LAPD メニュー

メッセージ・データを LAPD メニューを使用して作成する場合について説明します。

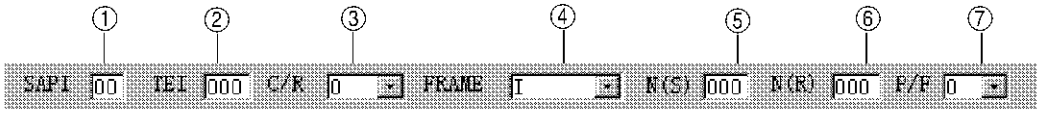


図 4-7 LAPD メニュー

① SAPI

SAPI 値を入力します。0～63 の値が設定可能です。

② TEI

TEI 値を入力します。0～127 の値が設定可能です。

③ C/R

コマンド / レスポンス値を設定します。カーソルを "C/R" の位置に移動させて、**Spc** または、**Enter** を押します。そして、以下のようなポップアップ・メニューを表示させます。

0
1

希望のビットを選択し、**Spc** または、**Enter** を押すと新たに設定されます。

④ FRAME

フレームの種類を設定します。カーソルを "FRAME" の位置に移動させて、**Spc** または、**Enter** を押します。そして、以下のようなポップアップ・メニューを表示させます。

I	RR	RNR
REJ	SABME	DISC
UI	XID	DM
UA	FRMR	

希望のフレームを選択し、**Spc** または、**Enter** を押すと新たに設定されます。

⑤ N (S)

送信シーケンス番号を入力します。0～127 の値が設定可能です。

⑥ N (R)

受信シーケンス番号を入力します。0～127 の値が設定可能です。

⑦ P/F

ポール / ファイナル・ビットを設定します。
カーソルを "**P/F**" の位置に移動させて、**Spc** または **Enter** を押します。そして以下のようなポップアップ・メニューを表示させます。

0
1

希望のビットを選択し、**Spc** または **Enter** を押すと新たに設定されます。

(3) LAPB メニュー

メッセージ・データを LAPB メニューを使用して作成する場合について説明します。

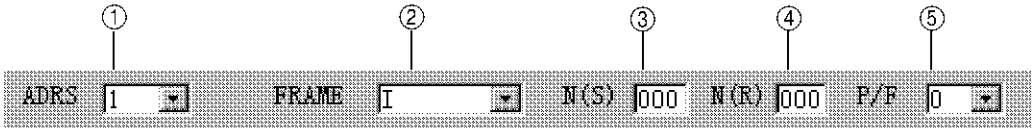


図 4-8 LAPB メニュー

① ADRS

アドレスを設定します。
カーソルを "**ADRS**" の位置に移動させて、**Spc** または **Enter** を押すと以下のポップアップ・メニューが表示されます。

1
3

↑、↓で希望の値を選択し、**Spc** または **Enter** で決定します。

② FRAME

フレームの種類を設定します。
カーソルを "**FRAME**" の位置に移動させて、**Spc** または **Enter** を押すと以下のポップアップ・メニューが表示されます。

i) LAPB8 の場合

I	RR	RNR
REJ	SABM	DISC
DM	UA	FRMR

ii) LAPB128 の場合

I	RR	RNR
REJ	SABME	DISC
DM	UA	FRMR

↑、↓、→、← で希望するフレームを選択し、**Spc** または **Enter** で決定します。

③ N (S)

送信シーケンス番号を入力します。
モジュロ 8 のとき、0～7 の値が設定可能です。
モジュロ 128 のとき、0～127 の値が設定可能です。

④ N (R)

受信シーケンス番号を入力します。
モジュロ 8 のとき、0～7 の値が設定可能です。
モジュロ 128 のとき、0～127 の値が設定可能です。

⑤ P/F

ポール / ファイナル・ビットを設定します。
カーソルを "**P/F**" の位置に移動させて、**Spc** または **Enter** を押します。そして以下のようなポップアップ・メニューを表示させます。

0
1

希望のビットを選択し、**Spc** または **Enter** を押すと新たに設定されます。

(4) Q.931 メニュー

メッセージ・データを "Q.931" メニューを使用して作成する場合について説明します。

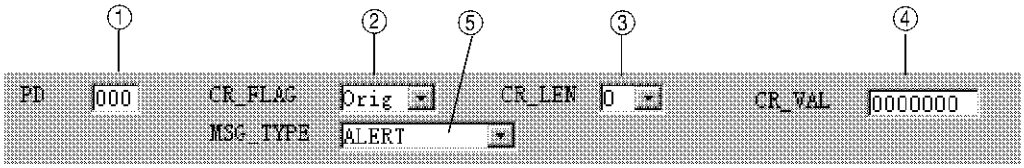


図 4-9 Q.931 メニュー

① PD

プロトコル識別子を設定します。数字を入力して下さい。"Q.931" プロトコルを使用する場合“8”に設定して下さい。

② CR_FLAG

呼番号フラグを設定します。
カーソルを "**CR_FLAG**" の位置に移動し、**Spc** または **Enter** を押すと、以下のポップアップ・メニューが表示されます。

Orig
Dest

↑、↓で希望のフラグを選択し、**Spc** または **Enter** を押すと、新たに設定されます。

③ CR_LEN

呼番号長を設定します。
カーソルを "**CR_LEN**" の位置に移動し **Spc** または **Enter** を押すと、以下のポップアップ・メニューが表示されます。

0	2
1	3

↑、↓、→、← で希望の呼番号長を選択し、**Spc** または **Enter** で設定して下さい。

④ CR_VAL

呼番号を設定します。
設定できる呼番号は ③の呼番号長に依存します。
③の呼番号長に "0" を選ぶと、0 以外は入力できません。

⑤ MSG_TYPE

メッセージ・タイプを設定します。カーソルを "**MSG_TYPE**" の位置に移動し、**Spc** または **Enter** を押すと、ポップアップ・メニューが表示されます。
希望のメッセージの位置にカーソルを移動させ、**Spc** または **Enter** を押すと、そのメッセージが選択されます。

(5) X.25 メニュー

メッセージ・データを "X.25" メニューを使用して作成する場合について説明します。

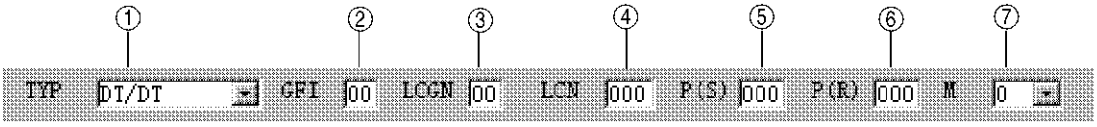


図 4-10 X.25 メニュー

① TYP

パケットの種類を設定します。
カーソルを "**TYP**" の位置に移動し、**Spc** または **Enter** を押すと、ポップアップ・メニューが表示されます。
希望のパケット名の位置にカーソルを移動させ、**Spc** または **Enter** を押すと、そのパケットが選択されます。

② GFI

ゼネラル・フォーマット識別子を設定します。

③ LCGN

論理チャネル・グループ番号を設定します。0～15 の値が設定可能です。

④ LCN

論理チャネル番号を設定します。0～255 の値が設定可能です。

⑤ P (S)

送信順序番号を入力します。
GFI ビットが「xx01」のとき、0～7 の値が設定可能です。
GFI ビットが「xx10」のとき、0～127 の値が設定可能です。

⑥ P (R)

受信順序番号を入力します。
GFI ビットが「xx01」のとき、0～7 の値が設定可能です。
GFI ビットが「xx10」のとき、0～127 の値が設定可能です。

⑦ M

モア・データ表示値を設定します。
カーソルを "M" の位置に移動し **Spc** または **Enter** を押すと、以下のポップアップ・メニューが表示されます。

0
1

↑、↓で希望の M ビット値を選択し、**Spc** または **Enter** を押すと、新たに設定されます。
このポップアップ・メニューはパケットの種類が "DT" のときのみ表示されます。

5. 保存 / 読み出し (セーブ / ロード) ・メニュー

5.1 メッセージ・データの保存 / 読み出し

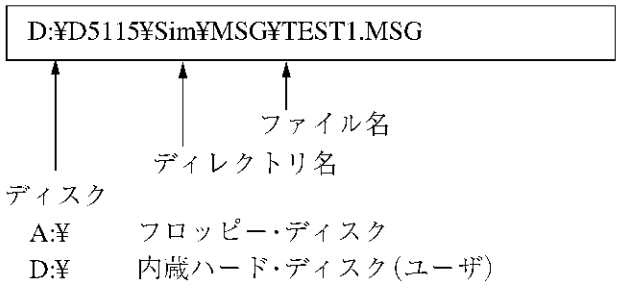
メッセージ・データを使用する命令 (SENDF、SENDI 、SENDUI...) を含んだプログラムを実行する場合、メッセージ・データが必要です。
メッセージ・データは、メッセージ・ビルダにより再びデータを作成するか、メッセージ・データをディスクからロードしてくることにより、用意できます。
ここでは、メッセージ・データのセーブ / ロードの手順について示します。

① シミュレーション・アプリケーションを選択します。
D チャネル用メッセージの場合は "**LAPD** メッセージ・ビルダ " を、B チャネル用メッセージは "**LAPB** メッセージ・ビルダ " を選択します。選択は **F1** を押して表示される機能モジュール選択メニューで行うか、すでにアプリケーションをロードしてある場合は、**ALT** ＋ファンクション・キーで行います。

② **F4** を押すと、セーブ / ロード画面が表示されます。

③ "**Operation**" の位置にカーソルを移動させ、**Enter** を押して下さい。セーブしたい場合は "**Save**" の位置に、ロードしたい場合は "**Load**" の位置にカーソルを移動させ、**Enter** を押して下さい。

④ ファイル名指定領域にカーソルを移動させ、セーブまたはロードするファイル名を指定して下さい。ファイル名指定領域のフォーマットは以下のようになっています。



カーソルを ***OK*** に移動させ、**Enter** を押すと、メッセージ・データが保存 / 読み出しされます。

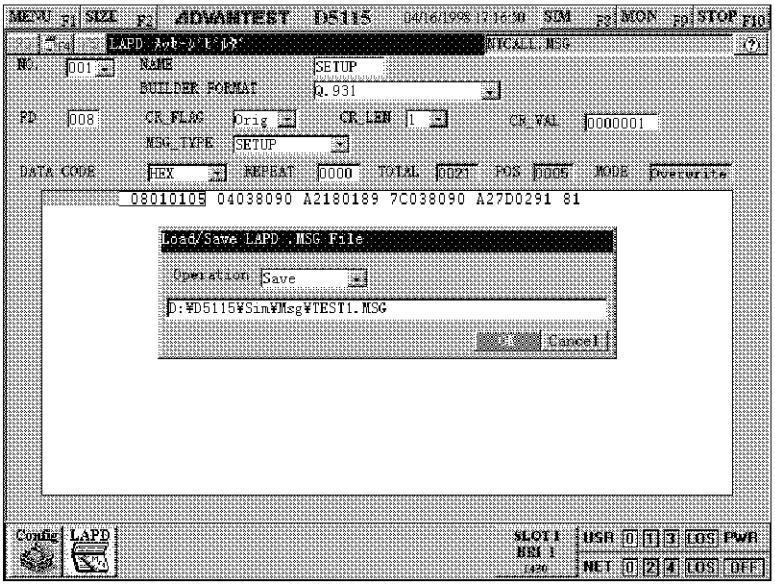


図 5-1 メッセージ・データのセーブ

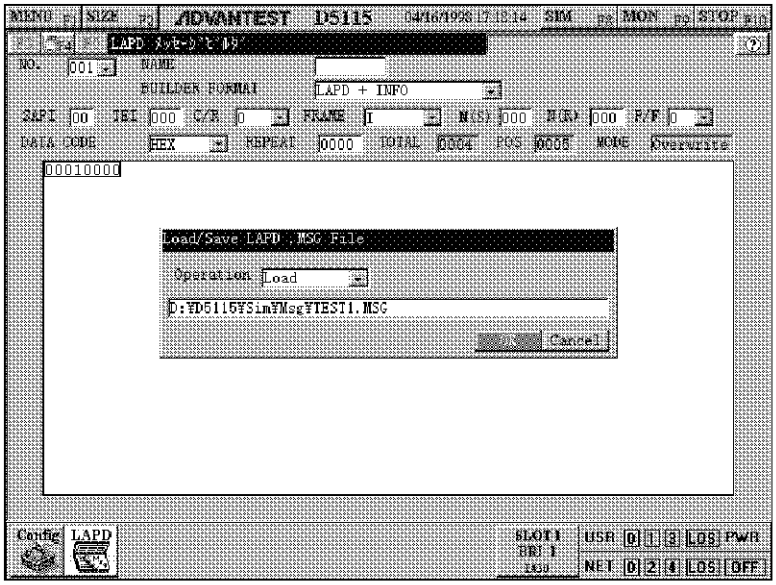


図 5-2 メッセージ・データのロード

5.2 メッセージ付きオブジェクト・プログラムの保存 / 読み出し

オブジェクト・プログラムとメッセージ・データを一つのファイルとして保存することができます。このとき、拡張子は ".EXC" を用います。
以下にメッセージ付きオブジェクト・プログラムの保存について説明します。

D チャネル・シミュレーションの場合は "LAPD シミュレータ x" を、B チャネル・シミュレーションの場合は "LAPB シミュレータ x" を選択します。(x : 本器に搭載されているシミュレーション機能モジュール数により表示される数値が異なり、1 ~ 4 までが表示されます。)
選択は **F1** を押して表示される機能モジュール選択メニューで行うか、すでにアプリケーションをロードしてある場合は、**ALT** ＋ファンクション・キーで行います。

(1) メッセージ付きオブジェクト・プログラムの保存

① 「3.2(3) オブジェクト・プログラムの指定」「3.2(4) メッセージ・ファイルの指定」を参照し、オブジェクト・プログラム表示領域、メッセージ・ファイル表示領域に、それぞれのファイルを設定します。

② **F4** を押して、Load/Save メニューを表示します。

③ **"Operation"** の位置にカーソルを移動し、**Enter** を押します。「**Save File**」を選択し、**Enter** を押します。**"Data Type"** が「**Message+Object**」に変更されます。

④ **"Message+Object"** の位置にカーソルを移動し、ファイル名を指定します。ファイル名の指定方法は「D5115 取扱説明書 3.7.8(3) ファイル名入力領域」を参照して下さい。

⑤ ***OK*** の位置にカーソルを移動し、**Enter** を押すと、メッセージ付きオブジェクト・プログラムが保存されます。

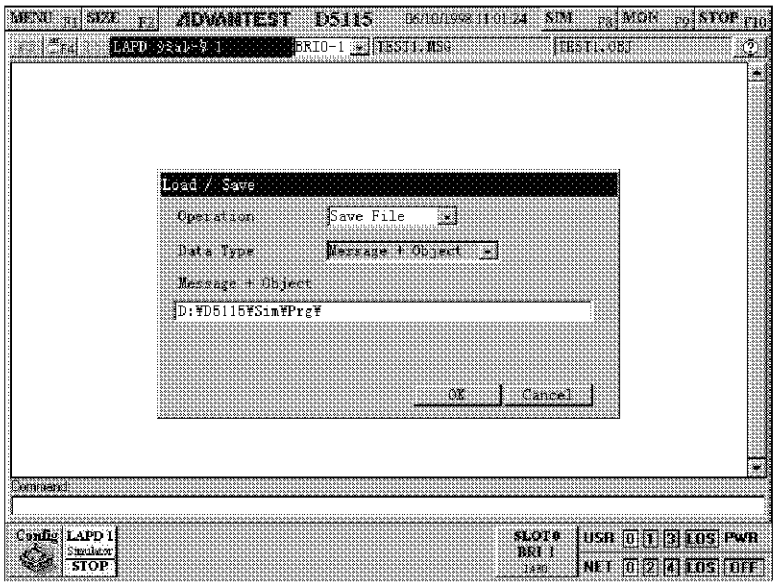


図 5-3 メッセージ付きオブジェクト・プログラムのセーブ

(2) メッセージ付きオブジェクト・プログラムの読み出し

① **F4** を押して、Load/Save メニューを表示します。

② **"Operation"** の位置にカーソルを移動し、**Enter** を押します。「**Load File**」を選択し、**Enter** を押します。

③ **"Data Type"** の位置にカーソルを移動し、**Enter** を押します。「**Message+Object**」を選択し、**Enter** を押します。

④ **"Message+Object"** の位置にカーソルを移動し、ファイル名を指定します。

⑤ ***OK*** の位置にカーソルを移動し、**Enter** を押すと、メッセージ付きオブジェクト・プログラムが読み出され、メッセージ付きオブジェクト・プログラム表示領域に表示されます。

6. シミュレーション言語 (PSL51) の構文について

6.1 ソース・プログラム

6.1.1 ソース・プログラムの構成

PSL51 のプログラムは、中心となる関数（メイン関数）が必要であり、MAIN という名前を付けます。

プログラムの実行は、MAIN 関数から開始します。この関数から、他の関数の呼び出し、または制御系ステートメントの使用によりプログラムの実行を制御します。プログラムの実行は、MAIN 関数の最後（RETURN 文）で終了します。

MAIN 関数から各関数を呼び出すと、MAIN 関数はその関数に制御を渡して、その関数の最初のステートメントから実行が開始されます。RETURN 文が実行されるか、または関数が最後まで実行されると、制御は MAIN 関数に戻されます。

関数は引数を持つことができます。ある関数が別の関数を呼び出すとき、呼び出される関数は、呼び出し側の関数から値を受け取ります。

シミュレーションの宣言文は、プログラムの先頭に記述する必要があります。シミュレーションの宣言文については「2.3 シミュレーションの宣言文」を参照して下さい。

関数外では、シミュレーションの宣言文のみ認識されます。

ソース・プログラムにおいて、予約語、シミュレーション関数は、大文字で記述して下さい。シミュレーション関数については、7 章～ 9 章を参照して下さい。

6.1.2 トークン

PSL51 コンパイラが認識するソース・プログラムの基本要素をトークンと呼びます。トークンは、空白文字（スペース、タブ、ラインフィード）と、演算子などの他のトークンで区切られます。PSL51 コンパイラがトークンを解釈するとき正しく区切られていないと、期待した通りにトークンを解釈できないことがあります。コメント文も空白文字として扱われます。

PSL51 コンパイラはソース・プログラムを解析するとき、トークンの区切り以外に使われる空白文字を無視するので、空白文字を使ってプログラムを見やすくすることができます。また、コメント文も空白文字として扱われます。

例 1

A1 と A2 を足し算したものを C に代入する式

```
C = A1 + A2
```

を誤って以下のように記述したとします。

```
C = A1 + A␣ 2          （␣ は、スペースを示します。）
```

この式は、"C","=","A1","+","A","2" の 6 つのトークンがあるものとして認識され、正しく計算されません。

例 2

B と C を引き算したものを A に代入する式

A = B - C

を誤って以下のように記述したとします。

A = B - - C

この式において、２つ目の "-" は単項演算子の "-" と認識されるので、

A = B - （- C）

と解釈されます。したがって、B と C を足し算したものを A に代入する式と認識され、正しく計算されません。

6.1.3 ステートメント

PSL51 コンパイラは、ステートメントの区切りを示す特殊な記号をもちません。ソース・プログラムの先頭から１つずつトークンを解釈していき、文法に照らし合わせてどのようなステートメントになっているのかを認識します。

例

C = A + B␣D =

C * 5␣PRINT("C=%D / D=%D¥N", C, D)　（␣は、スペースを示します。）

この式は、以下のように３つのステートメントとして認識されます。

C = A + B

D = C * 5

PRINT("C=%D / D=%D¥N", C, D)

6.1.4 予約語

以下の語は予約語として使用されているため、変数、関数名、配列名として使用できません。

ARRAY	STEP	R_SHIFT
CASE	THEN	L_SHIFT
DO	TO	PRINT
ELSE	WHILE	INPUT
END	LAYER	RND
EXIT	SIMMODE	RANDOMIZE
FOR	PFEED	GET_TIME
FUNC	CHANNEL	VAL_TIME
IF	INTERFACE	MAIN
OF	ADDRESS	BEEP 注
RETURN	MODULO	

注　現バージョンでは未対応ですが、D5112 シリーズとの互換性を保つため予約語に含まれています。

6.2 PSL51 の構成要素

6.2.1 関数

(1) 関数 (FUNCtion)

説明

プログラムは複数の関数と呼ばれる集合体から構成されています。
関数は、次のような構造をしていなくてはなりません。

```
FUNC   関数名 (引数 1 、引数 2 、… 引数 N )
  関数内での処理
RETURN( 戻り値 )
```

また、プログラムは必ず MAIN( ) という名前の関数から実行されるので、MAIN( ) という関数は必ず作成しなければなりません。
関数名は、英文字で始まり、英文字、数字または '_' で構成されます。認識される文字数は最大 10 文字です。
引数は、呼び出し側の関数から値を受け取ります。引数は、変数のみが可能であり、最大 15 個使用できます。全域的変数または配列と同じ名前の変数を引数としたとき、その関数内でその全域的変数または配列は参照できなくなります。
引数や戻り値は省略することもできます。引数を持たないとき、関数名の後の ( ) は省略することができます。

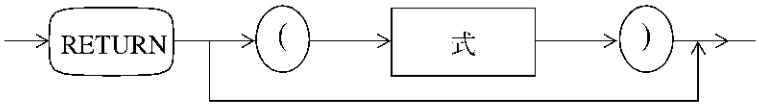
例

```
FUNC      MAIN( )
   A   =   123
   B   =   256
   C   =   SUB1(A,B)
RETURN
FUNC      SUB1(ARG1, ARG2)
 VAL   =   ARG1 + 3 * ARG2
RETURN(VAL )
```

6.2.2 RETURN 文

(1) RETURN 文

構文



説明　RETURN 文は、関数の実行が終了したとき、それを呼び出した関数に制御を戻すための文です。使用方法は、次表の通りです。

関数終了時 (戻り値がある場合)	RETURN(戻り値)
関数終了時 (戻り値がない場合)	RETURN

例

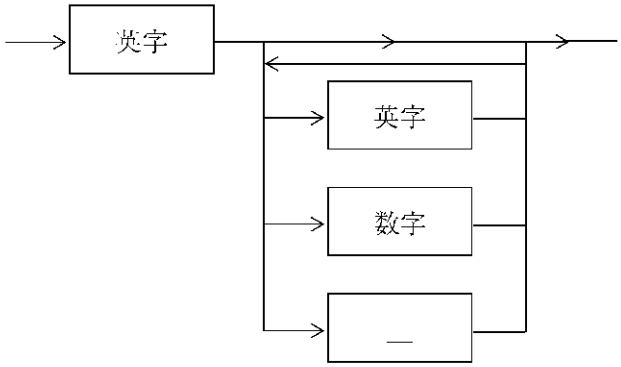
```
FUNC      FUNC1(ARG1, ARG2)
   .
   .
   .
   .
   .
RETURN   (XYZ+2)              /* 関数の終了(戻り値がある場合)*/

FUNC      FUNC1(ARG1, ARG2)
   .
   .
   .
RETURN                        /* 関数の終了(戻り値がない場合)*/
```

注　シミュレーション・プログラムの最後の RETURN 文の後ろには、改行を入れて下さい。改行が入っていない場合、ファイルの保存時に自動的に挿入されます。

6.2.3 変数・定数・配列

(1) 変数

構文

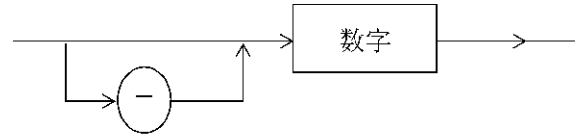


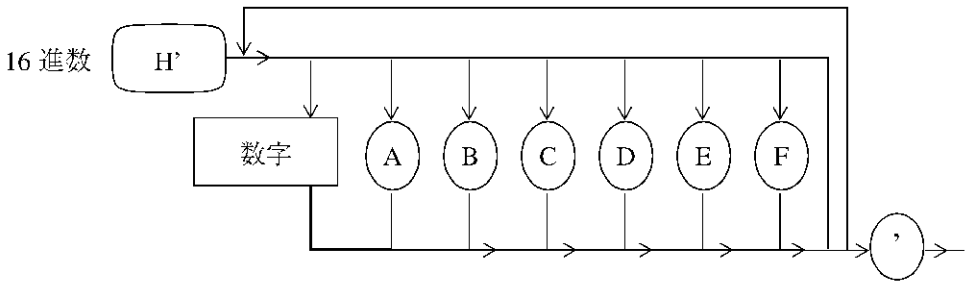
説明　変数は、英文字・数字または '_' で構成されます。また、認識される文字数は 10 文字までです。
変数は、すべて全域的変数として定義されます。

例　ABC
X1234
Z_XY
A123456789X　　(A123456789 と同じ)

(2) 定数

構文





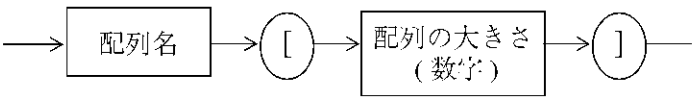
説明　定数には、10 進数と 16 進数があり、-2147483648 ～2147483647(符号付 32 ビット) の値をとります。また、16 進数は、H'F34'(10 進数では 3892) のように表現します。

例
```
ABC  =  32767
  X  =  H'F6D'
```

注意　PSL51 のデータ型は符号付 32 ビットなので -2147483648 ～ 2147483647 の範囲を超える値は、不定となります。

(3)　配列

構文



説明　配列は、1 次元配列のみが使用可能で、変数同様 -2147483648 ～2147483647 (符号付 32 ビット) の値をとります。また、配列はプログラムの先頭で宣言することにより使用可能となります。
(「2.2.1 PSL51 のプログラミング構造例」を参照して下さい。)
配列は、すべて全域的配列として定義されます。

次の例のように配列 (ARRAY) をプログラムの先頭で宣言します。また、例では配列 XYZ の初期化も以下のように行っています。

```
XYZ[0]  =  1
XYZ[1]  =  2
XYZ[2]  =  3
XYZ[3]  =  4
XYZ[4]  =  XYZ[5]  =  XYZ[6]  =  5
XYZ[7]  =  XYZ[8]  =  XYZ[9]  =  0
```

例
```
ARRAY  A[10] , B[20] , C[4]
ARRAY  XYZ[10] =  [1, 2, 3, 4, 5 [3], 0 [3] ]
```

注意
- 配列の添字は、0 から始まります。
- 配列の初期化を行う場合は、全要素の初期化を行って下さい。
- 全配列の大きさは、変数との総計が 32k バイト以下になるようにして下さい。
  32k バイト以上になると誤動作の原因となります。

6.2.4 演算子

演算子は、値の処理方法を指定するシンボルで、トークンとして扱われます。

また、以下の表 6-1 に示す優先順位と結合法則に従います。演算子の優先順位は、優先順位の異なる演算子があるときのみ意味があり、高い優先順位の演算子を含む式が先に解釈されます。同一行の演算子は同じ優先順位を持ち、結合規則に従って解釈されます。

表 6-1 演算子の優先順位と結合規則

優先順位	演 算 子	結 合 規 則
1	( )	左から右
2	* /	左から右
3	+ -	左から右
4	< <= > >= == !=	左から右
5	!	右から左
6	&	左から右
7	|	左から右
8	=	右から左

注　単項演算子の +、- は、二項演算子より高い優先順位をもちます。
結合規則は、実行される方向を示します。

演算式の例

a = b = c = 0	0 を c に代入し、結果を b に代入し、その結果を a に代入する。
4 * 8 - 9 / 3	4*8 がまず実行され、次に 9/3 が実行され、最後に引き算が行われる。
x = a == b	a と b を比較して、一致していれば x に 0 以外を、一致していなければ x に 0 を代入する。

PSL51 で使用可能な演算子を以下に説明します。

(1)　算術演算子

説明　演算子には、加算 (+)・減算 (-)・乗算 (*)・除算 (/) があります。

例　AB1 = X1[I] * 6 - XYZ + A / AB12

注意
- 除算では、結果は整数 (-2147483648 ～2147483647) になるように切り捨てられます。
  (例)
  123 / 12 = 10
- 除数が 0 のとき、値は (-1) になります。

(2) 関係演算子

説明 関係演算子には、>，<，>=，<=，==，!= の 6 種類があります。

	関係演算子	機能
(1)	>	より大きい
(2)	<	より小さい
(3)	>= (=>)	より大きいか、等しい
(4)	<= (=<)	より小さいか、等しい
(5)	==	等しい
(6)	!=	等しくない

例
```
IF  A==B   THEN  C=3 *A   ELSE  C=A  END
IF  X!=Y   THEN  Z=5 /W   ELSE  Z=W  END
```

(3) 論理演算子

説明 論理演算子には否定 (!) があります。
変数 AB1 を例をあげ、論理演算子を使用したときの値を以下に示します。

変数 AB1 の値	!AB1 の値
0	1
0 以外の値	0

例
```
IF(!AB1)  THEN  A = 1  ELSE  A = 0  END
```

(4) ビットごとの論理演算子

説明 ビットごとの論理演算子には、& (ビットごとの論理積) と | (ビットごとの論理和) があります。

例
```
A = 5
B = 6
C = A & B    /*C の値は 4*/
D = A | B    /*D の値は 7*/
```

6.2.5 標準関数

PSL51 で用意されている標準関数を以下に説明します。

(1) PRINT(" 書式 ", arg1, arg2, ...)

機能 1 引数で指定された書式にしたがって画面に表示します。" 書式 " 以外に引数がない場合は "　　" で囲まれた文字列をそのまま表示します。
" " で囲まれた文字列に、" および改行を含むことはできません。表示する文字列を改行させたいときは、¥N または ¥n を挿入することにより、改行されます。
arg1, arg2,…には、変数が与えられますが、その表示の指示は、書式内の % で始まる文字コードに従います。
文字コードの数と arg の数が異なる場合でもコンパイル・エラーとなりません。
文字コードの数が少ないとき、対応する文字コードを持たない arg は表示されません。また、文字コードの数が多いとき、対応する arg を持たない文字コードは、不定の値を表示します。
表示形式は、大きく分けて以下の 4 種類があります。（␣ は、スペースを示します。）
①②③の表示形式において、フィールド長 n が arg によって与えられた数字の桁数よりも小さいとき、フィールド長は自動的に拡張されます。

① %〔+〕〔0〕〔n〕D …10 進表示変換
　+　：数値が正の場合には“+”を先頭に表示します。
　0　：フィールドの先頭を 0 でうめます。
　n　：交換した文字をおさめるフィールド長を桁数で指定します。
　D　：10 進数に変換します。(小文字でも可能)
〔 〕で囲まれた指示は省略することもできます。
〔例〕 %D　→　123
　　%5D　→　␣␣123
　　%05D　→　00123
　　%+5D　→　␣+123

② %〔0〕〔n〕X … 16 進表示変換
　0　：フィールドの先頭を 0 でうめます。
　n　：交換した文字をおさめるフィールド長を桁数で指定します。
　X　：16 進数に変換します。(小文字でも可能)
〔 〕で囲まれた指示は省略することもできます。
〔例〕 %X　→　123
　　%5X　→　␣␣123
　　%05X　→　00123

③ %〔0〕〔n〕U
0 : フィールドの先頭を 0 でうめます。
n : 交換した文字をおさめるフィールド長を桁数で指定します。
U : 符号無し 10 進数に変換します。(小文字でも可能)
〔 〕で囲まれた指示は省略することもできます。
〔例〕%U → 123
%5U → ␣␣123
%05U → 00123

④ %〔n〕B
n : 交換した文字をおさめるフィールド長を桁数で指定します。
B : ブランクを表示します。(小文字でも可能)
〔例〕%5B → ␣␣␣␣␣

例

```
A = 123
B = -71
C = 51                                  改行指定
PRINT("SAMPLE ")
PRINT("PROGRAM !!¥N")
PRINT("A=%D(%3X) ",A,A )
PRINT("B=%5D¥N",B)
PRINT("C=%U B=%05D¥N",C,B)
```

結果

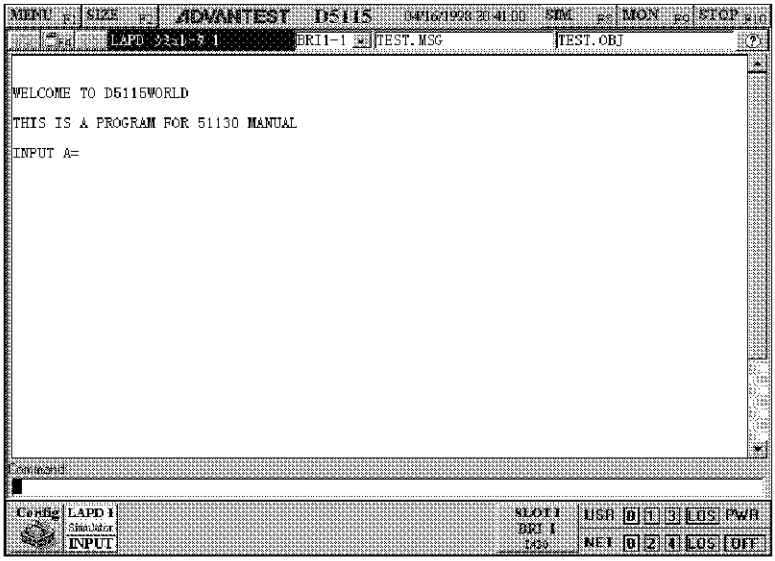
```
SAMPLE PROGRAM !!
A=123( 7B) B= -71
C=51 B=-0071
```

機能 2

PRINT 文を多数回実行するとシミュレーション実行結果表示画面から文字列がスクロール・アウトされてしまいます。が、本体内部のメモリに最大 700 行が保持されています。シミュレーションの実行停止後に、、キーにより前後スクロールして画面からスクロール・アウトした内容を確認することができます。700 行を超えて PRINT 文が実行されてしまうと、内部メモリ内に保持されないためシミュレーション停止後に内容確認することができません。
また、横にスクロールさせることができないため、1 行に半角 87 文字以上を表示させようとすると表示内容を確認できません。
これらの問題を解決するために、本器では PRINT 文により画面表示される内容を内蔵ハード・ディスク内に記録することが可能です。
設定は、システム・コンフィグレーションで行います。詳細は「1.5.1 シミュレータのログ・ファイル設定」を参照して下さい。
このファイルは、本体のエディタにより内容を確認することができます。また、3.5 インチ・フロッピー・ディスクにコピーすれば、MS-DOS のパソコン上でも内容確認することができます。

(2) INPUT()

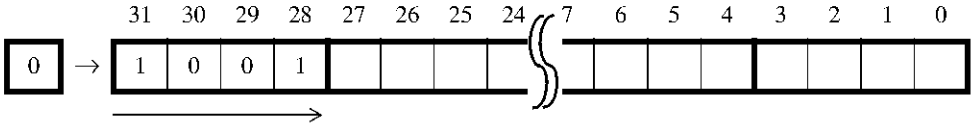
機能 キー入力された値を関数値として返します。
キー入力を行うときは、カーソルを Command の位置 (下図参照) に移動させて下さい。
キー入力は、10 進数入力と 16 進数入力の 2 通りがあります。詳しくは、「6.2.3 変数・定数・配列、(2) 定数」の説明を参照して下さい。



例
```
PRINT("INPUT A=")
A= INPUT( )
PRINT("%D¥N",A)
```

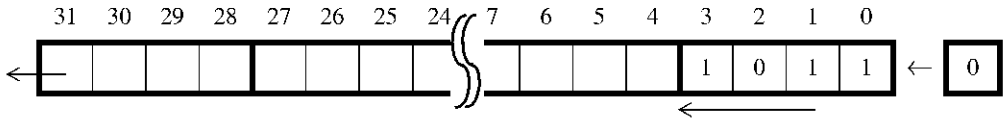
(3) R_SHIFT (val,times)

機能 関数値として、引数 val の値を引数 times で指定された回数だけ右 (Right) シフトした値を返します。ただし、論理シフトが以下のように実行されます。



(4) L_SHIFT (val,times)

機能 関数値として、引数 val の値を引数 times で指定された回数だけ左 (Left) シフトした値を返します。ただし、論理シフトが以下のように実行されます。



例
```
IF   DATA   >=   20
     THEN   DATA  = R_SHIFT (DATA, 4 )
     ELSE   DATA  = L_SHIFT (DATA, 4 )

END
```

(5) RND()

機能 0 ～255 のランダム値を発生します。
乱数の系列は、本器を立ち上げるごとに異なるように設定されています。乱数の系列を固定で使用する場合は、RANDOMIZE(SEED) 関数により乱数の種を指定して下さい。ただし、内部でもこの関数を使用しているので、プログラム中では、指定した系列どおりに発生しないことがあります。

例
```
B=RND( )
PRINT("RANDOM=%D¥N",B)
```

(6) RANDOMIZE (seed)

機能 新しい乱数の種 (seed) を指定することにより乱数の系列を変更することができます。ただし 0×1234567 は、内部で特別な意味を持っているので、使用しないで下さい。また、内部でも RND 関数を (TEI 割当手順などで) 使用しているので、seed が同じでもユーザ・プログラム中で RND 関数により得られるランダム値は、プログラムを実行させるごとに異なった発生をすることがあります。

例
```
RANDOMIZE(1)
B=RND( )
PRINT("RANDOM=%D¥N",B)
```

(7) GET_TIME()

機能 内部クロックの現在値を読み出します。(10msec 分解能です。)

例
```
TIME=GET_TIME( )
PRINT("TIME=%D¥N",TIME)
```

(8) VAL_TIME(time,type)

機能 引数 time（GET_TIME() 関数で読み出した現在の内部クロック値）を引数 type で指定した時刻の値に変換します。

引数 time ：GET_TIME() 関数で読み出した内部クロックの現在値
type ：時刻情報に変換する種類
0 ：年
1 ：月
2 ：日
3 ：時
4 ：分
5 ：秒
6 ：ミリ秒

関数値 0 ～ 2099：時刻情報
年 ：1980 ～ 2099
月 ：1 ～ 12
日 ：1 ～ 31
時 ：0 ～ 23
分 ：0 ～ 59
秒 ：0 ～ 59
ミリ秒 ：0 ～ 990（10msec 分解能です。）
-60 ：引数エラー
-170 ：オーバーフロー（2099 年 12 月 31 日以降に実行すると発生します。）

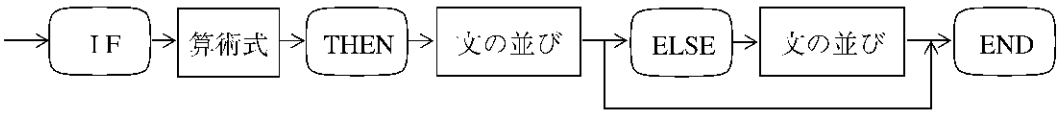
例
```
TIME  = GET_TIME()
YEAR  = VAL_TIME(TIME, 0)
MONTH = VAL_TIME(TIME, 1)
DAY   = VAL_TIME(TIME, 2)
HOUR  = VAL_TIME(TIME, 3)
MIN   = VAL_TIME(TIME, 4)
SEC   = VAL_TIME(TIME, 5)
PRINT("DATE IS %D/%D/%D¥N", YEAR, MONTH, DAY)
PRINT("TIME IS %D:%D:%D¥N", HOUR, MIN, SEC)
```

6.2.6 制御系ステートメント

PSL51 で使用可能な制御系ステートメントを以下に説明します。

(1) IF 文

構文



説明

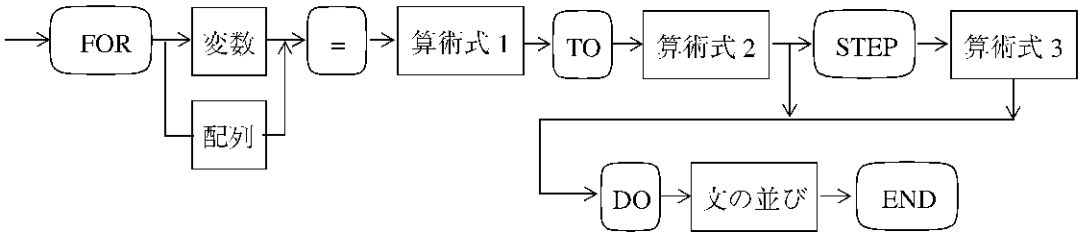
算術式で書かれた条件が真 (0 以外) ならば、THEN 直後の命令文を実行します。もし偽ならばELSE がある場合は、ELSE 直後の文を実行します。
ELSE がない場合は、END 以降の文を実行します。

例

```
IF ! CHECK(ARG1, ARG2)       THEN OK = 0
                             ELSE OK  = 1
END
.
.
.
.
FUNC   CHECK(X,Y)
  IF   MODE==0               THEN  RET = X+Y
                             ELSE  RET = X-Y
  END
RETURN(RET)
```

(2) FOR 文

構文



説明

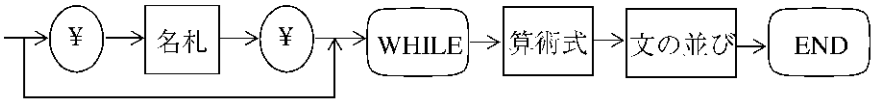
FOR 以下の条件式に従い、DO から END で囲まれた文の並びを繰り返して実行します。FOR 以下の条件式は、まず、変数あるいは配列に算術式 1 の初期値が代入され、DO ~ END の文の並びを実行するごとに、算術式 3 の値だけ、変数あるいは配列が加算されます。
DO ~ END の文の並びは変数あるいは配列が算術式 2 と等しくなるまで繰り返して実行されます。“STEP 算術式 3 ” は省略することもでき、省略すると増分は自動的に 1 に設定されます。

