資 料

パーソナルコンピュータ用回線制御言語(NCL)

取扱い説明書

昭 和 60 年 3 月

JIPDEC

財団法人　日本情報処理開発協会

財団法人 図書 1.11.6 日本情報処理開発協会

この資料は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて、昭和５９年度に実施した「マイクロコンピュータの利用に関する共通的な技術開発」の一環としてとりまとめたものであります。

目　　次

1. 概　要

当マニュアルは「パーソナルコンピュータ用回線制御言語」（以後ＮＣＬと呼びます）の、文法やコンパイル手法、使用方法等をまとめたものです。

1.1　開発の目的

現在パソコンを導入している企業や将来パソコンを導入したいと考えている企業の中で「パソコンとホストコンピュータを接続して使用したい」とか「パソコン相互間で通信をさせたい」といったニーズは、今後さらに多くなるものと考えられる。

しかしながら、メーカーの異なる機器を接続し通信する場合に必ずプロトコルの相違が問題になり、このことがパソコンの利用促進に大きな障害をあたえている。このような問題を解決し、パソコンの利用を大きく促進することを目的として、パソコン用通信ソフトを作るためのツール「パーソナルコンピュータ用回線制御言語」を開発した。

1.2　ＮＣＬの主な機能

① 当ソフトウェアは、パーソナルコンピュータ用通信ソフトを作成する為のプリコンパイラーであり、文法に従って記述したプロトコルに基づいて、通信ソフト用アセンブラーソースプログラムを作成する。

② 当ソフトウェアでは、既存のプロトコルをはじめ、任意のプロトコルが定義できる。

③ 当ソフトウェアで定義できるプロトコルは、ＩＳＯプロトコル階層で言う「データリンクレベル」の範囲の基本形データ伝送手順をサポートする。

1.3　ＮＣＬの適用機種及び構成

① ハードウェア

・日本電気製　　ＰＣ－９８０１

・東芝製　　ＰＡＳＯＰＩＡ１６、ＰＡＳＯＰＩＡ１６００

② オペレーティング　システム

・ＭＳ－ＤＯＳ（Ver. 2.0以上）

③ ＮＣＬのコンパイル環境

ＣＰＵ	インテル製８０８６及び相当品
メモリ	ＮＣＬコンパイル時８０ＫＢ以上のユーザーメモリーが必要
フロッピーデスク	５インチ又は８インチ１ドライブ
ディスプレイ	
キーボード	

④ システム　ユーティリティ　プログラム

・エディタ　　ＥＤＬＩＮ．ＣＯＭ又は相当品

・アセンブラ　　ＭＡＳＭ．ＥＸＥ

・リンカー　　ＬＩＮＫ．ＥＸＥ

・バイナリー変換　　ＥＸＥ２ＢＩＮ．ＥＸＥ

が必要です。

2. ＮＣＬコンパイルの方法

当コンパイラーはＮＣＬ１、ＮＣＬ２、ＮＣＬ３の３つのパスから構成されており、ユーザーが作成したＮＣＬソースプログラムより、最終的にＭＳ－ＤＯＳ用アセンブラソースプログラムを作成する。

2.1　ＮＣＬ１

構文

ＮＣＬ１　ＮＣＬソースファイル名　〔－Ｉ〕

説明

ＮＣＬ１では構文解析、チェックを行なう。

入力するＮＣＬソースファイル名の名前のつけ方はＭＳ－ＤＯＳに従う。

ただし、ファイル名の拡張子は「．ＮＣＬ」でなければならない。キー入力時には、この拡張子は省略できる。

オプションとして「－Ｉ」が指定できる。これは、コンパイルリストの作成を指示するものである。

コンパイルリストはＮＣＬ２実行時に作成される。

オプションを指定しない場合、コンパイルエラーが発生すると、行番号とエラーメッセージがCONSOLEに出力される。

2.2　ＮＣＬ２

構文

ＮＣＬ２　ＮＣＬソースファイル名

説明

ＮＣＬ２では定数のチェック、DATA　LINK　SECTION内のラベルのチェック等を行なう。

キー入力時のソースファイル名はＮＣＬ１と同じである。拡張子は省略でき

るが、あえて入力する場合は「．％Ｗ」である。（「．ＮＣＬ」ではない。）
ＮＣＬ１で「－ｌ」オプションを指定した場合、このＰＡＳＳでコンパイルリストがディスクに作られる。リストファイルのファイル名はソースファイル名と同じである。ただし、拡張子は「．％Ｐ」である。
「－ｌ」オプションを指定しない場合、エラーが発生すると、ＣＯＮＳＯＬＥに行番号とエラーメッセージが出力される。

2.3　ＮＣＬ３

構文

ＮＣＬ３　ＮＣＬソースファイル名

説明

ＮＣＬ３では、ＭＳ－ＤＯＳ用８０８６アセンブラソースファイルを作成する。
キー入力時のソースファイル名はＮＣＬ１と同じである。
拡張子を入力する場合は「．％Ｏ」である。
ＮＣＬ１、ＮＣＬ２でエラーが発生した場合、このＮＣＬ３は実行できない。
ＮＣＬ３の実行で、生成されたアセンブラソースファイルのファイル名はＮＣＬソースファイル名と同じである。ただし拡張子は「．ＡＳＭ」である。

