生産終了
DISCONTINUED

OMRON

プログラマブルコントローラ
SYSMAC

ＡＳＣⅡユニット

形C200H-ASC11/21/31

ユーザーズマニュアル

Man. No. SCCC-324H

おことわり

（1） 本マニュアルの内容の一部または全部を無断で複写、複製、転載することを禁じます。

（2） 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがありますので、予めご了承ください。

（3） 本マニュアルの内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがありましたら、お手数ですが巻末記載の弊社支店または営業所までご連絡ください。その際、巻末記載のマニュアルNo.も併せてお知らせください。

著作権・商標について

・ Windows はマイクロソフト社の登録商標です。
・ その他、本文中に掲載しているシステム名および製品名は、それぞれ各社の商標または登録商標です。

プログラマブルコントローラ
SYSMAC

ＡＳＣⅡユニット

形C200H-ASC11/ASC21/ASC31

ユーザーズマニュアル

はじめに

このたびは、SYSMAC αシリーズ（C200HX/HG/HE、C200HX-Z/HG-Z/HE-Z）、C200HS/Hシリーズ、CSシリーズASCIIユニットをお買い上げいただき、ありがとうございます。

ASCIIユニット 形C200H-ASC11/ASC21/ASC31（本ユニットをASCIIユニットと呼びます）は、弊社の高度な制御技術と豊富な経験により開発された、プログラマブルコントローラ SYSMAC αシリーズ（C200HX/HG/HE、C200HX-Z/HG-Z/HE-Z）、C200HS/Hシリーズ、CSシリーズの高機能I/Oユニットです。
本書では、C200HX/HG/HEとC200HX-Z/HG-Z/HE-Zシリーズの PCを、「C200HX/HG/HE（-Z）」と表記します。

本書は、ASCIIユニットの仕様、取り扱いについて詳しく説明した解説書です。機能、取り扱い方法などについて十分ご理解いただき、ASCIIユニットを正しくご使用ください。
なお、ご使用の際は、PC本体のユーザーズマニュアルを併せてご覧ください。

対象となる読者の方々

本マニュアルは、次の方を対象に記述しています。
電気の知識（電気工事士あるいは同等の知識）を有する方で

- FA機器の導入を担当される方
- FAシステムを設計される方
- FA現場を管理される方

お願い

本マニュアルは、ASCIIユニットを使用する上で、必要な情報を記載しています。お使いになる前に本マニュアルをよく読んで、十分に理解してください。また、お読みになった後も本マニュアルは大切に保管して、いつも手元においてお使いください。

「ご使用に際してのご承諾事項」について

1.保証内容

①保証期間

当社商品の保証期間は、ご購入後またはご指定の場所に納入後1年といたします。

②保証範囲

上記保証期間中に当社側の責により当社商品に故障を生じた場合は、代替品の提供または故障品の修理対応を、製品の購入場所において無償で実施いたします。
ただし、故障の原因が次に該当する場合は、この保証の対象範囲から除外いたします。

a) カタログまたは取扱説明書などに記載されている以外の条件・環境・取扱いならびにご使用による場合
b) 当社商品以外の原因の場合
c) 当社以外による改造または修理による場合
d) 当社商品本来の使い方以外の使用による場合
e) 当社出荷当時の科学・技術の水準では予見できなかった場合
f) その他、天災、災害など当社側の責ではない原因による場合

なお、ここでの保証は、当社商品単体の保証を意味するもので、当社商品の故障により誘発される損害は保証の対象から除かれるものとします。

2.責任の制限

① 当社商品に起因して生じた特別損害、間接損害、または消極損害に関しては、当社はいかなる場合も責任を負いません。

② プログラミング可能な当社商品については当社以外の者が行ったプログラム、またはそれにより生じた結果について当社は責任を負いません。

3.適合用途の条件

① 当社商品を他の商品と組み合わせて使用される場合、お客様が適合すべき規格・法規または規制をご確認ください。また、お客様が使用されるシステム、機械、装置への当社商品の適合性は、お客様自身でご確認ください。これらを実施されない場合は、当社は当社商品の適合性について責任を負いません。

② 下記用途に使用される場合、当社営業担当者までご相談のうえ仕様書などによりご確認いただくとともに、定格・性能に対し余裕を持った使い方や、万一故障があっても危険を最小にする安全回路などの安全対策を講じてください。

a) 屋外の用途、潜在的な化学的汚染あるいは電気的妨害を被る用途またはカタログ・取扱説明書などに記載のない条件や環境での使用

b) 原子力制御設備、焼却設備、鉄道・航空・車両設備、医用機械、娯楽機械、安全装置、および行政機関や個別業界の規制に従う設備

c) 人命や財産に危険が及びうるシステム・機械・装置

d) ガス、水道、電気の供給システムや24 時間連続運転システムなど高い信頼性が必要な設備

e) その他、上記a)～d)に準ずる、高度な安全性が必要とされる用途

③ お客様が当社商品を人命や財産に重大な危険を及ぼすような用途に使用される場合には、システム全体として危険を知らせたり、冗長設計により必要な安全性を確保できるよう設計されていること、および当社商品が全体の中で意図した用途に対して適切に配電・設置されていることを必ず事前に確認してください。

④ カタログなどに記載されているアプリケーション事例は参考用ですので、ご採用に際しては機器・装置の機能や安全性をご確認のうえ、ご使用ください。

⑤ 当社商品が正しく使用されずお客様または第三者に不測の損害が生じることがないよう使用上の禁止事項および注意事項をすべてご理解のうえ守ってください。

4.仕様の変更

カタログ・取扱説明書などに記載の商品の仕様および付属品は改善またはその他の事由により、必要に応じて、変更する場合があります。当社営業担当者までご相談のうえ当社商品の実際の仕様をご確認ください。

5.サービスの範囲

本製品の価格には、技術者派遣などのサービス費用は含まれておりません。
お客様のご要望がございましたら、当社営業担当者までご相談ください。

6.適用範囲

以上の内容は、日本国内での取引および使用を前提としております。

海外でのご使用について

本製品のうち、外国為替および外国貿易管理法に定める輸出許可、承認対象貨物(または技術)に該当するものを輸出(または非居住者に提供)する場合は同法に基づく輸出許可、承認(または役務取引許可)が必要です。

安全上のご注意

安全に使用していただくための表示と意味について

本マニュアルでは、ASCIIユニットおよびプログラマブルコントローラ(PC)を安全に使用していただくために、注意事項を次のような表示と図記号で示しています。

ここで示した注意事項は、安全に関する重大な内容を記載しています。必ず守ってください。

表示と意味は次のとおりです。

⚠警告	**正しい取扱いをしなければ、この危険のために、軽傷・中程度の傷害を負ったり、万一の場合には重傷や死亡に至る恐れがあります。また、同様に重大な物的損害をもたらす恐れがあります。**

⚠注意	**正しい取扱いをしなければ、この危険のために、時に軽傷・中程度の傷害を負ったり、あるいは物的損害を受ける恐れがあります。**

安全上の要点

製品を安全に使用するために実施または回避すべきことを示します。

使用上の注意

製品が動作不能、誤動作、または性能・機能への悪影響を予防するために実施または回避すべきことを示します。

お願い

本文中の「お願い」は安全上の要点、使用上の注意と同等の内容を示します。

図記号の説明



⃠ 記号は、禁止を意味しています。
具体的な内容は、⃠ の中と文章で示します。
左図の場合は、「分解禁止」を表します。



△ 記号は、注意（警告を含む）を意味しています。
具体的な内容は、△ の中と文章で示します。
左図の場合は、「感電注意」を表します。

⚠警告

通電中は、分解しないでください。



感電の恐れがあります。

⚠注意

ターミナルからROMSAVEコマンドを実行することによって、本ユニット内のフラッシュROMにBASICプログラムを書込み中（ターミナルに、プロンプト表示が出るまでの間）に、PCの電源を絶対に切らないでください。万一、書込み中にPCの電源を切ると、フラッシュROMが電気的に壊れ、セーブできなくなる可能性があります。



安全上の要点

ASCIIユニットおよびプログラマブルコントローラ（PC）を使用する際には、次のことに気を付けてください。

本ASCIIユニットをCSシリーズCPUユニットで使用する場合、BASICプログラム上のPC READ命令、PC WRITE命令、PC READ@命令、PC WRITE@命令のいずれかの命令で、CSシリーズ CPUユニットのI/Oメモリエリアを読み書きするとき、エリア種別／アドレスの指定において一定の制限があります。既存のBASICプログラムを使用してCSシリーズ上でASCIIユニットを使用する場合は、とくに注意が必要です。

- 信号線の断線、瞬時停電による異常信号などに備えて、ご使用者側でフェールセーフ対策を施してください。
- 強い高周波ノイズを発生する機器から離して取り付けてください。
- 次のことを行うときは、PC本体の電源をOFFにしてください。
  - 電源ユニット、I/Oユニットなどの各種ユニットやCPUユニット、メモリカセットの着脱をするとき
  - 装置を組み立てるとき
  - ディップスイッチやロータリスイッチを設定するとき
  - ケーブルを接続、配線するとき
  - コネクタを取り付けたり、取り外したりするとき
- 本製品を分解して修理や改造はしないでください。
- 製品を落下させたり、異常な振動・衝撃を与えないでください。
- PCのベース取り付けねじ、端子台のねじ、ケーブルのねじはPC本体のマニュアルで指定した規定トルクで締めてください。
- ワイヤくず混入防止のため、ラベルを付けたまま配線してください。
- 配線後は、放熱のため必ずラベルを外してください。
- 配線を十分に確認してから通電してください。
- コネクタ、配線を十分に確認してから装着してください。
- コネクタ、端子台、メモリユニット、増設ケーブルなどロック機構のあるものは、必ずロックしていることを確認してからご使用ください。
- 作成したユーザプログラムは、十分な動作確認を行った後、本運転に移行してください。
- 設備に影響がないこと確認してから、強制セット／リセットしてください。
- 次の操作は、設備に影響がないことを確認した上で行ってください。
  - PC本体およびユニットの動作モード切替え（電源投入時の動作モード設定を含む）
  - 接点の強制セット／リセット
  - 現在値や設定値の変更
- 接地された金属に触るなどして人体の静電気を放電させてから、ユニットに触れてください。

使用上の注意

●PC本体マニュアルに示す通り、正しく設置してください。

●次のような環境には設置しないでください。

- 日光が直接当たる場所
- 周囲温度や相対湿度が仕様値の範囲を超える場所
- 温度の変化が急激で結露するような場所
- 腐食性ガス、可燃性ガスのある場所
- ちり、ほこり、塩分、鉄粉が多い場所
- 水、油、薬品などの飛沫がかかる場所
- 本体に直接振動や衝撃が伝わる場所

●次のような場所で使用する際は、遮蔽対策を十分に行ってください。

- 静電気などによるノイズが発生する場所
- 強い電界や磁界が生じる場所
- 放射能を被曝する恐れのある場所
- 電源線が近くを通る場所

マニュアル改訂履歴

マニュアル改訂記号は、表表紙・裏表紙の左下に記載されているMan. No. の後尾に付記されます。



改訂記号	改訂年月	改訂理由・改訂ページ
—	1998年　5月	初版
B	1998年　11月	誤り修正、説明追加による改訂 1-9、1-12、2-2、2-4、2-5、2-9、3-6、3-8、3-9、4-2、4-4、4-6、4-7、4-10〜14、5-3、5-4、5-6、6-4、6-5、6-9、6-21、6-23、6-25、6-27、6-30、7-2、7-3、7-12、7-20、7-21、7-48、7-50、7-55、7-61、7-63、7-64、7-72、7-96、7-101、7-102、7-113、7-119、7-124、7-125、7-128、7-131〜7-133、9-4、9-6、9-10、付-5、付-8、付-14、付-19〜付-21
C	1999年　4月	CS1シリーズについての記述追加
D	2001年　2月	誤り修正、説明追加による改訂 3-6、6-34、7-6、7-8、7-10、7-26、7-94、7-97、7-122、7-139、9-15、付-7、付-8、付-28
E	2001年　10月	説明追加による改訂 7-113、7-114、付-5
F	2002年　4月	説明追加による改訂 7-87、付-9
G	2003年　7月	誤り修正、説明追加による改訂
H	2006年　3月	安全に関するご注意の追加および変更 「CS1」および「CS1シリーズ」を「CSシリーズ」へ変更

本マニュアルの構成

第1章　はじめに　1

第2章　取り付けと接続　2

第3章　リレー／DMの割付　3

第4章　外部汎用機器とのデータ交換　4

第5章　各種の処理　5

第6章　CPUユニットとのデータ交換　6

第7章　BASICプログラムの作成　7

第8章　プログラム例　8

第9章　エラーとその処置　9

付録　付

目次

第1章　はじめに

1—1　概要 1-2
1—2　システム構成 1-5
1—3　内部のしくみ 1-7
1—4　仕様 1-9
1—5　従来機種との比較の概要 1-11
1—6　基本的な使用手順 1-13
■BASICプログラムの作成手順 1-13
■BASICプログラムの設計手順 1-14
■外部機器との接続／運転 1-15

第2章　取り付けと接続

2—1　各部の名称と機能 2-2
■各部の名称 2-2
■前面 2-3
■裏面 2-5
2—2　取り付けと接続 2-6
■取り付け方法 2-6
■外形寸法 2-6
■取り付け可能位置と台数 2-7
■外部機器との接続 2-8
■ターミナルとの接続 2-11
■外部汎用機器との接続 2-11

第3章　リレー／DMの割付

3—1　内部補助リレーへの割付 3-2
■内部補助リレーへの割付の概要 3-2
3—2　データメモリ（DM）への割付 3-10
■データメモリ（DM）への共有メモリの割付の概要 3-10
■データメモリ（DM）の共有メモリへの割付／共有メモリ内容一覧 3-13

第4章　外部汎用機器とのデータ交換

4—1　外部汎用機器とのデータ交換 4-2
■通信ポートのオープン 4-2
■データ送信 4-5
■データ受信 4-8
■通信ポートのクローズ 4-11
■伝送制御信号のタイミングチャート 4-11
■通信エラー処理 4-18

第5章　各種の処理

5−1　各種の処理 5-2
■文字列処理 5-2
■ビット処理 5-2
■受信バッファの処理 5-2
■時間処理 5-3
■割込機能 5-4
■繰り返し処理 5-6

第6章　CPUユニットとのデータ交換

6−1　データ交換方法の概要 6-2
■5種類のデータ交換方法の概要 6-2
■CPUユニットからASCIIユニットへの割込のかけ方 6-3
6−2　データ交換方法の選択のしかた 6-4
■データ交換方法の比較表 6-5
6−3　データ交換方法の詳細 6-6
①CPUユニットからのトリガで、ASCIIユニットがI/Oリフレッシュ時に、
CPUユニットのI/Oメモリを読み書きする 6-6
②ASCIIユニット単独で、I/Oリフレッシュ時に、CPUユニットの
I/Oメモリを読み書きする 6-8
③ビット単位または少数ビットのデータをI/Oリフレッシュ時に読み書きする
（主体はどちらでも可） 6-10
④CPUユニットとASCIIユニットが非同期で、各々ASCIIユニットの
共有メモリを読み書きする 6-11
⑤ASCIIユニットからのトリガで、CPUユニットがIOWR/IORD（#FD00）
命令実行時にI/Oメモリのデータを ASCIIユニットに対して読み書きする 6-13
⑥CPUユニットがASCIIユニットの共有メモリへ書き込み後、ASCIIユニット
へ割込をかけて、割込サブルーチン内で、ASCIIユニットが共有メモリから
読み出す 6-15
⑦CPUユニットがASCIIユニットへ割込をかけ、ASCIIユニットは割込を受け
て、共有メモリにデータを書き込む
割込サブルーチン終了後、CPUユニットは共有メモリからデータを読み出す . 6-16
6−4　データ交換のタイミングチャート 6-17
①PC READ／PC WRITE命令のタイミング 6-17
②PC READ@／PC WRITE@命令のタイミング 6-17
③PC PUT命令／PC GET命令のタイミング 6-18
④IOWR（#00□□）/IORD（#00□□）命令のタイミング 6-18
⑤PC EPUT／PC EGET命令のタイミング 6-18
⑥IOWR（#FD00）/IORD（#FD00）と、PC QREAD／PC QWRITE命令のタイミング 6-19
⑦IOWR（#00□□）、IOWR（#CC00）、ON　PC GOSUB、PC EGET命令の組み合わせ時のタイミング . 6-19
⑧IOWR（#CC00）、ON PC GOSUB、PC EPUT、IORD（#00□□）命令の組み合わせ時のタイミング ... 6-20
⑨CPUユニットからASCIIユニットに対して、割込をかける（PC割込） 6-20

6-5 IOWR命令／IORD命令の仕様 6-25
■IOWR命令（インテリジェントI/O書込命令） 6-25
■IORD命令（インテリジェントI/O読出命令） 6-29
■IOWR命令／IORD命令の使い方 6-31
6-6 CSシリーズを使用する場合の留意事項 6-32

第7章 BASICプログラムの作成

7−1 プログラミングの手順 7-2
■BASICプログラムの作成 7-2
■フラッシュROMへの書込み／読出し 7-9
■ASCIIユニット←→ターミナル間のBASICプログラム読出し／書込み 7-10
7−2 文字変数エリアの割付け方式 7-12
7−3 BASICプログラムの起動／停止 7-14
■プログラムの起動方法 7-14
■プログラムを停止（ブレイク）させる方法 7-14
■プログラムを指定した行番号で停止（ブレイク）させる方法 7-15
■プログラムを途中の行で停止（ブレイク）させた後、継続起動させる方法 7-15
7−4 BASIC命令の構成 7-16
■命令実行モード 7-16
■プログラム／行／文 7-17
■BASICで使う文字 7-18
■定数 7-18
■変数 7-20
■配列変数 7-21
■型変換 7-22
■式 7-22
7−5 BASIC命令一覧 7-25
7−6 BASIC命令の詳細 7-34
7−7 ユーザ定義関数 7-136
■BASIC関数 7-136
7−8 デバッグ機能 7-137
■概要 7-137
■ブレイクポイント機能（BRKPTコマンドによる） 7-137
■ステップ機能（STEPコマンドによる） 7-138
■変数モニタ機能（WATCHコマンドによる） 7-138
■トレース機能（TRON命令、TRACEコマンドによる） 7-139

第8章 プログラム例

8−1 データ交換の応用例 8-2
■CPUユニットとASCIIユニットが非同期で各々の処理をする例 8-2
■CPUユニットとASCIIユニット間で高速のデータ交換をする例 8-3
■CPUユニットとASCIIユニット間で、やや大量のデータを交換する例 8-4
■ASCIIユニットが主体でCPUユニットへ伝える例 8-5
■CPUユニットとASCIIユニット間で、ビットデータを変換する例 8-5

第9章　エラーとその処置

9−1　エラーメッセージ一覧 9-2
■割付I/Oエリア 9-2
■エラー種別／コード 9-2
■ターミナル上での表示例 9-2
■エラー種別／コード・エラーメッセージの内容および各処置の一覧 9-3
9−2　異常とその処置 9-10
■異常一覧 9-10
■プログラムメモリ破壊時の処置フロー 9-12
9−3　CPU側の異常表示 9-13
■高機能I/Oユニットの異常一覧 9-13
■高機能I/Oユニット用記憶補助リレーの機能 9-14
9−4　異常履歴の読出し／クリア 9-11
■エラー（異常）履歴テーブルの概要 9-16
■エラー（異常）履歴テーブルの読出し方 9-16
■エラー（異常）をクリアする方法 9-17
9−5　保守 9-18
■取り扱いの際のお願い 9-18
■バッテリの交換方法 9-18

付録

付−1　ご使用上の留意事項 付-2
■形C200H-ASC02から置き換える場合 付-2
■ターミナルの操作時 付-5
■プログラムの実行時 付-5
■プログラムの保存時 付-6
■RS-422A/485ポートについて 付-6
■FCS命令による誤りチェックコードの算出方式について 付-6
■受信割込使用時の通信速度について 付-7
■PCとのデータ授受と通信速度について 付-7
■プログラム実行中断後の操作について 付-7
■命令実行中のBASIC割込実行について 付-7
■サブルーチンからの分岐の繰り返しによるユーザメモリの消費 付-7
■プログラムメモリ破壊時のユニット動作について 付-8
■プログラムの実行までの時間について（立ち上がり時間） 付-8
■プログラムメモリ操作中の電源断、リスタートについて 付-8
■ユーザプログラムメモリ不足時の動作について 付-9
付−2　従来機種との比較の詳細 付-10
付−3　PCフォーマットの説明 付-12
■PCフォーマットの説明 付-12
■PC READフォーマット変数列 付-15
■PC WRITEフォーマット変数列 付-16
付−4　変数のメモリ上の格納形式 付-19
付−5　ケーブルの作成 付-22
■ケーブル処理（シールド線をFGに接続する例） 付-22
■ハンダ付け 付-23
■フード組み立て 付-23

付－6　命令実行時間サンプル 付-24
付－7　サンプルプログラム(基本例) 付-27
付－8　予約語一覧 付-29
付－9　索引 付-30

第1章
はじめに

1−1　概要

ASCIIユニットは、RS-232CまたはRS-422A/485インタフェースを介して、パソコン、ボードコンピュータ、バーコードリーダ、IDシステム、他社PLCなどと、ASCIIデータの交換を自由に行うことができる高機能I/Oユニットです。内部にBASICプログラムを内蔵しているため、自立処理で、汎用外部機器のデータを読み書きできます。これにより、PC上でシーケンス制御と情報処理の融合が容易となります。

●あらゆる汎用外部機器とのASCII通信が可能

とくにプロトコルマクロ機能や上位リンク・無手順では実現できない、次のことが可能です。

・文字列の自由な処理(文字列の一部の抽出、比較、連結、長さ検出)
・受信バッファからの一部データのみの受信処理
・さまざまな割込サブルーチン処理(通信、時間、PC割込など)
・自由なレスポンス処理(リトライ、時間監視など)が可能です。
　これにより、あらゆる通信フレーム、プロトコルに対応でき、RS-232CまたはRS-422A/485ポートをもつ機器となら、ほとんどすべてと、データ交換が可能です。

●多ポート通信システムに最適

1台のCPUユニットに対して、最大10または16ユニット(CPUユニット形式による)装着可能なため、通信を介して、多数の汎用外部機器をモニタリングすることなどに最適です。

●特殊演算ユニットとして機能

ASCIIユニットは、汎用外部機器との通信をせずに、特殊演算(BASIC)ユニットとして使用することも可能です。これにより、PCでは実現しづらいあるいはわかりにくくなる、各種データ演算、文字列処理、論理フロー処理などを行い、CPUユニットのサブCPUとして機能することも可能。CPUユニット側の負荷軽減、システム全体としての高速処理に貢献します。

とくに、CPUユニットからの割込が入ったときに、特殊演算をして、結果を返すことにより、CPUユニットのコ・プロセッサとして機能することが可能です。

とくに、形C200H-ASC11/21/31は、次のような特長があります。

●大容量のユーザメモリエリア

200Kバイト(BASICプログラムエリアと変数エリア)のユーザメモリエリアをもつため、複雑なアプリケーションにも対応可能。

●高速処理

32ビットMPU(従来8ビットMPU)による高速処理が可能。

●RS-422A/485ポートタイプを用意

これにより、長距離通信や、1:N通信も可能です。

●CPUユニットと、さまざまな形での、データ交換が可能

従来の命令(PC READ/WRITE、PC READ@/WRITE@、PC PUT/PC GET命令)に加え、ASCIIユニットからのトリガが可能なPC QREAD/QWRITE命令、ASCIIユニット自身の(CPUユニットとの)共有メモリの読み書きが可能なPC EPUT/EGET命令を用意。さらに、SYSMAC αおよびCSシリーズの場合、CPUユニットのIOWR/IORD命令により、ASCIIユニットに対する即時の読み書きが可能です。

これにより、読み書きのトリガ／主体、読み書きのタイミング、転送データサイズ、プログラムの要／不要など、ユーザのアプリケーションに合わせて、自由にデータ交換方法を選択可能です。

●共有メモリエリアを使用した、高速データ交換を実現

内部にCPUユニットとの共有メモリエリアをもち、ASCIIユニットとCPUユニットが、非同期で、この共有メモリエリアをアクセス可能。これにより、従来のI/Oリフレッシュ時のデータ交換では、そのタイミングが来るまで、CPUユニットまたはASCIIユニットが待ち状態になることがあったが、それがなく、互いに拘束されることなく、命令実行時に即時で、共有メモリに対する読み書きが可能。

さらに、CPUユニットからASCIIユニットへのIOWR命令を使った即時の割込と組み合わせることで、高速データ交換が可能です。

●ビットON／OFF命令もサポート

変数をビット列とみなして、0～31ビットのON／OFFが可能。PCへのビット入力処理が容易。

●割り込み処理が豊富

CPUユニットからASCIIユニットへの割り込み方法に2種類(I/Oリフレッシュ時と、IOWR命令実行即時)可能です。通信回線からのデータ受信による割り込みに5種類(何らかのデータ受信、指定文字受信、指定バイト数受信、先頭文字と最終文字受信、先頭文字と指定文字数受信)可能。また、キー入力からの割り込み、エラー発生時の割り込み、時刻による割り込み、タイマによるワンショットまたはインターバルの割り込みが可能。

●伝送制御信号の制御が自在

伝送制御信号(RS, ER)を自由に制御する命令や、伝送制御信号(CS, DR)を任意にモニタする命令を用意。これにより、各種通信プロトコルへの対応が容易。

●各種誤りチェックコードの計算命令を用意

FCS命令によって、LRC(水平パリティ)、CRC-CCITT(XMODEM方式)、CRC-16(MODBUS方式)、SUM(1バイト)、SUM(2バイト)の誤りチェックコードを計算可能。これにより、従来誤りチェックコード用のプログラムを組んでいた部分が不要となります。

●特殊演算をサポート

指数関数(e^x)、平方根(√)、三角関数(tan、sin、cos)、逆三角関数($\tan^{-1}$、$\sin^{-1}$、$\cos^{-1}$)演算、対数(log)演算が可能です。また、演算結果を倍精度で得ることも可能です。

●各種命令をサポート

条件式を含む繰り返し命令(WHILE/WEND)、構造化プログラミングを可能とするIF命令(@IF...ENDIF)、割込が発生した行番号を記憶する命令(INTRB)などを追加サポート。

●豊富なBASICデバッガ機能

任意の行で実行を停止するBRKPT命令、1行または複数行ずつ実行するSTEP命令、実行停止時の変数の内容を表示するWATCH命令を用意。これにより、BASICプログラムが設計どおり動作するかを容易に確認できます。

また、実行した行番号のトレース(経緯)をモニタするTRON命令も用意。また、そのトレース結果データをトレースバッファに保存し、TRACEコマンドを使用して、後でモニタすることも可能です。これにより、BASICプログラムの実行経緯のチェックが可能です。

●エラー履歴機能

最大30個のエラー履歴を記憶可能。トラブルシューティングに役立ちます。

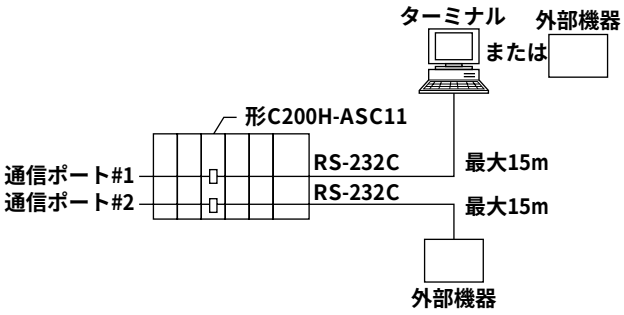
1−2　システム構成

ASCIIユニット形C200H-ASC11/ASC21/ASC31は、C200H、C200HS、SYSMACα（C200HX/HG/HE(-Z)）、CSの各シリーズの高機能I/Oユニットです。

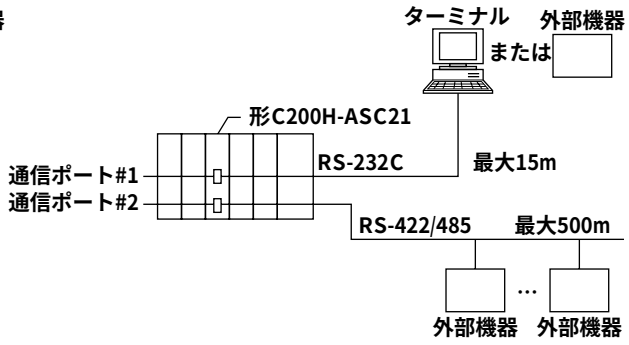
ASCIIユニットの通信ポート#1（形C200H-ASC11/ASC21の場合）またはターミナル専用ポート#3（形C200- ASC31の場合）に、パソコンなどのターミナルをRS-232Cケーブルを使用して接続して、BASICコマンドの入力、BASICプログラムの作成、デバッグ、RUN結果の表示が可能です。

ASCIIユニットの通信ポートに、通信相手となる外部機器を、RS-232Cケーブルまたはツイストペアケーブル（RS-422A/485）を使用して接続して、通信フレームなどのデータをBASICプログラムによって自由に送受信可能です。

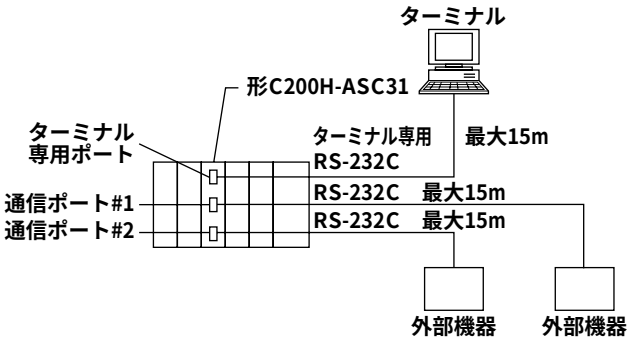
●形C200H-ASC11



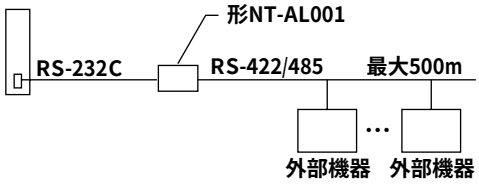
●形C200H-ASC21



●形C200H-ASC31



注：RS-232Cポートで1:N接続したい場合、形NT-AL001（RS-232C→422A/485変換アダプタ）を介すれば、接続可能です。（形C200H-ASC31のターミナル専用ポート#3は除く）



●接続可能なCPUユニット形式

PCシリーズ	CPUユニット形式	CPU装置およびI/O増設装置およびリモートI/O（SYSBUS）子局装置への合計装着可能台数*1	装着位置制限
SYSMACα	C200HE-CPU11/32/42(-Z) C200HG-CPU33/43(-Z) C200HX-CPU34/44(-Z)	10	なし
	C200HG-CPU53/63(-Z) C200HX-CPU54/64(-Z) C200HX-CPU65-Z/85-Z	16	
C200HS	C200HS-CPU01□/21□/31/03/23/33	10	
CSシリーズ	CS1H-CPU66/65/64/63 CS1G-CPU45/44/43/42	16	なし
C200H	C200H-CPU21/22/23/31	10	CPU装置の右側2スロットには装着不可

*1：リモートI/O子局装置（SYSBUS）上の使用制限

リモートI/O子局装置（SYSBUS）上に装着するときは、次の条件となります。

グループ	A	B	C	D
ユニット種類	ASCIIユニット、 高速カウンタユニット、 位置制御ユニット (NC111/112/113/213)、 アナログ入出力ユニット、 IDセンサユニット ファジィ推論ユニット	多点入出力ユニット	温度センサユニット、 音声ユニット	位置制御ユニット (NC211) モーションコントロールユニット
1台のリモートI/Oで使用できる、各グループのときのみの装着可能台数	4ユニット	8ユニット	6ユニット	2ユニット
各グループを混在させて使用するときの装着可能台数	3A+B+2C+6D≦12、かつA+B+C+D≦8			

1−3　内部のしくみ

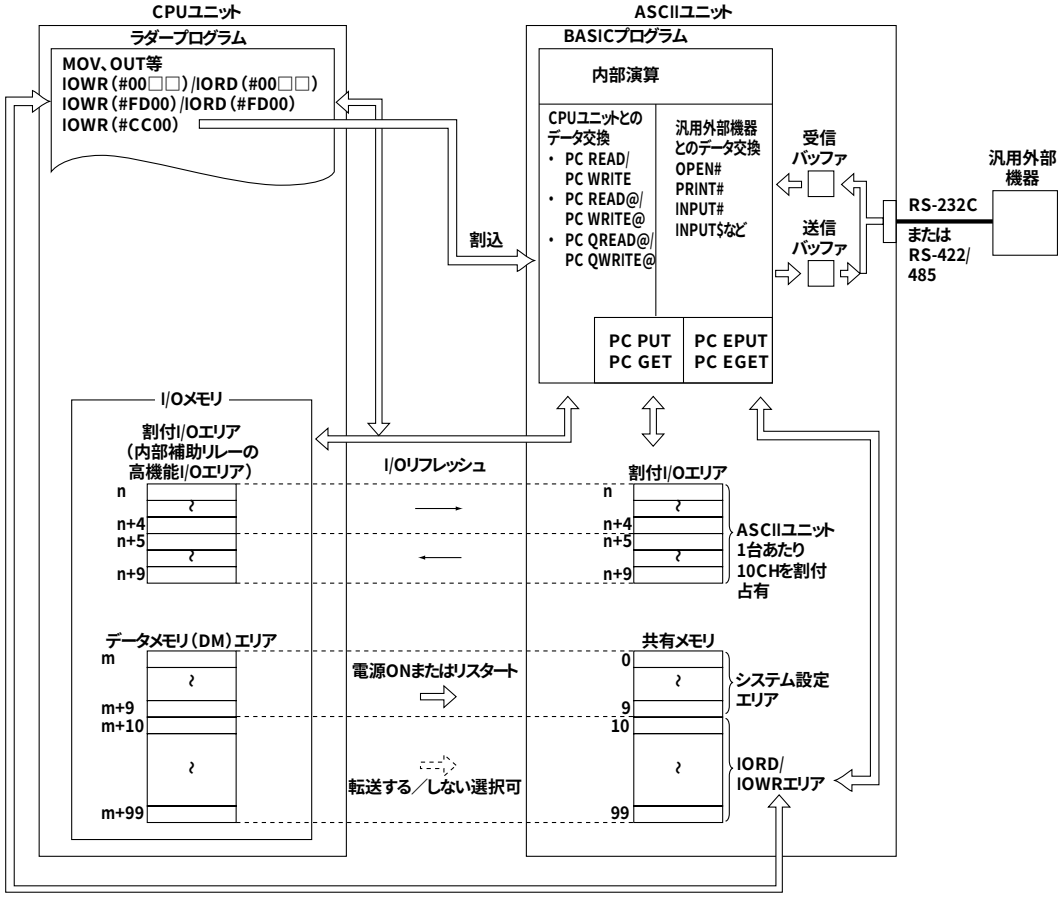
1　高機能I/Oユニットとして、内部補助リレーを10チャネル占有します。
これにより、データ読み書き用トリガ、少ビットのデータ、ステータス、CPUユニットからの割込などを、I/Oリフレッシュ時にプログラムなしで交換します。

2　また、データメモリ(DM)エリアを10チャネル占有します(システム設定エリア)。
これにより、システム設定データ(通信ポートの通信条件、起動時のプログラムNo.など)を、電源ONまたはASCIIユニットのリスタート時に、データメモリからASCIIユニットへ自動転送します。

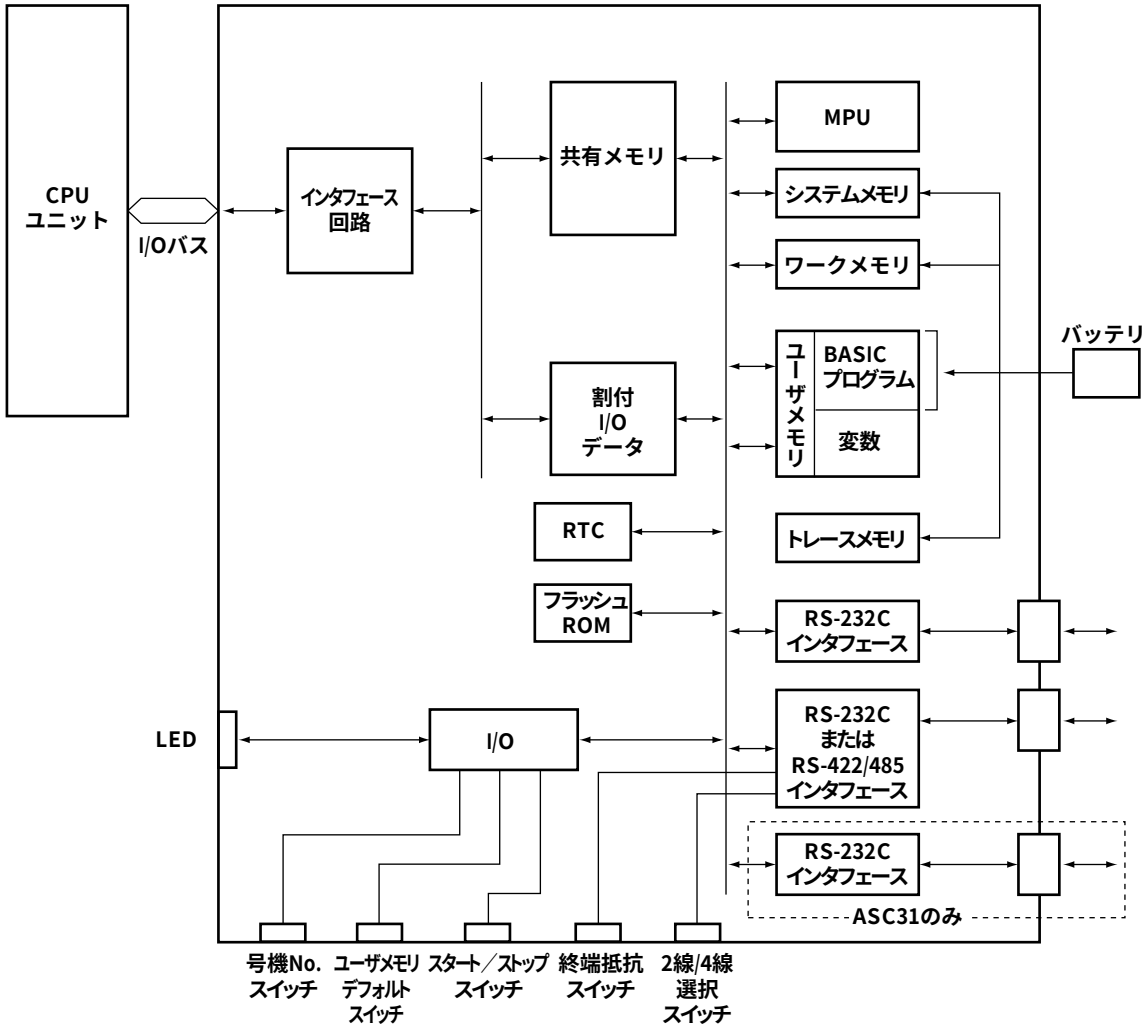
3　内部にCPUユニットとの共有メモリを100チャネル(システム設定エリア10チャネル、汎用エリア90チャネル)もちます。このエリアに対して、ASCIIユニット自身とCPUユニットが、独自に(非同期で)アクセス可能です。
しかも、CPUユニットからのIOWR(インテリジェントI/O書込)／IORD(インテリジェントI/O読出)命令によって、サイクルタイムに依存しないデータ読み書きが可能です。(PCがSYSMACα、CSシリーズの場合のみ)

4　IOWR(インテリジェントI/O書込)命令を使用して、CPUユニットからASCIIユニットに対しサイクルタイムに依存しない割込をかけることができます。(PCがSYSMACα、CSシリーズの場合のみ)

●データ交換の仕組み



●内部構成図



注：・すべてのメモリおよびRTCは、バッテリによってバックアップされています。

・ワークメモリは、ユーザメモリの一部です。

・トレースメモリは、システムメモリの一部です。

1−4　仕様

■一般仕様

SYSMAC C200H、C200HS、C200HX/HG/HE(-Z)、CSシリーズ本体の一般仕様に準じます。

■機能・性能仕様

項　目		仕　様		
形式		C200H-ASC11	C200H-ASC21	C200H-ASC31
通信ポート	ポート1	RS-232C（周辺機器またはターミナル接続）	RS-232C（周辺機器またはターミナル接続）	RS-232C（周辺機器接続）
	ポート2	RS-232C（周辺機器接続）	RS-422A/485（周辺機器接続）	RS-232C（周辺機器接続）
	ターミナル専用ポート	なし	なし	RS-232C（ターミナル接続）
通信仕様	通信方式	半二重（RS-232Cポートは全二重可能）		
	同期方式	調歩同期方式		
	伝送速度	300／600／1200／2400／4800／9600／19200／38400ビット/s（ターミナル専用ポートは38400ビット/sを除く） 注：通信割込み機能との組合せにより、使用可能な伝送速度の使用制限があります。		
	伝送方式	RS-232C：ポイントツーポイント　RS-422A/485：ポイントツーマルチポイント		
	伝送距離	RS-232C：15m、RS-422A/485：500m		
	通信バッファ	受信バッファ：512バイト、送信バッファ：512バイト		
	対応フロー制御	RS／CSフロー（相手外部機器の受信バッファに対して）、Xon／Xoffフロー（相手外部機器およびASCIIユニット自身の受信バッファに対して）		
プログラム	言語	BASIC言語		
	タスク数	シングルタスク		
メモリ	ユーザメモリエリア（BASICプログラム＋変数エリア）	200Kバイト 内容：BASICソースプログラム（ 異なるプログラムを4つまで格納可能）、中間コード、変数 注： BASICソースプログラムは、バッテリにてバックアップ 運転時は必ずバッテリを付けた状態で使用してください。		
	フラッシュROM	200Kバイト ユーザメモリエリアからフラッシュROMに、BASICプログラムを格納可能。電源ON時またはリスタート時に、ユーザメモリエリアにロードすることが可能。		
CPUユニットとのデータ交換エリア	割付入出力エリア	高機能I/Oエリア10CH		
	共有メモリエリア	100CH（バッテリバックアップされない。内訳は以下のとおり） 10CH：システム設定エリア（電源ON時またはリスタート時に、CPUユニットの割付DMエリアからASCIIユニットへ転送） 90CH：汎用エリア（IORD／IOWRエリア）注 注： CPUユニットのIORD／IOWR命令、およびASCIIユニットの PC EPUT／PC EGET 命令から読み書き可能。IORDエリア／IOWRエリアの位置およびサイズは、自由指定可能。電源ON時またはリスタート時に、CPUユニットの割付DMエリアからASCIIユニットへ転送可能。 PC EPUT、PC EGETはC200H/HSでは使用不可です。		

項　目			CPUユニット命令		ASCIIユニット命令	
CPUユニットとのデータ交換方法		CPUユニットからのトリガでASCIIユニットがI/Oメモリを読み書き	MOV、LD等	ー	PC READ / PC WRITE命令	1命令当たり最大255CH転送
		ASCIIユニット単独で、I/Oメモリを読み書き	不要	ー	PC READ@ / PC WRITE@命令	1命令当たり最大255CH転送
		割付I/Oエリアを読み書き	MOV、LD等	ー	PC PUT / PC GET命令	1命令当たり1CH(PC PUT)、2CH(PC GET)転送
		ASCIIユニットからのトリガでCPUユニットがIOWR／IORD #FD00命令で読み書き	IOWR(#FD00) /IORD(#FD00) 注：SYSMACαのみ	1命令当たり最大128CH転送	PC QREAD / PC QWRITE命令	1命令当たり最大128CH転送
		非同期で、各々が共有メモリを読み書き	IOWR(#00□□) /IORD(#00□□) 注：SYSMACαのみ	1命令当たり最大90CH	PC EPUT / PC EGET命令	1命令当たり最大90CH
時計機能	精度	年、月、日、曜日、時、分、秒(うるう年対応) 精度：月差＋70秒／－45秒(25℃) 月差＋5秒／－110秒(0℃) 月差＋0秒／－140秒(55℃) バッテリーによってバックアップ、CPUユニット内のタイマと同期させることも可能				
バッテリー	形式	形C200H-BAT09				
	寿命	25℃で 5年間(これ以上高い温度で使用する場合は、寿命が短くなります。)				
診断機能		CPUウォッチドッグタイマ、バッテリー電圧低下検出				
付属品		コネクタ用プラグ：形XM2A-0901または同等品(形C200H-ASC11/21：2個、形C200H-ASC31：3個) フード：形XM2S-0911または同等品 (形C200H-ASC11/21：2個、形C200H-ASC31：3個)				
内部消費電流(注)		ASC11 DC5V 250mA以下	ASC21 DC5V 300mA以下	ASC31 DC5V 300mA以下		
外形寸法(mm)		35(W)×130(H)×100.5(D)				
質量		400g以下				

注1：RS-232CポートにRS-232C/RS-422A変換ユニット形NT-AL001を使用する場合は、さらに150mA消費します。

注2：ASCIIユニットを使用するときは、必ずバッテリを付けて使用してください。。

1−5　従来機種との比較の概要

形C200H-ASC11/ASC21/ASC31と、従来機種の形C200H-ASC02との比較の概要を以下に示します。ご使用になる前に付録-1「ご使用上の留意事項」を参照して確認してください。

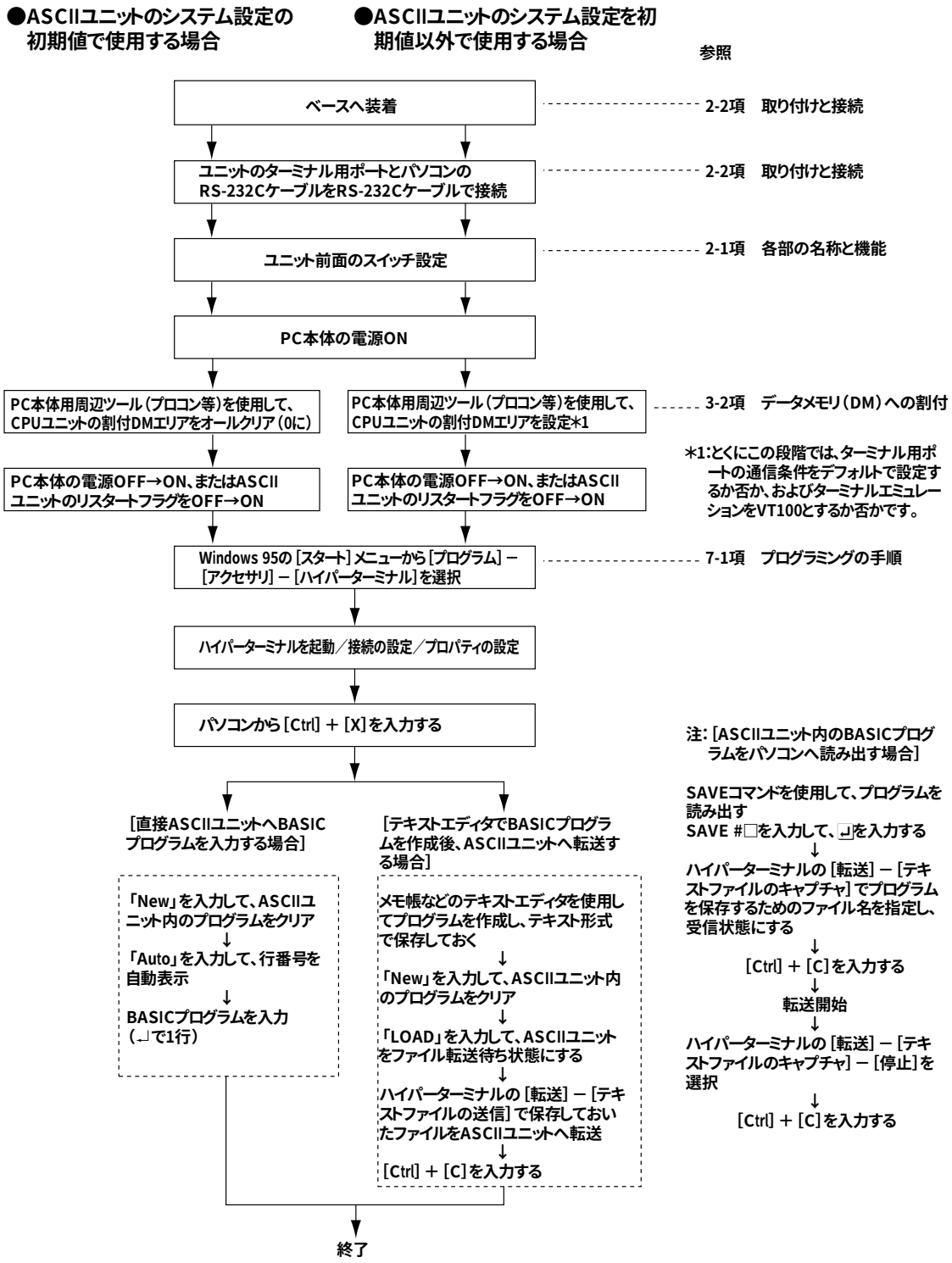
項　　目			形C200H-ASC02	形C200H-ASC11/ASC21/ASC31	備　　考
対応PCシリーズ			C200H、C200HS、SYSMACα、CS1		−
号機No.			0〜9号機	0〜15号機	−
MPU			8ビット	32ビット	−
インタフェース数			2×RS-232C	2×RS-232Cタイプ以外に、（1×RS232C）＋（1×RS-422A/485）タイプ、（2×RS-232C）＋（1×ターミナル専用ポート）タイプあり	−
通信速度			最大19200ビット/s	最大38400ビット/s	−
ユーザメモリエリア（BASICプログラム+変数）エリア			24Kバイト	200Kバイト	−
BASICプログラム保管エリア			EEPROM：24Kバイト	フラッシュROM：200Kバイト	−
BASICプログラム数（ユニットユーザメモリエリア内）			最大3	最大4	−
BASICプログラム行数			0〜63999	１〜65535	−
ASCIIユニットのシステム設定			ユニット裏面ディップスイッチ設定による	システム設定エリア（10CH）による（電源ONまたはリスタート時DMエリアから転送）	−
割付I/Oエリア			4CH	10CH	−
共有メモリ			なし	○（汎用エリア：90CH）	−
命令	ASCIIユニット側のCPUユニットとのデータ交換命令（BASIC）	PC READ / PC WRITE	○（1次元配列まで）	○（多次元配列使用可）	CPUユニットからのトリガでASCIIユニットがI/Oメモリを読み書き
		PC READ@ / PC WRITE@	○（1次元配列まで）	○（多次元配列使用可）	ASCIIユニット単独で、I/Oメモリを読み書き
		PC QREAD / PC QWRITE	なし	○	ASCIIユニットからのトリガでCPUユニットがIOWR／IORD命令で読み書き
		PC PUT / PC GET	○	○	割付I/Oエリアを読み書き
		PC EPUT / PC EGET	なし	○	自身の共有メモリの読み書き
	CPUユニット側のASCIIユニットとのデータ交換命令	IOWR／IORD命令（ASCIIユニットの共有メモリの読み書き、I/Oメモリの転送、割込あり）	なし	○（ただし、PCがSYSMACα、CS1のみ）	−
	誤りチェックコードの計算命令		なし	○（FCS）	−