例

```
FOR I = 0   TO  13  STEP  1  DO
   A[I]  = I+2
   PRINT ("A[%D]=%D ¥N",I,A[I] )
END
```

(3) WHILE 文

構文



説明

算術式が真 (0 以外) の間、算術式から END で囲まれた文の並びを繰り返して実行します。
名札は、英文字で始まり、英文字、数字または '_' で構成されます。認識される文字数は、最大 10 文字です。

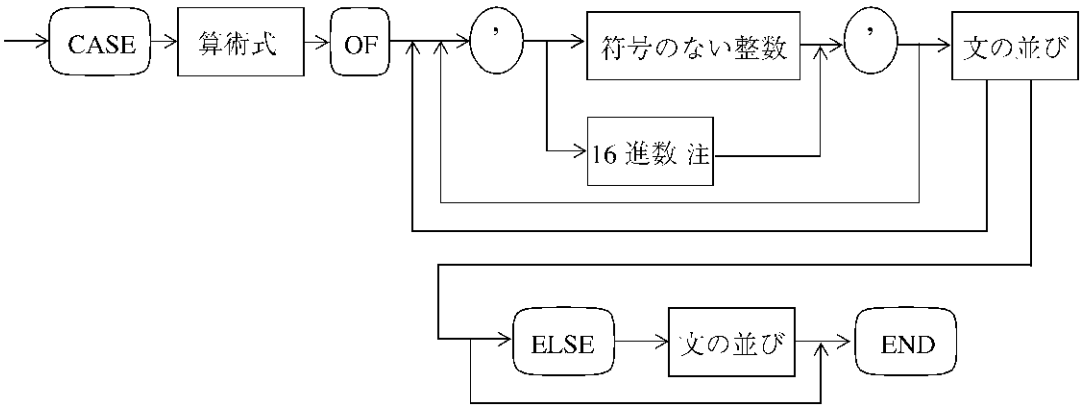
例

```
I = 10
WHILE I
  AB[I] = xyz+I
  I = I-1
  PRINT ("AB[%D]=%D¥N",I,AB[I])
END
```

注意　WHILE 文のループから抜け出すためには、EXIT 文を使用します。詳細は、(5) EXIT 文の項を参照して下さい。

(4)　CASE 文

構文



説明　次の例では、変数 xyz の値
　0 または1 または 3 のとき　関数 PROCL(0)
　2 または4 のとき　関数 PROCL(1)
　5 のとき　関数 PROCL(2)
それ以外のときは関数 PROCL(3) がそれぞれ実行されます。

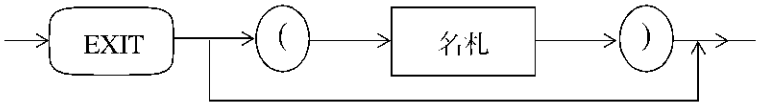
例

```
CASE   XYZ    OF
  '0' '1'  '3'          PROCL(0 )
  '2' '4'               PROCL(1 )
  '5'                   PROCL(2 )
  ELSE                  PROCL(3 )
END
```

注　　H '0' ～H'7FFFFFFF' の範囲で指定することができます。

(5) EXIT 文

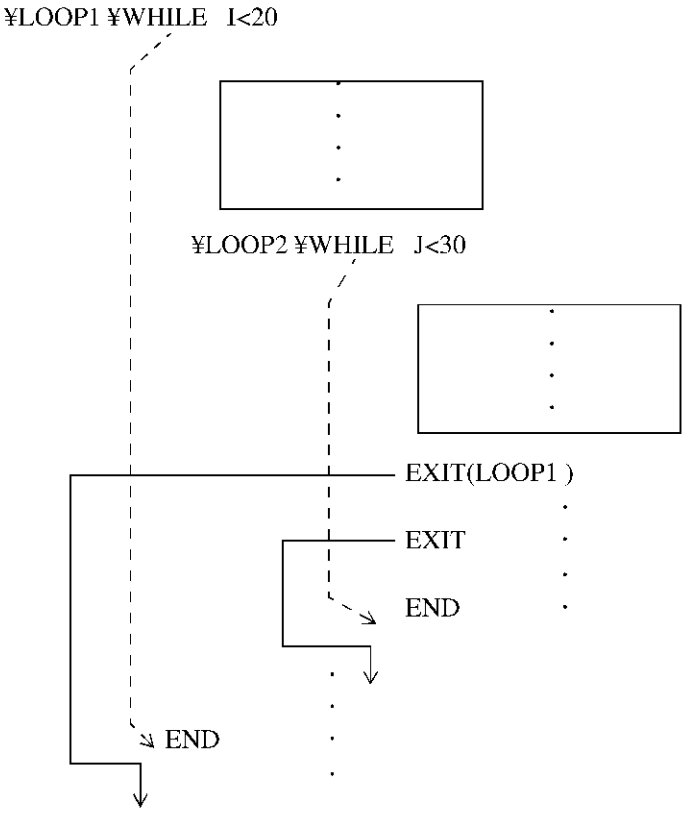
構文



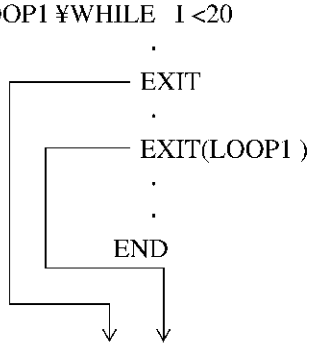
説明

EXIT 文は、現在実行されている最も内側の WHILE 文ループを強制的に終了し、そのループを抜け出します。また、名札が指定されているときには、その名札を持つ WHILE 文により構成されている WHILE ループを抜け出します。
名札は、英文字で始まり、英文字、数字または '_' で構成されます。認識される文字数は、最大 10 文字です。

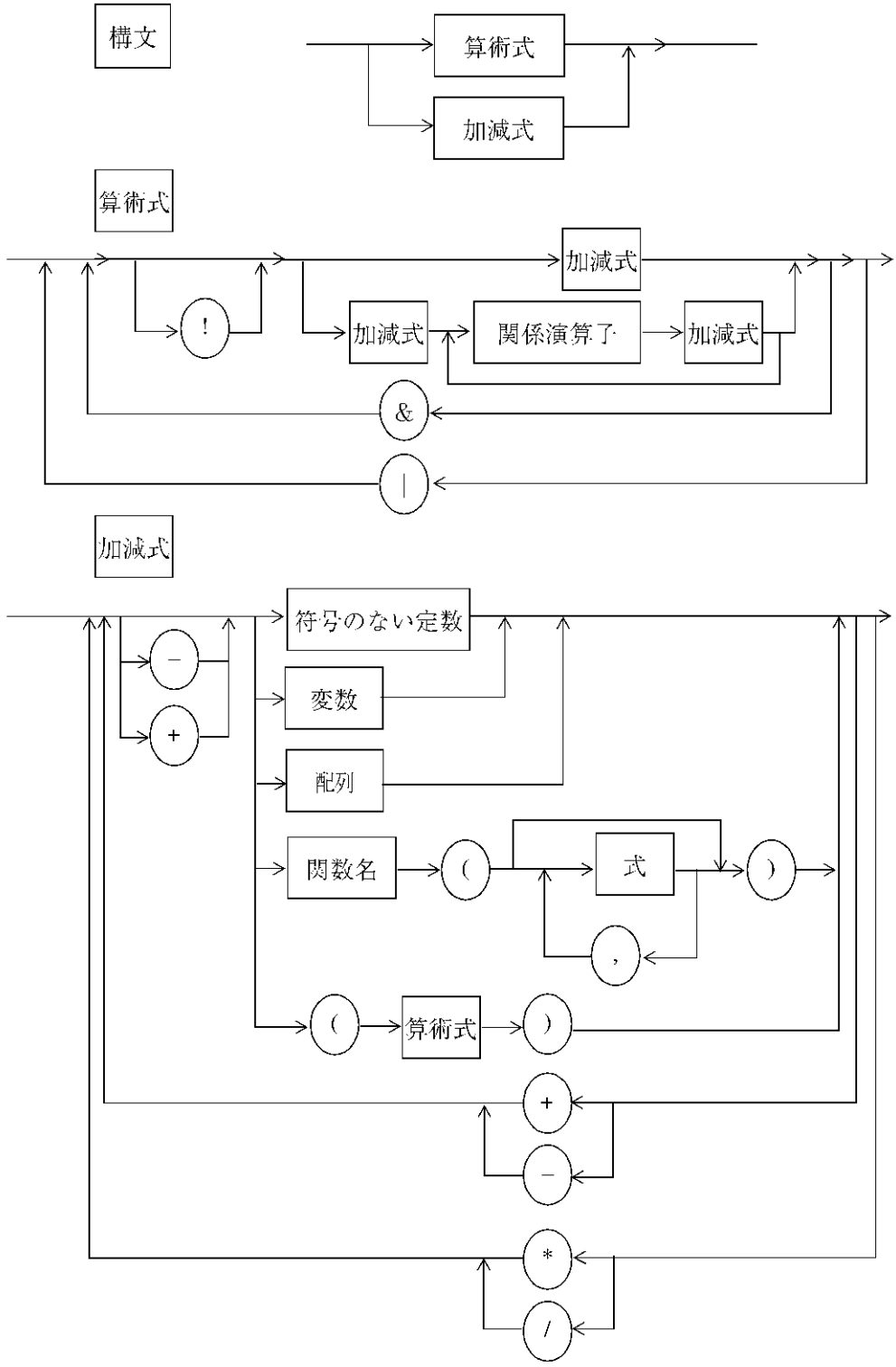
例 1



例 2

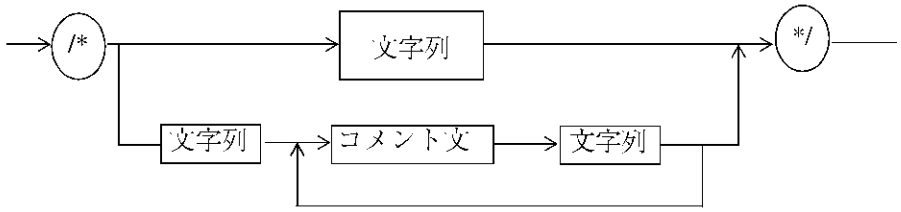


6.2.7 式



6.2.8 コメント文

構文



説明

プログラム中において、/* と */ で挟まれた文字列は、コメント (注釈) と解釈されコンパイルされません。
文字列は、空文字列も可能です。

例

```
/**********************************
   This is COMMENT example !!!
**********************************/
IF AB == XY            /* same value ? */
    THEN     PRINT    (" sb  == XY¥N" )
    ELSE     PRINT    (" sb  != XY¥N" )
END
```

7. 機能別関数一覧

7.1 D チャネル・シミュレーション

(1) 共通関数

関数名	内容
(1) INSERT	送信フレームの任意のオクテットの値を変える。
(2) SET_FRLEN	送信するフレームのフレーム長を変える。
(3) SET_CHANN	送受信するチャネルを指定する。
(4) RECEIVE	タイムアウト / フレーム / キー入力 / イベント受信待ち状態にする。
(5) T_START	タイマを起動する。(分解能 1 秒のタイマ起動)
TM_START	タイマを起動する。(分解能 100m 秒のタイマ起動)
(6) T_STOP	タイマを停止する。
(7) READ_TIMER	タイムアウトしたタイマの ID を得る。
(8) SEND_EVENT	他チャネルにイベントを送る。
(9) READ_EVENT	他チャネルから送られてきたイベントの内容を読み出す。
(10) EXTRACT	受信フレームの任意のオクテットの値を読み出す。
(11) RXFRLEN	受信フレーム・データ長を調べる。
(12) WAIT	指定の時間プログラムを停止する。
(13) PH_ACT	レイヤ 1 を起動する。
(14) PH_DEACT	レイヤ 1 を停止する。
(15) READ_VAL	キー入力された数値を読み出す。
(16) REG_EVTID	イベント ID を登録する。
(17) REL_EVTID	登録されているイベント ID を解放する。
(18) SEND_EVTS	他チャネルにイベントを送ります。
(19) WRITE_EVTS	他チャネルに送信するイベントを書き込みます。
(20) READ_EVTS	他チャネルから送信されたイベントを読み出します。
(21) COUNT_EVT	他チャネルから送信されたイベント数を得ます。
(22) CONN_BCH	B チャネル回線を接続します。
(23) DISC_BCH	B チャネル回線を開放します。
(24) GET_SIMID	現在実行中のシミュレーション ID を得ます。
(25) GET_LAY1ID	現在実行中のレイヤ 1 インタフェース ID を得ます。

(2) トランスペアレント・モード用関数

フレ･･ム送信関連	
関数名	内容
(1) SENDT	フレ･･ムを送信する。(メッセ･･ジ・デ･･タをそのままま送出)
(2) SENDF	フレームを送信する。(N(S),N(R),P/F を挿入して送出)
(3) INCVS	送信状態変数 V(S) を＋１する。
(4) INCVR	受信状態変数 V(R) を＋１する。
(5) SETVS	送信状態変数 V(S) の値を設定する。
(6) SETVR	受信状態変数 V(R) の値を設定する。
(7) INS_SAPI	送信フレ･･ムの SAPI 値を書き換える。
(8) INS_TEI	送信フレームの TEI 値を書き換える。
(9) INS_CR	送信フレームの C/R ビット値を書き換える。
(10) INS_NR	送信フレ･･ムの N(R) 値を書き換える。
(11) INS_NS	送信フレームの N(S) 値を書き換える。
(12) INS_PF	送信フレームの P/F ビット値を書き換える。
(13) INS_TYPE	送信フレ･･ムのフレ･･ム種別を書き換える。
(14) INS_CF1	送信フレーム制御フィールドの第 1 オクテット値を書き換える。
(15) INS_CF2	送信フレーム制御フィールドの第 2 オクテット値を書き換える。
(16) INS_FRCF1	送信 FRMR フレ･･ム情報フィ･･ルド内の第 1 オクテット値を書き換える。
(17) INS_FRCF2	送信 FRMR フレーム情報フィールド内の第 2 オクテット値を書き換える。
(18) INS_FRVS	送信 FRMR フレーム情報フィールド内の V(S) 値を書き換える。
(19) INS_FRVR	送信 FRMR フレ･･ム情報フィ･･ルド内の V(R) 値を書き換える。
(20) INS_FRCR	送信 FRMR フレーム情報フィールド内の C/R ビット値を書き換える。
(21) INS_FRWXYZ	送信 FRMR フレーム情報フィールド内の WXYZ ビット値を書き換える。

フレーム受信関連	
関数名	内容
(22) RXSAPI	受信フレームの SAPI 値を読み出す。
(23) RXTEI	受信フレ･･ムの TEI 値を読み出す。
(24) RXCR	受信フレームの C/R ビット値を読み出す。
(25) RXNR	受信フレームの N(R) 値を読み出す。
(26) RXNS	受信フレ･･ムの N(S) 値を読み出す。
(27) RXPF	受信フレームの P/F ビット値を読み出す。
(28) RXTYPE	受信フレームのフレーム種別を読み出す。
(29) RXCF1	受信フレ･･ム制御フィ･･ルド内の第 1 オクテット値を読み出す。
(30) RXCF2	受信フレーム制御フィールド内の第 2 オクテット値を読み出す。
(31) RXFRCF1	受信 FRMR フレーム情報フィールド内の第 1 オクテット値を読み出す。
(32) RXFRCF2	受信 FRMR フレ･･ム情報フィ･･ルド内の第 2 オクテット値を読み出す。
(33) RXFRVS	受信 FRMR フレーム情報フィールド内の V(S) 値を読み出す。
(34) RXFRVR	受信 FRMR フレーム情報フィールド内の V(R) 値を読み出す。
(35) RXFRCR	受信 FRMR フレ･･ム情報フィ･･ルド内の C/R ビット値を読み出す。
(36) RXFRWXYZ	受信 FRMR フレーム情報フィールド内の WXYZ ビット値を読み出す。

(3)　レイヤ 2 自動モード用関数

フレーム送信関連	
関数名	内容
(1) SENDI	I フレームを送信する。
(2) SENDUI	UI フレームを送信する。
(3) SENDXIDC	XID コマンドを送信する。
(4) SENDXIDR	XID レスポンスを送信する。
(5) SENDPKT	パケットを送信する。
(6) INCPS	送信順序番号 P(S) を＋ 1 する。
(7) INCPR	受信順序番号 P(R) を＋ 1 する。
(8) SETPS	送信順序番号 P(S) の値を設定する。
(9) SETPR	受信順序番号 P(R) の値を設定する。
(10) INS_PD	送信フレームのプロトコル識別子を書き換える。
(11) INS_CRL	送信フレームの呼番号長を書き換える。
(12) INS_CRF	送信フレームの呼番号フラグ値を書き換える。
(13) INS_CRV	送信フレームの呼番号値を書き換える。
(14) INS_MSG	送信フレームのメッセージ種別を書き換える。
(15) INS_INFO	送信フレームの情報要素群フィールドを書き換える。
(16) INS_CS_VAL	送信フレームの理由表示値を書き換える。
(17) INS_GFI	送信パケットのゼネラル・フォーマット識別子の値を書き換える。
(18) INS_Q	送信パケットの Q ビット値を書き換える。
(19) INS_D	送信パケットの D ビット値を書き換える。
(20) INS_LCGN	送信パケットの論理チャネル・グループ番号を書き換える。
(21) INS_LCN	送信パケットの論理チャネル番号を書き換える。
(22) INS_TYP	送信パケットのパケット・タイプ識別子を書き換える。
(23) INS_PR	送信パケットの受信順序番号 P(R) を書き換える。
(24) INS_PS	送信パケットの送信順序番号 P(S) を書き換える。
(25) INS_M	送信パケットのモアデータ表示値を書き換える。
(26) INS_CLL	送信パケットの発呼ユーザ・アドレス長を書き換える。
(27) INS_CDL	送信パケットの着呼ユーザ・アドレス長を書き換える。
(28) INS_DA	送信パケットの着呼ユーザ・アドレスを書き換える。
(29) INS_SA	送信パケットの発呼ユーザ・アドレスを書き換える。
(30) INS_FL	送信パケットのファシリティ長を書き換える。
(31) INS_F	送信パケットのファシリティを書き換える。
(32) INS_CAUSE	送信パケットの切断 / リスタート / リセット原因を書き換える。
(33) INS_DIAG	送信パケットの診断符号を書き換える。
(34) SETDTL	送信パケット内のデータ長を書き換える。
(35) INS_DATA	送信パケット内のデータを書き換える。

TEI 管理手順関連	
関数名	内容
(36) REQ_TEI	TEI 割当手順を起動する。(TE モード)
(37) CHKREQ_TEI	TEI チェック手順を行う。(NT モード)
(38) REMOVE_TEI	TEI 解除手順を行う。(NT モード)
(39) VERIFY_TEI	TEI 検証手順を行う。(TE モード)

リンク , SAPI, TEI 管理関連	
関数名	内容
(40) LINKON	リンクの設定を行う。
(41) LINKOFF	リンクの解放を行う。
(42) WAIT_LINK	相手局からのリンク設定待ち状態にする。
(43) L_STATUS	リンクが設定されているかを調べる。
(44) SET_BUSY	自局をビジー状態にする。
(45) REL_BUSY	自局のビジー状態を解除する。
(46) PROHIBIT_L	新たなリンクの設定を禁止する。
(47) PERMIT_L	リンク設定不許可状態を解除する。
(48) REG_TEI	TEI 値を本器に登録し、その TEI 値のフレームを受信可能にする。
(49) REL_TEI	TEI 値を解除し、その TEI 値のフレームを受信しないようにする。
(50) NEXT_TEI	新たな TEI 値を使用することを宣言する。(TE モード)
(51) ACT_SAPI	フレーム送出時に使用される SAPI 値を設定する。
(52) ACT_TEI	フレーム送出時に使用される TEI 値を設定する。
(53) ACT_LINK	フレーム送出時に使用されるリンク番号を設定する。
(54) LOCK_SAPI	SAPI 値が自動で設定変更されるのを禁止する。
(55) LOCK_TEI	TEI 値が自動で設定変更されるのを禁止する。
(56) LOCK_LINK	リンク番号が自動で設定変更されるのを禁止する。
(57) FLEX_SAPI	SAPI 値が自動で設定変更されるようにする。
(58) FLEX_TEI	TEI 値が自動で設定変更されるようにする。
(59) FLEX_LINK	リンク番号が自動で設定変更されるようにする。
(60) GET_SAPI	フレーム送出時に使用される SAPI 値を調べる。
(61) GET_TEI	フレーム送出時に使用される TEI 値を調べる。
(62) GET_LINK	フレーム送出時に使用されるリンク番号を調べる。
(63) SEE_LINK	現在設定されているリンク番号を調べる。

フレ･･ム受信関連	
関数名	内容
(64) RXSAPI	受信フレ･･ムの SAPI 値を読み出す
(65) RXTEI	受信フレームの TEI 値を読み出す。
(66) RXTYPE	受信フレームのフレーム種別を読み出す。
(67) RXPF	受信フレ･･ムの P/F ビット 値を読み出す。
(68) RXPD	受信フレームのプロトコル識別子を読み出す。
(69) RXCRL	受信フレームの呼番号長を読み出す。
(70) RXCRF	受信フレ･･ムの呼番号フラグ値を読み出す。
(71) RXCRV	受信フレームの呼番号値を読み出す。
(72) RXMSG	受信フレームのメッセージ種別を読み出す
(73) RXINFO_NUM	受信フレ･･ムの情報要素数を読み出す。
(74) RXINFO_ELM	受信フレームの情報要素識別子を読み出す。
(75) RXINFO_LEN	受信フレームの情報要素内容長を読み出す。
(76) RXINFO_VAL	受信フレ･･ムの情報内容を読み出す。
(77) RXINFO_POS	受信フレームの情報要素識別子の位置を調べる。
(78) RXCS_VAL	受信フレームの理由表示値を読み出す
(79) RXCHAN_NUM	受信フレ･･ムのチャネル番号を読み出す。
(80) RXGFI	受信パケットのゼネラル・フォーマット識別子の値を読み出す。
(81) RXQ	受信パケットの Q ビット値を読み出す。
(82) RXD	受信パケットの D ビット値を読み出す。
(83) RXLCGN	受信パケットの論理チャンネル・グループ番号を読み出す。
(84) RXLCN	受信パケットの論理チャネル番号を読み出す。
(85) RXTYP	受信パケットのパケットタイプ識別子を読み出す。
(86) RXPR	受信パケットの受信順序番号 P(R) を読み出す。
(87) RXPS	受信パケットの送信順序番号 P(S) を読み出す。
(88) RXM	受信パケットのモアデ･･タ表示値を読み出す。
(89) RXCLL	受信パケットの発呼ユーザ・アドレス長を読み出す
(90) RXCDL	受信パケットの着呼ユーザ・アドレス長を読み出す。
(91) RXDA	受信パケットの着呼ユ･･ザ・アドレスを読み出す。
(92) RXSA	受信パケットの発呼ユーザ・アドレスを読み出す。
(93) RXFL	受信パケットのファシリティ長を読み出す。
(94) RXF	受信パケットのファシリティを読み出す。
(95) RXCAUSE	受信パケットの切断 / リスタート / リセット原因を読み出す。
(96) RXDIAG	受信パケットの診断符号を読み出す。
(97) RXDTL	受信パケット内のデ･･タ長を読み出す。
(98) RXDATA	受信パケット内のデータを読み出す。

7.2 B チャネル・シミュレーション

(1) 共通関数

関数名	内容
(1) INSERT	送信フレームの任意のオクテットの値を変える。
(2) SET_FRLEN	送信するフレームのフレーム長を変える。
(3) SET_CHANN	送受信するチャネルを指定する。
(4) SET_CH64K	64kbps のシミュレータが使用するチャネルを指定する。
(5) SET_CH32K	32kbps のシミュレータが使用するチャネルを指定する。
(6) RECEIVE	タイムアウト / フレーム / キー入力 / イベント受信待ち状態にする。
(7) T_START	タイマを起動する。(分解能 1 秒のタイマ起動)
TM_START	タイマを起動する。(分解能 100m 秒のタイマ起動)
(8) T_STOP	タイマを停止する。
(9) READ_TIMER	タイムアウトしたタイマの ID を得る。
(10) SEND_EVENT	他チャネルにイベントを送る。
(11) READ_EVENT	他チャネルから送られてきたイベントの内容を読み出す。
(12) EXTRACT	受信フレームの任意のオクテットの値を読み出す。
(13) RXFRLEN	受信フレーム・データ長を調べる。
(14) WAIT	指定の時間プログラム停止する。
(15) PH_ACT	レイヤ 1 を起動する。
(16) PH_DEACT	レイヤ 1 を停止する。
(17) READ_VAL	キー入力された数値を読み出す。
(18) REG_EVTID	イベント ID を登録する。
(19) REL_EVTID	登録されているイベント ID を解放する。
(20) SEND_EVTS	他チャネルにイベントを送ります。
(21) WRITE_EVTS	他チャネルに送信するイベントを書き込みます。
(22) READ_EVTS	他チャネルから送信されたイベントを読み出します。
(23) COUNT_EVT	他チャネルから送信されたイベント数を得ます。
(24) GET_SIMID	現在実行中のシミュレーション ID を得ます。
(25) GET_LAY1ID	現在実行中のレイヤ 1 インタフェース ID を得ます。

(2)　トランスペアレント・モード用関数

フレ･･ム送信関連	
関数名	内容
(1)　SENDT	フレ･･ムを送信する。(メッセ･･ジ・デ･･タをそのままま送出)
(2)　SENDF	フレームを送信する。(N(S),N(R),P/F を挿入して送出)
(3)　INCVS	送信状態変数 V(S) を＋１する。
(4)　INCVR	受信状態変数 V(R) を＋１する。
(5)　SETVS	送信状態変数 V(S) の値を設定する。
(6)　SETVR	受信状態変数 V(R) の値を設定する。
(7)　INS_ADRS	送信フレ･･ムのアドレス値を書き換える。
(8)　INS_NR	送信フレームの N(R) 値を書き換える。
(9)　INS_NS	送信フレームの N(S) 値を書き換える。
(10)　INS_PF	送信フレ･･ムの P/F ビット値を書き換える。
(11)　INS_TYPE	送信フレームのフレーム種別を書き換える。
(12)　INS_CF1	送信フレーム制御フィールドの第１オクテット値を書き換える。
(13)　INS_CF2	送信フレ･･ム制御フィ･･ルドの第２オクテット値を書き換える。
(14)　INS_FRCF1	送信 FRMR フレーム情報フィールド内の第１オクテット値を書き換える。
(15)　INS_FRCF2	送信 FRMR フレーム情報フィールド内の第２オクテット値を書き換える。
(16)　INS_FRVS	送信 FRMR フレ･･ム情報フィ･･ルド内の V(S) 値を書き換える。
(17)　INS_FRVR	送信 FRMR フレーム情報フィールド内の V(R) 値を書き換える。
(18)　INS_FRCR	送信 FRMR フレーム情報フィールド内の C/R ビット値を書き換える。
(19)　INS_FRWXYZ	送信 FRMR フレ･･ム情報フィ･･ルド内の WXYZ ビット値を書き換える。

フレ･･ム受信関連	
関数名	内容
(20)　RXADRS	受信フレームのアドレス値を読み出す
(21)　RXNR	受信フレ･･ムの N(R) 値を読み出す。
(22)　RXNS	受信フレームの N(S) 値を読み出す。
(23)　RXPF	受信フレームの P/F ビット値を読み出す。
(24)　RXTYPE	受信フレ･･ムのフレ･･ム種別を読み出す。
(25)　RXCF1	受信フレーム制御フィールドの第１オクテットの値を読み出す。
(26　RXCF2	受信フレーム制御フィールドの第２オクテットの値を読み出す。
(27)　RXFRCF1	受信 FRMR フレ･･ム情報フィ･･ルド内の第１オクテット値を読み出す。
(28)　RXFRCF2	受信 FRMR フレーム情報フィールド内の第２オクテット値を読み出す。
(29)　RXFRVS	受信 FRMR フレーム情報フィールド内の V(S) 値を読み出す。
(30)　RXFRVR	受信 FRMR フレ･･ム情報フィ･･ルド内の V(R) 値を読み出す。
(31)　RXFRCR	受信 FRMR フレーム情報フィールド内の C/R ビット値を読み出す。
(32)　RXFRWXYZ	受信 FRMR フレーム情報フィールド内の WXYZ ビット値を読み出す

(3) レイヤ 2 自動モード用関数

フレーム送信関連	
関数名	内容
(1) SENDI	I フレームを送信する。
(2) SENDPKT	パケットを送信する。
(3) INCPS	送信順序番号 P(S) を＋1する。
(4) INCPR	受信順序番号 P(R) を＋1する。
(5) SETPS	送信順序番号 P(S) の値を設定する。
(6) SETPR	受信順序番号 P(R) の値を設定する。
(7) INS_GFI	送信パケットのゼネラル・フォーマット識別子の値を書き換える。
(8) INS_Q	送信パケットの Q ビット値を書き換える。
(9) INS_D	送信パケットの D ビット値を書き換える。
(10) INS_LCGN	送信パケットの論理チャネル・グループ番号を書き換える。
(11) INS_LCN	送信パケットの論理チャネル番号を書き換える。
(12) INS_TYP	送信パケットのパケット・タイプ識別子を書き換える。
(13) INS_PR	送信パケットの受信順序番号 P(R) を書き換える。
(14) INS_PS	送信パケットの送信順序番号 P(S) を書き換える。
(15) INS_M	送信パケットのモア・データ表示値を書き換える。
(16) INS_CLL	送信パケットの発呼ユーザ・アドレス長を書き換える。
(17) INS_CDL	送信パケットの着呼ユーザ・アドレス長を書き換える。
(18) INS_DA	送信パケットの着呼ユーザ・アドレスを書き換える
(19) INS_SA	送信パケットの発呼ユーザ・アドレスを書き換える。
(20) INS_FL	送信パケットのファシリティ長を書き換える。
(21) INS_F	送信パケットのファシリティを書き換える。
(22) INS_CAUSE	送信パケットの切断 / リスタート / リセット原因を書き換える。
(23) INS_DIAG	送信パケットの診断符号を書き換える。
(24) SET_DTL	送信パケット内のデータ長を書き換える。
(25) INS_DATA	送信パケット内のデータを書き換える。

リンク関連	
関数名	内容
(26) LINKON	リンクの設定を行う。
(27) LINKOFF	リンクの解放を行う。
(28) WAIT_LINK	相手からのリンク設定待ち状態にする。
(29) L_STATUS	リンクが設定されているかを調べる。
(30) SET_BUSY	自局をビジー状態にする。
(31) REL_BUSY	自局のビジー状態を解除する。

フレーム受信関連	
関数名	内容
(32) RXGFI	受信パケットのゼネラルフォーマット識別子の値を読み出す。
(33) RXQ	受信パケットの Q ビット値を読み出す。
(34) RXD	受信パケットの D ビット値を読み出す。
(35) RXLCGN	受信パケットの論理チャネル・グループ番号を読み出す
(36) RXLCN	受信パケットの論理チャネル番号を読み出す。
(37) RXTYP	受信パケットのパケット・タイプ識別子を読み出す。
(38) RXPR	受信パケットの受信順序番号 P(R) を読み出す。
(39) RXPS	受信パケットの送信順序番号 P(S) を読み出す。
(40) RXM	受信パケットのモア・データ表示値を読み出す。
(41) RXCLL	受信パケットの発呼ユーザ・アドレス長を読み出す。
(42) RXCDL	受信パケットの着呼ユーザ・アドレス長を読み出す。
(43) RXDA	受信パケットの着呼ユーザ・アドレスを読み出す。
(44) RXSA	受信パケットの発呼ユーザ・アドレスを読み出す。
(45) RXFL	受信パケットのファシリティ長を読み出す。
(46) RXF	受信パケットのファシリティを読み出す。
(47) RXCAUSE	受信パケットの切断 / リスタート / リセット原因を読み出す。
(48) RXDIAG	受信パケットの診断符号を読み出す。
(49) RXDTL	受信パケット内のデータ長を読み出す。
(50) RXDATA	受信パケット内のデータを読み出す。

(4) 音声、BERT 用関数

関数名	内容
(1) SOUND_ON	音声シミュレーション 1 用ヘッド・セットの通話を可能にする。
(2) SOUND_OFF	音声シミュレーション 1 用ヘッド・セットの通話を止める。
(3) SOUND1_ON	音声シミュレーション 1 用ヘッド・セットの通話を可能にする。
(4) SOUND1_OFF	音声シミュレーション 1 用ヘッド・セットの通話を止める。
(5) VOLUME	音声シミュレーション 1 用ヘッド・セットの音量を変える。
(6) SET_PRBS	BERT で使用する擬似ランダム・パターンを指定する。
(7) SET_WORD	BERT で使用するワード・パターンを指定する。
(8) SET_ADDERR	BERT の際、送信側に挿入するエラーの割合を指定する。
(9) BERT_ON	BERT を行う。
(10) PTNPCM_ON	1kHz, 0dBm の基準正弦波用デジタル・パターンを出力する。
(11) PTNPCM_OFF	1kHz, 0dBm の基準正弦波用デジタル・パターンの出力を止める。

8. D チャネル・シミュレーション

8.1 共通関数

表 8-1 に共通関数の一覧を示します。

表 8-1 共通関数

関数名	内容
(1) INSERT	送信フレームの任意のオクテットの値を変える。
(2) SET_FRLEN	送信するフレームのフレーム長を変える。
(3) SET_CHANN	送受信するチャネルを指定する。
(4) RECEIVE	タイムアウト / フレーム / キー入力 / イベント受信待ち状態にする。
(5) T_START	タイマを起動する。(分解能 1 秒のタイマ起動)
TM_START	タイマを起動する。(分解能 100m 秒のタイマ起動)
(6) T_STOP	タイマを停止する。
(7) READ_TIMER	タイムアウトしたタイマの ID を得る。
(8) SEND_EVENT	他チャネルにイベントを送る。
(9) READ_EVENT	他チャネルから送られてきたイベントの内容を読み出す。
(10) EXTRACT	受信フレームの任意のオクテットの値を読み出す。
(11) RXFRLEN	受信フレーム・データ長を調べる。
(12) WAIT	指定の時間プログラムを停止する。
(13) PH_ACT	レイヤ 1 を起動する。
(14) PH_DEACT	レイヤ 1 を停止する。
(15) READ_VAL	キー入力された数値を読み出す。
(16) REG_EVTID	イベント ID を登録する。
(17) REL_EVTID	登録されているイベント ID を解放する。
(18) SEND_EVTS	他チャネルにイベントを送ります。
(19) WRITE_EVTS	他チャネルに送信するイベントを書き込みます。
(20) READ_EVTS	他チャネルから送信されたイベントを読み出します。
(21) COUNT_EVT	他チャネルから送信されたイベント数を得ます。
(22) CONN_BCH	B チャネル回線を接続します。
(23) DISC_BCH	B チャネル回線を開放します。
(24) GET_SIMID	現在実行中のシミュレーション ID を得ます。
(25) GET_LAY1ID	現在実行中のレイヤ 1 インタフェース ID を得ます。

次ページ以降に各関数の説明をします。

(1) INSERT

呼出し形式　INSERT("NAME", n, DT)

概　要　送信フレームの任意のオクテット値を書き換えます。

引数の説明
- NAME　：データの変更を行うフレームのフレーム名
- n　：データの変更を行うオクテット値
  - トランスペアレント・モード時　：1～512
  - レイヤ2自動実行モード時：1～508 または 1～509
  - (最大値は、レイヤ2のメッセージに依存する)
- DT　：データの変更値 (0～255 の範囲)

関数値
- 0　：正常終了
- –60　：引数エラー
- –61　：フレーム名エラー

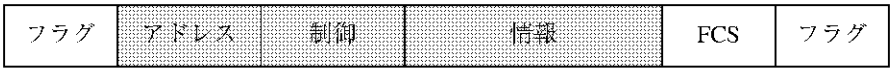
使用例　INSERT("SABME", 2, H'81')

機能説明　引数で指定したフレーム名の任意のオクテットのデータを変更する関数です。

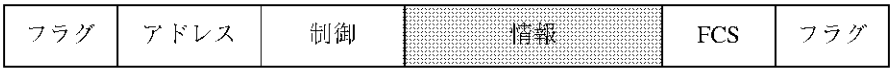
注意事項　一度、変更した値は、シミュレーションをストップするまで有効です。
トランスペアレント・モードとレイヤ2自動モードとでは変更できるデータの領域が違います。(下図参照)
下図で斜線部の領域が変更可能です。
また、変更位置は斜線部の左側から順に、1 から数えたオクテットとします。

制限事項　引数で指定したフレーム名のフレームがない場合は、フレーム名エラーになります。
また、データの変更を行うオクテット値やデータの変更値に範囲外の値を指定すると、引数エラーになります。

① トランスペアレント・モード



② レイヤ2自動モード



(2) SET_FRLEN

呼出し形式　SET_FRLEN("NAME", LEN)

概　要　送信フレームのデータ長を変えます。

引数の説明
NAME　: フレーム名
LEN　: 変更するデータ長
トランスペアレント・モード時　: 1～512
レイヤ 2 自動実行モード時 : 1～508 または 1～509
(最大値は、レイヤ 2 のメッセージに依存する)

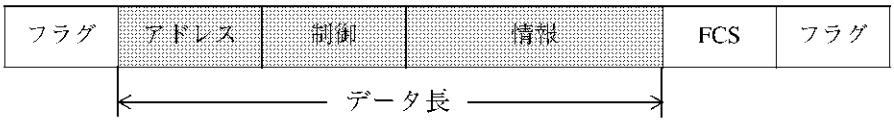
関数値
0　: 正常終了
–60　: 引数エラー
–61　: フレーム名エラー

使用例　SET_FRLEN("SPL1",10)

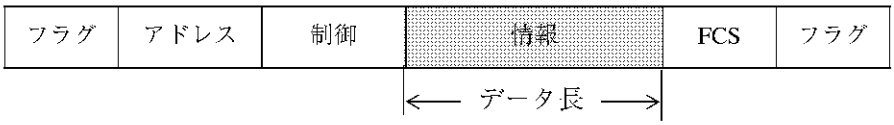
機能説明　引数で指定したフレーム名のデータ長を変更する関数です。

注意事項　一度フレーム長を変えると、シミュレーションをストップするまで有効です。トランスペアレントモードとレイヤ 2 自動実行モードでは、変更できるデータ長の意味が異なります。(下図参照)

① トランスペアレント・モード



② レイヤ 2 自動モード



(3) SET_CHANN

呼出し形式　SET_CHANN(CHANN)

概　要　送受信するチャネルを設定します。

引数の説明　CHANN　:チャネル番号
0: チャネルを設定していない状態
1: B1 チャネル
2: B2 チャネル
3: B3 チャネル
（基本、U 点インタフェースの場合は D チャネル）
・
・
・
・
24: B24 チャネル
（一次群インタフェースの場合は B1 ～ B24 チャネルの設定が可能）

関数値　0　:正常終了
-163　:使用しているインタフェースに指定のチャネルがない
-180　:指定したチャネルはすでに使用されている

使用例
```
PH_ACT( )
SET_CHANN(1)
SENDT("SPL")
```

機能説明　フレームを送受信するチャネルを設定する関数です。
基本または U 点インタフェースを使用している場合、D チャネルがデフォルトで設定されています。
一次群インタフェースの場合は必ず設定が必要です。

(4) RECEIVE

呼出し形式　RECEIVE(TIMER_ID)

概　要　タイムアウト / フレーム / キー入力 / イベント受信待ち状態にします。

引数の説明　TIMER_ID　: タイマ番号 (0 ~ 16777215)

関数値
0	: タイムアウト・イベント受信による終了
1	: フレーム受信による終了
2	: キー入力受信による終了
10	: D チャネルからのイベント受信による終了
20	: B チャネルからのイベント受信による終了
200	: 登録されたイベント ID を使用したイベントの受信

使用例
```
TM_ID=1
TM_SEC=10
T_START(TM_ID,TM_SEC)
RET=RECEIVE(TM_ID)
IF RET==0 THEN
   PRINT("TIME OUT¥N")
END
```

機能説明　本関数を実行すると、タイムアウト / フレーム / キー入力 / イベントの受信待ち状態になります。上記のイベントを受信するまでプログラムは停止します。上記のイベントを受信すると RECEIVE 関数を抜けプログラムは再実行します。このとき RECEIVE 関数が返す関数値は受信したイベントの種類を示します。
タイマは、本関数と T_START, TM_START または T_STOP 関数を用いて管理します。
タイマの起動は、T_START または TM_START 関数で行い、タイムアウトしたそのイベントは本関数で受け取ります。そのとき、本関数の引数 TIMER_ID 以外のタイマがタイムアウトした場合は受信されません。
ただし、TINER_ID が 0 のときは例外で、すべての TIMER_ID のタイムアウト・イベントを受信します。このときどのタイマがタイムアウトしたかは READ_TIMER 関数で知ることができます。タイマの停止は、T_STOP 関数で行います。
すでに、フレームを受信している状態で RECEIVE 関数が実行されると関数値として 1 を返し終了します。このとき RX で始まる受信フレーム読み出し関数が利用できます。
受信した次のフレームの内容を読み出したい場合は、再び RECEIVE 関数を実行します。RECEIVE 関数を実行するごとに受信したフレームの順に、その内容を RX で始まる関数を使用することにより読み出すことができます。
受信したフレームがない場合や、次のフレームをまだ受信していない場合は、受信待ち状態になりフレームを受信した後、上記のようにフレーム内容を読み出せるようになります。
INPUT 関数が実行されていないときに、Command ラインから数値が入力されると、関数値 2 を返します。この数値は、READ_VAL 関数で読み出すことができます。(INPUT 関数参照)

他チャネルからのイベントを受信した場合には、関数値として上記の値を返します。受信した他チャネルからのイベントは READ_EVENT 関数で読むことができます。

(5) T_START / TM_START

呼出し形式　T_START(TIMER_ID, SEC)
TM_START(TIMER_ID, SEC)