2.4　バッチファル

構文

ＮＣＬ　ＮＣＬソースファイル名　〔－ｌ〕

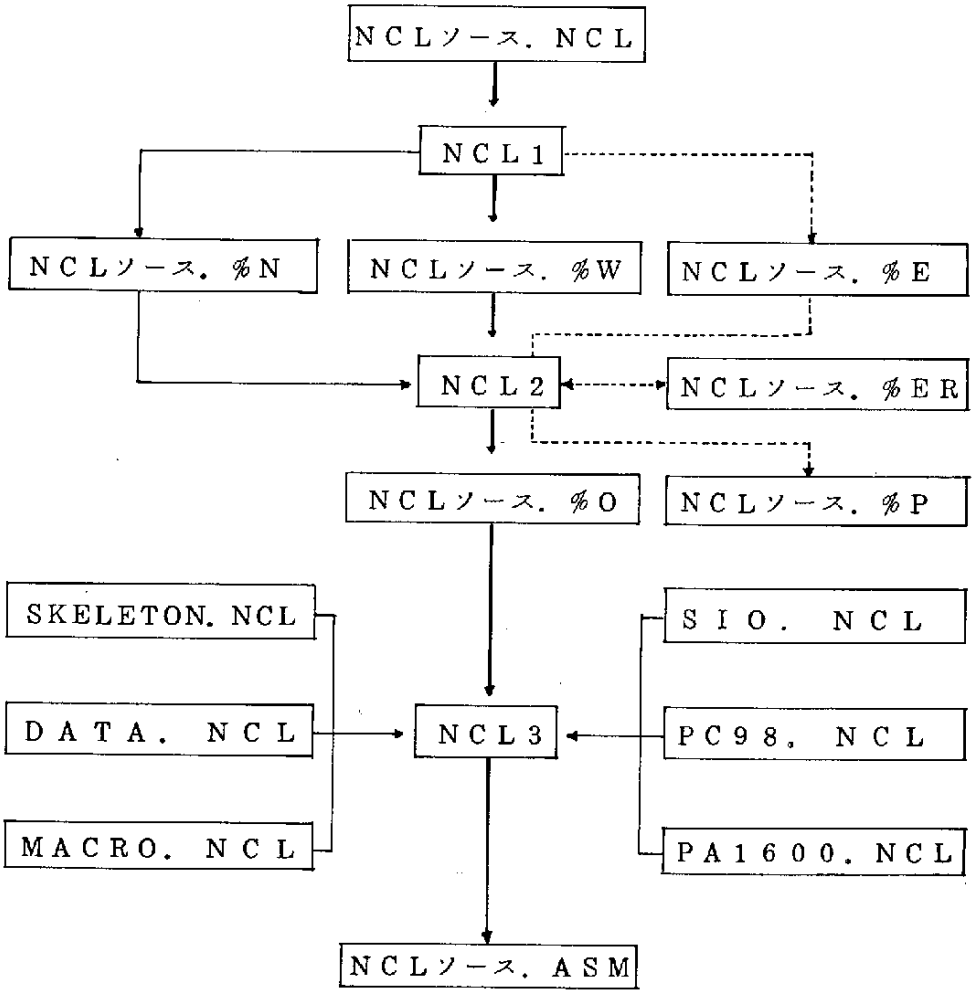
説明

ＮＣＬ１からＮＣＬ３までを連続して実行するバッチファイル「ＮＣＬ．ＢＡＴ」が使用できる。
ＮＣＬ１又はＮＣＬ２でエラーが発生すると、ＮＣＬ３は実行しない。
オプションの「－ｌ」はＮＣＬ１と同じである。

又、ユーザーがこのバッチファイルを修正することも可能である。

2.5 ＮＣＬコンパイルフローチャート

拡張子が「．％－」のファイルは中間ファイルでコンパイル後、自動的に消される。ただし、「．％Ｐ」のファイルだけは消されない。



3. 実行形式の回線制御プログラムの作成

ＮＣＬで作成された回線制御プログラムは、ＣＯＭ型式で実行するように作られている。以下に、アセンブルからＣＯＭ型式のプログラムの作成方法を説明する。

3.1 アセンブル（ＭＡＳＭ）

ＮＣＬコンパイラーの出力ファイル「ＮＣＬソース．ＡＳＭ」をＭＡＳＭでアセンブルする。

この時必要であれば、アセンブルリストの指定、アセンブラ出力ファイル名の指定等が可能である。

アセンブラの出力ファイル名の拡張子は、「．ＯＢＪ」である。

3.2 リンケージ（ＬＩＮＫ）

アセンブル結果のファイルをＬＩＮＫにより実行可能型式のプログラムにする。

ただし、リンケージをとるライブラリーは何も必要ない。

リンケージの出力ファイル名の拡張子は「．ＥＸＥ」である。

3.3 バイナリーフォーマットへの変換（ＥＸＥ２ＢＩＮ）

リンケージ結果の実行可能プログラムをＥＸＥ２ＢＩＮによりＣＯＭ型式のプログラムに変換する。

この時出力ファイル名の拡張子は「．ＣＯＭ」と指定しなければならない。

回線制御プログラムはＣＯＭ型式で実行するように作られている。

ＥＸＥ（実行可能）型式のファイルを実行させることはできない。

4. 回線制御プログラムの実行方法

ＮＣＬにより作成された回線制御プログラムは、メモリに常駐させてユーザープログラムのバックグラウンドで実行する。

そのため、ユーザープログラムを実行する前に、回線制御プログラムをメモリーにロードしなければならない。まず、ＣＯＭ型式の回線制御プログラム名を実行させると、メモリーに常駐して、ＭＳ－ＤＯＳはコマンド入力待ちになる。

この状態では、回線制御プログラムはメモリー上にロードされたままで、動作は何も行なわない。

次にユーザープログラムを実行させ、この内で「回線のオープン」命令を出すと、回線制御プログラムが動作を始める。この後、データの送受信が可能となります。

ユーザープログラムの実行をやめる場合は、「回線のクローズ」命令を出す。この命令により、回線制御プログラムはメモリー上に常駐したまま、動作を停止します。

5. 回線制御プログラムのコントロール方法

5.1 コントロールコマンドの種類と機能

ユーザープログラムから回線制御プログラムを制御するには下記の様に5種類のコマンドがある。

① 回線のオープン

回線制御プログラム内の変数、アドレスの初期化、シリアルポート、タイマー等の初期化を行ない、回線制御プログラムを使用可能な状態にする。

② 回線のクローズ

シリアルポート、タイマーをリセットし、回線制御プログラムを使用不可の状態にする。

③ データの受信

ユーザープログラムがＲＥＡＤＹ状態であり、データの受信が可能であることを回線制御プログラムに知らせ、データの受信を行なう。

④ データの送信

ユーザープログラム内にある送信バッファのデータを送信する。

⑤ 送受信の中断

送受信中にブレイクキー（CONTROL−C）を押すことにより、送受信を中断する。

5.2 コントロールコマンド、パラメータのフォーマット及びコントロール方法

ユーザープログラムと回線制御プログラムの間のパラメータのやりとりは、すべてレジスタを介して行なう。

① ユーザープログラムから回線制御プログラムへパラメータを渡す。

ＡＨレジスタ：各機能に対する1バイトのコマンド

ＤＳレジスタ：パラメータエリアのセグメントアドレス

ＤＸレジスタ：パラメータエリアのオフセットアドレス

ＳＳ、ＳＰレジスタ：スタックエリアのセグメントアドレス、スタックポインタ

② 回線制御プログラムからユーザープログラムへパラメータを返す。

ＡＨレジスタ：リターンコード

ＡＬレジスタ：エラーコード

ＤＳレジスタ：パラメータエリアのセグメントアドレス

ＤＸレジスタ：パラメータエリアのオフセットアドレス

ＳＳ、ＳＰレジスタ：スタックエリアのセグメントアドレス、スタックポインタ

③ パラメータエリアのフォーマット

ＰＡＲＡ０	……	ＰＡＲＡ７	送受信バッファアドレス	
			オフセット	セグメント
１バイト		１バイト	２バイト	２バイト

上記のフォーマットでＰＡＲＡ０～ＰＡＲＡ７はそれぞれの機能に対する副機能コードで、ユーザーが自由に設定できる。ただし、このフォーマットは固定であり、必要ない場合でもエリアは確保する必要がある。