項　目			形C200H-ASC02	形C200H-ASC11/ASC21/ASC31	備　考
命令	ビットON/OFF命令		なし	○(BITON、BITOFF)	―
	条件繰り返し命令		なし	○(WHILE／WEND)	―
	整数の検索		なし	○(SEARCH)	―
	ラベル分岐		なし	○	―
	デバッグ機能	ブレークポイント命令	なし	○(BRKPT)	―
		トレース命令	なし	○(TRACE)	―
		変数表示命令	なし	○(WATCH)	―
	伝送制御信号	制御命令	なし	○(RS/ERの制御、CS/DRのモニタ)	―
	特殊演算	指数関数	なし	○	―
		平方根	なし	○	―
	算術演算		結果は単精度まで	結果は単精度、倍精度共に可	―
割込	PC割込	割付I/Oによる	○	○	―
		IOWR(#CC00)命令による	なし	○	―
		最大設定数	15(0～F)	99(01～99)BCD	―
	通信(受信データによる)割込	何らかのデータを受信バッファに受信すると割込	○	○	―
		指定文字受信で割込	なし	○	―
		指定バイト数受信で割込	なし	○	―
		指定先頭文字と指定最終文字受信で割込	なし	○	―
		指定先頭文字と指定バイト数受信で割込	なし	○	―
	キー割込		○	○	―
	時刻割込		なし	○	―
	インターバルタイマ割込		なし	○	―
	ワンショットタイマ割込		なし	○	―
	エラー割込		○	○	―
	割込発生時の行番号格納命令		なし	○(INTRB、INTRR、INTRS)	―
異常履歴機能			なし	○(最大30個のエラー履歴を記憶可能。ただし。発生日時なし)	―
内部消費電流			DC5V 200mA	DC5V 250mA または300mA	―
機械語プログラム			あり	なし	―

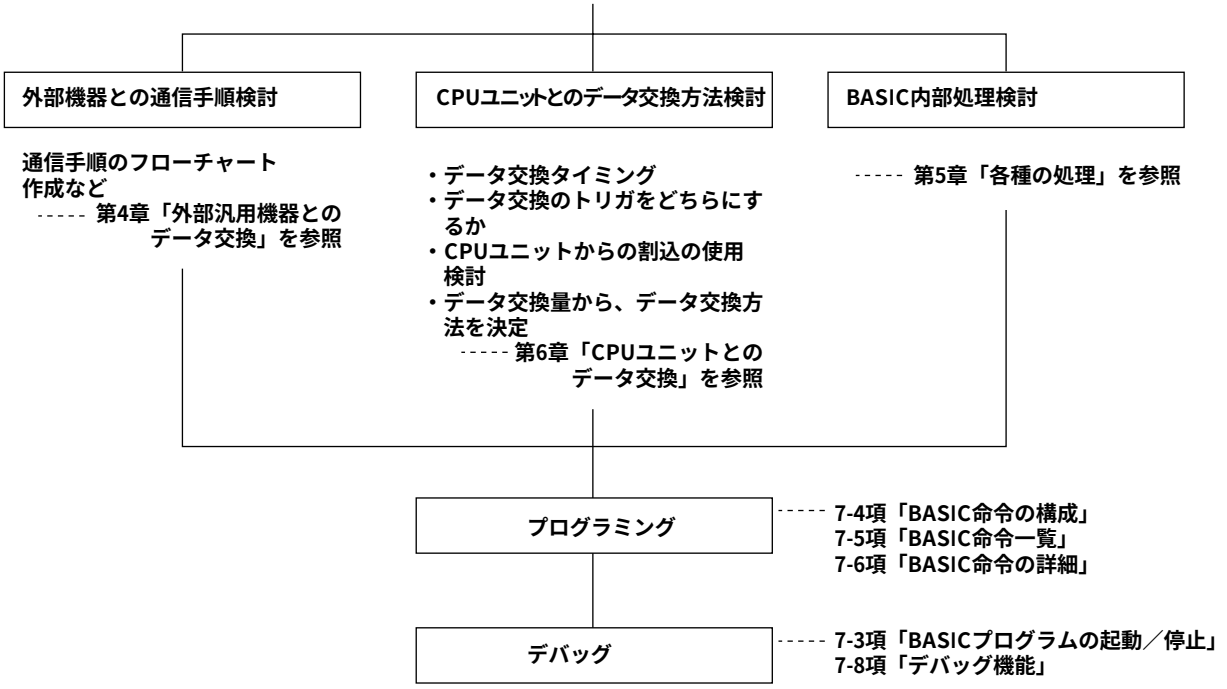
1−6　基本的な使用手順

■BASICプログラムの作成手順

Windows 95パソコンをターミナル(ハイパーターミナルのVT100エミュレーション機能)として使用する例を以下に示します。



■BASICプログラムの設計手順

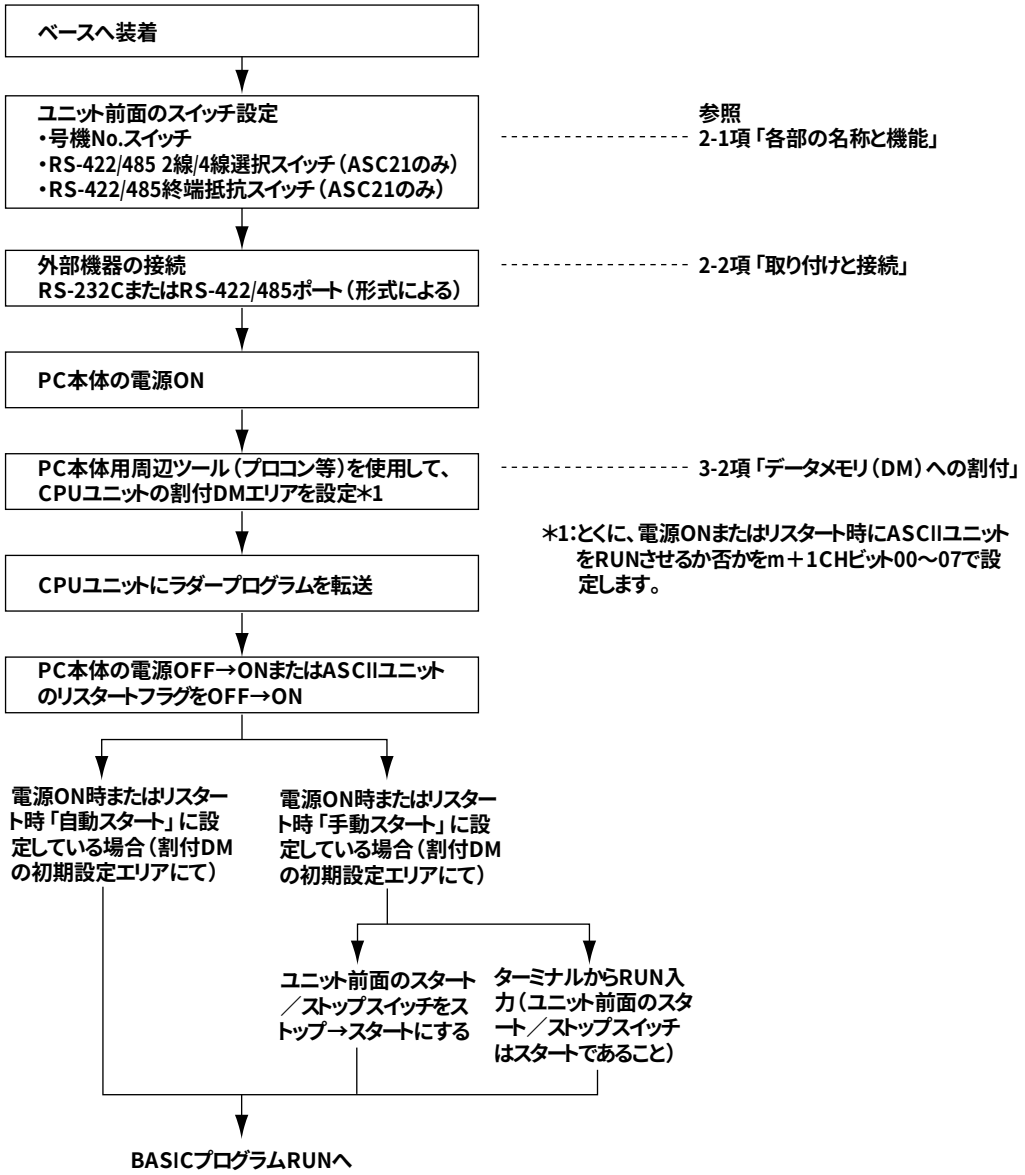


注1. プログラムをフラッシュROMへ保存する場合：
ターミナルからROMSAVEコマンドを入力

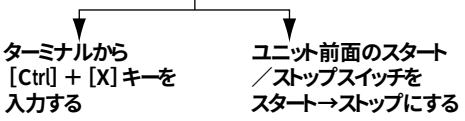
注2. 電源ONまたはリスタート時に、プログラムをフラッシュROMからユーザメモリへ自動読み出しをする場合：
DM割付のシステム設定エリアのmCHのビット08～15を5A（Hex）に設定

注3. ASC31でLOAD命令などを使用する場合は、ポート番号は ”#3 ”としてください。

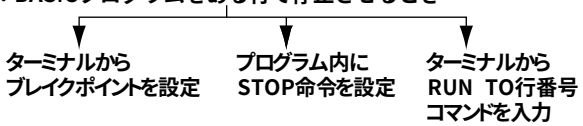
■外部機器との接続／運転



注1：BASICプログラムを停止させるとき



注2：BASICプログラムをある行で停止させるとき

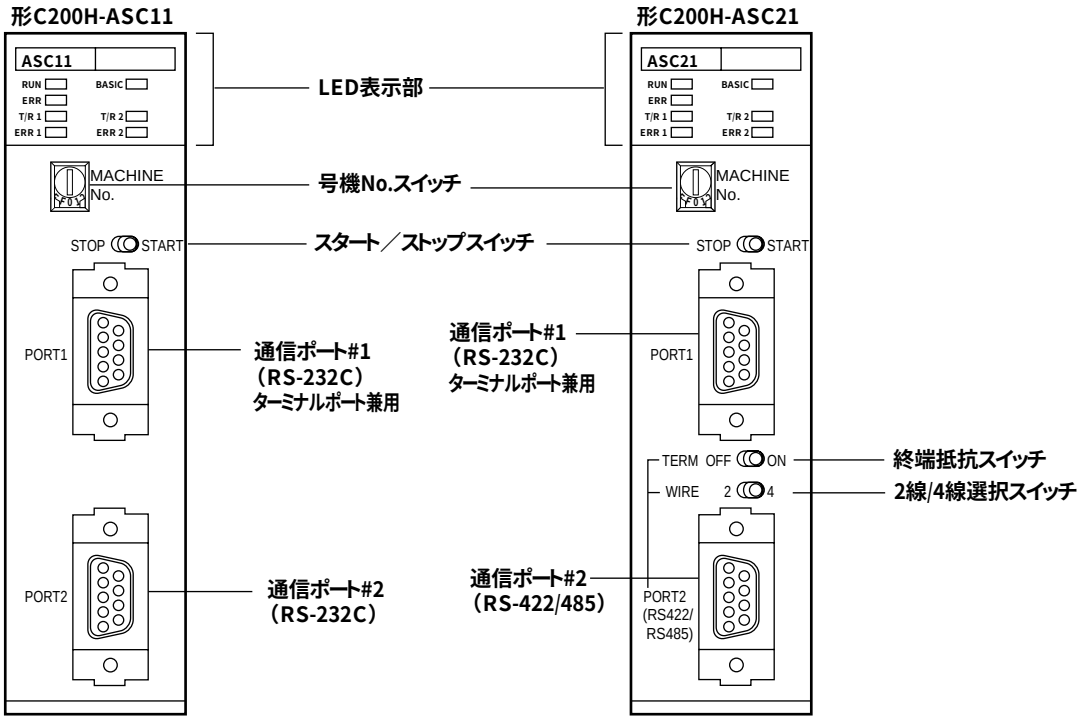


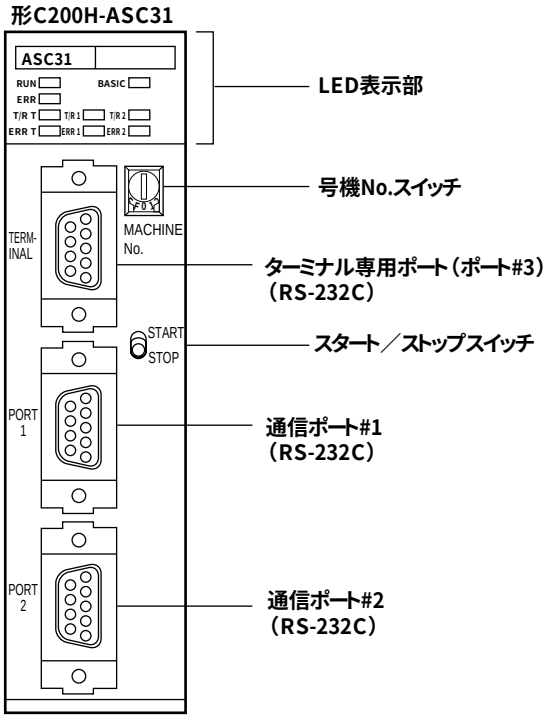
第2章
取り付けと接続

2−1　各部の名称と機能

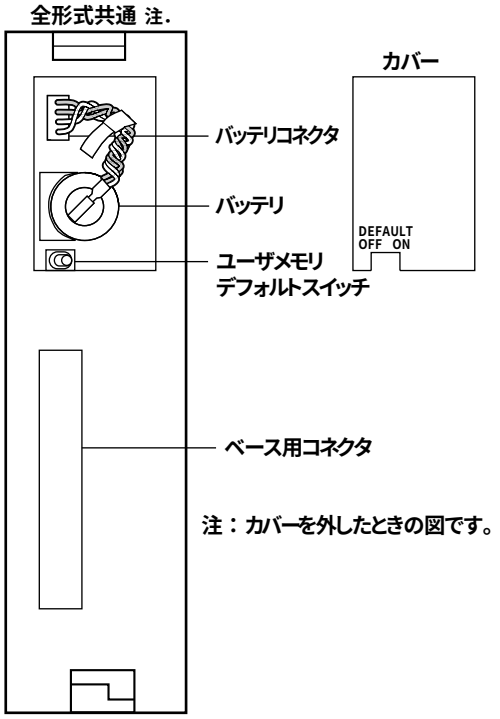
■各部の名称

●前面





●裏面



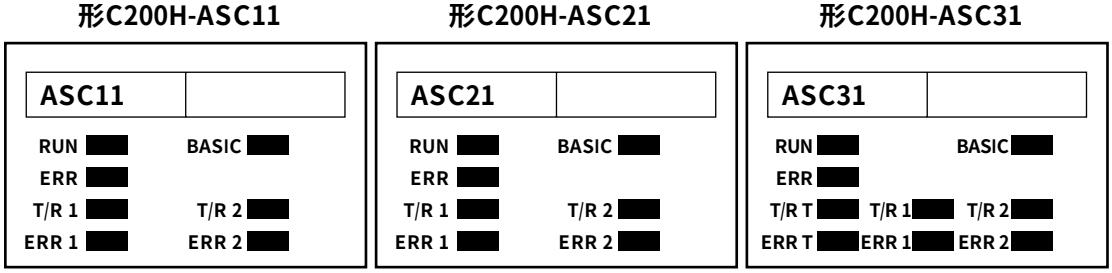
注：カバーを外したときの図です。

■前面

●LED表示の内容

LED	色	名　称	表　示	内　容	割付I/Oエリア
RUN	緑	ユニット動作	点灯	ASCIIユニットが正常動作中	―
			消灯	ASCIIユニットが異常（PC本体の電源がOFF、またはASCIIユニットのMPUが暴走）	―
BASIC	緑	システムモード	点灯	BASICプログラムが運転中	n+5のビット7
			ゆっくり点滅	INPUT/LINE INPUT/INPUT$命令待機中（入力待ち）、(L)PRINT命令待機中（送信可条件待ち）、またはコマンド入力待ち	n+5のビット7
			速く点滅	BASICプログラムの転送待機中（BASICプログラムの転送が始まると、点滅周期が不定となります。）	―
			消灯	BASICプログラム停止中、電源ON後の[Ctrl]+[X]入力待ち	―
ERR	赤	システム/BASIC/バッテリエラー	点灯	システムエラーまたはBASICエラー発生	n+5のビット2
			点滅	バッテリー電圧低下、またはバッテリー未挿入	n+5のビット6
T／R 1	黄	通信ポート#1データ伝送	点滅	通信ポート#1でデータ送信中または受信中	―
T／R 2	黄	通信ポート#2データ伝送	点滅	通信ポート#2でデータ送信中または受信中	―
T/R T	黄	ターミナルポートデータ伝送（ASC31のみ）	点滅	ターミナル専用ポート#3でデータ送信中または受信中（形C200H-ASC31のみ）	―
ERR 1	赤	通信ポート#1エラー	点灯	パリティエラー、受信バッファオーバーフローなどの受信エラーが通信ポート#1で発生	n+5のビット4
ERR 2	赤	通信ポート#2エラー	点灯	パリティエラー、受信バッファオーバーフローなどの受信エラーが通信ポート#2で発生	n+5のビット5
ERR T	赤	ターミナルポートエラー（ASC31のみ）	点灯	パリティエラー、受信バッファオーバーフローなどの受信エラーがターミナルポート#1で発生	n+5のビット3

注：PROGRAM MEMORY ERROR（BASICプログラムメモリの破壊：9-10ページ参照）が発生した場合は、すべてのERR LED（ERR、ERR 1、ERR 2、ERR T）が点滅します。



●スタート／ストップスイッチ

BASICプログラムをスタート／ストップさせます。RUN中にSTOPにすると、BASICプログラムは停止し、停止中にSTARTすると、BASICプログラムはRUN(先頭の行番号から)します。

初期設定とスイッチ位置の条件での、各操作時の動作を、以下に示します。このスイッチがSTOPの位置では、ターミナルからRUNコマンドを入力しても、また初期設定で自動スタートにしていても、プログラムは起動(RUN)しません。

STOP ◯◯ START

操　作		電源ONまたはリスタート時		停止中			RUN中	
				ターミナルからRUN入力		スイッチをSTOPからSTARTへ	RUN中に[Ctrl]＋[X]キーを押す	スイッチをSTARTからSTOPへ
スイッチ位置		START	STOP	START	STOP	STOP(→START)	START	START(→STOP)
初期設定(システム設定エリア)＊1	手動スタート	無関係	無関係	RUN	無関係	RUN	ブレイク	ブレイク
	自動スタート	RUN	無関係	RUN	無関係	RUN	ブレイク	ブレイク

＊1：DM割付エリアのm+1CH ビット00〜07で設定

●号機No．スイッチ

0〜Fまでの号機No.を設定します。他の高機能I/Oユニットと同じ番号に設定すると、「I/Oユニットオーバー」(運転停止異常)となり、動作しません。



スイッチ名	CPUユニット形式	
号機No．スイッチ	C200H-CPU21/22/23/31 C200HS-CPU01/21/31/03/23/33 C200HE-CPU11/32/42(-Z) C200HG -CPU33/43(-Z) C200HX-CPU34/44(-Z)	C200HG -CPU53/63(-Z) C200HX -CPU54/64(-Z) C200HX -CPU65-Z/85-Z CS1H-CPU66/65/64/63 CS1G-CPU45/44/43/42
号機No．設定範囲	0〜9	0〜F

●2線／4線選択スイッチ(形C200H-ASC21のみ)

形C200H-ASC21のRS-422A／485ポートに対して、2線／4線のいずれかを選択します。

WIRE 2 ◯◯ 4

スイッチ名	設　定	適　用
2線／4線選択スイッチ	2：2線 4：4線	形C200H-ASC21のみ

●終端抵抗スイッチ(形C200H-ASC21のみ)

形C200H-ASC21のRS-422A／485ポートの終端抵抗有無を選択します。

TERM OFF ◯◯ ON

スイッチ名	設　定	適　用
終端抵抗スイッチ(ASC21のみ)	ON：あり OFF：なし	形C200H-ASC21のみ

参　考　**終端抵抗の抵抗値は220Ωです。**

■裏面

●ユーザメモリデフォルトスイッチ(全機種)

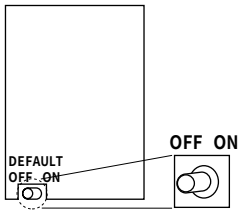
プログラムメモリ破壊が原因でASCIIユニットが異常状態にあるとき、このスイッチでプログラムメモリを強制的に初期化(プログラムが全く入っていない状態)し、正常状態に復帰させる場合に使用します。

⚠注意

通常はスイッチを必ずOFF(左)にしておいてください。

スイッチをON(右)にして電源をONすると、ユニット内にあるプログラムはすべて消去されてしまいます。

異常発生時の処置については「9-2 異常とその処置」を参照してください。



スイッチ名	設　定	適　用
ユーザメモリデフォルトスイッチ	ON(右)： プログラムメモリを強制的に初期化(デフォルト状態)する。 OFF(左)： プログラムメモリを初期化しない。	全機種

●バッテリ

バッテリは、ASCIIユニット内のBASICプログラム、トレースメモリの内容、および電源OFF中のリアルタイムクロックを保持します。

バッテリ電圧が低下すると、またはバッテリが接続されていないと、前面のERR LEDが点滅します。

品名(交換部品)	形　式
バッテリセット	形C200H-BAT09

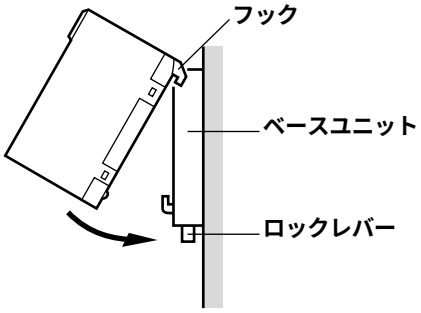
バッテリの交換方法については、9-5項「保守」を参照してください。

2−2　取り付けと接続

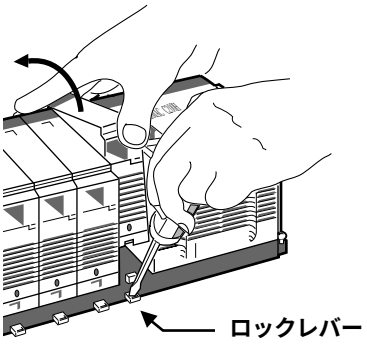
■取り付け方法

1　ASCIIユニットの上底部にあるツメを、ベースユニットにひっかけて取り付けます。

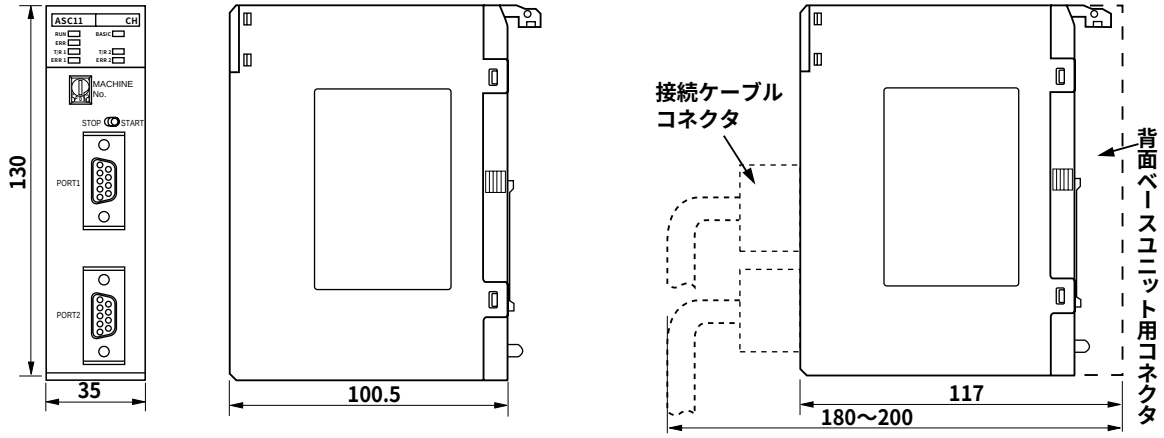
2　ASCIIユニットのコネクタをベースユニットのコネクタに確実に挿入してください。



参　考　**ASCIIユニットを取り外すときは、下図に示すように、ロックレバーをドライバなどで押しながら、取り外してください。**



■外形寸法



■取り付け可能位置と台数

PCシリーズ	CPUユニット形式	CPU装置およびI/O増設装置およびリモートI/O（SYSBUS）子局装置への合計装着可能台数*1	装着位置制限
SYSMACα	C200HE-CPU11/32/42(-Z) C200HG-CPU33/43(-Z) C200HX-CPU34/44(-Z)	10	なし
	C200HG-CPU53/63(-Z) C200HX-CPU54/64(-Z) C200HX-CPU65-Z/85-Z	16	
C200HS	C200HS-CPU01□/21□/31/03/23/33	10	
CSシリーズ	CS1H-CPU66/65/64/63 CS1G-CPU45/44/43/42	16	なし
C200H	C200H-CPU21/22/23/31	10	CPU装置の右側2スロットには装着不可

＊1：リモートI/O子局装置（SYSBUS）上の使用制限

リモートI/O子局装置（SYSBUS）上に装着するときは、次の条件となります。

グループ	A	B	C	D
ユニット種類	ASCIIユニット、 高速カウンタユニット、 位置制御ユニット (NC111/112/113/213)、 アナログ入出力ユニット、 IDセンサユニット ファジィ推論ユニット	多点入出力ユニット	温度センサユニット、 音声ユニット	位置制御ユニット (NC211) モーションコントロールユニット
1台のリモートI/Oで使用できる、各グループのときのみの装着可能台数	4ユニット	8ユニット	6ユニット	2ユニット
各グループを混在させて使用するときの装着可能台数	3A+B+2C+6D≦12、かつA+B+C+D≦8			

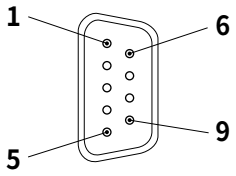
■外部機器との接続

RS-232CまたはRS-422A/485で接続します。インタフェース信号の動作／タイミングについては、第4章「外部汎用機器とのデータ交換」の章を参照してください。

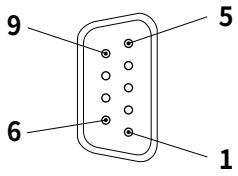
●RS-232Cでの接続（形C200H-ASC11／21／31）

標準付属の適合コネクタを使用して、外部機器のインタフェースに合わせて、各ピンを接続してください。

形C200H-ASC11/21



形C200H-ASC31



ピン番号	信号記号	信号名称	信号方向	備考
1	FG	保安用接地	－	－
2	SD	送信データ	ASCIIユニット→外部機器	－
3	RD	受信データ	外部機器→ASCIIユニット	－
4	RS（RTS）	送信要求信号	ASCIIユニット→外部機器	－
5	CS（CTS）	送信可能信号	外部機器→ASCIIユニット	－
6	－	使用しません	－	形C200H-ASC31のターミナル専用ポート#3のみ
	5V	5V出力	ASCIIユニット→外部機器	その他のポート
7	DR（DSR）	データセットレディ	外部機器→ASCIIユニット	－
8	ER（DTR）	データ端末レディ	ASCIIユニット→外部機器	－
9	SG	信号接地	－	－
口金	FG	フレーム接地	－	－

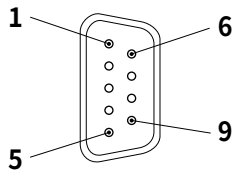
●RS-232Cインタフェース部仕様

項　　目	仕　　様	
電気的特性	EIA　RS-232C準拠	
同期方式	調歩同期式	
伝送速度	300／600／1200／2400／4800／9600／19200／38400ビット/s （ASC31のターミナル専用ポートは38400ビット/sを除く）	
伝送方式	ポイント・ツー・ポイント	
通信バッファ	送信：512バイト、受信：512バイト	
フロー制御	RS/CSフロー、Xon/Xoffフロー制御可能	
コネクタタイプ	Dサブ9ピン	
適合コネクタ （標準付属品）	プラグ：形XM2A-0901または同等品 コネクタフード：形XM2S-0911または同等品	
推奨ケーブル	UL2464 AWG28×5P IFS- RVV -SB（UL品） AWG28×5P IFVV-SB（非UL品）	藤倉電線製
	UL2464-SB（MA）5P×28AWG（7/0.127）（UL品） CO-MA-VV-SB 5P×28AWG（7/0.127）（非UL品）	日立電線製
線路長	15m以下	

注：ケーブルの作成のしかたについては、付録-5の「ケーブルの作成」の項を参照してください。

●RS-422A/485での接続（形C200H-ASC21）

標準付属の適合コネクタを使用して、外部機器のインタフェースに合わせて、各ピンを接続してください。



ピン番号	信号記号	信号名称	信号方向
1	SDA	送信データ A（−）	ASCIIユニット→外部機器
2	SDB	送信データ B（＋）	ASCIIユニット→外部機器
3	NC	未使用	−
4	NC	未使用	−
5	NC	未使用	−
6	RDA	受信データ A（−）	外部機器→ASCIIユニット
7	NC	未使用	−
8	RDB	受信データ B（＋）	外部機器→ASCIIユニット
9	NC	未使用	−
口金	FG	保安用接地	−

●RS-422A/485インタフェース部仕様

項　　目	仕　　様	
通信方式	半二重（送信完了から受信可能になるまで、1ms必要です）	
電気的特性	EIA　RS-422A/485準拠	
同期方式	調歩同期式	
伝送速度	300／600／1200／2400／4800／9600／19200／38400ビット／s	
伝送方式	ポイント・ツー・マルチポイント	
通信バッファ	送信：512バイト、受信：512バイト	
フロー制御	なし	
適合コネクタ（標準付属品）	プラグ：形XM2A-0901または同等品 コネクタフード：形XM2S-0911または同等品	
推奨ケーブル	UL2464 AWG28×5P IFS- RVV -SB（UL品） AWG28×5P IFVV-SB（非UL品）	藤倉電線製
	UL2464-SB（MA）5P×28AWG（7/0.127）（UL品） CO-MA-VV-SB 5P×28AWG（7/0.127）（非UL品）	日立電線製
線路長	500m以下	

注：ケーブルの作成のしかたについては、付録-5の「ケーブルの作成」の項を参照してください。

参　考　**2線/4線選択スイッチを2線に設定したときは、1、2番ピンまたは6、8番ピンのどちらかを使用します（1番ピンと6番ピン、2番ピンと8番ピンがそれぞれ短絡されます）。**

■ターミナルとの接続

●パソコンをターミナルとする場合

パソコン側のRS-232Cコネクタの配線を十分に確認の上、適合コネクタを使用して下記のピン配列に合致した接続ケーブルをご用意ください。



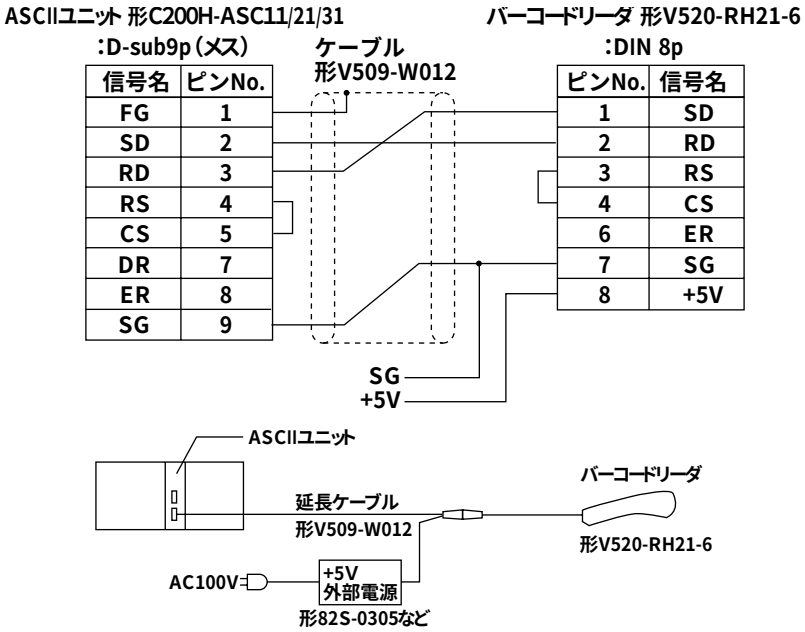
注1：上記形式のケーブルには、DR-ERの折り返しがありませんので、LOAD命令実行時などは、「DR信号をチェックしない」設定で使ってください(特に指定しなければ「チェックしない」になります)。

注2：上記形式のケーブルは、CS-RS、DR-ERがクロスしていませんので、フロー制御が必要なときは、Xon/Xoffフロー制御を使用してください。

■外部汎用機器との接続

●(RS-232Cによる)バーコードリーダとの接続

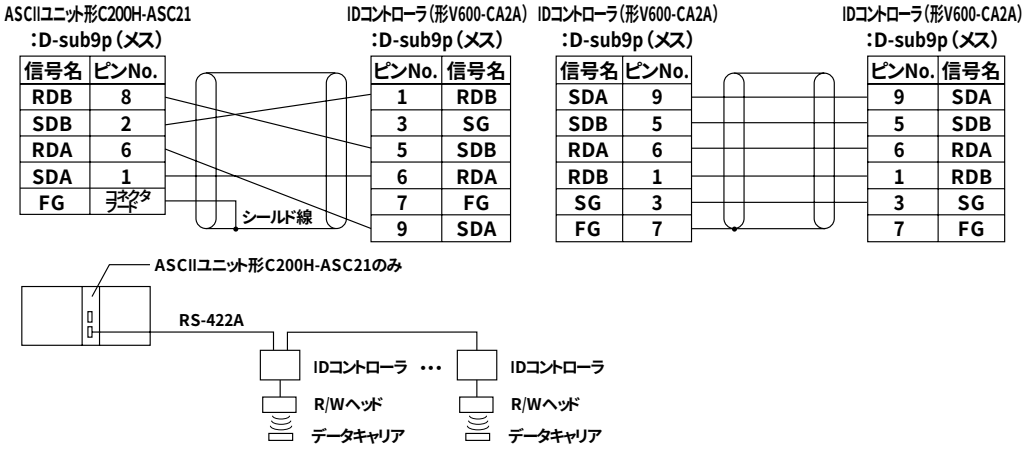
オムロン製形V500との接続



参 考 **外部機器にFGがある場合、誤動作防止のため外部機器側とASCIIユニット側の両方でシールド線の接地を行ってください。**

●（RS-422Aによる）IDコントローラとの接続（4線）

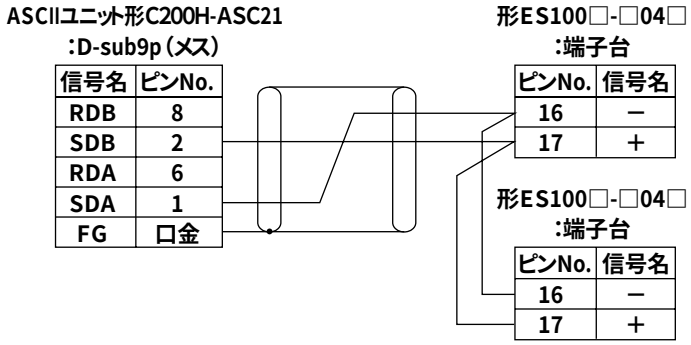
オムロン製形V600/620との接続



参 考
- **シールド線の接地は、誤動作防止のため、IDコントローラ側かASCIIユニット側のいずれか一方で行ってください。**
- **C200H-ASC21の2線/4線選択スイッチを4線に設定してください。**

●（RS-485による）デジタル調節計との接続（2線）

オムロン製形ES100Xとの接続



参 考
- **シールド線の接地は、誤動作防止のため、デジタル調節計側かASCIIユニット側のいずれか一方で行ってください。**
- **C200H-ASC21の2線/4線選択スイッチを2線に設定してください。**

第3章

リレー／DMの割付

3−1　内部補助リレーへの割付

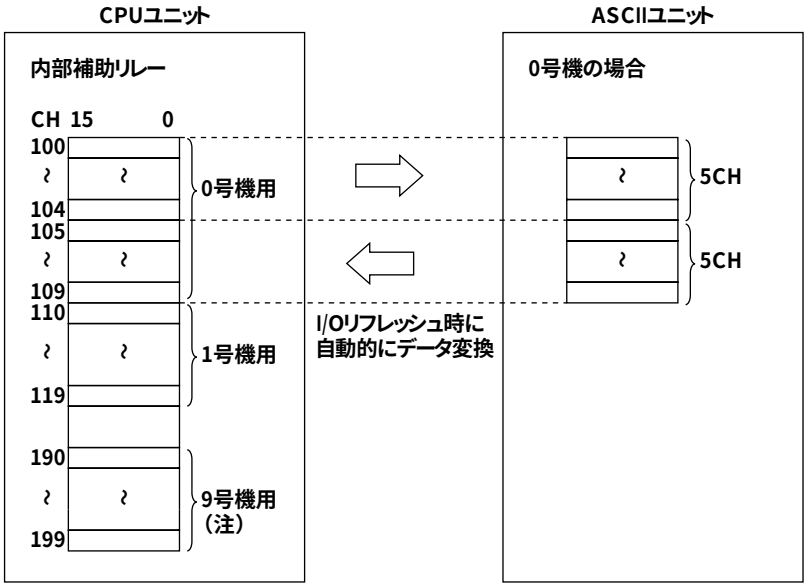
■内部補助リレーへの割付の概要

ASCIIユニットは、下図のように、CPUユニットの内部補助リレーの高機能I/Oエリアに、1台当たり10チャネルが割り付けられます。

どの10CHが割り付けられるかは、号機No.によります。号機No.は、ASCIIユニット前面のロータリスイッチで設定します。

割り付けられた10CHのデータは、CPUユニットのI/Oリフレッシュ毎に、CPUユニットとASCIIユニット間で自動的にデータ交換されます。

●SYSMACα、C200HS、C200Hシリーズ



（注）SYSMACαシリーズのCPUユニット C200HG-CPU53/63（-Z）、C200HX-CPU54/64（-Z）、C200HX-CPU65-Z/85-Zの場合、さらに400～459CHが割付可能（最大16号機）です。

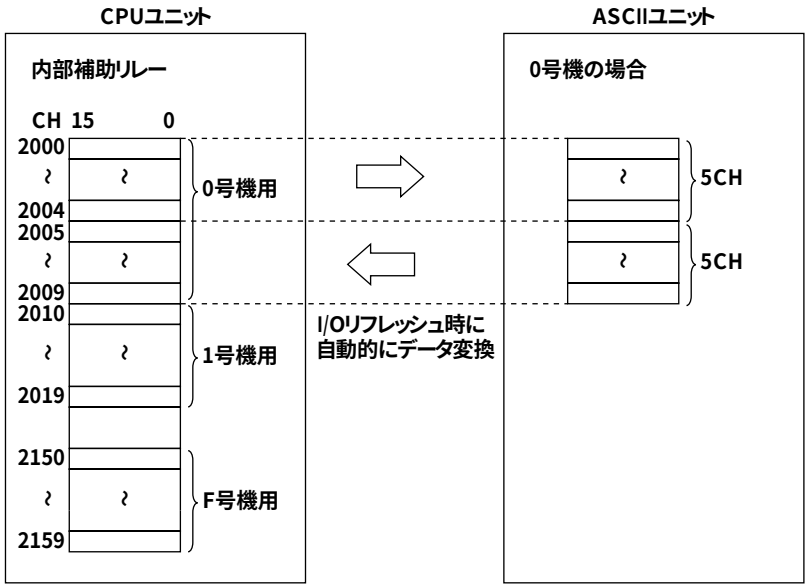
●高機能I/Oエリア割付

・SYSMACα、C200HS、C200Hシリーズ

号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア
0	100～109CH	4	140～149CH	8	180～189CH	C	420～429CH*
1	110～119CH	5	150～159CH	9	190～199CH	D	430～439CH*
2	120～129CH	6	160～169CH	A	400～409CH*	E	440～449CH*
3	130～139CH	7	170～179CH	B	410～419CH*	F	450～459CH*

＊：SYSMACαの内、C200HG-CPU53/63（-Z）、C200HX-CPU54/64（-Z）、C200HX-CPU65-Z/85-Zのみ可能

●CSシリーズ



●高機能I/Oエリア割付

・CSシリーズ

号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア
0	2000～2009CH	4	2040～2049CH	8	2080～2089CH	C	2120～2129CH
1	2010～2019CH	5	2050～2059CH	9	2090～2099CH	D	2130～2139CH
2	2020～2029CH	6	2060～2069CH	A	2100～2109CH	E	2140～2149CH
3	2030～2039CH	7	2070～2079CH	B	2110～2119CH	F	2150～2159CH

●内部補助リレーへの割付一覧

I/Oリフレッシュ時に自動的にデータ交換される内容は次のとおりです。

・SYSMACα、C200HS、200Hシリーズ

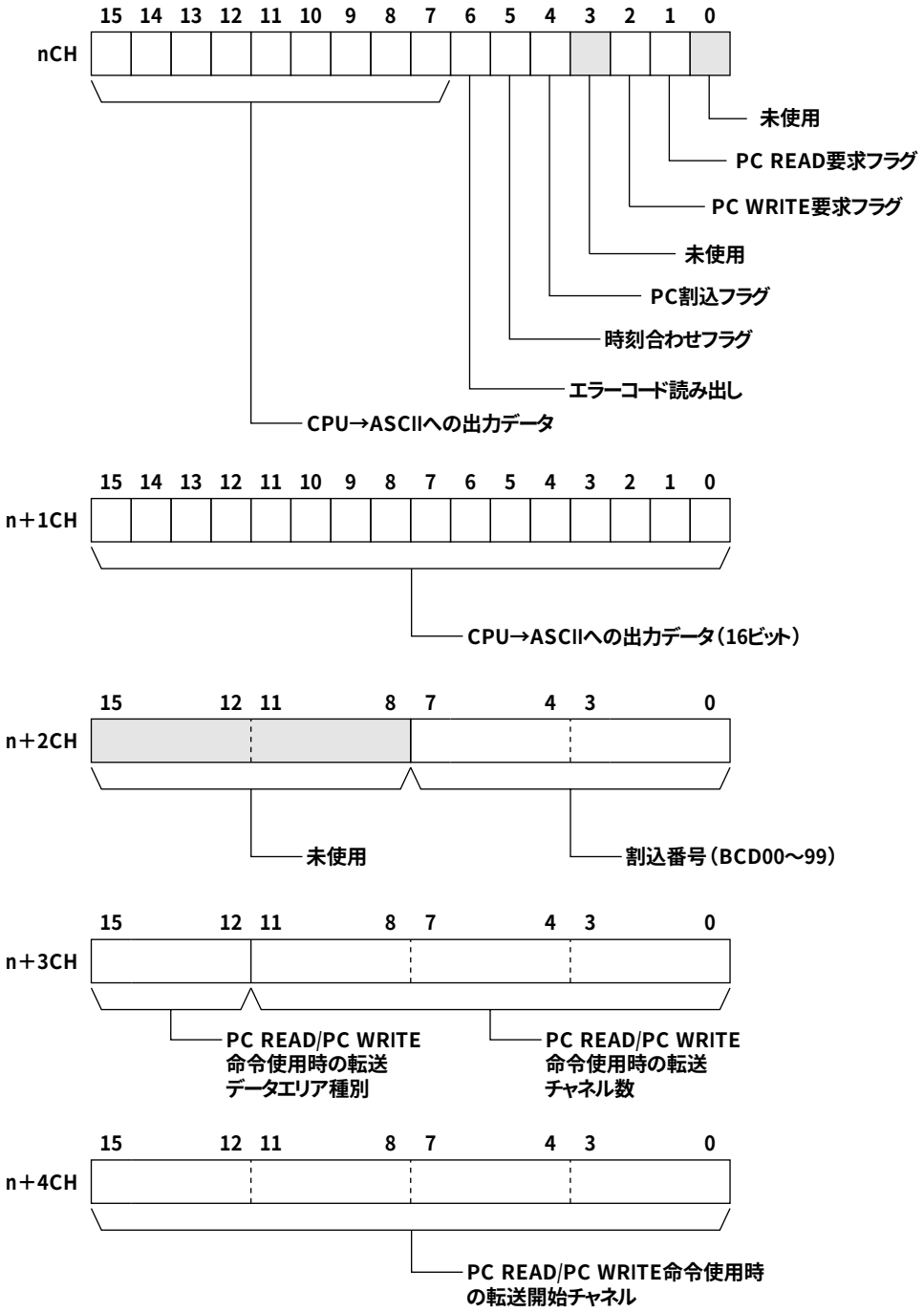
n＝100＋10×号機No.　　（号機No.が0～9の場合）

n＝400＋10×（号機No.－10）　　（号機No.がA（10）～F（15）の場合）

・CSシリーズ

n＝2000＋10×号機No.　　（号機No.は0～F（15））

●CPUユニット→ASCIIユニットへ



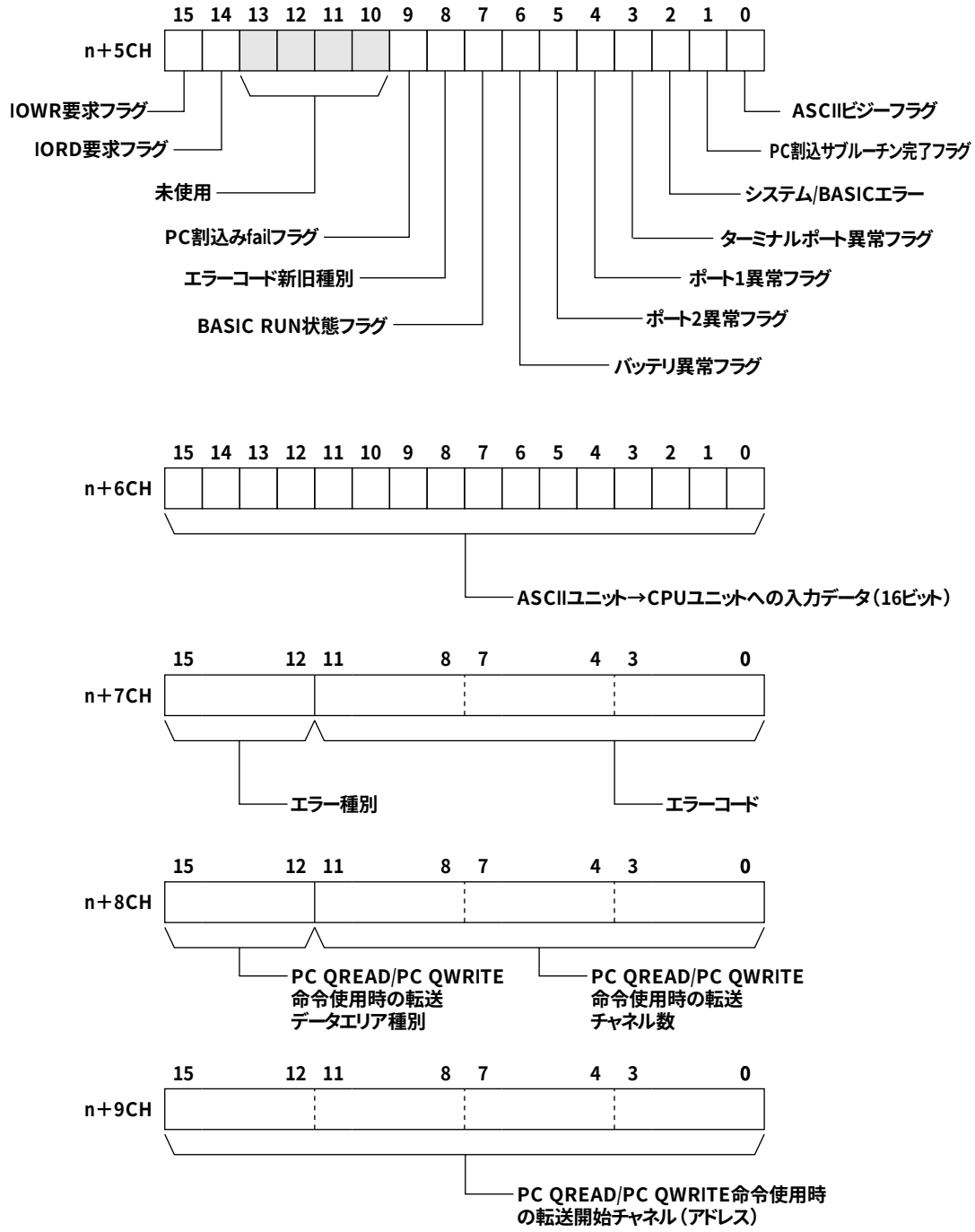
方　向	チャネル	ビット	名　称	機　能	使い方
出力 （CPUユニット ↓ ASCIIユニット）	n	00	未使用	—	—
		01	WRITE （PC READ 要求） フラグ	1：PC READ命令の実行条件成立（転送CH数、転送開始CHにしたがい、PC→ASCIIへデータが転送される） 0：PC READ命令実行待ち	CPUユニットのユーザプログラムにより、このビットをONにし、ASCIIユニット側のPC READ命令によりデータ転送を実行させます。
		02	READ（PC WRITE要求） フラグ	1：PC WRITE命令の実行条件成立（転送CH数、転送開始CHにしたがい、ASCII→PCへデータが転送される） 0：PC WRITE命令の実行待ち	CPUユニットのユーザログラムにより、このビットをONとし、ASCIIユニットのPC WRITE命令によりデータ転送を実行させます。
		03	未使用	—	—
		04	PC割込フラグ	0→1：ASCIIユニットに対して割込をかける 立上り（0→1）時に、ON PC GOSUB命令の割込サブルーチンへ分岐させる。	CPUユニットのユーザプログラムにより、このビットをOFF→ONとし、ASCIIユニットへ割込をかけます。 n+2CHビット0～7の割込番号を同時にセットしておくことが必要です。
		05	時刻合わせフラグ	0→1：立上がり（0→1）時にASCIIユニットのRTCをCPUユニットのRTCに合わせる。	—
		06	エラーコード読み出し	0→1、1→0：n+7CHのエラーコードを読み出す。	このビットをトグル（0→1、1→0）することで、ASCIIユニット内の異常履歴エリアから、1つ古い（直前の）エラーコードが読み出され n+7CHにセットされます。 このエラーコード読出しでASCIIユニットのエラー状態は解除されません。
		07～15	出力データ	CPUユニットからASCIIユニットへ渡されるデータ。ASCIIユニットはBASICプログラムのPC GET命令で読み出せます。	CPUユニットのユーザプログラム内のMOV命令等で、ここへデータを書き込む。ASCIIユニット側ではPC GET命令で、このデータが読み出せます。