概　要　タイマを起動します。

引数の説明　TIMER_ID　: 起動するタイマのタイマ番号 (1 ~ 16777215)
SEC　: タイムアウト値
関数 T_START では、1 秒単位　(0 ~ 16777215)
関数 TM_START では、100m 秒単位　(0 ~ 16777215)

関数値　0　: 正常終了
–60　: 引数エラー
–120　: 最大同時起動タイマ数オーバ

使用例
```
T_START(201,1)
T202=202
S=2
T_START(T202,S)
```

機能説明　引数で指定したタイマ番号のタイマを起動する関数です。
この関数が実行されたときに、同じタイマ番号のタイマが既に起動されている場合は、そのタイマを停止し、新たにタイマを起動させます。

制限事項　タイマ番号に 1 ~ 16777215 以外の値、タイムアウト値に 0 ~ 16777215 以外の値が設定されると引数エラーになります。
最大同時起動タイマ数以上タイマが起動されると最大同時起動タイマ数オーバのエラーになります。
最大同時起動タイマ数は、30 個です。

(6) T_STOP

呼出し形式	T_STOP(TIMER_ID)
概　要	タイマを停止します。
引数の説明	TIMER_ID　: 停止するタイマのタイマ番号 (1 ~ 16777215)
関数値	0　　: 正常終了 –60　　: 引数エラー –121　　: 起動しているタイマがない
使用例	T_STOP(201) T202=202 T_STOP(T202)
機能説明	引数で指定したタイマ番号のタイマを停止する関数です。
制限事項	引数で指定したタイマ番号のタイマが起動していない場合は、エラー・コードを関数値として返すだけで、他に何も起こりません。

(7) READ_TIMER

呼出し形式　　READ_TIMER( )

概　要　　タイムアウトしたタイマの ID を得ます。

引数の説明

関数値　　1 ～16777215 : タイムアウトしたタイマの番号
–121　　: タイムアウトしたタイマがない

使用例

```
T_START(1,5)
T_START(2,10)
RET=RECEIVE(0)
IF RET==0 THEN
   TIMER=READ_TIMER( )
END
PPINT("TIMER=%D¥N",TIMER)
```

機能説明　　タイムアウトしたタイマの番号を知ることができます。
タイマの使い方は、RECEIVE 関数の説明を参考にして下さい。

(8) SEND_EVENT

呼出し形式　SEND_EVENT(DEST, MSG)

概　要　シミュレータにイベントを送ります。

引数の説明
DEST　：イベントの送り先
　10　：D チャネル・シミュレータ
　20　：B チャネル・シミュレータ
　上記以外　：登録したイベント ID
MSG　：送出するイベント (0～16777215)

関数値
0　：正常終了
–60　：引数エラー
–155　：イベントの送信に失敗した

使用例
送出元のシミュレータ
```
SEND_EVENT(123,50)
```
送出先のシミュレータ
```
REG_EVTID(123)
RET = RECEIVE(0)
IF RET == 200 THEN  /* 登録されたイベント ID を使用したイベントを受信した場合 */
   EVT = READ_EVENT( )
   PRINT("RECEIVED EVENT:%D¥N", EVT)
END
REL_EVTID(123)
```

機能説明
シミュレータにイベントを送出します。
引数として、送出先とイベント内容を指定します。送出先には、送出先のシミュレータが登録しているイベント ID を指定します。また、D5112 との互換性を考慮し、送出先に 10, 20 を指定した場合は同じシミュレーション番号をもつ D チャンネル・シミュレータ、B チャンネル・シミュレータに送られます。シミュレーション番号とは LAPD シミュレータ x における x の値です。(x = 1～4)

制限事項
イベントの送り先に 10、20、登録したイベント ID 以外を指定すると引数エラーになります。また、送信するイベントに範囲外の値を指定したときも引数エラーになります。
送信先の受信キューが満杯などの理由で、イベントが送信先に送られなかった場合は“イベントの送信に失敗した”ことを示す関数値を返します。

(9) READ_EVENT

呼出し形式　READ_EVENT( )

概　要　他チャネルから送られてきたイベントの内容を読み出します。

引数の説明

関数値　0 ～16777215 : イベント内容
–156　　: イベント未受信

使用例

```
RET=RECEIVE(0)
IF RET==10 THEN
   MSG=READ_EVENT( )
   PRINT("MESSAGE=[%X]¥N",MSG)
END
```

機能説明　他チャネルから送られてきたイベントの内容を知ることができます。他チャネルからのイベントの送出があると、そのイベントの発生をRECEIVE 関数により知ることができます。
RECEIVE 関数の関数値により、他チャネルからのイベントの受信があったのを確認した後、本関数を実行するとイベント内容を知ることができます。

(10) EXTRACT

呼出し形式　EXTRACT(n)

概　要　受信フレームの任意のオクテットの値を読み出します。

引数の説明　n　:読み出すデータのオクテット値
1～受信フレーム長の値が設定可能

関数値　0 ～255　:読み出したデータ
–60　:引数エラー
–80　:フレーム未受信エラー

使用例
```
CF1=EXTRACT(3)
AA=4
CF2=EXTRACT(AA)
```

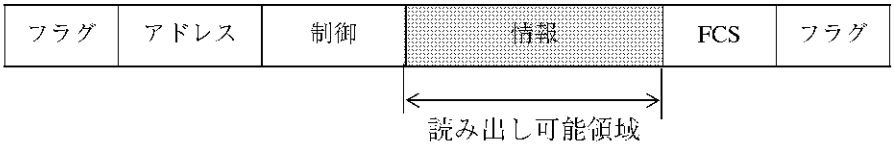
機能説明　受信したフレームの任意のオクテットの値を読み出す関数です。

注意事項　トランスペアレント・モードとレイヤ 2 自動モードとでは読み出せるデータの領域が違います。(下図参照)
下図で斜線部の領域が読み出し可能領域です。また、読み出す位置の指定は斜線部の左側から順に、1 から数えたオクテットで行います。

制限事項　引数で設定したオクテットのデータを関数値として返します。引数は、1 ～受信フレーム長までの値が設定可能です。それ以外を設定すると引数エラーになります。また、フレームを受信していないか RECEIVE 関数を実行していない状態でこの関数が実行されるとフレーム未受信エラーになります。

① トランスペアレント・モード

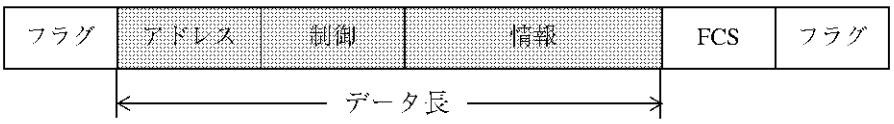
② レイヤ 2 自動モード



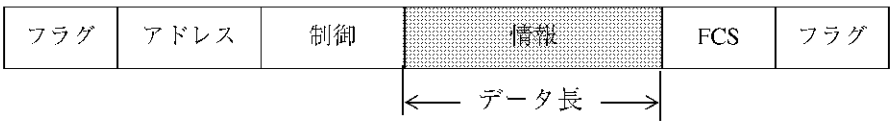
(11) RXFRLEN

呼出し形式	RXFRLEN( )
概　要	受信フレーム・データ長を調べます。
引数の説明	
関数値	1 ～512　　:受信フレームのデータ長 –80　　:フレーム未受信エラー
使用例	LENGTH=RXFRLEN( )
機能説明	受信フレームのデータ長を関数値として返します。
注意事項	トランスペアレント・モードとレイヤ 2 自動モードでは、データ長の定義が違います。(下図参照)
制限事項	この関数はフレームを受信していないか、またはフレームを受信していても RECEIVE 関数を実行していないと、フレーム未受信エラーとなります。

① トランスペアレント・モード



② レイヤ 2 自動モード



(12) WAIT

呼出し形式	WAIT(SEC)
概　要	指定の時間プログラムを停止します。
引数の説明	SEC　　: 待機する時間 100msec 単位 (0 ~ 16777215)
関数値	0　　: 正常終了 –60　　: 引数エラー
使用例	WAIT(10) SEC=5 WAIT(SEC)
機能説明	指定した時間プログラムの実行を停止します。

(13) PH_ACT

呼出し形式	PH_ACT( )
概　要	レイヤ 1 を起動させます。
引数の説明	
関数値	0　　: 正常終了 –160　　: レイヤ 1 エラー (起動しない) –161　　: インタフェース・エラー (PH_ACT をサポートしないインタフェースである)
使用例	PH_ACT( )
機能説明	レイヤ 1 の起動を行う関数です。
注意事項	レイヤ 1 の起動に失敗すると、レイヤ 1 エラーを示す関数値が返ります。

(14) PH_DEACT

呼出し形式　PH_DEACT( )

概　要　レイヤ 1 を停止させます。

引数の説明

関数値

0	:	正常終了
–53	:	NT モード・エラー (NT モードの宣言がされていない)
–56	:	基本インタフェース以外の宣言がされている
–160	:	レイヤ 1 エラー (停止しない)
–161	:	インタフェース・エラー (PH_DEACT をサポートしないインタフェースである)

使用例

```
PH_DEACT( )
```

機能説明　レイヤ 1 を停止する関数です。

制限事項　NT モード以外では、実行できません。

(15) READ_VAL

呼出し形式　READ_VAL( )

概　要　キー入力された数値を読み出します。

引数の説明

関数値

–2147483648 ～2147483647	:	キー入力された数値
–175	:	キー入力されていない数値

使用例

```
RET=RECEIVE(0)
IF RET==2 THEN
   VAL=READ_VAL( )
   PRINT("KEY VALUE=%D¥N",VAL)
END
```

機能説明　Command ラインから入力された数値を読み出します。Command ラインから入力された数値を読み出す関数として INPUT 関数がありますが、INPUT 関数によりキー入力受信待ち状態になっているときは、INPUT 関数が優先されます。したがって、RECEIVE 関数にキー入力の情報は渡されません。
INPUT 関数による受信待ち状態か、RECEIVE 関数による受信待ち状態かは、画面下部に表示されているアイコン表示で識別します。
(それぞれ、"INPUT","RCV?" と表示されます。)
Command ラインから入力できる数値は、–2147483648 ～2147483647 の範囲です。したがって、キー入力されていないときに返すエラー値 (–175) と実際に入力された値とで区別がつかないことがありますので、注意して下さい。

(16) REG_EVTID

呼出し形式　REG_EVTID(EVTID)

概　要　イベント ID を登録する関数です。

引数の説明　EVTID　: 登録するイベント ID
10 と 20 はそれぞれ D チャンネル、B チャンネル用に予約されているので使用できません。

関数値　0　: 正常に登録された
–60　: 引数エラー
–170　: イベント用のリソースに空きがなく登録できなかった

使用例

```
REG_EVTID(1000)
RET = RECEIVE(0)
IF RET==200 THEN
   EVT = READ_EVENT( )
   PRINT("EVENT=%X¥N", EVT)
END
```

機能説明　イベント ID は独立したシミュレータ間で、通信をする場合に使用する ID です。
2 つのシミュレータ間で通信をする場合、受信側のシミュレータは共通で認識するイベント ID を登録します。(送信側は イベント ID の登録をする必要はありません。ただし、送信側も受信する場合は別のイベント ID を登録します) 送信側は SEND_EVENT 関数で DEST にこのイベント ID を使用して送信します。受信側はイベントを受信したことを RECEIVE 関数で知ることができ、その内容については READ_EVENT 関数で読み出します。
登録することのできるイベント ID は他のシミュレータで登録されたものを含めシステム全体で 128 個までです。

(17) REL_EVTID

呼出し形式　　REL_EVTID(EVTID)

概　要　　登録してあるイベント ID を解放する関数です。

引数の説明　　EVTID　　: 解放するイベント ID
10 と 20 はそれぞれ D チャンネル、B チャンネル用に
予約されているので使用できません。

関数値　　0　　: 正常に解放された
–60　　: 引数で指定されたイベントは登録されていない

使用例

```
REG_EVTID(1000)
RET = RECEIVE(0)
IF RET==200 THEN
   EVT = READ_EVENT( )
   PRINT("EVENT=%X¥N", EVT)
END
REL_EVTID(1000)
```

機能説明　　登録されているイベント ID を解放する関数です。
登録できるイベント ID には限りがあるので、使用を終了したイベントは解放するようにして下さい。

(18) SEND_EVTS

呼出し形式　SEND_EVTS(EVENT_ID)

概　要　他チャネルにイベントを送ります。

引数の説明　EVENT_ID : 送信するイベント ID

関数値

0	: 正常終了
-60	: 引数エラー
-155	: イベントの送信に失敗した
-156	: 送信するイベントがない

使用例

```
WRITE_EVTS(1, 100)
WRITE_EVTS(2, 101)
SEND_EVTS(1234)
```

機能説明　他のチャネルにイベントを送ります。

送信するイベントは WRITE_EVTS() 関数で書き込みます。

WRITE_EVTS() 関数で設定したイベントは、一度送信するとクリアされます。しかし、エラーで終了した場合はクリアされません。

イベントを送信する関数として SEND_EVENT() 関数がありますが、SEND_EVENT() 関数が 1 ロングワードしか送信できないのに対し、本関数では、WRITE_EVTS() 関数で指定した数を一度に送信することができます。SEND_EVENT(DEST, EVENT) は以下のように置き換えることができます。

```
WRITE_EVTS(1, EVENT)
SEND_EVTS(DEST)
```

ここで送られるイベントを受信するためには、受信側でイベント ID を登録する必要があります。

(詳細は REG_EVTID, REL_EVTID を参照して下さい)

(19) WRITE_EVTS

呼び出し形式	WRITE_EVTS(N, EVENT)	
概 要	他チャネルに送信するイベントを書き込みます。	
引数の説明	N	: 書き込むイベントの位置（1 ～ 256）
	EVENT	: 書き込むイベントの値（-2147483648 ～ 2147483647）
関数値	0	: 正常終了
	-60	: 引数エラー
使用例	WRITE_EVTS(1, 100)	
	WRITE_EVTS(2, 101)	
	SEND_EVTS(1234)	
機能説明	他のチャネルに送信するイベントを書き込みます。 書き込んだイベントを送信するときは、SEND_EVTS() 関数を使用します。	

注意 WRITE_EVTS 関数では、-2147483648 ～ 2147483647 までを許容していますが、エラー値と同じ値を使用すると読み出す関数 (READ_EVTS) で識別が付きません。

(20) READ_EVTS

呼び出し形式　READ_EVTS(N)

概 要　他チャネルから送信されたイベントを読み出します。

引数の説明　N　：読み出すイベントの位置

関数値　-2147483648 ～ 2147483647：
読み出すイベント値

-60　：引数エラー

-156　：イベント未受信エラー

使用例

```
REG_EVTID(1234)
RET = RECEIVE(0)
IF RET == 200 THEN
  N = COUNT_EVT()
  FOR I=1 TO N DO
    EVENT = READ_EVTS(I)
    PRINT("RECEIVED EVENT = %D¥N", EVENT)
  END
END
REL_EVTID(1234)
```

機能説明　他のチャネルから送信されたイベントを読み出します。

読み出すイベントは引数で指定します。

同様の関数に READ_EVENT() 関数がありますが、READ_EVENT() 関数が 1 ロングワードしか読み出すことができないのに対し、本関数は引数で指定することにより複数のロングワードを読み出すことができます。READ_EVENT() 関数は READ_EVTS(1) 関数で置き換えることができます。

注意　READ_EVTS 関数は引数で指定することにより複数のロングワードを読み出すことができますが、エラー値との識別が付かなくなることがあります。

---

(21) COUNT_EVT

呼び出し形式　COUNT_EVT()

概 要　他チャネルから送信されたイベント数を得ます。

引数の説明

関数値　1 ～ 256　: 受信したイベント数

-156　: イベント未受信エラー

使用例

```
REG_EVTID(1234)
RET = RECEIVE(0)
IF RET == 200 THEN
  N = COUNT_EVT()
  FOR I=1 TO N DO
    EVENT = READ_EVTS(I)
    PRINT("RECEIVED EVENT = %D¥N", EVENT)
  END
END
REL_EVTID(1234)
```

機能説明　他チャネルから送信されたイベント数を得ます。

(22) CONN_BCH

呼び出し形式　CONN_BCH(IF1, CH1, IF2, CH2)

概 要　B チャネル回線を接続します。

引数の説明

IF1　: レイヤ 1 インタフェース ID

上位 16 ビット

基本インタフェース : 0x0181

一次群インタフェース : 0x0182

U 点インタフェース : 0x0183

下位 16 ビット

インタフェースシリアル番号

CH1　: IF1 で指定したインタフェースのチャネル番号

0: チャネルを設定していない状態

1: B1 チャネル

2: B2 チャネル

:

24: B24

IF2　: レイヤ 1 インタフェース ID

CH2　: IF2 で指定したインタフェースのチャネル番号

関数値

0 ～ 16777215:
チャネル接続ハンドル

-60　: 引数エラー

-163　: 使用しているインタフェースに指定のチャネルがない

-180　: 指定したチャネルが既に使用されている

使用例　U 点インタフェース 1（UI0-1）の B1 チャネルと U 点インタフェース 2(UI0-2) の B2 チャネルを接続する例を以下に示します。

```
CH_HANDLE = CONN_BCH(0x01810001, 1, 0x01810002, 2)
```

この接続を解放するには以下のように記述します。

```
DISC_BCH(CH_HANDLE)
```

機能説明　引数 IF1 で指定したインタフェースの引数 CH1 で指定したチャネルと引数 IF2 で指定したインタフェースの引数 CH2 で指定したチャネルを接続します。正常に接続された場合は関数値として、接続ハンドルを返します。この接続ハンドルは接続したチャネルを解放するために使用されます。チャネルを解放する関数は DISC_BCH です。チャネルを接続したままにしておくと他のシミュレータでそのチャネルが使用できなくなるので、使用を終えたチャネルは必ず解放して下さい。

また、現在実行中のシミュレータが使用しているレイヤ 1 インタフェース ID は GET_LAY1ID 関数で取得することができます。

(23) DISC_BCH

呼び出し形式	DISC_BCH(HANDLE)
概 要	B チャネル回線を解放します。
引数の説明	HANDLE : 接続のときに受け取ったチャネル接続ハンドル
関数値	0 : 正常終了
	-60 : 引数エラー
	-163 : 指定したチャネル接続ハンドルは使用されていない
使用例	CH_HANDLE = CONN_BCH(0x01810001, 1, 0x01810002, 2) DISC_BCH(CH_HANDLE)
機能説明	CONN_BCH 関数を使用して接続したチャネルを解放する関数です。チャネルを解放するためには、CONN_BCH 関数でチャネルを接続したときに返されるチャネル接続ハンドルを引数で与える必要があります。

(24) GET_SIMID

呼び出し形式	GET_SIMID()
概 要	現在実行中のシミュレーション ID を得ます。
引数の説明	
関数値	0 ～ 268435455：シミュレーション ID 上位 16 ビット LAPD シミュレータ：0x0281 LAPB シミュレータ：0x0282 下位 16 ビット シミュレーション番号
使用例	SID = GET_SIMID() PRINT("MY SIMULATION ID = [%X]¥N", SID)
機能説明	現在実行中のシミュレーション ID を得ます。ここで得るシミュレーション ID は本器でユニークな番号になります。 シミュレーション番号は、例えば LAPD シミュレータ x における x の値で、シミュレータ ID の下位 16 ビットに使われます。

(25) GET_LAY1ID

呼び出し形式　GET_LAY1ID()

概 要　現在実行中のレイヤ 1 インタフェース ID を得ます。

引数の説明

関数値　0 ～ 268435455：インタフェース ID

上位 16 ビット

基本インタフェース：0x0181

一次群インタフェース：0x0182

U 点インタフェース：0x0183

下位 16 ビット

レイヤ 1 インタフェース番号

使用例

```
LAY1_ID = GET_LAY1ID()
PRINT("MY LAYER1 ID = [%X]¥N", LAY1_ID)
```

機能説明　現在実行中のシミュレータが使用するレイヤ 1 インタフェース ID を得ます。ここで得るレイヤ 1 インタフェース ID は本器でユニークな番号になります。

レイヤ 1 インタフェース番号は、例えば UIx-y における y の値で、レイヤ 1 インタフェース ID の下位 16 ビットに使われます。

8.2 トランスペアレント・モード用関数

表 8-2 にトランスペアレント・モード用関数の一覧を示します。

表 8-2 トランスペアレント・モード用関数

フレーム送信関連	
(1) SENDT	フレームを送信する。(メッセージ・データをそのまま送出)
(2) SENDF	フレームを送信する。(N(S),N(R),P/F を挿入して送出)
(3) INCVS	送信状態変数 V(S) を＋１する。
(4) INCVR	受信状態変数 V(R) を＋１する。
(5) SETVS	送信状態変数 V(S) の値を設定する。
(6) SETVR	受信状態変数 V(R) の値を設定する。
(7) INS_SAPI	送信フレームの SAPI 値を書き換える。
(8) INS_TEI	送信フレームの TEI 値を書き換える。
(9) INS_CR	送信フレームの C/R ビット値を書き換える。
(10) INS_NR	送信フレームの N(R) 値を書き換える。
(11) INS_NS	送信フレームの N(S) 値を書き換える
(12) INS_PF	送信フレームの P/F ビット値を書き換える。
(13) INS_TYPE	送信フレームのフレーム種別を書き換える。
(14) INS_CF1	送信フレーム制御フィールドの第１オクテットの値を書き換える。
(15) INS_CF2	送信フレーム制御フィールドの第２オクテットの値を書き換える。
(16) INS_FRCF1	送信 FRMR フレーム情報フィールド内の第１オクテット値を書き換える。
(17) INS_FRCF2	送信 FRMR フレーム情報フィールド内の第２オクテット値を書き換える。
(18) INS_FRVS	送信 FRMR フレーム情報フィールド内の V(S) 値を書き換える。
(19) INS_FRVR	送信 FRMR フレーム情報フィールド内の V(R) 値を書き換える。
(20) INS_FRCR	送信 FRMR フレーム情報フィールド内の C/R ビット値を書き換える。
(21) INS_FRWXYZ	送信 FRMR フレーム情報フィールド内の WXYZ ビット値を書き換える。

フレーム受信関連	
(22) RXSAPI	受信フレームの SAPI 値を読み出す。
(23) RXTEI	受信フレームの TEI 値を読み出す。
(24) RXCR	受信フレームの C/R ビット値を読み出す。
(25) RXNR	受信フレームの N(R) 値を読み出す。
(26) RXNS	受信フレームの N(S) 値を読み出す。
(27) RXPF	受信フレームの P/F ビット値を読み出す。
(28) RXTYPE	受信フレームのフレーム種別を読み出す。
(29) RXCF1	受信フレーム制御フィールドの第１オクテットの値を読み出す。
(30) RXCF2	受信フレーム制御フィールドの第２オクテットの値を読み出す。
(31) RXFRCF1	受信 FRMR フレーム情報フィールド内の第１オクテット値を読み出す。
(32) RXFRCF2	受信 FRMR フレーム情報フィールド内の第２オクテット値を読み出す。
(33) RXFRVS	受信 FRMR フレーム情報フィールド内の V(S) 値を読み出す。
(34) RXFRVR	受信 FRMR フレーム情報フィールド内の V(R) 値を読み出す。
(35) RXFRCR	受信 FRMR フレーム情報フィールド内の C/R ビット値を読み出す。
(36) RXFRWXYZ	受信 FRMR フレーム情報フィールド内の WXYZ ビット値を読み出す。

次ページ以降に各関数の説明をします。

(1) SENDT

呼出し形式	SENDT("NAME")
概　要	フレームを送信します。(メッセージ・データをそのまま送出)
引数の説明	NAME　　: フレーム名
関数値	0　　: 正常終了 –50　　: トランスペアレント・モード・エラー –61　　: フレーム名エラー
使用例	SENDT("SPL1")
機能説明	トランスペアレント・モードでのフレームの送出を行う関数です。 引数で指定した名前のフレームをそのまま送出します。
注意事項	SENDF 関数では、内部の状態変数 V(S),V(R) を N(S),N(R) としてフレーム中に挿入して送出しますが、本関数は一切加工せずに回線上に送出します。
制限事項	レイヤ 2 自動モードでは、実行できません。

(2) SENDF

呼出し形式　SENDF("NAME",PF)

概　要　フレームを送信します。(N(S),N(R),P/F を挿入して送出)

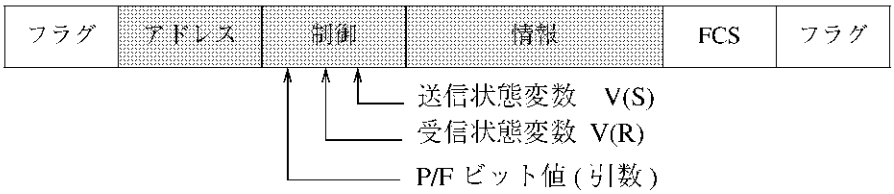
引数の説明
NAME　: 送出するフレーム名
PF　: 送出するフレームの P/F ビット値 (0, 1)

関数値
0　: 正常終了
–50　: トランスペアレント・モード・エラー
–60　: 引数エラー
–61　: フレーム名エラー

使用例
```
SENDF("ID_REQ",0)
PF= 1
SENDF("SABME",PF)
```

機能説明　トランスペアレント・モードでのフレームの送出を行う関数です。
引数で指定されたフレーム名のフレームを送出しますが、その際、引数で指定された P/F ビット値や、送信シーケンス番号 N(S)、受信シーケンス番号 N(R) を付加して送信します。
内部の状態変数 V(S)、V(R) を、N(S)、N(R) としてフレームに挿入します。
送信状態変数 V(S) は、INCVS 関数や SETVS 関数により変更が可能です。
また、受信状態変数 V(R) も、同様に INCVR 関数や SETVR 関数により変更が可能です。
SENDF 関数によりフレーム送信する際に付加されるデータを、下図に示します。

制限事項　レイヤ 2 自動モードでは、実行できません。



(3) INCVS

呼出し形式	INCVS( )
概　要	送信状態変数 V(S) を＋1します。
引数の説明	
関数値	0　　：正常終了 –50　　：トランスペアレント・モード・エラー
使用例	INCVS( )
機能説明	送信状態変数 V(S) の値を＋1更新する関数です。 この値は、番号制情報フレーム (I フレーム) を送出するときにのみ N(S) 値として送信フレームに付加されます。
制限事項	この関数は、レイヤ2自動モードでは実行できません。

(4) INCVR

呼出し形式	INCVR( )
概　要	受信状態変数 V(R) を＋1します。
引数の説明	
関数値	0　　：正常終了 –50　　：トランスペアレント・モード・エラー
使用例	INCVR( )
機能説明	受信状態変数 V(R) の値を＋1更新する関数です。 この値は、番号制情報フレーム (I フレーム) と番号制監視フレーム (S フレーム) を送出する時に N(R) 値として、送信フレームに付加されます。
制限事項	この関数は、レイヤ2自動モードでは実行できません。

(5) SETVS

呼出し形式　SETVS(VS)

概　要　送信状態変数 V(S) の値を設定します。

引数の説明　VS　: 設定する V(S) 値（0 ～127)

関数値
0　: 正常終了
–50　: トランスペアレント・モード・エラー
–60　: 引数エラー

使用例
```
SETVS(5)
VS=6
SETVS(VS)
```

機能説明　送信状態変数 V(S) の値を設定する関数です。
この値は、番号制情報フレーム (I フレーム) を送出するときにのみ N(S) 値として送信フレームに付加されます。

制限事項　この関数は、レイヤ 2 自動モードでは実行できません。

(6) SETVR

呼出し形式　SETVR(VR)

概　要　受信状態変数 V(R) の値を設定します。

引数の説明　VR　: 設定する V(R) 値（0 ～127)

関数値
0　: 正常終了
–50　: トランスペアレント・モード・エラー
–60　: 引数エラー

使用例
```
SETVR(5)
VR=6
SETVR(VR)
```

機能説明　受信状態変数 V(R) の値を設定する関数です。
この値は、番号制情報フレーム (I フレーム) と番号制監視フレーム (S フレーム) を送出するときにのみ、N(R) 値として送信フレームに付加されます。

制限事項　この関数は、レイヤ 2 自動モードでは実行できません。

(7) ~ (21)　LAPD パラメータ挿入関数群

呼出し形式

[関数名]	[挿入するパラメータ]
(7) INS_SAPI("NAME",SAPI)	SAPI 値
(8) INS_TEI("NAME",TEI)	TEI 値
(9) INS_CR("NAME",CR)	C/R ビット値
(10) INS_NR("NAME",NR)	N(R)
(11) INS_NS("NAME",NS)	N(S)
(12) INS_PF("NAME",PF)	P/F ビット値
(13) INS_TYPE("NAME",TYPE)	フレーム種別
(14) INS_CF1("NAME",CF1)	制御フィールド第 1 オクテット値
(15) INS_CF2("NAME",CF2)	制御フィールド第 2 オクテット値
(16) INS_FRCF1("NAME",FRCF1)	FRMR　フレーム情報フィールド内第 1 オクテット値
(17) INS_FRCF2("NAME",FRCF2)	FRMR　フレーム情報フィールド内第 2 オクテット値
(18) INS_FRVS("NAME",FRVS)	FRMR　フレーム情報フィールド内 V(S) 値
(19) INS_FRVR("NAME",FRVR)	FRMR　フレーム情報フィールド内 V(R) 値
(20) INS_FRCR("NAME",FRCR)	FRMR　フレーム情報フィールド内 C/R ビット値
(21) INS_FRWXYZ("NAME",FRWXYZ)	FRMR　フレーム情報フィールド内 WXYZ ビット値

概　要　　送出フレームの任意の LAPD パラメータを書き換えます。

引数の説明

NAME	: フレーム名
SAPI	: SAPI 値
TEI	: TEI 値
CR	: C/R ビット値
NR	: N(R)
NS	: N(S)
PF	: P/F ビット値
TYPE	: フレーム種別
CF1	: 制御フィールド第 1 オクテット値
CF2	: 制御フィールド第 2 オクテット値
FRCF1	: FRMR フレーム情報フィールド内第 1 オクテット値
FRCF2	: FRMR フレーム情報フィールド内第 2 オクテット値
FRVS	: FRMR フレーム情報フィールド内 V(S) 値
FRVR	: FRMR フレーム情報フィールド内 V(R) 値
FRCR	: FRMR フレーム情報フィールド内 C/R ビット値
FRWXYZ	: FRMR フレーム情報フィールド内 WXYZ ビット値

関数値

0	: 正常終了
–50	: トランスペアレント・モードエラー
–60	: 引数エラー
–61	: フレーム名エラー
–85	: 書き換えられるメッセージが不適切

使用例

```
INS_TEI("SABME",64)
SENDF("SABME",1)
```

機能説明

メッセージ中の任意の LAPD パラメータを挿入する関数です。
本関数群は、メッセージの先頭から内容を解釈していき挿入する LAPD パラメータの位置を探していきます。
挿入する LAPD パラメータの位置が発見されるとそこを引数で指定された値に書き換えます。
このとき挿入する LAPD パラメータの位置が発見されない場合には、" 書き換えられるメッセージが不適切である " という意味の関数値を返します。

補足説明

INS_TYPE("NAME",TYPE) 関数の引数 TYPE とフレーム種別は、以下の対応となっています。

フレーム種別	TYPE		フレーム種別	TYPE	
	10 進	16 進		10 進	16 進
I	0	00	UI	3	03
RR	1	01	DISC	67	43
RNR	5	05	UA	99	63
REJ	9	09	FRMR	135	87
SABME	111	6F	XID	175	AF
DM	15	0F			

制限事項

本関数群は、トランスペアレント・モードで実行する関数で、レイヤ 2 自動モードでは実行できません。

(22) ~ (36) LAPD パラメータ読み出し関数群

呼出し形式

[関数名]	[読み出すパラメータ]
(22) RXSAPI( )	SAPI 値
(23) RXTEI( )	TEI 値
(24) RXCR( )	C/R ビット値
(25) RXNR( )	N(R) 値
(26) RXNS( )	N(S) 値
(27) RXPF( )	P/F ビット値
(28) RXTYPE( )	フレーム種別
(29) RXCF1( )	制御フィールド第 1 オクテット値
(30) RXCF2( )	制御フィールド第 2 オクテット値
(31) RXFRCF1( )	FRMR フレーム情報フィールド内第 1 オクテット値
(32) RXFRCF2( )	FRMR フレーム情報フィールド内第 2 オクテット値
(33) RXFRVS( )	FRMR フレーム情報フィールド内 V(S) 値
(34) RXFRVR( )	FRMR フレーム情報フィールド内 V(R) 値
(35) RXFRCR( )	FRMR フレーム情報フィールド内 C/R ビット値
(36) RXFRWXYZ( )	FRMR フレーム情報フィールド内 WXYZ ビット値

概　要　受信フレームの任意の LAPD パラメータを読み出す関数群です。

引数の説明

関数値

0 ~ 255	: 受信フレームのパラメータ
–50	: トランスペアレント・モード・エラー
–80	: フレーム未受信エラー
–81	: 受信フレームが FRMR フレームでない
–82	: 読み出すパラメータが存在しない

使用例

```
RECEIVE(0)
TEI=RXTEI( )
PRINT("TEI=%D¥N",TEI)
```

機能説明

受信フレームの LAPD パラメータを読み出す関数群です。
受信したフレーム順に読み出すことができます。
受信したフレームを読み出したい場合は、RECEIVE 関数を実行し、受信フレームを内部に取り込んでから本関数群を実行します。次の受信フレームを読み出したい場合は、再び RECEIVE 関数を実行します。このとき受信フレームがない場合は、受信待ち状態になります。

注意事項

受信したフレームがない状態や RECEIVE 関数を実行していない状態で本関数が実行されるとフレーム未受信エラーとなります。
また本関数群は、受信したフレームを先頭から解釈していき LAPD パラメータのある位置を探していきます。LAPD パラメータのある位置が発見されるとその値を関数値として返し、終了します。
LAPD パラメータが見つからない場合には、" 読み出すパラメータが存在しない " という意味の関数値を返します。

補足説明

(31) ~ (36) の RXFR で始まる LAPD パラメータ読み出し関数において受信フレームが FRMR フレームでない場合、" 受信フレームが FRMR フレームでない " という意味の関数値を返します。
RXTYPE( ) 関数で返される関数値とフレーム種別は、以下の対応となっています。

フレーム種別	関数値		フレーム種別	関数値	
	10 進	16 進		10 進	16 進
I	0	00	UI	3	03
RR	1	01	DISC	67	43
RNR	5	05	UA	99	63
REJ	9	09	FRMR	135	87
SABME	111	6F	XID	175	AF
DM	15	0F			

制限事項

本関数はトランスペアレント・モードで実行する関数でレイヤ 2 自動モードでは実行できません。

8.3 レイヤ 2 自動モード用関数

表 8-3 にレイヤ 2 自動モード用関数の一覧を示します。

表 8-3 レイヤ 2 自動モード用関数 (1/3)

フレーム送信関連	
(1) SENDI	I フレームを送信する。
(2) SENDUI	UI フレームを送信する。
(3) SENDXIDC	XID コマンドを送信する。
(4) SENDXIDR	XID レスポンスを送信する。
(5) SENDPKT	パケットを送信する。
(6) INCPS	送信順序番号 P(S) を＋1 にする。
(7) INCPR	受信順序番号 P(R) を＋1 にする。
(8) SETPS	送信順序番号 P(S) の値を設定する。
(9) SETPR	受信順序番号 P(R) の値を設定する。
(10) INS_PD	送信フレームのプロトコル識別子を書き換える。
(11) INS_CRL	送信フレームの呼番号長を書き換える。
(12) INS_CRF	送信フレームの呼番号フラグ値を書き換える。
(13) INS_CRV	送信フレームの呼番号値を書き換える。
(14) INS_MSG	送信フレームのメッセージ種別を書き換える。
(15) INS_INFO	送信フレームの情報要素群フィールドを書き換える。
(16) INS_CS_VAL	送信フレームの理由表示値を書き換える。
(17) INS_GFI	送信パケットのゼネラル・フォーマット識別子の値を書き換える。
(18) INS_Q	送信パケットの Q ビット値を書き換える。
(19) INS_D	送信パケットの D ビット値を書き換える。
(20) INS_LCGN	送信パケットの論理チャネル・グループ番号を書き換える。
(21) INS_LCN	送信パケットの論理チャネル番号を書き換える。
(22) INS_TYP	送信パケットのパケット・タイプ識別子を書き換える。
(23) INS_PR	送信パケットの受信順序番号 P(R) を書き換える。
(24) INS_PS	送信パケットの送信順序番号 P(S) を書き換える。
(25) INS_M	送信パケットのモア・データ表示値を書き換える。
(26) INS_CLL	送信パケットの発呼ユーザ・アドレス長を書き換える。
(27) INS_CDL	送信パケットの着呼ユーザ・アドレス長を書き換える。
(28) INS_DA	送信パケットの着呼ユーザ・アドレスを書き換える。
(29) INS_SA	送信パケットの発呼ユーザ・アドレスを書き換える。
(30) INS_FL	送信パケットのファシリティ長を書き換える。
(31) INS_F	送信パケットのファシリティを書き換える。
(32) IN_ CAUSE	送信パケットの切断 / リスタート / リセット原因を書き換える。
(33) INS_DIAG	送信パケットの診断符号を書き換える。
(34) SETDTL	送信パケット内のデータ長を書き換える。
(35) INS_DATA	送信パケット内のデータを書き換える。

表 8-3 レイヤ 2 自動モード用関数 (2/3)

TEI 管理手順関連	
(36) REQ_TEI	TEI 割当手順を起動する。(TE モード)
(37) CHKREQ_TEI	TEI チェック手順を行う。(NT モード)
(38) REMOVE_TEI	TEI 解除手順を行う。(NT モード)
(39) VERIFY_TEI	TEI 検証手順を行う。(TE モード)
リンク ,SAPI,TEI 管理関連	
(40) LINKON	リンクの設定を行う。
(41) LINKOFF	リンクの解放を行う。
(42) WAIT_LINK	相手局からのリンク設定待ち状態にする。
(43) L_STATUS	リンクが設定されているかを調べる。
(44) SET_BUSY	自局をビジー状態にする
(45) REL_BUSY	自局のビジー状態を解除する。
(46) PROHIBIT_L	新たなリンクの設定を禁止する。
(47) PERMIT_L	リンク設定不許可状態を解除する。
(48) REG_TEI	TEI 値を本器に登録し、その TEI 値のフレームを受信可能にする。
(49) REL_TEI	TEI 値を解除し、その TEI 値のフレームを受信しないようにする。
(50) NEXT_TEI	新たな TEI 値を使用することを宣言する。(TE モード)
(51) ACT_SAPI	フレーム送出時に使用される SAPI 値を設定する。
(52) ACT_TEI	フレーム送出時に使用される TEI 値を設定する。
(53) ACT_LINK	フレーム送出時に使用されるリンク番号を設定する。
(54) LOCK_SAPI	SAPI 値が自動で設定変更されるのを禁止する。
(55) LOCK_TEI	TEI 値が自動で設定変更されるのを禁止する。
(56) LOCK_LINK	リンク番号が自動で設定変更されるのを禁止する。
(57) FLEX_SAPI	SAPI 値が自動で設定変更されるようにする。
(58) FLEX_TEI	TEI 値が自動で設定変更されるようにする。
(59) FLEX_LINK	リンク番号が自動で設定変更されるようにする。
(60) GET_SAPI	フレーム送出時に使用される SAPI 値を調べる。
(61) GET_TEI	フレーム送出時に使用される TEI 値を調べる。
(62) GET_LINK	フレーム送出時に使用されるリンク番号を調べる。
(63) SEE_LINK	現在設定されているリンク番号を調べる。

表 8-3 レイヤ 2 自動モード用関数 (3/3)

フレーム受信関連	
(64) RXSAPI	受信フレームの SAPI 値を読み出す。
(65) RXTEI	受信フレームの TEI 値を読み出す。
(66) RXTYPE	受信フレームのフレーム種別を読み出す。
(67) RXPF	受信フレームの P/F ビット値を読み出す。
(68) RXPD	受信フレームのプロトコル識別子を読み出す。
(69) RXCRL	受信フレームの呼番号長を読み出す。
(70) RXCRF	受信フレームの呼番号フラグ値を読み出す。
(71) RXCRV	受信フレームの呼番号値を読み出す。
(72) RXMSG	受信フレームのメッセージ種別を読み出す。
(73) RXINFO_NUM	受信フレームの情報要素数を読み出す。
(74) RXINFO_ELM	受信フレームの情報要素識別子を読み出す。
(75) RXINFO_LEN	受信フレームの情報要素内容長を読み出す。
(76) RXINFO_VAL	受信フレームの情報内容を読み出す。
(77) RXINFO_POS	受信フレームの情報要素識別子の位置を調べる。
(78) RXCS_VAL	受信フレームの理由表示値を読み出す。
(79) RXCHAN_NUM	受信フレームのチャネル番号を読み出す。
(80) RXGFI	受信パケットのゼネラル・フォーマット識別子の値を読み出す。
(81) RXQ	受信パケットの Q ビット値を読み出す。
(82) RXD	受信パケットの D ビット値を読み出す。
(83) RXLCGN	受信パケットの論理チャネル・グループ番号を読み出す。
(84) RXLCN	受信パケットの論理チャネル番号を読み出す
(85) RXTYP	受信パケットのパケット・タイプ識別子を読み出す。
(86) RXPR	受信パケットの受信順序番号 P(R) を読み出す。
(87) RXPS	受信パケットの送信順序番号 P(S) を読み出す。
(88) RXM	受信パケットのモア・データ表示値を読み出す。
(89) RXCLL	受信パケットの発呼ユーザ・アドレス長を読み出す。
(90) RXCDL	受信パケットの着呼ユーザ・アドレス長を読み出す。
(91 RXDA	受信パケットの着呼ユーザ・アドレスを読み出す。
(92) RXSA	受信パケットの発呼ユーザ・アドレスを読み出す。
(93) RXFL	受信パケットのファシリティ長を読み出す。
(94) RXF	受信パケットのファシリティを読み出す。
(95) RXCAUSE	受信パケットの切断 / リスタート / リセット原因を読み出す。
(96) RXDIAG	受信パケットの診断符号を読み出す。
(97) RXDTL	受信パケット内のデータ長を読み出す。
(98) RXDATA	受信パケット内のデータを読み出す。