送受信バッファアドレスはデータの送信、受信の時のみ必要なデータです。

④ 回線のオープン

コマンド：００Ｈ

パラメータ：自由

送受信バッファアドレス：必要なし

リターンコード：ＡＨレジスタ　００Ｈ

ＡＬレジスタ　００Ｈ

⑤ 回線のクローズ

コマンド：０１Ｈ

パラメータ：自由

送受信バッファアドレス：必要なし

リターンコード：ＡＨレジスタ　００Ｈ

ＡＬレジスタ　００Ｈ

⑥　データの送信

コマンド：１０Ｈ

パラメータ：自由

送受信バッファアドレス：送信バッファアドレス

リターンコード：ＡＨレジスタ　ＳＥＮＤ　ＰＡＲＴのＲＥＴＵＲＮで記述したリターンコード

ＡＬレジスタ　８０Ｈ＝ＴＩＭＥＯＵＴエラー

４０Ｈ＝送信エラー

１０Ｈ＝バッファエリアオーバーフロー

００Ｈ＝エラーなし

⑦　データの受信

コマンド：１１Ｈ

パラメータ：自由

送受信バッファアドレス：受信バッファアドレス

リターンコード：ＡＨレジスタ　ＲＥＣＥＩＶＥ　ＰＡＲＴのＲＥＴＵＲＮで記述したリターンコード

ＡＬレジスタ　１０Ｈ＝バッファエリアオーバーフロー

０４Ｈ＝フレーミングエラー

０２Ｈ＝オーバーランエラー

０１Ｈ＝パリティエラー

００Ｈ＝エラーなし

⑧　その他

前記④～⑦以外のコマンドをＡＨレジスタにセットすると、ＡＨレジスタ、

ＡＬレジスタ共に（ＯＦＦＨ）のリターンコードがセットされる。

5.3　ユーザープログラム作成時の注意点

① 回線制御プログラム内では約１６０バイトのスタックエリアを使用するが、回線制御プログラム内にはスタックエリアを用意していない。

従って、ユーザープログラム内で回線制御プログラム用のスタックを考慮してスタックエリアを確保しなければならない。

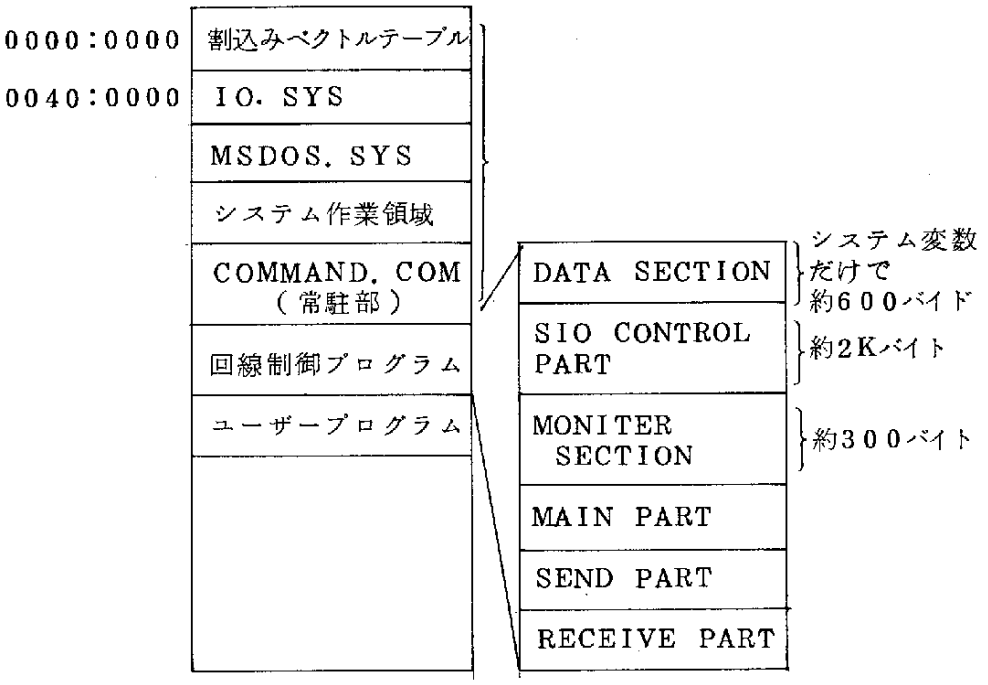
② 回線制御プログラム内には、送受信バッファを用意していない。

この為、ユーザープログラム内で送受信用バッファエリアを確保しなければならない。

回線制御プログラムは、ユーザープログラム内の指定された送受信バッファエリアを直接アクセスする。

6. NCLで作成された回線制御プログラムの構造及び機能

6.1 メモリーマップ



6.2 各セクションパートの機能

① MONITOR SECTION

ユーザープログラムと、回線制御プログラムとの間の情報のやりとりを行なう。

このセクションのプロトコルはユーザーが定義する必要はない。

② DATA LINK SECTION

a MAIN PART

データリンクの確立及び、SEND PART, RECEIVE PART の制御を行なう。

b SEND PART

テキストの送信及びその応答の受信等、一連のテキスト送信シーケンス

の実行を行なう。

c　RECEIVE PART

テキストの受信及びその応答の送信等、一連のテキスト受信シーケンスの実行を行なう。

d　SIO CONTROL PART

ハードウェア割込処理（受信割込、タイマー割込）及び、送受信モードのセット、データの送受信等、シリアルインターフェイスに対する命令を実行する。このＰＡＲＴのプロトコルはユーザーが定義する必要はない。

③　DATA SECTION

CONSTANT SECTIONで定義した定数、システムレジスタ、フラグ等が確保されている。

7. NCL言語仕様

以下の構文の説明で、{と}で囲まれた部分は、いずれか1つを選択することを意味する。また[と.]で囲まれた部分は必要なければ、省略できる。

Interface Section でディフォルトを使用する場合は、その項目は定義する必要はない。

7.1 NCL言語の構造

NCL言語は次の3種類のセクションにより構成され、各セクションは順番に記述されなければならない。

① INTERFACE SECTION

NCL言語で作成された回線制御プログラムが実行されるコンピュータシステムの環境を定義する。

a STATION PART

端末アドレスや、バッファの長さ等のステーション情報を定義する。

b CODE PART

回線上のコード体系等を定義する。

② CONSTANT SECTION

DATA LINK SECTION内で使用する定数を定義する。

③ DATA LINK SECTION

データ転送のプロトコルを定義する。

a MAIN PART

監視シーケンスの送受信のプロトコルを定義する。

b SEND PART

データの送信シーケンスのプロトコルを定義する。

c RECEIVE PART

データの受信シーケンスのプロトコルを定義する。

7.2 基本記号

ＮＣＬで使用できる文字は次のものから構成される。

英大字　：Ａ～Ｚ

英小字　：ａ～ｚ

カナ文字：ア～ン

数　字　：０～９

特殊文字：＋　－　＊　／　＝　：　＜　＞　．　（　）　”