3－1　内部補助リレーへの割付

方　向	チャネル	ビット	名　称	機　能	使い方
出力 （CPUユニット ↓ ASCIIユニット）	n+1	00～15	出力データ	CPUユニットからASCIIユニットへ渡されるデータ。ASCIIユニットはBASICプログラムのPC GET命令で読み出せます。	CPUユニットのユーザプログラム内のMOV命令等で、ここへデータを書き込む。ASCIIユニット側ではPC GET命令で、この16ビットデータが読み出せます。
	n+2	00～07	割込番号	00～99（BCD 2桁）	ON PC GOSUB命令の割込番号と、この割込番号が一致していて、かつPC割込フラグがOFF→ONでCPUユニットからASCIIユニットへ割込が通知されると、対応する割込サブルーチンを実行します。
		08～15	未使用	－	－
	n+3	00～11	転送チャネル数	ASCIIユニットのPC READ/PC WRITE命令実行時の転送CH数（1～255）	CPUユニットのユーザプログラム内のMOV命令で、ここへデータを書き込みます。WRITEフラグがONしたとき、直後のI/Oリフレッシュ時にこのデータがASCIIユニットのPC READ/PC WRITE命令に渡されます。
		12～15	転送データエリア種別	ASCIIユニットのPC READ／PC WRITE命令で読み書きするI/Oメモリのエリア種別を、下記コードで指定します。	
	n+4	00～15	転送開始チャネル（アドレス）	ASCIIユニットのPC READ／PC WRITE命令実行時の転送開始CH（CPUユニットのI/Oメモリのアドレス）	

No.	エリアの種別	
	SYSMACα、C200HS、C200Hシリーズ	CSシリーズ
0	データメモリ（DM）	
1	入出力リレー・内部補助リレー	入出力リレー（CIO）
2	リンクリレー（LR）	データリンクリレー
3	保持リレー（HR）	保持リレー1（H000～H099）
4	補助記憶リレー（AR）	保持リレー2（H100CH以降）
5	拡張データメモリ（EM）バンクNo.0のみ	
6	タイマ／カウンタ（T/C）	タイマ（T）

●ASCIIユニット→CPUユニットへ



方　向	チャネル	ビット	名 称	機　能	使い方
入力 （ASCIIユニット ↓ CPUユニット）	n+5	00	ASCIIビジーフラグ	1：ASCIIユニットがCPUユニットとデータ転送中、またはDM設定にエラーがあったとき、またはROMSAVEコマンドにより、BASICプログラムをフラッシュROMへ転送中 0：上記転送処理中でない	このビットがONの間は、CPUユニットのラダープログラムが次の転送を行わないように、CPUユニットに伝えるためのフラグです。 このフラグのb接点をCPUユニットのPC READ／PC WRITE要求フラグのOUT命令の入力条件やIOWR（＃00□□）命令の入力条件に使用します。
		01	PC割込サブルーチン完了フラグ	PCからの割込による割込サブルーチンが終了時、トグルします。	このビットがトグルするまで、サブルーチン結果待ちです。 参考：トグルの検出には、このフラグに対するDIFU/DIFD命令のORをとります。割込サブルーチン内のPC EPUT命令を受けて、IORD命令（#00□□）を実行するときなどに使用します。
		02	システム/BASICエラー	1：システムエラーまたはBASICエラー 0：正常	－
		03	ターミナルポート異常フラグ	1：ターミナルポートにて通信エラー 0：正常	これらのビットは、ERC命令実行でクリアされます。
		04	ポート1異常フラグ	1：ポート1が伝送異常（パリティエラー、受信バッファオーバーフローなど） 0：正常	
		05	ポート2異常フラグ	1：ポート2が伝送異常（パリティエラー、受信バッファオーバーフローなど） 0：正常	
		06	バッテリ異常フラグ	1：バッテリ電圧低下または未接続 0：正常	－
		07	BASIC RUN状態フラグ	1：BASICプログラム実行中 0：BASICプログラム停止中	－
		08	エラーコード新旧種別	1：新 0：旧	読み出しているエラーコードのエラーが、今回のRUNよりも前に発生していたものかを区別します。
		09	PC割込failフラグ	1：PC割込が失敗 0：正常	このフラグは、ASCIIユニットがPC OFF（割込禁止）状態などにより、PC割込が失敗したとき、PC割込フラグの立上がりの次のサイクルの間1（ON）となり、その次のサイクルでOFFとなります。
		10～13	未使用	－	－
		14	IORD要求フラグ	1：ASCIIユニットからCPUユニットへIORD（#FD00）命令実行要求あり（PC QWRITE 命令実行時に自動的に1（ON）となり、 CPUユニットのIORD（#FD00）命令によるデータ転送後の次のI/Oリフレッシュで、自動的に0（OFF）へ戻る） 0:要求なし	PC QWRITE命令実行時に、自動的にこのビットが1（ON）となることで、CPUユニットに対して、IORD（#FD00）命令実行を要求します。

<table>
<tr><th>方　向</th><th>チャネル</th><th>ビット</th><th>名　称</th><th>機　能</th><th>使い方</th></tr>
<tr><td rowspan="6">入力
(ASCIIユニット
↓
CPUユニット)</td><td>n+5</td><td>15</td><td>IOWR要求フラグ</td><td>1：ASCIIユニットからCPUユニットへIOWR(#FD00)命令実行要求あり（PC QREAD命令実行時に自動的に1(ON)となり、 CPUユニットのIOWR(#FD00)命令によるデータ転送後の次のI/Oリフレッシュで、自動的に0(OFF)へ戻る）
0：要求なし</td><td>PC QREAD命令実行時に、自動的にこのビットが1(ON)となることで、CPUユニットに対して、IOWR(#FD00)命令実行を要求します。</td></tr>
<tr><td>n+6</td><td>00～15</td><td>入力データ</td><td>ASCIIユニットのPC PUT命令で、ASCIIユニットからCPUユニットへ渡されるデータ(16ビット)</td><td>ASCIIユニットのPC PUT命令で割付I/Oデータエリアにデータを書き込みます。次のI/Oリフレッシュ後、CPUユニットのMOVまたはLD命令等で、その16ビットまたは8ビットまたは1ビットデータが読み込まれます。</td></tr>
<tr><td rowspan="2">n+7</td><td>00～11</td><td>エラーコード</td><td>エラーコード</td><td rowspan="2">CPUユニットからnCHビット06をトグル(1→0、0→1)することで、1つ古い(直前の)エラーコードおよびエラー種別をASCIIユニット内のエラーコード(異常履歴)テーブルから読み出し、CPUユニットに表示します。</td></tr>
<tr><td>12～15</td><td>エラー種別</td><td>B：BASICエラー
0：システムエラー
1：ポート1通信エラー
2：ポート2通信エラー
3：ターミナルポート通信エラー(ASC31のみ)</td></tr>
<tr><td rowspan="3">n+8</td><td>00～11</td><td>転送チャネル数</td><td>ASCIIユニットのPC QREAD/PC QWRITE命令実行時の転送CH数がセットされる。(1～128)</td><td rowspan="3">ASCIIユニットはPC QREAD／PC QWRITE命令を実行すると、これらのチャネルにデータを書き込みます。同時にn+5CH ビット14/15にIORD／IOWR要求フラグをセットします。CPUユニットは、LD命令等の命令を使ってその要求フラグがセットされたことを知ると、IOWR／IORD(#FD00)命令において指定されたエリアを読み書きします。(注)</td></tr>
<tr><td>12～15</td><td>転送データエリア種別</td><td>ASCIIユニットのPC QREAD／PC QWRITE命令で読み書きするI/Oメモリのエリア種別がセットされる。</td></tr>
<tr><td>n+9</td><td>00～15</td><td>転送開始チャネル(アドレス)</td><td>ASCIIユニットのPC QREAD／PC QWRITE命令実行時の転送開始CH(CPUユニットのI/Oメモリのアドレス)がセットされる。</td></tr>
</table>

No.	エリアの種別	
	SYSMACα、C200HS、C200Hシリーズ	CSシリーズ
0	データメモリ(DM)	
1	入出力リレー・内部補助リレー	入出力リレー(CIO)
2	リンクリレー(LR)	データリンクリレー
3	保持リレー(HR)	保持リレー1(H000～H099)
4	補助記憶リレー(AR)	保持リレー2(H100CH以降)
6	タイマ／カウンタ(T/C)	タイマ(T)

注：PC QREAD/PC QWRITE命令の実行によってセットされるこれらのチャネルのデータは、CPUユニットにとってのモニタ用です。実際のデータ転送は、IOWR/IORD(#FD00)命令のオペランドで指定されたエリアが転送されます。とくにCSシリーズの場合、PC QREAD/PC QWRITE命令によって指定できるエリア範囲と、IOWR/IORD(#FD00)命令によって指定できるエリア範囲が異なりますので、PC QREAD/PC QWRITE命令では指定できないエリアの場合、PC QREAD/PC QWRITE命令にはダミーのエリアを指定することが必要となります。詳細については、6-3項を参照してください。

3−2　データメモリ(DM)への割付

■データメモリ(DM)への共有メモリの割付の概要

CPUユニットのデータメモリエリアの100CHが、1台のASCIIユニット内の共有メモリに割付けられます。

データメモリエリアのどの100CHが割付られるかは、号機No.により決まります。

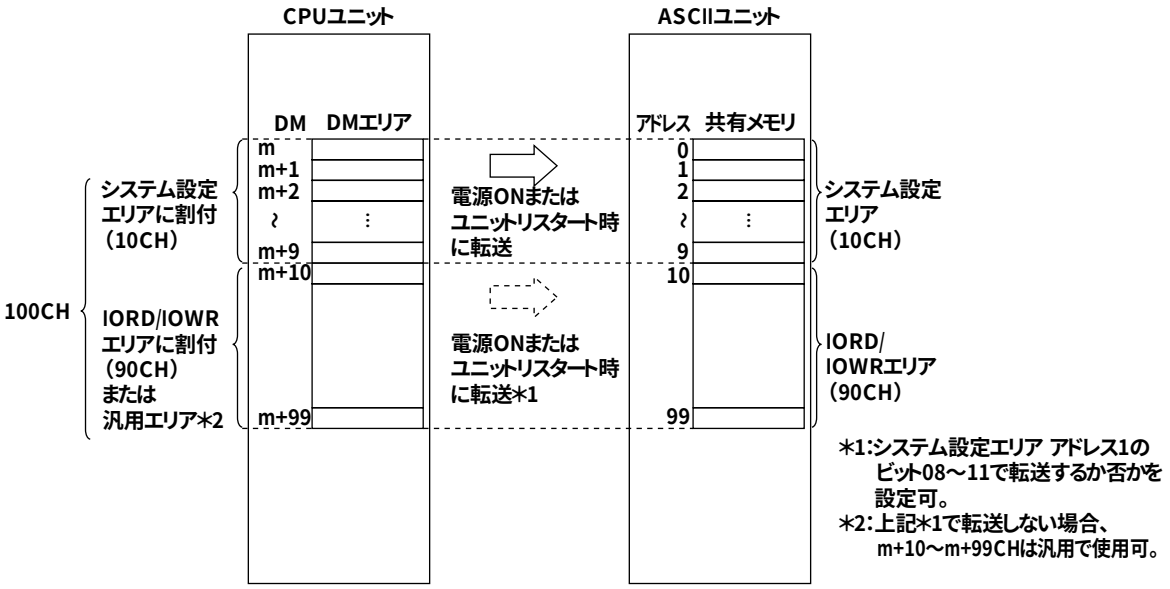
ASCIIユニット内の共有メモリ(100CH)は、次の2つの領域に分かれます。

1)システム設定エリア(10CH):アドレス0〜9

2)IORD/IOWRエリア(90CH):アドレス10〜99

なお、ASCIIユニット内の共有メモリは、バッテリバックアップされません。したがって、電源をOFFすると、初期値に戻ります。CPUユニットからデータを設定するときは、電源ON時またはリスタート時に、その都度、データを転送する必要があります。

注:m:DM割付エリアの先頭チャネルアドレス



●DMエリア割付

・SYSMACα、C200HS、C200Hシリーズ

号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア
0	DM1000〜1099	6	DM1600〜1699	A*1	DM2000〜2099
1	DM1100〜1199	7	DM1700〜1799	B*1	DM2100〜2199
2	DM1200〜1299	8	DM1800〜1899	C*1	DM2200〜2299
3	DM1300〜1399	9	DM1900〜1999	D*1	DM2300〜2399
4	DM1400〜1499			E*1	DM2400〜2499
5	DM1500〜1599			F*1	DM2500〜2599

*1:SYSMACαシリーズのCPUユニット C200HG-CPU53/63(-Z)、C200HX-CPU54/64(-Z)、C200HX-CPU65-Z/85-Zの場合のみ

・CSシリーズ

号機No.	割付エリア	号機No.	割付エリア	号機No.	割付エリア
0	DM20000～20099	6	DM20600～20699	A	DM21000～21099
1	DM20100～20199	7	DM20700～20799	B	DM21100～21199
2	DM20200～20299	8	DM20800～20899	C	DM21200～21299
3	DM20300～20399	9	DM20900～20999	D	DM21300～21399
4	DM20400～20499			E	DM21400～21499
5	DM20500～20599			F	DM21500～21599

1)システム設定エリア(10CH)

共有メモリ100CHの内の先頭の10CH(アドレス0～9)は、ASCIIユニットのシステム設定エリアです。

次のような内容から成ります。詳細は、3-12ページを参照してください。

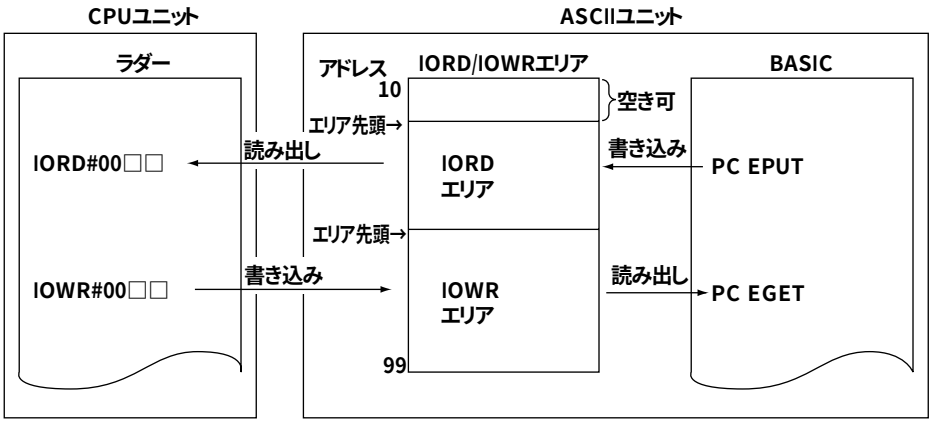
・ASCIIユニットの各通信ポートの通信条件
・起動時にアクティブになるプログラムNo.
・残り90CH(アドレス10～99)のIOWRエリア／IORDエリアの区分け
・電源ON時またはリスタート時、CPUユニットからの残り90CH(DMm+10～m+99CH)のASCIIユニットへの転送有無など

電源ON時またはユニットリスタート時に、CPUユニット内のDMエリア(DMm～m+9CH)からASCIIユニットへ転送することができます。

2)IORD／IOWRエリア(90CH)

共有メモリ100CHの内の残りの90CH(アドレス10～99)は、CPUユニットからのIORD／IOWR命令、およびASCIIユニットからのPC EGET／EPUT命令によって、読み／書きできるASCIIユニット内の汎用データメモリエリア(「IORD／IOWRエリア」と呼びます)です。

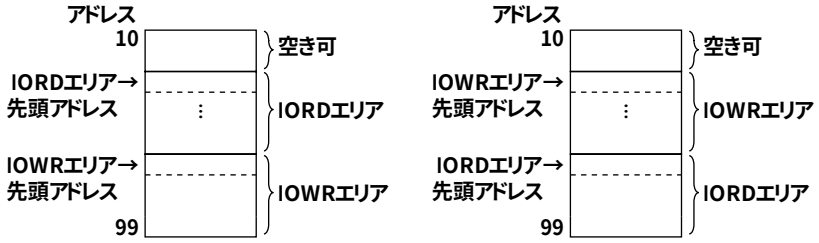
この90CHの内、どこからどこまでをCPUユニットがIORD命令によって読み出す、かつASCIIユニットがPC EPUT命令によって書き込むエリア(「IORDエリア」)とし、どこからどこまでをIOWR命令によって書き込む、かつASCIIユニットがPC EGET命令によって読み出すエリア(「IOWRエリア」)とするかを指定できます。この指定は、1)のシステム設定エリア(アドレス6/7､ビット00～07)によって行います。



また、CPUユニットのデータメモリ(DM)の割付けられた残りの90CH(DMm+10～m+99CH)を電源ON時またはリスタート時に、ASCIIユニットのIORD／IOWRエリア(アドレス10～99)に転送するかしないかを指定できます。この指定は、1)のシステム設定エリア(アドレス1､ビット08～11)によります。転送しない場合は、CPUユニット内のDMエリア(DMm+10CH～m+99CH)は、汎用として使用できます。

注：IORD/IOWRエリアの範囲設定について

IORDエリアの先頭アドレス、およびIOWRエリアの先頭アドレスを指定します。IORDエリアとIOWRエリアの順は、どちらでも可能です。先頭アドレス10から、IORDまたはIOWRエリア先頭アドレスまでは、空きとすることが可能です。



■データメモリ(DM)の共有メモリへの割付／共有メモリ内容一覧

m：CPUユニットのDM割付エリアの先頭チャネルアドレス
SYSMACα、C200HS、C200Hシリーズ：m＝DM1000＋100×号機No.
CSシリーズ：m＝D20000＋100×号機No.

共有メモリアドレス	DMチャネル	ビット	名　称	機　能	設定値（数値は16進）	ASCIIユニットの初期値（数値は16進）
0	m	00～07	プログラムNo.	起動時のプログラムNo.を指定します。	00:No.1　03:No.3 01:No.1　04:No.4 02:No.2	00
		08～15	フラッシュROMからの自動転送	電源ONまたはリスタート時に、フラッシュROMからユーザメモリエリアへプログラムを読み込むかどうかを指定します。	00：ユーザメモリエリアモード 5A：フラッシュROMからユーザメモリエリアへ読み込む	00
1	m+1	00～07	スタートモード	電源ONまたはリスタート時にプログラムを自動的にRUNさせるか否かを指定します。 注：手動スタート（00）を指定したときは、ASCIIユニット前面のスタート／ストップスイッチ、またはターミナルからのRUNコマンドで、スタートさせます。スタート／ストップスイッチとの関係は、2-4ページを参照してください。	00：手動スタート 5A：自動スタート	00
		08～11	IORD/IOWRエリアへの転送	電源ON時またはリスタート時に、CPUユニットのDMm+10～m+99の90CHをASCIIユニットの共有メモリ（IORD／IOWRエリア）へ転送するか否かを指定します。	0：転送しない 1：転送する	0
		12～15	システム予約	ー	0としてください。	0
2	m+2	00～07	ポート#1通信速度	通信ポート#1の通信速度を指定します。	00：9600　05：4800 01：300　06：9600 02：600　07：19200 03：1200　08：38400 04：2400	00
		08～15	ポート#1DMA送信（形C200H-ASC31のみ）	ポート#1でのデータ送信をDMA送信で行います。	00：DMA送信しない 5A：DMA送信する	00
3	m+3	00～07	ポート#2通信速度	通信ポート#2の通信速度を指定します。	00：9600　05：4800 01：300　06：9600 02：600　07：19200 03：1200　08：38400 04：2400	00
		08～15	ポート#2 DMA送信	ポート#2でのデータ送信をDMA送信で行います。	00：DMA送信しない 5A：DMA送信する	00
4	m+4	00～07	ターミナル専用ポート#3通信速度（形C200H-ASC31のみ）	ターミナル専用ポート#3の通信速度を指定します。	00：9600　04：2400 01：300　05：4800 02：600　06：9600 03：1200　07：19200	00
		08～15	1サイクル当たり転送チャネル数（SYSMACα、CSシリーズのみ）	CPUユニットの1サイクル当たりのPC READ／PC WRITE命令（@付き含む）による最大転送チャネル数を指定します。	00：最大20チャネル 5A：最大127チャネル 注：5AはPCがSYSMACα、CSシリーズのみ可能（ただし、リモートI/O上では使用不可）	00

共有メモリアドレス	DMチャネル	ビット	名　称	機　能	設定値（数値は16進）	ASCIIユニットの初期値（数値は16進）
5	m+5	00～07	未使用	－	－	00
		08～11	スクリーンサイズ：幅	エディットモード時にカーソル移動に関係します。	0：80文字　2：132文字 1：40文字	0
		12～15	ターミナルエミュレーション	キー操作などのターミナルエミュレーションモードを設定します。	0:VT100モード 1:FIT10モード	0
6	m+6	00～07	IOWRエリア先頭アドレス	アドレス10～99の共有エリアの内、IOWRエリアの先頭アドレスを指定します。	00：IOWRエリアなし 10～99：IOWRエリアの先頭アドレス	00
		08～15	未使用	－	－	
7	m+7	00～07	IORDエリア先頭アドレス	アドレス10～99の共有エリアの内、IORDエリアの先頭アドレスを指定します。	00：IORDエリアなし 10～99：IORDエリアの先頭アドレス	00
		08～15	未使用	－	－	
8	m+8	00～15	未使用	－	－	
9	m+9	00～15	未使用	－	－	
10～99	m+10 ～ m+99	00～15	IORDエリアまたはIOWRエリア（アドレス6、7で、各先頭アドレスを指定）			00

お願い　設定値欄以外の値は設定しないでください。

参　考　1．プログラムをRUNさせたときに、ターミナルポートに出力される ’Please wait compiling... ’のメッセージは、スタートモードを自動スタートに設定した場合には、電源OFF/ON、START/STOPスイッチ操作では出力されません。

2．DM設定の内容にエラーがあるとERR LEDが点灯します。このとき、ターミナルを接続することはできません。プロコンでエラーコードを確認し、再度DM設定を正しいものに修正した後、再度電源OFF/ON、またはリスタートを行ってください。

第4章
外部汎用機器とのデータ交換

4−1　外部汎用機器とのデータ交換

本項では次の操作について説明します。

用　途		使用命令・関数
通信ポートのオープン		OPEN 命令
データの送信		PRINT命令 （LPRINT命令はOPEN命令に関係なくデバイス:LPRTとなる）
データの受信	CR区切での受信時	INPUT命令 （LINE INPUT命令は文字変数1つにのみすべて受信）
	指定文字数の受信時	INPUT$関数
通信ポートのクローズ		CLOSE 命令
通信エラーの処理		ON ERROR命令、ERC命令、ERR命令、RESUME命令

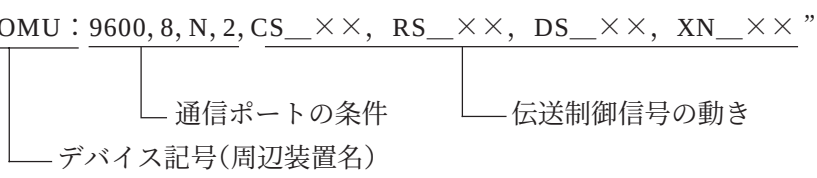
■通信ポートのオープン

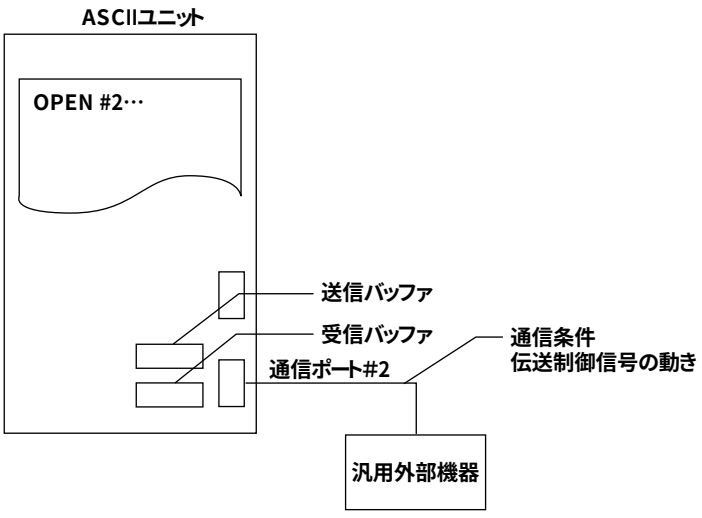
●OPEN 命令

OPEN 命令によります。

OPEN 命令で、デバイス記号(周辺装置名)、通信ポートの通信条件、伝送制御信号の動きを指定します。

例:OPEN #2,“COMU：9600, 8, N, 2, CS__××, RS__××, DS__××, XN__××”





●OPEN 命令による指定内容

OPEN 命令で、以下の内容を指定します。

項　目		設　定	内　容
デバイス記号（周辺装置名）		SCRN、LPRT、EKPRT、NKPRT、KYBD、COMU、TERM	1）送信／受信バッファの使用 2）伝送制御信号の状態制御
通信速度		300/600/1200/2400/4800/9600/19200/38400ビット/s（指定なし時:9600ビット/s）	
データ長		7,8ビット（指定なし：8ビット）	
パリティ		なし、奇数、偶数（指定なし：なし）	
ストップビット		1、2（指定なし：2）	
RS-232C ポートの場合	CS (CTS)	CS_ON	外部機器側の受信バッファのRS/CSフロー制御を行います。ASCIIユニットは、PRINT命令実行中に、CS信号をモニタします。CS信号がOFFとなると、データ送信を中断します。CS信号がONとなると、データ送信を再開します。
		CS_OFF	RS/CSフロー制御を行いません。ASCIIユニットは、PRINT命令実行中に、CS信号をモニタしません。
	RS (RTS)	RS_ON	通信ポートをオープンしてからクローズするまで、常にRS信号をONにします。周辺機器からつねにデータを受信バッファに受信する場合は、この RS_ON としてください。
		RS_OFF	入出力命令（PRINT,INPUT命令）実行時にのみ、RS信号をONとし、それ以外は、RS信号をOFFとします。
	DR (DSR)	DS_ON	ASCIIユニットは、出力命令（PRINT、LPRINTなど）実行開始時にDR信号をチェックし、DR信号がONならば送信を行います。DR信号がOFFならばDR信号がONになるなで待ちます。送信待ち状態のときBASIC LEDがゆっくり点滅します。
		DS_OFF	ASCIIユニットは、出力命令実行時にDR信号をチェックしません。
RS-422A/485 ポートの場合		CS,RS,DS指定なし（無効）	ー
Xon/Xoff (ASC21でのポート2では使用できません)		XN_ON	自身の受信バッファおよび外部機器の受信バッファの両方に対するXon/Xoffフロー制御を行います。 1）自身の受信バッファに対して： ASCIIユニットは自身の受信バッファが3/4（384バイト）に達すると、Xoffコードを送信し、外部機器に対してデータ送信停止を要求します。自身の受信バッファが1/4（128バイト）に下がると、Xonコードを送信し、外部機器に対してデータ送信再開を要求します。 2）外部機器の受信バッファに対して： ASCIIユニットは、PRINT命令実行中に、外部機器からのXoffコードを受信すると、データの送信を停止します。外部機器からXonコードを受信すると、データの送信を再開します。
		XN_OFF	Xon/Xoffフロー制御を行いません。

●伝送制御信号の制御命令／モニタ命令

OPEN命令による指定がない場合(OFFの場合)、下記の制御命令により伝送制御信号を制御し、モニタ命令によりモニタすることができます。

こんなとき	命令/関数	例	条　件
RS信号をON/OFFする	RTS命令	RTS #1 SET	OPEN命令によりRS_ONの場合のみ使用可能
ER信号をON/OFFする	DTR命令	DTR #1 SET	ポートがOPEN時使用可能
DR信号をモニタする	DSR命令	IF DSR(1) THEN…	条件なし
CS信号をモニタする	CTS命令	IF CTS(1) THEN…	条件なし

●デバイス記号(周辺装置名)

デバイス記号(周辺装置名)によって、次の内容を指定します。

・入出力の指定(送信／受信バッファの使用)

・伝送制御信号の状態制御

・PRINT命令によって出力するコード種類

注：PRINT命令によって出力するコード種類については、PRINT命令の項を参照してください。

デバイス記号	周辺装置	入出力(送受信)		伝送制御信号					
				通信ポートOPEN時				入出力命令(RRINT、INPUT命令)を実行中	
				RS		ER		RS	ER
		ASCIIユニットへの受信(受信バッファ使用)	ASCIIユニットからの送信(送信バッファ使用)	OPEN命令にてRS_OFF	OPEN命令にてRS_ON	ポート1/ターミナル専用ポート	ポート2	OPEN命令にてRS_ON RS_OFFとも	
COMU	汎用通信機器	○	○	OFF	ON	ON	OFF	ON	変化せず
TERM	ターミナル	○	○	OFF	ON	ON	OFF	ON	変化せず
SCRN	ディスプレイ	なし	○	OFF	ON	OFF	OFF	ON	変化せず
LPRT 注	プリンタ	なし	○	OFF	ON	OFF	OFF	ON	変化せず
NKPRT 注	プリンタ	なし	○	OFF	ON	OFF	OFF	ON	変化せず
EKPRT 注	プリンタ	なし	○	OFF	ON	OFF	OFF	ON	変化せず
KYBD	キーボード	○	なし	OFF	ON	ON	OFF	ON	変化せず

注：LPRT、EKPRT、NKPRTの違いは、以下のとおりです。

デバイス記号	対応外部機器	内　　容
LPRT	**漢字出力をしない場合**	**シフト**JIS**コード→**JIS**コード変換をしません。**
NKPRT	**漢字出力をする場合で、プリンタ制御コードが“** PC-PR **”のプリンタ**	**シフト**JIS**コードデータを**JIS**コードに変換の上、** KI**（漢字**IN**）：**[ESC]K（1B4BHex） KO**（漢字**OUT**）：**[ESC]H（1B48Hex） **ではさんで送信します**
EKPRT	**漢字出力をする場合で、プリンタ制御コードが“** ESC/P **”のプリンタ**	**シフト**JIS**コードデータを**JIS**コードに変換の上、** KI**（漢字**IN**）：**[FS]&（1C26Hex） KO**（漢字**OUT**）：**[FS]．（1C2EHex）**ではさんで送信します**

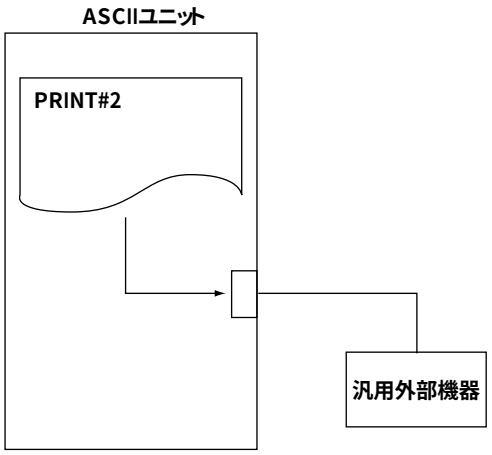
注：プリンタ制御コードが、“ Super　ESC/P ”の場合、NKPRT、EKPRTのどちらでも可能です。

参考：
PC-PR：NECのPC-PR××シリーズのプリンタで使用されたプリンタ制御コード
ESC/P：エプソンが提唱するプリンタ制御コード。多くのプリンタで使用されています。
Super　ESC/P： ESC/PとPC-PRを自動的に判断して使えるプリンタ制御コード

■データ送信

PRINT**命令**

PRINT命令によります。
指定通信ポートから、データを送信します。
たとえば、あらかじめ変数に格納しておいた通信コマンドフレームを送信する、プリンタに変数のデータを印字する、ディスプレイに変数内のデータを表示するなどが可能です。



注： デバイス記号がTERM、LPRT、SCRNのとき：
コンマ(，)で区切ると、9文字づつのフィールドに分けて出力します。
セミコロン(；)または空白で区切ると、続けて出力します。
最後がセミコロン（；)でない場合、出力後、改行します。
数値式の場合、前後に空白が1個とられます。（前の空白は、マイナス符号用）

●DMA(ダイレクトメモリアクセス)データ送信

ASCIIユニットでは、形式・通信ポートに応じて、DMAデータ送信が可能です。

形式	C200H-ASC11		C200H-ASC21		C200H-ASC31		
ポート	ポート#1	ポート#2	ポート#1	ポート#2	ポート#1	ポート#2	ポート#3
DMA送信*1	不可	可*1	不可	可*1	可*1	可*1	不可

*1：DMA送信可のポートで、DMA送信をする／しないの指定は、割付DMエリア(システム設定)で次のように設定します。

チャネル・ビット	名　　称	設定値(16進数)
m+2　ビット08〜15	ポート#1DMA送信 (形C200H-ASC31のみ可)	00：DMA送信しない(ASCIIユニット初期値) 5A：DMA送信する
m+3　ビット08〜15	ポート#2DMA送信 (全機種可)	00：DMA送信しない(ASCIIユニット初期値) 5A：DMA送信する

DMAデータ送信をしない通信ポートの場合、 PRINT命令によってメモリから通信ポートへ、データを1バイトづつ送信します。したがって、データ送信終了までBASICプログラムの実行は止まってしまいます。これに対して、DMAを使用した通信ポートの場合、PRINT命令はトリガをかけるのみで、DMAコントローラがかわりにデータ送信処理を行います。そのため、BASICプログラムは、次の行の実行に移ることができ、処理を高速化できます。

したがって、処理を高速化したい場合、DMA送信を使用してください。

なお、DMAデータ送信を「する」に指定した場合でも、OPEN命令によるRS/CSフロー制御またはXon/Xoffフロー制御の指定は有効です。ただし、フロー制御を指定した場合は、DMA送信を使用していても送信完了をPRINT命令内で待機することになります。

1度にDMAデータ送信ができるデータ量は、255バイトまでです。

255バイトを超えるデータを送信したい場合は、以下のようにしてください。

例：A$＝・・・・・・・：200バイト

　　B$＝・・・・・・・：100バイト

　　PRINT A$; B$ ←これで300バイトのデータ転送が可能です。

参考：

DMA(ダイレクトメモリアクセス)データ送信とは、データ送信において、MPUを介さず、直接通信ポートへデータを転送する方法です。MPUを使わないので、データ送信を行いながら並行してBASICプログラムを実行することが可能となります。データの転送は、専用の回路(DMAコントローラ)によって行います。

●伝送制御信号の動き

PRINT命令実行時にもしRS信号がOFFならば、RSをOFF→ONとした上、次の処理を行います。

OPEN命令時 CS_ON、DS_ON	CTSとDSRがともにONであれば、データを送信します。送信開始時DSRがOFFであれば、BASIC LEDがゆっくり点滅し、送信待ちとなります。また、データ送信中に、CTSがOFFとなると、データ送信を中断します。(BASIC LEDがゆっくり点滅します)
OPEN命令時 CS_ON、DS_OFF	CTSがONであれば、データを送信します。CTSがOFFであれば、データを送信せず、CTSがONとなるのを待ちます(BASIC LEDがゆっくり点滅します)。データ送信中に、CTSがOFFとなると、データ送信を中断します。
OPEN命令時 CS_OFF、DS_ON	送信開始時DSRがONであれば、データを送信します。DSRがOFFであれば、データを送信せず、BASIC LEDがゆっくり点滅し、DSRがONとなるのを待ちます。
OPEN命令時 CS_OFF、DS_OFF	CTS、DSRに関わらず、データを送信します。

●OPENしたデバイス記号による出力の可否

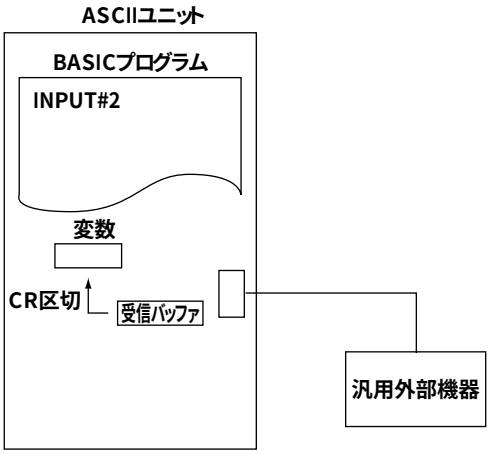
16進	記号	名　称	通信ポート1				通信ポート2(ASC31 ポート1)			
			TERM	SCRN	LPRT/ NKPRT/ EKPRT	COMU	TERM	SCRN	LPRT/ NKPRT/ EKPRT	COMU
00	NUL	空白								
01	SOH	ヘディング開始								
02	STX	テキスト開始								
03	ETX	テキスト終結								
04	EOT	伝送終了								
05	ENQ	問い合わせ								
06	ACK	肯定応答								
07	BEL	ベル								
08	BS	バックスペース								
09	HT	水平タブ								
0A	LF	改行(ラインフィールド)								
0B	VT	垂直タグ								
0C	FF	書式送り								
0D	CR	復帰								
0E	SO	シフトアウト	出力	出力	出力	出力	使用不可	出力	出力	出力
0F	SI	シフトイン								
10	DLE	伝送制御拡張								
11	DC1	装置制御1/Xon								
12	DC2	装置制御2								
13	DC3	装置制御3/Xoff								
14	DC4	装置制御4								
15	NAK	否定応答								
16	SYN	同期信号								
17	ETB	伝送ブロック終結								
18	CAN	取消し								
19	EM	媒体終端								
1A	SUB	置換キャラクタ								
1B	ESC	拡張								
1C	IS4	ファイル分離キャラク								
1D	IS3	グループ分離キャラクタ								
1E	IS2	レコード分離キャラクタ								
1F	IS1	ユニット分離キャラクタ								

■データ受信

INPUT命令またはINPUT$(文字数，#ポート番号)関数によります。

●INPUT命令

受信バッファから、CRコードを区切りとして、受信データを各変数に格納したい場合は、 INPUT命令を使います。
これにより、たとえば、CR区切りの通信レスポンスを受信したり、キーボードからのデータを入力したりできます。
CRコード入力までは、ユニット前面のBASIC LEDがゆっくり緑点滅し、入力待ちであることを示します。
ある時間待ってもデータを受信できないときに、受信待ちを打ち切りたい場合は、WAIT命令のあとにこの命令を使用します。



例：INPUT #2, A$　通信ポート#2の受信バッファから、CRコードまでのデータを文字変数A$に格納する。

例：INPUT #2, A$, B$　通信ポート#2の受信バッファから、CRコードまでのデータを，区切りで文字変数A$、およびB$に格納する。

注： INPUT命令では、CR, LFコードをデータとしては受信しません。

●伝送制御信号の動き

RS信号が常にONの場合、INPUT命令の実行に関係なく、受信データが受信バッファに格納されます。INPUT命令実行時に、CRコードを区切りとして、受信バッファからデータを各変数に格納します

INPUT命令実行時に、RS信号がOFFならば、 INPUT命令によってRSをOFF→ONとし、データが受信されると、データを受信バッファに格納し、かつCRコードを区切りとして、受信バッファからデータを各変数に格納します。(CS, DRはチェックしません)

受信バッファがオーバーフローしたとき、割付I/Oエリアのポート(1, 2, ターミナル)異常フラグがONとなります。

参　考

・ASCIIユニット自身の受信バッファがオーバーフローしないようにするには、Xon/Xoffフロー制御を使用します。

・通信相手の外部機器の受信バッファがオーバーしないようにするには、相手機器に合わせて、次のいずれかのフロー制御を選択してください。
OPEN命令実行時に、CS_ONとして、CS信号をモニタし、RS/CSフロー制御を行う、または、
OPEN命令実行時に、XN_ONとして、 XonXoffフロー制御を行う。

お願い

ON COM割込で受信処理を行う場合は、ポート1、ポート2の通信速度は
ポート1＋ポート2≦38.4kビット/s
となるようにしてください。
この条件を越えると、条件が成立した時点で「PORT CONFIGURATION ERROR」(エラーコード：0055)が発生します。

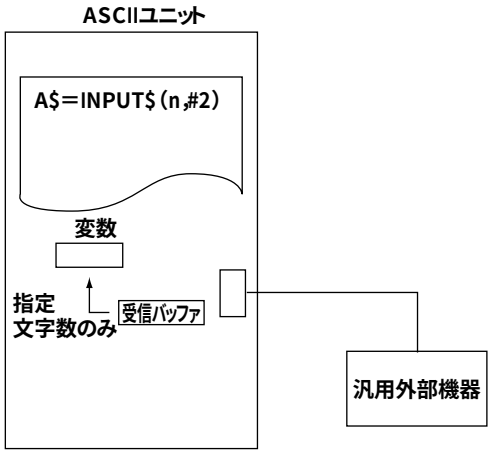
●INPUT$(文字数，#ポート番号)関数

受信バッファから、指定文字数だけ、データを文字変数に格納したい場合は、INPUT$(文字数，#ポート番号)関数を使います。

これにより、たとえば、CR以外をターミネータとする通信データを読み出したり、バーコードから指定文字数を読み出したり、受信バッファのデータを指定バイト数ずつ読み出したりすることが可能です。また、このINPUT$(文字数，ポート番号)関数をON COM(通信割込)命令またはON KEY(キー割込)命令と併用することで、通信割込またはキー割込が発生したとき、受信バッファの指定文字数のデータを、変数に格納することなどが可能です。

指定文字数を受信するまで、ユニット前面のBASIC LEDがゆっくり点滅し、入力待ちであることを示します。

ある時間待っても指定文字数を受信できないときに、受信待ちを打ち切りたい場合は、WAIT命令のあとにこの関数を使用します。



例：A$=INPUT$(10, #2)通信ポート#2の受信バッファから、10文字のデータを文字変数A$に格納する。

例)通信レスポンスフレームの受信

ターミネータがCRでないとき、通信レスポンスフレームの受信をこのINPUT$関数で行います。

A$=INPUT$(LOC(2)，#2) 通信ポート#2から受信バッファ内のデータをすべて文字変数A$に格納する

受信バッファに256バイト以上のデータが格納されていた場合、またはデータが空の場合、A$＝INPUT$(LOC(2)，#2)でデータを取り出そうとすると"ILLEGAL FUNCTION CALL" エラーとなってしまいます。

よって、下記のようなデータの取り出し方を行う必要があります。

```
I%=0
A%=LOC(2)
WHILE(A%<>0)
    IF (A%>255) THEN A%=255
    A$(I%)=INPUT$(A%, #2)
    I%=I%+1
    A%=LOC(2)
WEND
```

注：INPUT$(文字数，#ポート番号)関数では、CR, LFコードもそのままデータとして受信します。最大255文字格納できます。
また、受信バッファ内の文字数分を受信したいときは、LOC(ポート番号)命令を使用して文字数を指定することで、受信バッファにたまっているすべてのデータを指定変数に格納することができます。

例：A$=INPUT$(LOC(2)，#2)　通信ポート#2の受信バッファ内の文字数分をA$に格納します。

注：LOC(ポート番号)：指定ポート番号の受信バッファ内のデータのバイト数を与えます。

■通信ポートのクローズ

CLOSE 命令によります。

OPEN命令で送信用のデバイスが指定されている場合、CLOSE 命令を実行すると、送信バッファに残っていたデータは、はき出されからクローズされます。

OPEN 命令で受信用のデバイスが指定されている場合、CLOSE 命令を実行すると、受信バッファに残っていたデータはクリアされます。

すべての通信ポートは、END、NEW、CLEAR命令でも、クローズされます。なお、STOP命令では、クローズされません。

●伝送制御信号の動き

形　式	C200H-ASC11		C200H-ASC21		C200H-ASC31		
ポート	ポート1	ポート2	ポート1	ポート2	ポート1	ポート2	ターミナル専用ポート
RS	OFF			ー	OFF		
ER	ON	OFF	ON	ー	OFF		ON

■伝送制御信号のタイミングチャート

電源投入またはユニットリスタート直後、OPEN命令により各ポートをオープンするまでは、各ポートは次のデバイス記号(周辺装置名)の状態になっています。

形　式	ポート	デバイス記号(周辺装置名)
形C200H-ASC11 形C200H-ASC21	ポート1	TERM
	ポート2	LPRT
形C200H-ASC31	ポート1	LPRT
	ポート2	LPRT
	ターミナル専用ポート	TERM

●各伝送制御信号の状態

OPEN命令による「デバイス記号(周辺装置名)」の設定および「伝送制御信号の動き」の設定に応じて、各時点での各伝送制御信号(RS、ER、DR、CS)の状態は、以下のようになります。

●RS信号

デバイス記号（周辺装置名）の設定	伝送制御信号の動きの設定	RUN実行時	OPEN命令によるポートのオープン時	INPUT命令またはINPUT$関数によるデータ受信時	PRINT命令によるデータ送信時	ポートのクローズ時
・ポート1（ASC31の場合ポート3）がTERM、COMU、KYBD の場合	RS_ON	ON→OFF	OFF→ON	ONのまま	ONのまま	ON→OFF
	RS_OFF	ON→OFF	OFFのまま	OFF→ONし、受信後、ON→OFFに	OFF→ONし、（CS_ON、DR_ONのときはCS、DRをチェックし、ともにONならば）送信後、ON→OFFに	OFFのまま
・ポート1（ASC31の場合ポート3）が、KYBDの場合	RS_ON	ON→OFF	OFF→ON	ONのまま	ー	ON→OFF
	RS_OFF	ON→OFF	OFFのまま	OFF→ONし、受信後、ON→OFFに	ー	OFFのまま
・ポート2（ASC31の場合ポート1、2）がCOMUの場合	RS_ON	OFFのまま	OFF→ON	ONのまま	ONのまま	ON→OFF
	RS_OFF	OFFのまま	OFFのまま	OFF→ONし、受信後、ON→OFFに	OFF→ONし、（CS_ON、DR_ONのときはCS、DRをチェックし、ともにONならば）送信後、ON→OFFに	OFFのまま
・ポート2（ASC31の場合ポート1、2）が、KYBDの場合	RS_ON	OFFのまま	OFF→ON	ONのまま	ー	ON→OFF
	RS_OFF	OFFのまま	OFFのまま	OFF→ONし、受信後、ON→OFFに	ー	OFFのまま
・ポート1（ASC31の場合ポート3）がSCRN、LPRTの場合	RS_ON	ON→OFF	OFF→ON	ー	ONのまま	ON→OFF
	RS_OFF	ON→OFF	OFFのまま	ー	OFF→ONし、（CS_ON、DR_ONのときはCS、DRをチェックし、ともにONならば）送信後、ON→OFFに	OFFのまま
・ポート2（ASC31の場合ポート1、2）がSCRN、LPTRの場合	RS_ON	OFFのまま	OFF→ON	ー	ONのまま	ON→OFF
	RS_OFF	OFFのまま	OFFのまま	ー	OFF→ONし、（CS_ON、DR_ONのときはCS、DRをチェックし、ともにONならば）送信後、ON→OFFに	OFFのまま

●ER信号

デバイス記号(周辺装置名)の設定	RUN実行時	OPEN命令によるポートのオープン時	INPUT命令またはINPUT$関数によるデータ受信時	PRINT命令によるデータ送信時	ポートのクローズ時
ポート1(ASC31の場合ポート3)がTERM、COMUの場合	ONのまま	ONのまま	ONのまま	ONのまま	ONのまま
ポート2(ASC31の場合ポート1、2)がCOMUの場合	OFFのまま	OFF→ON	ONのまま	ONのまま	ON→OFF
ポート1(ASC31の場合ポート3)がKYBDの場合	ONのまま	ONのまま	ONのまま	−	ONのまま
ポート2(ASC31の場合ポート1、2)がKYBDの場合	OFFのまま	OFF→ON	ONのまま	−	ON→OFF
ポート1(ASC31の場合ポート3)がSCRNまたはLPRTの場合	ONのまま	ON→OFF	−	OFFのまま	OFF→ON
ポート2(ASC31の場合ポート1、2)がSCRNまたはLPRTの場合	OFFのまま	OFFのまま	−	OFFのまま	OFFのまま

●CS、DR信号

デバイス記号(周辺装置名)の設定	伝送制御信号の動きの設定	RUN実行時	OPEN命令によるポートのオープン時	INPUT命令またはINPUT$関数によるデータ受信時	PRINT命令によるデータ送信時	ポートのクローズ時
ポート1(ASC31の場合ポート3)がTERM、COMU、SCRN、LPRTの場合	CS_ON	チェックなし	チェックなし	チェックなし	送信中チェックし、CS信号がOFFであれば送信を中断*1	チェックなし
ポート2(ASC31の場合ポート1、2)がCOMU、SCRN、LPRTの場合	CS_OFF	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし
	DS_ON	チェックなし	チェックなし	チェックなし	DR信号がONであることをチェックし、ONであれば送信*2	チェックなし
	DS_OFF	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし

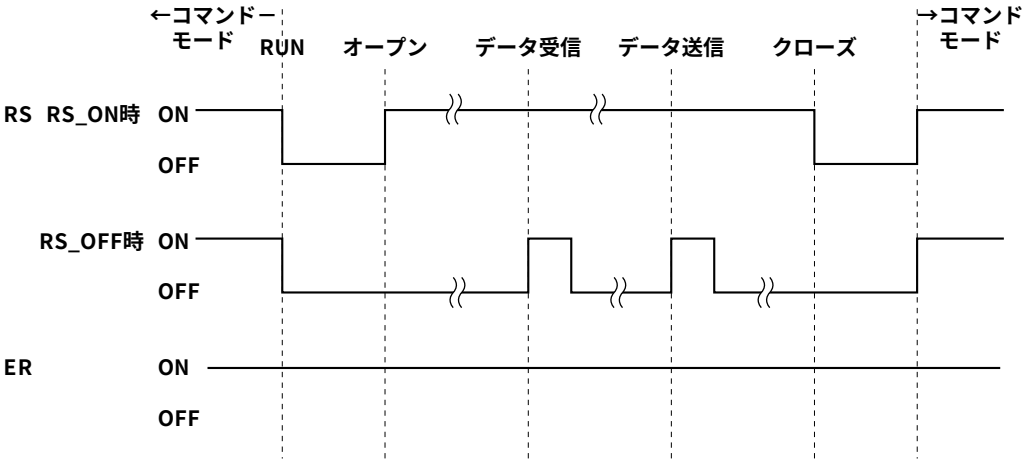
*1：CS信号は、CS_ONでPRINT命令によるデータ送信中にモニタ(チェック)されます。OFFであれば、データ送信を中断、ONであれば再開します(外部機器側の受信バッファのRS/CSフロー制御を行います)。