次ページ以降に各関数の説明をします。

(1) SENDI

呼出し形式　SENDI("NAME")

概　要　I フレームを送信します。

引数の説明　NAME　: 送出するフレーム名

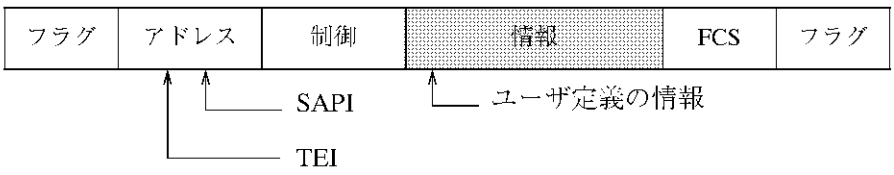
関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–61　: フレーム名エラー
–71　: TEI エラー
–72　: リンク未設定エラー
–110　: 送出フレーム長エラー

使用例　SENDI("SETUP")

機能説明　レイヤ 2 自動モードで、I フレームの送出を行う関数です。
引数で指定されたフレーム名の I フレームを送出しますが、その際、アドレスには、あらかじめ設定された SAPI 値と TEI 値を使います。(下図参照)

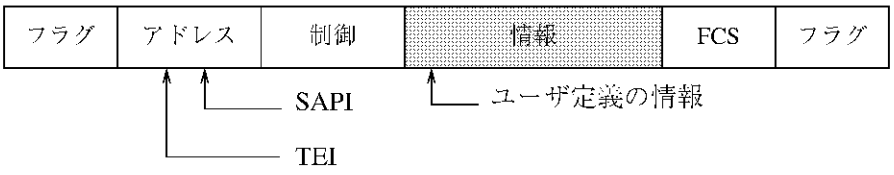
注意事項　I フレームを送出するためには、まずリンクの設定を行わなければなりません。
リンクの設定は LINKON 関数を用いて行うか、相手局からのリンク設定要求を受け付けて行います。
リンク設定が行われていない SAPI, TEI ペアを用いてこの関数が実行されると、リンク未設定エラーになります。
I フレームは一番最近リンクが設定された SAPI, TEI ペアを用いて通常は送出されますが、一度設定した SAPI, TEI ペアを固定で変更したくない場合は、それぞれ LOCK_SAPI, LOCK_TEI 関数または LOCK_LINK 関数を使用します。

制限事項　以前に設定したリンクについてのフレームを送出したい場合には、ACT_SAPI, ACT_TEI 関数を用いて正しく設定して下さい。
TEI 値に未登録な TEI が設定された状態で、この関数が実行されると TEI エラーになります。
TEI 値の登録は REG_TEI 関数を用いて行うことができます。
引数で指定されたフレーム名のフレームがない場合や、送信フレームのフレーム長が短い場合にはエラーになります。
この関数は、トランスペアレント・モードでは使用できません。



(2) SENDUI

呼出し形式	SENDUI("NAME")
概　要	UI フレームを送信します。
引数の説明	NAME　　: 送出するフレーム名
関数値	0　　: 正常終了 –51　　: レイヤ２自動モード・エラー –61　　: フレーム名エラー –71　　: TEI エラー –110　　: 送出フレーム長エラー
使用例	SENDUI("IDCHK")
機能説明	レイヤ２自動モードで、UI フレームの送出を行う関数です。 引数で指定されたフレーム名の UI フレームを送出しますが、その際、アドレスにはあらかじめ設定された SAPI 値と TEI 値を使います。 (下図参照)
注意事項	SAPI 値と TEI 値の初期値は、それぞれ 0 、127 です。 SAPI 値と TEI 値の設定には、それぞれ ACT_SAPI, ACT_TEI 関数を用いて行います。 ただし、TEI 値の設定は、TEI の割当手順が行われた場合や、リンクの設定が行われた場合にも生じます。 また、SAPI 値の設定もリンクの設定が行われた場合に生じます。 新たな SAPI、TEI の設定を禁止したい場合は、それぞれ LOCK_SAPI, LOCK_TEI 関数を用いて行います。
制限事項	TEI 値に未登録な TEI が設定された状態で、この関数が実行されると TEI エラーになります。 TEI 値の登録は REG_TEI 関数を用いて行うことができます。 引数で指定されたフレーム名のフレームがない場合や、送信フレームのフレーム長が短い場合にはエラーになります。 この関数は、トランスペアレント・モードでは使用できません。



(3) SENDXIDC

呼出し形式　SENDXIDC("NAME")

概　要　XID コマンドを送信します。

引数の説明　NAME　: 送出するフレーム名

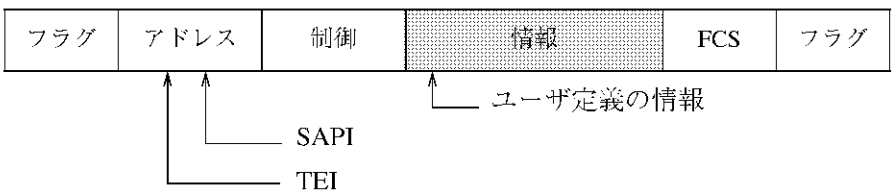
関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–61　: フレーム名エラー
–71　: TEI エラー
–110　: 送出フレーム長エラー

使用例　SENDXIDC("XID1")

機能説明　レイヤ 2 自動モードで、XID コマンドの送出を行う関数です。
引数で指定されたフレーム名の XID コマンドを送出しますが、その際、アドレスにはあらかじめ設定された SAPI 値と TEI 値を使います。(下図参照)

注意事項　SAPI 値と TEI 値の初期値は、それぞれ 0 、127 です。
SAPI 値と TEI 値の設定には、それぞれ ACT_SAPI, ACI_TEI 関数を用いて行います。
ただし、TEI 値の設定は、TEI の割当手順が行われた場合や、リンクの設定が行われた場合にも生じます。
また、SAPI 値の設定もリンクの設定が行われた場合に生じます。
新たな SAPI, TEI の設定を禁止したい場合は、それぞれ LOCK_SAPI, LOCK_TEI 関数を用いて行います。

制限事項　TEI 値に未登録な TEI が設定された状態で、この関数が実行されると TEI エラーになります。
TEI 値の登録は REG_TEI 関数を用いて行うことができます。
引数で指定されたフレーム名のフレームがない場合や、送信フレームのフレーム長が短い場合にはエラーになります。
この関数は、トランスペアレント・モードでは使用できません。



(4) SENDXIDR

呼出し形式　SENDXIDR("NAME")

概　要　XID レスポンス送信します。

引数の説明　NAME　: 送出するフレーム名

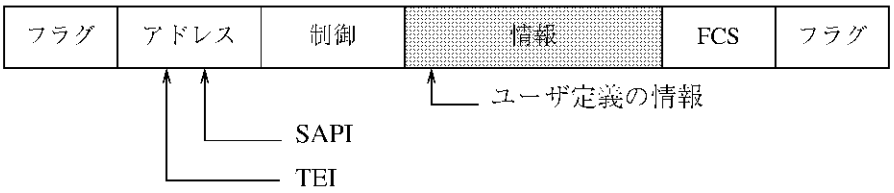
関数値
0　: 正常終了
–51　: レイヤ２自動モード・エラー
–61　: フレーム名エラー
–71　: TEI エラー
–110　: 送出フレーム長エラー

使用例　SENDXIDR("XID2")

機能説明　レイヤ２自動モードで、XIR レスポンスの送出を行う関数です。
引数で指定されたフレーム名の XID レスポンスを送出しますが、その際、アドレスにはあらかじめ設定された SAPI 値と TEI 値を使います。
(下図参照)

注意事項　SAPI 値と TEI 値の初期値は、それぞれ 0 、127 です。
SAPI 値と TEI 値の設定には、それぞれ ACT_SAPI, ACT_TEI 関数を用いて行います。
ただし、TEI 値の設定は、TEI の割当手順が行われた場合や、リンクの設定が行われた場合にも生じます。
また、SAPI 値の設定もリンクの設定が行われた場合に生じます。
新たな SAPI, TEI の設定を禁止したい場合は、それぞれ LOCK_SAPI, LOCK_TEI 関数を用いて行います。

制限事項　TEI 値に未登録な TEI が設定された状態で、この関数が実行されると TEI エラーになります。
TEI 値の登録は REG_TEI 関数を用いて行うことができます。
引数で指定されたフレーム名のフレームがない場合や、送信フレームのフレーム長が短い場合にはエラーになります。
この関数は、トランスペアレント・モードでは使用できません。



(5) SENDPKT

呼出し形式　SENDPKT("NAME", LCGN, LCN)

概　要　パケットを送信します。

引数の説明
NAME　: フレーム名
LCGN　: 論理チャネルグループ番号 (0 ~ 15)
LCN　: 論理チャネル番号 (0 ~ 255)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー
–61　: フレーム名エラー
–71　: TEI エラー
–72　: リンク未設定エラー
–110　: 送出フレーム長エラー

使用例
```
PH_ACT( )
LINKON( )
LCGN=0
LCN=1
SENDPKT("CR",LCGN,LCN)
```

機能説明　レイヤ 2 自動モードで、パケットフレームを送出する関数です。
引数で指定した名前のメッセージを I フレームとして送出します。
このとき引数で指定した LCGN と LCN を送出するメッセージに挿入します。また、送出するメッセージのタイプが DT パケットのときは、引数 LCGN,LCN で指定された論理チャネルの P(R),P(S) が挿入されます。同様、RR,RNR パケットの時も指定された論理チャネルの P(R) がメッセージ中に挿入されます。
メッセージは、あらかじめメッセージ・ビルダで作成しておく必要があります。作成したメッセージに名前は必ず付けて下さい。
本関数でフレームを送出するためにはレイヤ 2 リンクが設定されていなければなりません。リンクの設定は LINKON( ) 関数を用いて行うか、相手局からのリンク設定要求を受け付けて行います。

注意事項　本関数でフレームが送出されない場合以下の点をチェックして下さい。
1. レイヤ 1 は起動しているか。
2. レイヤ 2 リンクは設定されているか。
3. メッセージ・ビルダで引数で指定した名前のメッセージを作成してあるか。

(6) INCPS

呼出し形式　INCPS(LCGN, LCN)

概　要　送信順序番号 P(S) を＋ 1 します。

引数の説明
LCGN　: 論理チャネルグループ番号 (0 ~ 15)
LCN　: 論理チャネル番号 (0 ~ 255)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
INCPS(0,1)
SENDPKT("DT",0,1)
```

機能説明　送信順序番号 P(S) を＋ 1 更新する関数でする。
引数で指定した論理チャネル (LCGN と LCN) の P(S) 値を変更します。
この値は、DT パケットを送出するときにメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(7) INCPR

呼出し形式　INCPR(LCGN, LCN)

概　要　受信順序番号 P(R) を＋ 1 します。

引数の説明
LCGN　: 論理チャネルグループ番号 (0 ~ 15)
LCN　: 論理チャネル番号 (0 ~ 255)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
RECEIVE (0)
IF RXTYP==DT THEN
   LCGN=RXLCGN( )
   LCN=RXLCN( )
   INCPR (LCGN, LCN)
   SENDPKT("RR", LCGN, LCN)
END
```

機能説明　受信順序番号 P(R) を＋ 1 更新する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(R) 値を変更します。
この値は、DT, RR, RNR パケットを送出するときにメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(8) SETPS

呼出し形式	SETPS(LCGN,LCN,PS)
概　要	送信順序番号 P(S) の値を設定します。
引数の説明	LCGN　: 論理チャネルグループ番号 (0～15) LCN　: 論理チャネル番号 (0～255) PS　: 送信順序番号 (0～127)
関数値	0　: 正常終了 –51　: レイヤ 2 自動モード・エラー –60　: 引数エラー
使用例	RECEIVE (0) IF RXTYP==RI THEN 　LCGN=RXLCGN( ) 　LCN=RXLCN( ) 　SETPR (LCGN, LCN, 0) 　SETPS (LCGN, LCN, 0) 　SENDPKT("RF", LCGN, LCN) END
機能説明	送信順序番号 P(S) の値を設定する関数です。 引数で指定した論理チャネル (LCGN と LCN) の P(S) 値を変更します。 この値は、DT パケットを送出するときのみメッセージ中に挿入されます。
制限事項	本関数は、トランスペアレント・モードでは実行できません。

(9) SETPR

呼出し形式　SETPR(LCGN,LCN,PR)

概　要　受信順序番号 P(R) の値を設定します。

引数の説明
LCGN　: 論理チャネル・グループ番号 (0 ~ 15)
LCN　: 論理チャネル番号 (0 ~ 255)
PR　: 受信順序番号 (0 ~ 127)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例

```
RECEIVE (0)
IF RXTYP==RI THEN
   LCGN=RXLCGN( )
   LCN=RXLCN( )
   SETPR (LCGN, LCN, 0)
   SETPS (LCGN, LCN, 0)
   SENDPKT("RF", LCGN, LCN)
END
```

機能説明　受信順序番号 P(R) の値を設定する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(R) 値を変更します。
この値は、DT, RR, RNR パケットを送出するときのみメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(10) ~ (16)　Q931 パラメータ挿入関数群

呼出し形式

[関数名]	[挿入するパラメータ]
(10) INS_PD("NAME",PD)	プロトコル識別子
(11) INS_CRL("NAME",CRL)	呼番号長
(12) INS_CRF("NAME",CRF)	呼番号フラグ
(13) INS_CRV("NAME",CRV)	呼番号
(14) INS_MSG("NAME",MSG)	メッセージ種別
(15) INS_INFO("NAME",N,INFO)	情報要素群フィールド
(16) INS_CS_VAL("NAME",CAUSE)	理由表示値

概　要

送出フレームの任意の Q931 パラメータを書き換えます。

引数の説明

NAME	: フレーム名
PD	: プロトコル識別子
CRL	: 呼番号長
CRF	: 呼番号フラグ
CRV	: 呼番号
MSG	: メッセージ種別
N	: N 番目を指定
INFO	: 情報要素群フィールドの値
CAUSE	: 理由表示値

関数値

0	: 正常終了
–60	: 引数エラー
–61	: フレーム名エラー
–85	: 書き換えられるメッセージが不適切

使用例

```
INS_CRV("SETUP",5)
SEND1("SETUP")
```

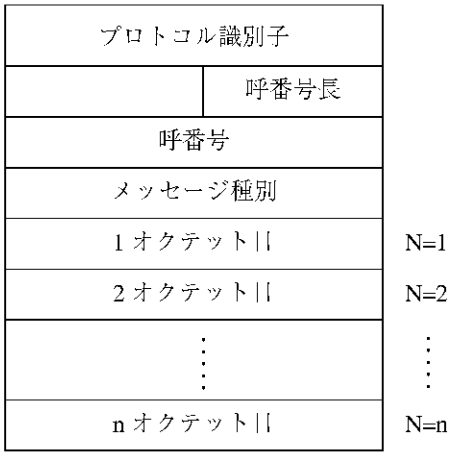
機能説明

メッセージ中の任意のパラメータを書き変える関数群です。
本関数群は、メッセージの先頭から内容を解釈していき挿入するパラメータの位置を探していきます。
挿入するパラメータの位置が発見されるとそこを引数で指定された値に書き換えます。
このとき該当するパラメータが複数の位置にわたって存在する場合、引数 N でその位置を確定させる必要があります。(INS_INFO 関数がこれに該当します。)
また、メッセージを解釈した結果、挿入するパラメータの位置が存在しない場合には、" 書き換えられるメッセージが不適切 "" という意味の関数値を返します。

補足説明

INS_INFO 関数は、メッセージ種別以降のオクテットを書き換えたい場合に使用します。
下図に INS_INFO の引数の指定方法を説明します。

① INS_INFO("NAME",N,INFO)



(17) ~ (35) X.25 パラメータ挿入関数群

呼出し形式

[関数名]	[挿入するパラメータ]
(17) INS_GFI("NAME",GFI)	ゼネラル・フォーマット識別子 (GFI)
(18) INS_Q("NAME",Q)	Q ビット
(19) INS_D("NAME",D)	D ビット
(20) INS_LCGN("NAME",LCGN)	論理チャネル・グループ番号 (LCGN)
(21) INS_LCN("NAME",LCN)	論理チャネル番号 (LCN)
(22) INS_TYP("NAME",TYP)	パケット・タイプ識別子 (TYP)
(23) INS_PR("NAME",PR)	受信順序番号 (P(R))
(24) INS_PS("NAME",PS)	送信順序番号 (P(S))
(25) INS_M("NAME",M)	M ビット
(26) INS_CLL("NAME",CLL)	発呼ユーザ・アドレス長
(27) INS_CDL("NAME",CDL)	着呼ユーザ・アドレス長
(28) INS_DA("NAME",N,DA)	着呼ユーザ・アドレス (N 番目)
(29) INS_SA("NAME",N,SA)	発呼ユーザ・アドレス (N 番目)
(30) INS_FL("NAME",FL)	ファシリティ長
(31) INS_F("NAME",N,F)	ファシリティ (N オクテット目)
(32) INS_CAUSE("NAME",CAUSE)	原因
(33) INS_DIAG("NAME",DIAG)	診断符号
(34) SETDTL("NAME",LEN)	データ長
(35) INS_DATA("NAME",N,DATA)	データ (N オクテット目)

概　要　　送出フレームの任意のパラメータを書き換えます。

引数の説明

NAME	: フレーム名
GFI	: ゼネラル・フォーマット識別子 (GFI)
Q	: Q ビット
D	: D ビット
LCGN	: 論理チャネル・グループ番号 (LCGN)
LCN	: 論理チャネル番号 (LCN)
TYP	: パケット・タイプ識別子
PR	: 受信順序番号
PS	: 送信順序番号
M	: M ビット
CLL	: 発呼ユーザ・アドレス長
CDL	: 着呼ユーザ・アドレス長
N	: N 番目を指定
DA	: 着呼ユーザ・アドレス
SA	: 発呼ユーザ・アドレス
FL	: ファシリティ長
F	: ファシリティ
CAUSE	: 原因
DIAG	: 診断符号
LEN	: データ長
DATA	: データ

(36) REQ_TEI

呼出し形式　REQ_TEI( )

概　要　TEI 割当手順を起動します。(TE モード)

引数の説明

関数値
0 ~ 126　: 割当られた TEI 値
–51　: レイヤ 2 自動モード・エラー
–52　: TE モード・エラー
–90　: TEI 割当エラー

使用例
```
TEI=REQ_TEI( )
```

機能説明　レイヤ 2 自動モードで、TEI 割当手順を起動する関数です。
この関数は、本器が TE モードのときに使用します。
TEI 割当が正常に終了すると、割り当てられた TEI 値を関数値として返します。
また、このとき割り当てられた TEI 値の登録も行います。

制限事項　この関数は、トランスペアレント・モードや NT モードでは使用できません。

(37) CHKREQ_TEI

呼出し形式　CHKREQ_TEI(TEI)

概　要　TEI チェック手順を行います。(NT モード)

引数の説明　TEI　: ID チェックを行う TEI 値 (0 ~ 127)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–53　: NT モード・エラー
–60　: 引数エラー

使用例
```
CHKREQ_TEI(64)
TEI=65
CHKREQ_TEI(TEI)
```

機能説明　レイヤ 2 自動モードで、TEI チェック手順を行う関数です。
この関数は、本器が NT モードのときに使用します。
引数で指定された TEI 値について、TEI チェック手順が行われます。
引数が 127 の場合はすべての TEI 値についての TEI チェック手順が行われます。

制限事項　引数が、0 ~ 127 以外の場合は引数エラーになります。
また、モードがトランスペアレント・モードや TE モードになっていると実行できません。

(38) REMOVE_TEI

呼出し形式　REMOVE_TEI(TEI)

概　要　TEI 解除手順を行います。(NT モード)

引数の説明　TEI　: 解除する TEI 値 (0～127)

関数値
- 0　: 正常終了
- –51　: レイヤ 2 自動モード・エラー
- –53　: NT モード・エラー
- –60　: 引数エラー

使用例
```
REMOVE_TEI(64)
TEI=64
REMOVE_TEI(TEI)
```

機能説明　レイヤ 2 自動モードで、TEI 解除手順を行う関数です。
この関数は、本器が NT モードのときに使用します。
この関数が実行されると、引数で指定された TEI 値を Ai フィールド中に付加した ID 解除メッセージを 2 回続けて送出します。

制限事項　引数が 0 ～127 以外の場合は、引数エラーになります。
モードがトランスペアレント・モードや TE モードでは使用できません。

(39) VERIFY_TEI

呼出し形式　VERIFY_TEI( )

概　要　TEI 検証手順を行います。(TE モード)

引数の説明

関数値
- 0　: 正常終了
- –51　: レイヤ 2 自動モード・エラー
- –52　: TE モード・エラー
- –71　: TEI エラー

使用例
```
VERIFY_TEI( )
```

機能説明　レイヤ 2 自動モードで、TEI 検証手順を起動する関数です。
この関数は本器が TE モードのときに使用します。
この関数が実行されると ID 検証要求メッセージが送出されます。Ai フィールドには、あらかじめ設定されている TEI 値が入れられます。

制限事項　あらかじめ設定されている TEI 値が 127 の場合は、TEI エラーになります。
TEI 値の設定変更は、ACT_TEI 関数などを用いて行うことができます。
この関数はトランスペアレント・モードや NT モードでは使用できません。

(40) LINKON

呼出し形式　LINKON( )

概　要　リンクの設定を行います。

引数の説明

関数値
0～7　：リンク番号
–51　：レイヤ２自動モード・エラー
–71　：TEI エラー
–100　：リンク設定拒否
–101　：最大同時リンク数オーバー

使用例　LINKON( )

機能説明

レイヤ２自動モードでリンクの設定を行う関数です。
リンクの設定に使用する SAPI, TEI ペアには、あらかじめ設定されたものを通常は使用します。
しかし、本器が TE モードのとき、この関数の実行がシミュレーションが行われてから１回目である場合は、リンクの設定を行う前に TEI の割当手順が行われ、そこで割り当てられた TEI 値が使用されます。
ただし、LINKON 関数が実行される前に、REQ_TEI 関数が実行されている場合は、シミュレーションが行われてから１回目の実行であっても TEI 割当手順は行われず、あらかじめ設定してあった TEI 値が使用されます。
逆に LINKON 関数が実行される前に、NEXT_TEI 関数が実行されている場合は、シミュレーションが行われてから１回目の実行でなくても TEI 割当手順が行われ、そこで設定された TEI 値が使用されます。
上記の例は、本器が TE モードのときに有効であり、本器が NT モードの場合には常にあらかじめ設定されている SAPI, TEI ペアが使用されます。
SAPI 値の設定変更には、ACT_SAPI 関数を使用します。また、リンクの設定が行われた場合には、その SAPI 値に設定変更されます。
しかし、相手からのリンクの設定を受け付ける前に LOCK_SAPI 関数が実行されている場合は、SAPI 値の設定変更は起こりません。TEI 値の設定には、ACT_TEI 関数を使用します。
TEI 値の設定は、ACT_TEI 関数の他に REQ_TEI 関数が実行された場合や、リンクの設定が行われた場合に起こります。また、本器が NT モード時に TEI の割当をした場合にも、起こります。
しかし、これらのイベントが行われる前に LOCK_TEI 関数が実行されている場合は、相手からのリンク設定による TEI 値の設定変更は起こりません。

制限事項

設定されている TEI 値に未登録の TEI 値を使用すると、TEI エラーになります。
TEI 値の登録には、REG_TEI 関数を使用します。
本器では、最大 8 リンクまで同時にリンク設定が可能です。これ以上同時にリンクを設定しようとすると、最大同時リンク数エラーになります。

(41) LINKOFF

呼出し形式　LINKOFF( )

概　要　リンクの解放を行います。

引数の説明

関数値
0　：正常終了
–51　：レイヤ２自動モード・エラー
–102　：リンク未設定エラー

使用例
```
LINKOFF( )
```

機能説明　レイヤ２自動モードで、リンクの解放を行う関数です。
あらかじめ設定してある SAPI 値、TEI 値をもつリンクの解放を行います。

制限事項　解放するはずのリンクが設定されていなかった場合には、リンク未設定エラーになります。
SAPI 値の設定には、ACT_SAPI 関数を用いて行います。
TEI 値の設定には、ACT_TEI 関数を用いて行います。
この関数は、トランスペアレント・モードでは使用できません。

(42) WAIT_LINK

呼出し形式　WAIT_LINK(SEC)

概　要　相手からのレイヤ２リンク設定待ち状態にします。

引数の説明　SEC　：相手からリンクが設定されるのを待つ時間 (100msec)
(0～16777215)

関数値
0～7　：リンク番号
–51　：レイヤ２自動モード・エラー
–60　：引数エラー
–125　：タイムアウト

使用例
```
PH_ACT( )
WAIT_LINK(3000)
SENDI("CALPRO")
```

機能説明　レイヤ２リンクが設定されるまで待つ関数です。
本関数実行後に、相手からレイヤ２リンクが設定されると内部的なリンク番号が関数値として返されます。このリンク番号から GET_SAPI(LINK),GET_TEI(LINK) 関数を用いて設定されたレイヤ２リンクの SAPI 値、TEI 値を知ることができます。
本関数実行前に、既にレイヤ２リンクが設定されている場合、新たなレイヤ２リンクが設定されるまで待ちます。引数で指定した時間を過ぎてもレイヤ２リンクが設定されない場合は、"タイムアウト"を示す関数値を返して終了します。

制限事項　本関数はレイヤ２自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(43) L_STATUS

呼出し形式　L_STATUS(SAPI,TEI)

概　要　リンクが設定されているかを調べます。

引数の説明
SAPI　：リンクの状態を調べる SAPI 値 (0, 16)
TEI　：リンクの状態を調べる TEI 値 (0 ~ 126)

関数値
0　：リンク未設定
1　：リンク設定済
–51　：レイヤ 2 自動モード・エラー

使用例
```
LST1=L_STATUS(0,64)
SAPI=16
TEI =65
LST2=L_STATUS(SAPI,TEI)
```

機能説明　レイヤ 2 自動モードで、引数で指定した SAPI, TEI ペアのリンクが設定されているかどうか調べる関数です。
調べた結果は、関数値によって返されますが、1 が関数値として返された場合には、リンクが設定済みである事を示し、0 が関数値として返された場合には、リンクは設定されていないことを示します。

制限事項　この関数は、トランスペアレント・モードでは使用できません。

(44) SET_BUSY

呼出し形式　SET_BUSY( )

概　要　自局をビジー状態にします。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー
–71　：TEI エラー
–72　：リンク・エラー

使用例
```
SET_BUSY( )
```

機能説明　レイヤ 2 自動モードで自局をビジー状態にする関数です。
現在設定されている SAPI、TEI ペアのリンクをビジー状態にします。
TEI が 127 の場合はあらかじめ設定された SAPI 値を持つすべてのリンクをビジー状態にします。

制限事項　設定されている TEI 値に、未登録の TEI 値を使用すると、TEI エラーになります。
TEI 値の登録には、REG_TEI 関数を使用します。
リンクの設定がされていない SAPI, TEI ペアに対して、この関数が実行されるとリンク・エラーになります。ただし、TEI 値が 127 の場合にはこの限りではありません。
この関数は、トランスペアレント・モードでは使用できません。

(45) REL_BUSY

呼出し形式　REL_BUSY( )

概　要　自局のビジー状態を解除します。

引数の説明

関数値
0　：正常終了
–51：レイヤ 2 自動モード・エラー
–71：TEI エラー
–72：リンク・エラー

使用例　REL_BUSY( )

機能説明　レイヤ 2 自動モードで、自局のビジー状態を解除する関数です。
現在設定されている SAPI, TEI ペアに対してビジー状態を解除します。
設定されている TEI 値が 127 の場合は、設定されている SAPI 値を持つすべてのリンクのビジー状態を解除します。

制限事項　設定されている TEI 値に、未登録の TEI 値を使用すると、TEI エラーになります。
TEI 値の登録には、REG_TEI 関数を使用します。
リンクの設定がされていない SAPI, TEI ペアに対して、この関数が実行されるとリンク・エラーになります。ただし、TEI 値が 127 の場合はこの限りではありません。
この関数は、トランスペアレント・モードでは使用できません。

(46) PROHIBIT_L

呼出し形式　PROHIBIT_L( )

概　要　新たなリンクの設定を禁止します。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー

使用例　PROHIBIT_L( )

機能説明　レイヤ 2 自動モードで、あらかじめ設定された SAPI 値について新たなリンクの設定を禁止する関数です。
この関数が実行されると、相手局からの新たなリンクの設定は禁止されますが、自局からのリンクの設定はできます。

制限事項　この関数はトランスペアレント・モードでは使用できません。

(47) PERMIT_L

呼出し形式　PERMIT_L( )

概　要　リンク設定不許可状態を解除します。

引数の説明

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–70　: SAPI エラー

使用例
```
PERMIT_L( )
```

機能説明　レイヤ 2 自動モードで、PROHIBIT_L 関数で設定したリンク設定不許可状態を解除する関数です。
あらかじめ設定されている SAPI 値に対して、リンク不許可状態を解除します。

制限事項　この関数は、トランスペアレント・モードでは使用できません。

(48) REG_TEI

呼出し形式　REG_TEI(TEI)

概　要　TEI 値を本器に登録し、その TEI 値のフレームを受信可能にします。

引数の説明　TEI　: 登録を行う TEI 値 (0 ~ 126)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
REG_TEI(64)
TEI=65
REG_TEI(TEI)
```

機能説明　レイヤ 2 自動モードで、TEI 値の登録を行う関数です。
引数で指定された TEI 値について、TEI の登録が行われます。

制限事項　一度登録を行った TEI 値を再び登録しようとすると引数エラーになります。
引数が 0 ~ 126 以外の場合は、引数エラーになります。
この関数はトランスペアレント・モードでは使用できません。

(49) REL_TEI

呼出し形式　REL_TEI(TEI)

概　要　TEI 値を解除し、その TEI 値のフレームを受信しないようにします。

引数の説明　TEI　: 解除する TEI 値 (0 ~ 127)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
REL_TEI(64)
TEI = 65
REL_TEI(TEI)
```

機能説明　レイヤ 2 自動モードで、登録されている TEI 値の解除を行う関数です。引数には、0 ~ 127 の値が設定できますが、127 を設定すると登録されているすべての TEI 値が解除されます。

制限事項　引数に 0 ~ 127 以外の値が設定されると引数エラーになります。この関数は、トランスペアレント・モードでは使用できません。

(50) NEXT_TEI

呼出し形式　NEXT_TEI( )

概　要　新たな TEI 値を使用することを宣言します。(TE モード)

引数の説明

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー

使用例
```
NEXT_TEI( )
```

機能説明　レイヤ 2 自動モードで、新たな TEI 値を用いてリンクを設定したいときに使用する関数です。
この関数は、本器が TE モードのときに LINKON 関数とペアで使用します。
LINKON 関数はリンクの設定を行う関数です。シミュレーションが起動されてから 1 回目の実行の際にのみ、リンクの設定を行う前に TEI の割当手順が起動されます。したがって、2 回目からの LINKON 関数では、あらかじめ設定された SAPI, TEI ペアについてリンクの設定が行われ、TEI の割当手順は起動されません。
NEXT_TEI 関数は、TEI の割当手順を再び行って、新たな TEI 値でリンクの設定をしたい場合に、LINKON 関数を実行する前に使用します。
この関数の代わりに、REQ_TEI 関数を用いても同じことができます。

制限事項　この関数はトランスペアレント・モードでは使用できません。
また、NT モードで使用した場合は、TEI 値がデフォルト値に戻ります。

(51) ACT_SAPI

呼出し形式	ACT_SAPI(SAPI)
概　要	フレーム送出時に使用される SAPI 値を設定します。
引数の説明	SAPI　　: 設定をする SAPI 値 (0, 16, 63)
関数値	0　　: 正常終了 –51　　: レイヤ 2 自動モード・エラー –60　　: 引数エラー
使用例	ACT_SAPI(0) SAPI=16 ACT_SAPI(SAPI)
機能説明	レイヤ 2 自動モードで、SAPI 値を設定変更する場合に使用する関数です。
制限事項	引数で指定された SAPI 値が 0 、16、63 以外の場合は、引数エラーになります。 この関数はトランスペアレント・モードでは使用できません。

(52) ACT_TEI

呼出し形式	ACT_TEI(TEI)
概　要	フレーム送出時に使用される TEI 値を設定します。
引数の説明	TEI　　: 設定を行う TEI 値 (0 ~ 127)
関数値	0　　: 正常終了 –51　　: レイヤ 2 自動モード・エラー –60　　: 引数エラー
使用例	ACT_TEI(64) TEI=65 ACT_TEI(TEI)
機能説明	レイヤ 2 自動モードで、TEI 値を設定変更する場合に使用する関数です。
制限事項	引数で指定された TEI 値が未登録の場合は、引数エラーになります。 この関数は、トランスペアレント・モードでは使用できません。

(53) ACT_LINK

呼出し形式　ACT_LINK(LINK)

概　要　フレーム送出時に使用されるリンク番号を設定します。

引数の説明　LINK　：リンク番号 (0～7)

関数値
0　：正常終了
–51　：レイヤ２自動モード・エラー
–60　：引数エラー
–102　：リンク未設定エラー

使用例
```
PH_ACT( )
REQ_TEI( )
LINK1=LINKON( )
REQ_TEI( )
LINK2=LINKON( )
ACT_LINK(LINK1)
SENDI("SETUP1")
ACT_LINK(LINK2)
SENDI("SETUP2")
```

機能説明　I フレーム送出時に、使用するレイヤ２リンクを変更する関数です。
引数で与えられる LINK は内部的に使用しているリンク番号で、LINKON, WAIT_LINK, SEE_LINK, GET_LINK 関数などで知ることができます。
引数 LINK が指定したレイヤ２リンクが設定されていない場合は、" リンク未設定エラー " となります。

制限事項　本関数はレイヤ２自動モード用関数のため、トランスペアレント・モードでは実行できません。

(54) LOCK_SAPI

呼出し形式　LOCK_SAPI( )

概　要　SAPI 値が自動で設定変更されるのを禁止します。

引数の説明

関数値
0　：正常終了
–51　：レイヤ２自動モード・エラー

使用例
```
LOCK_SAPI( )
```

機能説明　レイヤ２自動モードで、相手からのリンク設定により SAPI 値が設定変更されることを禁止する関数です。
ただし、この関数が実行された後にも ACT_SAPI 関数により SAPI 値を設定変更することはできます。

制限事項　LOCK_SAPI 関数により SAPI 値変更不可状態になりますが、これを解除するには、FLEX_SAPI 関数を用いて行います。
この関数は、トランスペアレント・モードでは使用できません。

(55) LOCK_TEI

呼出し形式　LOCK_TEI( )

概　要　TEI 値が自動で設定変更されるのを禁止します。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー

使用例
```
LOCK_TEI( )
```

機能説明　レイヤ 2 自動モードで、相手からのリンク設定による TEI 値の設定変更を禁止する関数です。
ただし、この関数が実行された後でも、ACT_TEI 関数を用いて、TEI 値を設定変更することはできます。

制限事項　LOCK_TEI 関数を実行することにより TEI 値変更不可状態になりますが、この状態を解除するためには、FLEX_TEI 関数を用います。
この関数は、トランスペアレント・モードでは使用できません。

(56) LOCK_LINK

呼出し形式　LOCK_LINK( )

概　要　リンク番号が自動で設定変更されるのを禁止します。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー

使用例
```
LINKON( )
LOCK_LINK( )
SEND1("SETUP")
```

機能説明　相手からのリンク設定により SAPI 値、TEI 値が設定変更されるのを禁止する関数です。
ただし、ACT_SAPI, ACT_TEI, ACT_LINK 関数などで SAPI 値、TEI 値を設定変更することはできます。LOCK_LINK 関数で設定した SAPI 値、TEI 値変更不可状態は FLEX_LINK 関数により解除できます。

制限事項　本関数は、レイヤ 2 自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(57) FLEX_SAPI

呼出し形式	FLEX_SAPI( )
概　要	SAPI 値が自動で設定変更されるようにします。
引数の説明	
関数値	0　　：正常終了 –51　　：レイヤ２自動モード・エラー
使用例	FLEX_SAPI( )
機能説明	レイヤ２自動モードで、SAPI 値変更不可状態を解除する関数です。
制限事項	この関数は、トランスペアレント・モードでは使用できません。

(58) FLEX_TEI

呼出し形式	FLEX_TEI( )
概　要	TEI 値が自動で設定変更されるようにします。
引数の説明	
関数値	0　　：正常終了 –51　　：レイヤ２自動モード・エラー
使用例	FLEX_TEI( )
機能説明	レイヤ２自動モードで、TEI 値変更不可状態を解除する関数です。
制限事項	この関数はトランスペアレント・モードでは使用できません。

(59) FLEX_LINK

呼出し形式	FLEX_LINK( )
概　要	リンク番号が自動で設定変更されるようにします。
引数の説明	
関数値	0　　：正常終了 –51　　：レイヤ２自動モード・エラー
使用例	LINKON( ) LOCK_LINK( ) SENDI("SETUP") FLEX_LINK( )
機能説明	LOCK_LINK 関数によりリンクが自動で変更されるのを禁止した状態を解除し、再びリンクが自動変更されるのを許可する関数です。
制限事項	本関数はレイヤ２自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(60) GET_SAPI

呼出し形式　GET_SAPI(LINK)

概　要　フレーム送出時に使用する SAPI 値を調べます。

引数の説明　LINK　: リンク番号　(0 ~ 7)

関数値
- 0, 16, 63　: 調べた SAPI 値
- –51　: レイヤ 2 自動モード・エラー
- –60　: 引数エラー
- –102　: リンク未設定エラー

仕様実例
```
LINK=WAIT_LINK(3000)
SAPI=GET_SAPI(LINK)
```

機能説明　引数 LINK のレイヤ 2 リンクが設定されているか調べにいき、設定されていればそのリンクの SAPI 値を関数値として返します。引数 LINK のレイヤ 2 リンクが設定されていない場合に、" リンク未設定エラー " の関数値を返します。

制限事項　本関数はレイヤ 2 自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(61) GET_TEI

呼出し形式　GET_TEI(LINK)

概　要　フレーム送出時に使用する TEI 値を調べます。

引数の説明　LINK　: リンク番号　(0 ~ 7)

関数値
- 0 ~ 126　: 調べた TEI 値
- –51　: レイヤ 2 自動モード・エラー
- –60　: 引数エラー
- –102　: リンク未設定エラー

使用例
```
LINK=WAIT_LINK(3000)
TEI=GET_TEI(LINK)
```

機能説明　引数 LINK のレイヤ 2 リンクが設定されているか調べにいき、設定されていればそのリンクの TEI 値を関数値として返します。引数 LINK のレイヤ 2 リンクが設定されていない場合、" リンク未設定エラー " の関数値を返します。

制限事項　本関数はレイヤ 2 自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(62) GET_LINK

呼出し形式　GET_LINK(SAPI,TEI)

概　要　フレーム送出時に使用されるリンク番号を調べます。

引数の説明
SAPI　: SAPI 値 (0, 16, 63)
TEI　: TEI 値 (0 ~ 126)

関数値
0 ~ 7　: 調べたリンク番号
–51　: レイヤ２自動モード・エラー
–60　: 引数エラー
–102　: リンク未設定エラー

使用例
```
LINK=GET_LINK(0,64)
ACT_LINK(LINK)
SENDI("SETUP")
```

機能説明　引数 SAPI,TEI ペアリンクが設定されているか調べにいき、設定されていればそのリンク番号を関数値として返します。引数 SAPI,TEI ペアのリンクが設定されていなければ " リンク未設定エラー " の関数値を返します。

制限事項　本関数はレイヤ２自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(63) SEE_LINK

呼出し形式　SEE_LINK( )

概　要　現在設定されているリンク番号を調べます。

引数の説明

関数値
0 ~ 7　: リンク番号
–51　: レイヤ２自動モード・エラー
–102　: リンク未設定エラー

使用例
```
RECEIVE(0)
LINK=SEE_LINK()
PRINT("LINK=%D\N",LINK)
```

機能説明　本関数が実行されている時点で、設定されているレイヤ２リンクがあれば有効になっているレイヤ２リンクのリンク番号を関数値として返します。
しかし、設定されているレイヤ２リンクがない場合は " リンク未設定エラー " となります。

制限事項　本関数はレイヤ２自動モードで使用する関数のため、トランスペアレント・モードでは実行できません。

(64) ~ (79) LAPD, Q.931 パラメータ読み出し関数

呼出し形式

[関数名]	[読み出すパラメータ]
(64) RXSAPI( )	SAPI 値
(65) RXTEI( )	TEI 値
(66) RXTYPE( )	レイヤ 2 フレームのタイプ
(67) RXPF( )	P/F ビット値
(68) RXPD( )	プロトコル識別子
(69) RXCRL( )	呼番号長
(70) RXCRF( )	呼番号フラグ
(71) RXCRV( )	呼番号値
(72) RXMSG( )	メッセージ種別
(73) RXINFO_NUM( )	情報要素数
(74) RXINFO_ELM(N)	情報要素識別子
(75) RXINFO_LEN(N)	情報要素内容長
(76) RXINFO_VAL(N,M)	情報内容
(77) RXINFO_POS(ELM)	情報要素識別子の位置
(78) RXCS_VAL( )	理山表示値
(79) RXCHAN_NUM( )	チャネル番号