7.3 語

語には、CONSTANT SECTIONで定義する定数名、DATA LINK SECTION内で使用するラベル名とＮＣＬ言語が定義した指定語（レジスター名、フラグ名、動詞等）がある。

語は英大字、英小字、数字、カナ文字で構成される。語の長さは１桁～８桁である。先頭の１桁は英字であること。指定語、定数名、ラベル名の英字には大文字と小文字の区別はない。

定数名、ラベル名は指定語と重複せず、かつユニークでなければならない。

語の区切り記号は、スペース、演算子、句読点記号である。

7.4 文

文は語と演算子、句読点記号で構成され、手続きを記述する最小単位である。

１行は最大１２８桁であり、１行内に複数の文があってもよい。この場合、文の区切り記号はスペースである。１行内の文の終りはピリオド、行の終りはキャリッジ・リターンである。１つの文が複数行にまたがってはいけない。

7.5 ラベル

ラベルは「ラベル名：」と記述されたもので、DATA LINK SECTION

の各ＰＡＲＴ内に自由に設定できる。ラベル名の記述は語の記述方法に従う。

7.6　定数

① 文字定数

A〝○○○○〟

先頭のＡは文字定数であることを表わす。〝と〟で囲まれた定数がＡＳＣⅡコードで連続して確保される。使用可能な文字は、ＡＳＣⅡで定義されている全ての文字及び漢字コードである。又英字の場合、大文字と小文字は区別される。

最大８バイト。

② １６進定数

X〝○○○○〟

先頭のＸは１６進定数であることを表わす。〝と〟で囲まれた定数が１６進として連続して確保される。使用可能な文字は、数字（０～９）と英字（Ａ～Ｆ、ａ～ｆ）である。最大８バイト。

③ 数値定数

○○○○○

０～９までの数字から成る１０進の整数定数。０～６５５３５の整数が定義可能である。

7.7　システムレジスタ、システムフラグ及び送受信バッファ

ＮＣＬは以下に示す７種類３１個の１バイトレジスタと３種類３個の１バイトフラグを用意している。

これらのレジスタ、フラグはCONSTANT　SECTION内で定義する必要はない。

DATA　LINK　SECTION 内でこれらのレジスタは自由に参照、変更することができる。ただし、フラグは参照だけが可能である。

① レジスタ

a ＢＣＣ

ブロックチェック用の１バイトのレジスタ。

b ＣＲＣ

フレームチェック用の２バイトのレジスタ。

バイト単位でアクセスする場合はＣＲＣＯ（上位バイト）、ＣＲＣ１（下位バイト）を使用する。

c ＲＧ０～ＲＧ７

汎用の１バイトレジスタ。

d ＰＡＲＡ０～ＰＡＲＡ７

ユーザープログラムと回線制御プログラムとの間の情報交換エリア中にあるパラメータを示す８個の１バイトレジスタ。

ユーザープログラムから回線制御プログラムに割込み命令が出されると、無条件にパラメータはＰＡＲＡ０～ＰＡＲＡ７に自動的にセットされる。

e ＢＩＴＳ

ビット操作用の１バイトのレジスタ。

ビット操作を行なう時は、時前にデータをＢＩＴＳにセットする必要がある。

f ＣＨＲ

送受信用の１バイトのレジスタ。

g ＡＤＤＲ０、ＡＤＤＲ１

自局の端末アドレスが格納されている２個の１バイトレジスタ。

端末アドレスはコンパイル時に設定される為、DATA LINK SECTION内で設定する必要はない。

１バイトのアドレスを指定すると、上位バイトがＡＤＤＲ０に設定されＡＤＤＲ１はＮＵＬＬとなる。２バイトのアドレスを指定すると、上位バイトがＡＤＤＲ１に設定される。

② フラグ

a ＥＲＲＯＲ

送受信時等にエラーが発生した場合、このフラグにエラーコードがセットされる。又、ユーザープログラムに送られるリターンコード（ＡＨレジスタ）にもセットされる。

b ＲＲＥＱＵＥＳＴ

ユーザープログラムから「テキスト受信」コマンドが出されると、このフラグはＯＮの状態（０１Ｈ）になる。

テキスト受信完了後、又は受信中断後にＯＦＦの状態（００Ｈ）になる。

c ＳＲＥＱＵＥＳＴ

ユーザープログラムから「テキスト送信」コマンドが出されると、このフラグはＯＮの状態（０１Ｈ）になる。

テキスト送信完了後、又は送信中断後にＯＦＦの状態（００Ｈ）になる。

③ 送受信バッファ

回線制御プログラム内には送受信用バッファは用意していない為、ユーザープログラム内で確保する必要がある。

ユーザープログラムから「テキストの送信」「テキストの受信」コマンドを出す時に、送受信バッファのアドレスをセットしなければならない。

ＮＣＬ内でこれらのバッファを使用するときのバッファ名は送信バッファは「ＳＢＵＦ」、受信バッファは「ＲＢＵＦ」である。

7.8 INTERFACE SECTION

① STATION PART

a MACHINE NAME

構文

$$\text{MACHINE NAME} = \left\{ \begin{array}{l} \text{PC9800} \\ \text{PASOPIA1600} \end{array} \right\}$$

説明

ＮＣＬで作成された回線制御プログラムを実行させるコンピュータシステムの名称を記述する。

b　STATION ADDRESS

構文

$$\text{STATION ADDRESS} = \left\{ \begin{array}{l} \text{文字定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

端末アドレスを２桁以内の文字定数、又は４桁以内の１６進定数で定義する。ここで定義したアドレスはＡＤＤＲＯ、ＡＤＤＲ１レジスタに格納され、DATA LINK SECTION内で参照可能である。

c　BUFFER LENGTH

構文

$$\text{BUFFER LENGTH} = \left\{ \begin{array}{l} \text{１０進定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

送受信バッファの容量を定義する。単位はバイトである。

最大６５５３５バイトである。

d　INT VECTOR

構文

$$\text{INT VECTOR} = \left\{ \begin{array}{l} \text{１０進定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

アプリケーションプログラムから回線制御プログラムへ動作要求をする時のソフトウェア割込みベクター番号を定義する。

最大２５５である。

ＭＳ－ＤＯＳシステムが使用しているベクター番号を使用すると、暴走

の可能性があり、十分注意してベクター番号を設定しなければならない。

e　COMMUNICATION MODE

構文

$$\text{COMMUNICATION MODE} = \left\{ \begin{array}{l} \text{HALF} \\ \text{FULL} \end{array} \right\}$$

説明

使用する回線が半二重（ＨＡＬＦ）が全二重（ＦＵＬＬ）かを定義する。

ディフォルトはＨＡＬＦである。

f　ＣＬＯＣＫ

構文

$$\text{CLOCK} = \left\{ \begin{array}{l} \text{１０進定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