*2：DR信号は、DS_ON時でPRINT命令実行時にモニタ(チェック)されます。OFFであれば送信待ち、ONであれば送信を行います。

●タイミングチャート

デバイス記号：TERM

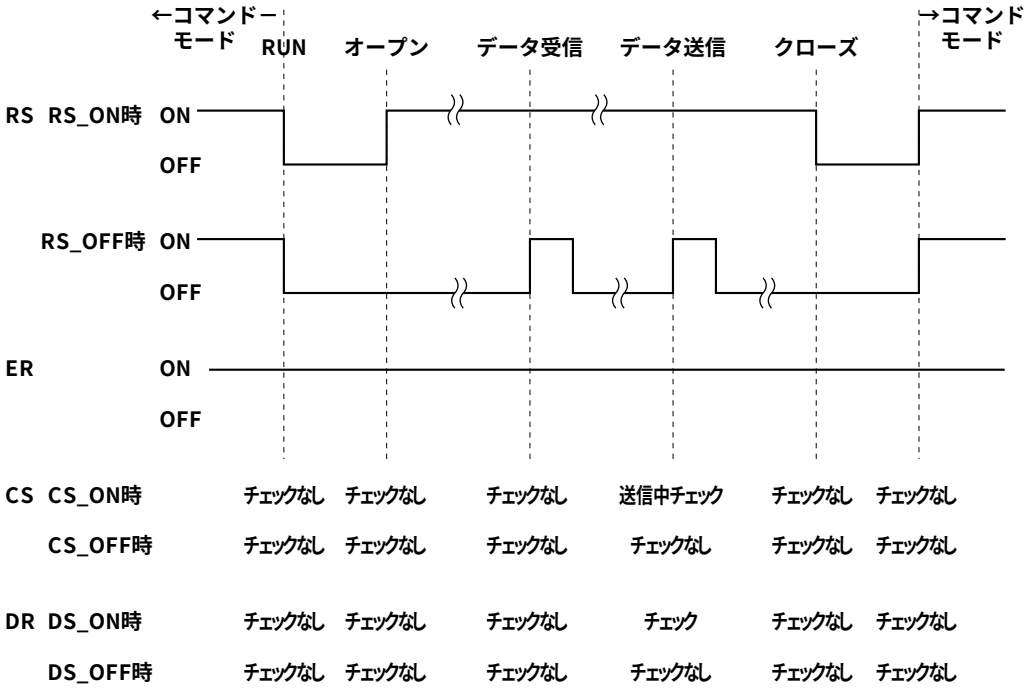
ポート1、ターミナル専用ポートがTERMの場合



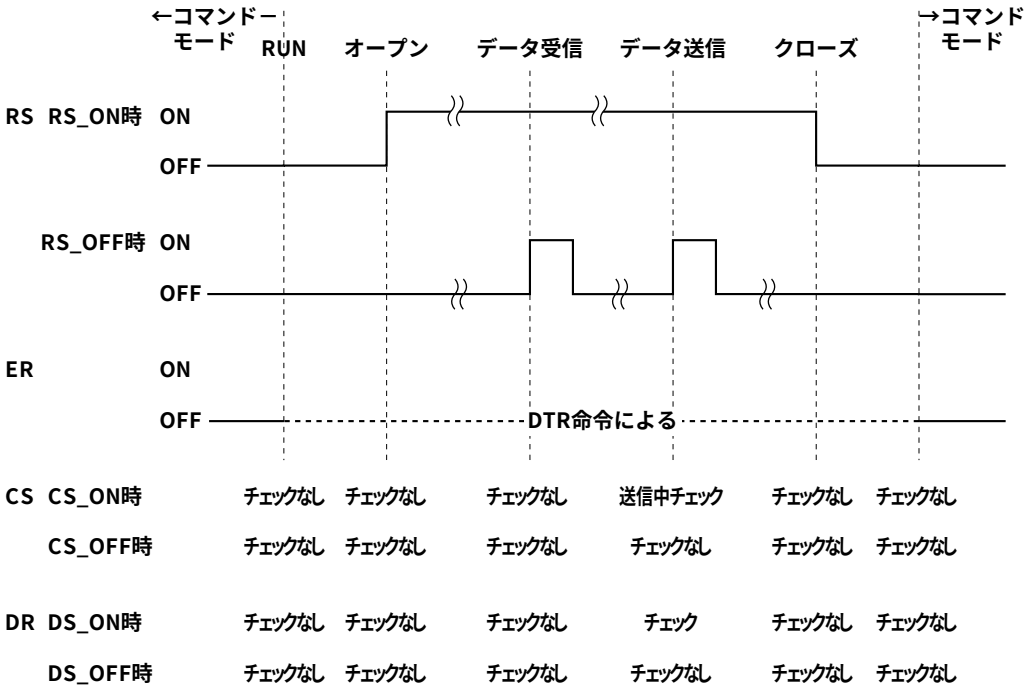
CS	CS_ON時	チェックなし	チェックなし	チェックなし	送信中チェック	チェックなし	チェックなし
	CS_OFF時	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし
DR	DS_ON時	チェックなし	チェックなし	チェックなし	チェック	チェックなし	チェックなし
	DS_OFF時	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし	チェックなし

デバイス記号：COMU

ポート1がCOMUの場合（ASC11、ASC21の場合）

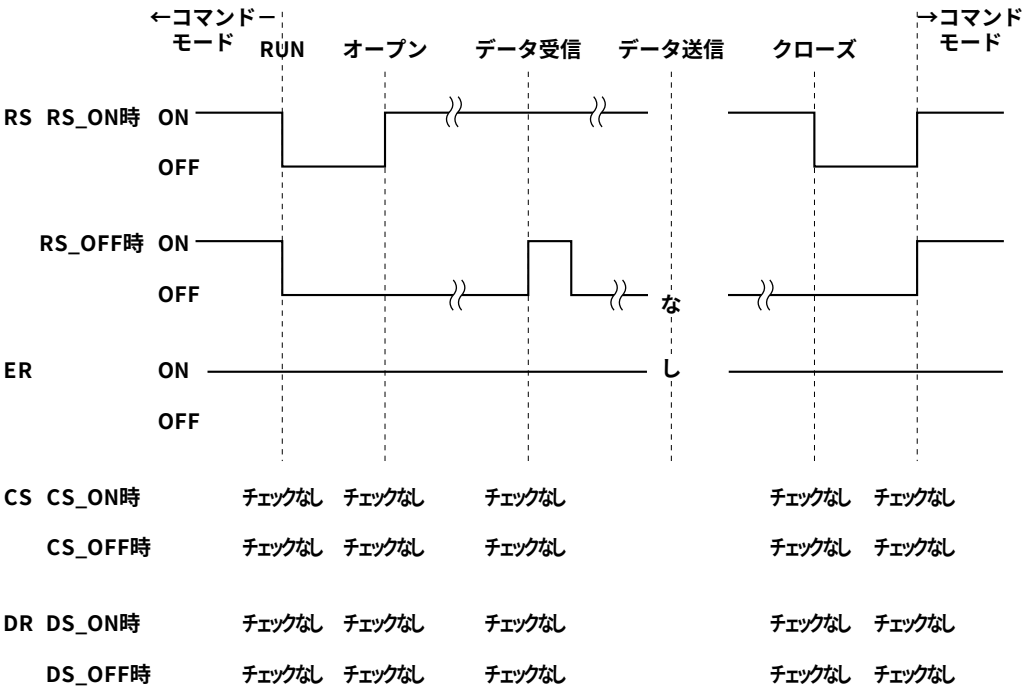


ポート2がCOMUの場合（またはASC31のポート1がCOMU）

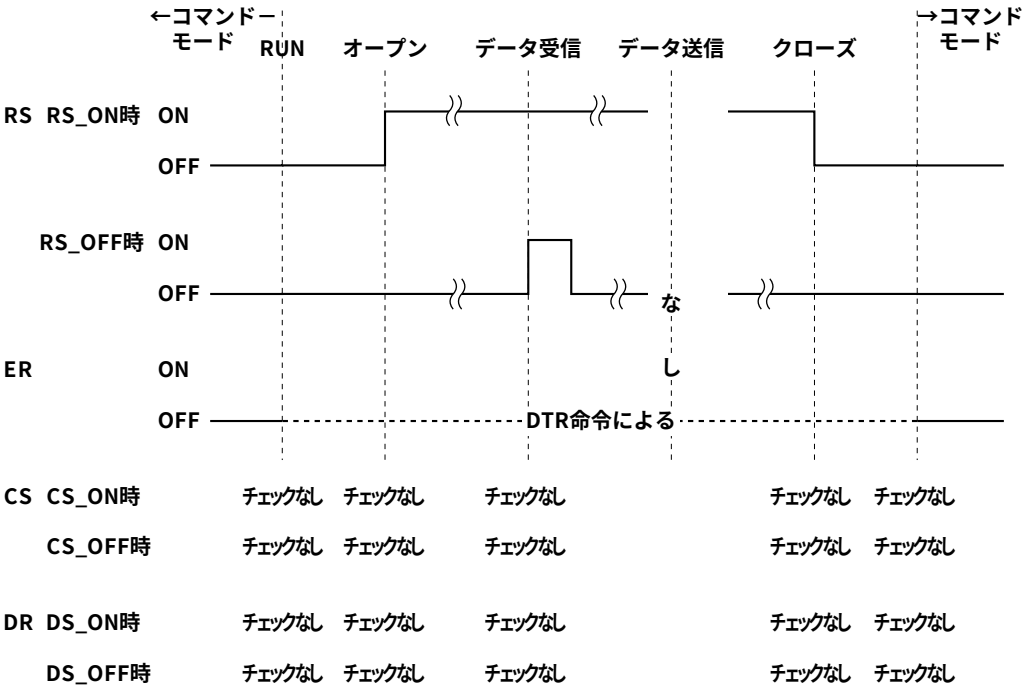


デバイス記号：KYBD

ポート1がKYBDの場合（ASC11、ASC21の場合）

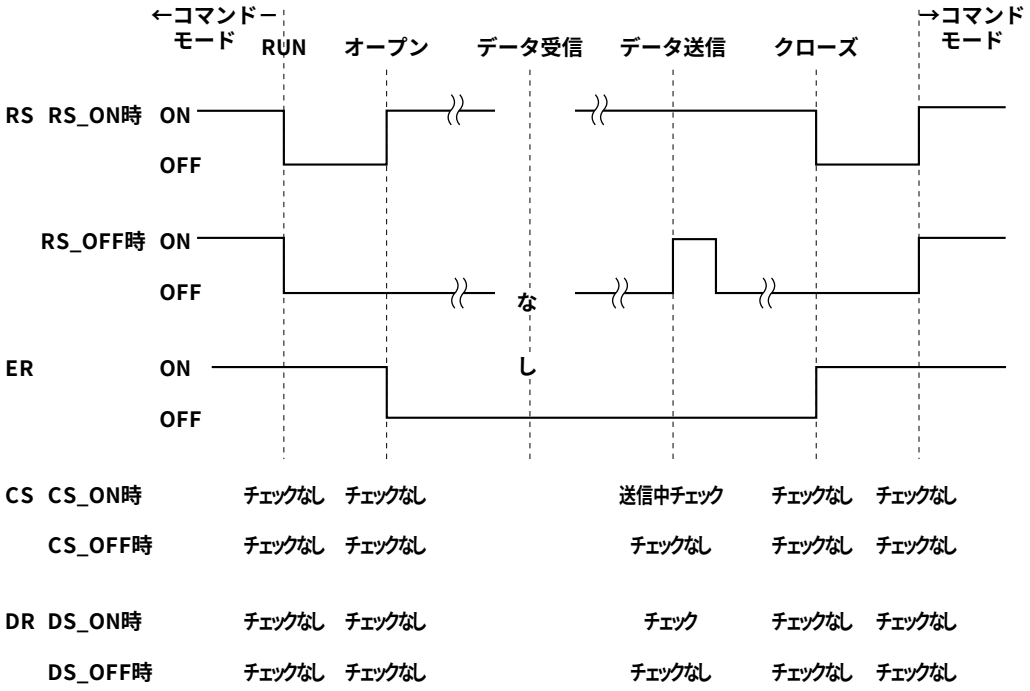


ポート2がKYBDの場合

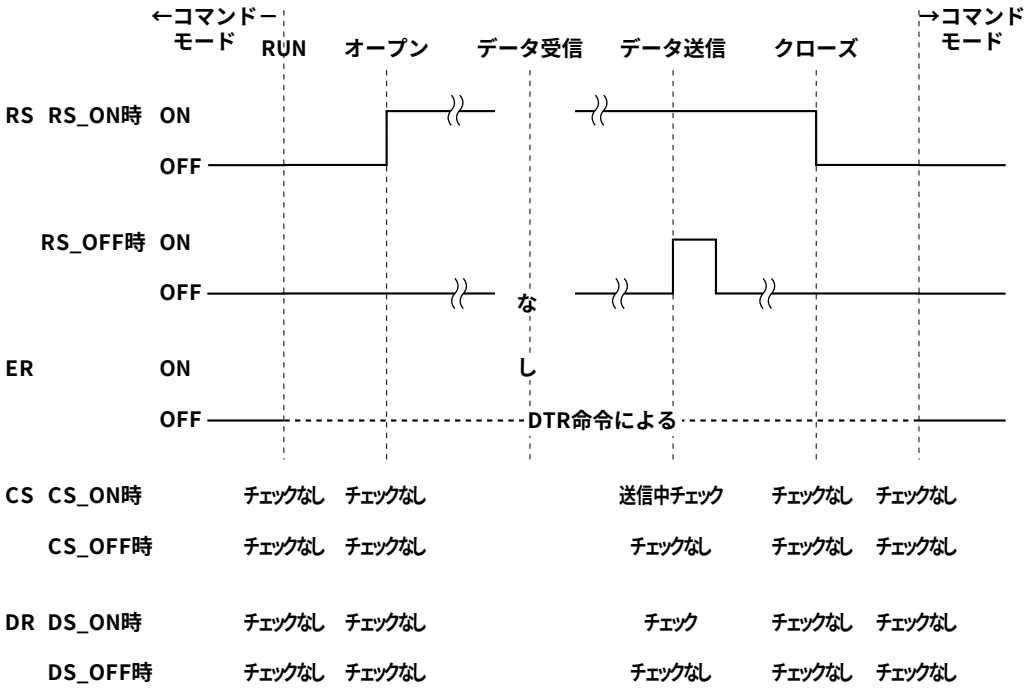


デバイス記号：SCRN**または**LPRT

ポート1がSCRNまたはLPRTの場合



ポート2がSCRNまたはLPRTの場合



■通信エラー処理

BASICプログラム実行中に、通信エラーが発生すると、ASCIIユニットのERR1またはERR2 LEDが点灯し、同時にCPUユニットのポート1／2異常フラグ(n+5CHのビット04／05)が1(ON)となります。このとき、BASICプログラムの実行は継続します。

ON ERROR命令をBASICプログラム中に使用した場合は、通信エラーが発生したときに、指定した割込サブルーチンへ分岐します。

ただし、エラー割込サブルーチン実行中に、もう1度エラーが発生すると、(そのエラーはエラー割込の対象とならず)BASICプログラムの実行が終了してしまいます。

したがって、以下の可能性がある場合には、プログラム実行が終了してしまうことがあり得ますので、 ON ERROR命令を使用するときには、注意してください。逆に、通信エラーが発生しても、BASICプログラムを停止させることなく必ず実行を継続したい場合には、ON ERROR命令を使用しないでください。

・通信エラーによる割込処理中の他のエラーの発生

参 考 **ON ERROR命令について**

ON ERRORは、他のエラー処置とも共通ですので、処理は、IF文などで判定する必要があります。

エラー処置が完了すれば、RESUME命令で、分岐元に戻り処理を継続します。

エラー処置後に、エラー状態を解除したい場合には、ERC命令で解除できます。

発生したエラー種類を判別するには、ERR関数を使用することで確認できます。

通信エラーが発生したことをBASICプログラム上で知ることができますので、BASICプログラムでリトライ処理などが容易に行えます。

```
ON ERROR GOTO *EINT
```

メインプログラム

```
*EINT
```

エラー処理

第5章
各種の処理

5－1　各種の処理

■文字列処理

A$の先頭からn文字を取り出す	B$=MID$(A$,1,n)またはB$=LEFT$(A$,n)
A$のm文字目からn文字を取り出す	B$=MID$(A$,m,n)
A$の右(最終)からn文字を取り出す	B$=RIGHT$(A$,n)
A$の中で、m文字目から、□の文字列をさがし、その先頭文字の位置を得る	B=INSTR(m,A$,□)
A$の文字数を得る	B=LEN(A$)
文字列A$を数値に変換する	B=VAL(A$)
数値Bを文字列A$に変換する	A$=STR$(B)

例：レスポンスデータR$の中から「STX」の位置を知り、そこから「ETX」までの文字数を取り出し、A$に格納する

```
100  STX$=CHR$(&H02)          STX作成
110  ETX$=CHR$(&H03)          ETX作成
120  A=INSTR(1, R$, STX$)     R$の中からSTX$の位置を知る
130  B=INSTR(1, R$, ETX$)     R$の中からETX$の位置を知る
140  A$=MID$(R$, A, B-A+1)    R$の中から、STX$からETX$までの文字列を
                              取り出し、A$に格納する
```

例：レスポンスデータA$の先頭から、4文字目から2文字が、00なら、正常レスポンスとして、行番号1000へ

```
150  IF MID$(A$, 4, 2)= "00 "THEN 1000
```

■ビット処理

変数の1ビットをON/OFFする	BITON A%,3　変数A%のビット3をONする
	BITOFF A%,3　変数A%のビット3をOFFする

注：

・変数に10進数を入力せずに、1ビットをセットする。

・ビットデータをPCに渡すときなどに、この命令でビットをON/OFFしてから、 PC PUT命令を使用します。

■受信バッファの処理

受信バッファ内の文字数を得る	A=LOC(1)　通信ポート#1の受信バッファ内の文字数をAに格納
受信バッファの文字数が0でなければ(何かあれば)、処理Aを行う	100 A=LOC(1) 110 IF A<>0 THEN 200 200　処理A

■時間処理

その時点の時刻をT$に格納する（HH:MM:SS）		T$=TIME$
その時点の時刻のうち、時間をH$に格納する（HH）		H$=LEFT$（TIME$,2）**または** H$=MID$（TIME$,1,2）
その時点の時刻のうち、分秒をMS$に格納する（MM:SS）		MS$=MID$（TIME$,4,5）
一定時間毎に処理Aを行う	TIME$**割込で、毎分（HH時MM分00秒）割込サブルーチン処理を行う**	100 ON　TIME$ ”＊＊:＊＊:00 ” GOSUB　＊A 110 TIME$　ON ⋮ 500 ＊A **処理**A
	TIMER**割込で60秒後毎に割込サブルーチンで処理Aへ**	100 ON TIMER 600 GOSUB　500 110 TIMER ON ⋮ 500 **処理**A
待機処理を行う	**コマンド（C$）送信後、3秒以内にレスポンスが返ってこないとこきは、処理Aへ。**	100 PRINT #1,C$ 110 WAIT ”3.0 ”, 200 120 A$=INPUT$（1,#1） ⋮ 200 **処理**A
	PC READ**命令実行に入って、10秒以内に、CPUユニットからのデータ転送が完了しなかったとき、処理Aへ**	100 WAIT ”10.0 ”, 200 110 PC READ・・・ ⋮ 200 **処理**A

注：WAIT命令で待機中にBASIC割込が発生すると、その間は監視タイマが停止しています。

■割込機能

種　類	内　容	命　令	許可／禁止／停止命令 ＊2
通信割込	受信バッファに何らかのデータを受信すると、割込サブルーチンを実行	ON COM n GOSUB n：1、2	COM n（ON / OFF/STOP） n：1、2
	受信バッファに指定した指定1文字を受信すると、割込サブルーチンを実行（指定文字以降のデータも受信バッファには受信されます）＊1	ON COM n GOSUB,CODE=XX n：1、2	
	受信バッファに指定バイト数のデータを受信すると、割込サブルーチンを実行（指定バイト数以降のデータも受信バッファには受信されます）＊1	ON COM n GOSUB,BYTE=XX n：1、2	
	受信バッファに指定した先頭文字を受信した後、指定した最終文字を受信すると、割込サブルーチンを実行（指定先頭文字から指定最終文字までの文字列は、受信バッファに残ります。指定最終文字以降のデータも受信バッファには受信されます）＊1	ON COM n GOSUB,HEAD=XX,TERM=YY n：1、2	
	受信バッファに指定した先頭文字を受信した後、指定した文字数を受信すると、割込サブルーチンを実行（指定先頭文字から指定文字数までの文字列は、受信バッファに残ります。指定バイト数以降のデータも受信バッファには受信されます）＊1	ON COM n GOSUB,HEAD=XX,BYTE=YY n：1、2	
キー割込	あるキー番号入力で割込サブルーチンを実行＊3	ON KEY(n) GOSUB n：0〜9	KEY n（ON / OFF /STOP） n：0〜9
PC割込	PCからの割付I/OまたはIOWR(#CC00)命令による割込により、指定PC割込番号の割込サブルーチンを実行＊4 注：割付I/O：nCH ビット04でPC割込有効 n+2CHビット00〜07でPC割込番号指示 IOWR(#CC00)：PC割込番号を指示	ON PC n GOSUB n：01〜99	PC n（ON/ OFF/STOP） n：1〜99
タイマ割込	タイマ(インターバルタイマ)のアップで、割込サブルーチンを繰り返し実行(1度実行すると、TIME OFF指定まで再実行)	ON TIMER 300 GOSUB 本命令実行の30秒後に、GOSUB行番号へサブルーチン分岐、30秒後またサブルーチン分岐を繰り返す タイマ：0〜864000(0.1s単位)	TIMER(ON / OFF /STOP)
ONアラーム割込	タイマ(ワンショットタイマ)のアップで、割込サブルーチンを実行(1度実行すると、再実行せず)	ON ALARM 300 GOSUB 本命令実行の30秒後に、一度だけGOSUB行番号へサブルーチン分岐、 タイマ：0〜864000(0.1s単位)	ALARM(ON / OFF / STOP)
時刻割込	ある時刻になると、割込サブルーチンを実行	ON TIME$= "□:□:□ "GOSUB	TIME$(ON / OFF /STOP)
エラー割込	エラーが発生すると、割込サブルーチンを実行	ON ERROR GOTO	

＊1：指定文字数のデータまたは指定文字までのデータを受信して割込をかける場合、それ以降のデータも受信バッファには受信されます。必要な文字数をINPUT$関数で取り出すと、その文字数分だけ受信バッファの中で前つめされるため、先入れ先出し(FIFO)処理が可能です。

＊2：いずれも、関連命令の項で示す命令で、割込許可、禁止、停止を指定することが必要です。なお、停止(STOP)命令は、割込がはいっても、割込がはいったことを記憶しておくのみで、処理分岐せず、その後、許可(ON)命令で処理分岐します。

＊3：GOTO文を使った割込み指定は、実行中の命令が終了するまで割込み待機します。

＊4：ASCIIビジーフラグがON中は、割込みを受け付けられません。

●割込優先順位

同時に割込みが入ったときは、下記の優先順位テーブルにしたがい、割込サブルーチンが実行されます。

割込優先順位テーブル

優先順位	割　込
1	ON　COM　1
2	ON　COM　2
3	ON　KEY　0
≀	≀
12	ON　KEY　9
13	ON　PC　1
≀	≀
111	ON　PC　99
112	ON　ALARM
113	ON　TIMER
114	ON　TIME$

ある割込サブルーチン実行中に、別の割込が入ったときは、多重割込はせず、実行中の割込サブルーチンを終了後、上記の割込優先順位テーブルに従い、次の割込サブルーチンを実行します。ただし、ON ERROR割込の場合は、ある割込サブルーチン実行中でもすぐに実行されます。

例：ON PC1の割込サブルーチン実行中にON COM 1、ON ALARM割込が同時に発生すると、ON PC1割込サブルーチン実行完了後、ON COM1はON ALARMより優先されるため、ON COM1の割込サブルーチンが先に実行されます。

ON COM1の割込サブルーチンの実行が完了すると、ON ALARMの割込サブルーチンが実行されます。ON COM1の割込サブルーチンの実行中に、PCの割込サブルーチンが入ると、ON ALARMの割込よりも先にPCの割込サブルーチンが実行されます。

なお、同時に割込が入った場合も、またある割込サブルーチン実行中に他の割込が入った場合も、その割込が許可(ON)または停止(STOP)状態であるならば、割込が入ったことはそれぞれ1個分記憶されます。

●使い方

```
200   ON COM 1 GOSUB 1000
210   COM 1 ON
      通信割込可能状態
1000  割込サブルーチン処理
      RETURN
```

使い方

ON COM命令は、しばしば以下のようにINPUT$関数と組み合わせて使用します。
STX%=&H02、ETX%=&H03、受信データは0＜LOC(2)＜255とします。

例1
```
100 ON COM 2 GOSUB 500
110 COM 2 ON
120 GOTO120
500 A$=INPUT$( LOC(2), #2)
510 RETURN
```
何らかのデータを受信すると、受信バッファ内の全データをA$に格納

例2
```
100 ON COM 2 GOSUB 500, CODE=ETX%
110 COM 2 ON
120 GOTO 120
500 A$=INPUT$( LOC(2), #2)
510 RETURN
```
ETX%を受信すると、受信バッファ内の全データをA$に格納

例3
```
100 ON COM 2 GOSUB 500, BYTE=20
110 COM 2 ON
120 GOTO 120
500 A$=INPUT$(20, #2)
510 RETURN
```
20文字を受信すると、受信バッファ内の20文字をA$に格納

例4
```
100 ON COM 2 GOSUB 500, HEAD=02, TERM=03
110 COM 2 ON
120 GOTO 120
500 A$=INPUT$(LOC(2), #2)
510 RETURN
```
STX%から ETX%を受信すると、受信バッファ内の全データをA$に格納

例5
```
100 ON COM 2 GOSUB 500, HEAD= STX%, BYTE=20
110 COM 2 ON
120 GOTO 120
500 A$=INPUT$(LOC(2), #2)
510 A$=RIGHT$(A$, 20)
520 RETURN
```
STX%から 20文字を受信すると、受信バッファ内の20文字をA$に格納

受信バッファにはすでに次のデータが入ってきていることがありますので注意してください。

■繰り返し処理

指定回数繰り返す。 コマンド(C$)送信後、3秒以内にレスポンスが返ってこないときは、再度コマンドを送信する。3回送信しても、レスポンスがないとき、エラー処理へ。	10 FOR A=1 TO 3 20 PRINT #2,C$ 30 WAIT "3.0", 50 40 A$=INPUT$(1,#2):GOTO 1000 50 NEXT A 60 **エラー処理**
条件成立の間、繰り返す。 コマンド(C$)を発行し、レスポンスB$の 4文字目からの2文字が20以下の間、 コマンド発行・レスポンス受信を繰り返す。	10 WHILE A<=20 20 PRINT #2,C$ 20 B$=INPUT$(LOC(2), #2) 30 A$= MID$(B$,4,2) 40 A=VAL(A$) 100 WEND

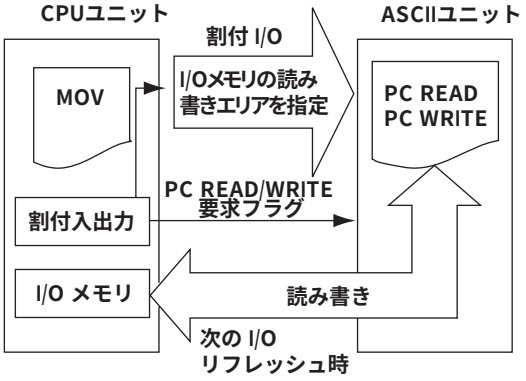
第6章
CPUユニットとのデータ交換

6−1　データ交換方法の概要

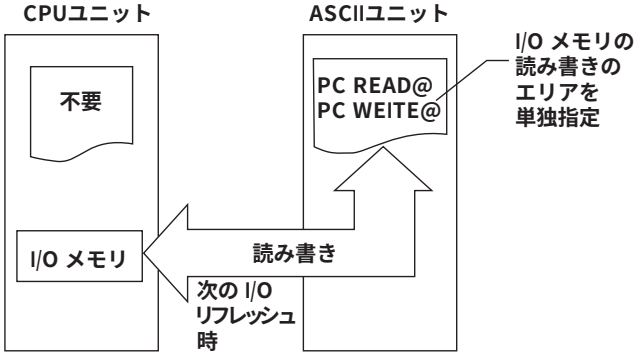
■5種類のデータ交換方法の概要

ASCIIユニットは、CPUユニットと、次の5種類のデータ交換を行うことができます。

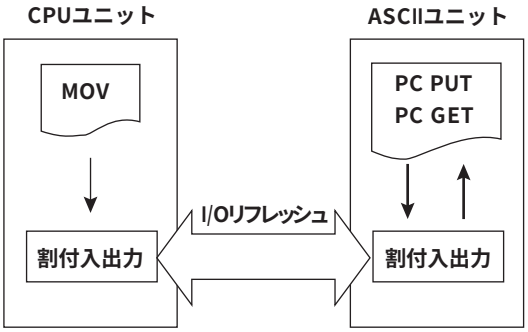
1. CPU ユニットが 自身の I/O メモリの読み書きエリアを指定する。CPUユニットからPC READ/WRITE命令の実行を指示されると、ASCIIユニットは、PC READ/WRITE命令によって、CPUユニットの指定された I/O メモリエリアを読み書きする。



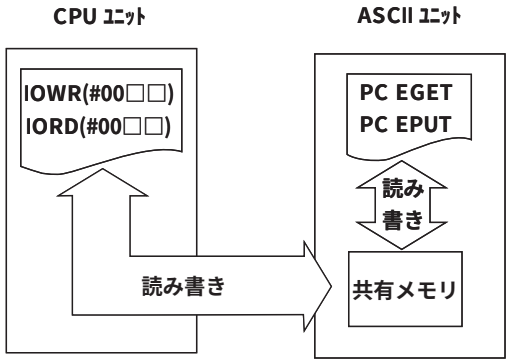
2. ASCIIユニットが主体でCPUユニットのI/Oメモリを読み書きする。CPUユニットのユーザープログラムは全く不要です。



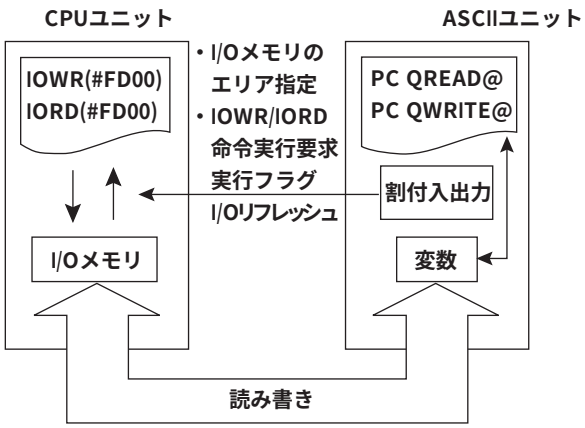
3. CPUユニットとASCIIユニットは、各々自身の割付入出力データを読み書きする。（ASCIIユニットは、PC PUT/GET 命令によって自身の割付入出力データを16ビット単位で読み書きする。）CPUユニットの割付入出力データと ASCIIユニットのそれは、I/Oリフレッシュ時にデータ交換される



4. ASCIIユニットは、自身の共有メモリを PC EGET/EPUT 命令で読み書きします。
それとは非同期で、CPUユニットは ASCIIユニットの共有メモリを、IOWR(#00□□)/IORD(#00□□)命令で読み書きします。（ただしIOWR(#00□□)、IORD(#00□□)命令は、PCがSYSMACα、CSシリーズのみ可）



5. ASCIIユニットが CPUユニットに対し、IOWR 命令実行要求フラグをセットする。IOWR(#FD00)命令が実行されると、CPUユニットがデータを書き込みます。
ASCIIユニットがCPUユニットに対し、IORD命令実行要求フラグをセットする。IORD(#FD00)命令が実行され、CPUユニットがデータを読み出します。
（ただし、PC がSYSMACα、CSシリーズのみ可）

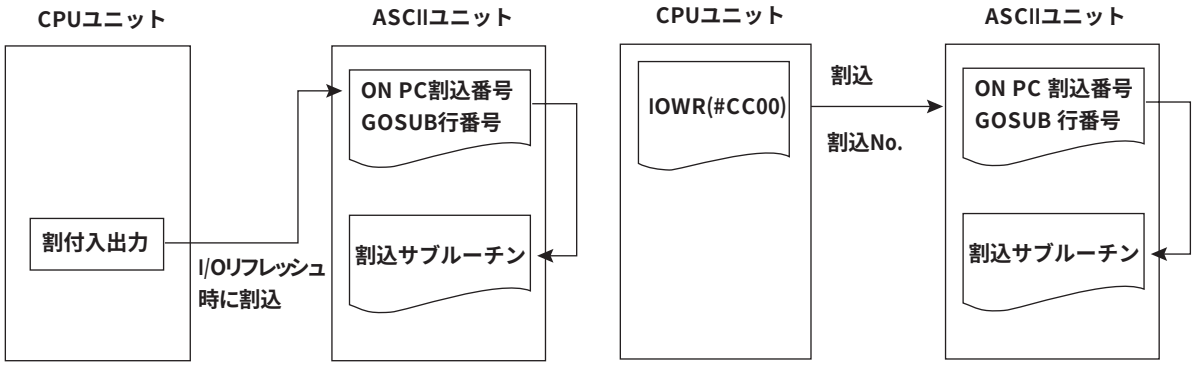


■CPUユニットからASCIIユニットへの割込のかけ方

CPUユニットからASCIIユニットへの割込のかけ方に次の2種類があります。（ASCIIユニットからCPUユニットへ割込をかけることはできません。）

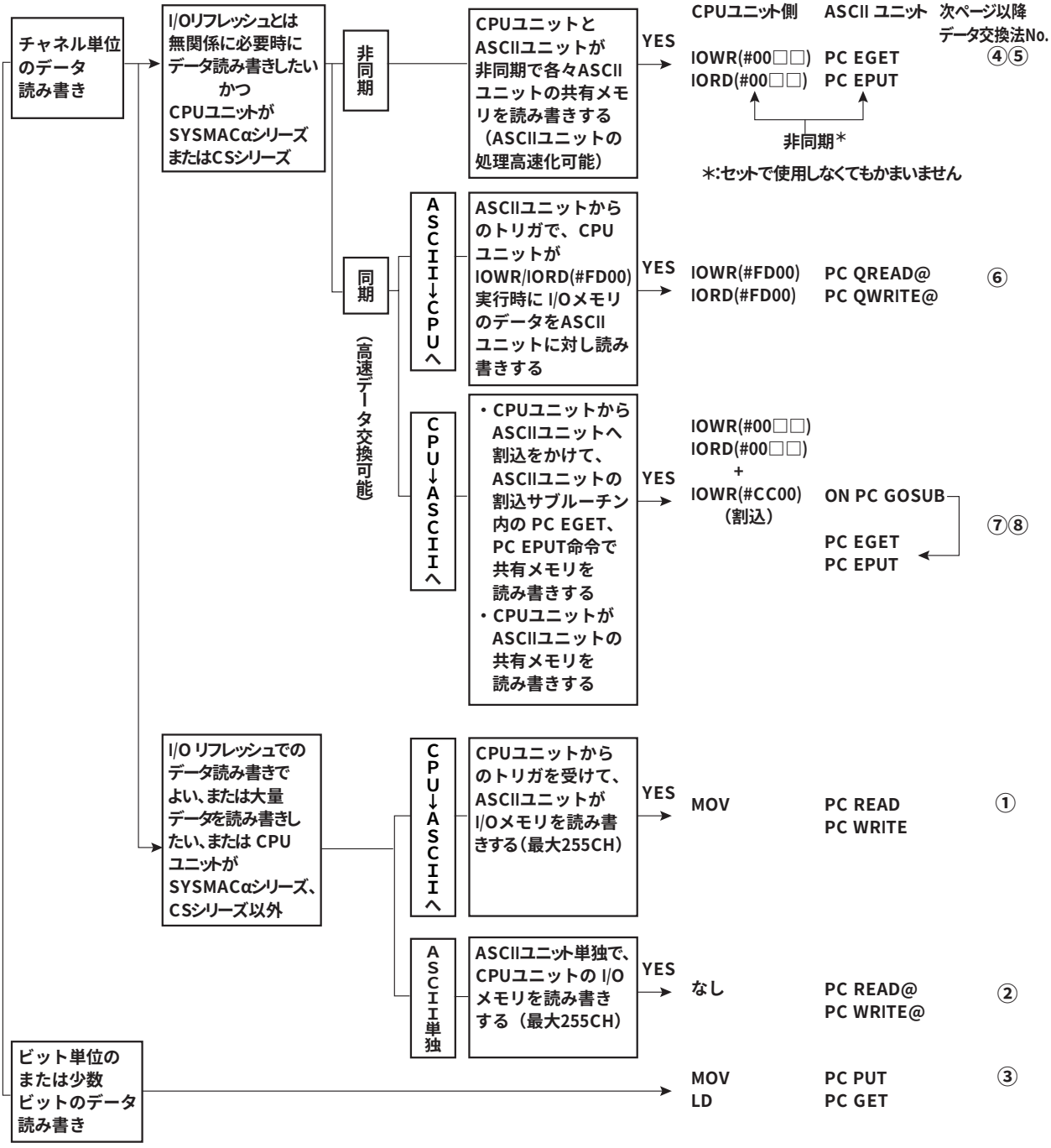
1. I/Oリフレッシュ時に割込をかける。
割込入出力データ内に割込番号を設定し、PC 割込フラグを立てると、ON PC命令によって指定された行番号へ分岐する。

2. IOWR(#CC00)命令実行時に割込をかける。
CPU ユニットのユーザープログラム上の IOWR(#CC00)命令によって、割込をかけるとともに割込番号が ASCII ユニットに渡され、ON PC 命令の指定行番号に分岐します。
あらかじめ IOWR(#00□□)命令で共有メモリに入力データをセットしておけば、割込サブルーチン内で、PC EGET/EPUT 命令による共有メモリの読み書きなどを行うこともできます。（ただし、SYSMACα、CSシリーズのみ可）



注：ASCIIビジーフラグがONの間は、PC割込をかけることはできません。

6−2　データ交換方法の選択のしかた



■データ交換方法の比較表

各データ交換方法の違い(特長)は、以下のとおりです。

データ交換方法	命令		読み書きの主体(どこが読み書きに行くか)	読み書きのトリガ	転送のタイミング	ASCII側のプログラム	最大転送チャネル数	
	CPUユニット側	ASCIIユニット側					1命令毎	1サイクル毎
①	MOV等 (PC WRITE/ PC READ命令実行時のI/Oメモリの読み書きエリア指定)	PC WRITE PC READ (CPUユニットのI/Oメモリ読み書き)	ASCIIユニット	CPUユニット	I/Oリフレッシュ	PC WRITE / READ実行から、転送終了まで停止	PC WRITE/PC READ：255CH	SYSMACαまたはCSシリーズの場合でシステム設定(DMm+4のビット8～15)で1サイクルあたり転送バイト数を127CHとしているとき：127CH それ以外：20CH
②	(プログラム不要)	PC WRITE@ PC READ@ (CPUユニットのI/Oメモリ読み書き)	ASCIIユニット	ASCIIユニット	I/Oリフレッシュ	PC WRITE@/PC READ@実行から、転送終了まで停止	PC WRITE@/PC READ@：255CH	
③	MOV, LD等	PC GET PC PUT (CPUユニットリフレッシュデータエリアの割付、入出力データ読み書き)	どちらでも可	どちらでも可	I/Oリフレッシュ	止まらない	PC PUT：1CH PC GET：2CH	PC PUT：1CH PCGET：2CH
④ ⑤	IOWR(#00□□) IORD(#00□□) 注：SYSMACα、CSシリーズのみ可	PC EGET PC EPUT (ASCIIユニット自身内の共有メモリ読み書き)	CPUユニット	どちらでも可	IOWR/IORD(#00□□)命令実行時	止まらない	PC EGET/PC EPUT：90CH IOWR/IORD(#00□□)：90CH	制限なし
⑥	IOWR(#FD00) IORD(#FD00) 注：SYSMACα、CSシリーズのみ可	PC QREAD@ PC QWRITE@	CPUユニット	ASCIIユニット	IOWR/IORD(#FD00)実行時	PC QREAD/QWRITE実行から、IOWR,IORD(#FD00)実行まで停止	PC QREAD/PC QWRITE：128CH IOWR,IORD(#FD00):128CH	制限なし
⑦ ⑧	IOWR(#00□□) IORD(#00□□) ＋IOWR(#CC00) 注：SYSMACα、CSシリーズのみ可	ON PC GOSUB PC EGET/EPUT	CPUユニット	CPUユニット	IOWR/IORD命令実行	止まらない	同上	制限なし

お願い　**上記①および②の方法は、CSシリーズ CPUユニットのI/Oメモリエリアを読み書きする場合に、一定の制限があります。詳細は6－6項「CSシリーズで使用する場合の制限」を参照してください。③～⑧の方法が使える場合は、③～⑧の方法を使用してください。**

6−3　データ交換方法の詳細

①CPUユニットからのトリガで、ASCIIユニットがI/Oリフレッシュ時に、CPUユニットのI/Oメモリを読み書きする

（比較的大量のデータ交換が可能。最大255チャネル。ただし、ASCIIユニットの待機が長い。C200H、C200HSも可）

●PC READ

ASCIIユニットは、CPUユニットからのPC READ要求を受けると、割付I/Oで指定されたI/Oメモリを読み書きし、指定変数に格納します。

手順

① PC READ命令で要求

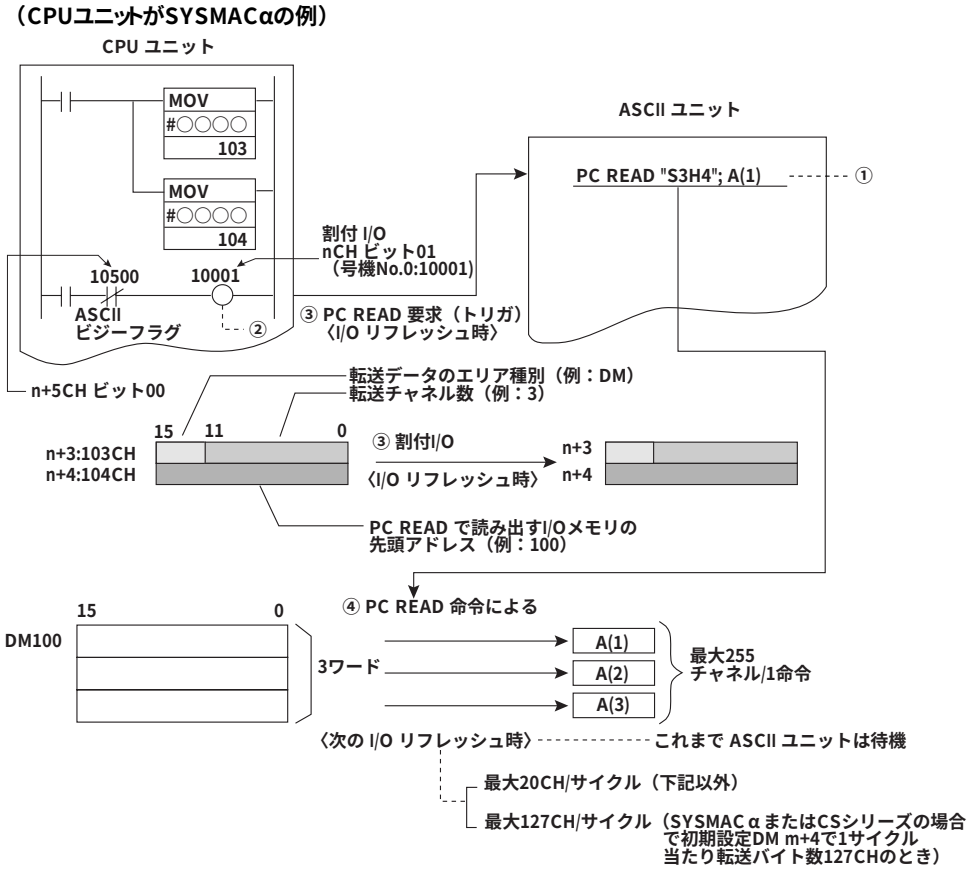
② OUT命令でPC READ要求ON（ラダー）

③ I/Oリフレッシュ時、CPUユニット→ASCIIユニットへPC READ要求（nCH、ビット01）をOFF→ONへ

④ 次のI/Oリフレッシュ時、ASCIIユニットがデータ読み出し（ASCIIビジーフラグOFF→ON）（システム）

⑤ ASCIIビジーフラグがOFF→ONで、PC READ要求をON→OFF（ラダー）

⑥ PC READ命令のデータ転送が完了したら、ASCIIビジーフラグがON→OFF（システム）



●PC WRITE

ASCIIユニットは、CPUユニットからのPC WRITE要求を受けると、割付I/Oで指定されたI/Oメモリエリアへ指定変数の内容を書き込みます。

手順

① PC WRITE命令で要求

② OUT命令でPC WRITE要求ON（ラダー）

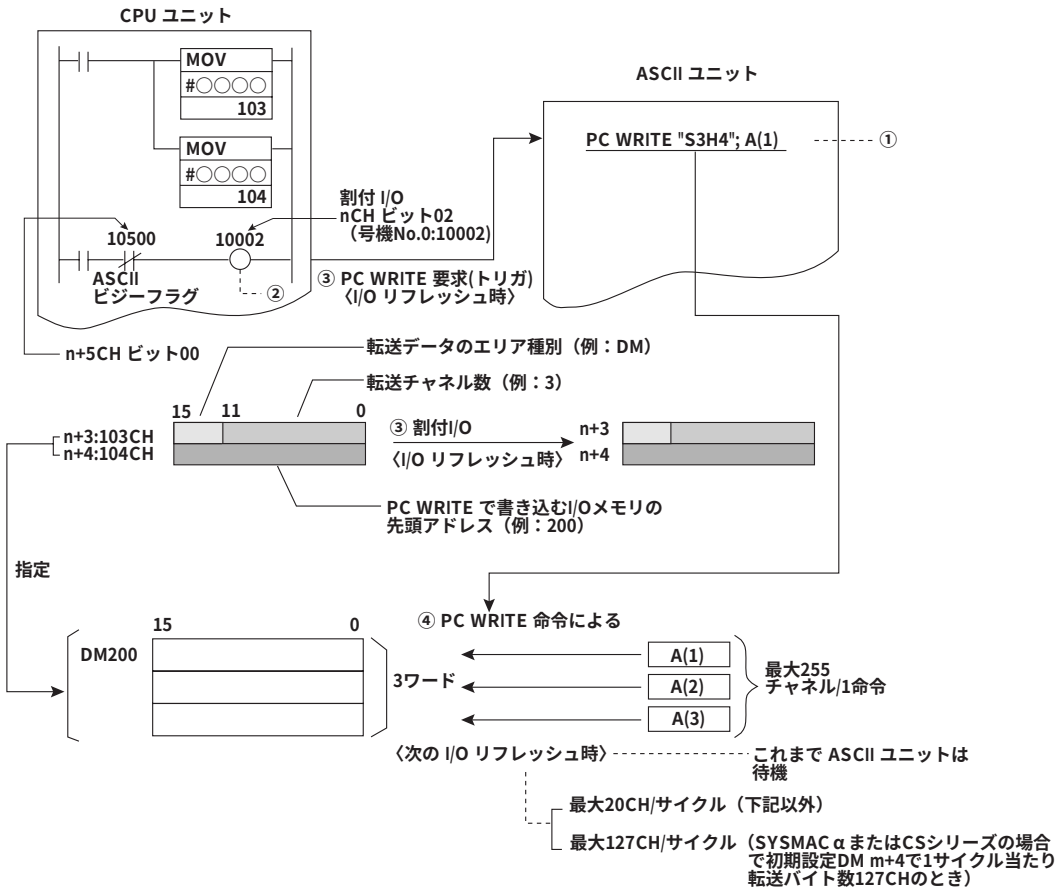
③ I/Oリフレッシュ時、CPUユニット→ASCIIユニットへPC WRITE要求（nCH、ビット02）をOFF→ONへ