概　要　受信フレームの任意の Q.931 パラメータを読み出します。

引数の説明

N	: 情報要素の位置
M	: 情報内容の位置
ELM	: 情報要素識別子 (0 ~ 255)

関数値

0 ~ 255	: 読み出した Q.931 パラメータの値
–60	: 引数エラー
–80	: フレーム未受信エラー
–82	: 読み出すパラメータが受信フレーム中に存在しない

使用例

```
RECEIVE(0)
CRV=RXCRV()
PRINT("Call Ref=%D¥N",CRV)
```

機能説明　受信フレームの Q.931 パラメータを読み出す関数群です。
受信したフレームを順に読むことができます。
受信フレームを読み出したい場合は RECEIVE 関数を実行し、受信フレームを内部に取り込んでから本関数群を実行します。次の受信フレームを読み出したい場合は、再び RECEIVE 関数を実行します。
このときも受信フレームがない場合は受信待ち状態になります。本関数は受信フレームの先頭から内容を解釈していき、該当する Q.931 パラメータを関数値として返しますが、該当する Q.931 パラメータが複数存在する場合 (RXINFO_ELM, RXINFO_LEN, RXINFO_VAL) は、引数 N, M でその位置を確定させる必要があります。
また、メッセージの内容を解釈していった結果読み出すパラメータが存在しない場合 " 読み出すパラメータが存在しない " という意味の関数値を返します。

補足説明　RXINFO_LEN, RXINFO_VAL, RXINFO_POS, RXCHAN_NUM 関数については以降に説明します。

これらの関数は、トランスペアレント・モードでも実行できます。

(75) RXINFO_LEN

呼出し形式　RXINFO_LEN(N)

概　要　受信フレームの情報要素内容長を読み出します。

引数の説明　N　:情報要素内容長を読み出す情報要素の位置

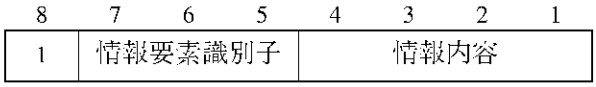
関数値
- 0 ～ 255　:受信フレームのレイヤ３メッセージ内に含まれる情報要素内容長
- –60　:引数エラー
- –80　:フレーム未受信エラー
- –82　:JT-Q931 で示す情報内容が存在しない

使用例
```
LEN1=RXINFO_LEN(1)
NUM=2
LEN=RXINFO_LEN(NUM)
```

機能説明　"(64) ～(79) LAPD, Q.931 パラメータ読み出し関数" の頁を参照して下さい。

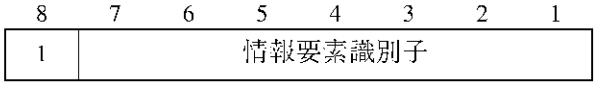
詳細説明　引数で指定した情報要素の情報要素内容長を関数値として返しますが、引数で指定した位置に情報要素が存在しない場合や、情報要素があっても情報内容がない場合には、引数エラーになります。情報要素数以上の値が引数として指定されたときには、引数エラーが起こります。下図を参照して下さい。

a) 単一固定長情報要素のフォーマット (タイプ 1)



RXINFO_LEN (N) = 0

b) 単一固定長情報要素のフォーマット (タイプ 2)



RXINFO_LEN (N) = –82 (エラー)

c) 可変長情報要素の場合
RXINFO_LEN (N) = 情報要素内容長

(76) RXINFO_VAL

呼出し形式　RXINFO_VAL(N,M)

概　要　受信フレームの情報内容を読み出します。

引数の説明
- N　: 情報要素内容を読み出す情報要素の位置
- M　: 情報内容中から読み出すデータのオクテット値

関数値
- 0 ~ 255　: 受信フレームのレイヤ 3 メッセージ内に含まれる情報内容
- –60　: 引数エラー
- –80　: フレーム未受信エラー
- –82　: JT-Q931 で示す情報内容が存在しない

使用例
```
VAL=RXINFO_VAL(1,1)
N=1
M=2
VAL2=RXINFO_VAL(N,M)
```

機能説明　"(64) ~(79) LAPD, Q.931 パラメータ読み出し関数" の頁を参照して下さい。

詳細説明　引数で指定した位置の情報内容を関数値として返します。
引数の設定の仕方は、下図に示します。下図以外の設定がされた場合には、引数エラーになります。

① RXINFO VAL(N,M) 関数における引数 N の指定の仕方

注　引数 N は、$1 \leqq N \leqq N_1$ の範囲で指定します。
(N_1 は情報要素数)

② RXINFO_VAL(N,M) 関数における引数 M の指定の仕方

A) 単一固定長情報要素のフォーマット (タイプ 1)

8	7	6	5	4	3	2	1	
1	情報要素識別子			情報内容				M = 0

RXINFO_LEN (N) = 0
RXINFO_VAL(N,0)="情報内容"

B) 単一固定長情報要素のフォーマット (タイプ 2)

8	7	6	5	4	3	2	1	
1	情報要素識別子							有効な M はない。

RXINFO_LEN(N)=–82(引数エラー)
RXINFO_VAL(N,M)=–82(引数エラー)

注　このタイプの情報要素の場合には、RXINFO_LEN 関数や RXINFO_VAL 関数は実行できません。

C) 可変長情報要素のフォーマット

8	7	6	5	4	3	2	1	
0	情報要素識別子							
情報要素内容長								
情　報　内　容　1								M=1
情　報　内　容　2								M=2
情　報　内　容　3								M=3
～							～	
情　報　内　容 M_{1-1}								$M=M_{1-1}$
情　報　内　容 M_1								$M=M_1$

注　引数 M は、$1 \leqq M \leqq M_1$ の範囲で指定します。
(M_1 は情報要素内容長)

(77) RXINFO_POS

呼出し形式　RXINFO_POS(ELM)

概　要　情報要素識別子の位置を調べます。

引数の説明　ELM　: 位置を調べる情報要素識別子 (0 ~ 255)

関数値
1 ~ 512　: 情報要素識別子の位置
–60　: 引数エラー
–80　: フレーム未受信エラー
–82　: 読み出すパラメータが受信フレーム中に存在しない

使用例
```
RECEIVE(0)
POS=RXINFO_POS(H'18')
CHAN=EXTRACT(POS+2)&H'03'
PRINT("CHANNEL=%D¥N",CHAN)
```

機能説明　受信フレーム中にある引数 ELM の情報要素識別子の位置を関数値として返す関数です。
特定の情報要素識別子の内容について読み出したい場合に、本関数は有効です。本関数で指定の情報要素識別子の位置を調べ、EXTRACT 関数でその位置をもとに内容を読み出すと簡単に知りたい内容を得ることができます。ただし、コード群 0 以外の識別子についても同様に位置を返すので注意が必要です。
本関数を実行する前に必ず RECEIVE 関数を実行し、受信フレームを内部に取り込んで下さい。複数のフレームを受信している場合、RECEIVE 関数を実行するたびに読み出す対象となる受信フレームが更新されていきます。この順番は、フレームを受信した順です。
RECEIVE 関数を実行すると次の受信フレームに読み出す対象のフレームを切り換えます。このとき次のフレームがまだ受信されていない場合は、フレーム受信待ち状態になります。
RECEIVE 関数が実行されていない場合または、フレーム未受信の状態で本関数を実行する場合に、フレーム未受信エラーになります。
また、引数 ELM の情報要素識別子が受信フレーム中に存在しない場合は引数エラーになります。
受信フレームが Q.931 フォーマットでない場合には、" 読み出すパラメータが受信フレーム中に存在しない " と言う意味の関数値を返します。

(79) RXCHAN_NUM

呼出し形式　RXCHAN_NUM( )

概　要　受信フレームのチャネル番号を読み出します。

引数の説明

関数値

0	: 任意のチャネル
1 ～ 24	: B チャネルの番号
101	: H0 チャネル・ユニット (ch1 ～ 6 使用)
102	: H0 チャネル・ユニット (ch7 ～ 12 使用)
103	: H0 チャネル・ユニット (ch13 ～ 18 使用)
104	: H0 チャネル・ユニット (ch19 ～ 24 使用)
200	: H11 チャネル
–80	: フレーム未受信エラー
–82	: 読み出すフレームが受信フレーム中に存在しない

使用例

```
RECEIVE(0)
CHAN=RXCHAN_NUM( )
PRINT("CHANNEL=%D¥N",CHAN)
```

機能説明　"(64) ～ (79) LAPD, Q.931 パラメータ読み出し関数 " を参照して下さい。

詳細説明　受信したフレームのチャネル識別子内にあるチャネル番号を関数値として返します。関数値として 1 ～ 24 が返される場合、情報チャネルの選択はそれぞれ B1 ～ B24 の関数値で示された番号の B チャネルであることが分かります。また 0 が返される場合は、情報チャネル選択は任意のチャネルであることを意味します。情報チャネル選択が " チャネルなし " の場合には、関数値として –82 が返されます。
インタフェース・タイプが一次群の場合で、H0 チャネルが指定される場合は、上記の関数値に示すとおり、101 ～ 104 の値が返されます。ただし、一次群の場合で B チャネルが複数指定される場合、先頭に書かれる B チャネル (番号で指定の場合) または、B チャネルの一番若い番号 (マップで指定の場合) が返されます。また H11 チャネルが指定された場合は、200 が関数値として返されます。

(80) ~ (98)　X.25 パラメータ読み出し関数群

呼出し形式	[関数名]	[読み出すパラメータ]
	(80) RXGFI( )	ゼネラル・フォーマット識別子 (GFI)
	(81) RXQ( )	Q ビット
	(82) RXD( )	D ビット
	(83) RXLCGN( )	論理チャネル・グループ番号 (LCGN)
	(84) RXLCN( )	論理チャネル番号 (LCN)
	(85) RXTYP( )	パケットタイプ識別子
	(86) RXPR( )	受信順序番号 P(R)
	(87) RXPS( )	送信順序番号 P(S)
	(88) RXM( )	M ビット
	(89) RXCLL( )	発呼ユーザ・アドレス長
	(90) RXCDL( )	着呼ユーザ・アドレス長
	(91) RXDA(N)	着呼ユーザ・アドレス (N 番目)
	(92) RXSA(N)	発呼ユーザ・アドレス (N 番目)
	(93) RXFL( )	ファシリティ長
	(94) RXF(N)	ファシリティ (N オクテット目)
	(95) RXCAUSE( )	原因
	(96) RXDIAG( )	診断符号
	(97) RXDTL( )	データ長
	(98) RXDATA(N)	データ (N オクテット目)

概　要　受信フレームの任意の X.25 パラメータを読み出します。

引数の説明

N	:N 番目を指定

関数値

0 ~ 255	: 読み出したパラメータの値
–60	: 引数エラー
–80	: フレーム未受信エラー
–82	: 読み出すパラメータが受信フレーム中に存在しない

使用例

```
RECEIVE(0)
GFI=RXGFI( )
PRINT("GFI=%D¥N",GFI)
```

機能説明

受信フレームの X.25 パラメータを読み出す関数です。
フレームを受信した順に読むことができます。
受信フレームを読み出したい場合は RECEIVE 関数を実行し、受信フレームを内部に取り込んでから本関数群を実行します。次の受信フレームを読み出したい場合は、再び RECEIVE 関数を実行します。このとき受信フレームがない場合は、受信待ち状態になります。本関数は受信フレームの先頭から内容を解釈していき、該当する X.25 パラメータを関数値として返しますが、該当の X.25 パラメータが複数存在する場合 (RXSA,RXDA,RXF,RXDATA など) は、引数 N で何番目のパラメータを読み出すのか指定する必要があります。
このとき N が不適切であると、引数エラーとなります。
また、メッセージの内容を解釈していった結果、読み出すパラメータが存在しない場合、”読み出すパラメータが受信フレーム中に存在しない” という意味の関数値を返します。

補足説明

これらの関数は、トランスペアレント・モードでも実行できます。
RXTYP( ) 関数で返される関数値とパケット名は、以下の対応となっています。

パケット名	関数値		パケット名	関数値	
	10 進	16 進		10 進	16 進
CR	11	0B	SF	255	FF
CN	11	0B	DT	0	00
CA	15	0F	RR	1	01
CC	15	0F	RNR	5	05
CQ	19	13	RQ	27	1B
CI	19	13	RI	27	1B
CF	23	17	RF	31	1F
SQ	251	FB	IT	35	23
SI	251	FB	IF	39	27

9. B チャネル・シミュレーション

9.1 共通関数

表 9-1 に共通関数を表示します。

表 9-1 共通関数

関数名	内容
(1) INSERT	送信フレームの任意のオクテットの値を変える
(2) SET_FRLEN	送信フレームのフレーム長を変える。
(3) SET_CHANN	送受信するチャネルを指定する。
(4) SET_CH64K	64kbps のシミュレータが使用するチャネルを指定する。
(5) SET_CH32K	32kbps のシミュレータが使用するチャネルを指定する。
(6) RECEIVE	タイムアウト / フレーム / キー入力 / イベント受信待ち状態にする。
(7) T_START	タイマを起動する。(分解能 1 秒のタイマ起動)
TM_START	タイマを起動する。(分解能 100m 秒のタイマ起動)
(8) T_STOP	タイマを停止する。
(9) READ_TIMER	タイムアウトしたタイマの ID を得る。
(10) SEND_EVENT	他チャネルにイベントを送る。
(11) READ_EVENT	他チャネルから送られてきたイベントの内容を読み出す。
(12) EXTRACT	受信フレームの任意のオクテットの値を読み出す。
(13) RXFRLEN	受信フレーム・データ長を調べる。
(14) WAIT	指定の時間プログラムを停止する。
(15) PH_ACT	レイヤ 1 を起動する。
(16) PH_DEACT	レイヤ 1 を停止する。
(17) READ_VAL	キー入力された数値を読み出す。
(18) REG_EVTID	イベント ID を登録する。
(19) REL_EVTID	登録されているイベント ID を解放する。
(20) SEND_EVTS	他チャネルにイベントを送ります。
(21) WRITE_EVTS	他チャネルに送信するイベントを書き込みます。
(22) READ_EVTS	他チャネルから送信されたイベントを読み出します。
(23) COUNT_EVT	他チャネルから送信されたイベント数を得ます。
(24) GET_SIMID	現在実行中のシミュレーション ID を得ます。
(25) GET_LAY1ID	現在実行中のレイヤ 1 インタフェース ID を得ます。

次ページ以降に各関数の説明をします。

(1) INSERT

呼出し形式　INSERT("NAME",n,DT)

概　要　送信フレームの任意のオクテット値を書き換えます。

引数の説明
- NAME　:データの変更を行うフレームのフレーム名
- n　:データの変更を行うオクテット値
  - トランスペアレント・モード時 :1～4200
  - レイヤ 2 自動実行モード時　:1～4198 または 1～4197
  - (最大値は、レイヤ 2 のメッセージに依存する)
- DT　:データの変更値 (0～255 の範囲)

関数値
- 0　:正常終了
- –60　:引数エラー
- –61　:フレーム名エラー

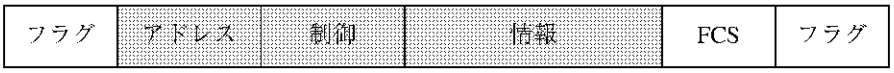
使用例　INSERT("SABME",2,H'81')

機能説明　引数で指定したフレーム名の任意のオクテットのデータを変更する関数です。

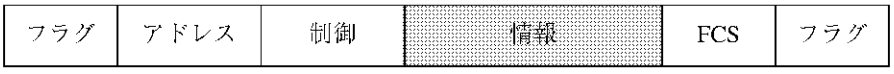
注意事項　一度、変更した値は、シミュレーションをストップするまで有効です。
トランスペアレント・モードとレイヤ 2 自動モードとでは変更できるデータの領域が違います。(下図参照)
下図で斜線部の領域が変更可能です。
また、変更位置は斜線部の左側から順に、1 から数えたオクテットとします。

制限事項　引数で指定したフレーム名のフレームがない場合は、フレーム名エラーになります。
また、データの変更を行うオクテット値やデータの変更値に範囲外の値を指定すると、引数エラーになります。

① トランスペアレント・モード



② レイヤ 2 自動モード



(2) SET_FRLEN

呼出し形式　SET_FRLEN("NAME",LEN)

概　要　送信フレームのデータ長を変えます。

引数の説明
NAME　:フレーム名
LEN　:変更するデータ長
トランスペアレントモード時 :1～4200
レイヤ 2 自動実行モード時　:1～4198 または 1～4197
(最大値は、レイヤ 2 のメッセージに依存する)

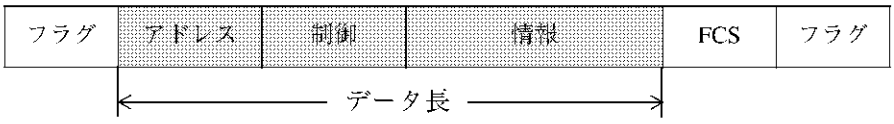
関数値
0　:正常終了
–60　:引数エラー
–61　:フレーム名エラー

使用例　SET_FRLEN("SPL1",10)

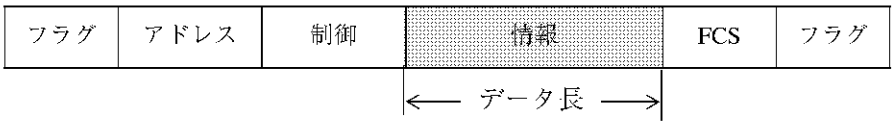
機能説明　引数で指定したフレーム名のデータ長を変更する関数です。

注意事項　一度フレーム長を変えるとシミュレーションを停止するまで有効です。トランスペアレント・モードとレイヤ 2 自動実行モードでは、変更できるデータ長の意味が異なります。(下図参照)

① トランスペアレント・モード



② レイヤ 2 自動モード



(3) SET_CHANN

呼出し形式　SET_CHANN(CHANN)

概　要　送受信するチャネルを設定します。

引数の説明　CHANN　:チャネル番号
0: チャネルを設定していない状態
1: B1 チャネル
2: B2 チャネル
3: B3 チャネル
（基本、U 点インタフェースの場合は D チャネル）
⋮
24: B24 チャネル
（一次群インタフェースの場合は B1 ～ B24 チャネルの設定が可能）

関数値
0　:正常終了
-163　:使用しているインタフェースに指定のチャネルがない
-180　:指定したチャネルはすでに使用されている

使用例
```
PH_ACT( )
SET_CHANN(1)
SENDT("SPL")
```

機能説明　フレームを送受信するチャネルを設定する関数です。
デフォルトでは、チャネルは設定されていないので、必ず設定して下さい。
また、本関数は LAPB フレーム用です。SOUND_ON、BERT_ON などの関数を実行するには、SET_CH32K、SET_CH64K 関数を使用します。

(4) SET_CH64K

呼出し形式　SET_CH64K (TYPE, CHANN)

概　要　64Kbps の通信速度を持つシミュレータが使用するチャネルを設定します。

引数の説明

TYPE　:チャネルを使用するシミュレータの種類
- 1 : LPAB
- 11: BERT
- 21: Sound 1
- 22: Sound 2
- 31: Ext. data

CHANN　:チャネル番号
- 0: チャネルを設定していない状態
- 1: B1 チャネル
- 2: B2 チャネル
- 3: B3 チャネル
  （基本、U 点インタフェースの場合は D チャネル）
- ・
- ・
- ・
- 24: B24 チャネル
  （一次群インタフェースの場合は B1 ～ B24 チャネルの設定が可能）

関数値

0	:正常終了
-60	:引数エラー
-163	:使用しているインタフェースに指定のチャネルがない
-180	:指定したチャネルはすでに使用されている
-181	:指定したシミュレータは使用できない

使用例

```
PH_ACT( )
SET_CH64K (21, 1)
SOUND1_ON (1)
```

機能説明　64Kbps の通信速度を持つシミュレータが使用するチャネルを設定する関数です。
引数でチャネルを使用する種類とチャネル番号を指定します。
すでに使用しているチャネルを指定するとエラーになります。その場合、設定していない状態にしてから、再度設定して下さい。

注　Sound2 は、現バージョンでは対応していません。

---

(5) SET_CH32K

呼出し形式　SET_CH32K (TYPE, CHANN, SEL)

概　要　32Kbps の通信速度を持つシミュレータが使用するチャネルを設定します。

引数の説明

TYPE　: チャネルを使用するシミュレータの種類
- 21: Sound 1
- 22: Sound 2

CHANN　: チャネル番号
- 0: チャネルを設定していない状態
- 1: B1 チャネル
- 2: B2 チャネル
- 3: B3 チャネル
  （基本、U 点インタフェースの場合は D チャネル）
- ⋮
- 24: B24 チャネル
  （一次群インタフェースの場合は B1 ～ B24 チャネルの設定が可能）

SEL　: 32K で使用するビットの選択
- 1: 前段 (First 4 Bits)
- 2: 後段 (Last 4 Bits)

関数値

0	: 正常終了
-60	: 引数エラー
-163	: 使用しているインタフェースに指定のチャネルがない
-180	: 指定したチャネルはすでに使用されている
-181	: 指定したシミュレータは使用できない

使用例

```
PH_ACT( )
SET_CH32K (21, 1, 1)
SOUND1_ON (3)
```

機能説明　32Kbps の通信速度を持つシミュレータが使用するチャネルを設定します。
引数でチャネルを使用する種類とチャネル番号を指定します。
すでに使用しているチャネルを指定すると、エラーになります。その場合、設定していない状態にしてから、再度設定して下さい。

注意　Sound1 と Sound2 との個別設定はできません。

注　Sound2 は、現バージョンでは対応していません。

---

(6) RECEIVE

呼出し形式　RECEIVE(TIMER_ID)

概　要　タイムアウト / フレーム / キー入力 / イベント受信待ち状態にします。

引数の説明　TIMER_ID　: タイマ番号 (0 ~ 16777215)

関数値
0	: タイムアウト・イベント受信による終了
1	: フレーム受信による終了
2	: キー入力受信による終了
10	: D チャネルからのイベント受信による終了
20	: B チャネルからのイベント受信による終了
200	: 登録されたイベント ID を使用したイベントの受信

使用例
```
TM_ID=1
TM_SEC=10
T_START(TM_ID,TM_SEC)
RET=RECEIVE(TM_ID)
IF RET==0 THEN
   PRINT("TIME OUT¥N")
END
```

機能説明　本関数を実行するとタイムアウト / フレーム / キー入力 / イベントの受信待ち状態になります。上記のイベントを受信するまでプログラムは停止します。上記のイベントを受信すると RECEIVE 関数を抜けプログラムは再実行します。このとき RECEIVE 関数が返す関数値は受信したイベントの種類を示します。
タイマは、本関数と T_START, TM_START, T_STOP 関数を用いて管理します。
タイマの起動は、T_START または TM_START 関数で行い、タイムアウトしたそのイベントは本関数で受け取ります。そのとき、本関数の引数 TIMER_ID 以外のタイマがタイムアウトした場合は受信されません。ただし、TIMER_ID が 0 の時は例外で、すべての TIMER_ID のタイムアウトイベントを受信します。このときどのタイマがタイムアウトしたかは READ_TIMER 関数で知ることができます。タイマの停止は、T_STOP 関数で行います。
すでに、フレームを受信している状態で RECEIVE 関数が実行されると関数値として 1 を返し終了します。このとき RX で始まる受信フレーム読み出し関数が利用できます。
受信した次のフレームの内容を読み出したい場合は、再び RECEIVE 関数を実行します。RECEIVE 関数を実行するごとに受信したフレームの順に、その内容を RX で始まる関数を使用することにより読み出すことができます。
受信したフレームがない場合や、次のフレームをまだ受信していない場合は、受信待ち状態になりフレームを受信した後、上記のようにフレーム内容を読み出せるようになります。
INPUT 関数が実行されていないときに、Command ラインから数値が入力されると、関数値 2 を返します。この数値は、READ_VAL 関数で読み出すことができます。(INPUT 関数参照)

他チャネルからのイベントを受信した場合には、関数値として上記の値を返します。受信した他チャネルからのイベントは READ_ EVENT 関数で読むことができます。

(7) T_START/TM_START

呼出し形式　T_START(TIMER_ID, SEC)
TM_START(TIMER_ID, SEC)

概　要　タイマを起動します。

引数の説明　TIMER_ID　: 起動するタイマのタイマ番号 (1 ~ 16777215)
SEC　: タイムアウト値
関数 T_START では、1 秒単位　(0 ~ 16777215)
関数 TM_START では、100m 秒単位　(0 ~ 16777215)

関数値　0　: 正常終了
–60　: 引数エラー
–120　: 最大同時起動タイマ数オーバ

使用例
```
T_START(201,1)
T202=202
S=2
T_START(T202,S)
```

機能説明　引数で指定したタイマ番号のタイマを起動する関数です。
この関数が実行されたときに、同じタイマ番号のタイマが既に起動されている場合は、そのタイマを停止し、新たにタイマを起動させます。

制限事項　タイムアウト値に 16777215 をこえる値が設定されると引数エラーになります。
最大同時起動タイマ数以上タイマが起動されると最大同時起動タイマ数オーバのエラーになります。
最大同時起動タイマ数は 30 個です。

(8) T_STOP

呼出し形式　T_STOP(TIMER_ID)

概　要　タイマを停止します。

引数の説明　TIMER_ID　: 停止するタイマのタイマ番号 (1 ~ 16777215)

関数値　0　: 正常終了
–60　: 引数エラー
–121　: 起動しているタイマがない

使用例
```
T_STOP(201)
T202=202
T_STOP(T202)
```

機能説明　引数で指定したタイマ番号のタイマを停止する関数です。

制限事項　引数で指定したタイマ番号のタイマが起動していない場合は、エラー・コードを関数値として返すだけで、他に何もしません。

(9) READ_TIMER

呼出し形式　READ_TIMER( )

概　要　タイムアウトしたタイマの ID を得ます。

引数の説明

関数値　1 ～16777215　：タイムアウトしたタイマの番号
–121　：タイムアウトしたタイマがない

使用例

```
T_START(1,5)
T_START(2,10)
RET=RECEIVE(0)
IF RET==0 THEN
   TIMER=READ_TIMER( )
END
PPINT("TIMER=%D¥N",TIMER)
```

機能説明　タイムアウトしたタイマの番号を知ることができます。
タイマの使い方は、RECEIVE 関数の説明を参考にして下さい。

(10) SEND_EVENT

呼出し形式　SEND_EVENT(DEST, MSG)

概　要　シミュレータにイベントを送ります。

引数の説明
DEST　：イベントの送り先
　10　：D チャネル・シミュレータ
　20　：B チャネル・シミュレータ
　上記以外　：登録したイベント ID
MSG　：送出するイベント (0 ~ 16777215)

関数値
0　：正常終了
–60　：引数エラー
–155　：イベントの送信に失敗した

使用例
送出元のシミュレータ
```
SEND_EVENT(123, 50)
```
送出先のシミュレータ
```
REG_EVTID(123)
RET = RECEIVE(0)
IF RET == 200 THEN  /* 登録されたイベント ID を使用したイベントを受信した場合 */
    EVT = READ_EVENT( )
    PRINT("RECEIVED EVENT:%D¥N", EVT)
END
REL_EVTID(123)
```

機能説明
シミュレータにイベント ID を送出します。
引数として、送出先とイベント内容を指定します。送出先には、送出先のシミュレータが登録しているイベント ID を指定します。また、D5112 との互換性を考慮し、送出先に 10, 20 を指定した場合は同じシミュレーション番号を持つ D チャンネル・シミュレータ、B チャンネル・シミュレータに送られます。シミュレーション番号とは、LAPBx シミュレータにおける x の値です。(x = 1 ~ 4)

制限事項
イベントの送り先に 10、20、登録したイベント ID 以外を指定すると引数エラーになります。また、送信するイベントに範囲外の値を指定したときも引数エラーになります。
送信先の受信キューが満杯などの理由で、イベントが送信先に送られなかった場合は“イベントの送信に失敗した”ことを示す関数値を返します。

(11) READ_EVENT

呼出し形式	READ_EVENT( )
概　要	他チャネルから送られてきたイベントの内容を読み出します。
引数の説明	
関数値	0～16777215: イベント内容 –156　　: イベント未受信
使用例	RET=RECEIVE(0) IF RET==10 THEN 　MSG=READ_EVENT( ) 　PRINT("MESSAGE=[%X]¥N",MSG) END
機能説明	他チャネルから送られてきたイベントの内容を知ることができます。他チャネルからのイベントの送出があると、そのイベントの発生をRECEIVE 関数により知ることができます。RECEIVE 関数の関数値により、他チャネルからのイベントの受信があったのを確認した後、本関数を実行するとイベント内容を知ることができます。

(12) EXTRACT

呼出し形式　EXTRACT(n)

概　要　受信フレームの任意のオクテットの値を読み出します。

引数の説明　n　:読み出すデータのオクテット値
(1～受信フレーム長) の値が設定可能です。

関数値　0 ～255　:読み出したデータの値
–60　:引数エラー
–80　:フレーム未受信エラー

使用例
```
CF1=EXTRACT(3)
AA=4
CF2=EXTRACT(AA)
```

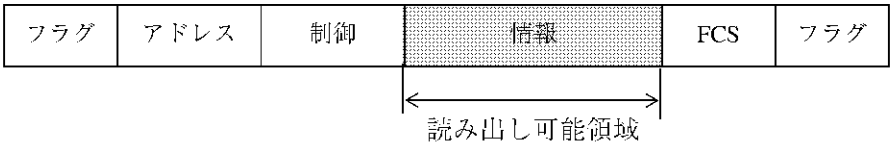
機能説明　受信したフレームの任意のオクテットの値を読み出す関数です。

注意事項　トランスペアレント・モードとレイヤ 2 自動とでは読み出せるデータの領域が違います。(下図参照)
下図で斜線部の領域が読み出し可能領域です。また、読み出す位置の指定は斜線部の左側から順に、1 から数えたオクテットで行います。

制限事項　引数で設定したオクテットのデータを関数値として返します。引数は、1 ～受信フレーム長までの値が設定可能です。それ以外を設定すると引数エラーになります。また、フレームを受信していないか RECEIVE 関数を実行していない状態でこの関数が実行されるとフレーム未受信エラーになります。

① トランスペアレント・モード

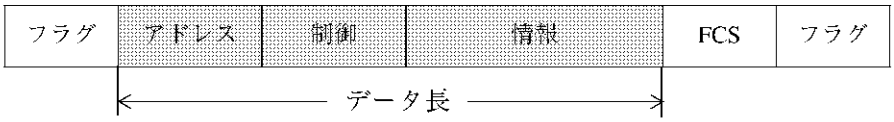
② レイヤ 2 自動モード



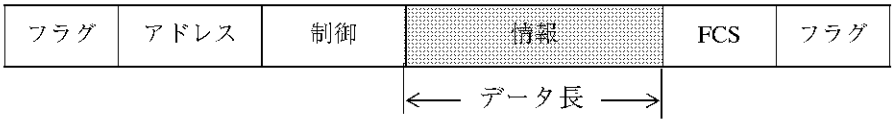
(13) RXFRLEN

呼出し形式	RXFRLEN( )
概　要	受信フレーム・データ長を調べます。
引数の説明	
関数値	1 ～4200　：受信フレームのデータ長 –80　：フレーム未受信エラー
使用例	LENGTH=RXFRLEN( )
機能説明	受信フレームのデータ長を関数値として返します。
注意事項	トランスペアレント・モードとレイヤ 2 自動モードでは、データ長の定義が違います。(下図参照)
制限事項	この関数はフレームを受信していないか、またはフレームを受信していても RECEIVE 関数を実行していないと、フレーム未受信エラーとなります。

① トランスペアレント・モード



② レイヤ 2 自動モード



(14) WAIT

呼出し形式	WAIT(SEC)
概　要	指定の時間プログラムを停止します。
引数の説明	SEC　:待機する時間 100msec 単位 (0 ~ 16777215)
関数値	0　:正常終了 –60　:引数エラー
使用例	WAIT(10) SEC=5 WAIT(SEC)
機能説明	指定した時間プログラムの実行を停止します。

(15) PH_ACT

呼出し形式	PH_ACT( )
概　要	レイヤ 1 を起動します。
引数の説明	
関数値	0　: 正常終了 –160　: レイヤ 1 エラー (起動しない) –161　: インタフェース・エラー (PH_ACT をサポートしないインタフェースである)
使用例	PH_ACT( )
機能説明	レイヤ 1 の起動を行う関数です。
注意事項	レイヤ 1 の起動に失敗すると、レイヤ 1 エラーを示す関数値が返ります。

(16) PH_DEACT

呼出し形式　PH_DEACT( )

概　要　レイヤ 1 を停止させます。

引数の説明

関数値

0	: 正常終了
–53	: NT モードエラー (NT モードの宣言がされていない)
–160	: レイヤ 1 エラー (停止しない)
–161	: インタフェース・エラー (PH_DEACT をサポートしないインタフェースである)

使用例　PH_DEACT( )

機能説明　レイヤ 1 を停止する関数です。

制限事項　NT モード以外では、実行できません。

(17) READ_VAL

呼出し形式　READ_VAL( )

概　要　キー入力された数値を読み出します。

引数の説明

関数値

–2147483648 ～2147483647:	キー入力された数値
–175:	キー入力されていない

使用例

```
RET=RECEIVE(0)
IF RET==2 THEN
   VAL=READ VAL( )
   PRINT("KEY VALUE=%D¥N",VAL)
END
```

機能説明　Command ラインから入力された数値を読み出します。Command ラインから入力された数値を読み出す関数として INPUT 関数がありますが、INPUT 関数によりキー入力受信待ち状態になっているときは、INPUT 関数が優先されます。従って、RECEIVE 関数にキー入力の情報は渡されません。
INPUT 関数による受信待ち状態か、RECEIVE 関数による受信待ち状態かは、画面下部に表示されているアイコン表示で識別します。
(それぞれ、"INPUT","RCV?" と表示されます。)
Command ラインから入力できる数値は、–2147483648 ～2147483647 の範囲です。したがって、キー入力されていないときに返すエラー値と実際に入力された値とで、区別がつかないことがあるので注意して下さい。

(18) REG_EVTID

呼出し形式　　REG_EVTID(EVTID)

概　要　　イベント ID を登録する関数です。

引数の説明　　EVTID　　: 登録するイベント ID
10 と 20 はそれぞれ D チャンネル、B チャンネル用に予約されているので使用できません。

関数値　　0　　: 正常に登録された
–170　　: イベント用のリソースに空きがなく登録できなかった。

使用例

```
REG_EVTID(1000)
RET = RECEIVE(0)
IF RET==200 THEN
   EVT = READ_EVENT( )
   PRINT("EVENT=%X¥N", EVT)
END
```

機能説明　　イベント ID は独立したシミュレータ間で通信をする場合に使用する ID です。
2 つのシミュレータ間で通信をする場合、受信側のシミュレータは共通で認識するイベント ID を登録します。(送信側は イベント ID の登録をする必要はありません。ただし、送信側も受信する場合は別のイベント ID を登録します) 送信側は SEND_EVENT 関数で DEST にこのイベント ID を使用して送信します。受信側はイベントを受信したことを RECEIVE 関数で知ることができ、その内容については READ_EVENT 関数で読み出します。
登録することのできるイベント ID は、他のシミュレータで登録されたものを含め、システム全体で 128 個までです。

(19) REL_EVTID

呼出し形式　　REL_EVTID(EVTID)

概　要　　登録してあるイベント ID を解放する関数です。

引数の説明　　EVTID　　: 解放するイベント ID
10 と 20 はそれぞれ D チャンネル、B チャンネル用に
予約されているので使用できません。

関数値　　0　　: 正常に解放された
–60　　: 引数で指定されたイベントは登録されていない

使用例

```
REG_EVTID(1000)
RET = RECEIVE(0)
IF RET==200 THEN
   EVT = READ_EVENT( )
   PRINT("EVENT=%X¥N", EVT)
END
REL_EVTID(1000)
```

機能説明　　登録されているイベントを解放する関数です。
登録できるイベントには限りがあるので、使用を終了したイベントは
解放するようにして下さい。

(20) SEND_EVTS

呼出し形式	SEND_EVTS(EVENT_ID)
概　要	他チャネルにイベントを送ります。
引数の説明	EVENT_ID : 送信するイベント ID
関数値	0 : 正常終了
	-60 : 引数エラー
	-155 : イベントの送信に失敗した
	-156 : 送信するイベントがない
使用例	WRITE_EVTS(1, 100)
	WRITE_EVTS(2, 101)
	SEND_EVTS(1234)
機能説明	他のチャネルにイベントを送ります。
	送信するイベントはWRITE_EVTS() 関数で書き込みます。
	WRITE_EVTS() 関数で設定したイベントは、一度送信するとクリアされます。しかし、エラーで終了した場合はクリアされません。
	イベントを送信する関数として SEND_EVENT() 関数がありますが、SEND_EVENT() 関数が 1 ロングワードしか送信できないのに対し、本関数では、WRITE_EVTS() 関数で指定した数を一度に送信することができます。SEND_EVENT(DEST, EVENT) は以下のように置き換えることができます。
	WRITE_EVTS(1, EVENT)
	SEND_EVTS(DEST)
	ここで送られるイベントを受信するためには、受信側でイベント ID を登録する必要があります。
	(詳細は REG_EVTID, REL_EVTID を参照して下さい)

(21) WRITE_EVTS

呼び出し形式	WRITE_EVTS(N, EVENT)	
概 要	他チャネルに送信するイベントを書き込みます。	
引数の説明	N	：書き込むイベントの位置 (1 ～ 256)
	EVENT	：書き込むイベントの値 (-2147483648 ～ 2147483647)
関数値	0	：正常終了
	-60	：引数エラー
使用例	WRITE_EVTS(1, 100) WRITE_EVTS(2, 101) SEND_EVTS(1234)	
機能説明	他のチャネルに送信するイベントを書き込みます。 書き込んだイベントを送信するときは、SEND_EVTS() 関数を使用します。	

注意　WRITE_EVTS 関数では、-2147483648 ～ 2147483647 までを許容していますが、エラー値と同じ値を使用すると読み出す関数 (READ_EVTS) で識別が付きません。

(22) READ_EVTS

呼び出し形式　READ_EVTS(N)

概 要　他チャネルから送信されたイベントを読み出します。

引数の説明　N　：読み出すイベントの位置

関数値　-2147483648 ～ 2147483647:
読み出すイベント値

-60　：引数エラー

-156　：イベント未受信エラー

使用例

```
REG_EVTID(1234)
RET = RECEIVE(0)
IF RET == 200 THEN
  N = COUNT_EVT()
  FOR I=1 TO N DO
    EVENT = READ_EVTS(I)
    PRINT("RECEIVED EVENT = %D¥N", EVENT)
  END
END
REL_EVTID(1234)
```

機能説明　他のチャネルから送信されたイベントを読み出します。

読み出すイベントは引数で指定します。

同様の関数に READ_EVENT() 関数がありますが、READ_EVENT() 関数が 1 ロングワードしか読み出すことができないのに対し、本関数は引数で指定することにより複数のロングワードを読み出すことができます。READ_EVENT() 関数は READ_EVTS(1) 関数で置き換えることができます。

注意　READ_EVTS 関数は引数で指定することにより複数のロングワードを読み出すことができますが、エラー値との識別が付かなくなることがあります。

---

(23) COUNT_EVT

呼び出し形式　COUNT_EVT()

概 要　他チャネルから送信されたイベント数を得ます。

引数の説明

関数値　1 ～ 256　: 受信したイベント数

-156　: イベント未受信エラー

使用例

```
REG_EVTID(1234)
RET = RECEIVE(0)
IF RET == 200 THEN
  N = COUNT_EVT()
  FOR I=1 TO N DO
    EVENT = READ_EVTS(I)
    PRINT("RECEIVED EVENT = %D¥N", EVENT)
  END
END
REL_EVTID(1234)
```

機能説明　他チャネルから送信されたイベント数を得ます。

(24) GET_SIMID

呼び出し形式　GET_SIMID()

概 要　現在実行中のシミュレーション ID を得ます。

引数の説明

関数値　0 ～ 268435455: シミュレーション ID

上位 16 ビット

LAPD シミュレータ：0x0281

LAPB シミュレータ：0x0282

下位 16 ビット

シミュレーション番号

使用例

```
SID = GET_SIMID()
PRINT("MY SIMULATION ID = [%X]¥N", SID)
```

機能説明　現在実行中のシミュレーション ID を得ます。ここで得るシミュレーション ID は本器でユニークな番号になります。

シミュレーション番号は、例えば LAPB シミュレータ x における x の値で、シミュレータ ID の下位 16 ビットに使われます。

(25) GET_LAY1ID

呼び出し形式　GET_LAY1ID()

概 要　現在実行中のレイヤ 1 インタフェース ID を得ます。

引数の説明

関数値　0 ～ 268435455：インタフェース ID

上位 16 ビット

基本インタフェース：0x0181

一次群インタフェース：0x0182

U 点インタフェース：0x0183

下位 16 ビット

レイヤ 1 インタフェース番号

使用例

```
LAY1_ID = GET_LAY1ID()
PRINT("MY LAYER1 ID = [%X]¥N", LAY1_ID)
```

機能説明　現在実行中のシミュレータが使用するレイヤ 1 インタフェース ID を得ます。ここで得るレイヤ 1 インタフェース ID は本器でユニークな番号になります。