コンピュータセットのクロック周波数を定義する。

単位はMHz。

g　ＴＩＭＥＯＵＴ

構文

$$\text{TIMEOUT} = \left\{ \begin{array}{l} \text{１０進定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

タイマー割込み用のタイムアウト時間を定義する。

単位はmsec。ディフォルトは５秒である。

h　SEND DELAY TIME

構文

$$\text{SEND DELAY TIME} = \left\{ \begin{array}{l} \text{１０進定数} \\ \text{１６進定数} \end{array} \right\}$$

説明

受信状態から送信状態に変わる待ち時間を定義する。

単位はmsec。ディフォルトは1秒である。

i　RECEIVE DELAY TIME

構文

$$\text{RECEIVE DELAY TIME} = \left\{ \begin{array}{l} 10\text{進定数} \\ 16\text{進定数} \end{array} \right\}$$

説明

送信状態から受信状態に変わる待ち時間を定義する。

単位はmsec。ディフォルトは0秒である。

j　SPEED

構文

SPEED＝10進定数

説明

回線上のデータ伝送速度を定義する。

可能な速度は、300、600、1200、2400、4800、9600、19200、38400　BPSである。

② CODE PART

a　SET

構文

$$\text{SET} = \left\{ \begin{array}{l} \text{ASCII} \\ \text{EBCDIC} \end{array} \right\}$$

説明

回線上のコード体系がASCII系かEBCDIC系かを定義する。

ディフォルトはASCIIである。

b　BIT

構文

$$\mathrm{BIT}=\left\{\begin{array}{l}7\\8\end{array}\right\}$$

説明

1キャラクターが7ビット系か8ビット系かを定義する。

EBCDIC系の場合、7ビットはエラーとなる。

ディフォルトは8ビットである。

c　PARITY

構文

$$\mathrm{PARITY}=\left\{\begin{array}{l}\mathrm{NO}\\\mathrm{ODD}\\\mathrm{EVEN}\end{array}\right\}$$

説明

パリティチェックの種類を定義する。

NO：パリティ無し

ODD：奇数パリティ

EVEN：偶数パリティ

8ビット系の場合、ODD、EVENはエラーとなる。

ディフォルトはNOである。

d　MODE

構文

$$\mathrm{MODE}=\left\{\begin{array}{l}\mathrm{SYNC}\\\mathrm{ASYNC}\end{array}\right\}$$

説明

伝送方式がSYNC（同期式）かASYNC（非同期式）かを定義する。

ディフォルトはASYNCである。

e　SYNC　CHR

構文

$$\text{SYNC}\quad \text{CHR}=\left\{\begin{array}{l}10\text{進定数}\\ \\ 16\text{進定数}\end{array}\right\}$$

説明

伝送方式が同期式の場合の同期コードを定義する。最大2バイト。

非同期式の場合、ＳＹＮＣ　ＣＨＲを定義するとワーニングとなり、無視される。

f　ＳＴＯＰ　ＢＩＴ

構文

$$\text{STOP}\quad \text{BIT}=\left\{\begin{array}{l}1\\ \\ 2\end{array}\right\}$$

説明

伝送方式が非同期式の場合のストップビットの数を定義する。

同期式の場合、ＳＴＯＰ　ＢＩＴを定義するとワーニングとなり、無視される。ディフォルトは1ビットである。

g　ＳＩ　ＣＨＲ

構文

$$\text{SI}\quad \text{CHR}=\left\{\begin{array}{l}10\text{進定数}\\ \\ 16\text{進定数}\end{array}\right\}$$

説明

7ビット系の場合、シフトインキャラクターを１０進定数又は１６進定数で定義する。１バイト。

8ビット系の場合、ＳＩ　ＣＨＲを定義すると、ワーニングとなり、無視される。

h　ＳＯ　ＣＨＲ

構文

$$\text{SO}\quad \text{CHR}=\left\{\begin{array}{l}10\text{進定数}\\ \\ 16\text{進定数}\end{array}\right\}$$

説明

7ビット系の場合、シフトアウトキャラクターを10進定数又は16進定数で定義する。1バイト。

8ビット系の場合、SO　CHRを定義すると、ワーニングとなり、無視される。

7.9 CONSTANT SECTION

構文

$$\text{定数名} = \left\{ \begin{array}{l} \text{文字定数} \\ \text{10進定数} \\ \text{16進定数} \end{array} \right\}$$

説明

右辺の定数に左辺の定数名が付けられる。

7.10 DATA LINK SECTION

① 送信状態、受信状態のセット

構文

$$\text{SET} \left\{ \begin{array}{l} \text{SEND} \\ \text{RECEIVE} \end{array} \right\}$$

説明

端末を送信可能状態（SEND）あるいは受信状態（RECEIVE）にセットする。データを送信する場合、端末は必ず送信可能状態でなければならない。又、受信する場合は、必ず受信可能状態でなければならない。

② データの送信

構文

$$\text{SEND} \left\{ \begin{array}{l} \text{定数名} \\ \text{レジスタ名} \\ \text{イミディエイトデータ} \end{array} \right\} \quad [\,(\text{ERR}=\text{ラベル名})\,]$$

説明

CONSTANT SECTIONで定義した定数、レジスタの内容、又はイミディエイトデータを送信する。

エラーのラベル名を指定すると、送信時にエラーが発生した場合，ラベルへＪＵＭＰする。この時、エラー情報はＥＲＲＯＲレジスタにセットされており、これを参照することができる。

エラーのラベル名を指定しないと、送信時にエラーが発生した場合、ＥＲＲＯＲレジスタにエラーコードがセットされ、次のステップにコントロールが移る。

ＳＥＮＤ命令を行う場合は、あらかじめ「ＳＥＴ　ＳＥＮＤ」命令で送信可能状態にしておかねばならない。

③　データの受信

構文

ＲＥＣＥＩＶＥ　（〔ＴＩＭＥＯＵＴ＝タイムアウト時間、〕

〔ＥＲＲ＝ラベル名〕）

説明

回線より１キャラクターのデータを受信し、ＣＨＲレジスタにセットする。

ＴＩＭＥＯＵＴ時間を指定すると、その時間内に受信がされない場合、タイムアウトエラーになる。ＴＩＭＥＯＵＴ時間を指定しないと、INTERFACE SECTIONで指定したＴＩＭＥＯＵＴが使用される。

エラーのラベル名を指定すると、タイムアウトエラーを含めて受信時にエラーが発生した場合、ERRORレジスターにエラーコードがセットされて、指定されたラベルへＪＵＭＰする。