④ 次のI/Oリフレッシュ時、ASCIIユニットがデータ書き込み（ASCIIビジーフラグOFF→ON）（システム）

⑤ さらに次のI/Oリフレッシュ時、ASCIIビジーフラグがOFF→ONで、PC WRITE要求をON→OFF（ラダー）

⑥ PC WRITE命令のデータ転送が完了したら、ASCIIビジーフラグがON→OFF（システム）

（CPUユニットがSYSMACαの例）



②ASCIIユニット単独で、I/Oリフレッシュ時に、CPUユニットのI/Oメモリを読み書きする

●PC READ@

ASCIIユニットは、PC READ@命令で指定されたとおり、CPUユニット内のI/Oメモリエリアを読み出します。

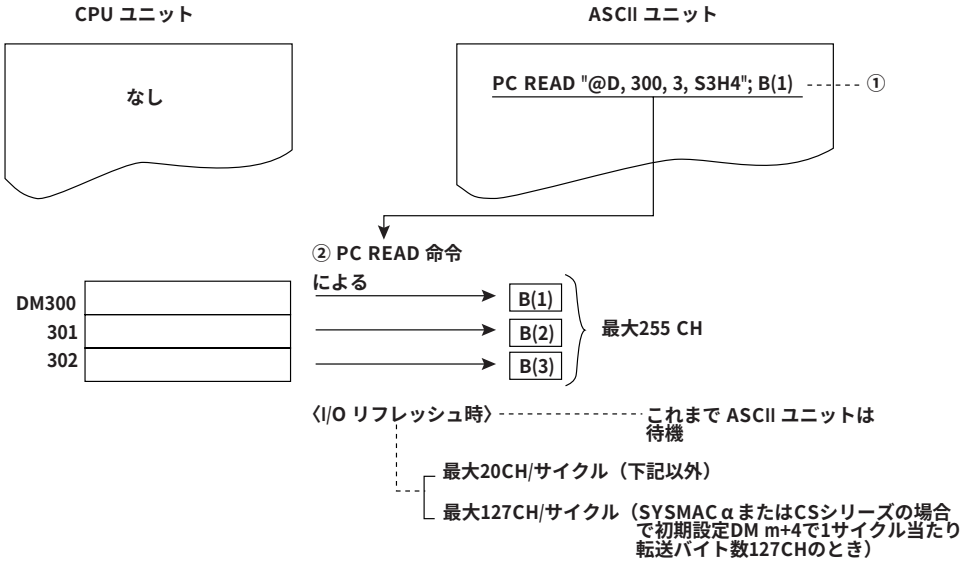
手順

① PC READ@命令実行

② I/Oリフレッシュ時、ASCIIユニットがデータ読み出し（ASCIIビジーフラグがOFF → ONへ）

③ PC READ@命令のデータ転送が完了したら、ASCIIビジーフラグがON→OFFへ

（CPUユニットがSYSMACαの例）



●PC WRITE@

ASCIIユニットは、PC WRITE @命令で指定されたとおり、CPUユニット内のI/Oメモリエリアへ書き込みます。

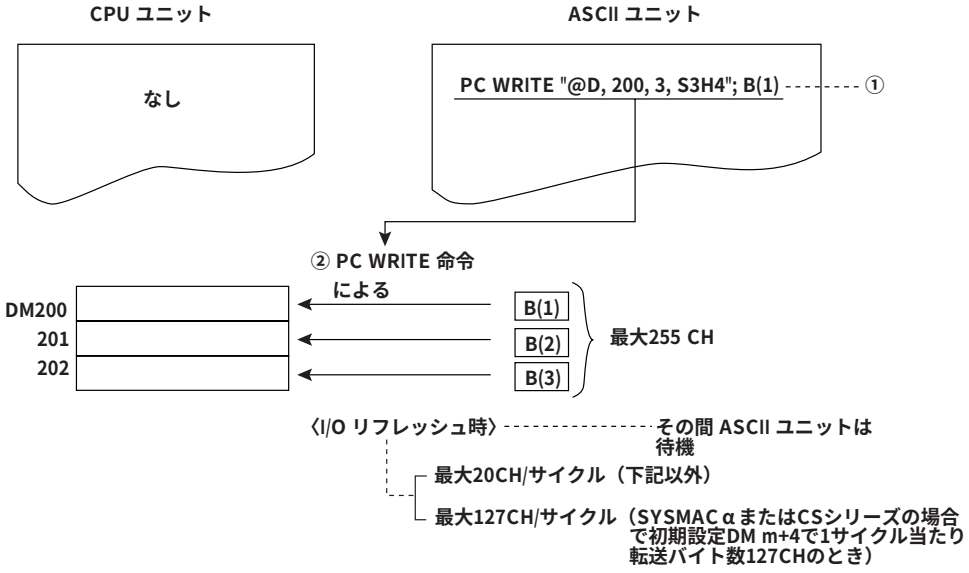
手順

① PC WRITE@命令実行

② I/Oリフレッシュ時、ASCIIユニットがデータ書き込み(ASCIIビジーフラグがOFF → ONへ)

③ PC WRITE@命令のデータ転送が完了したら、ASCIIビジーフラグがON→OFFへ

(CPUユニットがSYSMACαの例)



③ビット単位または少数ビットのデータをI/Oリフレッシュ時に読み書きする(主体はどちらでも可)

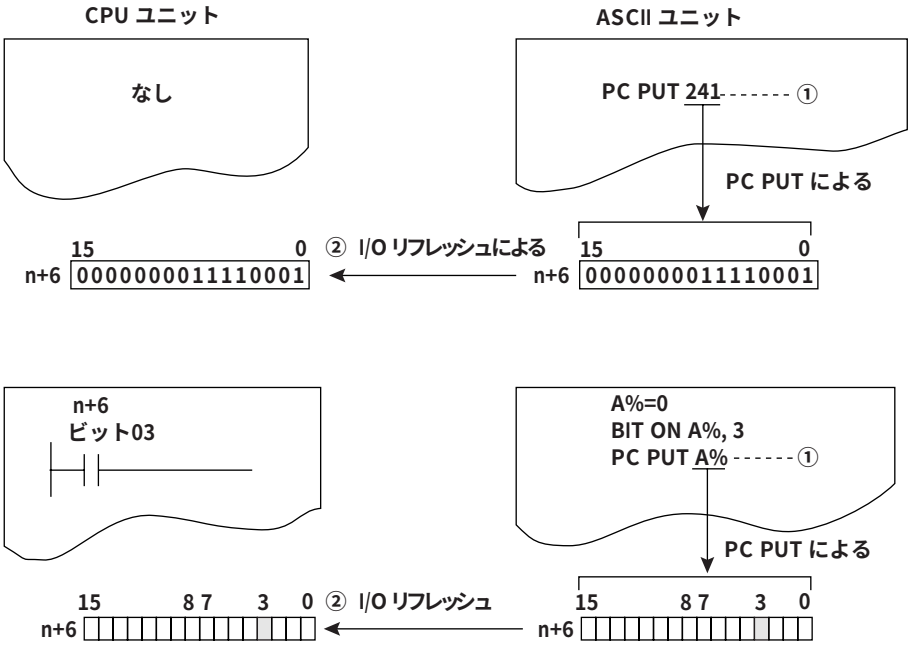
●PC PUT

ASCIIユニットは、割付I/Oのn+6CHへ、10進数を16進数に変換して格納します。10進数で−32768～＋32767の値を書き込みます。

手順

① PC PUT命令実行

② I/Oリフレッシュ時、ASCIIユニットがデータ書き込み



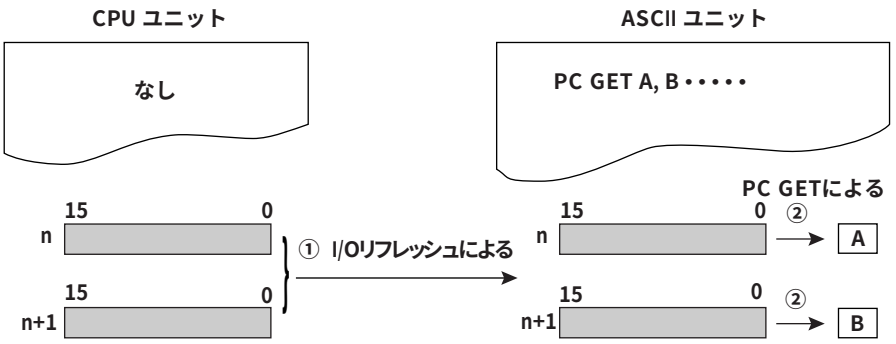
●PC GET

ASCIIユニットは、割付I/OのnCHのビット0～15の16進数を10進数に変換して第1の変数に格納します。n+1CHのビット0～15の16進数を10進数に変換して、第2の変数に格納します。

手順

① I/Oリフレッシュ時、データセット

② PC GET命令実行で、セットされたデータをASCIIユニットが読み込み



④CPUユニットとASCIIユニットが非同期で、各々ASCIIユニットの共有メモリを読み書きする

●PC EGET

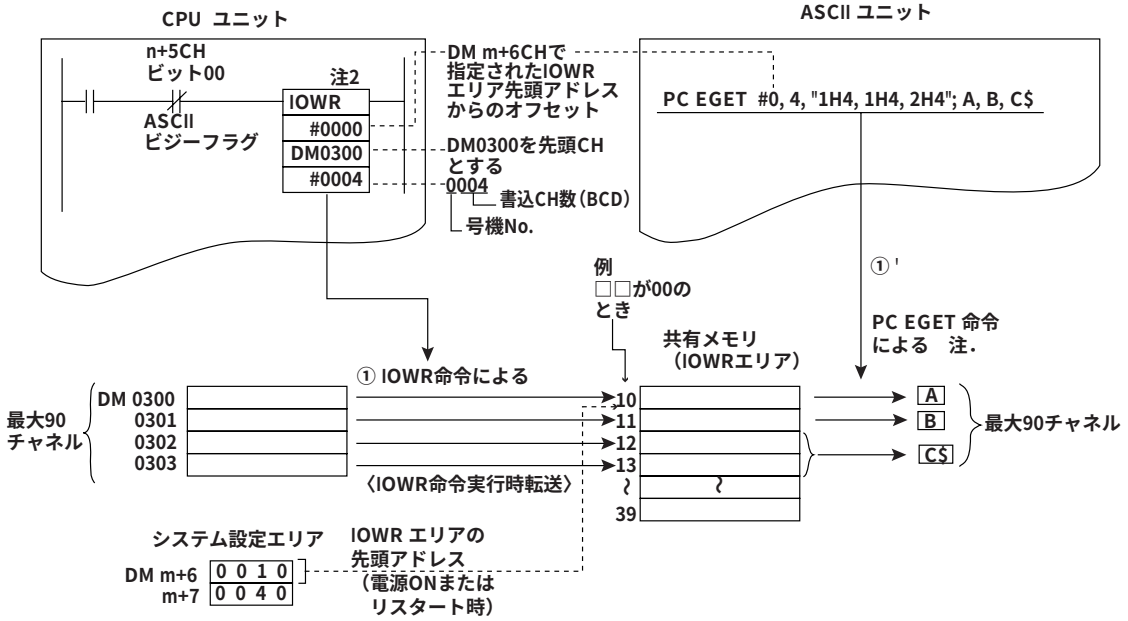
ASCIIユニットは、PC EGET命令で、自身の共有メモリのIOWRエリアを指定変数に読み出します。それと、非同期で、CPUユニットは、IOWR命令でASCIIユニットの共有メモリに指定I/Oメモリエリアのデータを書き込みます。

手順

①　IOWR(＃00□□)命令により、ASCIIユニットの共有メモリへデータを書き込み

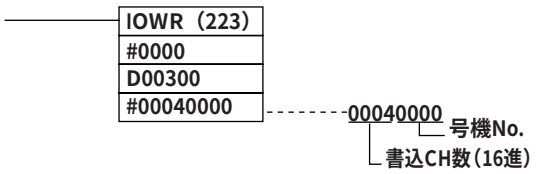
②　PC EGET命令により、共有メモリからデータを読み出し

（CPUユニットがSYSMACαの例）



注1：PC EGET／EPUT命令は、PCからの割込によって実行することで、PCとの同期をとることができます。PCからの割込には、IOWR命令(コントロールコード＃CC00)による方法と、割付I/O(nCHのビット04とn+2CHのビット0～7)による方法があります。

注2：CPUユニットがCSシリーズの場合、IOWR命令の指定は次のようになります。書込チャネル数の指定がBCDでなく、16進であることにご注意ください。



●PC EPUT

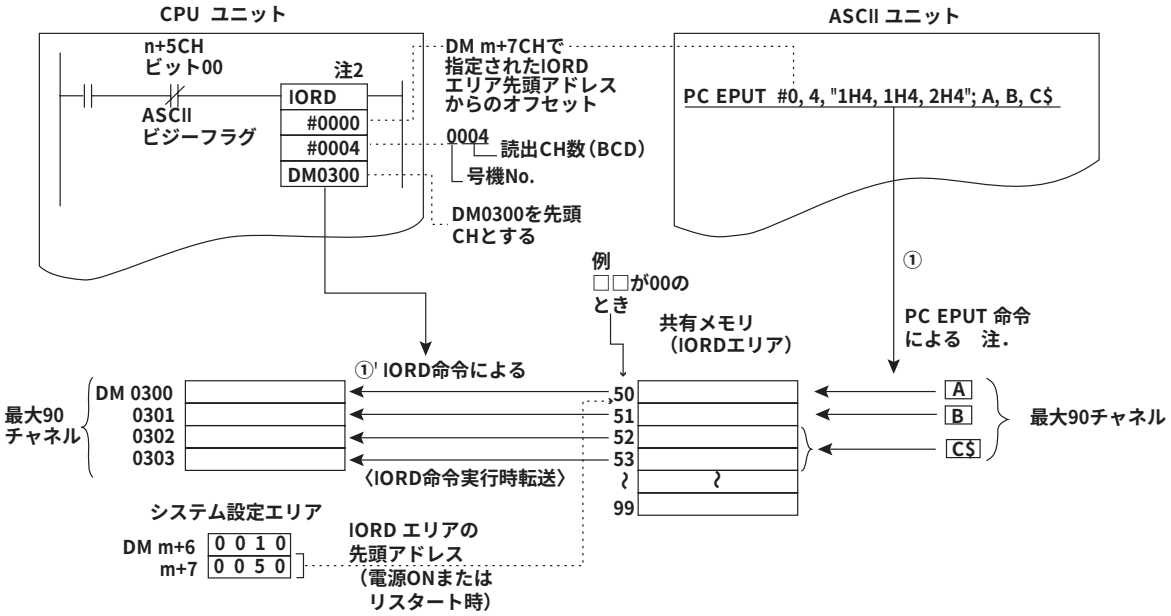
ASCIIユニットは、PC EPUT命令で、自身の共有メモリのIORDエリアに指定変数の内容を書き込みます。それと、非同期で、CPUユニットは、IORD命令でASCIIユニットの共有メモリから指定I/Oメモリエリアへデータを読み出します。

手順

① PC EPUT命令により、共有メモリへデータを書き込み

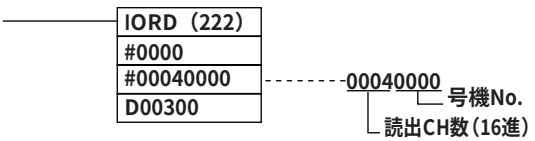
② IORD(＃00□□)命令により、ASCIIユニットの共有メモリからデータを読み出し(CPUユニットがSYSMACαの例)

(CPUユニットがSYSMACαの例)



注1：PC EGET／EPUT命令は、PCからの割込によって実行することで、PCとの同期をとることができます。PCからの割込には、IOWR命令(コントロールコード＃CC00)による方法と、割付I/O(nCHのビット04とn+2CHのビット0～7)による方法があります。

注2：CPUユニットがCSシリーズの場合、IORD命令の指定は次のようになります。読出チャネル数の指定がBCDでなく、16進であることにご注意ください。



⑤ASCIIユニットからのトリガで、CPUユニットがIOWR/IORD(#FD00)命令実行時にI/Oメモリのデータをasciiユニットに対して読み書きする

●PC QREAD(IOWR(#FD00)と組み合わせ)

ASCIIユニットは、割付I/Oのn+5CHのビット15でCPUユニットに対して、IOWR命令の実行を要求します。CPUユニットは、それを入力条件として、IOWR(#FD00)命令を実行します。割付I/Oのn+8CH、n+9CHにもとづいて、指定I/OメモリエリアをASCIIユニットへ書き込みます。

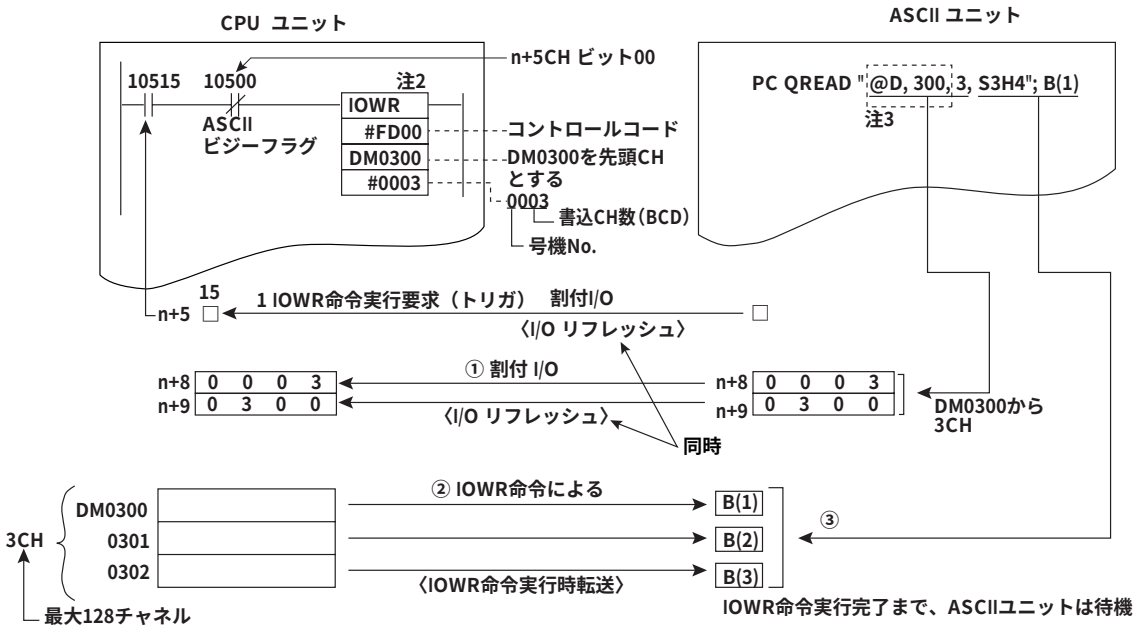
手順

① I/Oリフレッシュ時に、ASCIIユニット→CPUユニットへIOWR命令を実行要求(システム)

② IOWR(#FD00)命令により、ASCIIユニットへデータ書き込み(ラダー)

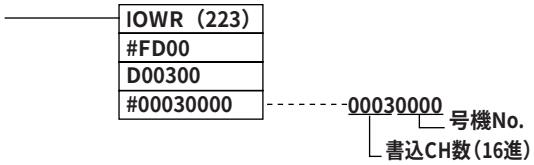
③ PC QREAD命令により、受信したデータを指定変数に読み込み

(CPUユニットがSYSMACαの例)



注1：リモートI/O上ではIORD/IOWR命令は使用できません。

注2：CPUユニットがCSシリーズの場合、IOWR命令の指定は次のようになります。書込チャネル数の指定がBCDでなく、16進であることにご注意ください。



注3：PC QREAD命令の実行によってセットされる割付リレーエリア(n+8、n+9CH)のデータは、CPUユニットにとっては、モニタ用です。実際の転送は、 IOWR命令のオペランド指定にしたがいます。この点は、PC READ@命令と異なります(PC READ@命令は、自身で指定したI/Oメモリエリアからデータを転送します)。
したがって、CSシリーズの場合、PC QREAD命令によって指定できるエリア範囲とIOWR命令によって指定できるエリア範囲が異なるため、PC QREAD命令によっては指定できないエリアを転送するときは、 PC QREAD命令にダミーのエリアを指定してください(その場合、割付リレーエリアを使用したCPUユニットでのモニタは行えません)。
例えば、上記の例で、先頭アドレスがD07000の場合、PC QREAD命令ではD07000を指定できませんが、IOWR命令では指令可能です。その場合、 PC QREAD命令には、@D,0などのダミーの指定を行ってください。

●PC QWRITE（IORD（#FD00）と組み合わせ）

ASCIIユニットは、割付I/Oのn+5CHのビット14でCPUユニットに対して、IORD命令の実行を要求します。CPUユニットは、それを入力条件として、IORD命令を実行します。割付I/Oのn+8CH、n+9CHにもとづいて、指定I/Oメモリエリアへ ASCIIユニットから読み出します。

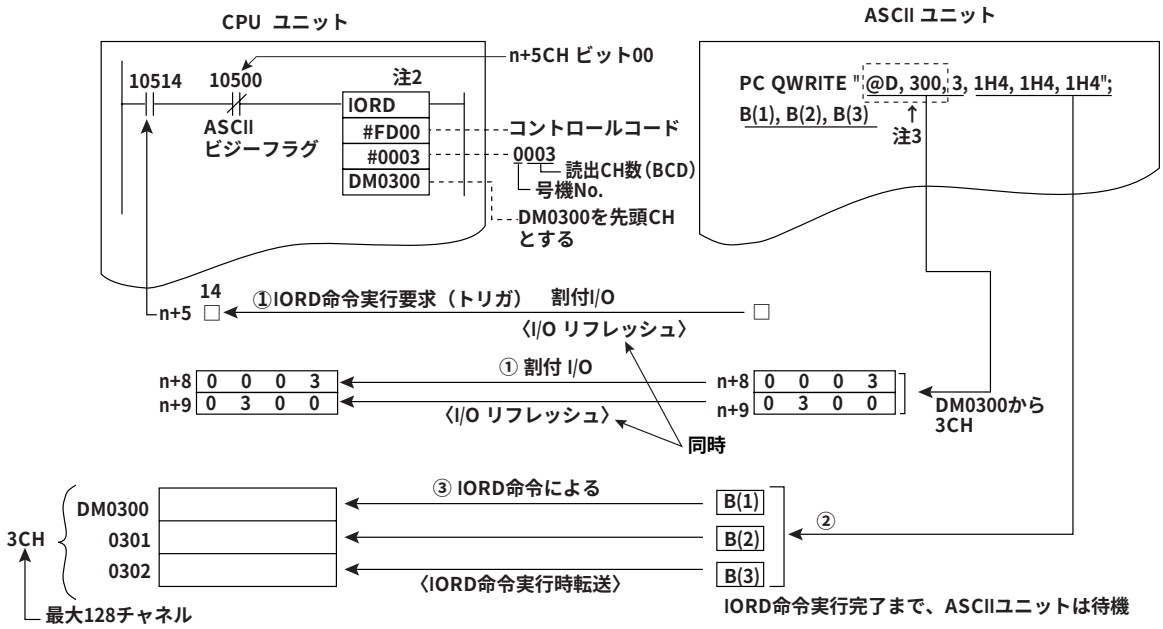
手順

① I/Oリフレッシュ時に、ASCIIユニット→CPUユニットへIORD命令を実行要求（システム）

② PC QWRITE命令により、データを指定変数から送信用意。

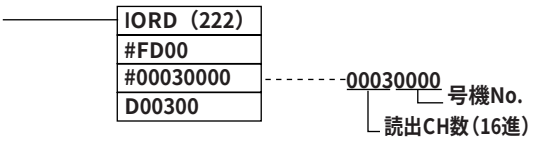
③ IORD（#FD00）命令により、ASCIIユニットからデータ読み出し（ラダー）

（CPUユニットがSYSMACαの例）



注1：リモート I/O上ではIORD/IOWR命令は使用できません。

注2：CPUユニットがCSシリーズの場合、IORD命令の指定は次のようになります。読出チャネル数の指定がBCDでなく、16進であることにご注意ください。



注3：PC QWRITE命令の実行によってセットされる割付リレーエリア（n+8、n+9CH）のデータは、CPUユニットにとっては、モニタ用です。実際の転送は、 IORD命令のオペランド指定にしたがいます。この点は、PC WRITE@命令と異なります（PC WRITE@命令は、自身で指定したI/Oメモリエリアからデータを転送します）。
したがって、CSシリーズの場合、PC QWRITE命令によって指定できるエリア範囲とIORD命令によって指定できるエリア範囲が異なるため、PC QWRITE命令によっては指定できないエリアを転送するときは、 PC QWRITE命令にダミーのエリアを指定してください（その場合、割付リレーエリアを使用したCPUユニットでのモニタは行えません）。
例えば、上記の例で、先頭アドレスがD07000の場合、PC QWRITE命令ではD07000を指定できませんが、IORD命令では指令可能です。その場合、 PC QWRITE命令には、@D,0などのダミーの指定を行ってください。

⑥CPUユニットがASCIIユニットの共有メモリへ書き込み後、ASCIIユニットへ割込をかけて、割込サブルーチン内で、ASCIIユニットが共有メモリから読み出す

注：PC割込については、6-4項「データ交換のタイミングチャート」の⑨を参照してください。

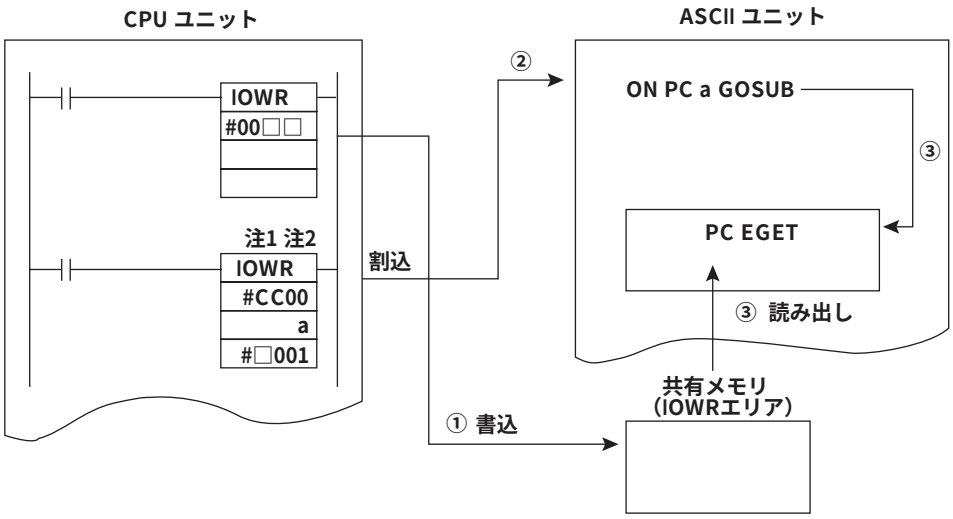
手順

① CPUユニットは、IOWR命令（＃00□□）でASCIIユニットの共有メモリ（IOWRエリア）へ書き込み（ラダー）

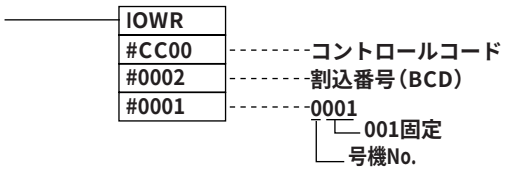
② CPUユニットは、IOWR命令（#CC00）で、ASCIIユニットへ割込（ラダー）

③ 割込を受けて、ASCIIユニット内で、PC EGET命令でデータ読み出し

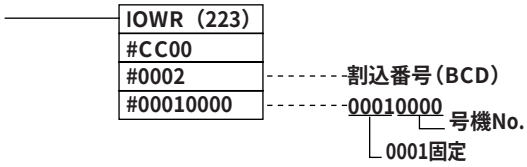
（CPUユニットがSYSMACαの例）



注1：ASCIIユニットが0号機で、割込番号が2の場合、IOWR命令の指定は、次のようになります。



注2：CPUユニットがCSシリーズの場合、IOWR命令の指定は、次のようになります。



⑦CPUユニットがASCIIユニットへ割込をかけ、ASCIIユニットは割込を受けて、共有メモリにデータを書き込む
割込サブルーチン終了後、CPUユニットは共有メモリからデータを読み出す

注：PC割込については、6-4項「データ交換のタイミングチャート」の⑨を参照してください。

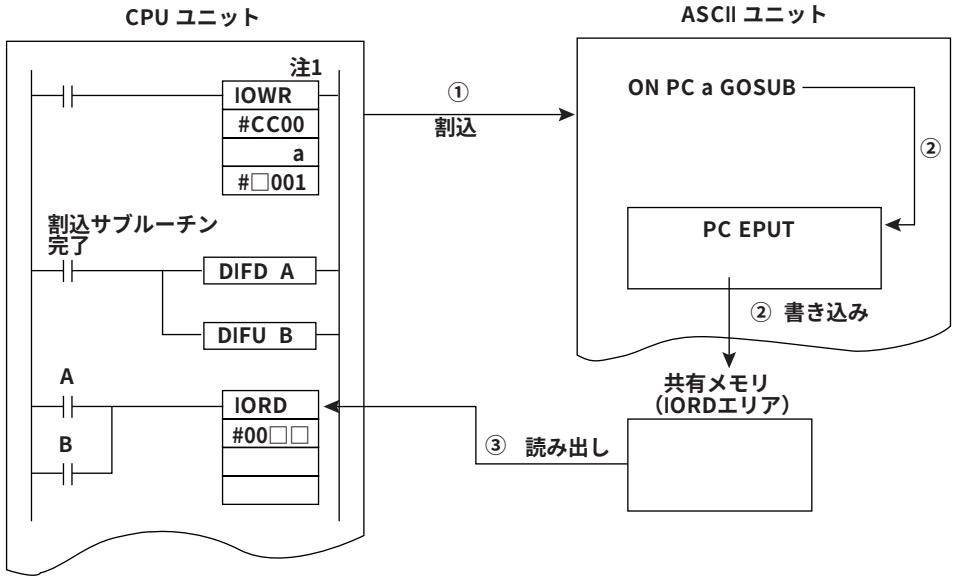
手順

① CPUユニットは、IOWR命令(#CC00)で、ASCIIユニットへ割込(ラダー)

② ASCIIユニットは、割込を受けて、PC EPUT命令で共有メモリにデータ書き込み

③ PC割込サブルーチン完了フラグの変化(1→0、または0→1)を見て、IORD命令(＃00□□)で共有メモリから読み出し(ラダー)

(CPUユニットがSYSMACαの例)

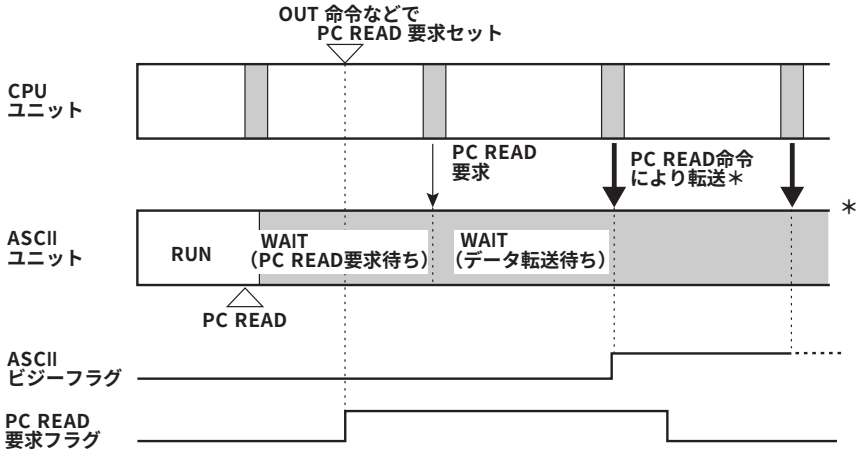


注1：⑥項の注1、注2を参照してください。

6−4　データ交換のタイミングチャート

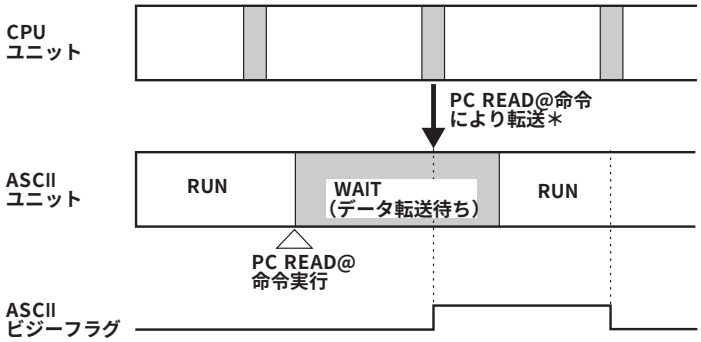
①PC READ／PC WRITE命令のタイミング

CPUユニットからトリガをかけて、直後のI/Oリフレッシュの次のI/Oリフレッシュ時にデータ転送されます。ASCIIユニットは、PC READ命令またはPC WRITE命令を実行してから、データ転送終了までBASICプログラムの実行を待機します。



②PC READ@／PC WRITE@命令のタイミング

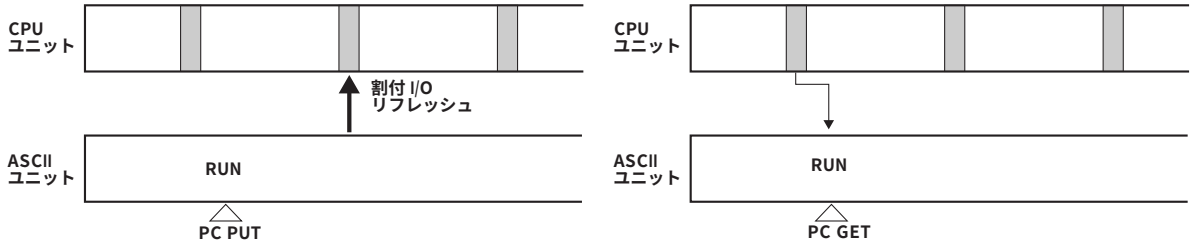
PC READ@／PC WRITE@命令実行直後のI/Oリフレッシュ時にデータ転送されます。ASCIIユニットは、PC READ@／PC WRITE@命令実行からデータ転送終了まで、BASICプログラムの実行を待機します。



③PC PUT命令／PC GET命令のタイミング

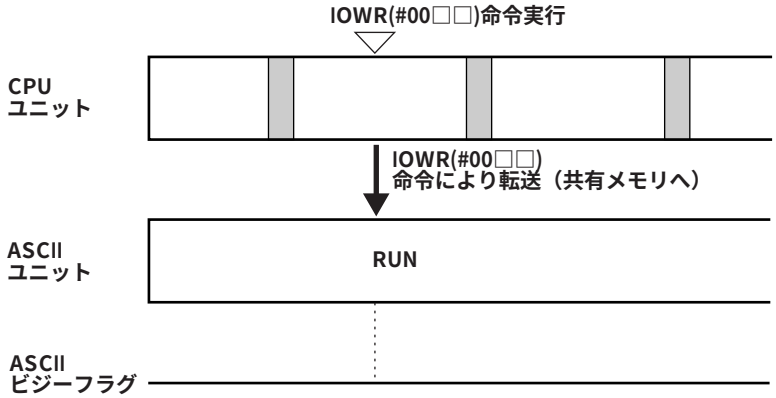
PC PUT命令は、命令実行直後のI/Oリフレッシュ時にデータを書き込みます。ASCIIユニットのBASICプログラムは待機しません。

PC GET命令は、命令実行直前のI/Oリフレッシュ時のデータを読み込みます。



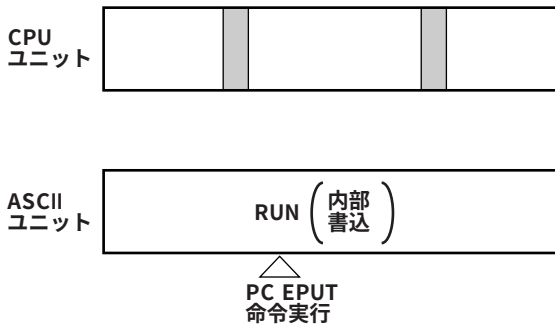
④IOWR(#00□□)/IORD(#00□□)命令のタイミング

IOWR(#00□□)/IORD(#00□□)命令実行時に、すぐデータ転送されます。ASCIIユニットのBASICプログラムは待機しません。



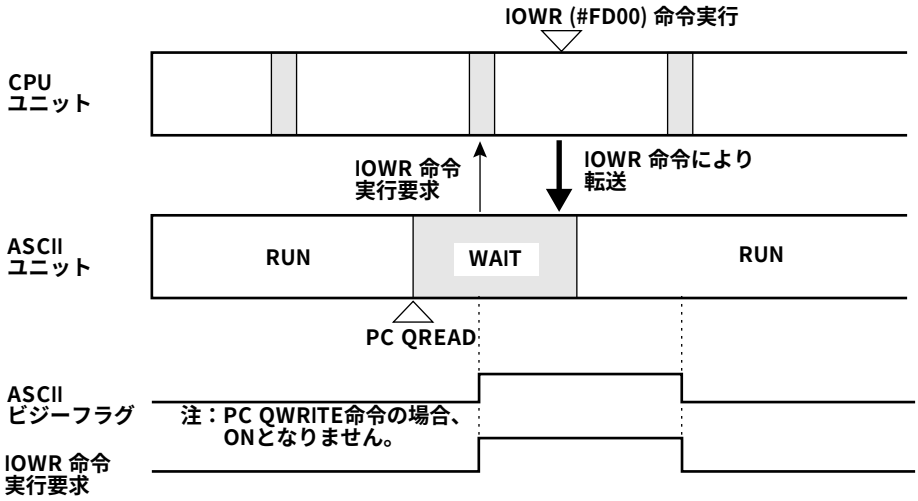
⑤PC EPUT／PC EGET命令のタイミング

PC EPUT／PC EGET命令実行時にすぐ内部共有メモリへ読み書きされます。ASCIIユニットのBASICプログラムは待機しません。



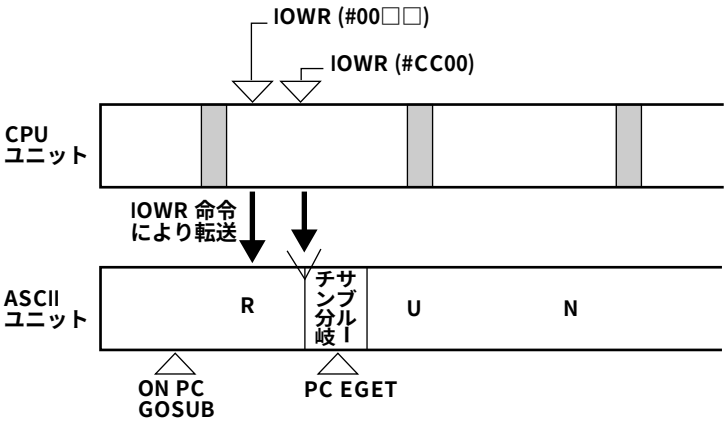
⑥IOWR(#FD00)/IORD(#FD00)と、PC QREAD／PC QWRITE命令のタイミング

ASCIIユニットからトリガが入ると、IOWR(#FD00)/IORD(#FD00)命令実行時に、すぐデータが転送されます。ASCIIユニットは、PC QREAD／PC QWRITE命令実行からIOWR(#FD00)/IORD(#FD00)命令実行まで待機します。



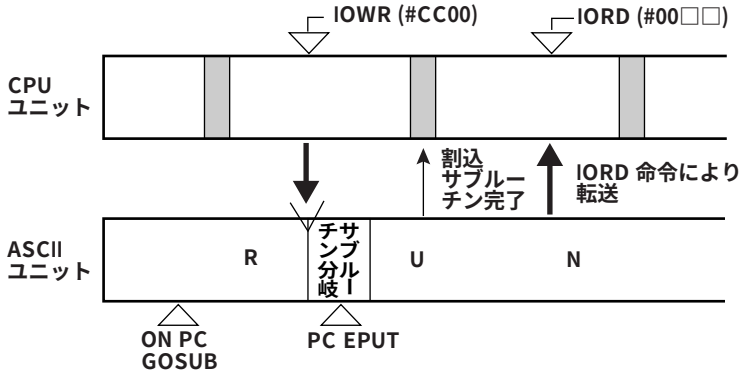
⑦IOWR(#00□□)、IOWR(#CC00)、ON　PC GOSUB、PC EGET命令の組み合わせ時のタイミング

IOWR(＃00□□)命令ですぐデータ転送。IOWR(#CC00)命令でASCIIユニットへ割込み。割込サブルーチンに入り、PC EGET命令でASCIIユニットは、データを入手します。CPUユニットの1サイクル内でデータ転送が完了します。



⑧IOWR(#CC00)、ON PC GOSUB、PC EPUT、IORD(#00□□)命令の組み合わせ時のタイミング

IOWR(#CC00)命令でASCIIユニットへ割込。すぐに割込サブルーチンに入り、PC EPUT命令で内部共有メモリへ書き込み。次のI/Oリフレッシュ時に、CPUユニットに対して、割込サブルーチン終了を伝え、次サイクルでIORD(＃00□□)命令でデータが転送されます。2～3サイクルでデータ転送が完了します。

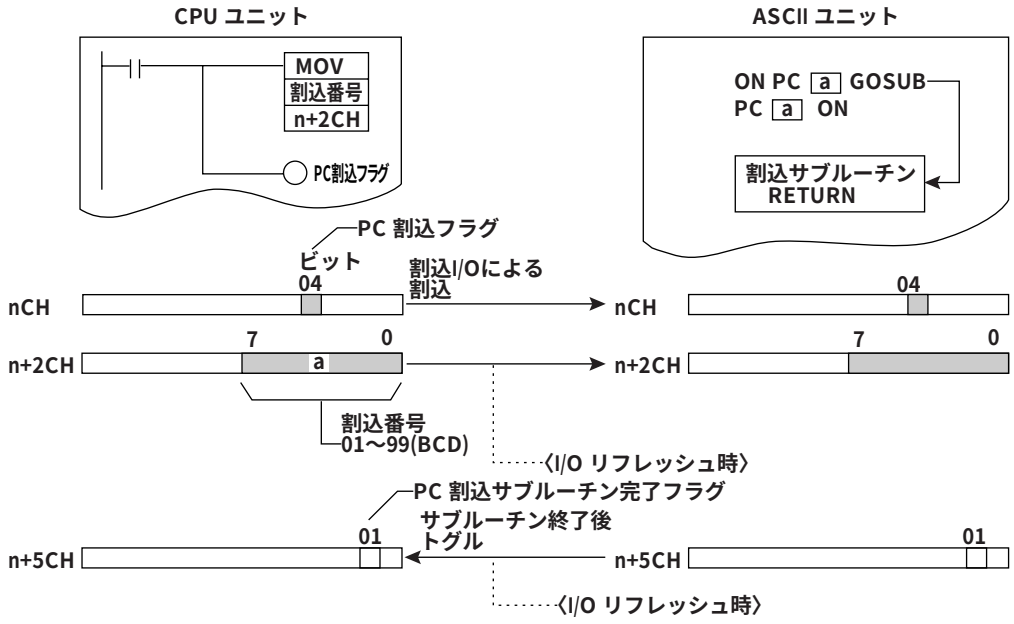


⑨CPUユニットからASCIIユニットに対して、割込をかける(PC割込)

次の2つの方法があります。

●PC割込フラグによる割込

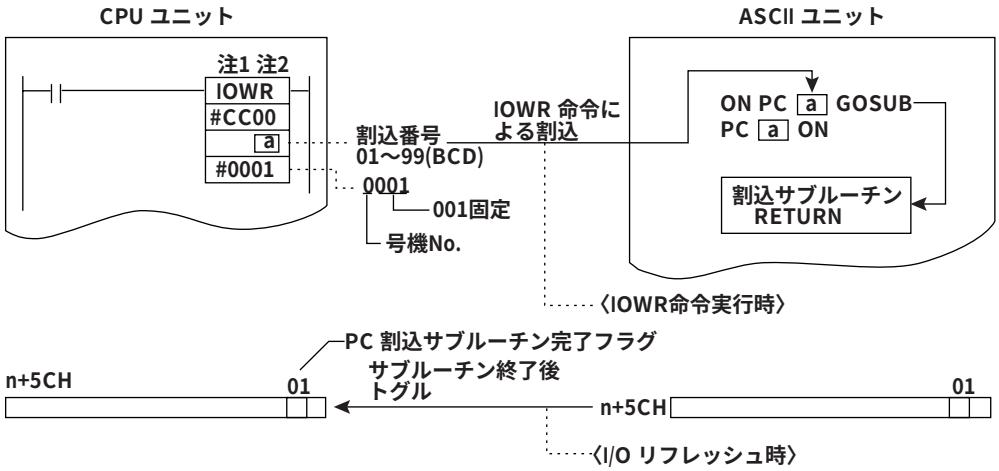
CPUユニットが、割付I/Oエリアによって、ASCIIユニットに対して割込番号をセット。かつ、PC割込フラグを0→1に立ち上げ、割込をかけます。ASCIIユニットは、ON PC命令の割込番号と一致していると、割込サブルーチンを実行します。割込サブルーチンが終了すると、PC割込サブルーチン完了フラグがトグル(1→0、0→1)します。



●IOWR命令（コントロールコード：#CC00）による割込（SYSMACαシリーズ、CSシリーズのみ使用可）

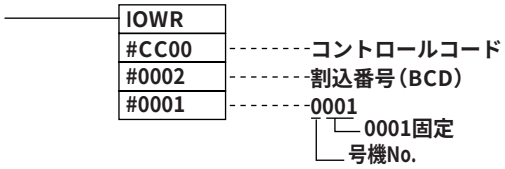
CPUユニットは、IOWR（#CC00）命令で、ASCIIユニットに対して割込番号を指示して、割込をかけます。ASCIIユニットは、ON PC命令の割込番号と一致している割込サブルーチンを実行します。PC割込サブルーチンが終了すると、PC割込サブルーチン完了フラグがトグル（0→1、1→0）します。ただし、この方法は、SYSMACα、CSシリーズの場合のみ可能です。

（CPUユニットがSYSMACαの例）

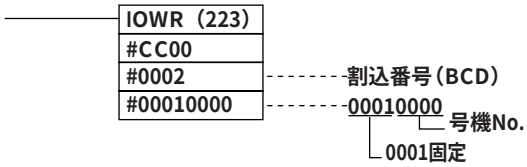


この方法による割込は、PC割込フラグによる方法（前ページ）がI/Oリフレッシュ時に割込をかけるのに対して、IOWR命令実行時に割込をかけます。そのため、次のサイクルでなく、そのときのサイクル内でASCIIユニットに対して、高速の割込が可能です。CPUユニット側のIOWR／IORD（＃00□□）命令と、ASCIIユニット側の割込サブルーチン内のPC EGET／PC EPUT命令を組み合わせることで、1つのサイクル内でCPUユニットとASCIIユニット間でデータ読み書きが可能です。

注1：ASCIIユニットが0号機で、割込番号が2の場合、IOWR命令の指定は、次のようになります。



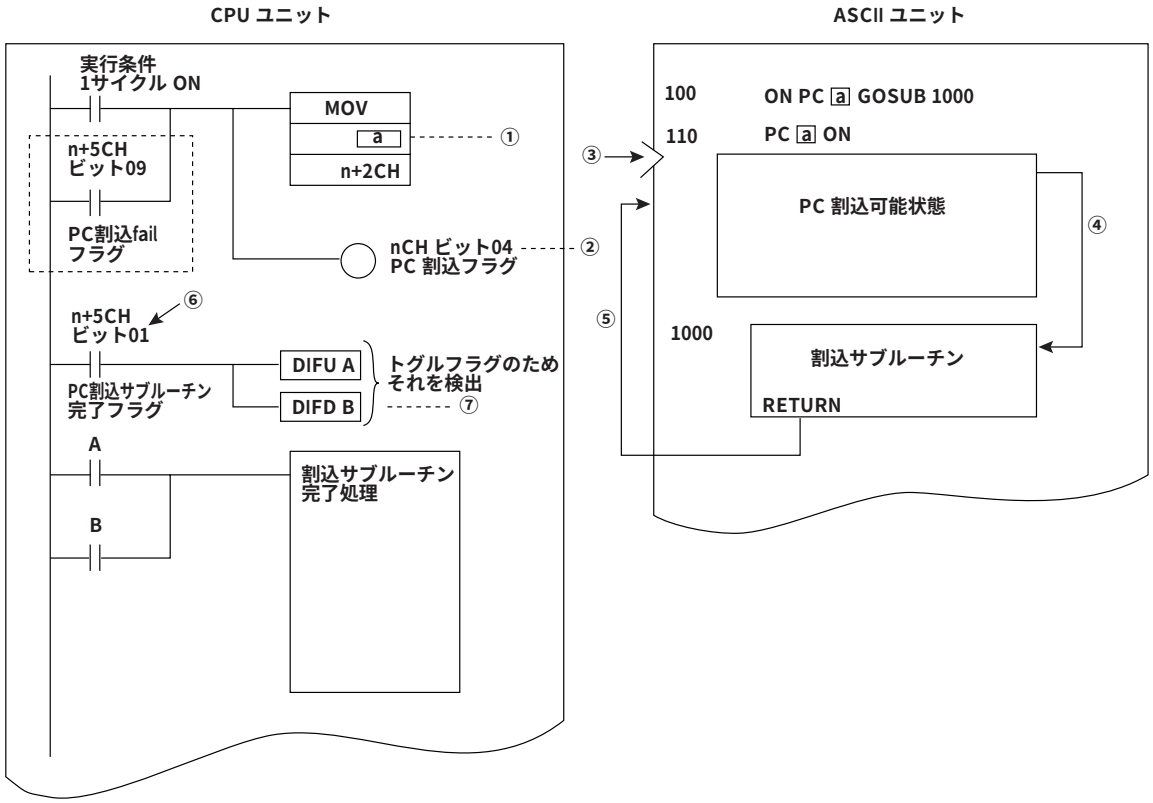
注2：CPUユニットがCSシリーズの場合、IOWR命令の指定は、次のようになります。



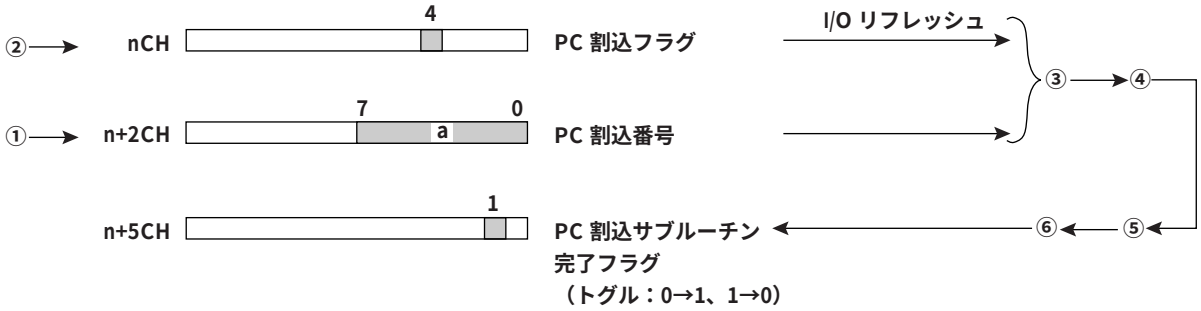
下記に、プログラム例を示します。

注：下記プログラム例は、PC割込を1つ使用する場合の例です。PC割込を複数使用する場合は、割込の発行をコントロールするためのプログラムが必要になります。

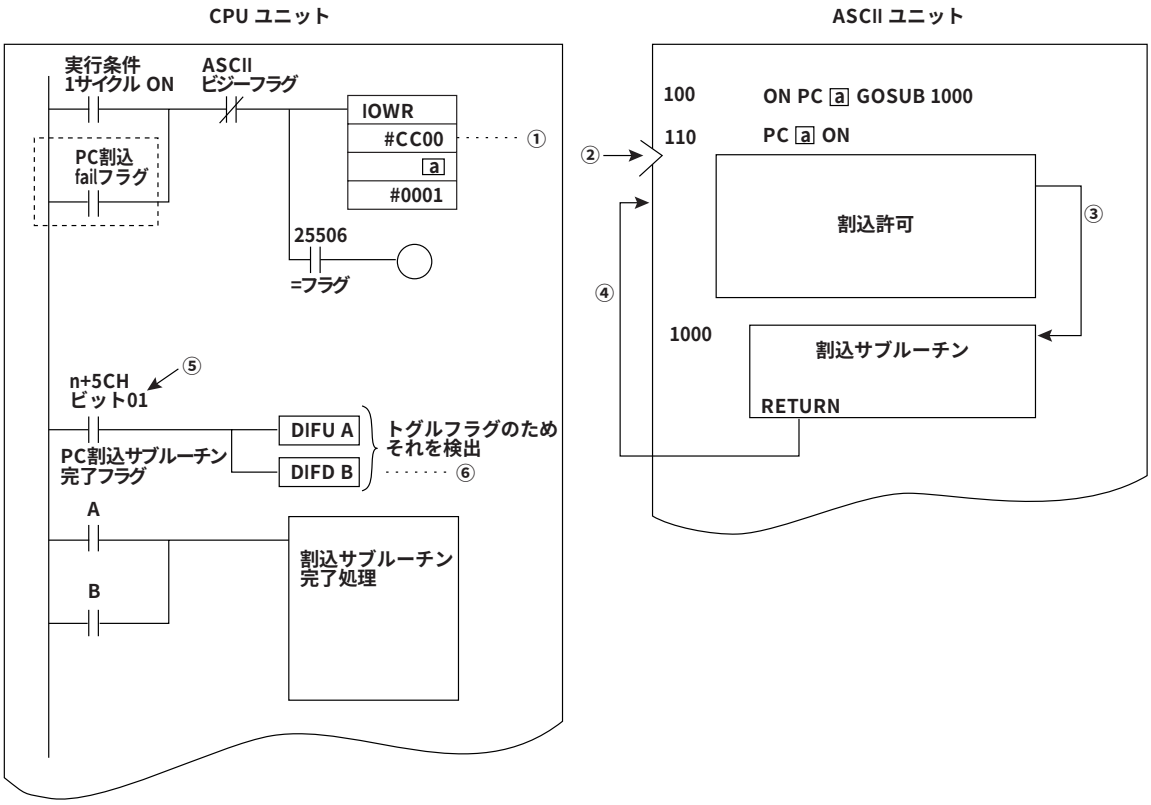
a)PC割込フラグによる割込



注. ┆┆┆ 部は、ASCII ユニットのPC OFF（割込禁止）状態などにより、PC 割込が失敗した場合のリトライ処理に相当します。PC割込failフラグ（n+5CH ビット09）は、PC割込フラグ（nCH ビット04）の立ち上がりにより割込をかけてもPC 割込がOFF状態のため受け付けられなかったとき、次のサイクルで1スキャン間1(ON)となります。