レイヤ 1 インタフェース番号は、例えば UIx-y における y の値で、レイヤ 1 インタフェース ID の下位 16 ビットに使われます。

9.2 トランスペアレント・モード用関数

表 9-2 トランスペアレント・モード用関数

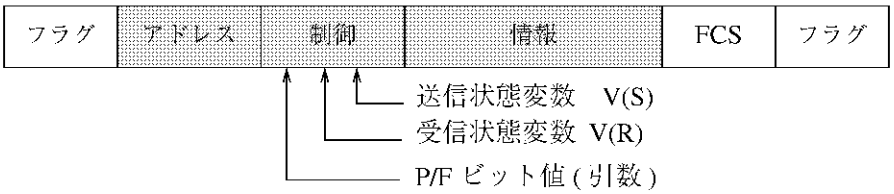
フレーム送信関連	
(1) SENDT	フレームを送信する。(メッセージ・データをそのまま送出)
(2) SENDF	フレームを送信する。(N(S),N(R),P/F を挿入して送出)
(3) INCVS	送信状態変数 V(S) を＋1する。
(4) INCVR	受信状態変数 V(R) を＋1する。
(5) SETVS	送信状態変数 V(S) の値を設定する。
(6) SETVR	受信状態変数 V(R) の値を設定する。
(7) INS_ADRS	送信フレームのアドレス値を書き換える。
(8) INS_NR	送信フレームの N(R) 値を書き換える。
(9) INS_NS	送信フレームの N(S) 値を書き換える。
(10) INS_PF	送信フレームの P/F ビット値を書き換える。
(11) INS_TYPE	送信フレームのフレーム種別を書き換える。
(12) INS_CF1	送信フレーム制御フィールドの第 1 オクテットの値を書き換える。
(13) INS_CF2	送信フレーム制御フィールドの第 2 オクテットの値を書き換える。
(14) INS_FRCF1	送信 FRMR フレーム情報フィールド内の第 1 オクテット値を書き換える。
(15) INS_FRCF2	送信 FRMR フレーム情報フィールド内の第 2 オクテット値を書き換える。
(16) INS_FRVS	送信 FRMR フレーム情報フィールド内の V(S) 値を書き換える。
(17) INS_FRVR	送信 FRMR フレーム情報フィールド内の V(R) 値を書き換える。
(18) INS_FRCR	送信 FRMR フレーム情報フィールド内の C/R ビット値を書き換える。
(19) INS_FRWXYZ	送信 FRMR フレーム情報フィールド内の WXYZ ビット値を書き換える。
フレーム受信関連	
(20) RXADRS	受信フレームのアドレス値を読み出す。
(21) RXNR	受信フレームの N(R) 値を読み出す。
(22) RXNS	受信フレームの N(S) 値を読み出す。
(23) RXPF	受信フレームの P/F ビット値を読み出す。
(24) RXTYPE	受信フレームのフレーム種別を読み出す。
(25) RXCF1	受信フレーム制御フィールドの第 1 オクテットの値を読み出す。
(26) RXCF2	受信フレーム制御フィールドの第 2 オクテットの値を読み出す。
(27) RXFRCF1	受信 FRMR フレーム情報フィールド内の第 1 オクテット値を読み出す。
(28) RXFRCF2	受信 FRMR フレーム情報フィールド内の第 2 オクテット値を読み出す。
(29) RXFRVS	受信 FRMR フレーム情報フィールド内の V(S) 値を読み出す。
(30) RXFRVR	受信 FRMR フレーム情報フィールド内の V(R) 値を読み出す。
(31) RXFRCR	受信 FRMR フレーム情報フィールド内の C/R ビット値を読み出す。
(32) RXFRWXYZ	受信 FRMR フレーム情報フィールド内の WXYZ ビット値を読み出す。

(1) SENDT

呼出し形式	SENDT("NAME")
概　要	フレームを送信します。(メッセージ・データをそのまま送出)
引数の説明	NAME　　: フレーム名
関数値	0　　: 正常終了 –50　　: トランスペアレント・モード・エラー –61　　: フレーム名エラー
使用例	SENDT("SPL1")
機能説明	トランスペアレント・モードでのフレームの送出を行う関数です。 引数で指定した名前のフレームをそのまま送出します。
注意事項	SENDF 関数では、内部の状態変数 V(S),V(R) を N(S),N(R) としてフレーム中に挿入して送出しますが、本関数は一切加工せずに同線上に送出します。
制限事項	レイヤ 2 自動モードでは、実行できません。

(2) SENDF

呼出し形式	SENDF("NAME", PF)
概　要	フレームを送信します。(N(S),N(R),P/F を挿入して送出)
引数の説明	NAME　: 送出するフレーム名 PF　: 送出するフレームの P/F ビット値 (0, 1)
関数値	0　: 正常終了 –50　: トランスペアレント・モード・エラー –60　: 引数エラー –61　: フレーム名エラー
使用例	SENDF("ID_REQ",0) PF= 1 SENDF("SABME",PF)
機能説明	トランスペアレント・モードでのフレームの送出を行う関数です。 引数で指定されたフレーム名のフレームを送出します。その際、引数で指定された P/F ビット値、送信シーケンス番号 N(S)、または受信シーケンス番号 N(R) を付加して送信します。 内部の状態変数 V(S)、V(R) を、N(S)、N(R) としてフレームに挿入します。 送信状態変数 V(S) は、INCVS 関数や SETVS 関数により変更が可能です。 また、受信状態変数 V(R) も、同様に INCVR 関数や SETVR 関数により変更が可能です。 SENDF 関数によりフレーム送信する際に付加されるデータを下図に示します。
制限事項	レイヤ 2 自動モードでは、実行できません。



(3) INCVS

呼出し形式	INCVS( )
概　要	送信状態変数 V(S) を＋ 1 します。
引数の説明	
関数値	0　　: 正常終了 –50　　: トランスペアレント・モード・エラー
使用例	INCVS( )
機能説明	送信状態変数 V(S) の値を＋ 1 更新する関数です。 この値は、番号制情報フレーム (I フレーム) を送出するときにのみ N(S) 値としてフレームに付加されます。
制限事項	この関数は、レイヤ 2 自動モードでは実行できません。

(4) INCVR

呼出し形式	INCVR( )
概　要	受信状態変数 V(R) を＋ 1 します。
引数の説明	
関数値	0　　: 正常終了 –50　　: トランスペアレント・モード・エラー
使用例	INCVR( )
機能説明	受信状態変数 V(R) の値を＋ 1 更新する関数です。 この値は、番号制情報フレーム (I フレーム) と番号制監視フレーム (S フレーム) を送出するときに N(R) 値として、送信フレームに付加されます。
制限事項	この関数は、レイヤ 2 自動モードでは実行できません。

(5) SETVS

呼出し形式　SETVS(VS)

概　要　送信状態変数 V(S) の値を設定します。

引数の説明
VS　: 設定する V(S) 値
モジュロ 8 宣言時 : 0 ~ 7
モジュロ 128 宣言時 : 0 ~ 127

関数値
0　: 正常終了
–50　: トランスペアレント・モード・エラー
–60　: 引数エラー

使用例
```
SETVS(5)
VS=6
SETVS(VS)
```

機能説明　送信状態変数 V(S) の値を設定する関数です。
この値は、番号制情報フレーム (I フレーム) を送出するときにのみ N(S) 値として送信フレームに付加されます。

制限事項　この関数は、レイヤ 2 自動モードでは実行できません。

(6) SETVR

呼出し形式　SETVR(VR)

概　要　受信状態変数 V(R) の値を設定します。

引数の説明
VR　: 設定する V(R) 値
モジュロ 8 宣言時 : 0 ~ 7
モジュロ 128 宣言時 : 0 ~ 127

関数値
0　: 正常終了
–50　: トランスペアレント・モード・エラー
–60　: 引数エラー

使用例
```
SETVR(5)
VR=6
SETVR(VR)
```

機能説明　受信状態変数 V(R) の値を設定する関数です。
この値は、番号制情報フレーム (I フレーム) と番号制監視フレーム (S フレーム) を送出するときにのみ、N(R) 値として送信フレームに付加されます。

制限事項　この関数は、レイヤ 2 自動モードでは実行できません。

(7) ~ (19)　LAPB パラメータ挿入関数群

	[関数名]	[挿入するパラメータ]
呼出し形式	(7)　INS_ADRS("NAME",ADRS)	アドレス値
	(8)　INS_NR("NAME",NR)	N(R) 値
	(9)　INS_NS("NAME",NS)	N(S) 値
	(10) INS_PF("NAME",PF)	P/F ビット値
	(11) INS_TYPE("NAME",TYPE)	フレーム種別
	(12) INS_CF1("NAME",CF1)	制御フィールド第 1 オクテット値
	(13) INS_CF2("NAME",CF2)	制御フィールド第 2 オクテット値
	(14) INS_FRCF1("NAME",FRCF1)	FRMR フレーム情報フィールド内第 1 オクテット値
	(15) INS_FRCF2("NAME",FRCF2)	FRMR フレーム情報フィールド内第 2 オクテット値
	(16) INS_FRVS("NAME",FRCVS)	FRMR フレーム情報フィールド内 V(S) 値
	(17) INS_FRVR("NAME",FRVR)	FRMR フレーム情報フィールド内 V(R) 値
	(18) INS_FRCR("NAME",FRCR)	FRMR フレーム情報フィールド内 C/R ビット値
	(19) INS_FRWXYZ("NAME",FRWXYZ)	FRMR フレーム情報フィールド内 WXYZ ビット値

概　要　　送出フレームの任意のパラメータを書き換えます。

引数の説明

NAME	: フレーム名
ADRS	: アドレス値
NR	: N(R) 値
NS	: N(S) 値
PF	: P/F ビット値
TYPE	: フレーム種別
CF1	: 制御フィールド第 1 オクテット値
CF2	: 制御フィールド第 2 オクテット値
FRCF1	: FRMR フレーム情報フィールド内第 1 オクテット値
FRCF2	: FRMR フレーム情報フィールド内第 2 オクテット値
FRVS	: FRMR フレーム情報フィールド内 V(S) 値
FRVR	: FRMR フレーム情報フィールド内 V(R) 値
FRCR	: FRMR フレーム情報フィールド内 C/R ビット値
FRWXYZ	: FRMR フレーム情報フィールド内 WXYZ ビット値

関数値

0	: 正常終了
–50	: トランスペアレント・モードエラー
–60	: 引数エラー
–61	: フレーム名エラー
–85	: 書き換えられるメッセージが不適切

使用例

```
INS_ADRS("SPL1",1)
SENDF("SPL1",0)
```

機能説明

メッセージ中の任意の LAPB パラメータを挿入する関数群です。
本関数群は、メッセージの先頭から内容を解釈していき、挿入する LAPB パラメータの位置を探していきます。
挿入する LAPB パラメータの位置が発見されると、そこを引数で指定された値に書き換えます。
このとき挿入する LAPB パラメータの位置が発見されないと”書き換えられるメッセージが不適切である”という意味の関数値を返します。

補足説明

INS_TYPE("NAME",TYPE) 関数の引数 TYPE とフレーム種別は以下の対応となっています。

フレーム種別	TYPE		フレーム種別	TYPE	
	10 進	16 進		10 進	16 進
I	0	00	DM	15	0F
RR	1	01	UI	3	03
RNR	5	05	DISC	67	43
REJ	9	09	UA	99	63
SABME	111	6F	FRMR	135	87
SABM	47	2F	XID	175	AF

制限事項

本関数は、レイヤ 2 自動モードでは実行できません。

(20) ~ (32)　LAPB パラメータ読み出し関数群

呼出し形式

[関数名]	[読み出すパラメータ]
(20) RXADRS( )	アドレス値
(21) RXNR( )	N(R) 値
(22) RXNS( )	N(S) 値
(23) RXPF( )	P/F ビット値
(24) RXTYPE( )	フレーム種別
(25) RXCF1( )	制御フィールド第 1 オクテット値
(26) RXCF2( )	制御フィールド第 2 オクテット値
(27) RXFRCF1( )	FRMR フレーム情報フィールド内第 1 オクテット値
(28) RXFRCF2( )	FRMR フレーム情報フィールド内第 2 オクテット値
(29) RXFRVS( )	FRMR フレーム情報フィールド内 V(S) 値
(30) RXFRVR( )	FRMR フレーム情報フィールド内 V(R) 値
(31) RXFRCR( )	FRMR フレーム情報フィールド内 C/R ビット値
(32) RXFRWXYZ( )	FRMR フレーム情報フィールド内 WXYZ ビット値

概　要　　受信フレームの任意のパラメータを読み出します。

引数の説明

関数値

0 ~ 255	: 受信フレームのパラメータ
–50	: トランスペアレント・モード・エラー
–80	: フレーム未受信エラー
–82	: 読み出すパラメータが存在しない

使用例

```
RECEIVE(0)
ADRS=RXADRS( )
PRINT("ADDRESS=%D¥N",ADRS)
```

機能説明　　受信フレームの LAPB パラメータを読み出す関数です。
フレームを受信した順に読むことができます。
受信したフレームを読み出したい場合は、RECEIVE 関数を実行し、受信フレームを内部に取り込んでから本関数群を実行します。次の受信フレームを読み出したい場合は、再び RECEIVE 関数を実行します。このとき受信フレームがない場合は、受信待ち状態になります。

注意事項

受信したフレームがない状態や RECEIVE 関数を実行していない状態で本関数が実行されるとフレーム未受信エラーとなります。
また、本関数は受信したフレームを先頭から解釈していき、LAPB パラメータのある位置を探していきます。
LAPB パラメータのある位置が発見されると、その値を関数値として返し終了します。
しかし、LAPB パラメータが見つからないと”読み出すパラメータが存在しない”という意味のエラー値を返します。

補足説明

RXTYPE( ) 関数で返される関数値とフレーム種別は以下の対応となっています。

フレーム種別	関数値		フレーム種別	関数値	
	10 進	16 進		10 進	16 進
I	0	00	DM	15	0F
RR	1	01	UI	3	03
RNR	5	05	DISC	67	43
REJ	9	09	UA	99	63
SABME	111	6F	FRMR	135	87
SABM	47	2F	XID	175	AF

制限事項

本関数はトランスペアレント・モードで実行する関数のため、レイヤ 2 自動モードでは実行できません。

9.3 レイヤ 2 自動モード用関数

表 9-3 レイヤ 2 自動モード用関数 (1/2)

フレーム送信関連	
(1) SENDI	I フレームを送信する。
(2) SENDPKT	パケットを送信する。
(3) INCPS	送信順序番号 P(S) を＋１する。
(4) INCPR	受信順序番号 P(R) を＋１する。
(5) SETPS	送信順序番号 P(S) の値を設定する。
(6) SETPR	受信順序番号 P(R) の値を設定する。
(7) INS_GFI	送信パケットのゼネラル・フォーマット識別子の値を書き換える。
(8) INS_Q	送信パケットの Q ビット値を書き換える。
(9) INS_D	送信パケットの D ビット値を書き換える。
(10) INS_LCGN	送信パケットの論理チャネル・グループ番号を書き換える。
(11) INS_LCN	送信パケットの論理チャネル番号を書き換える。
(12) INS_TYP	送信パケットのパケット・タイプ識別子を書き換える。
(13) INS_PR	送信パケットの受信順序番号 P(R) を書き換える。
(14) INS_PS	送信パケットの送信順序番号 P(S) を書き換える。
(15) INS_M	送信パケットのモア・データ表示値を書き換える。
(16) INS_CLL	送信パケットの発呼ユーザ・アドレス長を書き換える。
(17) INS_CDL	送信パケットの着呼ユーザ・アドレス長を書き換える。
(18) INS_DA	送信パケットの着呼ユーザ・アドレスを書き換える。
(19) INS_SA	送信パケットの発呼ユーザ・アドレスを書き換える。
(20) INS_FL	送信パケットのファシリティ長を書き換える。
(21) INS_F	送信パケットのファシリティを書き換える。
(22) INS_CAUSE	送信パケットの切断 / リスタート / リセット原因を書き換える。
(23) INS_DIAG	送信パケットの診断符号を書き換える。
(24) SETDTL	送信パケット内のデータ長を書き換える。
(25) INS_DATA	送信パケット内のデータを書き換える。
リンク関連	
(26) LINKON	リンクの設定を行う。
(27) LINKOFF	リンクの解放を行う。
(28) WAIT_LINK	相手からのリンク設定待ち状態にする。
(29) L_STATUS	リンクが設定されているかを調べる。
(30) SET_BUSY	自局をビジー状態にする。
(31) REL_BUSY	自局のビジー状態を解除する。

表 9-3 レイヤ 2 自動モード用関数 (2/2)

フレーム受信関連	
(32) RXGFI	受信パケットのゼネラルフォーマット識別子の値を読み出す。
(33) RXQ	受信パケットの Q ビット値を読み出す。
(34) RXD	受信パケットの D ビット値を読み出す。
(35) RXLCGN	受信パケットの論理チャネル・グループ番号を読み出す。
(36) RXLCN	受信パケットの論理チャネル番号を読み出す。
(37) RXTYP	受信パケットのパケット・タイプ識別子を読み出す。
(38) RXPR	受信パケットの受信順序番号 P(R) を読み出す。
(39) RXPS	受信パケットの送信順序番号 P(S) を読み出す。
(40) RXM	受信パケットのモア・データ表示値を読み出す。
(41) RXCLL	受信パケットの発呼ユーザ・アドレス長を読み出す。
(42) RXCDL	受信パケットの着呼ユーザ・アドレス長を読み出す。
(43) RXDA	受信パケットの着呼ユーザ・アドレスを読み出す。
(44) RXSA	受信パケットの発呼ユーザ・アドレスを読み出す。
(45) RXFL	受信パケットのファシリティ長を読み出す。
(46) RXF	受信パケットのファシリティを読み出す。
(47) RXCAUSE	受信パケットの切断 / リスタート / リセット原因を読み出す。
(48) RXDIAG	受信パケットの診断符号を読み出す。
(49) RXDTL	受信パケット内のデータ長を読み出す。
(50) RXDATA	受信パケット内のデータを読み出す。

(1) SENDI

呼出し形式	SENDI("NAME")
概　要	I フレームを送信します。
引数の説明	NAME　　: フレーム名
関数値	0　　: 正常終了 –51　　: レイヤ 2 自動モード・エラー –61　　: フレーム名エラー –72　　: リンク未設定エラー –110　　: 送出フレーム長エラー
使用例	PH_ACT( ) LINKON( ) SENDI("SPL1")
機能説明	レイヤ 2 自動モードで、I フレームを送出する関数です。 引数で指定した名前のメッセージを I フレームとして送出します。 メッセージは、あらかじめメッセージ・ビルダで作成する必要があります。作成したメッセージに名前は必ず付けて下さい。 I フレームを送出するためにはレイヤ 2 リンクが設定されていなければなりません。リンクの設定は LINKON 関数を用いて行うか、相手局からのリンク設定要求を受け付けて行います。
注意事項	本関数でフレームが送出されない場合以下の点をチェックして下さい。 1. レイヤ 1 は起動しているか。 2. レイヤ 2 リンクは設定されているか。 3. メッセージ･ビルダで引数で指定した名前のメッセージを作成してあるか。

(2) SENDPKT

呼出し形式　SENDPKT("NAME", LCGN, LCN)

概　要　パケットを送信します。

引数の説明
NAME　：フレーム名
LCGN　：論理チャネルグループ番号 (0～15)
LCN　：論理チャネル番号 (0～255)

関数値
0　：正常終了
–51　：レイヤ２自動モード・エラー
–60　：引数エラー
–61　：フレーム名エラー
–72　：リンク未設定エラー
–110　：送出フレーム長エラー

使用例
```
PH_ACT( )
LINKON( )
LCGN=0
LCN=1
SENDPKT("CR",LCGN,LCN)
```

機能説明
レイヤ２自動モードで、パケット・フレームを送出する関数です。
引数で指定した名前のメッセージを１フレームとして送出します。
このとき引数で指定した LCGN と LCN を送出するメッセージに挿入します。また、送出するメッセージのタイプが DT パケットのときは、引数 LCGN,LCN で指定された論理チャネルの P(R),P(S) が挿入されます。同様、RR,RNR パケットのときも指定された論理チャネルの P(R) がメッセージ中に挿入されます。
メッセージは、あらかじめメッセージ・ビルダで作成しておく必要があります。作成したメッセージに名前を必ず付けて下さい。
本関数でフレームを送出するためにはレイヤ２リンクが設定されていなければなりません。リンクの設定は LINKON( ) 関数を用いて行うか、相手局からのリンク設定要求を受け付けて行います。

注意事項
本関数でフレームが送出されない場合、以下の点をチェックして下さい。

1. レイヤ１は起動しているか。
2. レイヤ２リンクは設定されているか。
3. メッセージ･ビルダで引数で指定した名前のメッセージを作成してあるか。

(3) INCPS

呼出し形式　INCPS(LCGN, LCN)

概　要　送信順序番号 P(S) を＋1します。

引数の説明
LCGN　: 論理チャネルグループ番号 (0～15)
LCN　: 論理チャネル番号 (0～255)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
INCPS(0,1)
SENDPKT("DT",0,1)
```

機能説明　送信順序番号 P(S) を＋1更新する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(S) 値を変更します。
この値は、DT パケットを送出するときにメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(4) INCPR

呼出し形式　INCPR(LCGN, LCN)

概　要　受信順序番号 P(R) を＋1します。

引数の説明
LCGN　: 論理チャネルグループ番号 (0～15)
LCN　: 論理チャネル番号 (0～255)

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
RECEIVE (0)
IF RXTYP==DT THEN
   LCGN=RXLCGN( )
   LCN=RXLCN( )
   INCPR (LCGN, LCN)
   SENDPKT("RR", LCGN, LCN)
END
```

機能説明　受信順序番号 P(R) を＋1更新する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(R) 値を変更します。
この値は、DT,RR,RNR パケットを送出するときにメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(5) SETPS

呼出し形式　SETPS(LCGN, LCN, PS)

概　要　送信順序番号 P(S) の値を設定します。

引数の説明

LCGN	: 論理チャネル・グループ番号 (0 ~ 15)
LCN	: 論理チャネル番号 (0 ~ 255)
PS	: 送信順序番号 (0 ~ 127) GFI ビットが「xx01」のとき : 0 ~ 7 GFI ビットが「xx10」のとき : 0 ~ 127

関数値

0	: 正常終了
–51	: レイヤ 2 自動モード・エラー
–60	: 引数エラー

使用例

```
RECEIVE (0)
IF RXTYP==RI THEN
   LCGN=RXLCGN( )
   LCN=RXLCN( )
   SETPR (LCGN, LCN, 0)
   SETPS (LCGN, LCN, 0)
   SENDPKT("RF", LCGN, LCN)
END
```

機能説明　送信順序番号 P(S) の値を設定する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(S) 値を変更します。
この値は、DT パケットを送出するときのみメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(6) SETPR

呼出し形式　SETPR(LCGN, LCN,PR)

概　要　受信順序番号 P(R) の値を設定します。

引数の説明
LCGN　: 論理チャネル・グループ番号 (0 ~ 15)
LCN　: 論理チャネル番号 (0 ~ 255)
PR　: 受信順序番号 (0 ~ 127)
GFI ビットが「xx01」のとき : 0 ~ 7
GFI ビットが「xx10」のとき : 0 ~ 127

関数値
0　: 正常終了
–51　: レイヤ 2 自動モード・エラー
–60　: 引数エラー

使用例
```
RECEIVE (0)
IF RXTYP==RI THEN
   LCGN=RXLCGN( )
   LCN=RXLCN( )
   SETPR (LCGN, LCN, 0)
   SETPS (LCGN, LCN, 0)
   SENDPKT("RF", LCGN, LCN)
END
```

機能説明　受信順序番号 P(R) の値を設定する関数です。
引数で指定した論理チャネル (LCGN と LCN) の P(R) 値を変更します。
この値は、DT,RR,RNR パケットを送出するときのみメッセージ中に挿入されます。

制限事項　本関数は、トランスペアレント・モードでは実行できません。

(7) ~ (25)　X.25 パラメータ挿入関数群

呼出し形式

[関数名]	[挿入するパラメータ]
(7) INS_GFI("NAME",GFI)	ゼネラル･フォーマット識別子 (GFI)
(8) INS_Q("NAME",Q)	Q ビット
(9) INS_D("NAME",D)	D ビット
(10) INS_LCGN("NAME",LCGN)	論理チャネル・グループ番号 (LCGN)
(11) INS_LCN("NAME",LCN)	論理チャネル番号 (LCN)
(12) INS_TYP("NAME",TYP)	パケット・タイプ識別子 (TYP)
(13 INS_PR("NAME",PR)	受信順序番号 (P(R))
(14) INS_PS("NAME",PS)	送信順序番号 (P(S))
(15) INS_M("NAME",M)	M ビット
(16) INS_CLL("NAME",CLL)	発呼ユーザ・アドレス長
(17) INS_CDL("NAME",CDL)	着呼ユーザ・アドレス長
(18) INS_DA("NAME",N,DA)	着呼ユーザ・アドレス (N 番目)
(19) INS_SA("NAME",N,SA)	発呼ユーザ・アドレス (N 番目)
(20) INS_FL("NAME",FL)	ファシリティ長
(21) INS_F("NAME",N,F)	ファシリティ (N オクテット目)
(22) INS_CAUSE("NAME",CAUSE)	原因
(23) INS_DIAG("NAME",DIAG)	診断符号
(24) SETDTL("NAME",LEN)	データ長
(25) INS_DATA("NAME",N,DATA)	データ (N オクテット目)

概　要　　送出フレームの任意のパラメータを書き換えます。

引数の説明

NAME	: フレーム名
GFI	: ゼネラル・フォーマット識別子 (GFI)
Q	: Q ビット
D	: D ビット
LCGN	: 論理チャネル・グループ番号 (LCGN)
LCN	: 論理チャネル番号 (LCN)
TYP	: パケット・タイプ識別子
PR	: 受信順序番号
PS	: 送信順序番号
M	: M ビット
CLL	: 発呼ユーザ・アドレス長
CDL	: 着呼ユーザ・アドレス長
N	: N 番目を指定
DA	: 着呼ユーザ・アドレス
SA	: 発呼ユーザ・アドレス
FL	: ファシリティ長
F	: ファシリティ
CAUSE	: 原因
DIAG	: 診断符号
LEN	: データ長
DATA	: データ

関数値
0 : 正常終了
–60 : 引数エラー
–61 : フレーム名エラー
–85 : 書き換えられるメッセージが不適切

使用例
INS_GFI("CR",1)
SENDPKT("CR",0,1)

機能説明
メッセージ中の任意のパラメータを書き換える関数群です。
本関数は、メッセージの先頭から内容を解釈していき、挿入するパラメータの位置を探します。
挿入するパラメータの位置が発見されるとそこを引数で指定された値に書き換えます。このとき該当するパラメータが複数の位置に渡って存在する場合、引数 N でその位置を確定させる必要があります。
(INS_DA, INS_SA, INS_F, INS_DATA 関数などがこれに当てはまります。)
また、メッセージ内容を解釈していった結果、挿入するパラメータの位置が存在しない場合、”書き換えられるメッセージが不適切” という意味の関数を返します。

補足説明
INS_TYP("NAME",TYP) 関数で、引数 TYP とパケット名は以下の対応となっています。

パケット名	関数値		パケット名	関数値	
	10 進	16 進		10 進	16 進
CR	11	0B	SF	255	FF
CN	11	0B	DT	0	00
CA	15	0F	RR	1	01
CC	15	0F	RNR	5	05
CQ	19	13	RQ	27	1B
CI	19	13	RI	27	1B
CF	23	17	RF	31	1F
SQ	251	FB	IT	35	23
SI	251	FB	IF	39	27

SETDTL("NAME",LEN) 関数は、挿入関数ではありませんが INS_DATA と密接な関係があります。
INS_DATA 関数がパケット中のデータを変えるのに対して SETDTL 関数は、そのデータ長を変える関数です。

(26) LINKON

呼出し形式　LINKON( )

概　要　リンクの設定を行います。

引数の説明

関数値

0	: 正常終了
–51	: レイヤ 2 自動モード・エラー
–100	: リンクの設定が拒否された
–131	: SABM/SABME コマンドを N2 回送出したが応答がない

使用例

```
PH_ACT( )
LINKON( )
SENDI("SPL1")
```

機能説明

レイヤ２リンクの設定を行う関数です。
このとき、アドレスとモジュロは宣言文によりあらかじめ宣言しておく必要があります。
レイヤ 2 リンクが設定されていないと、SENDI, SENDPKT 関数でフレームを送出することはできません。
すでにレイヤ２リンクが設定されている場合は、再設定となります。
レイヤ２リンクを管理する他の関数として以下のものがあります。

LINKOFF( )	: リンクの解放をする
WAIT_LINK(SEC)	: リンクの設定待ち状態にする
L_STATUS( )	: リンクの状態を調べる

制限事項　本関数は、レイヤ２自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(27) LINKOFF

呼出し形式　LINKOFF( )

概　要　リンクの解放を行います。

引数の説明

関数値

0	: 正常終了
–51	: レイヤ 2 自動モード・エラー
–102	: レイヤ２リンクが設定されていない
–131	:DISC コマンドを N2 回送出したが応答がない

使用例

```
PH_ACT( )
LINKON( )
LINKOFF( )
```

機能説明

レイヤ２リンクを解放する関数です。
レイヤ 2 リンクが設定されていない状態で本関数が実行された場合は、”レイヤ２リンクが設定されていない”という意味の関数値が返され終了します。

制限事項　本関数は、レイヤ２自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(28) WAIT_LINK

呼出し形式　WAIT_LINK(SEC)

概　要　相手からのリンク設定待ち状態にします。

引数の説明

SEC	: 相手からリンクが設定されるのを待つ時間 (100msec 単位) (0 ~ 16777215)

関数値

0	: 正常終了
–51	: レイヤ 2 自動モード・エラー
–60	: 引数エラー
–125	: タイム・アウト

使用例

```
PH_ACT( )
WAIT_LINK(3000)
SENDI("SPL1")
```

機能説明　レイヤ 2 リンクが設定されるまで待つ関数です。
すでにレイヤ 2 リンクが設定されている場合には、正常終了を示す関数値を返して終了します。
レイヤ 2 リンクを待つ時間は引数で指定できます。
引数で指定された時間待ってもレイヤ 2 リンクが設定されない場合は”タイムアウト”を示す関数値を返して終了します。

制限事項　本関数は、レイヤ 2 自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(29) L_STATUS

呼出し形式　L_STATUS( )

概　要　リンクが設定されているかを調べます。

引数の説明

関数値

0	: リンクが設定されていない
1	: リンクが設定されている
–51	: レイヤ 2 自動モード・エラー

使用例

```
PH_ACT( )
PRINT("LINK STATUS=%D¥N",L_STATUS( ))
LINKON( )
PRINT("LINK STATUS=%D¥N",L_STATUS( ))
```

機能説明　レイヤ 2 リンクが設定されているかどうかを調べる関数です。
レイヤ 2 リンクが設定されている場合には、関数値として 1 が返されます。逆にレイヤ 2 リンクが設定されていない場合には、関数値として 0 が返されます。

制限事項　本関数は、レイヤ 2 自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(30) SET_BUSY

呼出し形式　SET_BUSY( )

概　要　自局をビジー状態にします。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー
–102　：リンクが設定されていない

使用例
```
PH_ACT( )
LINKON( )
SET_BUSY( )
```

機能説明　自局をビジー状態にする関数です。
本関数を実行するためには、レイヤ 2 リンクが設定されている必要があります。
ビジー状態を解除するときには、REL_BUSY 関数を使用します。

制限事項　本関数は、レイヤ 2 自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(31) REL_BUSY

呼出し形式　REL_BUSY( )

概　要　自局のビジー状態を解除します。

引数の説明

関数値
0　：正常終了
–51　：レイヤ 2 自動モード・エラー
–102　：リンクが設定されていない
–109　：ビジー状態になっていない

使用例
```
PH_ACT( )
LINKON( )
SET_BUSY( )
WAIT(30)
REL_BUSY( )
```

機能説明　自局のビジー状態を解除する関数です。
本関数は SET_BUSY 関数で設定されたビジー状態を解除するときに使用します。

制限事項　本関数は、レイヤ 2 自動モードで実行する関数のため、トランスペアレント・モードでは実行できません。

(32) ~ (50)　X.25 パラメータ読み出し関数群

呼出し形式	[関数名]	[読み出すパラメータ]
	(32) RXGFI( )	ゼネラル・フォーマット識別子 (GFI)
	(33) RXQ( )	Q ビット
	(34) RXD( )	D ビット
	(35) RXLCGN( )	論理チャネル・グループ番号 (LCGN)
	(36) RXLCN( )	論理チャネル番号 (LCN)
	(37) RXTYP( )	パケット・タイプ識別子
	(38) RXPR( )	受信順序番号 P(R)
	(39) RXPS( )	送信順序番号 P(S)
	(40) RXM( )	M ビット
	(41) RXCLL( )	発呼ユーザ・アドレス長
	(42) RXCDL( )	着呼ユーザ・アドレス長
	(43) RXDA(N)	着呼ユーザ・アドレス (N 番目)
	(44) RXSA(N)	発呼ユーザ・アドレス (N 番目)
	(45) RXFL( )	ファシリティ長
	(46) RXF(N)	ファシリティ (N オクテット目)
	(47) RXCAUSE( )	原因
	(48) RXDIAG( )	診断符号
	(49) RXDTL( )	データ長
	(50) RXDATA(N)	データ (N オクテット目)

概　要　　受信フレームの任意のパラメータを読み出します。

引数の説明　　N　　: N 番目を指定

関数値

0 ~ 255	: 読み出したパラメータの値
–60	: 引数エラー
–80	: フレーム未受信エラー
–82	: 読み出すパラメータが受信フレーム中に存在しない

使用例

```
RECEIVE(0)
GFI=RXGFI( )
PRINT("GFI=%D¥N",GFI)
```

機能説明

受信フレームの X.25 パラメータを読み出す関数です。
フレームを受信した順に読むことができます。
受信フレームを読み出したい場合は RECEIVE 関数を実行し、受信フレームを内部に取り込んでから本関数群を実行します。次の受信フレームを読み出したい場合は、再び RECEIVE 関数を実行します。このとき受信フレームがない場合は、受信待ち状態になります。
本関数は受信フレームの先頭から内容を解釈していき、該当する X.25 パラメータを関数値として返します。該当の X.25 パラメータが複数存在する (RXSA,RXDA,RXF,RXDATA など) 場合、引数 N で何番目のパラメータを読み出すのかを指定する必要があります。このとき N が不適切であると引数エラーとなります。
また、メッセージの内容を解釈していった結果、読み出すパラメータが存在しない場合、”読み出すパラメータが受信フレーム中に存在しない” という意味の関数値を返します。

補足説明

これらの関数は、トランスペアレント・モードでも実行できます。
RXTYP( ) 関数で返される関数値とパケット名は、以下の対応となっています。

パケット名	関数値		パケット名	関数値	
	10 進	16 進		10 進	16 進
CR	11	0B	SF	255	FF
CN	11	0B	DT	0	00
CA	15	0F	RR	1	01
CC	15	0F	RNR	5	05
CQ	19	13	RQ	27	1B
CI	19	13	RI	27	1B
CF	23	17	RF	31	1F
SQ	251	FB	IT	35	23
SI	251	FB	IF	39	27

9.4 音声、 BERT 用関数

表 9-4 音声、BERT 用関数

関数名	内容
(1) SOUND_ON	音声シミュレーション 1 用ヘッド・セットの通話を可能にする。
(2) SOUND_OFF	音声シミュレーション 1 用ヘッド・セットの通話を止める。
(3) SOUND1_ON	音声シミュレーション 1 用ヘッド・セットの通話を可能にする。
(4) SOUND1_OFF	音声シミュレーション 1 用ヘッド・セットの通話を止める。
(5) VOLUME	音声シミュレーション 1 用ヘッド・セットの音量を変える。
(6) SET_PRBS	BERT で使用する擬似ランダム・パターンを指定する。
(7) SET_WORD	BERT で使用するワード・パターンを指定する。
(8) SET_ADDERR	BERT の際、送信側に挿入するエラーの割合を指定する。
(9) BERT_ON	BERT を行う。
(10) PTNPCM_ON	1kHz, 0dBm の基準正弦波用デジタル・パターンを出力する。
(11) PTNPCM_OFF	1kHz, 0dBm の基準正弦波用デジタル・パターンの出力を止める。

次ページ以降に各関数の説明をします。

(1) SOUND_ON

呼出し形式	SOUND_ON(TYPE)
概　要	ヘッドセットによる音声の送信、受信を有効にします。
引数の説明	TYPE　:発生するサウンドの種類 20　:B チャネル音声
関数値	0　:正常終了 –60　:引数エラー –181　:指定したシミュレータは使用できない
使用例	PH_ACT( ) SET_CH64K(21, 1) SOUND_ON(20)
機能説明	ヘッドセットにより、相手からの音声を聞くことができるとともに、相手に音声を伝えることもできます。 音を制御する関数として以下の関数があります。 SOUND_OFF( ) VOLUME(TYPE, VOL) SOUND1_ON(TYPE) SOUND1_OFF( )
制限事項	音は、レイヤ 1 が起動しているときのみ発生します。 SOUND_ON(20) は SOUND1_ON(1) と同じ動作をします。

(2) SOUND_OFF

呼出し形式　SOUND_OFF( )

概　要　ヘッドセットによる音声の送信、受信を止めます。

引数の説明

関数値
0　：正常終了
–181　：指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(21, 1)
SOUND_ON(20)
WAIT(50)
SOUND_OFF( )
```

機能説明　ヘッドセットによる音声の送信、受信を止める関数です。

(3) SOUND1_ON

呼出し形式　SOUND1_ON(TYPE)

概　要　音声シミュレーション 1 用ヘッドセットによる音声の送信、受信を有効にします。

引数の説明
TYPE　：符号化形式の選択
1　：64kbps, μ 則
2　：64kbps, A 則
3　：32kbps, μ 則
4　：32kbps, A 則

関数値
0　：正常終了
–60　：引数エラー
–181　：指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(21, 1)
SOUND1_ON(1)
```

機能説明　ヘッドセットにより、相手からの音声を聞くことができるとともに、相手に音声を伝えることもできます。
音を制御する関数として以下の関数があります。
SOUND_OFF( )
VOLUME(TYPE, VOL)
SOUND1_ON(TYPE)
SOUND1_OFF( )

制限事項　音は、レイヤ 1 が起動しているときのみ発生します。

(4) SOUND1_OFF

呼出し形式　SOUND1_OFF( )

概　要　ヘッドセットによる音声の送信、受信を止めます。

引数の説明

関数値
0　:正常終了
–181　:指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(21, 1)
SOUND1_ON(1)
WAIT(50)
SOUND1_OFF( )
```

機能説明　ヘッドセットによる音声の送信、受信を止める関数です。

(5) VOLUME

呼出し形式　VOLUME(TYPE, VOL)

概　要　マイク / ヘッドホンから出ている音量を変えます。

引数の説明
TYPE　:音量変化させる対象の選択
0　:マイク
1　:ヘッドホン
VOL　:音量値。
0　:最小音量 (無音)
/
63　:最大音量

関数値
0　:正常終了
–60　:引数エラー
–181　:指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(21, 1)
SOUND1_ON(1)
VOLUME(1, 50)
```

機能説明　マイク / ヘッドホンから出ている音量を変える関数です。

(6) SET_PRBS

呼出し形式　SET_PRBS(PRBS)

概　要　BERT で使用する擬似ランダムのパターンを指定します。

引数の説明　PRBS　:擬似ランダムのパターン選択

1	: 7 段の擬似ランダム
2	: 9 段の擬似ランダム
3	: 10 段の擬似ランダム
4	: 11 段の擬似ランダム
5	: 15 段の擬似ランダム
6	: 17 段の擬似ランダム
7	: 19 段の擬似ランダム
8	: 20 段の擬似ランダム
9	: 23 段の擬似ランダム

関数値

0	:正常終了
–60	:引数エラー
–181	:指定したシミュレータは使用できない

使用例

```
SET_PRBS(1)
CHECK=1000000
ERR=BERT_ON(CHECK)
PRINT("CHECK BITS=%D ERROR BITS=%D¥N", CHECK, ERR)
```

機能説明　BERT で使用する擬似ランダムのパターンを指定する関数です。
実際のエラー・レート試験は BERT_ON 関数で行います。
擬似ランダム・パターンの生成多項式を以下に示します。

基準パターン　:2^N-1

段数 N	生成多項式
7	X^7+X^6+1
9	X^9+X^5+1
10	$X^{10}+X^7+1$
11	$X^{11}+X^9+1$
15	$X^{15}+X^{14}+1$
17	$X^{17}+X^{14}+1$
19	$X^{19}+X^5+X^2+X^1+1$
20	$X^{20}+X^3+1$
23	$X^{23}+X^{18}+1$

(7) SET_WORD

呼出し形式　SET_WORD(WORD)

概　要　BERT で使用するワード・パターンを指定します。

引数の説明　WORD　: ワード・パターン (0～65535)

関数値
0　: 正常終了
–60　: 引数エラー
–181　: 指定したシミュレータは使用できない

使用例
```
SET_WORD(1234)
CHECK=1000000
ERR=BERT_ON(CHECK)
PRINT("CHECK BITS=%D ERROR BITS=%D¥N", CHECK, ERR)
```

(8) SET_ADDERR

呼出し形式　SET_ADDERR(ERR)

概　要　BERT で挿入するエラーのレートを指定します。

引数の説明　ERR　: エラー・レートの選択
0　: エラー挿入なし。
1　: 10^{-1} の割合でエラーを挿入する
2　: 10^{-2} の割合でエラーを挿入する
3　: 10^{-3} の割合でエラーを挿入する
4　: 10^{-4} の割合でエラーを挿入する
5　: 10^{-5} の割合でエラーを挿入する
6　: 10^{-6} の割合でエラーを挿入する