エラーのラベル名を指定しないと、受信時にエラーが発生した場合、ＥＲＲＯＲレジスタにエラーコードがセットされ、次のステップにコントロールが移る。

RECEIVE命令を行う場合は、あらかじめ「SET　RECEIVE」命令

で受信可能状態にしておかねばならない。

④　レジスタからバッファエリアへの１バイト転送

構文

ＰＵＴ　レジスタ名〔（ＥＲＲ＝ラベル名）〕

説明

指定したレジスタより、１バイトのデータを、ユーザープログラムから渡されたバッファアドレスのエリアにセットする。ＰＵＴ命令実行後、受信バッファアドレスは自動的にインクリメントされる。受信バッファアドレスがバッファエリアをオーバーした場合、エラーとなりＰＵＴ命令は実行されない。この時、エラーのラベル名を指定すると、そのラベルへJUNPする。エラーのラベル名を指定しないと、次のステップへコントロールは移る。

ＲＥＣＥＩＶＥ　ＰＡＲＴ内でのみ使用できる。

⑤　バッファエリアからレジスタへの１バイトの転送

構文

ＧＥＴ　レジスタ名　〔（ＥＲＲ＝ラベル名）〕

説明

ユーザープログラムから渡されたバッファアドレスのエリアから、１バイトのデータを取り出し、指定したレジスタにセットする。ＧＥＴ命令実行後、送信バッファアドレスは自動的にインクリメントされる。送信バッファアドレスがバッファエリアをオーバーした場合、エラーとなり、ＧＥＴ命命令は実行されない。この時、エラーのラベル名を指定すると、そのラベルへＪＵＭＰする。

エラーのラベル名を指定しないと、次のステップへコントロールは移る。

ＳＥＮＤ　ＰＡＲＴ内でのみ使用できる。

⑥　バッファアドレスの初期化

構文

$$INIT = \left\{ \begin{array}{l} SBUF \\ RBUF \end{array} \right\}$$

説明

送信バッファ（ＳＢＵＦ）アドレスあるいは受信バッファ（ＲＢＵＦ）アドレスをバッファエリアの先頭にセットする。

バッファアドレスの初期化を行わないと、ＧＥＴ命令あるいはＰＵＴ命令の実行時にエラーとなる恐れがある。

⑦ レジスタ、送受信バッファ内容の初期化

構文

$$CLEAR = \left\{ \begin{array}{l} \text{レジスタ名} \\ SBUF \\ RBUF \end{array} \right\}$$

説明

レジスタあるいは送受信バッファの内容をＮＵＬＬレコードで初期化する。

⑧ 代入

構文

$$\text{レジスタ名} = \left\{ \begin{array}{l} \text{レジスタ名} \\ \text{定数名}[\,(P)\,] \\ \text{イミディエイトデータ} \end{array} \right\}$$

説明

右辺で指定したエリアの内容を左辺のレジスタに代入する。

Ｐは定数エリアの先頭バイトを０とした相対位置を示す。

Ｐを指定しない場合は、先頭の１バイトが代入される。

⑨ 算術演算（２進）

構文

$$\text{レジスタ名} = \left\{ \begin{array}{l} \text{レジスタ名} \\ \\ \text{定数名}[\,(P)\,] \end{array} \right\} \left\{ \begin{array}{l} \pm \\ \times \\ / \end{array} \right\} \left\{ \begin{array}{l} \text{レジスタ名} \\ \text{定数名}[\,(P)\,] \\ \text{イミディエイトデータ} \end{array} \right\}$$

説明

右辺の１バイト同志の符号無し四則演算を行ない、結果を左辺のレジスタに代入する。

加算はアセンブラのＡＤＤ

減算はアセンブラのＳＵＢ

乗算はアセンブラのＭＵＬ

除算はアセンブラのＤＩＶ

を実行する。

演算時のオーバーフロー、アンダーフロー、及び除算時の０による除算はチェックをしていない為、必要ならば演算直後にフラグチェックのアセンブラ。ルーチンを組み込むことができる。

特に０による除算を行なった場合は、システムがＨＡＬＴするため、事前に必ずチェックする必要がある。

⑩　論理演算

構文

$$\text{レジスタ名} = \left\{\begin{array}{l}\text{レジスタ名}\\ \\ \text{定数名〔（Ｐ）〕}\end{array}\right\} \left\{\begin{array}{l}\text{ＡＮＤ}\\ \text{ＯＲ}\\ \text{ＸＯＲ}\end{array}\right\} \left\{\begin{array}{l}\text{レジスタ名}\\ \text{定数名〔（Ｐ）〕}\\ \text{イミディエイトデータ}\end{array}\right\}$$

説明

右辺の１バイト同志の論理演算を行ない、結果を左辺のレジスタに代入する。

ＡＮＤ＝論理積

ＯＲ　＝論理和

ＸＯＲ＝排他的論理和

必要ならば演算直後にフラグチェックのアセンブラ・ルーチンを組み込むことが出来る。

⑪　ＢＩＴＳレジスタ内の特定ビットのセット、リセット

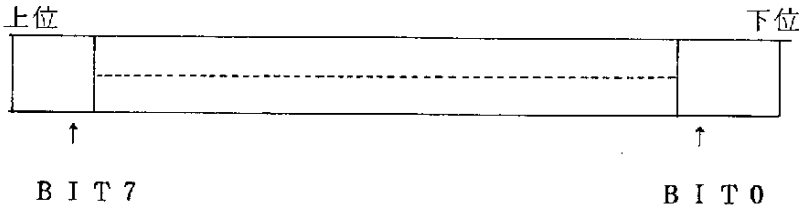
構文

$$SET = \left\{ \begin{array}{l} BIT0 \\ BIT7 \end{array} \right\} \left\{ \begin{array}{l} ON \\ OFF \end{array} \right\}$$