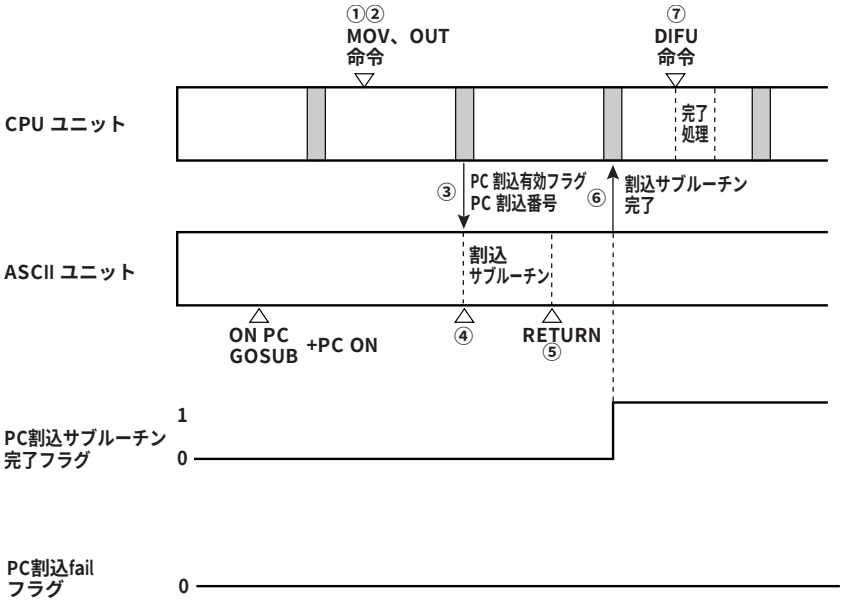
b)IOWR(#CC00)命令による割込(SYSMACα、CSシリーズのみ使用可)



注. [ ] 部は、ASCII ユニットのPC OFF（割込禁止）状態などにより、PC 割込が失敗した場合のリトライ処理に相当します。PC割込failフラグは、IOWR(#CC00)命令により割込をかけても受け付けられなかったとき、次のサイクルで1サイクル間1（ON）となります。割込失敗時のPC割込failフラグの動作は、後述のようになります。

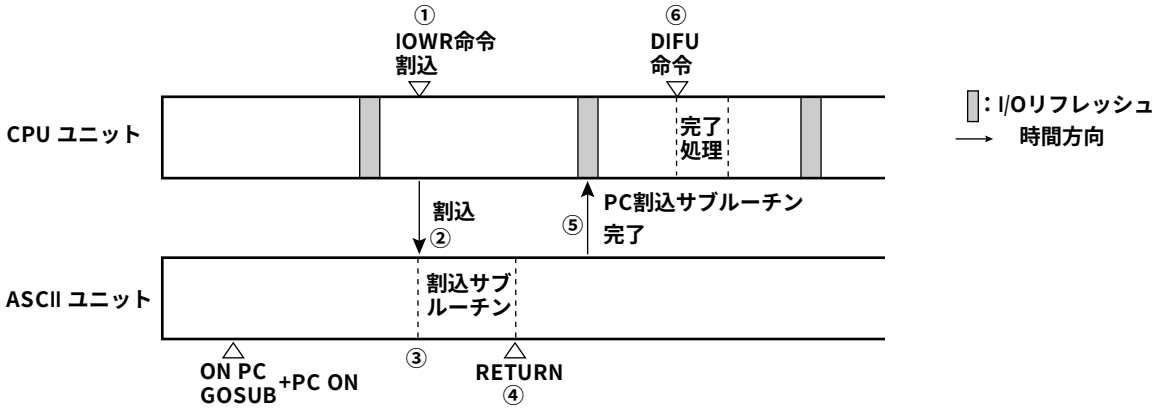
●タイミングチャート

割込実行時＜ a)PC割込フラグによる割込＞



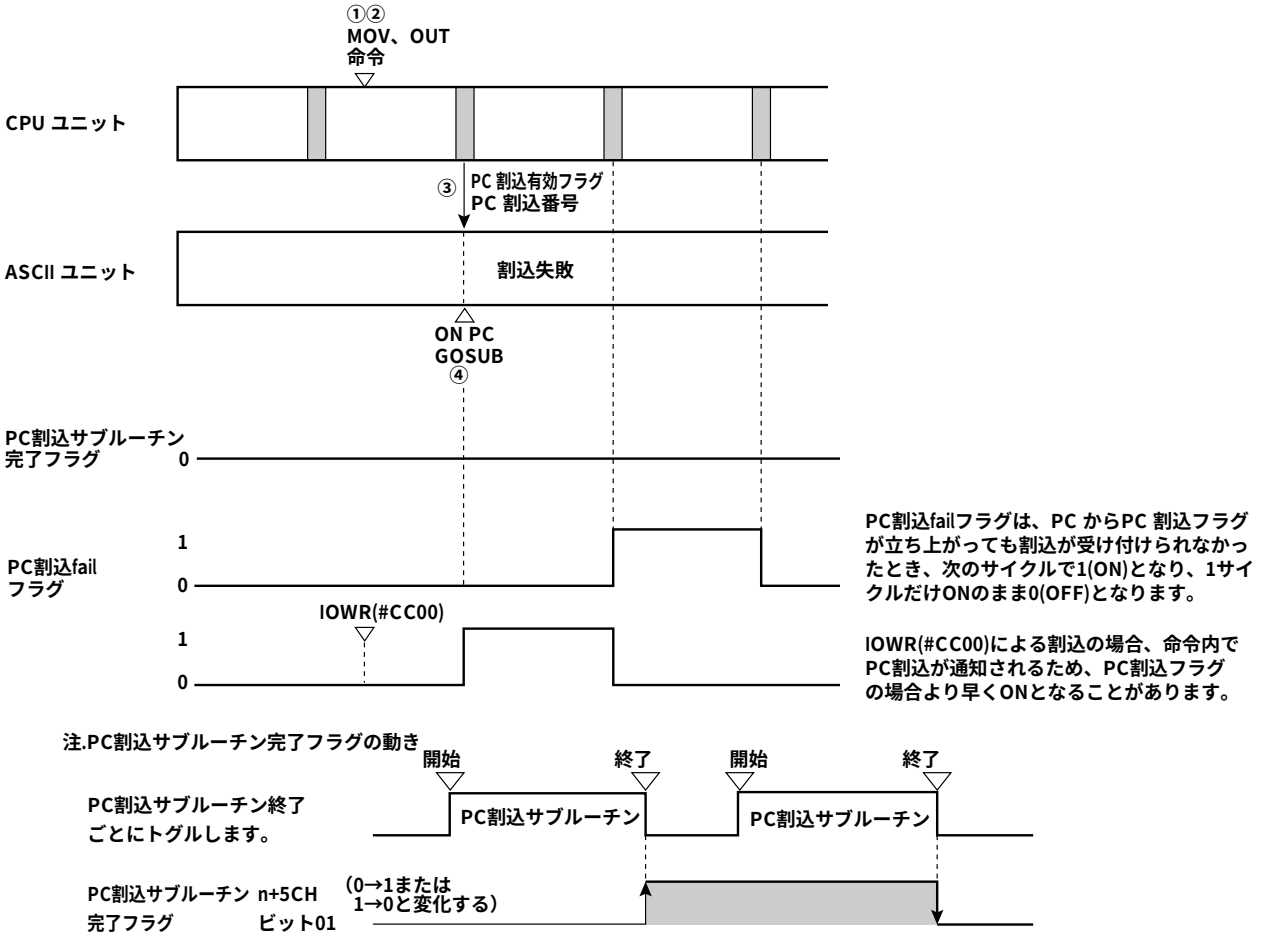
6－4　データ交換のタイミングチャート

割込実行時＜ b)IOWR(#CC00)命令による割込＞



上記のように、IOWR 命令(#CC00) による割込は、PC 割込フラグによる割込にくらべ、1サイクル短い速度で割込処理を行えます。

割込失敗時＜ a)PC割込フラグによる割込、b)IOWR(#CC00)命令による割込で共通＞



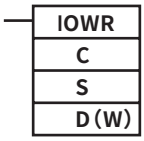
注.PC割込サブルーチン完了フラグの動き

注：このタイミングチャートは、フラグ動作を意味するもので、PC上のフラグ動作とは一致しません。

6−5　IOWR命令／IORD命令の仕様

本命令は、CPUユニットがCSシリーズまたはSYSMACαシリーズ(C200HX/HG/HE-(Z))の場合だけです。また、リモートI/O子局装置上に実装されたASCIIユニットに対して本命令を使用することはできません。

■IOWR命令(インテリジェントI/O書込命令)



C：コントロールコード
S：（転送元先頭CH番号）
D(W)：転送先号機No.、(転送CH数)　(　)はコントロールコードによって異なります

本ASCIIユニットでは、IOWR命令の使い方には、次の4つの使い方があります。

1　ASCIIユニットの変数への書き込み

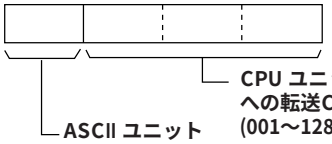
C:

値（16進4桁）	内　容	組み合わせ
#FD00	割付I/Oのn+8、n+9CHで指定されたI/Oメモリエリア をASCIIユニットの変数へ転送	ASCIIユニットのPC QREAD@命令と組み合わせて使用することが必要

S: CPUユニットのI/Oメモリの転送元CH番号

SYSMACαシリーズの場合

D: 15　12 11　0

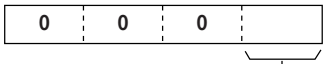


CPU ユニット→ASCII ユニットへの転送CH数 (001〜128)

ASCII ユニットの号機No. (0〜F)

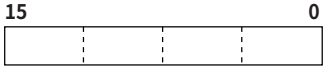
CSシリーズの場合

W: 15　4 3　0



ASCII ユニットの号機No.(0〜F)

W+1: 15　0



CPU ユニット→ASCII ユニットへの転送CH数 (0001〜0080Hex)(16進)

W+1が上位4桁
Wが下位4桁

●データ内容

機種	C200HX/HG/HE
エリア	S
リレーエリア1	000〜255
リレーエリア2	256〜511
保持リレー	HR00〜99
補助記憶リレー	AR00〜27
リンクリレー	LR00〜63
タイマ/カウンタ	T/C000〜511
一時記憶リレー	−
データメモリ	D0000〜6655
拡張データメモリ	−
間接データメモリ	＊D0000〜6655
定数	−

機種	CSシリーズ
エリア	S
CIO(入出力リレーなど)	0000〜6143
内部補助リレー	W000〜511
保持リレー	H000〜511
特殊補助リレー	A000〜959
タイマ	T0000〜4095
カウンタ	C0000〜4095
データメモリ	D00000〜32767
拡張データメモリ	E00000〜32767
拡張データメモリ（バンク指定含）	En_00000〜32767 (n=0〜C)
DM／EM間接（BIN）	@D00000〜32767 @E00000〜32767 @En_00000〜32767 (n=0〜C)
DM／EM間接（BCD）	＊D00000〜32767 ＊E00000〜32767 ＊En_00000〜32767 (n=0〜C)
定数	#0000〜FFFF (BINデータ)
データレジスタ	−
インデックスレジスタ（直接）	−
インデックスレジスタ（間接）	,IR0〜15 −2048〜+2047,IR0〜15 DR0〜15，IR0〜15 ,IR0〜15+(++) −（−−）IR0〜15

2 ASCIIユニットへの共有メモリへの書き込み

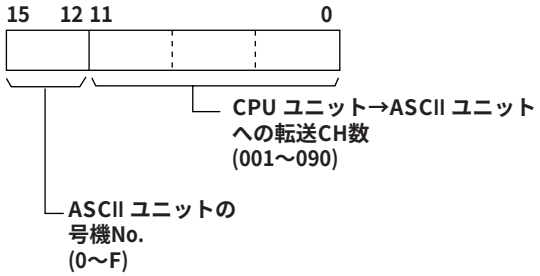
C:

値（16進4桁）	内　容	組み合わせ
#00□□	指定I/Oメモリエリアの指定CH数のデータを、ASCIIユニットの共有メモリのIOWRエリア先頭*1＋□□チャネルへの転送 ＊1：システム設定エリア DMm+6CHで指定	ASCIIユニットのPC EGET命令と組み合わせて使用可。 組み合わせず単独でも使用可。

S: CPUユニットのI/Oメモリの転送元CH番号

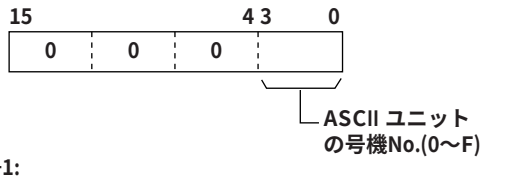
SYSMAC αシリーズの場合

D:

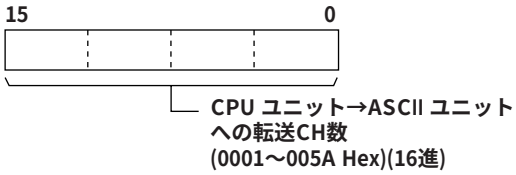


CSシリーズの場合

W:



W+1:



W+1が上位4桁
Wが下位4桁

●データ内容

機種	C200HX/HG/HE
エリア	S
リレーエリア1	000～255
リレーエリア2	256～511
保持リレー	HR00～99
補助記憶リレー	AR00～27
リンクリレー	LR00～63
タイマ/カウンタ	T/C000～511
一時記憶リレー	—
データメモリ	D0000～6655
拡張データメモリ	—
間接データメモリ	＊D0000～6655
定数	—

機種	CSシリーズ
エリア	S
CIO（入出力リレーなど）	0000～6143
内部補助リレー	W000～511
保持リレー	H000～511
特殊補助リレー	A000～959
タイマ	T0000～4095
カウンタ	C0000～4095
データメモリ	D00000～32767
拡張データメモリ	E00000～32767
拡張データメモリ（バンク指定含）	En_00000～32767（n=0～C）
DM／EM間接（BIN）	@D00000～32767 @E00000～32767 @En_00000～32767（n=0～C）
DM／EM間接（BCD）	＊D00000～32767 ＊E00000～32767 ＊En_00000～32767（n=0～C）
定数	#0000～FFFF（BINデータ）
データレジスタ	—
インデックスレジスタ（直接）	—
インデックスレジスタ（間接）	,IR0～15 －2048～+2047,IR0～15 DR0～15，IR0～15 ,IR0～15+(++) －（－－）IR0～15

3 ASCIIユニットへの割込

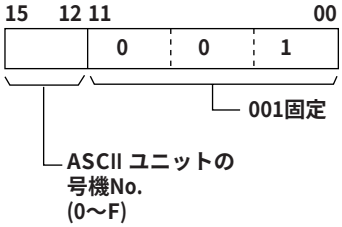
C:

値（16進4桁）	内　容	組み合わせ
#CC00	CPUユニットからASCIIユニットに対して、割込をかけます。	ASCIIユニットのON PC（割込番号）命令と組み合わせて使用することが必要です。

S: 割込番号（ASCIIユニット内のON PC（割込番号）命令に対応）（BCD）

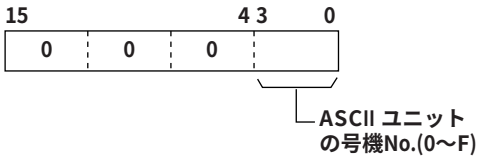
SYSMACαシリーズの場合

D:

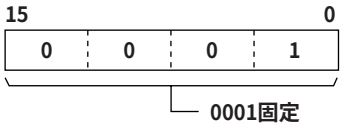


CSシリーズの場合

W:



W+1:



W+1が上位4桁
Wが下位4桁

4　ASCIIユニット内のエラーのクリア

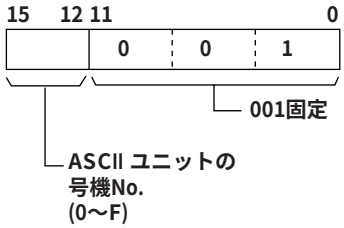
C:

値	内　容
#EC00	ASCIIユニット内で発生したエラー状態を解除します。

S: #0000固定

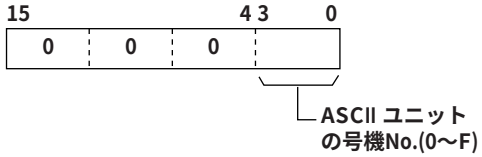
SYSMACαシリーズの場合

D:

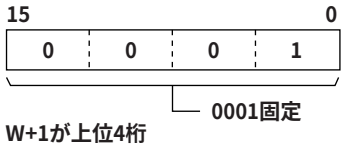


CSシリーズの場合

W:



W+1:



W+1が上位4桁
Wが下位4桁

●フラグ

値*1	ON	OFF
ER (25503)	・転送チャネル数がBCDでないとき、または0、129チャネル以上のとき ・＊DM指定時、DMの内容が6656以上またはBCDでないとき ・ASCIIユニットの号機No.が0～F以外のとき、またはSYSBUS上にあるとき ・転送チャネル数がBCDでないとき ・命令が正常に終了しなかったとき	C、S、Dの設定が正常なとき
CY(25504)	－	－
>(25505)	－	－
=(25506)	書き込みが正常に終了したとき	書き込みが正常に終了しなかったとき
<(25507)	－	－
OF(25404)	－	－
UF(25405)	－	－
N(25402)	－	－

＊1：（　）内は、SYSMACα、C200HS、C200Hでのアドレス

■IORD命令（インテリジェントI/O読出命令）

IORD
C
S（W）
D

C：コントロールコード
S：転送先号機No.、転送CH数
D：転送先開始CH番号

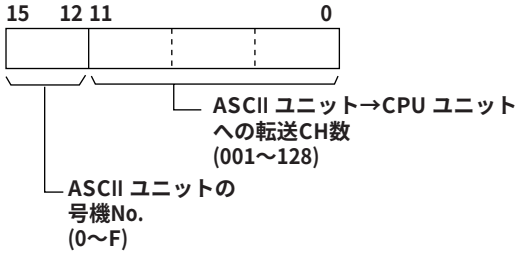
本ユニットでは、IORD命令の使い方には、次の2つの使い方があります

1　ASCIIユニットの変数からの読み出し

C:	値（16進4桁）	内　容	組み合わせ
	#FD00	割付I/Oのn+8、n+9CHで指定されたI/Oメモリエリアへ ASCIIユニットの変数から転送	ASCIIユニットのPC QWRITE@命令と組み合わせて使用要

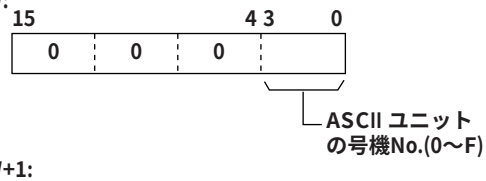
SYSMACαシリーズの場合

S:

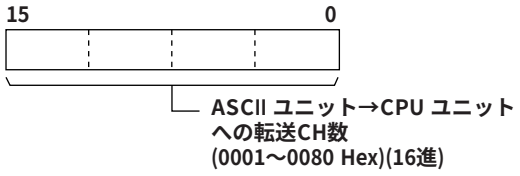


CSシリーズの場合

W:



W+1:



W+1が上位4桁
Wが下位4桁

D: CPUユニットのI/Oメモリの転送先CH番号

●データ内容

機種	C200HX/HG/HE
エリア	D
リレーエリア1	000～252
リレーエリア2	256～511
保持リレー	HR00～99
補助記憶リレー	AR00～27
リンクリレー	LR00～63
タイマ/カウンタ	T/C000～511
一時記憶リレー	－
データメモリ	D0000～6143
拡張データメモリ	－
間接データメモリ	＊D0000～6655
定数	－

機種	CSシリーズ
エリア	D
CIO（入出力リレーなど）	0000～6143
内部補助リレー	W000～511
保持リレー	H000～511
特殊補助リレー	A448～959
タイマ	T0000～4095
カウンタ	C0000～4095
データメモリ	D00000～32767
拡張データメモリ	E00000～32767
拡張データメモリ（バンク指定含）	En_00000～32767 (n=0～C)
DM／EM間接（BIN）	@D00000～32767 @E00000～32767 @En_00000～32767 (n=0～C)
DM／EM間接（BCD）	＊D00000～32767 ＊E00000～32767 ＊En_00000～32767 (n=0～C)
定数	—
データレジスタ	—
インデックスレジスタ（直接）	—
インデックスレジスタ（間接）	,IR0～15 －2048～+2047,IR0～15 DR0～15，IR0～15 ,IR0～15+(++) －（－－）IR0～15

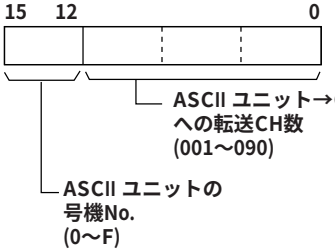
2　ASCIIユニットの共有メモリからの読み出し

C:

値（16進4桁）	内　　容	組み合わせ
#00□□	ASCIIユニットの共有メモリのIORDエリア先頭*1＋□□チャネルから指定CH数のデータを、I/Oメモリエリアの指定CH番号へ転送 ＊1：システム設定エリア DMm+7CHで指定	ASCIIユニットのPC EPUT命令と組み合わせて使用可。 組合わせず単独でも使用可。

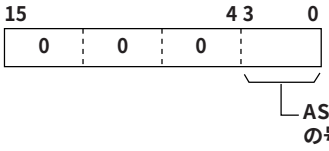
SYSMACαシリーズの場合

S:

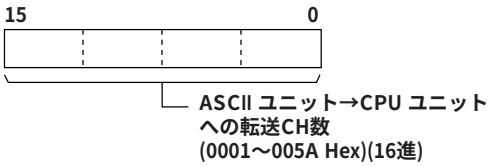


CSシリーズの場合

W:



W+1:



W+1が上位4桁
Wが下位4桁

D: CPUユニットのI/Oメモリの転送先CH番号

●データ内容

機種	C200HX/HG/HE
エリア	D
リレーエリア1	000～252
リレーエリア2	256～511
保持リレー	HR00～99
補助記憶リレー	AR00～27
リンクリレー	LR00～63
タイマ/カウンタ	T/C000～511
一時記憶リレー	－
データメモリ	D0000～6143
拡張データメモリ	－
間接データメモリ	＊D0000～6655
定数	－

機種	CSシリーズ
エリア	D
CIO（入出力リレーなど）	0000～6143
内部補助リレー	W000～511
保持リレー	H000～511
特殊補助リレー	A448～959
タイマ	T0000～4095
カウンタ	C0000～4095
データメモリ	D00000～32767
拡張データメモリ	E00000～32767
拡張データメモリ（バンク指定含）	En_00000～32767 (n=0～C)
DM／EM間接（BIN）	@D00000～32767 @E00000～32767 @En_00000～32767 (n=0～C)
DM／EM間接（BCD）	＊D00000～32767 ＊E00000～32767 ＊En_00000～32767 (n=0～C)
定数	—
データレジスタ	—
インデックスレジスタ（直接）	—
インデックスレジスタ（間接）	,IR0～15 －2048～+2047,IR0～15 DR0～15，IR0～15 ,IR0～15+(++) －（－－）IR0～15

●フラグ

値*1	ON	OFF
ER (25503)	・転送チャネル数がBCDでないとき、または0、129チャネル以上のとき ・＊DM指定時DMの内容が6656以上またはBCDでないとき ・転送元号機No.が0～F以外のときまたはSYSBUS上にあるとき ・転送チャネル数がBCDでないとき	C、S、Dの設定が正常なとき
CY(25504)	－	－
>(25505)	－	－
=(25506)	読み出しが正常に終了したとき	読み出しが正常に終了しなかったとき
<(25507)	－	－
OF(25404)	－	－
UF(25405)	－	－
N(25402)	－	－

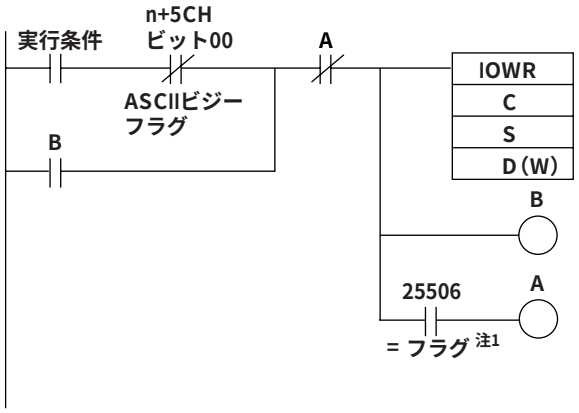
＊1：（　）内は、SYSMACα、C200HS、C200Hでのアドレス

■IOWR命令／IORD命令の使い方

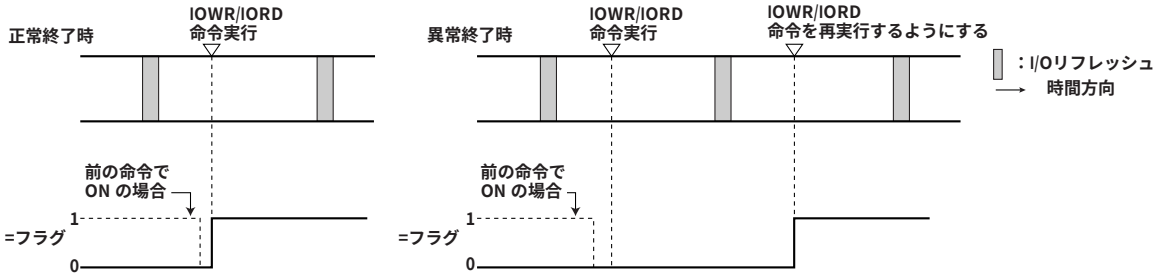
1　IOWR命令／IORD命令によるデータ転送が失敗したときのリトライ処理のため、入力条件に自己保持をかけ、＝フラグ(SYSMACαシリーズの場合25506)の1(ON)で自己保持を解除するようにしてください。

2　ASCIIユニットがCPUユニットとデータ転送中にIOWR命令／IORD命令を実行しないようにするため、ASCIIビジーフラグ(n＋5CH、ビット00)のb接点を入力条件としてください。

お願い　IOWR/IORD命令(コントロールコード#FD00)命令は、同一サイクル内でいずれか1つの命令しか動作できませんのでご注意ください。



注1：＝フラグ(SYSMACαシリーズの場合、25506)は、IOWR命令／IORD命令によるASCIIユニットに対する処理が正常終了時に1(ON)となります。ASCIIユニットに対する処理が正常に終了しなかったとき0(OFF)のままです。1(ON)となった後は、次のIOWR命令／IORD命令で0(OFF)となります。
ただし、＝フラグは、他の命令(比較命令など)を実行しても変化しますので、プログラム作成時は留意してください。



注2：ASCIIビジーフラグ(n＋5CH、ビット00)は、ASCIIユニットがCPUユニットとデータ転送処理中に1(ON)となります。

6－6　CSシリーズを使用する場合の留意事項

CPUユニットとASCIIユニットがデータ交換する方法には、5種類あります(6-2項参照)。ただし、CSシリーズで使用する場合、以下の5種類の内、方法1、2を使用する場合は、データ交換できるエリアが限定されます。

方法	CPUユニット側	ASCIIユニット側	CSシリーズでのエリア制限
1	MOV命令等	PC WRITE／PC READ命令	あり
2	（プログラム不要）	PC WRITE@／PC READ@命令	あり
3	MOV,LD命令等	PC GET／PC PUT命令	なし
4	IOWR(#00□□)／IORD(#00□□)命令	PC EGET／PC EPUT命令	なし
5	IOWR(#FD00)／IORD(#FD00)命令	PC QREAD@／PC QWRITE@ 命令	なし（ただし、PC QREAD@／PC QWRITE@命令はダミーエリアの指定が必要）

CSシリーズの場合

1. 割付リレーエリアを使用して転送エリア種別／アドレス指定をする場合は、制限があります。（方法1）
2. ASCIIユニット内のBASIC命令(PC WRITE@／PC READ@命令)によって転送エリア種別／アドレス指定をする場合、制限があります。（方法2）
3. インテリジェントI/O読み書き命令(IORD／IOWR命令)を使用したデータ転送時には、I/Oメモリのエリア種別／アドレス指定においてとくに制限はありません。CSシリーズのI/Oメモリすべてを指定することができます。（方法4、5）

●方法1の場合

PC READ/PC WRITE命令で読み書きするI/Oメモリのアドレス指定について

転送可能なエリアは、割付リレーエリア(n+3、n+4CH)で指定できる以下のエリアのみです。

エリア種別	CSシリーズの指定したいアドレス*1	割付リレーエリアでの指定方法	
		エリア種別 n+3CH ビット12～15（BCD1桁）	先頭アドレス n+4CH ビット00～15（BCD4桁）
データメモリ(DM) 汎用エリア	D00000～D06655	0	0000～6655
入出力リレー(CIO)	0000～0511CH 注：0281CHは指定不可	1	0000～0511 注：0281は指定不可
データリンクリレー(CIO)	1000～1063CH	2	0000～0063
保持リレー(HR)1	H000～H099CH	3	0000～0099
保持リレー(HR)2	H100CH～H127CH 注：H101CHは指定不可	4	0000～0027 注：0001は指定不可
拡張データメモリ(EM)バンクNo.0	E0_00000～E0_06143	5	0000～6143
タイマ	T0000～0511	6	0000～0511

*1：これ以外のCSシリーズのI/Oメモリアドレスを指定することはできません。もし、これ以外のI/Oメモリアドレスを指定したい場合、方法3～5のいずれかで代替してください。

●方法2の場合

PC READ@/PC WRITE@命令で読み書きするI/Oメモリのアドレス指定について

転送可能なエリアは、 PC READ@/PC WRITE@命令で指定できる以下のエリアのみです。

エリア種別	CSシリーズの指定したいアドレス＊1	PC READ@/PC WRITE@命令での指定方法	
		コード	先頭アドレス
データメモリ(DM) 汎用エリア	D00000～D06655	@D	0000～6655
入出力リレー(CIO)	0000～0511CH 注:0281CHは指定不可	@R	0000～0511 注:0281は指定不可
データリンクリレー(CIO)	1000～1063CH	@L	0000～0063
保持リレー(HR)1	H000～H099CH	@H	0000～0099
保持リレー(HR)2	H100CH～H127CH 注:H101CHは指定不可	@A	0000～0027 注:0001は指定不可
拡張データメモリ(EM)バンクNo.0	E0_00000～E0_06143	@E	0000～6143
タイマ	T0000～0511	@G	0000～0511

＊1：これ以外のCSシリーズのI/Oメモリアドレスを指定することはできません。もし、これ以外のI/Oメモリアドレスを指定したい場合、方法3～5のいずれかで代替してください。

●方法3の場合

PC GET ／PC PUT 命令による割付I/Oの読み書きについて

とくに制限はありません。

●方法4の場合

PC EGET／PC EPUT 命令によるASCII内共有メモリの読み書き、IOWR／IORD(#00□□)命令によるASCII内共有メモリの読み書きについて

とくに制限はありません。

注：CSシリーズの場合、IOWR／IORD(#00□□)命令での転送CH数の指定が、BCDでなく、16進であることに注意してください。

●方法5の場合

IOWR／IORD(#FD00)命令によるデータ転送、PC QREAD@／PC QWRITE@命令による割付エリアへの転送エリア格納について

エリア範囲に関してとくに制限はありません。

この場合、実際に転送されるエリアは、IOWR／IORD(#FD00)命令のオペランド指定にしたがいます。PC QREAD@／PC QWRITE@命令によって指定されるI/Oメモリのエリアは、実際に転送されるエリアの指定ではなく、割付リレーエリア(n+8、n+9CH)に格納されるエリアの指定です。この割付リレーエリアは、CPUユニットがモニタするためのエリアです。

したがって、CSシリーズの場合、PC QREAD@／PC QWRITE@命令によって指定できるエリア範囲とIOWR／IORD(#FD00)命令によって指定できるエリア範囲が異なるため、PC QREAD@命令／PC WRITE@によっては指定できないエリアを転送するときは、PC QREAD@／PC WRITE@命令にダミーのエリアを指定することが必要です(ただし、その場合、割付リレーエリアを使用したCPUユニットでのモニタは行えません)。

注：CSシリーズの場合、IOWR／IORD(#FD00)命令での転送CH数の指定が、BCDでなく、16進であることに注意してください。

■CSシリーズで時刻合わせフラグ(nCH ビット05)を使用する場合

ASCIIユニットは、時刻合わせフラグを立ち上げ(0→1)たとき、SYSMAC α シリーズCPUユニットの時計データエリア(AR18～21CH)を参照して、自ユニット内のRTCを更新します。

しかし、CSシリーズでASCIIユニットを使用する場合は、CSシリーズ CPUユニットの時計データエリアはA351～354CHになりますので、そのままASCIIユニットの時刻合わせフラグを立ち上げても正しく時刻がセットされません。

そこで、CSシリーズで時刻合わせフラグを使用する場合は、、以下のようなラダープログラムを追加してください(ただし、ラダープログラムを使用するため、CPUユニットが「プログラム」モードでは使用できません)。



CSシリーズでのH118～H121CHは、SYSMAC α シリーズでの時計データエリアAR18～21CHに該当しますので、上記XFER命令の転送処理により、時計データがHR118～121CHにあれば(時、分、秒などの配置がSYSMAC α と同じのため)、ASCIIユニットは時刻合わせフラグで時計データを更新することができます。

第7章
BASICプログラムの作成

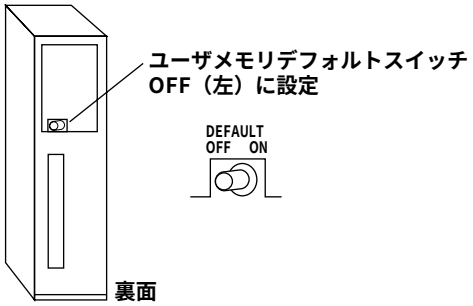
7−1　プログラミングの手順

■BASICプログラムの作成

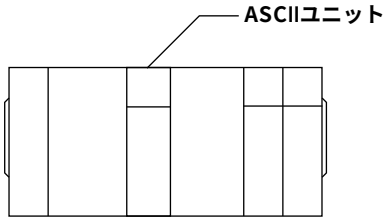
Windows95付属の通信／エミュレーションソフトHyper Terminalをターミナルとして BASICプログラムを作成する方法を、以下に例として示します。

●接続と初期設定

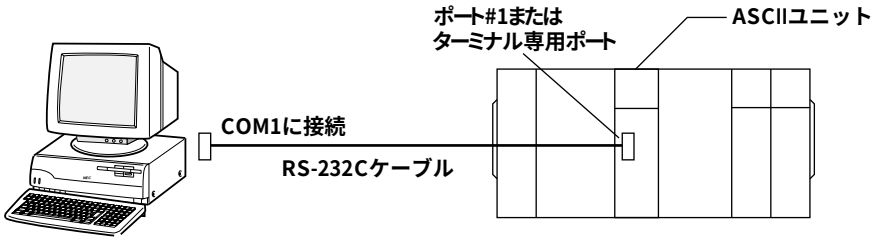
1　ユーザメモリデフォルトスイッチがOFF(左)に設定されていることを確認します。デフォルトスイッチをON(右：→側)にすると、電源ON時にメモリをデフォルト状態(初期化)にします。このスイッチは異常処理以外では必ず左に設定しておいてください。



2　ASCIIユニットをベースに取り付けます。



3 PC本体の電源をOFFにして、ASCIIユニットの通信ポート#1(形C200H-ASC11/21の場合)またはターミナル専用ポート(形C200H-ASC31の場合)と、パソコンのRS-232Cポート(この例では、COM1)を、RS-232Cケーブルで接続します。



4 ASCIIユニット前面のスイッチを設定します。

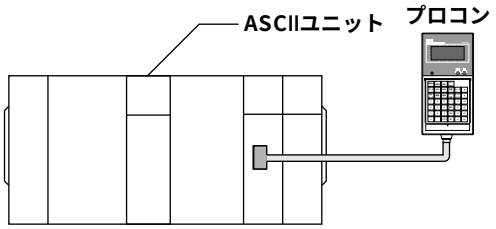
- 号機No.スイッチ
- START／STOPスイッチをSTARTに
- 終端抵抗スイッチ、2線／4線選択スイッチ(形C200H-ASC21の通信ポート#2使用時のみ)

5 PC本体の電源をONにします。

6 PC本体用周辺ツール(プロコンなど)を使用して、以下の操作を行います。

1) I/Oテーブル作成操作

2) CPUユニットのDMエリア(DMm～m+7)を設定(詳細は、3-2項を参照してください)。設定後、電源をOFF→ONにするか、またはPC本体用周辺ツール(プロコンなど)を使用して、ASCIIユニットのリスタートフラグをOFF→ON→OFFとします。



●ターミナルの接続

1　Windows95の[スタート]メニューから、[プログラム]―[アクセサリ]―[ハイパーターミナル]を選択します。

2　Hyper trmアイコンをダブルクリックします。「接続する前に、モデムをインストールしなければなりません。インストールを始めますか？」と表示されます。[いいえ]を選択します。

3　[接続の設定]ダイアログが表示されます。名前を入力します。



4　[電話番号]ダイアログが表示されます。接続方法のフィールドで、パソコンの通信ポート番号を選択します。"Com１へダイレクト"を選択し、[OK]をクリックします。



5　[COM１のプロパティ]ダイアログが、表示されます。

6　[ポートの設定]フィールドで、次の設定を選択します。

ビット／秒	9600
データビット	8
パリティ	なし
ストップビット	2
フロー制御	Xon/Xoff



上記のように、設定が完了したら、[OK]をクリックします。

注： ASCIIユニットの通信ポート#1／ターミナル専用ポートのデフォルトの通信条件は、いずれも9600ビット／s、8ビット、パリティなし、2ストップビット、フロー制御なしです。

7 [ファイル]メニューから[プロパティ]を選択します。[××(入力した名前)のプロパティ]ダイアログが表示されます。

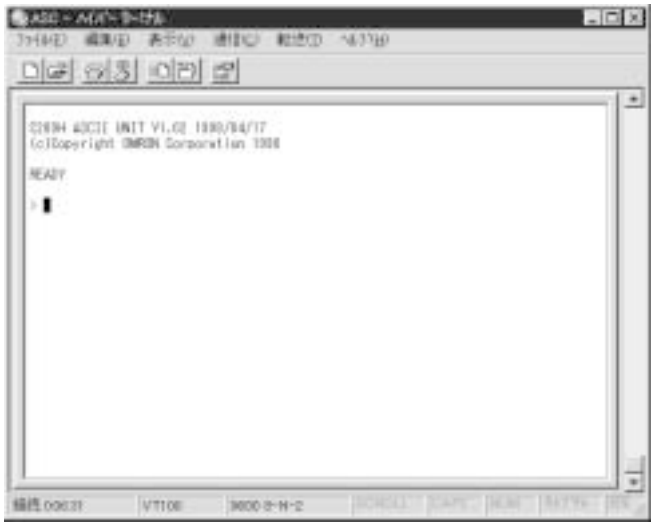
8 [設定]タグを選択し、[エミュレーション]フィールドで"VT100"を選択し、[ファンクションキー、方向キー、Ctrlキーの使い方]は"ターミナルキー"を選択した後、[OK]をクリックします。[バッファの行数]は、500のままとします。



お願い **半角かなを使用する場合はVT100Jを選択してください。**

9　パソコンから、[Ctrl]＋[X]を入力します。
ターミナルに下記のメッセージが表示されたら、接続完了です。

注：　下記メッセージは、PC本体の電源ON後、最初の[Ctrl]＋[X]キー時のみ表示されます。ターミナルのみ終了して再接続した場合は、プロンプト">"のみの表示となります。



接続できないときは、以下の点をチェックしてください。

・接続ケーブルのチェック（パソコンの接続ポートが正しいかなど）
・ASCIIユニットの通信条件（DM割付のm+2またはm+4がターミナルの通信条件が合っているか
・ASCIIユニットの前面ERR LEDが点灯している場合は、n+7CHのエラー（ビット00～11：エラーコード、ビット12～15：エラー種別）を確認して対応した後、PC本体の電源リセットをして、再度、[Ctrl]＋[X]を入力してください。
DM、リレーの割付については第3章を参照してください。
・ポートをOPENするプログラムを実行している状態で、START/STOP SWでSTOPさせた場合、プロンプト“＞”が表示されない場合があります。この場合、再度[Ctrl]＋[X]を入力してからターミナル操作を行ってください。

●プログラムの入力

ASCII**ユニットへ直接**BASIC**プログラムを入力する場合**

1　NEW↵を入力して、ASCIIユニット内のプログラムをクリアします。



注：本ASCIIユニットは、ユーザメモリエリア（RAM）内に４つのプログラムエリア（No.1～４）をもちます。デフォルトの状態で入力されるプログラムは、プログラムNo.１のプログラムエリアです。プログラムNo.1以外のプログラムを作成する場合は、PGENコマンドで、アクティブの（対象としている）プログラムNo.を切り替えてください。
例：＞PGEN 2↵　プログラムNo.2がアクティブになります。

2　AUTO↵を入力して、行番号を自動表示します。



3 BASICプログラムを入力します。AUTOを終了させる場合は、[Ctrl]＋[X]を入力します。
例：

```
> NEW
> AUTO
10 A$=CHR$(67)
20 PRINT A$
30 _
```

参 考

- **作成したBASICプログラムを表示するときは、LISTコマンドを実行します。**
- **作成したプログラムにプログラム名を付けるときは、PNAMEコマンドを実行します。詳細は、7-6項「BASIC命令の詳細」のPNAMEコマンドの項を参照してください。**
  例：>PNAME "SAMPLE" ↵ プログラム名をSAMPLEとする。
- **ASCIIユニット内のユーザメモリエリア(RAM)の情報を画面に表示したいときは、PINFコマンドを実行します。詳細は、7-6項「BASIC命令の詳細」のPINFコマンドの項を参照してください。**
  例：>PINF ALL↵

```
> PINF ALL
P1      :                [R/W]      76 bytes
P2      :                [R/W]       0 bytes
P3      :                [R/W]       0 bytes
P4      :                [R/W]       0 bytes
LIB     :                            0 bytes
I-CODE  :                           41 bytes
```

テキストエディタでBASICプログラムを作成後、ASCIIユニットへ転送する場合

1 メモ帳等のテキストエディタを使用して、プログラムを作成し、テキスト形式で保存しておきます。

2 START/STOPスイッチをSTOPにし、NEWを入力して、ASCIIユニット内のプログラムをクリアします。

3 LOADコマンドを入力して、ASCIIユニットをファイル転送待ち状態にします。(BASIC LED(緑)が速く点滅します。)

4 ハイパーターミナルの[転送]―[テキストファイルの送信]を選択して、保存しておいたファイルを読み込んで、ASCIIユニットへ転送します。アクティブなプログラムNo.に、上書きで格納されます。



5　転送が完了したら、[Ctrl]+[C]を入力してください。再度プロンプト(">")が表示されます。
転送の完了は、T/R LEDが消灯することで確認することができます。

お願い　**・テキストファイルの送信でファイルをASCIIユニットへ転送する場合は、フロー制御または送信ディレーの設定を使用するようにしてください。正しくファイルが受信されないことがあります。**

例)フロー制御の設定

ターミナル側：[ポートの設定]フィールドでフロー制御Xon/Xoffを選択します。

ASCIIユニット側：LOADコマンド入力時に、X-ON/X-OFF制御をONに指定します。

例)LOAD #1, "TERM : 9600, 8, N, 2, CS_OFF, RS_OFF, DS_OFF, XN_ON"

・本ASCIIユニットでは、プログラムを実行するためにソースプログラムをコンパイルし中間コードを生成します。そのため、中間コードの生成のためにメモリを消費します。比較的大きなプログラムを作成する場合は、ときどきRUNを実行し、PINF ALLでメモリの消費状態を確認してください。

●ターミナルの終了

1　接続情報(セッション)を保存するときは、[ファイル]－[名前を付けて保存]を選択し、保存します。

2　ハイパーターミナルを終了するときは、[ファイル]－[ハイパーターミナルの終了]を選択します。「現在接続されています。切断してもよろしいですか？」と表示されます。[はい]をクリックします。

●次回接続時の操作

1　作成したアイコン(セッションファイル)をダブルクリックします。(または、[ファイル]－[開く]を選択して、セッションファイルを選択します。)

2　「接続する前に、モデムをインストールしなければなりません。インストールを始めますか？」と表示されます。[いいえ]を選択します。

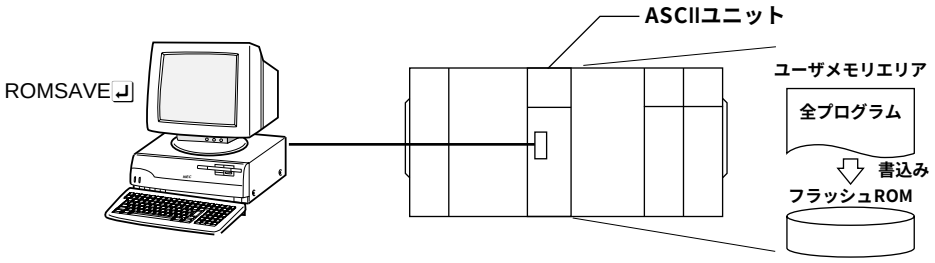
3　[Ctrl]+[X]を入力します。ターミナルに、前記と同様なメッセージが表示されたら、接続完了です。

■フラッシュROMへの書込み／読出し

●BASICプログラムのフラッシュROMへの書込み

ターミナルからROMSAVEコマンドを入力することで、ユーザメモリエリア(RAM)内のBASICプログラムをASCIIユニットのフラッシュROMへ書き込むことができます。

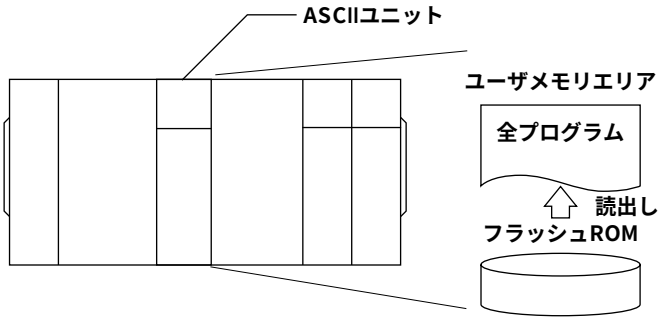
＞ROMSAVE↵



●BASICプログラムのフラッシュROMからの読出し

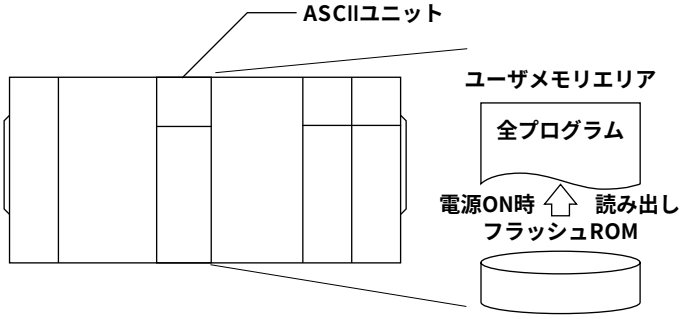
ターミナルからROMLOADコマンドを入力することで、ASCIIユニットのフラッシュROM内のBASICプログラムを、ASCIIユニットのユーザメモリエリア(RAM)に読み出す方法は、以下のとおりです。

＞ROMLOAD↵



●電源ON時にフラッシュROM内のプログラムを自動読み出し

DM割付エリアのmCH　ビット08～15を5AHexに設定することで、電源ON時またはユニットリスタート時に、フラッシュROM内のプログラムをユーザメモリエリア(RAM)に自動的に読み出すことができます。



⚠注意

ROMSAVEコマンドによって、フラッシュROMにプログラムを書込み中（ターミナルに、終了表示がでるまでの間）に、PCの電源を絶対に切らないでください。書込み中にPC電源を切ると、フラッシュROMが電気的に壊れ、セーブできなくなる可能性があります。



お願い
- **ROMLOADコマンドまたは電源ON時自動転送によって、フラッシュROMからプログラムを読み出し中に、 PCの電源断またはユニットのリスタートを実行しないでください。（読み出し中か否かは、 ROMLOADコマンドの場合は、ターミナルに終了表示（プロンプト）が出るまでを指します。電源ON時自動転送の場合は、割付DMエリアの設定で、スタートモードを「自動スタート」にした上で、電源ONまたはリスタートを実行し、BASIC　LEDが点灯するか、またはターミナルがコマンド入力待ちになるまでを指します。）**
- **また、PC側のラダープログラム内でASCIIユニットとのデータ交換用にIOWR/IORD命令を使用している場合、ROMSAVEコマンドによるフラッシュROMへのBASICプログラム書き込み時は、PC側のラダープログラムを必ず停止させてから書き込みを実行してください。**