関数値
0　: 正常終了
–60　: 引数エラー
–181　: 指定したシミュレータは使用できない

使用例
```
SET_ADDERR(3)
SET_WORD(0)
CHECK=1000000
ERR=BERT_ON(CHECK)
PRINT("CHECK BITS=%D ERROR BITS=%D¥N", CHECK, ERR)
```

注意事項　10^{-1} の割合でエラーを挿入し、BER 試験を行うと同期がとれないため、正常に BER 試験を行うことができません。

(9) BERT_ON

呼出し形式　BERT_ON(BITS)

概　要　BERT 試験を行います。

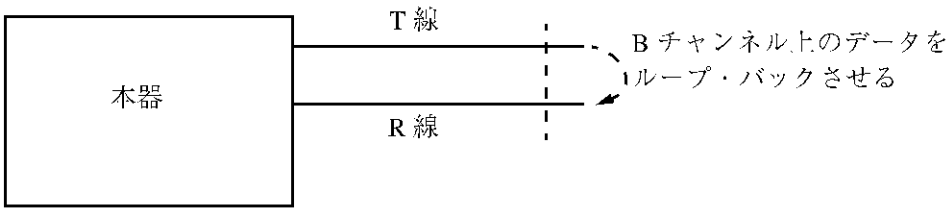
引数の説明　BITS　: BERT 試験を行うビット数 (0～16777215)

関数値
0～65535　: エラー・ビット数
–60　: 引数エラー
–165　: 同期外れ
–170　: カウンタのオーバ・フロー
–181　: 指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(11, 1)
SET_PRBS(1)
CHECK=1000000
ERR=BERT_ON(CHECK)
PRINT("CHECK BITS=%D ERROR BITS=%D¥N", CHECK, ERR)
```

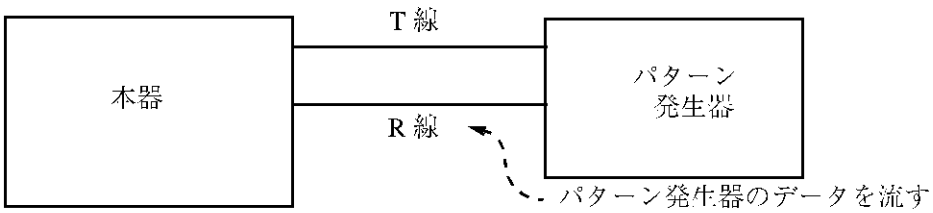
機能説明　引数 BITS ビットのエラー・レート試験を行います。
この引数 BITS の試験が終了すると関数値としてエラービット数を返します。
引数 BITS ビットとエラー・ビット数からエラー・レートが分かります。BERT 試験を行う環境として以下の2つのどちらかの接続が必要になります。

① ループ・バック試験



相手局またはエンド・ユーザなどで BERT 試験をしている B チャネルをループ・バックさせる必要があります。

② パターン発生器を使用した試験



相手局から本器で設定したパターンと同じものを発生させる必要があります。

本関数が実行されると、まずパターンの同期をとりにいきます。ある一定時間たっても同期がとれない場合、同期外れエラーとなります。
またエラー数が多く、内部のエラー・カウンタがオーバ・フローすると、" カウンタがオーバ・フローした " という意味の関数値を返して終了します。

(10) PTNPCM_ON

呼出し形式　PTNPCM_ON(TYPE)

概　要　1kHz. 0dBm の基準正弦波用デジタルパターン (CCITT 勧告 G.711 準拠) を出力します。

引数の説明　TYPE　: 符号化形式の選択
1　: 64kbps, μ 則
2　: 64kbps, A 則

関数値　0　: 正常終了
–60　: 引数エラー
–181　: 指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(11,1)
PTNPCM_ON(1)
```

機能説明　1kHz. 0dBm の基準正弦波用デジタルパターン (CCITT 勧告 G.711 準拠) を出力する関数です。

(11) PTNPCM_OFF

呼出し形式　PTNPCM_OFF( )

概　要　PTNPCM_ON の基準正弦波出力用デジタルパターンを止めます。

引数の説明

関数値　0　: 正常終了
–181　: 指定したシミュレータは使用できない

使用例
```
PH_ACT( )
SET_CH64K(11,1)
PTNPCM_ON(1)
WAIT(50)
PTNPCM_OFF( )
```

機能説明　PTNPCM_ON の基準正弦波出力用デジタルパターンを止める関数です。

10. 性能諸元

- モード　NT (網側)/TE (端末側)　(基本インタフェースと組み合わせ時)
  LT (交換局側)　(U 点インタフェースと組み合わせ時)
- Dch/Bch 共通

言語	PSL51(Protocol Simulation Language for D51 シリーズ)
プログラムの容量	128k Byte (Dch/Bch それぞれ独立にプログラムを作成する。)
メッセージの作成	独自の送出メッセージ作成メニュー (メッセージビルダ) により作成
モード	レイヤ 2 モード / レイヤ 3 モード (レイヤ 2 自動実行モード)
メッセージ・ビルダ	対応プロトコル以外に HEX 入力モードにより任意
タイマ	ソフトウェア・タイマ 30 個 (1 秒または 100 ミリ秒分解能)
関数	受信フレームの任意オクテット読み出し 作成したメッセージの送出 作成したメッセージの任意オクテットの内容変更 タイマの起動・停止 レイヤ 1 の起動・停止 (基本インタフェース時) 等 使用チャネルの指定

- Dch シミュレーション

対応プロトコル	Q.921(LAPD), Q.931, X.25
関数	Bch プログラムとのイベント送信 / 受信

- Bch シミュレーション

対応プロトコル	X.25
関数	Dch プログラムとのイベント送信 / 受信
音声	付属ヘッドセットによる任意チャネルの音声入出力 (A-law / μ -law, 32k ADPCM / 64k PCM)
ビット・エラー測定	PRBS パターン (2^{n-1} n=7,9,10,11,15,17,19,20,23) WORD パターン (16 ビット)

- 外部入出力　任意チャネルへの入出力 (D-Sub 9 ピン)
- 一般仕様　使用環境範囲 ： ＋ 5 ～＋ 40 ℃、相対湿度：80% 以下
  保存環境範囲 ： － 10 ～＋ 60 ℃、相対湿度：80% 以下
  外形 ： 25 (W) × 158 (H) × 246 (D)
  質量 ： 600g 以下

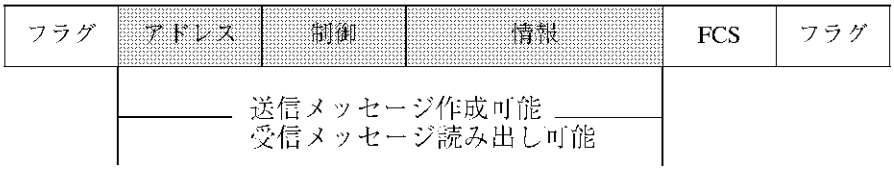
付録 シミュレーションを使いこなすために

A-1 トランスペアレント・モードとレイヤ 2 自動実行モードの相違

(1) トランスペアレント・モード

トランスペアレント・モードは、レイヤ 2 レベルでシミュレーションを行いたい場合に指定します。

フレーム中の以下の部分についてメッセージを作成したり、受信したフレームの内容を読み出したりすることができます。



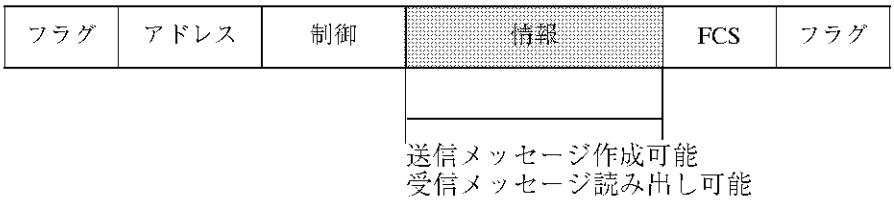
トランスペアレント・モードでフレームを送出する場合、送信状態変数 V(S) や受信状態変数 V(R) の値を自己管理する必要があります。

(2) レイヤ 2 自動実行モード

レイヤ 2 自動実行モードとは、レイヤ 3 レベルでシミュレーションを行いたい場合に指定します。

したがってレイヤ 2 レベルのフレームのやりとり (RR 送信、送・受信シーケンス番号の管理、TEI 管理手順など) を意識することなくレイヤ 3 レベルのシミュレーションを行うことができます。

メッセージを作成したり、受信したフレームの内容を読み出せる領域は情報部だけです。(下図参照)



(3) 使い分け

トランスペアレント・モードは、レイヤ 2 レベルのソフトのデバックを行う場合やレイヤ 2 で異常シーケンス (V(S),V(R) の値が正しくないなど) を起こしたい場合、異常フレーム (SAPI 値不適切など) を送出したい場合などに用います。

レイヤ 2 は正常シーケンスに指定し、レイヤ 3 のみをシミュレーションしたい場合にはレイヤ 2 自動実行モードを使用します。ただし、このレイヤ 2 自動実行モードではレイヤ 2 レベルで異常フレーム、および異常シーケンスを起こすことはできません。

目的に応じて使い分けて下さい。

A-2 タイマの使用方法

決められた時間だけしか、期待したフレームを待ちたくない場合にタイマを使用します。
タイマを使用するためには、以下の関数を使用します。

RECEIVE(タイマ番号)	: フレーム、タイムアウト・イベント、他チャネルからのイベント受信用の関数。
T_START(タイマ番号、タイムアウト値)	: タイマ番号のタイマを起動させる。(1 秒単位)
TM_START(タイマ番号、タイムアウト値	: タイマ番号のタイマを起動させる。(100m 秒単位)
T_STOP(タイマ番号)	: タイマ番号のタイマを停止させる。

[例 1]

10 秒ごとにシミュレーション画面に "TIME OUT!!" と表示させるプログラムについて示します。

```
FUNC MAIN( )                                  …関数の始まり
     PRINT("***  TIMER1. PRG  ***¥N")
     WHILE(1)                                 …永久ループの始まり

       T_START(1,10)                          …タイマ番号 1 のタイマを起動
       RECEIVE(1)                             …タイマ番号 1 で受信待ち
       PRINT("TIMEOUT!!¥N")                   …メッセージの表示

     END                                      …永久ループの終り
RETURN                                        …関数の終り
```

上記のプログラムで、RECEIVE(1) を RECEIVE(2) にするとタイマ番号 2 のタイマは起動されていないので、上記のプログラムのように 10 秒ごとに表示することはありません。

つまり、RECEIVE(タイマ番号) 関数は引数であるタイマ番号のタイムアウト・イベントのみを監視していることになります。したがって、引数以外のタイマ番号のタイマがタイムアウトしたとしても、この関数に対して何ら影響を与えません。

しかし、RECEIVE(0) のように引数タイマ番号が 0 のときは特別な動きをします。

RECEIVE(0) は、すべてのタイマのタイムアウト・イベントを受信します。また、このとき READ_TIMER( ) 関数を用いることにより、どのタイマがタイムアウトしたのかが分かります。タイマの停止は T_STOP(タイマ番号) 関数を使用することにより、引数のタイマ番号を持つタイマが停止します。

[例 2]

複数のタイマを起動させ、タイムアウトしたタイマの番号を表示させるプログラムを以下に示します。

受信関数には RECEIVE(0) を用い、すべてのタイムアウト・イベントの受信を許可します。

```
FUNC MAIN( )
     PRINT("***  TIMER2. PRG  ***¥N")
     T_START(1,10)                          …タイマ番号 1 のタイマを起動 ( タイムアウト値 10 秒 )
     T_START(2,12)                          …タイマ番号 2 のタイマを起動 ( タイムアウト値 12 秒 )
     T_START(3,14)                          …タイマ番号 3 のタイマを起動 ( タイムアウト値 14 秒 )

     WHILE(1)

       RECEIVE(0)                           …タイムアウトしたタイマ番号を TIMER 変数に代入
       TIMER=READ_TIMER( )                  …タイムアウトしたタイマ番号の表示
       PRINT("TIMER ID=%D¥N",TIMER)

     END

RETURN
```

実行結果

```
***  TIMER2. PRG  ***
TIMER ID=1
TIMER ID=2
TIMER ID=3
```

[例 3]

タイムアウト値 10 秒のタイマを起動させ、タイムアウトすると "TIMEOUT!!" と表示し、タイムアウトする前にフレームを受信すると "RECEIVE FRAME" と表示するプログラムを以下に示します。

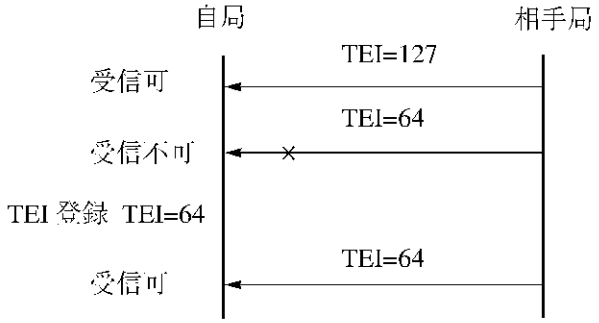
```
FUNC MAIN( )
     T_START(1,10)          …タイマ番号 1 ( タイムアウト 10 秒 ) を起動
     RET=RECEIVE(1)         …タイマ番号 1 のイベントとフレーム受信を待つ
     IF RET==0 THEN
        PRINT("TIMEOUT!!")  …タイムアウトの場合 "TIMEOUT" を表示
     ELSE
       PRINT("RECEIVE FRAME")  …タイムアウト以外の場合 "RECEIVE FRAME" を表示
     END
RETURN
```

A-3 本器での SAPI, TEI 管理 (レイヤ 2 自動実行モード) (D チャネル・シミュレーション)

(1) TEI 値の登録

① 登録

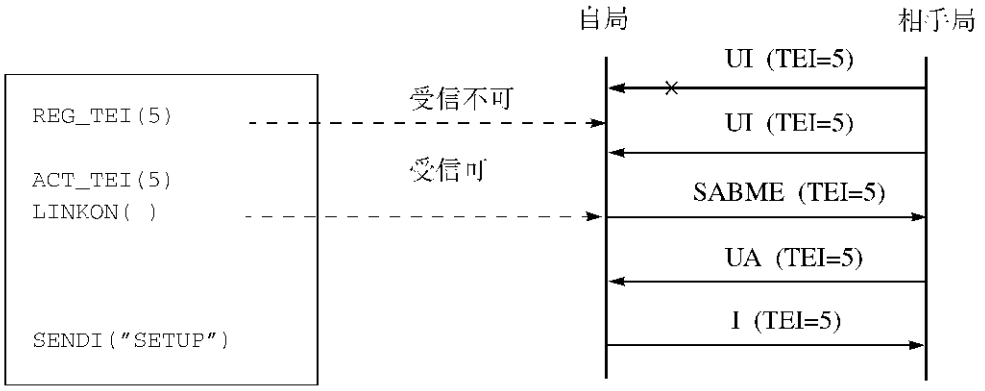
TEI 値を登録すると登録した TEI 値のフレームを受信することができるとともに、登録した TEI 値の UI コマンド、XID コマンド、XID レスポンスが送出可能になります。
ただし、TEI=127 は最初から登録されているので登録する必要はありません。
また、NT モードの場合、0 ～63 は最初から登録済です。



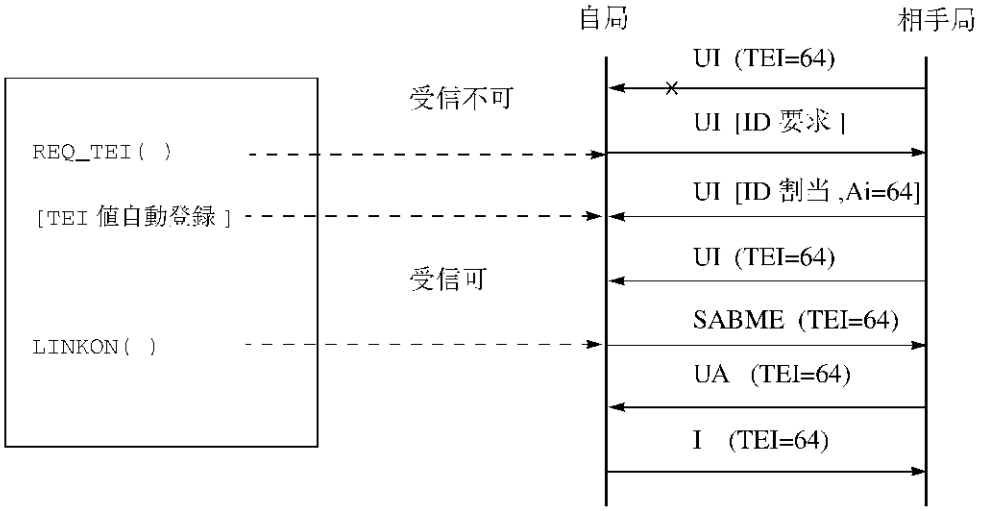
TEI の登録は、REG_TEI( ) 関数を用います。
REG_TEI( ) 関数による TEI 登録は、非自動割当端末をシミュレートするとき以外は使用しません。
TEI 登録は、TEI 割当手順により新たな TEI が割り当てられたときに自動的に行われます。

② 登録例

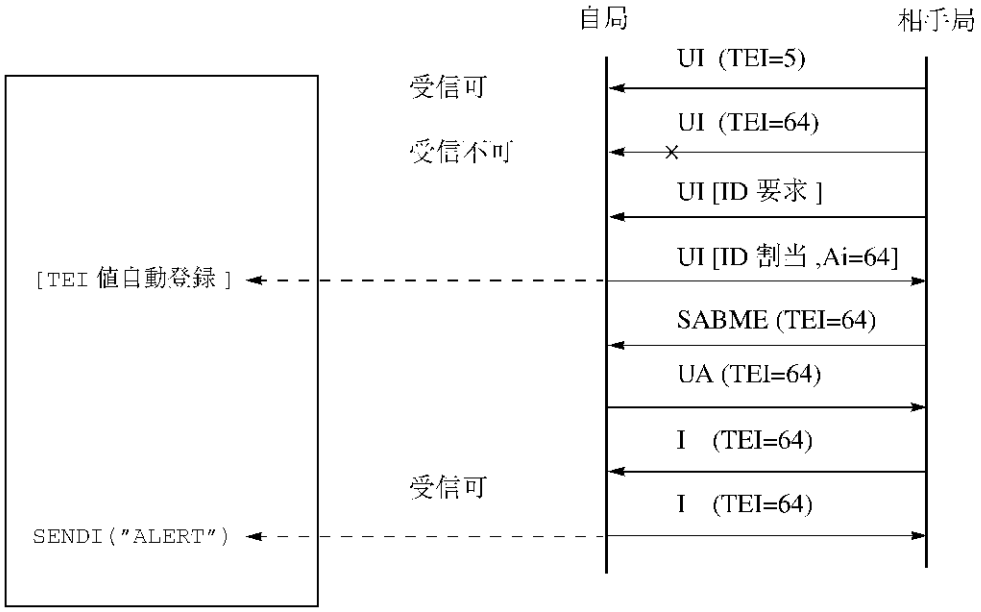
[例 1] 非自動割当端末のシミュレート



[例 2] 自動割当端末のシミュレート



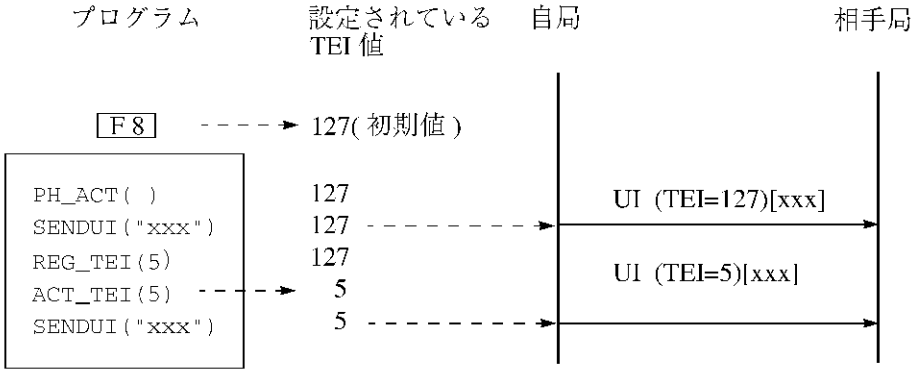
[例 3] 網側のシミュレート



シミュレーション・モードを宣言文により NT と指定すると TEI 値の 0 ～63 と 127 は登録済の状態になります。
これらの TEI 値をもった UI フレームは、シミュレーションが開始してレイヤ 1 が起動されるとすぐ受信可能となります。

③ 設定

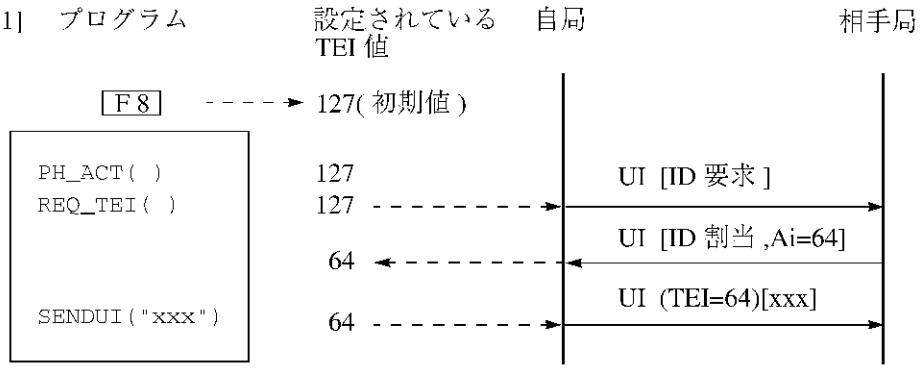
TEI 値を設定するとそのTEI 値を持つフレームが次回からの送信関数により送出されます。
TEI 値の設定は ACT_TEI( ) 関数を用いて行います。設定できる TEI 値は、登録されている TEI 値だけです。
非自動端末が TEI 値 5 でフレームを送出するプログラムの例を以下に示します。

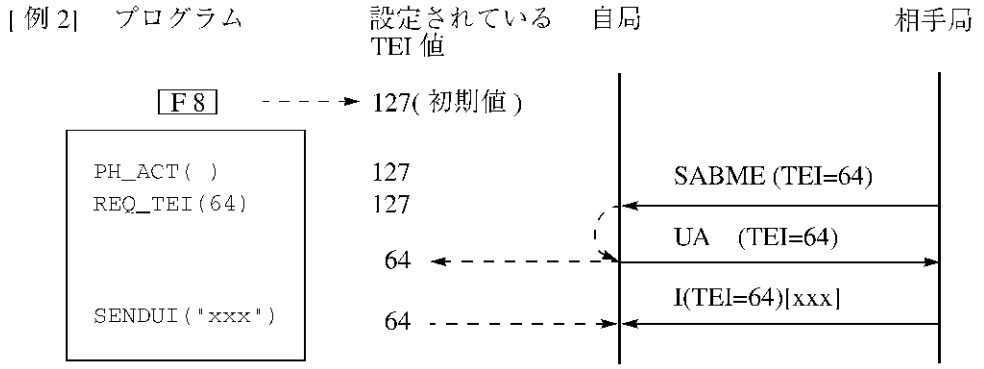


TEI 値は ACT_TEI( ) 関数で設定する以外に下記の条件のときに自動設定されます。

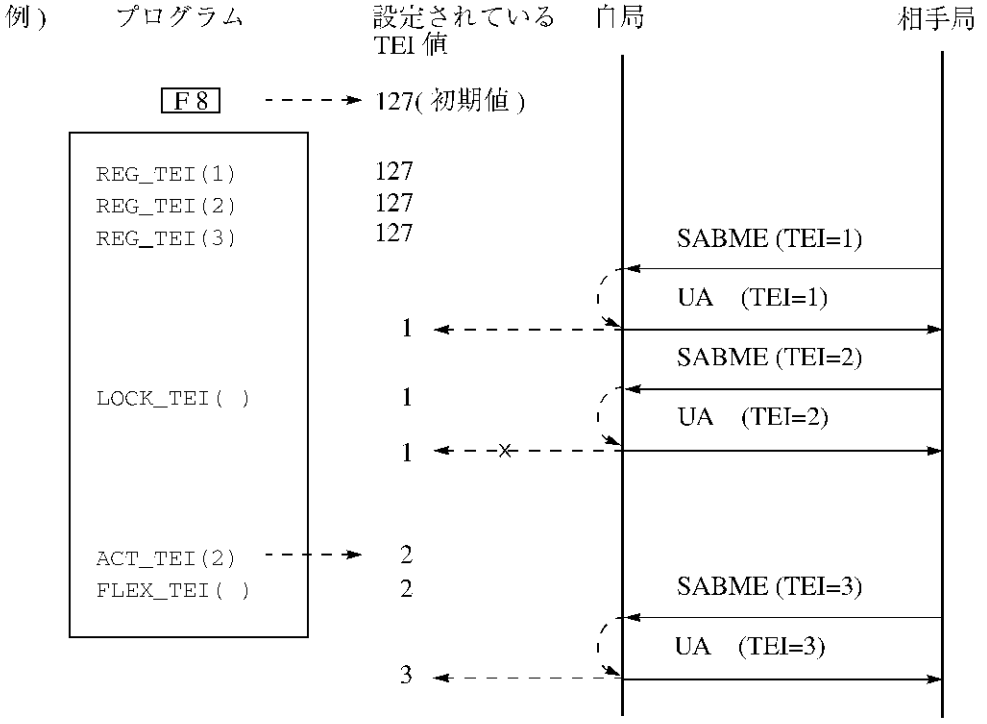
- REQ_TEI( ) 関数により、TEI 割当要求を起動し、網から TEI 値を割当てられた場合
- レイヤ 2 リンクの設定が起きた場合
  LINKON( ) 関数によりリンク設定をした場合、または
  相手局からリンクの設定をした場合

以下に TEI 値自動設定の例を示します。



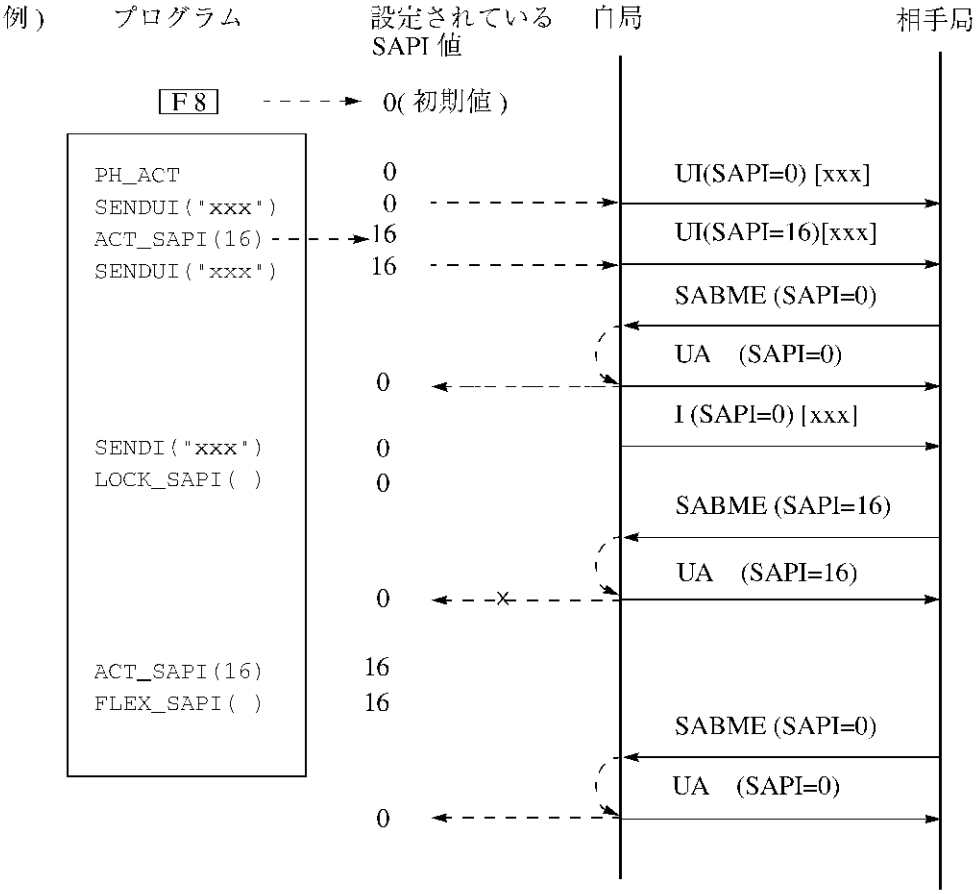


この自動設定を禁止したい場合は LOCK_TEI( ) 関数を使用します。これにより TEI 値変更不可状態になります。
これは TEI 値の自動設定を禁止するだけで、ACT_TEI( ) 関数による TEI 値の設定変更は禁止されていません。
この TEI 値変更不可状態は、FLEX_TEI( ) 関数により解除できます。



(2) SAPI 値の設定

SAPI 値には、0 、16、63 の 3 種類が登録されています。この 3 種類のどれかを設定することにより、その SAPI 値でフレームを送出することができます。
SAPI 値の設定には、ACT_SAPI 関数を用います。
また、レイヤ 2 リンクの設定が生じた場合には、その SAPI 値が新たに自動設定されます。この自動設定を禁止したい場合は、LOCK_SAPI( ) 関数を用います。これにより SAPI 値変更不可状態になります。しかし、これは相手局からのレイヤ 2 リンク設定による SAPI 値の自動設定を禁止するだけで ACT_SAPI( ) 関数による SAPI 値の変更は可能です。この SAPI 値変更不可状態は、FLEX_SAPI( ) 関数により解除することができます。



(3) リンク番号の使い方

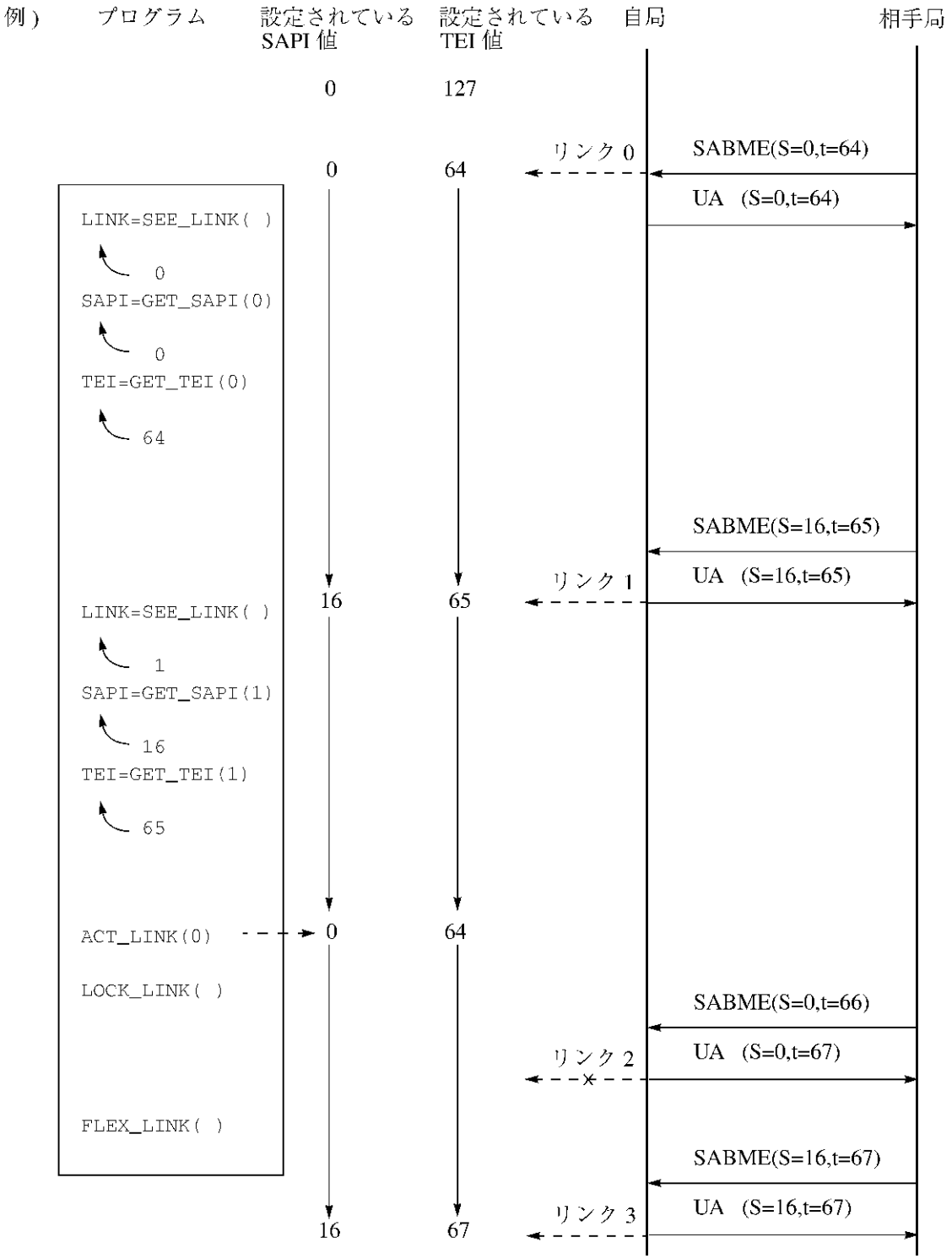
本器では最大 8 リンクまで同時に接続して、シミュレーションを行うことができます。このとき、それぞれのリンクは SAPI、TEI で管理します。この管理を行うために以下の (前述もしている) 関数を使用します。

REG_TEI
ACT_TEI
LOCK_TEI
FLEX_TEI
ACT_SAPI
LOCK_SAPI
FLEX_SAPI

本器では、リンクが張られるごとに内部的にリンク番号が付けられます。このリンク番号から SAPI 値、TEI 値を知ることもできます。
また、SAPI、TEI 値両方の自動設定変更を許可したり、禁止したりすることも可能です。
リンク番号を用いた関数の概要を以下に示します。

関数	概要
SEE_LINK( )	現在設定されているリンク番号を調べる。
GET_SAPI(LINK)	引数のリンク番号のリンクの SAPI 値を調べる。
GET_TEI(LINK)	引数のリンク番号のリンクの TEI 値を調べる。
GET_ LINK(SAPI,TEI)	引数 SAPI、TEI のリンク番号を調べる。
ACT_LINK(LINK)	引数のリンク番号のリンクに設定する。
LOCK_LINK( )	リンクの自動設定を禁止する。
FLEX_LINK( )	リンクの自動設定を許可する。

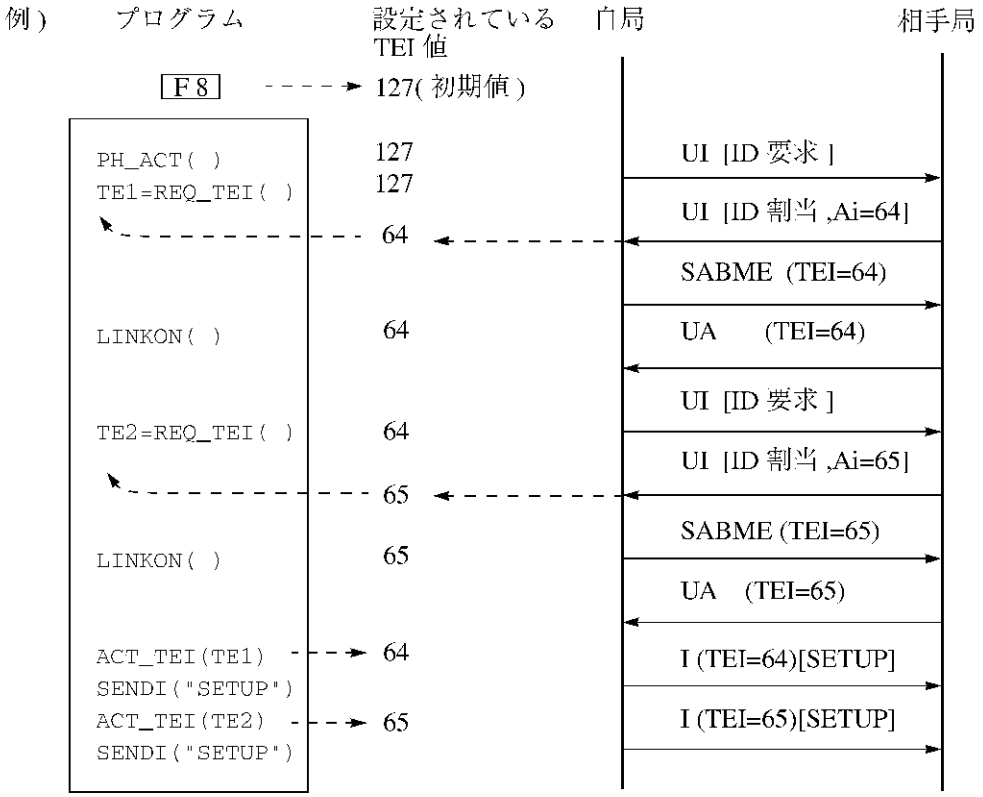
これらの関数を使用した例を以下に示します。


例)
プログラム
設定されている SAPI 値
設定されている TEI 値
自局
相手局
0
127
リンク 0
SABME(S=0,t=64)
0
64
UA (S=0,t=64)
LINK=SEE_LINK( )
0
SAPI=GET_SAPI(0)
0
TEI=GET_TEI(0)
64
SABME(S=16,t=65)
UA (S=16,t=65)
リンク 1
16
65
LINK=SEE_LINK( )
1
SAPI=GET_SAPI(1)
16
TEI=GET_TEI(1)
65
ACT_LINK(0)
0
64
LOCK_LINK( )
SABME(S=0,t=66)
UA (S=0,t=67)
リンク 2
FLEX_LINK( )
SABME(S=16,t=67)
UA (S=16,t=67)
リンク 3
16
67

(4) レイヤ 2 マルチリンクの管理例

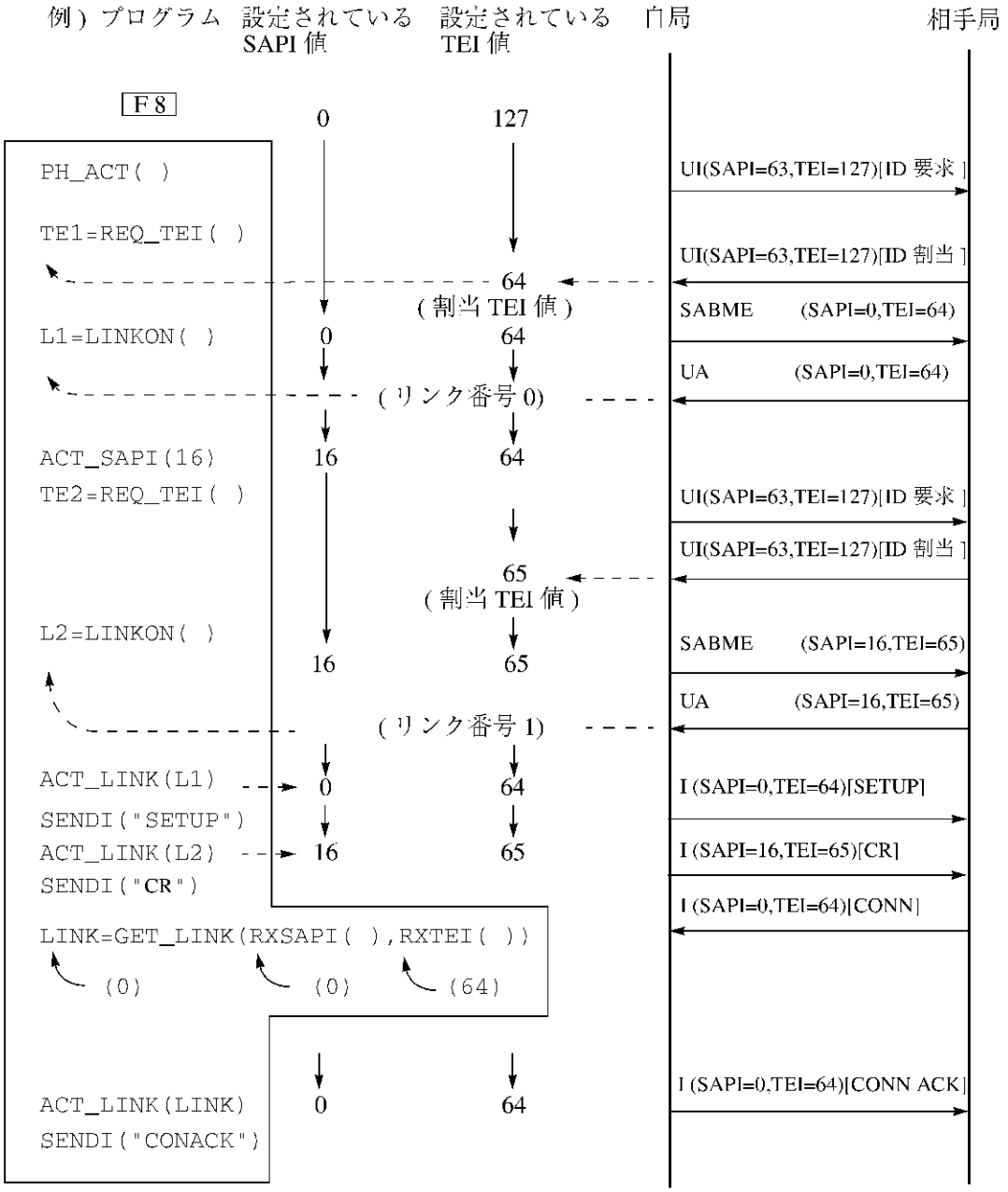
① TEI を用いたマルチリンクの設定

本器が端末 2 台分をシミュレートするプログラムについて示します。
ここでは網から TEI 値を割り当ててもらい、それぞれの TEI 値で SETUP フレームを送出するケースについて示します。