説明

ＢＩＴ０～ＢＩＴ７はＢＩＴＳ内の特定のビットを表わす。

ＯＮの時は１に、ＯＦＦの時は０にセットされる。



⑫　ビットシフト

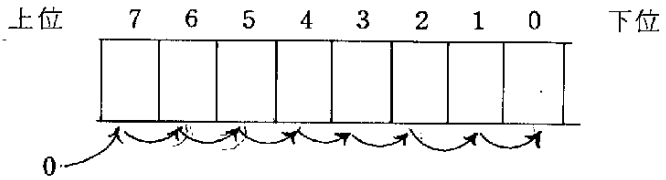
構文

ＲＳＨＩＦＴ　レジスタ名

説明

指定したレジスタ内のデータを右に１ビットシフトする。

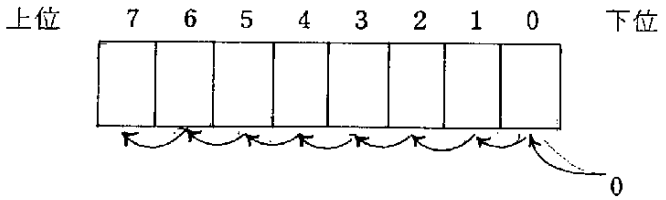
左端のビットには０がセットされる。



説明

指定したレジスタ内のデータを左に１ビットシフトする。

右端のビットには０がセットされる。



⑬ 比較

構文

ＩＦ　条件式　　ＴＨＥＮ　処理１　〔ＥＬＳＥ　処理２〕。

説明

条件式が真ならば、処理１を実行する。条件式が偽ならば、処理２を実行する。ただし、ＥＬＳＥ句がなければ、偽の時は次のステップにコントロールが移る。

処理１、処理２はそれぞれ１つの文でも複数の文でもかまわない。

又、ＩＦ文のネスティングは許されない。

条件式の構文

レジスタ名　比較演算子 { レジスタ名 / 定数名〔（Ｐ）〕 / リテラル }

フラグ名　　比較演算子 { ＯＮ / ＯＦＦ }

比較演算子

＝	：	等しい
＞	：	大きい
＜	：	小さい
ＮＯＴ＝	：	等しくない
ＮＯＴ＞	：	大きくない
ＮＯＴ＜	：	小さくない

ただし、フラグの条件式に使える比較演算子は「＝」と「ＮＯＴ＝」だけである。

⑭ 分岐

構文

ＧＯＴＯ　ラベル名

説明

同一パート内のラベルに、無条件にコントロールが移る。

他のパートへの「ＧＯＴＯ」命令は許されない。

構文

$$\mathrm{CALL}=\left\{\begin{array}{l}\mathrm{SEND\ \ PART}\\ \mathrm{RECEIVE\ \ PART}\end{array}\right\}$$

説明

ＭＡＩＮ　ＰＡＲＴ内からSEND　PARTあるいはRECEIVE　PARTの先頭に、コントロールを移す。

SEND　PARTあるいはRECEIVE　PARTから、他のＰＡＲＴへの「ＣＡＬＬ」命令は許されない。

⑮　ＣＡＬＬされたＰＡＲＴからＭＡＩＮ　ＰＡＲＴへの戻り

構文

ＲＥＴＵＲＮ　［（リターンコード）］

説明

SEND　PARTあるいはRECEIVE　PARTから、ＭＡＩＮ　ＰＡＲＴのＣＡＬＬした次のステップへ、コントロールを戻す。この時、１バイト００Ｈ～０ＦＦＨのリターンコードをユーザープログラムに返すことができる。リターンコードを指定しないと、００Ｈがユーザープログラムに返される。

⑯　コード変換の制御

構文

$$\mathrm{SET\ \ TRANS}=\left\{\begin{array}{l}\mathrm{ON}\\ \mathrm{OFF}\end{array}\right\}$$

説明

TRANS　ONの状態で、データを送信すると、データはCODE　PARTのＳＥＴ命令で定義したコード体系に変換され、送信される。又、受信し

た時は、受信したデータはCODE PARTで定義したコード体系とみなし、内部コード（ＡＳＣⅡ）に変換される。

TRANS OFFの状態では、送受信時のコード体系の変換は行なわない。

⑰ ブロックチェックの制御

構文

$$\text{SET CHECK} = \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$$

説明

CHECK ONの状態では、送受信時にブロックチェックキャラクタはＢＣＣに、ブロックチェックシーケンスはCRC（CRC0、CRC1）にセットされる。

ＢＣＣ、ＣＲＣレジスタのリセットはシステムでは行なわないので、ＣＬＥＡＲ命令で必要なレジスタを初期化する必要がある。

ブロックチェックキャラクタ及びシーケンスの生成方法はＪＩＳ６３６０およびＪＩＳ６３６２に定める方法で行なっている。これ以外のチェック方法を使用する場合は、CHECK OFFの状態でDATA LINK SECTION内に記述する必要がある。

⑱ 待ち状態の保持

構文

WAIT（〔TIMEOUT＝タイムアウト時間〕、〔ＥＲＲ＝ラベル名〕）

説明

指定したタイムアウト時間だけ待ち状態となる。タイムアウト時間を指定しないと、INTERFACE SECTIONで定義したTIMEOUT時間が使われる。

待ち状態で、データを受信すると、次のステップへコントロールが移る。又、タイムアウト時間に受信がされない場合は、タイムアウトエラーとなり、指定したエラーラベルのステップにコントロールが移る。エラーのラ

ベルを指定しないと、ERRORレジスタにエラーコードがセットされ、次のステップへコントロールが移る。

⑲ アセンブラソースプログラムの組み込み

構文

ＥＮＴＥＲ

説明

アセンブラソースプログラムの開始。

この命令の直後から８０８６アセンブラソースを記述できる。

構文

ＥＸＩＴ

説明

アセンブラソースプログラムの終了。

アセンブラソースプログラムは、上記ENTERとEXIT 命令の間でのみ記述できる。

アセンブラの構文は、マイクロソフト製８０８６アセンブラに従う。

次の命令は使用してはいけない。

CLI、ESC、HALT、INT、INTO、IRET、LOCK、WAIT

ＥＸＩＴでアセンブラプログラムから出る時、８０８６用の全てのレジスタ（ＡＸ、ＢＸ、ＣＸ、ＤＸ、ＳＩ、ＤＩ、ＣＳ、ＤＳ、ＥＳ、ＳＰ、等）及びフラグは、アセンブラプログラムに入る直前の状態に戻しておかねばならない。

アセンブラソースプログラム内で使用するラベル名は全て、“@”で始まらなければならない。

又、ＮＣＬ用のレジスタを使用する場合、1バイトレジスタは「ＢＹＴＥ ＰＴＲ　レジスタ名」

2バイトレジスタは「ＷＯＲＤ　ＰＴＲ　レジスタ名」と記述する。

⑳ その他、コーティング時の注意事項

1. 定数名、ラベル名の使用可能な最大個数は特に指定はないが定数名の個数とラベル名の個数の合計が約１０００以内が目安である。

   コンパイル時に「TOO MANY CONSTANT」あるいは「TOO MANY LAVEL」というエラーメッセージが出た場合、使用定数、ラベルの数を減らす必要がある。

2. ＰＣ９８０１で回線スピードを１９２００以上で使用する時は、ＰＣ９８００のクロックは5 MHz で使用しなければならない。

3. データの送信時以外は常にRECEIVE状態にしておくべきである。

4. 回線スピードは最高３８４００ＢＰＳまで定義可能だが、DATA LINK SECTION内の処理内容によっては、処理可能な回線スピードは低くなる恐れがある。

5. 送信時のシフトイン、シフトアウトの付加、送信、受信時の同期コード、シフトイン、シフトアウトの判断、コード変換はシステム内で行なう。従って、これらのコードの制御をDATA LINK SECTION内に定義する必要はない。