■ASCIIユニット←→ターミナル間のBASICプログラム読出し／書込み

●BASICプログラムのターミナルへの読出し

ASCIIユニットのユーザメモリエリア（RAM）内のBASICプログラムをターミナルに読み出す方法は、以下のとおりです。

1．ターミナルがVT100モードで使用の場合（Windows ハイパーターミナルの場合）（注1）

1） ターミナルがつながっている状態でSAVE #1 またはSAVE #3（ASC31のみ）……↵と入力します。
例）SAVE #1, "TERM：9600, 8, n, 2, CS_OFF, RS_OFF, DS_OFF, XN_ON"↵

2） ターミナルをテキストデータ受信状態にします。

3） ターミナルが受信待ち状態になったら、［Ctrl］＋［C］を入力します。
ASCIIユニットは［Ctrl］＋［C］の入力により、データの転送を開始します。
このときT/R LEDが点滅します。

4） ASCIIユニットが全データの転送を終了すると、T/R LEDが消灯します。

5） ターミナルでのテキストデータ受信状態を終了にします。

6） ［Ctrl］＋［C］を入力するとプロンプトが再表示されます。

注1：BASICプログラムにPWORD命令でパスワードが設定されている場合、次の手順となります。
1）SAVE #1 またはSAVE #3（ASC31のみ）と入力し↵を入力します。
このとき、ASCIIユニットは"PASSWORD"と表示します。そうしたら、パスワードを入力します。
2）ターミナルをテキストデータ受信状態にします。
3）ターミナルが受信待ち状態になったら［Ctrl］＋［C］を入力します。
4）、5）の手順は上記と同じです。

2．ターミナルがFIT10モードで使用の場合

1） START/STOPスイッチをSTOPにしておきます。

2） ターミナル側からSAVE #1 またはSAVE #3（ASC31のみ）……↵と入力します。ASCIIユニットは↵の入力により、送信待機状態（BASIC LEDが速く点滅）となります。（注2）
例）SAVE #1, "TERM：9600, 8, n, 2, CS_OFF, RS_OFF, DS_OFF, XN_ON"↵

3） ターミナル側をテキストデータ受信状態にします。

4） ターミナルが受信待ち状態になったらSTART/STOPスイッチをSTARTにします。

ASCIIユニットがデータの転送を開始すると、T/R LEDが点滅します。

5） ASCIIユニットが全データの転送を終了したら、START/STOPスイッチをSTOPにすると、プロンプトが再表示されます。

6） ターミナルでのテキストデータ受信状態を終了にします。

注2：BASICプログラムにパスワードが設定されている場合、"PASSWORD"の表示が出たら、パスワードを入力し、最後に↵を入力します。ASCIIユニットが送信待機状態となったら後の手順は上記と同じです。

●ターミナルからASCIIユニットへの書込み

ターミナルからASCIIユニットのユーザメモリエリア（RAM）へBASICプログラムを書き込む場合、LOADコマンドを入力してASCIIユニットをファイル転送待ち状態にしてから、テキストファイルを転送します。

参 考 BASICプログラムの転送と使用コマンド

目的		使用コマンド	例
ターミナル←→ASCIIユニット間の BASICプログラムの転送	ターミナル→ASCIIユニットのユーザメモリエリア（RAM）へ	LOAD	＞LOAD #1
	ASCIIユニットのユーザメモリエリア（RAM）→ターミナルへ	SAVE	＞SAVE #1
フラッシュROM←→ASCIIユニット間のBASICプログラムの転送	フラッシュROM→ ASCIIユニットのユーザメモリエリア（RAM）へ	ROMLOAD	＞ROMLOAD
	ASCIIユニットのユーザメモリエリア（RAM）→ フラッシュROMへ	ROMSAVE	＞ROMSAVE

7－2　文字変数エリアの割付け方式

本ASCIIユニットでは、BASICプログラムの実行の際に、文字変数の変数エリアを、ユーザメモリに割り付ける方式を選択できます。次の2つの方式が可能です。

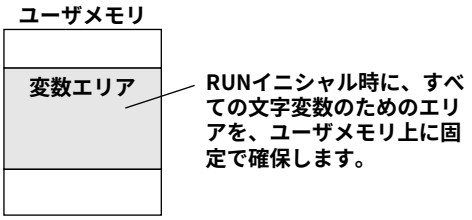
注：数値変数の場合は、型によってサイズが固定のため、下記で言うスタティック(固定)方式で、ユーザメモリに割り付けられます。

1．スタティック(固定)方式

BASICプログラムRUNコンパイル時に、BASICプログラム中にあるすべての文字変数のためのエリアを、ユーザメモリ上に固定で確保します。

指定方法：BASICプログラムの先頭に、OPTION LENGTH命令によって、BASICプログラム中で使用する文字変数の内容の最大長を指定します。最大長として、1～255文字を指定できます。

内容：　固定的に文字変数のためのエリアを確保してしまうため、ダイナミック方式のような空きスペースを生成することはありませんが、実際に使用していなくとも、ユーザメモリを最大値で消費します。



この方式の場合、次の制約事項がありますので、ご注意ください。

・OPTION LENGTH命令によって指定した文字変数内容の最大長を超えることはできません。もし。それを超える文字数を格納した場合、オーバーした分は、変数には格納されません。

・スタティック方式では、一度指定した変数名の型を、プログラム実行中に、同じ変数名で別の型に変更することはできません。

・スタティック方式では、プログラム中で、同じ変数名に対して1度だけ、DEF(変数型宣言)命令を実行できます。

・スタティック方式では、RUNコンパイル時に、添字の数をもとに、配列変数のメモリサイズを固定で割り付けるため、DIM命令の添字には、固定数値のみ指定可能です。数値式や数値変数を指定することはできません。(一方、ダイナミック方式では、プログラム中で、数値式や数値変数によって添字の数を指定することで、必要に応じて添字の数を変化されることが可能です。)

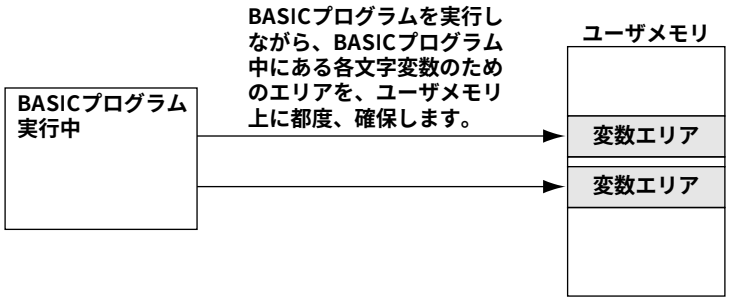
2．ダイナミック(自由)方式

BASICプログラムを実行しながら、BASICプログラム中にある各文字変数のためのエリアを、ユーザメモリ上に都度、確保します。

指定方法：OPTION LENGTH命令を使用しない

内容：　ユーザメモリが都度必要分だけ使用されます。BASICプログラム実行中に変数のサイズが変わったとき(サイズが大きくなるとき)、確保していたエリアは一度解放され、新たに再度確保されます。(そのとき、解放されたエリアは、サイズをもつ空きスペースとなります。)

ただし、再度エリアを確保しようとしたとき、ユーザメモリの容量不足のため、割り付けることができなかった場合は、エラーとなり、停止してしまうことがありますので、ご注意ください。その場合、プログラムを変更し、文字変数の使い方を変えることが必要となります。



お願い　**上記２つの方式の違いにより、ASCIIユニットの動作が以下の点で異なりますので、ご注意ください。**

- **スタティック方式では、ダイナミック方式に比べ、RUN時のコンパイル処理時間が増えます。**
- **逆に、ダイナミック方式では、文字変数の代入時、ユーザメモリの解放および割り付け処理が発生することがあるため、命令としての実行時間がスタティック方式に比べ、長くなることがあります。**
- **OPTION LENGTH命令とPWORD命令を組み合わせて使用する場合は、PWORD命令を先頭にしてください。**

7－3　BASICプログラムの起動／停止

■プログラムの起動方法

1．プログラムNo.の選択

・CPUユニットのDM割付データを電源ONまたはユニットリスタート時にASCIIユニットに転送する場合、システム設定エリアのmCH ビット00～07によって、PC本体電源ON時のプログラムNo.を指定します。

DMの割付については第3章を参照してください。

00： No.1　　01： No.1

02： No.2　　03： No.3

04： No.4

・ターミナルから、プログラムNo.を切替える場合、PGENコマンドで切り替えます。

例：＞PGEN 2↵　　プログラムNo.2に切り替え

2．プログラムの起動

ターミナルから起動する場合

ユニット前面のSTART／STOPスイッチがSTART位置の条件で、ターミナルからRUNコマンドを入力します。

ユニット前面のSTART／STOPスイッチによる場合

START／STOPスイッチをSTOP → STARTへ切り替えます

電源ONまたはリスタート時に自動起動する場合

ユニット前面のSTART／STOPスイッチがSTART位置の条件で、 DM割付のシステム設定エリアのm+1CH ビット08～11によって、自動スタート（5AHex）に設定します。

DMの割付については第3章を参照してください。

■プログラムを停止（ブレイク）させる方法

1．ターミナルから停止させる場合

ターミナルから、[Ctrl]+[X]キーを入力します。

ただし、ターミナルを接続したポートが、"COMU"や出力デバイスでOPENされている場合は、[Ctrl]+[X]の入力は効きません。このような場合は、ユニット前面のSTART/STOPスイッチを使用してください。

2．ユニット前面のSTART／STOPスイッチによる場合

START／STOPスイッチをスタート→ストップへ切り替えます

■プログラムを指定した行番号で停止(ブレイク)させる方法

1．ターミナルから停止させる場合

a)RUN　TO行番号を入力して、プログラムを起動させます。

b)BRKPTコマンドを入力して、ブレイクポイントを設定し、RUNさせます。

c)STEPコマンドを入力して、指定行数分だけRUNさせます。

2．プログラム内の命令による場合

指定行番号にSTOP命令を設定しておき、RUNさせます。

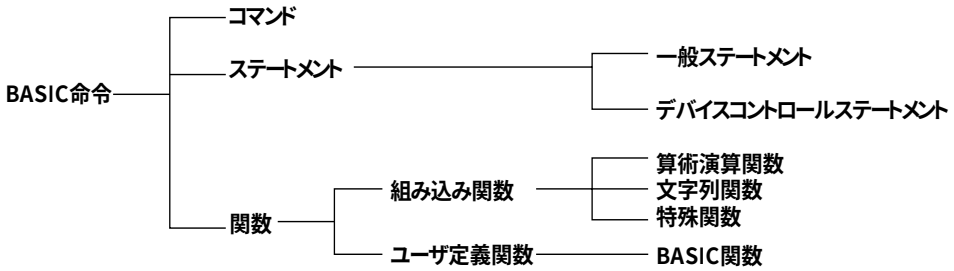
■プログラムを途中の行で停止(ブレイク)させた後、継続起動させる方法

1) ターミナルからのCONTコマンドを入力します。

2) 指定行数分だけの場合は、STEPコマンドを入力します。だけの場合は、STEPコマンドを入力します。

7－4　BASIC命令の構成

本ユニットで使用できるBASIC命令には、以下のように、コマンド、ステートメント、関数があります。



種　類	内　容
コマンド	ASCIIユニットがコマンドモードのときに、ターミナルから直接入力し、実行する命令。 注：プログラム内に作成し、"RUN"で実行させることはできません。実行させた場合、正常に動作しないことがあります
ステートメント	プログラムとしてあらかじめ作成しておき、"RUN"で実行させる命令。 注：行番号なしで、ターミナルから直接入力し、実行することも可能です。
関数	与えられた任意の値（引数）に対して、1対1、または1対多で1つの値（戻り値）を返す命令 注：関数は単独では使用できず、ステートメントと組み合わせて使用します。

■命令実行モード

命令の実行には、次の2つのモードがあります。

●コマンドモード

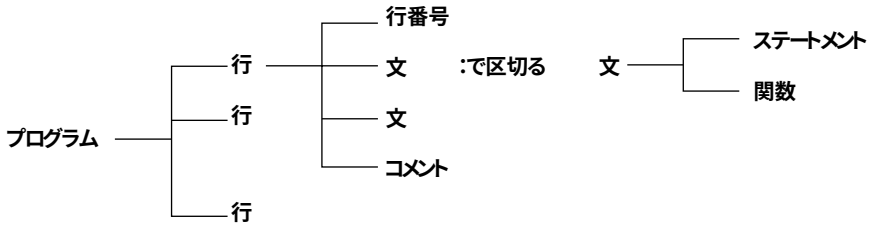
ターミナル画面に"＞"が表示されているとき、コマンドを入力して、［Enter］キーを押すと、BASICがすぐにそのコマンドを実行します。コマンド入力待ちの状態をコマンドモードと呼びます。

●プログラムモード

先頭に行番号をつけて、BASIC命令を入力し、［Enter］キーを押すことで、行が作成されます。この行が複数集まったものをプログラムと呼びます。このプログラムを単位として、命令を実行している状態をプログラムモードと呼びます。

■プログラム／行／文

BASICプログラムは、行の集まりによって構成されます。行は1つの文または、複数の文によって構成されます。文は、ステートメント、関数から構成されます。



●行番号

・行の最初には行番号が必要です。(行番号の無い場合は直接実行します。)

・[Enter]までを1行として、行番号順にユーザメモリに格納します。行番号順に実行されます。

また、行番号はGOTO、GOSUBなどの分岐先や、LIST、DELなどの対象行の参照に使用されます。

・行番号は1～65535までの整数が許されます。行番号が順に大きくなるように各行を記述します。

・一行は255文字以内です(行番号、スペースを含む)。

行番号	[ラベル:]	文	:	文	:	['コメント]	Enter

注：プログラムの最終行番号(最も新しく入力した行番号もしくは最も新しく表示した行番号)の参照にはピリオド (.) が使えます。

ピリオドの使用例：　LIST.

EDIT.

DEL　100－.

●ラベル

ラベルは、GOTOまたはGOSUB命令の中で、行番号の代りに使用されます。参照先では、行の最初の文として、＊のあとに英文字からはじめます。＊を含めて最大16文字です。予約語で始まる文字列は使用できません。1行中でラベルのあとに、文を続けるときは、コロン(：)で区切ります。同一ラベル名が存在する場合、最も小さい行番号にあるラベルが参照されます。

例：

```
10 ON COM 1 GOSUB ＊ABC
                   └─ラベル

100 ＊ABC:
      └─ラベル
```

参　考　**分岐先に行番号を指定すると、プログラム修正時に行番号も修正しなければなりません。ラベルを使用すれば、行番号を修正する必要がありませんので、プログラム開発効率を上げることができます。**

ラベルに予約語で始まる文字列を使用すると「SYNTAX ERROR」となります。

●文

・文はステートメント、関数から成ります。

・行を複数の文から構成する複文の場合は、文と文をコロン(：)で区切ってください。

複文の例　10　FOR　L＝1　TO　100：J＝L＊I：PRINT　J：NEXT　L

行番号　文1　文2　文3　文4

●コメント

・コメントは、シングルクォート（ ’ ）の後に入力してください。

注：ただし、DATA命令の行にコメントを入力することはできません。

■BASICで使う文字

・英字はA～Zまでの26文字の大文字

・数字は0～9までの10文字

・その他の特殊記号として次のものが使えます。

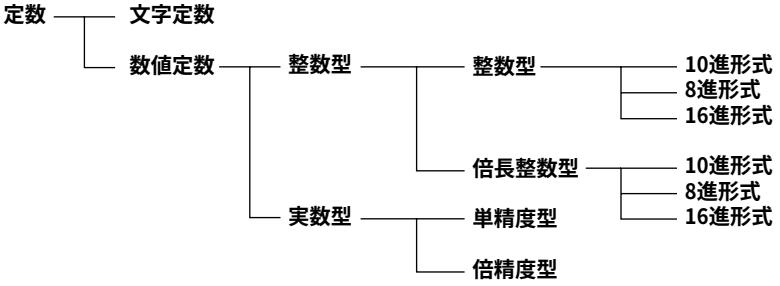
SP(スペース)！　”　＃　＄　％　＆　’（　）　＊　＋　，　－　．　／　：　；

＜　＝　＞　？　〔　¥　〕＾＿

・コメントおよび文字定数には、上記の他に英字の小文字、カタカナ、ひらがな、漢字も使用できます。

■定数

定数には以下のものがあります。



●文字定数

文字セットの並びをダブルクォーテーション(”　”)で囲って指定します。文字数の長さは255文字以下です。特に0の場合は空文字列と言います。

例）”ASCII－BASIC ”

●数値定数

・整数型

整数型には2種類あります。

・整数型(16ビット)

・倍長整数型(32ビット)

10進形式
整数型の場合：－32768～＋32767、倍長整数型の場合：－2147483648～＋2147483647の範囲です。小数点を付けることはできません。10進数整数では数の後に整数型の場合：%、倍長整数型の場合：＆を付けて明示することもできます。

例）　1234
　　－1234
　　1234%（整数型）
　　12345678&（倍長整数型）

8進形式
＆Oまたは＆で始まる8進数（0～7）の並びです。
整数型の場合：＆0～＆177777、倍長整数型の場合：＆0～＆37777777777の範囲です。

例）　＆O127
　　＆7777

16進形式
＆Hで始まる16進数（0～9、A～F）の並びです。
整数型の場合：&H0～&HFFFF、倍長整数型の場合：＆H0～＆HFFFFFFFFの範囲です。

例）　＆H5E
　　＆HBF4

・実数型

指数形式で表わされる正または負の数です。仮数部の後にE（単精度）またはD（倍精度）を入力し、その後に指数部を入力します。
例えば、12300000という数値の場合、

＝1.23×10000000
＝1.23×10^7
＝1.23E＋7（単精度の場合）

または＝1.23D＋7（倍精度の場合）という表現形式で記述します。
このとき、1.23の部分が仮数部、E＋7またはD＋7の部分が指数部です。

単精度型

精度はIEEE－754　32ビット　浮動小数点フォーマットに準じています。
±1.2×10^{-38}～±3.4×10^{38}の範囲内の値で、7桁までの精度で格納され、7桁目を四捨五入して6桁までの桁数で出力されます。
次のいずれかで表現します。

(a)有効桁数が6桁以下の定数　1234.5
(b)Eを用いた指数形式の定数　－1.2E＋3
(c)最後に！が付いた定数2.34！

倍精度型

精度はIEEE－754　64ビット　浮動小数点フォーマットに準じています。
±2.23×10^{-308}～±1.79×10^{308}の範囲内の値で、16桁までの精度で格納され、16桁目を四捨五入して15桁までの桁数で出力されます。
次のいずれかで表現します。

(a)有効桁数が7桁以上の定数　1.23456789
(b)Dを用いた指数形式の定数　－1.2D－3
(c)最後に＃が付いた定数2.34＃

お願い　**実数についての留意点**

浮動小数点フォーマットで表される実数は、扱える値の範囲が広いという長所がありますが、２進数による内部表現の性質上すべての値が正確に表現できるわけでなく、その近似値で表されます。したがって、メモリに格納した値および演算結果には、ほとんどの場合誤差が含まれています。通常この誤差は非常に小さいため、問題になることはありませんが、アプリケーションによっては大きな誤差となるため、注意が必要です。

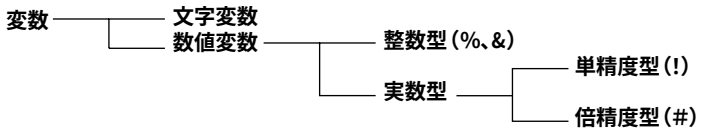
例えば0.1を10倍して小数点以下を切り取るという操作を行う場合、1という結果が期待されますが、実際は0.1をメモリに格納する際に２進数では表しきれないため、0.0999･･･といった近似値として格納されます。その結果、10倍しても0.999･･･という結果になるため、小数点以下を切り取った結果は0となります。

この問題の回避方法はアプリケーションによって異なりますが、例えば、整数型を使って目的の動作を実現できないか、あるいは演算結果を丸めて期待する整数を得られないか、などの検討をして設計してください。

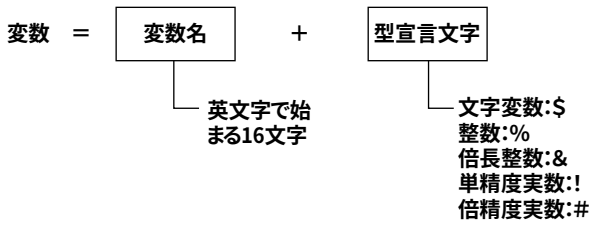
■変数

変数には、数値変数と文字変数があります。

注：　変数に値を与える前に変数を用いると、数値変数の場合、その値は0になっています。文字変数の場合、その値は空文字列(何の文字も持たない状態)になっています。



変数は、変数名と型宣言文字から構成されます。



●変数名

変数名は、英字で始まる255文字以内の英数字並びで表現されます。最初の16文字を有効とします。先頭文字は、英文字でなければなりません。

変数名の先頭がFN、またはBASICコマンドの予約語であってはいけません。予約語とは、すべてのステートメント、関数名、演算子名を指します。FNで始める場合は、ユーザ定義の関数の呼び出しとみなされます。

※先頭に予約語が含まれていた変数の場合、シンタックスエラーが発生します。

例えば、「TOTAL」「ABSOL」の場合でも、先頭の「TO」「ABS」が予約語であるためシンタックスエラーが発生し、変数として使用することはできません。

●型宣言文字

変数名の後につけて、その変数の型を定義します。 変数名が同じでも、型が違えば別の変数として扱われます。数値変数名の型宣言文字を省略した場合は、単精度実数型とみなされます。

%	整数型	1変数につき2バイトのエリアを使用する。	例：A%
&	倍長整数型	1変数につき4バイトのエリアを使用する。	例：A&
!	単精度実数型	1変数につき4バイトのエリアを使用する。	例：AまたはA!
#	倍精度実数型	1変数につき8バイトのエリアを使用する。	例：A#
$	文字型	最大255文字のデータを格納できます。	例：A$

注： 型宣言をする他の方法に、DEFINT、DEFLNG、DEFSNG、DEFDBL、DEFSTR命令がありますが、この型宣言文字による宣言の方が優先されます。

■配列変数

配列は、同じ型(数値型または文字型が可能)のデータを持つ複数個のデータの集まりです。添字付きの同一変数名で、各配列要素を指定できます。(添字は、配列中の要素の位置を示します。)

配列の宣言は、DIM命令によって行います。配列変数の宣言をすると、数値型は0に、文字型は空文字に初期化されます。

注： 配列の次元数、および添字の最大値はメモリ容量に依存します。配列の宣言をせずに、配列変数を使用した場合、添字の最大値は10となります。

例：

DIM A$(3)　配列変数がA$、文字列を値とした、次元数が1次元の配列。次の4個の変数を使用できます。

A$(0)	A$(1)	A$(2)	A$(3)

DIM B(2, 3)配列変数がB、数値を値とした、次元数が2次元の配列。次の12個の変数を使用できます。

B(0, 0)	B(1, 0)	B(2, 0)
B(0, 1)	B(1, 1)	B(2, 1)
B(0, 2)	B(1, 2)	B(2, 2)
B(0, 3)	B(1, 3)	B(2, 3)

■型変換

数値データを使って演算する場合には、演算実行時に必要に応じて型変換を自動的に行います。
変換規則は以下に従います。

(1) 代入文において、右辺の数値データの型と、左辺の数値変数の型が異なる場合には、左辺の型に変換されて代入されます。
ただし、文字型を数値型へまたは、数値型を文字型へ代入することはできません。
例)A%＝12.3→A%には12が代入されます。

(2) 代入文において倍精度型データを単精度変数に代入する場合は、四捨五入が行われます。
例)A＝12.3456789＃→Aには12.34568(7桁に丸められる)が代入され
表示すると12.3457(6桁に四捨五入)となります。

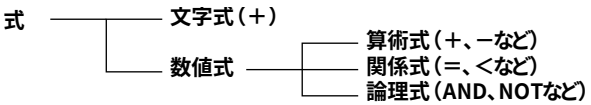
(3) 単精度型と倍精度型のデータ間で演算する場合は単精度型の数値を倍精度型に変換してから演算しますので、結果は倍精度型となります。
例)10＃／3→結果は倍精度型

(4) 論理演算の場合には、数値はすべて整数型に変換してから演算します。
－2147483648～2147483647の範囲内の整数に変換できない場合はエラーとなります。
例)A＝NOT　12.34→Aには－13が代入されます。

(5) 実数を整数に変換する場合は、小数点以下を四捨五入します。
例)　(1)と同じ

■式

定数、変数、関数を演算子で結んだものや、単に数値、変数、文字だけのものも式と呼びます。
式には次の4種類があります。
・算術式　・関係式　・論理式　・文字式
この内、最初の3種はその評価結果が数値であることから、まとめて数値式と呼び文字式と区別します。



●文字式

文字定数、文字変数を演算子＋で連結したもので、2つの文字データの加算ではなく、2つの文字列を連結して1つの文字列を得る場合に使います。

```
（例）A$＝“ ASCII ”
      B$＝“ BASIC ”
      C$＝A$+“ － ”＋B$
      PRINT C$

ASCII－BASICと出力される
```

●算術式

定数、変数、関数などの数値データを下表の演算子で結んだものです。

算術演算子	記述例	説　明
＋	A＋B	加算
－	A－B，－A	減算または負号
＊	A＊B	乗算
／	A／B	実数除算
¥	A¥B	整数除算
MOD	A　MOD　B	整数除算の乗余
＾	A＾B	べき乗

¥または、MOD演算子でAまたはBが実数の場合は、小数部分を四捨五入し、整数にしてから演算します。

●関係式

関係式は2つの値を比較し、その結果が真なら－1(&HFFFF)、偽なら0となります。

関係演算子	記述例	説　明
＝	A＝B	等しい
＜＞，＞＜	A＜＞B	等しくない
＜	A＜B	小さい
＞	A＞B	大きい
＜＝，＝＜	A＜＝B	小さいか等しい
＞＝，＝＞	A＞＝B	大きいか等しい

●論理式

定数、変数、関数などの数値データを下表の論理演算子で結んだもので、ビット操作や論理演算をしたり、複数の関係式の結果を調べたりするときに使います。

8進、16進形式でのビットごとの論理演算を実行します。

算術演算子	論　理	
NOT(否定)	A	NOT A
	1 0	0 1
AND(論理積)	A B	A AND B
	1 1 1 0 0 1 0 0	1 0 0 0
OR(論理和)	A B	A OR B
	1 1 1 0 0 1 0 0	1 1 1 0
XOR(排他的論理和)	A B	A XOR B
	1 1 1 0 0 1 0 0	0 1 1 0
EQV(同値)	A B	A EQV B
	1 1 1 0 0 1 0 0	1 0 0 1
IMP(包含)	A B	A IMP B
	1 1 1 0 0 1 0 0	1 0 1 1

7−5　BASIC命令一覧

この一覧表は、全命令をアルファベット順に並べて、その概要を示したものです。各命令の詳細については、7−6項「BASIC命令の詳細」を参照してください。

なお、構文欄の[　]は省略可能な項目であることを示します。{　}は繰り返しが可能な項目であることを示します。（構文の詳細については、7−6項を参照してください。）

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
@IF	@IF 数値式　THEN プログラム ELSE プログラム ENDIF	数値式の結果により THEN以降とELSE以降のいずれかを選択する		一般		7-35
ABS	ABS(数値式)	引数で与えた数値の絶対値を得る			算術演算	7-35
ACOS	ACOS(数値式)	引数で与えられた数値の逆余弦(アークコサイン)を得る			算術演算	7-36
ALARM ON/ OFF/ STOP	ALARM　ON または ALARM　OFF または ALARM　STOP	ワンショットタイマ(ON　ALARM)割込みの許可、禁止、停止を制御する		一般		7-36
ASC	ASC(文字式)	引数で与えた文字列の先頭文字のASCIIコードを得る			文字列	7-37
ASIN	ASIN(数値式)	引数で与えられた数値の逆正弦(アークサイン)を得る			算術演算	7-37
ATN	ATN(数値式)	引数で与えられた数値の逆正接(アークタンジェント)を得る			算術演算	7-37
AUTO	AUTO [行番号][,増分値]	行の先頭に番号を自動的に発生させる(増分値の指定がないときは,増分値=10)	コマンド			7-38
BITOFF/ BITON	BITOFF　数値変数,ビット位置 BITON　数値変数,ビット位置	整数型の数値変数のビット位置(0〜31)をOFF/ONさせる		一般		7-38
BRKPT	BRKPT [SET 行番号] またはBRKPT [DEL 行番号] またはBRKPT [DEL ALL] または BRKPT	ブレイクポイントの設定(SET),削除(DEL),リスト表示(なし時)を行う	コマンド			7-39
CDBL	CDBL(数値式)	数値を倍精度実数値に変換する			算術演算	7-39
CHR$	CHR$(数値式)	引数で与えたASCIIコード(整数)に対応する１文字を得る			文字列	7-40
CINT	CINT(数値式)	数値を四捨五入して整数値にする			算術演算	7-40

7－5　BASIC命令一覧

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
CLEAR	CLEAR	数値変数,文字変数の初期化を行う、また、すべてのポートを閉じる		一般		7-41
CLNG	CLNG (数値式)	数値を倍長整数に変換する			算術演算	7-41
CLOSE	CLOSE［＃ポート番号]	ポートを閉じる		デバイスコントロール		7-42
CLS	CLS［＃ポート番号]	画面をクリアする		デバイスコントロール		7-42
COM ON / OFF/STOP	COM [ポート番号] ON または COM [ポート番号] OFF または COM [ポート番号] STOP	通信割込み(ON COM)の許可、禁止、停止を制御する		一般		7-43
CONT	CONT	実行停止しているプログラムの実行を再開する	コマンド			7-43
COS	COS(数値式)	引数で与えられた数値の余弦(COS)を得る			算術演算	7-44
CSNG	CSNG(数値式)	数値を単精度実数に変換する			算術演算	7-44
CTS	CTS(ポート番号)	CS信号の状態を読む			特殊	7-44
DATA	DATA[定数{,定数}]	READ命令に与える数値および文字定数を定義する		一般		7-45
DATE$	DATE$["YY/MM/DD"]	日付をセットする、または日付を与える			特殊	7-45
DAY	DAY[=数値式]	曜日をセットする、または与える			特殊	7-46
DEF FN	DEF　FN　名前[(引数[,引数…])]=式	ユーザ作成のBASIC関数を定義する		一般		7-46
DEFINT/ DEFLNG/ DEFSNG/ DEFDBL/ DEFSTR	DEFINT　文字[－文字]{,文字[－文字]} または DEFLNG　文字[－文字]{,文字[－文字]} または DEFSNG　文字[－文字]{,文字[－文字]} または DEFDBL　文字[－文字]{,文字[－文字]} または DEFSTR　文字[－文字]{,文字[－文字]}	変数の型を整数、倍長整数、単精度、倍精度、文字に宣言する		一般		7-47
DEL	DEL［行番号１][－[行番号２]]	プログラム行の一部を削除する	コマンド			7-47
DIM	DIM 変数名(添字最大値{,添字最大値}){,変数名(添字最大値{,添字最大値})}	配列変数の次元と添字の最大値を指定する		一般		7-48
DMA	DMA(ポート番号)	DMAデータ送信の送信中/非送信中を得る			特殊	7-49

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
DSR	DSR(**ポート番号**)	DR**信号の状態を読む**			**特殊**	7-49
DTR	DTR #**ポート番号**　SET**または** DTR #**ポート番号**　RESET	ER**信号をセット/リセットする**		**デバイスコントロール**		7-49
EDIT	EDIT [**行番号**]	**プログラムの１行を編集する**	**コマンド**			7-50
END	END	**プログラムの実行を終了し、全ポートを閉じる**		**一般**		7-51
EOF	EOF(**ポート番号**)	**指定ポートの受信ポートのバッファが空であるかどうかを調べる**			**特殊**	7-51
ERC	ERC	**エラー(異常)履歴テーブル内のエラーコードを消去する、または発生中のエラー状態を解除する**		**一般**		7-52
ERL	ERL	**エラーが発生した時点で実行中の行番号を得る**			**特殊**	7-52
ERR	ERR	**エラーが発生したエラーコードを得る**			**特殊**	7-53
ERROR	ERROR **エラー番号**	**エラー発生をシミュレートする**		**一般**		7-53
EXP	EXP(**数値式**)	**指数関数を得る**			**算術演算**	7-54
FCS	FCS(**文字式**,[,FCS**タイプ**])	**文字式の誤りチェックコードを得る**			**算術演算**	7-54
FIX	FIX(**数値式**)	**引数で与えた数値の整数部分を得る**			**算術演算**	7-55
FOR …TO… STEP～NEXT	FOR**変数**=**数値式** TO **数値式** [STEP **数値式**]: **プログラム** NEXT **変数**{,**変数**} **または** FOR **変数**=**数値式** TO **数値式** [STEP **数値式**]**行最後** **プログラム** NEXT **変数**{,**変数**}	FOR**からNEXTまでを指定回数繰返す**		**一般**		7-56
FRE	FRE	**ユーザメモリの未使用領域の大きさを与える**			**特殊**	7-57
GOSUB ～ RETURN	GOSUB **行番号** ～RETURN **または** GOSUB **ラベル** ～RETURN	**サブルーチンを呼出し実行し**, RETURN**で戻る**		**一般**		7-57
GOTO	GOTO **行番号またはラベル**	**指定の行番号またはラベルへ分岐する**		**一般**		7-58
HEX$	HEX$(**数値式**)	**引数で与えた**10**進数値を**16**進数の文字列に変換する**			**文字列**	7-59

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
IF	IF 数値式 THEN[ステートメント{:ステートメント}または行番号またはラベル] GOTO 行番号またはラベル [ELSE[ステートメント{:ステートメント}または行番号またはラベル]]	数値式の結果により、実行文または分岐先を選択する		一般		7-60
INKEY$	INKEY$[＃ポート番号]	指定ポートの受信バッファに新たに受信された文字を得る			特殊	7-60
INPUT	INPUT[；] [＃ポート番号,]["プロンプト文"；]変数{,変数} INPUT[；] [＃ポート番号,]["プロンプト文" ,]変数{,変数}	指定ポートの受信バッファからのCR区切りのデータを指定した変数に読み込む		一般		7-61
INPUT$	INPUT$(文字数[, ＃ポート番号])	指定ポートの受信バッファから、指定文字数の文字列を読込む			特殊	7-62
INSTR	INSTR([検索開始位置,]文字式１, 文字式2)	文字式1の中から文字式２を探しその先頭文字の位置を得る			文字列	7-62
INT	INT(数値式)	小数点以下を切捨てた整数値を得る			算術演算	7-63
INTRB	INTRB	割込みが発生したときに実行していた行番号を得る			特殊	7-63
INTRR	INTRR	割込みサブルーチンの中で,元となった割込みのタイプを得る。			特殊	7-64
INTRS	INTRS	発生した割込みを定義した行番号を得る			特殊	7-64
KEY ON/ OFF /STOP	KEY[キー番号]　ONまたは KEY[キー番号]　OFFまたは KEY[キー番号]　STOP	キー入力による割込み(ON　KEY)の許可、禁止、停止を制御する		一般		7-65
LEFT$	LEFT$(文字式 , 文字数)	文字列の左側から文字数分の文字列を得る			文字列	7-65
LEN	LEN(文字式)	文字列の文字数を得る			文字列	7-65
LET	[LET]変数名=式	式の結果を変数に代入する		一般		7-66

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
LINE INPUT	LINE INPUT [；][#ポート番号,]["" プロンプト文"" ；]文字変数	指定ポートの受信バッファから１行分のデータを文字変数に読み込む		一般		7-66
LIST	LIST [行番号１][－行番号2]	プログラムを表示する	コマンド			7-67
LLIST	LLIST[行番号１][－行番号2]	プログラムをプリンタに出力する	コマンド			7-68
LOAD	LOAD #ポート番号[, 通信条件 文字式]	ターミナルからプログラムをASCIIユニットのユーザメモリに転送する	コマンド			7-69
LOC	LOC(ポート番号)	指定ポートの受信バッファにたまっているデータのバイト数を与える			特殊	7-69
LOG	LOG(数値式)	自然対数を与える			算術演算	7-70
LPRINT [USING]	LPRINT[#ポート番号][USINGプリントフォーマット;]式{ [,]式} または LPRINT[#ポート番号][USINGプリントフォーマット;]式{ [;]式} または LPRINT[#ポート番号][USINGプリントフォーマット;]式{ [空白]式}	（指定フォーマットで）デバイス記号をLPRTとして、式の値を指定ポートに出力する		一般		7-71
MID$	MID$(文字式,文字位置[,文字数])	指定の文字位置から文字数分を取出し,得る			文字列	7-72
MID$	MID$(文字式,文字位置[,文字数])= 文字式	指定の文字位置から文字数分を指定文字列で置換する		一般		7-73
MODEL	MODEL	ASCIIユニットの形式を得る			特殊	7-73
NEW	NEW	現在利用のプログラム,変数をすべてクリアする	コマンド			7-74
OCT$	OCT$(数値式)	10進数値を8進数の文字列に変換する			文字列	7-75
ON ALARM	ON ALARM 時間式GOSUB 行番号 または ON ALARM 時間式 GOSUB ラベル	ワンショットタイムアップ後、割込みサブルーチンへ分岐する		一般		7-76
ON COM GOSUB	ON COM [ポート番号] GOSUB 行番号 または ON COM [ポート番号] GOSUB ラベル	指定ポートからの割込み時に割込サブルーチンへ分岐する		一般		7-77
ON ERROR GOTO 行番号	ON ERROR GOTO 行番号 または ON ERROR GOTO ラベル	エラー発生に割込みサブルーチンへ分岐する		一般		7-79
ON 数値式 GOSUB またはGOTO	ON 数値式 GOSUB 行番号{,行番号またはラベル} または ON 数値式 GOSUB ラベル{,行番号またはラベル} または ON 数値式 GOTO 行番号{,行番号またはラベル} ON 数値式 GOTO ラベル{,行番号またはラベル}	数値式の値によって指定された行番号へ分岐する		一般		7-80

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
ON KEY GOTOまたはGOSUB	ON KEY [キー番号] GOTO 行番号またはラベル ON KEY [キー番号] GOSUB 行番号またはラベル	キー入力時、割込みサブルーチンへ分岐する		一般		7-81
ON PC GOSUB	ON PC [割込番号] GOSUB 行番号 または ON PC [割込番号] GOSUB ラベル	PCからの割込みが入ると割込みサブルーチンへ分岐する		一般		7-82
ON TIME$	ON TIME$=時刻文字式 GOSUB 行番号またはラベル	ある時刻になると割込みサブルーチンへ分岐する		一般		7-84
ON TIMER	ON TIMER 時間式 GOSUB行番号またはラベル	インターバルタイムアップ後割込みサブルーチンへ分岐し、これを繰り返す		一般		7-85
OPEN	OPEN #ポート番号,"デバイス記号:[通信速度[,データ長[,パリティ[,ストップビット[,CS_ONまたはOFF[,RS_ONまたはOFF[,DS_ONまたはOFF[,XN_ONまたはOFF]]]]]]]""""	指定ポートを指定条件（デバイス種類、通信条件、制御信号の動作）で開く		デバイスコントロール		7-86
OPTION BASE	OPTION BASE 0 または OPTION BASE 1	配列変数の添字の開始数を１または０に指定する		一般		7-89
OPTION LENGTH	OPTION LENGTH 数値式	文字列変数の内容の最大長(1〜255)を設定する		一般		7-89
PC EGET	PC EGET #共有メモリアドレス,読出CH数,"フォーマット{,フォーマット}"；変数{,変数}	ASCIIユニット自身の共有メモリからデータを変数へ読み出す		一般		7-91
PC EPUT	PC EPUT #共有メモリアドレス,書込CH数,"フォーマット{,フォーマット}"；数値式{数値式}	ASCIIユニット自身の共有メモリへデータを書き込む		一般		7-92
PC GET	PC GET 変数名1[,変数名2]	PCの割付I/Oの出力データの値を変数に読込む		一般		7-93
PC ON/OFF/STOP	PC [割込番号] ON または PC [割込番号] OFF または PC [割込番号] STOP	PCからの割込みの許可、禁止、停止を制御する		一般		7-90
PC PUT	PC PUT 数値式	数値式の値をPCの割付I/Oの入力データエリアに書き込む		一般		7-93
PC QREAD@	PC QREAD"@エリア,アドレス,読出CH数,PCフォーマット{,PCフォーマット},変数{,変数}	PCに対してIOWR命令実行要求を発行し,指定I/Oメモリエリアを読み出す		一般		7-94
PC QWRITE@	PC QWRITE"@エリア,アドレス,書込CH数,PCフォーマット{,PCフォーマット},数値式{,数値式}	PCに対してIORD命令実行要求を発行し,指定I/Oメモリエリアにデータを書き込む		一般		7-97
PC READ または PC READ"@...."	PC READ" PCフォーマット{,PCフォーマット}";変数{,変数} または PC READ" @エリア,アドレス, 書込CH数,PCフォーマット{,PCフォーマット}";変数{,変数}	PCからの書込みデータをフォーマット変換し読み込む		一般		7-99 7-102

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
PC WRITE または PC WRITE"@...."	PC WRITE" PCフォーマット{,PCフォーマット}";変数{,変数} または PC WRITE" @エリア,アドレス, 読出CH数,PCフォーマット{,PCフォーマット}";変数{,変数}	数値式の値をフォーマット変換し PCへ転送する		一般		7-104 7-107
PEEK	PEEK(番地)	メモリ上の指定された番地の内容を読み出す			特殊	7-109
PGEN	PGEN 数値式	利用するプログラムエリア番号に切り替える	コマンド			7-109
PINF	PINF [ALL] または PINF [プログラムNo.]	ユーザメモリエリアの情報を画面に表示する	コマンド			7-110
PMEM	PMEM ONまたは PMEM OFF	ユーザメモリエリアの書込み保護の有効/無効を切り替える	コマンド			7-111
PNAME	PNAME "プログラム名"	プログラムに名前をつける、または消す	コマンド			7-111
POKE	POKE 番地,,数値	ユーザメモリ上の指定番地へデータを書き込む		一般		7-112
PRINT [USING]	PRINT[#ポート番号][USINGプリントフォーマット;]式{ [,]式} PRINT[#ポート番号][USINGプリントフォーマット;]式{ [;]式}または PRINT[#ポート番号][USINGプリントフォーマット;]式{[空白]式}	(指定フォーマットで)式の値を指定ポートに出力する		一般		7-113
PWORD	PWORD文字式	BASICプログラムエリアにパスワードを割り当てる		一般		7-115
RANDOM	RANDOM(式)	式で与えた乱数の元値により乱数の系列を変更する		一般		7-115
READ	READ 変数名[,変数名…]	DATA命令で定義されたデータを変数に読み込む		一般		7-116
REM	REM[コメント文]	プログラム中にコメント文を入れる		一般		7-116
RENUM	RENUM[新行番号][,旧行番号][,増分値]]	プログラムの行番号を付け直す	コマンド			7-117
RESTORE	RESTORE [行番号]または RESTORE [ラベル]	READ命令で読み始めるDATA命令の行番号を指定する		一般		7-118
RESUME	RESUME [行番号]または RESUME [NEXT]	エラー処理終了後、再開行を指定する		一般		7-119
RIGHT$	RIGHT$(文字式,文字数)	文字列の右側から文字数分の文字列を得る			文字列	7-119
RND	RND[(数値式)]	0と1の間の乱数を得る			算術演算	7-120

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
ROMLOAD	ROMLOAD	フラッシュROMからBASICプログラムを読み出しユーザメモリ(RAM)に格納する	コマンド			7-120
ROMSAVE	ROMSAVE	ユーザメモリ(RAM)上のBASICプログラムをフラッシュROMに格納する	コマンド			7-121
ROMVERIFY	ROMVERIFY	RAM内とフラッシュROM内のBASICプログラムを比較する	コマンド			7-121
RTS	RTS　#ポート番号　SET　または RTS　#ポート番号　RESET	RS信号をセット/リセットする		デバイスコントロール		7-122
RUN	RUN[開始行番号][TO 終了行番号]	プログラムの実行を始める。また、指定終了行番号までプログラムを実行する。	コマンド			7-122
SAVE	SAVE　#ポート番号[通信条件文字式]	ユーザメモリエリア内のBASICプログラムをターミナルに読み出す	コマンド			7-123
SEARCH	SEARCH(整数型配列変数,数値式1,[数値式2][,数値式3])	指定配列変数の中に指定数値があるもっとも小さい添字番号を得る			特殊	7-123
SGN	SGN(数値式)	引数の符合を得る			算術演算	7-124
SIN	SIN(数値式)	引数で与えられる正弦(SIN)を得る			算術演算	7-124
SPACE$	SPACE$(文字数)	文字数分の空白文字列を得る			文字列	7-124
SPC	SPC(数値式)	PRINTまたはLPPRINT文で空白を出力する			文字列	7-125
SQR	SQR(数値式)	平方根を得る			算術演算	7-125
STEP	STEP[数値式]	BASICプログラムの指定数の行を実行する	コマンド			7-126
STOP	STOP	プログラムを実行停止する		一般		7-126
STR$	STR$(数値式)	数値を文字列に変換する			文字列	7-127
STRING$	STRING$(数値式,数値式)　または STRING$(数値式,文字式)	文字数分の指定文字列を得る			文字列	7-127
TAB	TAB(桁位置)	PRINT、LPRINT命令内で、指定桁位置まで空白を出力する			文字列	7-128

命　令	構　文	機　能	コマンド	ステートメント	関数	ページ
TAN	TAN(**数値式**)	**引数で与えられた数値の正接**(TAN)**を得る**			**算術演算**	7-128
TIME$	TIME$[="HH:MM:SS"]	**時刻をセットする、または与える**			**特殊**	7-129
TIME$ ON/OFF/STOP	TIME$　ON **または** TIME$　OFF **または** TIME$　STOP	**時刻割込みの許可/禁止/停止を制御する**		**一般**		7-129
TIMER ON/OFF/STOP	TIMER　ON **または** TIMER　OFF **または** TIMER　STOP	**インターバルタイマ割込みの許可/禁止/停止を制御する**		**一般**		7-129
TRACE	TRACE	**トレースバッファの内容を表示する**	**コマンド**			7-130
TROFF	TROFF	**以降の行番号のトレースを停止する**		**一般**		7-130
TRON	TRON[M]	**以降の行番号のトレースを開始する** **M**オプションで、トレースバッファに出力する		**一般**		7-131
VAL	VAL(**文字式**)	**文字列が表す数値に変換する**			**文字列**	7-131
VARPTR	VARPTR(**変数名**)	**変数の格納されているメモリ番地を与える**			**特殊**	7-132
WAIT	WAIT "**待ち時間**"[,**行番号**]	**次の命令語の実行終了までの監視時間を設定する**		**一般**		7-133
WATCH	WATCH [SET　**変数名**] **または** WATCH [DEL　**変数名**]	WATCH**対象の変数をセット**(SET)**、削除**(DEL)**、リスト表示(なし時)する**	**コマンド**			7-134
WHILE/WEND	WHILE **数値式** **プログラム** WEND	**数値式が成立している間、プログラムを実行する**		**一般**		7-135

7－6　BASIC 命令の詳細

以下に、本 ASCII ユニットが使用する BASIC 命令（コマンド、ステートメント、関数）の定義およびその使用例を示します。全命令をアルファベット順に並べてあります。

BASIC 命令の構文の中で使用している規約は、次のとおりです。

1. < >は、表下段の説明欄で定義されている用語、または共通の用語であることを示します。
   例えば、<式>は、式の定義が、表中の説明欄で説明されていることを示します。
2. [ ] は、省略可能である（オプション項目である）ことを示します。
   例えば、PINF [<プログラム番号>]と記されている場合、PINF コマンドでは <プログラム番号> が省略可能であることを示します。
3. | |は、複数の選択肢間の選択項目であることを示します。 | A / B | は、A または B であることを示します。
   例えば、ALARM | ON / OFF / STOP | と記されている場合、ALARM の後に、ON、OFF、STOP のいずれかを記述します。
4. { }は、繰り返しが可能な項目であることを示します。{ }で囲まれる項目は、省略または 1 回以上繰り返すことが可能です。
   例えば、変数 {,変数 } の場合、変数の後、変数を , で区切って繰り返すことが可能であることを示します。

@IF	
構文：	@IF <数値式>THEN [<プログラム>] [ELSE [<プログラム>]] ENDIF
機能：	ステートメント。数値式の結果に基づいてプログラムの流れを制御します。このステートメントは、THEN 以降と ELSE 以降で複数の行をサポートする点で IF 文と異なります。また、ネスティングが可能です。
説明：	<数値式> の結果によって、どちらの分岐プログラム（複数行）を実行するかを決めます。<数値式> の結果が真（0 以外）の場合、THEN に続く複数行が実行されます。0（偽）の場合、ELSE に続く複数行が実行されます。すべての@IF 命令は ENIDIF で終了させる必要があります。 ELSE 文がない場合、<数値式> の結果が 0（偽）のとき、ENDIF 以降の文が実行されます。
参考：	ENDIF が対応する@IF の前に存在するか、または ENDIF が存在しないままのプログラムの場合（@IF がすでに存在していた場合）、”MISMATCH：@IF/ENDIF”（@IF/ENDIF 不一致エラー, エラーコード：B029）が発生します。
例：	> 10 A= -5 > 20 @IF A<0 THEN > 30 B = -A > 40 PRINT ”ABS OF”; A; ” IS ”; B > 50 ELSE > 60 PRINT ”ABS OF”; A; ” IS ”; A > 70 PRINT ”END” > 80 ENDIF > 90 END > RUN ABS OF-5 IS 5
参照：	IF

ABS	
構文：	ABS(<数値式>)
機能：	関数。数値式の絶対値を計算します。
説明：	<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。入力（引数）および出力（戻り）の範囲は、倍精度実数の許容範囲と同等です。出力（戻り）の型は、入力（引数）の型と同じです。
例：	> 10 A = -4 > 20 PRINT ABS(A) > RUN 4
参照：	

ACOS	
構文：	ACOS(<**数値式**>)
機能：	**関数。数値式の逆余弦を計算します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。入力（引数）の有効範囲は、**-1～1 **です。出力（戻り）の範囲は、ラジアン単位で、**0～π **です。入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。入力（引数）の型が、倍精度実数型の場合、出力（戻り）の型は倍精度実数型になります。**
例：	> 10 A = 0.4 > 20 PRINT ACOS(A) > RUN 1.15928
参照：	ASIN, ATN