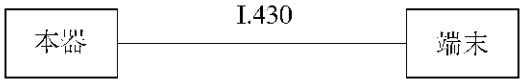
② リンク番号を用いたマルチリンクの設定

本器が端末 2 台分をシミュレートするプログラムについて示します。
①の例では、TEI 値でリンクを管理していますが、リンク番号で管理することもできます。リンクの設定は ACT_LINK(LINK) で行い、受信フレームからリンク番号を得るのは GET_LINK(SAPI, TEI) で行います。
以下にその例と内部的な動作について説明します。



(5) 網の着呼側シミュレーション (D チャネルの呼制御) 例

以下に示すプログラムは、次の環境で使われるのを前提としています。



プログラムを実行すると、本器は端末に対して呼設定を送ります。

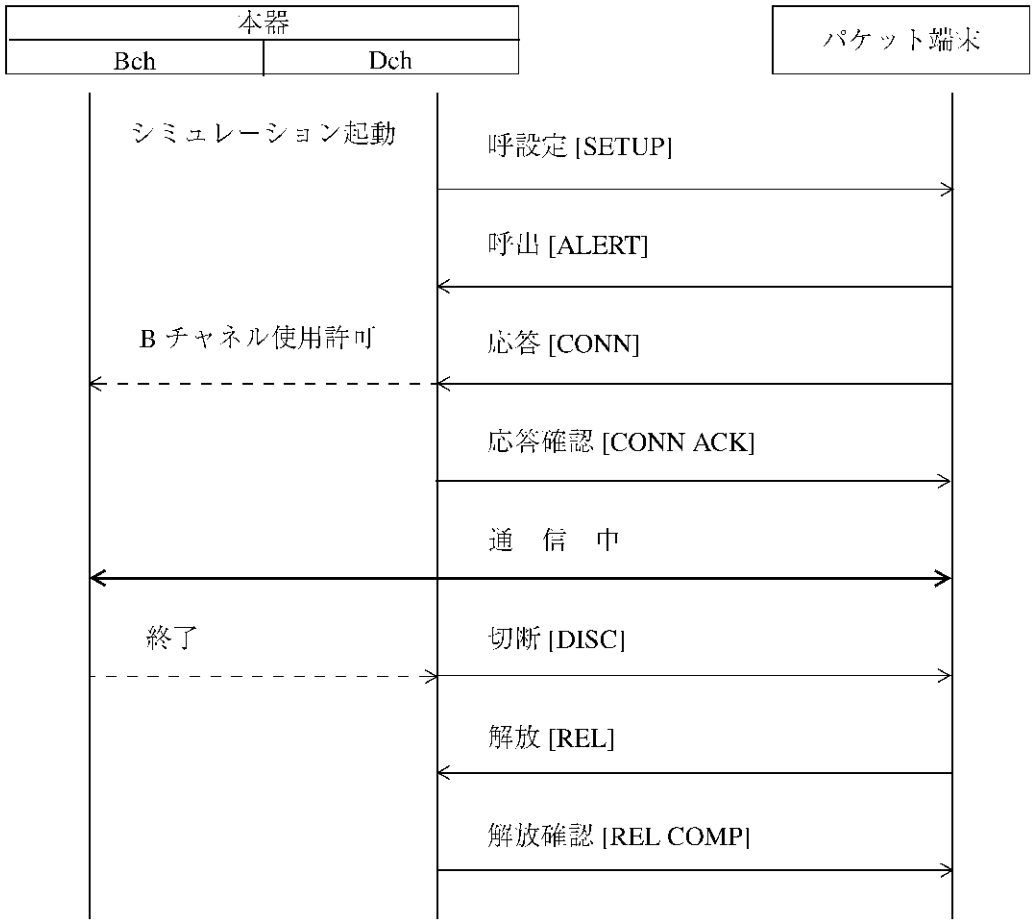
端末から応答メッセージを受信すると、D チャネル・シミュレータは、B チャネル・シミュレータに対してこのメッセージの受信と使用する B チャネルの番号を通知します。これにより、B チャネルが使用できるようになり、B チャネルを用いて端末とパケットのやりとりを行います。

B チャネル上でのパケットのやりとりが終了すると、B チャネル・シミュレータは、D チャネル・シミュレータに対して B チャネルシミュレーション終了のメッセージを送ります。

D チャネル・シミュレータはそれを受信すると切断手順に入ります。

図示すると (例) のようになります。

(例)



- サンプル・プログラム

注意　行番号はプログラム説明のため、便宜上使用しています。
　　実際のプログラムに行番号を付与すると、コンパイルが実行できません。

(1/2)

```
1 /*       ***  NTCOV.PRG  ***
2
3        This is a sample in Layer 2 auto execute mode.
4        This program simulates a network.
5        This instrument is trying to establish the link.
6        This program requires B channel simulator to run together.
7        According to your purpose, please use certain message and
8        program on B channel
9          For example
10               packet       D message     "NTPKT_B.MSG"
11                            B program     "DCECOV.PRG"
12                            B message     "DCHCOV.MSG"
13
14               Created by ADVANTEST  On Feb./13/1998
15 */
16
17 CHANNEL                D
18 INTERFACE              BRI
19 LAYER                  3
20 SIMMODE                NT
21 SPEED                  RVS
22
23 FUNC MAIN()
24
25      CALL_PROC = H'02'
26      ALERT     = H'01'
27      CONN      = H'07'
28      DISC      = H'45'
29      REL       = H'4D'
30      REL_COMP  = H'5A'
31
32      COMM_DCH  = 10
33      COMM_BCH  = 20
34
35      PRINTF("    NTCOV.PRG  ***¥N")
36      PRINT("*** TE             NT¥N")
37      PRINTF("                          ¥\")
38      PH_ACT()
39
40      SENDD ("SETUP")
41      PRINT("            <-  SETUP¥N")
42      CHANN_NUM = 1
43
44      WHILE(1)
45          RET = RECEIVE(0)
46          IF RET == 1 THEN
47              IF RXMSG() == CONN THEN
48                      PRINT("  CONN      ->¥N")
49                      SENDI("CONACK")
50                      PRINTF("              <  CONN ACK¥N")
```

(2/2)

```
51                      CH = RXCHAN_NUM()
52                      IF CH > 0 THEN CHANN_NUM = CH END
53                      MSG_CONN = CONN | L-SHIFT(CHANN_NUM, 8)
54                      SEND_EVENT(COMM_BCH, MSG_CONN)
55                      PRINT("  ***  CONNECTED B%) C-ANNEL  ***¥N", CHANN_NUM)
56              END
57
58              IF RXMSG() == CALL_PROC THEN
59                      PRINT("  CALL PROC ->¥N")
60                      C= = RXCHA\_\UV()
61                      IF CH > 0 THEN CHANN_NUM = CH END
62                      SEND_EVENT(COMM_BCH, CALL_PROC)
63              END
64
65              IF RXMSG() == ALERT THEN
66                      PRINT("  ALERT      ->¥N")
67                      CH = RXCHAN_NUM()
68                      IF CH > 0 THEN CHANN_NUM = CH END
69                      SEND_EVENT(COMM_BCH, ALERT)
70              END
71
72              IF RXMSG() == DISC THEN
73                      PRINT("  DISC       >¥N")
74                      SENDI("REL")
75                      PRINT("                <-  REL ¥N")
76                      SEND_EVENT(COMM_BCH, DISC)
77              END
78
79              IF RXMSG() == REL THEN
80                      PRINT("  REL        ->¥N")
81                      SENDI("RELCOM")
82                      PRINT("           <-  REL COMP¥N")
83                      SEND_EVENT(COMM_BCH, REL)
84                      EXIT
85              END
86
87              IF RXMSG() == REL_COMP THEN
88                      PRINT("  REL COMP   >¥N")
89                      SEND_EVENT(COMM_BCH, REL_COMP)
90                      EXIT
91              END
92          END
93
94          IF RET == COMM_BCH THEN
95              MSG = READ_EVENT()
96              IF MSG == DISC THEN
97                      SENDI("DISC")
98                      PRINT("           <- DISC¥N")
99              END
100         END
101     END
102
103     LINKOFF()
104 RETURN
```

- プログラム解説

(1/3)

行番号	解説
1 ~ 15	コメントです。 /* ~ */ で囲まれた領域は、実行しません。
17 ~ 21	宣言文です。(このプログラムは、D チャネル用) レイヤ 2 を自動実行して、網をシミュレートし、端末に対してリバース給電を供給することを宣言しています。
23 ~ 104	主関数です。 主関数 : FUNC MAIN() ~ RETURN の間に記述したプログラム
25 ~ 33	変数の初期化を行っています。H'' は 16 進表示であることを示します。
35 ~ 37	プリント文です。 シミュレーション画面に " " で囲まれた文字列を表示します。
38	レイヤ 1 を起動する関数です。
40	メッセージ・ビルダで作成した "SETUP" という名前のメッセージを UI フレームに乗せて送出します。
44 ~ 101	WHILE(1) ~ END で囲まれた部分で永久ループを作っています。 EXIT 文を実行するまで永久にこの部分を実行し続けます。
45	端末からのフレーム、または B チャネルからのメッセージを受信するまで待ちます。上記のイベントを受信するまでプログラム・カウンタはこの行を指したまま動きません。 上記のイベントを受信すると、RECEIVE 関数は以下に示す関数値を返します。 端末からのフレーム受信: 関数値 1 (RET=1) Bch からのメッセージ受信: 関数値 20(RET=20)
46 ~ 92	RECEIVE(0) で受信したイベントが端末からのフレームだった場合の処理です。
47 ~ 56	受信したフレームが、応答だった場合の処理です。 CONN ->とシミュレーション画面に表示した後、端末に応答確認 (CONACK) を I フレームとして送出します。そして B チャネル・シミュレータに応答フレームを受信したことと、使用する B チャネルの番号を通知します。
47	受信したフレームが応答フレームかどうか調べます。 RXMSG() は、受信したフレームのメッセージ種別のオクテットを読み出す関数です。この場合、RXMSG() の関数値が 7(受信したフレームのメッセージ種別のオクテットが 7) のときに 48 ~ 55 の行を実行します。

(2/3)

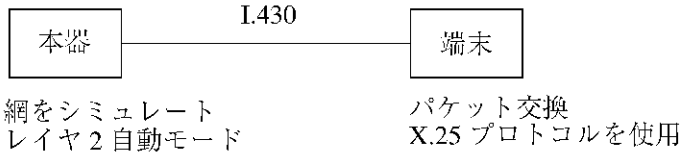
行番号	解説
49	メッセージ・ビルダで作成した "CONACK" という名前のメッセージを I フレームに乗せて送出します。
51 ~ 54	B チャネルに応答フレームを受信したことと、使用するチャネルを通知します。B チャネルの番号は、CHANN_NUM の値を使用します。チャネル番号は、呼設定で B1 チャネルを指示し、受信した呼設定受付、呼出または応答メッセージに含まれているチャネル番号を使用します。42、51、52、60、61、67、68 行がこれにあたります。 関数、演算子を使って B チャネルに以下のフォーマットのメッセージを送ります。 L_SHIFT : 左へ指定のビット数シフトする関数 | : ビット OR をとる演算子。 SEND_EVENT : 指定のチャネルにメッセージを送る関数。 15 8 7 0 チャネル番号 | メッセージ種別
58 ~ 63	受信したフレームが呼設定受付だった場合の処理です。 B チャネル・シミュレータに呼設定受付を受信したことを通知します
65 ~ 70	受信したフレームが呼出だった場合の処理です。B チャネル・シミュレータに呼出を受信したことを通知します
72 ~ 77	受信したフレームが切断だった場合の処理です。 解放 (REL) メッセージを I フレームに乗せ、端末に送出します。 B チャネル・シミュレータに切断を受信したことを通知します。
79 ~ 85	受信したフレームが解放だった場合の処理です。 解放完了 (RELCOM) メッセージを I フレームに乗せて端末に送出します。 B チャネル・シミュレータに解放を受信したことを通知します。 EXIT: WHILE ~ END で作るループを抜ける命令。(この場合、44 行 ~ 101 行のループを抜け、103 行を実行します。)
87 ~ 91	受信したフレームが解放完了だった場合の処理です。 B チャネル・シミュレータに解放完了を受信したことを通知します。
94 ~ 100	B チャネルからのメッセージを受信したときの処理です。 B チャネルからのメッセージ内容が切断要求の場合は、切断手順に入ります。

(3/3)

行番号	解説
95	B チャネルからのメッセージを読み出す関数です。 変数 MSG に B チャネルからのメッセージ内容を代入しています。 B チャネルからのメッセージは、RECEIVE(0) の関数値が 20 のときのみ READ_EVENT 関数で読むことができます。
96 ~ 99	B チャネルからのメッセージがDISC (16進の45) の場合は、端末に切断(DISC) メッセージをＩフレームに乗せて送出します。
103	レイヤ 2 を解放する関数です。

(6) 網の着呼側シミュレーション (B チャネルのパケットデータ送出) の例

以下に示すプログラムは、次の環境で使われることを前提としています。



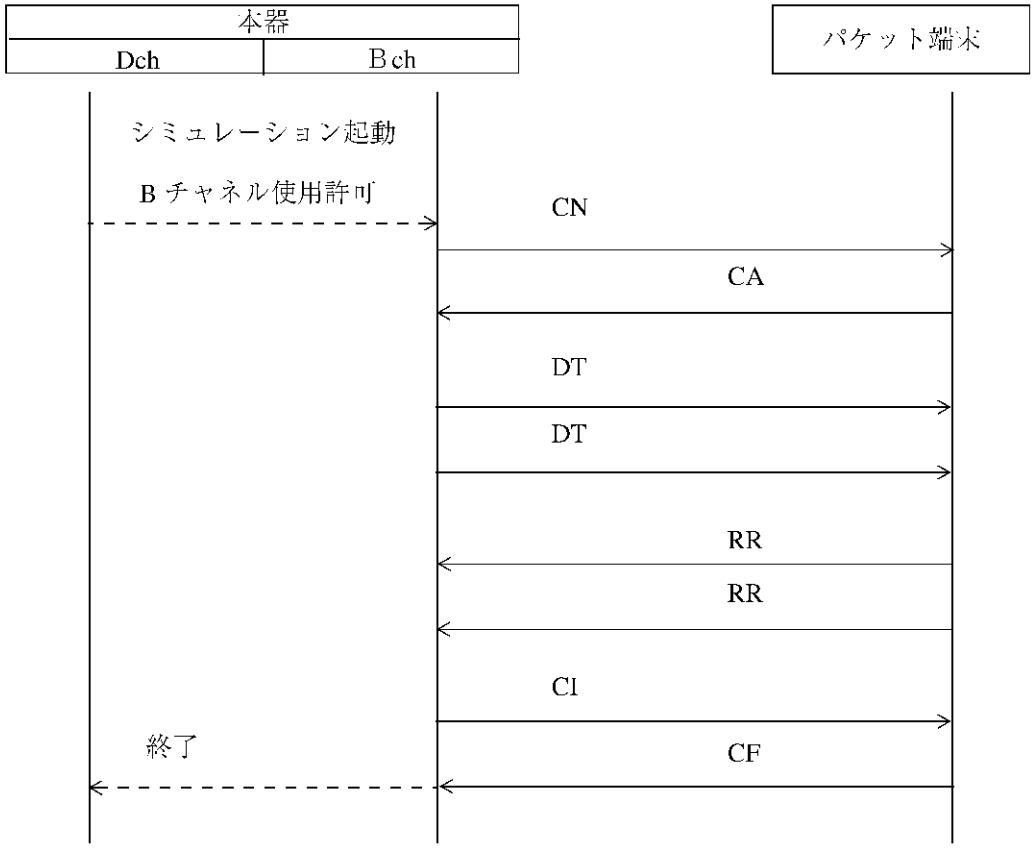
プログラムを実行すると、本器は D チャネルからの応答メッセージ待ち状態になります。

D チャネル・シミュレータが端末との間で呼の接続が終了すると、B チャネル・シミュレータに応答メッセージを送ります。これを受信すると、B チャネル・シミュレータは、端末に CN パケットを送ります。端末から CA パケットが返ってくると、端末に DT パケットを送出します。これから、5 秒後に端末に CI パケットを送出して切断手順に入ります。

端末から CF パケットを受信すると、D チャネルに切断メッセージを送り、シミュレーションを終了します。

図示すると (例) のようになります。

(例)



- サンプル・プログラム

注意　行番号はプログラム説明のため、便宜上使用しています。
　　実際のプログラムに行番号を付与すると、コンパイルが実行できません。

(1/2)

```
1 /*        ***  DCHCOM.PRG  ***
2
3         This is a sample in Layer 2 auto execute mode.
4         This instrument is trying to establish the link.
5         Please use the message "XCHCOV.MSG".
6         This program requires 2 channel simulator to run together.
7         As for D channel, use the program and the message as follows.
8                 Dch program      "\TCOV.PRG"
9                 Dch message      "\TPKT_3.MSG"
10                Created by ADVANTEST  On Feb./13/1998
11 */
12
13 C-ANNEL                  3
14  NTERFACE                BR
15 AYER                     3
16 SIMMODE                  NT
17 ADDRESS                  DCH
18 MODULO                   8
19
20
21 FUNC MAIN()
22
23       CALL_PROC - H'02'
24       ALERT     - H'01'
25       CONN      - H'07'
26       D_SC      - H'05'
27       REL       - H'0D'
28       REL_COMP  - H'5A'
29
30       CA        - H'04'
31       DT        - H'00'
32       CQ        - H'13'
33       CF        - H'17'
34       RR        - H'01'
35
36        CG        - 0
37        CN        - 3
38       TV_ID     - 1
39       TV_SHC    - 5
40
41       COMV_DCH  - 10
42       COMV_BCH  - 20
43
44       PRINTF("  TE          NT¥N")
45       PRINT(" ----------------¥N")
46
47       WHILE(1)
48               WHILE(1)
49                   RET - RECEIVE(TM_ID)
50                   IF RET == 0 THEN
51                       SENDPKT("CI", LCON, LCN)
52                       PRINT("          < CI¥N")
53                   END
54
```

(2/2)

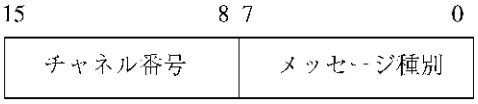
```
55                  IF RET == 1 THEN EXIT END
56
57                  IF RET ==  COMM_DCH THEN
58                      MSG_DCH = READ_EVENT()
59                      D_MSG = MSG_DCH & H'FF'
60                      D_CONT = R_SHIFT(MSG_DCH, 8)
61
62                      IF D_MSG == CON\ THEN
63                              SET CHANN(D_CONT)
64                              SETPS(LOGN, LCN, 0)
65                              SETPR(LOGN, LCN, 0)
66                              LINKON()
67                              SENDPKT("CN", LOGN, LC\)
68                              PRINT("          < CN¥N")
69                      END\D
70                  END
71              END
72
73              IF RXTYP() == CA THEN
74                      PRINT("  CA   >¥N")
75                      SENDPKT("DT1", LOGN, LCN)
76                      PRINT("          < DT¥N")
77                      INCPS(LOG\, LCN)
78                      SENDPKT("DT2", LOGN, LCN)
79                      PRINT("          < DT¥N")
80                      INCPS(LOGN, LCN)
81                      T_START(TM_ID, TM_SEC)
82              END\D
83
84              IF RXTYP() == DT THEN
85                      PRINT("   DT  > ¥N")
86                      INCPR(LOGN, LCN)
87                      SENDPKT("RR", LOGN, LCN)
88                      PRINT("          < RR¥N")
89              END
90
91              IF RXTYP() == RR THEN
92                      PRINT("   RR  > ¥N")
93              END
94
95              IF RXTYP() == CQ THEN
96                      PRINT("   CQ  > ¥N")
97                      SENDPKT("CF", LOGN, LCN)
98                      PRINT("          < CF¥N")
99                      EXIT
100             END\D
101
102             IF RXTYP() == CF THEN
103                     PRINT("   CF  >¥N")
104                     EXIT
105             END\D
106     END
107
108     WAIT(30)
109     LINKOFF()
110     SEND_EVENT(COMM_DCH, DISC)
111 RETURN
```

- プログラム解説

(1/4)

行番号	解説
1 ~ 11	コメントです。 /* ~ */ で囲まれた領域は、実行しません。
13 ~ 18	宣言文です。(このプログラムは、B チャネル用) レイヤ 2 を自動実行して、網をシミュレートし、自アドレスは DCE で、モジュロ 8 を使用することを宣言しています。
21 ~ 111	主関数です。 主関数 : FUNC MAIN() ~ RETURN の間に記述したプログラム。
23 ~ 42	変数の初期化を行っています。H'' は、16 進数表示であることを示します。
44 ~ 45	プリント文です。 シミュレーション画面に " " で囲まれた文字列を表示します。
47 ~ 106	WHILE(1) ~ END で囲まれた部分で永久ループを作っています。
48 ~ 71	EXIT 文を実行するまで永久にこの部分を実行し続けます。
49	端末からのフレーム、D チャネルからのメッセージまたはタイムアウト・イベントを受信するまで待ちます。上記のイベントを受信するまでプログラム・カウンタはこの行を指したまま動きません。上記のイベントを受信すると、RECEIVE 関数は以下に示す関数値を返します。 タイムアウト・イベント受信: 関数値 0 (RET=0) 端末からのフレーム受信 : 関数値 1 (RET=1) Dch からのメッセージ受信 : 関数値 10(RET=10)
50 ~ 53	RECEIVE (TM-ID) で受信したイベントがタイムアウト・イベントだった場合の処理です。81 行で起動したタイマがタイムアウトすると、この処理に入ります。
51	メッセージ・ビルダで作成した "CI" という名前のメッセージを I フレームに乗せて送出します。このとき論理チャネル・グループ番号 (LCGN) と論理チャネル番号 (LCN) は、引数で与えられた値を "CI" メッセージの該当のオクテットに上書きした後、送出します。
55	RECEIVE (TM-ID) で受信したイベントが端末からのフレーム受信だった場合の処理です。 EXIT 命令 : WHILE ~ END のループを強制的に抜ける関数。(この場合は 48 行 ~ 71 行のループを抜け、73 行を実行します。)

(2/4)

行番号	解説
57 ~ 70	RECEIVE (TM-ID) で受信したイベントが D チャネルからのメッセージだった場合の処理です。 指定の B チャネルを選択した後、レイヤ 2 を起動し、CN パケットを端末に送ります。
58	D チャネルからのメッセージを読み、その値を MSG_DCH 変数に代入しています。　READ_EVENT ： 他のチャネルからのイベントを読み出す関数。
59 ~ 60	D チャネルから送られてくるメッセージ種別とチャネル番号を読み出します。D チャネルから送られてくるメッセージは以下のフォーマットをしています。  D_MSG(変数)　：D チャネル から送られてきたメッセージ種別を代入する。 D_CONT (変数)　：D チャネル から送られてきたチャネル番号を代入する。 &　：ビット AND をする演算子。 R_SHIFT　：右へ指定のビット数シフトする関数。
62 ~ 69	D チャネルからのメッセージ種別が応答だった場合の処理です。使用する B チャネルを設定した後、送信順序番号 P(S) と受信順序番号 P(R) を初期化し、端末に CN パケットを送出します。
63	使用する B チャネルを D チャネルから送られてきたチャネル番号に設定します。 SET_CHANN 関数 ： 使用する B チャネルを指定する関数。
64 ~ 65	引数で指定した論理チャネル・グループ番号 (LCGN) と論理チャネル番号 (LCN) の送信順序番号 P(S) と受信順序番号 P(R) を初期化しています。 SETPS ： 送信順序番号 P(S) の値を設定する関数。 SETPR ： 受信順序番号 P(R) の値を設定する関数。
66	レイヤ 2 リンクを設定します。
67	メッセージ・ビルダで作成した "CN" という名前のメッセージを I フレームに乗せて送出します。このとき論理チャネル・グループ番号 (LCGN) と論理チャネル番号 (LCN) は、引数で与えられた値を "CN" メッセージの該当のオクテットに上書きした後、送出します。

(3/4)

行番号	解説
73 ~ 105	端末からパケットを受信したときの処理です。 端末からのパケット受信の情報は、49 行の RECEIVE (TM_ID) で受信します。そして、55 行の条件文で端末からパケットを受信したと判断すると、EXIT 命令で WHILE ループを抜け、これらの行が実行されます。
73 ~ 82	受信したパケットが CA パケットだった場合の処理です。 DT パケットを 2 つ送出し、タイマを起動します。このタイマは、ある一定時間経過した後、CI パケットにより切断するために使われます
75	メッセージ・ビルダで作成した "DT1" という名前のメッセージを I フレームに乗せて送出します。このとき論理チャネル・グループ番号 (LCGN) と論理チャネル番号 (LCN) は、引数で与えられた値を "DT1" メッセージの該当するオクテットに上書きした後、送出します。また、送信順序番号 P(S) と受信順序番号 P(R) も同様に、送出する前に上書きします。 送信順序番号 P(S) の値を +1 増加させます。
77	DT パケットを送出したので、送信順序番号 P(S) の値を +1 増加させています。
81	タイマを起動しています。 この場合、タイマ番号 TM_ID(1) でタイムアウト値 TM_SEC (5 秒) のタイマを起動しています。 このタイムアウト・イベントは 49 行の RECEIVE (TM_ID) で受信します。
84 ~ 89	受信したパケットが DT パケットだった場合の処理です。 受信順序番号 P(R) の値を +1 増加させ、RR パケットを送出します。
86	受信順序番号 P(R) の値を +1 増加させます。 DT パケットを受信したので、受信順序番号 P(R) の値を +1 します。
87	メッセージ・ビルダで作成した "RR" という名前のメッセージを I フレームに乗せて送出します。このとき論理チャネル・グループ番号 (LCGN) と論理チャネル番号 (LCN) は、引数で与えられた値を "RR" メッセージの該当するオクテットに上書きした後、送出します。また、受信順序番号 P(R) も同様に、送出する前に上書きします。
91 ~ 93	受信したパケットが RR パケットだった場合の処理です。
95 ~ 100	受信したパケットが CQ パケットだった場合の処理です。 CF パケットを送出し、47 行 ~ 106 行の WHILE ループを抜けます。

(4/4)

行番号	解説
102 ~ 105	受信したパケットが CF パケットだった場合の処理です。 47 行 ~ 106 行の WHILE ループを抜けます。
108	プログラムを指定の時間、中断する関数です。 この場合、3 秒プログラムを中断させた後、109 行の処理に入ります。
109	レイヤ 2 リンクを解放します。
110	D チャネルに、切断メッセージを送出します。

A-4 サンプル・プログラム

サンプル・プログラムは、本器 D ドライブの D5115¥SIM¥SAMPLE ディレクトリに収録されています。
メッセージ付きオブジェクト・プログラム (EXC ファイル) は、収録されていません。

(1) D チャネル・シミュレーション

(1/3)

メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER	概要
SAMPLE1.EXC	SAMPLE1.PRG	SAMPLE1.MSG	TE	3	発呼端末の正常シーケンスをシミュレート。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・ SETUP → ← CALL PROC ← ALERT ← CONN CONN ACK → ← DISC REL → ← REL COMP
SAMPLE2.EXC	SAMPLE2.PRG	SAMPLE2.MSG	TE	2	発呼端末の正常シーケンスをシミュレート。 SAMPLE1.EXC のトランスペアレント版。
TIMER1.EXC	TIMER1.PRG	—	—	—	10秒のタイマがタイムアウトするごとに、"TIMEOUT!!" と表示する。
TIMER2.EXC	TIMER2.PRG	—	—	—	10 秒、12 秒、14 秒のタイマが、それぞれタイムアウトするごとに、タイムアウトしたタイマの番号を表示する。
TIMER3.EXC	TIMER3.PRG	—	NT	2	10 秒のタイマがタイムアウトしたら、"TIMEOUT!!" と表示し、フレームを受信したら "RECEIVED FRAME" と表示する。
T303.EXC	T303.PRG	T303.MSG	TE	3	タイマ T303 を用いた例題プログラム。 (T303 : (「呼設定」送信時に使うタイマ)
T202.EXC	T202.PRG	T202.MSG	TE	2	タイマ T202 を用いた例題プログラム。 (T202 : TEI の ID 要求時に使うタイマ)
TECALL.EXC	TECALL.PRG	TECALL.MSG	TE	3	発呼端末の正常シーケンスをシミュレート。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・ SETUP → ← CALL PROC ← ALERT ← CONN CONN ACK → ← DISC REL → ← REL COMP SAMPLE1.EXC のシーケンス表示付版。

(2/3)

メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER	概要
TECALLED.EXC	TECALLED.PRG	TECALLED.MSG	TE	3	着呼端末の正常シーケンスをシミュレート。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・・ ← SETUP CALL PROC → ALERT → CONN → ← CONN ACK ← DISC REL → ← REL COMP
NTCALL.EXC	NTCALL.PRG	NTCALL.MSG	NT	3	着呼端末側の網をシミュレート。
NTCALLED.EXC	NTCALLED.PRG	NTCALLED.MSG	NT	3	発呼端末側の網をシミュレート。
SETUP1.EXC	SETUP1.PRG	SETUP1.MSG	TE	3	発呼端末の正常シーケンスをシミュレート。 以下のシーケンスを 30 秒ごとに繰り返す。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・・ SETUP → ← CALL PROC ← ALERT ← CONN CONN ACK → DISC → ← REL REL COMP →
SETUP2.EXC	SETUP2.PRG	SETUP2.MSG	TE	3	発呼を 2 回行う端末のシミュレート。
SETUP2D.EXC	SETUP2D.PRG	SETUP2D.MSG	TE	3	着呼端末の正常シーケンスをシミュレート。 NTCALLED.EXC と基本シーケンスは同じで、マルチリンクに対応させた。
DECODE.EXC	DECODE.PRG		NT	3	受信フレームを翻訳表示する。
PACK_D.EXC	PACK_D.PRG	PACK_D.MSG	TE	3	発呼端末の D チャネル・パケット正常シーケンスをシミュレート。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・・ SQ → ← SF CR → ← CC DT → DT → ← RR ← RR CQ → ← CF

(3/3)

メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER	概要
PACKD_D.EXC	PACKD_D.PRG	PACKD_D.MSG	TE	3	着呼端末のＤチャネル・パケット正常シーケンスをシミュレート。 端末　　網 ・・・・・・・・・・・・・・・・・・・・・・・・・・ ← SETUP CALL PROC → CONN →　　Q.931 手順 ← REL REL COMP → ← CN CA → DT → DT →　　X.25 手順 ← RR ← RR CQ → ← CF

(2) Bチャネル・シミュレーション

メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER	概要
SAMPLE1.EXC	SAMPLE1.PRG	SAMPLE1.MSG	TE	3	発呼端末の正常シーケンスをシミュレート。 端末　網 CR → ← CC CQ → ← CF
TIMER1.EXC	TIMER1.PRG	—	—	—	10秒のタイマがタイムアウトするごとに、"TIMEOUT!!" と表示する。
TIMER2.EXC	TIMER2.PRG	—	—	—	10秒、12秒、14秒のタイマが、それぞれタイムアウトするごとに、タイムアウトしたタイマの番号を表示する。
TIMER3.EXC	TIMER3.PRG	—	NT	2	10秒のタイマがタイムアウトしたら、"TIMEOUT!!" と表示し、フレームを受信したら "RECEIVED FRAME" と表示する。
T1.EXC	T1.PRG	T1.MSG	NT	2	タイマT1を用いた例題プログラム。 (T1:「SABM」送信時に使うタイマ)
T21.EXC	T21.PRG	T21.MSG	TE	3	タイマT21を用いた例題プログラム。 (T21:「CR」パケット送信時に使うタイマ)
TECALL.EXC	TECALL.PRG	TECALL.MSG	TE	3	発呼端末の正常シーケンスをシミュレート。 端末　網 CR → ← CC DT ← DT → ← RR ← RR CQ → ← CF
TECALLED.EXC	TECALLED.PRG	TECALLED.MSG	TE	3	着呼端末の正常シーケンスをシミュレート。 端末　網 ← CN CA → ← DT RR → ← DT RR → ← CI CF →
NTCALL.EXC	NTCALL.PRG	NTCALL.MSG	NT	3	着呼端末側の網をシミュレート。
NTCALLED.EXC	NTCALLED.PRG	NTCALLED.MSG	NT	3	発呼端末側の網をシミュレート。
DECX25.EXC	DECX25.PRG	—	NT	3	受信フレームを翻訳表示する。

Dチャネルと Bチャネルの同時シミュレーション

① 音声シミュレーション (発呼側の電話)

Dチャネルで呼制御を行い、Bチャネルで音声シミュレーションを行います。
Dチャネルは Bチャネルと通信を行い、呼の情報を Bチャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、その状況にあった音声シミュレーションを行います。
以下の組み合わせは、発呼側の電話をシミュレートします。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	PHONE.EXC	TECOM.PRG	PHONE.MSG	TE	3
B	PHONE.EXC	PHONE.PRG	—	—	—

② 音声シミュレーション (着呼側の電話)

Dチャネルで呼制御を行い、Bチャネルで音声シミュレーションを行います。
Dチャネルは Bチャネルと通信を行い、呼の情報を Bチャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、その状況にあった音声シミュレーションを行います。
以下の組み合わせは、着呼側の電話をシミュレートします。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	PHONED.EXC	TECOMD.PRG	PHONED.MSG	TE	3
B	PHONED.EXC	PHONED.PRG	—	—	—

③ パケット・シミュレーション (発呼側の端末)

Dチャネルで呼制御を行い、Bチャネルでパケット・シミュレーションを行います。
Dチャネルは Bチャネルと通信を行い、呼の情報を Bチャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、指定のチャネルで、パケットのシミュレーションを行います。
以下の組み合わせは、発呼側のパケット端末をシミュレートします。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	PACK_B.EXC	TECOM.PRG	PACK_B.MSG	TE	3
B	DTECOM.EXC	DTECOM.PRG	DTECOM.MSG	TE	3

④ パケット・シミュレーション (着呼側の端末)

D チャネルで呼制御を行い、B チャネルでパケット・シミュレーションを行います。
D チャネルは B チャネルと通信を行い、呼の情報を B チャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、指定のチャネルで、パケットのシミュレーションを行います。
以下の組み合わせは、着呼側のパケット端末をシミュレートします。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	PACKD_B.EXC	TECOMD.PRG	PACKD_B.MSG	TE	3
B	DTECOMD.EXC	DTECOMD.PRG	DTECOMD.MSG	TE	3

⑤ パケット・シミュレーション (着呼側の網)

D チャネルで呼制御を行い、B チャネルでパケット・シミュレーションを行います。
D チャネルは B チャネルと通信を行い、呼の情報を B チャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、指定のチャネルで、パケットのシミュレーションを行います。
以下の組み合わせは、着呼側の網をシミュレートします。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	NTCOM.EXC	NTCOM.PRG	NTPKT_B.MSG	NT	3
B	DCECOM.EXC	DCECOM.PRG	DCECOM.MSG	NT	3

⑥ パケット・シミュレーション (発呼側の網)

D チャネルで呼制御を行い、B チャネルでパケット・シミュレーションを行います。
D チャネルは B チャネルと通信を行い、呼の情報を B チャネルに知らせます。
B チャネルは、D チャネルからの情報をもとに、指定のチャネルで、パケットのシミュレーションを行います。
以下の組み合わせは、発呼側の網をシミュレートします。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	NTCOMD.EXC	NTCOMD.PRG	NTCOMD.MSG	NT	3
B	DCECOMD.EXC	DCECOMD.PRG	DCECOMD.MSG	NT	3

⑦ ビット・エラー試験

Dチャネルで呼制御を行い、Bチャネルでビット・エラー試験を行います。
DチャネルはBチャネルと通信を行い、呼の情報をBチャネルに知らせます。
Bチャネルは、Dチャネルからの情報をもとに、指定のチャネルで、ビット・エラー試験を行います。
ビット・エラー試験を行う前に、Bチャネルのシミュレータ画面で、試験に使うパターンと試験するビット数を入力します。試験が終了すると、結果を画面に表示します。
この試験を行うためには、網あるいは着端末側で指定の Bチャネルをループ・バックする必要があります。
これらのプログラムを実際に利用する場合、メッセージ・ビルダ中の電話番号を変更する必要があります。

チャネル	メッセージ付オブジェクト	ソース・プログラム	メッセージ	SIMMODE	LAYER
D	BERT.EXC	TECOM.PRG	BERT.MSG	TE	3
B	BERT.EXC	BERT.PRG	—	—	—

関数索引

ここでは、各関数に対応する本書でのページを記載しています。
(Dch)、(Bch) の表記は、それぞれ D チャネル・シミュレーション、B チャネル・シミュレーション用に使用する関数であることを示します。

[S]

[T]

[V]

[W]

本製品に含まれるソフトウェアのご使用について

本製品に含まれるソフトウェア（以下本ソフトウェア）のご使用について以下のことにご注意下さい。
ここでいうソフトウェアには、本製品に含まれる又は共に使用されるコンピュータ・プログラム、将来弊社よりお客様に提供されることのある追加、変更、修正プログラムおよびアップデート版のコンピュータ・プログラム、ならびに本製品に関する取扱説明書等の付随資料を含みます。

使用許諾

本ソフトウェアの著作権を含む一切の権利は弊社に帰属いたします。
弊社は、本ソフトウェアを本製品上または本製品とともに使用する限りにおいて、お客様に使用を許諾するものといたします。

禁止事項

お客様は、本ソフトウェアのご使用に際し以下の事項は行わないで下さい。

- 本製品使用目的以外で使用すること
- 許可なく複製、修正、改変を行うこと
- リバース・エンジニアリング、逆コンパイル、逆アセンブルなどを行うこと

免　　責

お客様が、本製品を通常の用法以外の用法で使用したことにより本製品に不具合が発生した場合、およびお客様と第三者との間で著作権等に関する紛争が発生した場合、弊社は一切の責任を負いかねますのでご了承下さい。

保証について

製品の保証期間は、お客様と別段の取り決めがある場合または当社が特に指定した場合を除き、製品の納入日（システム機器については検収日）から1年間といたします。保証期間中に、当社の責めに帰する製造上の欠陥により製品が故障した場合、無償で修理いたします。ただし、下記に該当する場合は、保証期間中であっても保証の対象から除外させていただきます。

- 当社が認めていない改造または修理を行った場合
- 支給品等当社指定品以外の部品を使用した場合
- 取扱説明書に記載する使用条件を超えて製品を使用した場合（定められた許容範囲を超える物理的ストレスまたは電流電圧がかかった場合など）
- 通常想定される使用環境以外で製品を使用した場合（腐食性の強いガス、塵埃の多い環境等による電気回路の腐食、部品の劣化が早められた場合など）
- 取扱説明書または各種製品マニュアルの指示事項に従わずに使用された場合
- 不注意または不当な取扱により不具合が生じた場合
- お客様のご指示に起因する場合
- 消耗品や消耗材料に基づく場合
- 火災、天変地異等の不可抗力による場合
- 日本国外に持出された場合
- 製品を使用できなかったことによる損失および逸失利益

当社の製品の保証は、本取扱説明書に記載する内容に限られるものとします。

保守に関するお問い合わせについて

長期間にわたる信頼性の保証、国家標準とのトレーサビリティを実現するためにアドバンテストでは、工場から出荷された製品の保守に対し、カスタマ・エンジニアを配置しています。
カスタマ・エンジニアは、故障などの不慮の事故は元より、製品の長期間にわたる性能の保証活動にフィールド・エンジニアとしても活動しています。
万一、動作不良などの故障が発生した場合には、当社のMS（計測器）コールセンタにご連絡下さい。

製品修理サービス

- **製品修理期間**
  製品の修理サービス期間は、製品の納入後10年間とさせていただきます。
- **製品修理活動**
  当社の製品に故障が発生した場合、当社に送っていただく引取り修理、または当社技術員が現地に出張しての出張修理にて対応いたします。

製品校正サービス

- **校正サービス**
  ご使用中の製品に対し、品質および信頼性の維持を図ることを目的に行うもので、校正後の製品には校正ラベルを貼付けし、品質を保証いたします。
- **校正サービス活動**
  校正サービス活動は、株式会社アドバンテスト カスタマサポートに送っていただく引取り校正、または当社技術員が現地に出張しての出張校正にて対応いたします。

予防保守のおすすめ

製品にはエレクトロニクス部品およびメカニカル部品の一部に寿命を考慮すべき部品を使用しているため、定期的な交換を必要とします。適正な交換期間を過ぎて使用し発生した障害に対しては、修理および性能の保証ができない場合があります。
アドバンテストでは、このようなトラブルを未然に防ぐため、予防保守が有効な手段と考え、予防保守作業を実施する体制を整えています。
各種の予防保守を定期的に実施することで、製品の安定稼働を図り、不意の費用発生を防ぐため、年間保守契約による予防保守の実施をお薦めいたします。
なお、年間保守契約は、製品、使用状況および使用環境により内容が変わりますので、最寄りの弊社営業支店にお問い合わせ下さい。

ADVANTEST

http://www.advantest.co.jp

株式会社アドバンテスト

本社事務所
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 03-3214-7500（代）

第4アカウント販売部（東日本）
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 0120-988-971
FAX: 0120-988-973

第4アカウント販売部（西日本）
〒564-0062 吹田市垂水町3-34-1
TEL: 0120-638-557
FAX: 0120-638-568

★計測器に関するお問い合わせ先
（製品の仕様，取扱い，修理・校正等計測器関連全般）
MS（計測器）コールセンタ
TEL 0120-919-570
FAX 0120-057-508
E-mail : icc@acs.advantest.co.jp