6. 次のページにサンプルプログラムを示す。

```
;*****************************************
;**      CONTENTION SAMPLE PROGRAM        **
;*****************************************
INTERFACE SECTION.
STATION PART.
        MACHINE NAME = PASOPIA1600.
        RECEIVE DELAY TIME = 0.
        SEND    DELAY TIME = 50.
        TIMEOUT = 100.
        SPEED = 9600.
        INT VECTOR = X"77".
        BUFFER LENGTH = 1026.
        STATION ADDRESS = A"XX".
CODE PART.
        SET =ASCII.
        BIT=8.
CONSTANT SECTION.
        EOT = X"04".
        ENQ = X"05".
        ACK = X"06".
        NAK = X"15".
        STX = X"02".
        ETX = X"03".
DATA LINK SECTION.
MAIN PART.
L00:
        SET RECEIVE.
L01:
        WAIT (ERR=L10).
        RECEIVE.
        IF CHR = ENQ THEN GOTO L20.
        GOTO L01.
L10:
        IF SREQUEST = ON THEN CALL SEND PART
                              GOTO L00.
        GOTO L01.
L20:
        IF RREQUEST = ON THEN CALL RECEIVE PART
                              GOTO L00.
        SET SEND.
        SEND NAK.
        GOTO L00.

SEND PART.
        RG0=0.
LS00:
        SET SEND.
        SEND ENQ.
        RG0=RG0+1.

        SET RECEIVE.
        RECEIVE.
        IF CHR=ACK THEN GOTO LS10.
```

```
        IF CHR=NAK THEN GOTO LS00.
        IF RG0=30  THEN RETURN (1).
LS01:   RECEIVE (ERR=LS00).
        GOTO LS01.
LS10:
        SET SEND.
        SET CHECK ON.
        CLEAR BCC.
        INIT SBUF.
        SEND STX.
LS11:
        GET CHR (ERR=LS20).
        SEND CHR.
        GOTO LS11.
LS20:
        SEND ETX.
        SEND BCC.

        SET RECEIVE.
        RECEIVE (ERR=LS30).
        IF CHR NOT= ACK THEN GOTO LS30.
        RETURN (X"00").
LS30:
        RETURN (X"FF").

RECEIVE PART.
        SET SEND.
        SEND ACK.
LR00:
        SET RECEIVE.
        CLEAR BCC.
        SET CHECK ON.
        RECEIVE.
        IF CHR NOT= STX THEN GOTO LR30.

        INIT RBUF.
LR10:
        RECEIVE (ERR=LR30).
        PUT CHR (ERR=LR11).
        GOTO LR10.
LR11:
        IF CHR NOT= ETX THEN GOTO LR30.
        SET CHECK OFF.
        RECEIVE (ERR=LR30).
        IF BCC = CHR THEN GOTO LR20.
        SET SEND.
        SEND NAK.
        GOTO LR00.
LR20:
        SET SEND.
        SEND ACK.
        RETURN (X"00").
LR30:
        RETURN (X"FF").
```

```
;*****************************************
;**      POL/SEL SECONDLY SAMPLE PROGRAM **
;*****************************************
INTERFACE SECTION.
STATION PART.
        MACHINE NAME = PC9800.
        RECEIVE DELAY TIME = 0.
        SEND    DELAY TIME = 1000.
        SPEED = 300.
        TIMEOUT = 5000.
        INT VECTOR = X"77".
        BUFFER LENGTH = 2000.
        STATION ADDRESS = A"2A".
CODE PART.
        SET =ASCII.
        BIT=7.
        PARITY=EVEN.
        SI CHR = X"0F".
        SO CHR = X"0E".
CONSTANT SECTION.
        EOT = X"04".
        POL = A"p".
        SEL = A"q".
        ENQ = X"05".
        ACK = X"06".
        NAK = X"15".
        SOH = X"01".
        STX = X"02".
        ETX = X"03".
DATA LINK SECTION.
MAIN PART.
L00:
        SET RECEIVE.
L01:
        RECEIVE (ERR=L01).
        IF CHR NOT= EOT THEN GOTO L01.
        RECEIVE (ERR=L01).
        IF CHR NOT= ADDR0 THEN GOTO L01.
        RECEIVE (ERR=L01).
        IF CHR NOT= ADDR1 THEN GOTO L01.
        RECEIVE (ERR=L01).
        IF CHR=POL THEN GOTO L10.
        IF CHR=SEL THEN GOTO L20.
        GOTO L01.
L10:
        RECEIVE (ERR=L01).
        IF CHR NOT=ENQ THEN GOTO L01.
        IF SREQUEST = ON THEN CALL SEND PART
                              GOTO L00.
        SET SEND.
        SEND EOT.
        GOTO L00.
L20:
        RECEIVE (ERR=L01).
        IF CHR NOT=ENQ THEN GOTO L01.
        IF RREQUEST = ON THEN CALL RECEIVE PART
                              GOTO L00.
        SET SEND.
        SEND NAK.
        GOTO L00.
```

```
SEND PART.
        SET SEND.
        SEND SOH.
        CLEAR BCC.
        SET CHECK ON.
        SEND ADDR0.
        SEND ADDR1.
        SEND STX.
        INIT SBUF.
LS10:
        GET CHR.
        IF CHR=ETX THEN GOTO LS20.
        SEND CHR.
        GOTO LS10.
LS20:
        SEND CHR.
        SEND BCC.
        SET RECEIVE.
        RECEIVE (ERR=LS30).
        IF CHR NOT= ACK THEN GOTO LS30.
        SET SEND.
        SEND EOT.
        RETURN (X"00").
LS30:
        RETURN (X"FF").
;
RECEIVE PART.
        SET SEND.
        SEND ACK.
LR00:
        SET RECEIVE.
        RECEIVE (ERR=LR40).
        IF CHR NOT= SOH THEN GOTO LR40.
        CLEAR BCC.
        SET CHECK ON.
        RECEIVE (ERR=LR40).    ; ADDR0
        RECEIVE (ERR=LR40).    ; ADDR1
        RECEIVE (ERR=LR40).
        IF CHR NOT= STX THEN GOTO LR40.

        INIT RBUF.
LR10:   RECEIVE (ERR=LR40).
        PUT CHR.
        IF CHR NOT= ETX THEN GOTO LR10.
        SET CHECK OFF.
        RECEIVE (ERR=LR40).
        IF BCC = CHR THEN GOTO LR20.
        SET SEND.
        SEND NAK.
        GOTO LR00.
LR20:
        SET SEND.
        SEND ACK.
        RETURN (X"00").

LR40:
        RETURN (X"FF").
```

—— 禁 無 断 転 載 ——

昭和６０年 ３ 月 発行

発行所 財団法人日本情報処理開発協会
東京都港区芝公園３－５－８
機械振興会館内
TEL（４３４）８２１１（代表）

印刷所 株式会社 昌 文 社
東京都港区芝５－２６－３０
TEL（４５２）４９３１（代表）