ALARM ON/OFF/STOP	
構文	ALARM | ON OFF STOP |
機能：	**ステートメント。**ON **アラーム割込み（**ON　ALARM **命令によるワンショットタイマ割込み）を許可、禁止、または停止します。**
説明：	ON **は、**ON **アラーム割込みを許可（マスク解除）にします。この** ALARM ON **命令から** ALARM OFF **命令までを実行中に、タイマがアップし割込みが発生すると、定義された 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降の発生したすべての** ON **アラーム割込みは、無視されます。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくのみで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンへの分岐を実行します。**
参考：	1. ON **アラーム割込みは、１つの** ON ALARM GOSUB **に対して** 1 **回だけ発生します。複数の** ON ALARM GOSUB **文を設定すると、最後の** ON ALARM GOSUB **文が有効になります。** 2. ON **アラーム割込みの分岐処理後、** ON **アラーム割込みは無効になります。以降再度** ON ALARM **命令によりアラーム割込みが定義されるまで** ON **アラーム割込みは発生しません。** 3. ALARM ON/OFF/STOP **命令は、**ON ALARM **命令を実行した後にのみ実行可能です。** 4. ON ALARM GOSUB **文の実行直後は割込み禁止状態です。**
例：	> 10 ON ALARM 30 GOSUB 1000 > 20 ALARM ON > 30 GOTO 30 > 1000 PRINT ”ALARM INTERRUPT OCCURRED” > 1010 RETURN > RUN ALARM INTERRUPT OCCURRED
参照：	ON ALARM

ASC	
構文：	ASC(<**文字列式**>)
機能：	**関数。<文字列式> の先頭の文字の** ASCII **文字コードを返します。**
説明：	**出力（戻り）は、文字列式の先頭の文字の** ASCII **文字コードを表わす整数です。**10 **進数で返します。**
参考：	1. **出力（戻り）値の範囲は、**0〜255 **です。** 2. **この逆の機能が、**CHR$**関数です。**
例：	> 10 B$ = ”B” > 20 PRINT ASC(B$) > RUN 66
参照：	CHR$

ASIN	
構文：	ASIN(<**数値式**>)
機能：	関数。数値式の逆正弦を計算します。
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。入力（引数）の有効範囲は、**-1〜1 **です。出力（戻り）の範囲は、ラジアン単位で、-**π/2〜π/2 **です。** **入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。入力（引数）の型が、倍精度実数型の場合、出力（戻り）の型は倍精度実数型になります。**
例：	> 10 A = 0.4 > 20 PRINT ASIN(A) > RUN .411517
参照：	ACOS、ATN

ATN	
構文：	ATN(<**数値式**>)
機能：	**関数。数値式の逆正接を計算します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。出力（戻り）の範囲は、ラジアン単位で、-**π/2〜π/2 **です。** **入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。** **入力の型が倍精度実数型の場合、出力の型は倍精度実数型になります。**
例：	> 10 A = 0.4 > 20 PRINT ATN(A) > RUN .380506
参照：	ACOS, ASIN

AUTO	
構文：	AUTO [<行番号>] [, <増分>]]
機能：	コマンド。プログラム入力時に、行番号を自動的に生成します。
説明：	<行番号> は、1〜65535 の範囲の整数です。 <増分> は、1〜65535 の範囲の整数です。 <行番号>を最初の番号とし、<増分>の値づつ増加させた行番号を自動的に発生します。 <行番号> と <増分> を省略すると、いずれも 10 になります。
参考：	1. AUTO コマンドは、[Ctrl] +[X] キーにより解除できます。 2. プログラムとしてすでに存在する行番号を指定したときは、行番号の直後に、アスタリスク（*）が表示されます。このとき、新規の行内容を上書きすることができます。 3. 新規の行番号を入力し、その後に CR キーを押すと、新しい行番号を入力できます。新規の行内容（プログラム）を入力せずに CR キー押すと、次の増分された行番号が表示されます。
例：	> AUTO 10,10
参照：	EDIT

BITON/BITOFF	
構文：	BITON <整数型変数>,<ビット位置> BITOFF <整数型変数>,<ビット位置>
機能：	ステートメント。整数型変数の指定ビットをオン（1）またはオフ（0）にします。
説明：	<ビット位置> は、0〜31 の範囲の整数値です。 BASIC で、ビット操作をするときに使用します。
参考：	0 の <ビット位置> は最下位ビットを示します。<ビット位置> は整数に対しては 0〜15 の範囲、倍長整数に対しては 0〜31 の範囲で使用可能です。
例：	> 10 N% = 182 > 20 PRINT N% > 30 BITON N%,3 > 40 PRINT N% > 50 BITOFF N%,5 > 60 PRINT N% > RUN 182 190 158
参照：	

お願い BITON および BITOFF では、配列変数は使用できません。

BRKPT	
文：	BRKPT [SET | DEL] <行番号> | ALL
機能：	コマンド。指定行番号にブレイクポイント（一時停止）を設定、指定行番号からブレイクポイントを削除、またはプログラム内に現在設定されている全ブレイクポイントをリスト表示します。
説明：	<行番号> は、BASIC プログラム内の任意の行です。有効範囲は 1〜65535 です。 SET は、指定 <行番号> に、ブレイクポイントを設定します。 DEL は、指定 <行番号> から、ブレイクポイントを削除します。 BRKPT コマンドを SET または DEL なしで使用すると、このコマンドは、現在設定されているブレイクポイントをすべてリスト表示します。 最大 255 個のブレイクポイントを任意の行番号に設定できます。ブレイクポイントを設定した行番号にプログラムが到達すると、その行番号の行を実行する直前で、プログラムが停止（ブレイク）します。
参考：	存在しない行番号を指定すると、”UNDEFINED LINE NUMBER”（行番号未定義エラー，エラーコード：B008）が発生します。 BRKPT SET ALL を実行すると、”NO SUPPORT”（サポートなしエラー，エラーコード：0064）が発生します。 ユーザメモリエリアに空き実行エリアがなくなった場合、255 個まで設定できないこともあります。 割込みプログラム中にブレイクポイントを設定して停止することはできますが、そこからの再開はできません。
例：	> BRKPT SET 200
参照：	CONT, STEP, STOP, END

CDBL	
構文：	CDBL(<数値式>)
機能：	関数。<数値式> を倍精度実数に変換します。
説明：	<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。<数値式> が、整数、倍長整数、単精度実数である場合、CDBL は数値式を倍精度実数に変換します。<数値式> がすでに倍精度実数である場合、返される値の精度は変換前と同じです。
参考：	倍精度実数へ変換しても単精度と同じ桁数しか表示されないことがありますが、数値としては倍精度桁数持っています。 表示上、0 が続く場合は上記のとおり単精度の桁数までしか出力されないことがあります。
例：	> 10 A# = CDBL(22)/CDBL(7) > 20 B# = 22!/7! > 30 PRINT A#, B# > RUN 3.14285714285714 3.14286
参照：	CINT, CSNG

CHR$	
構文：	CHR$(<数値式>)
機能：	関数。<数値式> を ASCII 文字コードとみなして、文字に変換します。
説明：	<数値式> は、0～255 の範囲の任意の整数です。数値式が実数値の場合は、小数点以下を四捨五入したものを値とします。ASCII 文字コードを 10 進数に変換した値とみなして、対応する文字に変換します。
参考：	1. <数値式> が指定範囲外にある場合、"ILLEGAL FUNCTION CALL"（関数コール不正エラー, エラーコード：B005）、または "OVERFLOW"（桁溢れエラー, エラーコード：B006）が発生します。 2. この逆の機能が ASC 関数です。 3. 例えば、CHR$（67.4）は 67 とみなし「C」を返し、CHR$（67.5）は 68 とみなし「D」を返します。
例：	> 10 A$ = CHR$(67) > 20 PRINT A$ > RUN C
参照：	ASC

CINT	
構文：	CINT <数値式>
機能：	関数。<数値式> を小数点以下四捨五入して整数値に変換します。
説明：	<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。入力（引数）の有効範囲は、整数の有効範囲（-32768～32767）になります。
参考：	小数点以下は、正の値の場合上に四捨五入され、負の値の場合下に四捨五入されます。 例えば、CINT（67.5）は、68 を返します。CINT（-67.5）は、-67 ではなく、-68 を返します。
例：	> 10 A% = CINT(67.45) > 20 B% = CINT(67.55) > 30 PRINT A%, B% > RUN 67　　68
参照：	CDBL, CLNG, CSNG, FIX, INT

CLEAR	
構文：	CLEAR
機能：	**ステートメント。すべての数値変数と文字変数を初期化します。**
説明：	**数値変数はすべて** 0 **に初期化されます。文字変数はすべて空に初期化されます。**
参考：	**文字変数エリアの割付け方式に応じて、この** CLEAR **命令による処理が以下のように異なります。** **注：文字変数エリアの割付け方式についての詳細は、**7 -2 **項を参照してください。** <**ダイナミック方式の場合**> **・開いているすべてのポートを閉じます。** **・各文字変数のために都度確保されたエリアがすべて開放されます。** <**スタティック方式の場合**> **・開いているすべてのポートを閉じます。** **・ユーザメモリ上に固定的に確保された変数エリアは確保されたまま、中の値がすべて空きになります。**
例：	CLEAR
参照：	

CLNG	
構文：	CLNG(<**数値式**>)
機能：	**関数。**<**数値式**> **を倍長整数に変換します。**
説明：	<**数値式**> **は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。入力の有効範囲は、倍長整数の範囲、**-2147483648**～**2147483647 **になります。**
参考：	**小数点以下は、正の場合上に四捨五入され、負の場合下に四捨五入されます。例えば、**CLNG**（**67.5**）は、**68 **を返しますが、**CLNG**（**-67.5**）は**-67 **ではなく、**-68 **を返します。**
例：	10 A& = CLNG(40000.45) 20 B& = CLNG(40000.55) 30 PRINT A&, B& > RUN 40000 40001
参照：	CDBL, CINT, CSNG, FIX, INT

CLOSE	
構文：	CLOSE [# <ポート式>]
機能：	ステートメント。ポートを閉じます。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。ポート式を省略する と、すべてのポートを閉じます。 ポートを閉じる際、ポートの送信バッファに残っているすべてのデータが掃き出され（送信され）、ポートの受信バッファに残っているすべてのデータはクリアされます。 一度ポートを閉じると、再び OPEN 命令によってポートを開くまでデータ送受信はできません。 なお、END と NEW の各コマンドを実行すると、自動的にすべてのポートを閉じます。STOP コマンドはポートを閉じません。
参考：	1. ポート#3 は形 C200H-ASC31 のターミナルポートに対応しています。 2. ポート#3 は形 C200H-ASC11 または形 C200H-ASC21 で指定されると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。 3. ポートを閉じる場合の RS と DR の信号線の状態は、4-1 項「外部汎用機器とのデータ交換」の「■伝送制御信号のタイムチャート」を参照してください。
例：	CLOSE #2
参照：	END, NEW, OPEN

CLS	
構文：	CLS [#<ポート式>]
機能：	ステートメント。画面をクリアします。
説明：	<ポート式> は、1〜3 の整数を返す式です。 ポート式を省略すると、ターミナルポートのポート No.（形 C200H-ASC11/21 の場合#1、形 C200H-ASC31 の場合#3）となります。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. ポート#3 を形 C200H-ASC11 または形 C200H-ASC21 に指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	CLS
参照：	

COM ON/OFF/STOP	
構文：	COM [<**ポート式**>] | ON OFF STOP |
機能：	**ステートメント。**ON COM **命令による通信割込みを許可、禁止または停止にします。**
説明：	<**ポート式**> **は、**1～2 **の範囲の整数を返す式です。** **ポート番号を省略すると、ポート** No #1 **とみなします。** ON **は、通信割込みを許可（マスク解除）にします。この** COM ON **命令から** COM OFF **命令までを実行中に、通信割込みが発生すると、定義された 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降の発生したすべての通信割込みは、無視されます。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくだけで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンへの分岐を実行します。**
参考：	1. COM ON/OFF/STOP **命令は、**ON COM **命令を実行した後に有効となります。** 2. ON COM GOSUB **文の実行直後は割込み禁止状態です。**
例：	COM 1 STOP
参照：	ON COM

CONT	
構文：	CONT
機能：	**コマンド。停止（ブレイク）位置から実行を再開します。**
説明：	**停止（ブレイク）中にプログラムを変更した場合、また外部機器とのデータ送受信中にブレイクが発生した場合は、再度停止した行から プログラムを再開できません。** **割込みプログラム中でブレイクした場合も、**CONT **で実行再開できません。**
参考：	**プログラムが再開できない場合、**"CANNOT CONTINUE"**（継続実行不可エラー**, **エラーコード：**B017**）が発生します。**
例：	CONT
参照：	STEP, STOP, END, BRKPT

COS	
構文：	COS(<数値式>)
機能：	関数。数値式の余弦（COS）を計算します。
説明：	<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。 出力の（戻り）型は、引数が整数型または単精度実数型の場合、単精度実数型です。引数が倍精度実数型の場合、倍精度実数型です。 出力（戻り）は、-1〜1 の範囲になります。
参考：	引数は、ラジアン単位で指定してください。
例：	> 10 A = 0.4 > 20 PRINT COS(A) > RUN .921061
参照：	SIN, TAN

CSNG	
構文：	CSNG(<数値式>)
機能：	関数。数値式を単精度実数型数値に変換します。
説明：	<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。 <数値式> が整数、倍長整数、倍精度実数である場合、CSNG は数値式を単精度実数型数値に変換します。 <数値式> がすでに単精度実数である場合、返される値の精度は変数前と同じです。
例：	> 10 A# = 1.23456789 > 20 A! = A# > 30 B! = CSNG(A#) > 40 PRINT A#, A!, B! > RUN 1.23456789 1.23457 1.23457
参照：	CDBL, CINT

CTS	
構文：	CTS(<ポート式>)
機能：	関数。指定されたポートの CS 信号線の状態を読み出します。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。 CS は、指定したポートの CS 信号線が ON である場合-１を返し、CS 信号線が OFF である場合は 0 を返します。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. 形 C200H-ASC11 にポート#3 を指定するか、または形 C200H-形 C200H-ASC21 にポート#2（RS-422A/485）またはポート#3 を指定すると、”ILLEGAL FUNCTION CALL”（関数不正コールエラー, エラーコード：B005）のエラーが発生します。
例：	> 10 IF CTS(2) THEN PRINT ”PORT 2 READY”
参照：	DSR, DTR, RTS

DATA	
構文：	DATA [<定数> {, <定数> }]
機能：	ステートメント。以降の READ 命令で使用可能な数値定数と文字定数を定義します。
説明：	<定数> は、任意の数値定数または文字定数です。 「,」、「:」および空白を定数に組み込むためには、引用符（ ” ）でこれらを囲んでください。 DATA 命令は常に READ 命令とセットで使用します。定数の型、および順序は、READ 命令で指定する、変数の型および順序と一致していることが必要です。 各 DATA 命令で定義された定数は、READ 命令により順に変数に読み込まれます。DATA 命令の最後の定数が読み出されると、次に読み出される定数は、次の DATA 命令が存在する場合はその中の最初の定数になります。
参考：	DATA 文には、コメント文はつけられません。
例：	> 10 DATA 10, ”HELLO”, 1.6, ”WORLD” > 20 READ A, B$, C, D$ > 30 PRINT B$+” ”+D$ > 40 PRINT A+C > RUN HELLO WORLD 11.6
参照：	READ, RESTORE

DATE$	
構文：	DATE$ [= <日付文字列式>]
機能：	特殊関数。内部クロックの日付を返すか、または内部クロックの日付を設定します。
説明：	<日付文字列式> は、「yy/mm/dd」形式です。yy は年で、有効範囲は 00〜99 です。mm は月で、有効範囲は 01〜12 です。dd は日で、有効範囲は 01〜31 です。 一度、DATE$ [= <日付文字列式>] で設定すれば、DATE$は、現在の日付を返します。
参考：	<日付文字列式> が有効日付でない場合、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	DATE$ = ”97/06/26”
参照：	DAY, TIME$

DAY	
構文：	DAY [= <数値式>]
機能：	特殊関数。現在の曜日を返すか、または曜日を設定します。
説明：	<数値式> は、曜日を表す次のコード（整数）です。 例えば、次のように決めておいて使用します。 コード：曜日 0：日曜 1：月曜 2：火曜 3：水曜 4：木曜 5：金曜 6：土曜 一度、 DAY [= <数値式>] で設定すれば、DAY には、一日経つごとに+1 された値（0〜6 の繰り返し）がセットされます。 DAY 関数は、現在の DATE$の曜日と無関係に設定することが可能です。
参考：	<数値式> が上記のコードでない場合、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。
例：	DAY = 4
参照：	DATE$, TIME$

DEF FN	
構文：	DEF FN <関数名 > [(<変数名 1> [, <変数名 n>])] = <式>
機能：	ステートメント。ユーザ作成の BASIC 関数を定義します。
説明：	<関数名> は、定義された関数の名前です。これは、任意の数値変数名または文字変数名で、FN＋名前が関数名となります。名前には予約語から始まる文字列を使用できません。名前の最後に型宣言文字（$ % & ! #）を使用できません。 <変数名 1>　[, <変数名 n>] は、ユーザ定義の BASIC 関数に入力する引数（数値変数または文字変数）です。任意の変数名を使用できます。ただし、定義する関数名と同じ名前をもつ引数をもつことができません。引数として使用した変数名をプログラムの他の行で使用してもかまいません。 <式> は、<変数名 1> [, <変数名 n>] を使用した数値式または文字式です。
参考：	ここに定義する引数の型と、BASIC 関数として呼び出して使用する際の引数の型は、一致していなければなりません。一致していない場合、”TYPE MISMATCH”（型不一致エラー，エラーコード：B013） が発生します。 定義した BASIC 関数を呼び出すには、FN+関数名で呼び出します。 ユーザ定義関数の出力（戻り）は、呼び出すときに、変数に割り当てる必要があります。そうでないと、”SYNTAX ERROR IN LINExxxxx”（文法エラー，エラーコード：B002）が発生します。 例： A=FNMYFUNC（1,2） これは、下記の例によって定義される MYFUNC 関数を呼び出します。
例：	> 10 DEF FNMYFUNC(A,B) = A+B > 20 PRINT FNMYFUNC(1,2) > RUN 3
参照：	

DEFINT/DEFLNG/DEFSNG/DEFDBL/DEFSTR	
構文：	DEF | INT | LNG | SNG | DBL | STR | **<文字>** [-**<文字>**] { , **<文字>** [-**<文字>**] }
機能：	**ステートメント。指定された文字で始まる変数型を定義します。**
説明：	**指定した <文字> 、または指定した <文字>-<文字>（文字範囲）で始まる変数を、指定した型に宣言します。** **変数の型は、整数型（**INT**）、倍長整数型（**LNG**）、単精度実数型（**SNG**）、倍精度実数型（**DBL**）、または文字列型（**STR**）のいずれかです。**
参考：	1. **変数の型が、型定義文字（**%、&、!、#または$**）により宣言されている場合、この** DEFINT/DEFLNG/DEFSNG/DEFDBL/DEFSTR **により宣言される型より優先されます。** 2. **型宣言がされていない変数は、単精度実数型と見なされます。** 3. OPTION LENGTH **命令を使用するプログラムでは、**DEF **命令で定義する変数名を、この** DEF **命令より前で使用することはできません。**
例：	> 10 DEFINT A > 20 A = 1.2 > 30 PRINT A > RUN 1
参照：	

DEL		
構文：	DEL | **<行番号>** | • | [− [| **<行番号>** | • |]]	
機能：	**コマンド。指定プログラム行または行範囲を削除します。**	
説明：	**<行番号> は、**BASIC **プログラムの任意の有効行です。有効範囲は、**1〜65535 **です。**	
参考：	1. **ピリオド（.）は、現在の行番号の代わりに使用できます。** 2. **行範囲の最初の数字は、**2 **番目の数字より小さくなければなりません。そうでないと、**"ILLEGAL FUNCTION CALL"**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。**	
例：	DEL 100-200 DEL 100- DEL -600	100〜200**（両端を含む）の行範囲を削除します。** 100 **からプログラムの終わりまでの行範囲を削除します。** **プログラムの始まりから行番号** 600 **までの行範囲を削除します。**
参照：		

DIM	
構文：	DIM<変数> (<添字最大値> {, <添字最大値> }) {, <変数>(<添字最大値> {, <添字最大値> }) }
機能：	ステートメント。配列変数の次元数と添字の最大値を指定します。（配列変数のための必要なメモリを割付けます。）
説明：	<変数> は、任意の有効な変数名です。 配列変数に宣言されると、数値型配列変数は 0 に初期化されます。文字列型配列変数は、空（NULL）に初期化されます。配列変数は、最大 255 個の添字（次元）をもつことができます。ただし、ASCII ユニット内の使用可能なメモリを超えて指定することはできません。 <添字最大値> は、0〜65335 の範囲の任意の有効な整数です。文字変数エリアの割付方式がスタティック方式の場合は、定数でなければなりません。 注：配列変数を DIM 命令なしで（配列の宣言をせずに）使用すると、添字の最大値は 10 となります。 注：DIM 命令は、すべての代入式よりも前で実行する必要があります。
参考：	配列変数とは、変数名のひとつで、複数のデータを指定できる変数のことです。変数名（数値型または文字型）のあとに、カッコでくくった数値で複数のデータを指定します。この数値のことを、添字と呼びます。添字の数が 1 つを 1 次元、2 つを 2 次元・・と呼びます。 例：DIM A$(3)：配列変数 A$、文字列を値とした、次元数が 1 次元の配列。A$(0) , A$(1), A$(2), A$(3)を変数として使用できます。 例：DIM B(3,2)：配列変数 B、数値を値とした、次元数が 2 次元の配列。B(0,0), B(1,0), B(2,0), B(3,0), B(0,1), B(1,1), B(2,1), B(3,1), B(0,2), B(1,2), B(2,2), B(3,2)を変数として使用できます。 注：上記のように、配列の添字は、0 から始まります。配列の添字を 1 から始めたいときは、OPTION BASE 命令を使用します。
例：	> 10 DIM A(5) > 20 FOR I = 0 TO 5 > 30 A(I)=I > 40 PRINT A(I) > 50 NEXT I > RUN 0 1 2 3 4 5
参照：	OPTION BASE

DMA	
構文：	DMA(<ポート式>)
機能：	関数。指定されるポートの DMA データ送信の状態（実行中／停止中）を読み出します。
説明：	<ポート式> は、1〜2 の範囲内の整数を返す式です。 指定ポートにおいて、DMA データ送信実行中の場合、-１を返し、DMA データ送信停止中の場合、0 を返します。
参考：	ポート式が指定範囲外の場合、または指定ポート番号が DMA データ送信不可の場合（形 C200H-ASC11 または形 C200H-ASC21 上のポート#1、形 C200H-ASC31 のポート#3 の）場合、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。
例：	IF DMA(2) THEN PRINT ”DMA OF PORT 2 IS ACTIVE”
参照：	

DSR	
構文：	DSR (<ポート式>)
機能：	関数。指定されたポートの DR 信号線の状態を読み出します。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。 DSR は、指定されるポートの DR 信号線が ON である場合は-1 を返し、DR 信号線が OFF である場合は 0 を返します。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. 形 C200H-ASC11 にポート#3 を指定するか、または形 C200H-ASC21 にポート#2（RS-422A/485）またはポート#3 を指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。
例：	IF DSR(2) THEN PRINT ”DSR OF PORT 2 IS ACTIVE”
参照：	CTS, DTR, RTS

DTR	
構文：	DTR ＃<ポート式> { SET / RESET }
機能：	ステートメント。指定されるポートの ER 信号線をセット（ON）またはリセット（OFF）します。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。 SET を指定すると、ER 信号線が ON になります。RESET を指定すると、ER 信号線が OFF になります。ポートは OPEN 状態でないと使用できません。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. 形 C200H-ASC11 にポート#3 を指定するか、または形 C200H-ASC21 にポート#2（RS-422A/485）またはポート#3 を指定すると、”BAD PORT NUMBER”（ポート番号不正エラー，エラーコード：0050）が発生します。
例：	DTR #2 SET
参照：	CTS, DSR, RTS

EDIT	
構文：	EDIT | <行番号> | •
機能：	コマンド。BASIC プログラムの指定行を編集します。
説明：	BASIC プログラムの編集を可能にします。指定行番号のプログラムの範囲内で、文字挿入、削除、コピーおよびカーソル移動ができます。 エディタはラインエディットモードです。 EDIT.の場合は最も新しく表示した行、または最も新しく入力した行のプログラムが表示されます。 PNAME コマンドによって入力されたプログラムを、編集することもできます。 次の編集キーは、編集中に使用できます。

キー	動作
右向きキー（→）	カーソルを右に移動します。カーソルが行の最後の位置にある場合、このキーは無効です。
左向きキー（←）	カーソルを左に移動します。カーソルが行の最初の位置にある場合、このキーは無効です。
下向きキー（↓）	現在、編集中の行を BASIC プログラムに入力し、次のプログラム行を表示します。現在行に続くプログラム行が存在しない場合、編集を終了します。
上向きキー（↑）	現在、編集中の行を BASIC プログラムに入力し、以前のプログラム行を表示します。現在行の前にプログラム行が存在しない場合、編集を終了します。
Ctrl+O （挿入/上書き）	エディタを挿入モードと上書きモードで切り替えます。 挿入/上書きモードのステータスは表示されません。
DEL または BS	カーソルの左側の文字を削除し、次の文字を左に移動します。このキーは、行の始まりでは無効です。
CR（改行）	CR（改行）によって、現在編集中の行を BASIC プログラムに入力します。 現在表示している行番号を変更して、 CR（改行）キーを押すと、その行番号が新たに作成されます。
Ctrl+X	エディットを終了させ、コマンドモードに戻ります。編集中だった行は、編集前のものが残ります。

参考：	行番号だけを修正すると、その修正前の行内容が、修正後の行内容として、コピーされます。 指定した行番号が存在しない場合、”UNDEFINED LINE NUMBER”（行番号未定義エラー, エラーコード：B008）が発生します。
例：	EDIT 200　　This edits line 200 of the active program
参照：	

END	
構文：	END
機能：	ステートメント。プログラム実行を終了し、コマンドモードに戻ります。現在開いているポートはすべて閉じられます。
説明：	END は、BASIC プログラムの任意の場所で使用できます。 開いているポートは閉じられ、CLOSE 命令と同様に、送信バッファに残っていたデータは、掃き出されてからクローズされ、受信バッファに残っていたデータは、クリアされてからクローズされます。 プログラムの最後の END 命令は、省略できます。ただし、この場合、ポートは閉じません。
例：	> 10 INPUT A > 20 @IF A = 1 THEN GOSUB 1000 > 30 ELSE > 40 PRINT ”A WAS NOT 1” > 50 ENDIF > 60 END > 1000 PRINT ”A WAS 1” > 1010 RETURN
参照：	STOP

EOF	
構文：	EOF（ <ポート式> ）
機能：	関数。指定ポート番号の受信バッファが、空きであるかどうかを調べます。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。 指定ポート番号の受信バッファが、空きである場合、-1 を返します。空きでない場合、0 を返します。
参考：	1. 指定ポートは、OPEN 命令でオープンされ、そのデバイス記号指定で、受信用に設定されている必要があります。 2. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 3. 形 C200H-ASC11 または形 C200H-ASC21 でポート#3 を指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	IF EOF(2) THEN PRINT ”PORT 2 IS EMPTY”
参照：	

ERC	
構文：	ERC
機能：	ステートメント。 ASCII ユニット内の、発生した最大 30 個のエラーコードを記憶したエラーコード（異常履歴）テーブル からエラー種別／コードエラーをクリアします。
説明：	エラーコード（異常履歴）テーブルから、エラーコードをクリアします。 コマンドとして実行した場合には、すべてのエラー（異常履歴テーブルを含む）をクリアします。 プログラム中でステートメントとして実行した場合は、現在発生しているエラー状態を解除し、そのエラーは旧エラー扱いとして異常履歴テーブルに登録されます。 ERC コマンドによって 90〜99 のエラー（DM 設定エラー）をクリアすることはできません。
参考：	エラーコード（異常履歴）テーブルは、ASCII ユニット内で発生したエラー（異常）のエラーコード／種別を最大 30 個記憶しておくテーブルです。エラーコードは、ASCII ユニット内で発生したエラー内容を示します。詳細は、9-1 項「エラーメッセージ一覧」を参照してください。
例：	ERC
参照：	ERL, ERR

ERL	
構文：	ERL
機能：	特殊関数。エラーが発生した行番号を返します。
説明：	ERL は、読み出し専用の関数です。 ON ERROR GOTO 命令、ERR 関数と組み合わせて、エラー割込みサブルーチンでこの関数の内容を確認することで、エラー内容（コード・発生行）に応じたエラー処理をするときなどに、使用します。
参考：	1. コマンドモードでエラーが発生した場合は、ERL は 0 を返します。 2. エラー発生時の割込みサブルーチン内でこの ERL 関数を使用すれば、プログラムの流れ制御が可能です。
例：	>10 ON ERROR GOTO 1000 >20 A=A/0 >30 END >1000 PRINT ”ERROR # ”;ERR;” OCCURRED AT LINE ”;ERL >1010 END >RUN ERROR #11 OCCURRED AT LINE 20
参照：	ERC, ERR

ERR	
構文：	ERR
機能：	**特殊関数。発生したエラーのエラーコードを返します。**
説明：	ERR **は、読み出し専用の関数です。エラーコードについては、**9-1 **項「エラーコード一覧」を参照してください。** ON ERROR GOTO **命令と組み合わせて、エラー割込みサブルーチンでこの関数の内容を確認することで、エラー内容に応じたエラー処理をするときなどに、使用します。**
参考：	1. ERR **は、エラーが発生しなかった場合は、デフォルト値** 0 **が格納されています。** 2. **エラー発生時の割込みサブルーチン内でこの** ERR **関数を使用すれば、プログラムの流れ制御が可能です。**
例：	>10 ON ERROR GOTO 1000 >20 A=A/0 >30 END >1000 PRINT ”ERROR # ”;ERR;” OCCURRED AT LINE ”;ERL >1010 END >RUN ERROR #11 OCCURRED AT LINE 20
参照：	ERC, ERL

ERROR	
構文：	ERROR <**エラー番号**>
機能：	**ステートメント。エラー発生をシミュレーションします。**
説明：	<**エラー番号**> **は、**1〜255 **の範囲の整数です。** **この** ERROR **命令を実行すると、指定したエラー番号のエラーが発生したとみなして、そのエラーコードが割付** I/O **の** n+7CH **に格納され、同時に（エラーコードによっては）エラーメッセージが画面に表示されます。また、エラー（異常）履歴テーブルに、発生したエラーコードが格納されます。**
参考：	1. **未使用のエラー番号を指定すると、**”UNDEFINED ERROR”**（エラーコード：＜エラー番号＞）が発生します。** 2. ON ERROR GOTO **命令と組み合わせて使用することで、エラー発生をシミュレーションしながらエラー割込みサブルーチン内のエラー処理プログラムを作成することが可能です。**
例：	>10 ON ERROR GOTO 1000 >20 ERROR 6 >30 END >1000 PRINT ”ERROR # ”;ERR;” OCCURRED AT LINE ”;ERL >1010 END >RUN ERROR #6 OCCURRED AT LINE 20
参照：	ON ERROR GOTO, ERL, ERR, ERC

EXP	
構文：	EXP(<**数値式**>)
機能：	**関数。基数をｅとした指数を計算します。**
説明：	**<数値式> がＸの場合、ｅ^{x}を返します。** **<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **<数値式> が整数または単精度実数の場合、出力（戻り）の型は単精度実数です。<数値式>が倍精度実数の場合、出力（戻り）の型は倍精度実数です。**
例：	> 10 A = EXP(1) > 20 PRINT A > RUN 2.71828
参照：	

FCS	
構文：	FCS(<**文字列式**> [, <**誤りチェックコード種類** >])
機能：	**関数。文字列式の誤りチェックコードを計算します。**
説明：	FCS **は、文字列式のすべてのバイトに対して次のいずれかの誤りチェックコードの計算をします。** <**誤りチェックコード種類**> **は、**0～9 **の範囲の整数値です。次の中から 選択します。** **なお、出力（戻り）の型は、チェックコードのデータ形式** BIN**（バイナリ）または** ASCII **を選択できます。**BIN **形式 は、整数を返し、**ASCII **形式は、文字列を返します。**
参考：	1. CRC-CCITT **の算出ロジックは** XMODEM **方式です。** 2. CRC-16 **の算出ロジックは** MODBUS **方式です。例えば、文字列データに対し、データは次のように扱われます。** ”1 2 3 4”=<31 32 33 34>$_{x^n+x^{n-1}\cdots x^1+x^0}$
例：	> 10 B$ = ”RD89” > 20 PRINT FCS(B$,1) > RUN 17
参照：	

誤りチェックコード種類	**計算方法**	**チェックコードデータ**	
		データ形式	サイズ（バイト）
0	LRC（水平パリティ）	BIN	1
1		ASCII	2
2	CRC-CCITT	BIN	2
3		ASCII	4
4	SUM（1 **バイト**）	BIN	1
5		ASCII	2
6	SUM（2 **バイト**）	BIN	2
7		ASCII	4
8	CRC-16	BIN	2
9		ASCII	4

FIX	
構文：	FIX(<**数値式**>)
機能：	**関数。入力（引数）として指定する数値式の整数部のみをそのまま返します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **出力（戻り）の型は、整数または倍長整数です**
参考：	1. FIX **は、数値の小数点以下の四捨五入や切り捨てをせず、そのまま整数部を返します。** 2. **入力（引数）の値が正の場合、**FIX **は** INT **と同じ値を返します。** **入力（引数）の値が負である場合、**FIX **は** INT **より** 1 **だけ大きい値を返します。例えば、**FIX(-6.1) **の場合、**-6 **を返します。** 3. **入力（引数）の値の整数部が**±8388607 **を越える値を使用すると、入力値がそのまま出力されます。** 4. 7－4 **項　実数についての留意点（**7-20 **ページ）を参照してください。**
例：	> 10 A = FIX(6.1) > 20 B = FIX(6.9) > 30 C = FIX(-6.1) > 40 D = FIX(-6.9) > 50 PRINT A, B, C, D > RUN 6　　6　　-6　　-6
参照：	INT

FOR TO STEP/NEXT	
構文：	FOR <変数> = <数値式１> TO <数値式２> [STEP <数値式３>] ～NEXT <変数> {, <変数> } ｜ ： <行の終わり> ｜ <プログラム>
機能：	ステートメント。FOR から NEXT までのプログラムを指定回数だけ繰り返します。
説明：	FOR は、ループの始まりを指定し、また NEXT はループの終わりを指定します。ループは、FOR の数値式の値によって指定回数だけ実行されます。 <変数> は、FOR ループのカウンタです。 <数値式１> は、<変数> の初期値です。 <数値式 2 > は、<変数> の終値です。 <数値式３> は、<変数> の増分値です。 NEXT の後の<変数> は、FOR の後の <変数> に対応している必要があります。 <変数> は、FOR～NEXT までの文が実行されたるたびに、STEP で指定する <数値式３> 分だけ増分します。 <変数> の値が、<数値式２> の値以下の場合、FOR～NEXT までの文を再度実行します。 <変数> の値が、<数値式２> の値を超えると、FOR～NEXT までの文の実行が終了します。
参考：	注：1. STEP <数値式３> を省略した場合、STEP 増分は 1 となります。STEP 増分が 0 である場合、ループは無期限に実行され、無限ループとなります。 <数値式１> が <数値式２> より大きい場合、STEP 増分は負でなければなりません。そうでないと FOR ループは実行されず、NEXT の次の命令が実行されます。 2. FOR ループは、ネスティングが可能です。その場合、異なる変数名を使用して、複数の FOR～NEXT をネスティングします。 複数のループが同じところで終了する場合、1 つの NEXT の後に、変数名をカンマ (,)で区切って複数個指定できます。 例：10 FOR A = 1 TO 10 20 FOR B = 1 TO 10 30 PRINT A, B 40 NEXT B, A
例：	> 10 FOR A=1 TO 3 > 20 PRINT A > 30 NEXT A >RUN 1 2 3
参照：	WHILE/WEND

お願い　FOR TO STEP/NEXT では配列変数は使用できません。

FRE	
構文：	FRE
機能：	**特殊関数。ユーザメモリの未使用エリアの大きさを返します。**
説明：	**FRE は、読み出し専用の関数です。ユーザメモリの未使用エリアのサイズをバイト単位で返します。**
参考：	**出力は、使用可能な合計ユーザメモリ（プログラムエリア+変数エリアなど）です。プログラムエリアと変数エリアなどとを区別していません。**
例：	> 10 PRINT ”THE FREE USER MEMORY IS ” FRE; ” BYTES” > RUN THE FREE USER MEMORY IS 204656 BYTES
参照：	

GOSUB	
構文：	GOSUB **<行番号>** / **<ラベル>** ...RETURN
機能：	**ステートメント。行番号またはラベルで指定したサブルーチンに無条件に分岐し、そのサブルーチンを実行後 RETURN 位置から戻ります。**
説明：	**<行番号> は、サブルーチンの最初の行番号で、1〜65535 の範囲の任意の有効な整数です。 <ラベル> は、行番号の代わりに使用されます。 サブルーチンプログラムは、RETURN で終わっていることが必要です。RETURN まで達すると、プログラムの実行は、GOSUB の次の命令に戻ります。 指定された行番号、ラベルが非実行文のときは、その次の文より実行します。 サブルーチンは、他のサブルーチンからさらに呼び出すことができます（ネスティング可能です）。**
参考：	**不用意にサブルーチンプログラムを実行しないように、メインルーチンの END、STOP、または GOTO 命令の後にサブルーチンプログラムを記述することをお勧めします。 指定した行番号がプログラム内に存在しない場合、”UNDEFINED LINE NUMBER”（行番号未定義エラー, エラーコード：B008）が発生します。**
例：	> 70 INPUT C > 80 IF C=1 THEN GOSUB 1000 > 90 IF C=2 THEN GOSUB 2000 > 100 GOTO 70 > 1000 CLS > 1010 PRINT ”1 SELECTED” > 1020 RETURN > 2000 CLS > 2010 PRINT ”2 SELECTED” > 2020 RETURN > RUN >?1<CR> 1 SELECTED ?
参照：	END、GOTO、STOP

GOTO	
構文：	GOTO　<**行番号**> 　　<**ラベル**>
機能：	**ステートメント。指定される行番号またはラベルに無条件に分岐します。**
説明：	**<行番号> は、サブルーチンの最初の行番号で、**1**〜**65535 **の範囲の任意の有効な整数です。** **<ラベル> は、 行番号の代わりに使用されます。**
参考：	**指定した行番号がプログラム内に存在しない場合、**”UNDEFINED LINE NUMBER”**（行番号未定義エラー**, **エラーコード：**B008**）が発生します。**
例：	> 10 INPUT A > 20 PRINT HEX$(A) > 30 GOTO 10 >RUN ?11 B ?
参照：	GOSUB

HEX$	
構文：	HEX$(<**数値式**>)
機能：	**関数。<数値式> を 16 進数に変換します。**
説明：	**<数値式> は、-2147483648〜2147483647 の範囲であることが必要です。** **出力（戻り）の範囲は、0〜FFFFFFFF（16 進数）です。**
参考：	**単精度または倍精度実数は、変換の前に内部で INT 関数を実行したものと同じで、小数点以下を切り捨てられた整数になります。**
例：	> 10 PRINT ”DEC”, ”HEX” > 20 FOR I = 1 TO 16 > 30 PRINT I, HEX$(I) > 40 NEXT I >RUN DEC HEX 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 A 11 B 12 C 13 D 14 E 15 F 16 10 >
参照：	OCT$

IF	
構文：	IF<**数値式**> THEN <**文**>{:<**文**>} <**行番号**> <**ラベル**> GOTO <**行番号**> <**ラベル**> [ELSE <**文**>{:**文**} <**行番号**> <**ラベル**>]
機能：	**ステートメント。数値式の結果に基づいてプログラムの流れを制御します。このステートメントは、**THEN**（または** GOTO **）以降と** ELSE **以降に、文または行番号ラベルを指定し、**1 **行で記述する点が、**@IF **文と異なります。**
説明：	<**数値式**> **の結果によって** THEN**（または** GOTO**）以降と** ELSE **以降のどちらの文を実行するかを決めます** <**数値式**> **の結果が真（**0 **以外）の場合、**THEN**（または** GOTO**）以降の文が実行されます。** 0**（偽）の場合、**ELSE **以降の文が実行されます。** <**ラベル**> **は、 行番号の代わりに使用されます。**
参考：	1. ELSE **以降が省略されている場合、次の文を実行します。** 2. THEN**（または** GOTO**）以降の文、**ELSE **以降の文の代わりに、行番号又はラベルを指定することも可能です。その場合、指定行番号以降を実行します。** 3. IF **文のネスティングは１行に書ける範囲で可能です。**
例：	> 10 A = 3 > 20 IF A=2 THEN PRINT ”A IS 2” ELSE PRINT ”A IS NOT 2” > RUN A IS NOT 2
参照：	@IF

INKEY$	
構文：	INKEY$ [(# <**ポート式**>)]
機能：	**関数。指定ポートの受信バッファ内で、新たに受信された文字を返します。**
説明：	**出力（戻り）は、**[Ctrl] +[X] **キーを除く任意の文字です。指定ポートの受信バッファが空きの場合、**NULL **文字列が返ります。**
参考：	<**ポート式**> **が省略された場合、ターミナルポートが選択されます。**
例：	> 10 IF INKEY$ <> ”Q” GOTO 10 > 20 PRINT ”QUIT!” > 30 END
参照：	

INPUT	
構文：	INPUT [;] [# <ポート式> ,] [<"プロンプト"> ; | <"プロンプト"> ,] <変数> {, <変数>}
機能：	ステートメント。指定ポートの受信バッファから、CR コードを区切りとして、データを各変数へ格納します。
説明：	<ポート式> は、1〜3 の範囲内の整数を返す式です。ポート式を省略すると、ターミナルポートのポート No.（形 C200H-ASC11/21 の場合#1、形 C200H-ASC31 の場合#3）となります。 <プロンプト> は、入力を促す文字列です。ターミナル上に表示されます。プロンプトの次がセミコロン（ ; ）の場合、プロンプトに続いて（?）が表示され、プロンプトの次がカンマ（ , ）の場合、プロンプトの後に何も表示されません。指定ポートを TERM および COMU 以外のデバイスでオープンしている場合、プロンプト文や？は表示されません。（デバイスに COMU を指定しているポートで、？を表示させたくない場合、LINE INPUT 命令を使用してください。） なお、プロンプトに続いて文字列を入力するとき、文字列を引用符で囲む必要はありません。 <変数> は、任意の有効な変数名です。受信バッファからデータを CR（改行）コードを区切りとして、変数に格納します。 受信される CR 区切りのデータの数は、INPUT 命令で指定される変数の数と一致している必要があります。プロンプトに続いて入力される値の型は、INPUT 命令で指定される変数の型と一致している必要があります。 複数の変数にデータを受信する場合は、変数をカンマ（, ）で区切ってください。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. ポート#3 を形 C200H-ASC11 または形 C200H-ASC21 に指定すると、"ILLEGAL FUNCTION CALL"エラー（関数コール不正エラー, エラーコード：B005）が発生します。 3. <ポート式> が省略された場合、ターミナルポートが選択されます。 4. 入力値の型と、指定される変数の型が一致していない場合、"NO SUPPORT"（サポートなしエラー, エラーコード：0064）が発生します。 5. CR（改行）コードの入力までは、BASIC プログラムは待機し、ユニット前面の BASIC LED はゆっくりと点滅します。 6. INPUT が入力待機中に、割込みが発生した場合、割込みが禁止または停止にされていないと、割込みサブルーチンがすぐに処理されます。割込みサブルーチンが完了したら、プログラムの実行は INPUT コマンドから再開されます。 7. INPUT の後に；を付加すると、入力後その同じ行にカーソルが残ります。 8. 受信したデータのターミネータ部（末尾）が CR+LF の場合は、INPUT 文では CR までしか取り込まないため、受信バッファに LF が残ることになります。
例：	> 10 INPUT "ENTER YOUR NAME：", A$ > 20 PRINT "HELLO "; A$ > 30 END > RUN ENTER YOUR NAME：FRED <CR> HELLO FRED
参照：	LINE INPUT, WAIT

INPUT$	
構文：	INPUT$(<数値式> [, # <ポート式>])
機能：	関数。指定ポートの受信バッファ内の指定文字数の文字列を返します。
説明：	<数値式> は、受信バッファから読み出す文字数を示す整数式です。有効な範囲は、1〜255 です。 <ポート式> は、1〜3 の範囲の整数を返す式です。 [Ctrl] + [X] を除くすべての文字を、この INPUT$関数により読み出すことができます。（INPUT 命令や LINE INPUT 命令では読み込めない CR や LF コードも、そのまま読み込めます。）
参考：	1. <ポート式> が省略された場合、ターミナルポートが選択されます。 2. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 3. ポート#3 を形 C200H-ASC11 または形 C200H-ASC21 で指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。 4. INPUT$が入力待機中に割込みが発生した場合、割込みが禁止または停止にされていないと、割込みサブルーチンがすぐに処理されます。割込みサブルーチンが完了した後、プログラムの実行は INPUT$コマンドから再開されます。 5. 指定された文字数が読み出されるまで、BASIC プログラムは待機します。（待機の間、BASIC LED はゆっくり点滅します。）
例：	>10 OPEN #1, ”TERM：” >20 A$ = INPUT$(10, #1) >30 PRINT A$
参照：	WAIT

INSTR	
構文：	INSTR([<数値式>,] <文字列式 1>, <文字列式 2>)
機能：	関数。文字列の中から、指定文字列を探します。文字式 1 の文字列データに対し数値式の検索開始位置から文字式 2 の文字列を探し、その先頭文字の位置（先頭から何文字目）を返します。
説明：	<数値式> は、文字列 1 内の検索開始位置（何文字目から検索するか）です。有効範囲は 1〜255 です。 <文字列式 1> は、検索される文字列です。 <文字列式 2> は、探したい文字列です。 検索開始位置を省略すると、検索開始位置=1 として、検索される文字列の最初の文字から検索を開始します。 検索開始位置が検索文字列長より大きい場合、0 を返します。 探したい文字列が検索される文字列にない場合には、0 を返します。探したい文字列が空の文字列の場合、数値式の検索開始位置を返します。 文字式 1 が空の文字列の場合、0 を返します。
参考：	検索は大文字と小文字を識別します。
例：	> 10 A = INSTR(2, ”HELLO WORLD”, ”WORLD”) > 20 PRINT A > RUN 7
参照：	

INT	
構文：	INT(<**数値式**>)
機能：	**関数。入力（引数）として指定する数値式の小数点以下を切り捨てた整数値を返します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。出力（戻り）の型は、整数または倍長整数です。** INT **は、入力（引数）値を超えない最大の整数値を返します。**
参考：	1. **入力（引数）の値が正の場合、**INT **は** FIX **と同じ値を返します。** 2. **入力（引数）の値が負の場合、**INT **は** FIX **より** 1 **だけ小さい値を返します。例えば、**INT(-6.1)**の場合、**-7 **を返します。** 3. **入力（引数）の値の整数部が**±8388607 **を越える値を使用すると、入力値がそのまま出力されます。** 4. 7－4 **項　実数についての留意点（**7-20 **ページ）を参照してください。**
例：	> 10 A = INT(6.1) > 20 B = INT(6.9) > 30 C = INT(-6.1) > 40 D = INT(-6.9) > 50 PRINT A, B, C, D 6　　6　　-7　　-7
参照：	FIX

INTRB	
構文：	INTRB
機能：	**特殊関数。割込みが発生したときに実行中であった行番号を返します。**
説明：	INTRB **は、読み出し専用変数です。これに値を割付けることはできません。** INTRB **の値は、割込みが発生することで、命令の実行を中断した場合、その命令のあった行番号が格納されます。**
参考：	INTRB **関数は、割込みが発生して、実際に割込みサブルーチンに分岐した時点の分岐元の行番号を返します。**
例：	> 10 ON KEY 1 GOSUB 1000 > 20 KEY ON > 30 GOTO 30 > 1000 PRINT ”YOU PRESSED A KEY” > 1010 PRINT ”THE BASIC LINE NUMBER THAT WAS INTERRUPTED BY THIS R OUTINE LINE ”; INTRB > 1020 RETURN
参照：	INTRB, INTRR

INTRR	
構文：	INTRR
機能：	特殊関数。割込みサブルーチンの中で、元となった割込みのタイプを返します。
説明：	INTRR は、読出し専用特殊関数です。これに値を割り付けることはできません。 INTRR の値は、割込みサブルーチンの中だけで設定されます。割込みサブルーチン外（RETURN で復帰した後など）では、INTRR は 0 です。 INTRR の値と割込み種類の関係は、次のとおりです。

INTRR の値	割込み種類	
1, 2	通信割込み	COM1、COM2
3, 4	-	未使用
5	ON アラーム割込み	ALARM
6	タイマ割込み	TIMER
7	時刻割込み	TIME$
10-19	キー割込み	KEY(0～9)
20-100	-	未使用
101-199	エラー割込み	ERROR(1～99)
200	-	未使用
201-299	PC 割込み	PC(1～99)

例：	> 10 ON KEY 1 GOSUB 1000 > 20 KEY ON > 30 GOTO 30 > 1000 PRINT ”YOU PRESSED A KEY” > 1010 PRINT ”THE SOURCE OF THIS INTERRUPT IS TYPE ”; INTRR > 1020 RETURN
参照：	INTRB, INTRL

INTRS	
構文：	INTRS
機能：	特殊関数。割込みサブルーチンに飛んだ割込みを定義した行番号を返します。
説明：	INTRS は、読み出し専用変数です。これに値を割付けることはできません。 BASIC プログラムの初期化時に、INTRS の値は、0 に設定されます。 割込みサブルーチンに飛んだ割込みを定義した行番号の値が、割込みルーチン内で、INTRS に格納されます。
参考：	INTRS の値は、割込みサブルーチンに飛んだ時点で、設定され、RETURN で復帰した時点で、元の値に戻ります。
例：	> 10 ON KEY 1 GOSUB 1000 > 20 KEY ON > 30 GOTO 30 > 1000 PRINT ”YOU PRESSED A KEY” > 1010 PRINT ”THIS ISR WAS SET-UP INLINE ”; INTRSB; ”OF THE BASIC PROGRAM” > 1020 RETURN
参照：:	INTRL, INTRR

KEY ON/OFF/STOP	
構文：	KEY [<**キー式**>] | ON OFF STOP |
機能：	**ステートメント。**ON KEY **命令によるキー割込みを許可、禁止または停止します。**
説明：	<**キー式**> **は、**0～9 **の値を返す整数式です。この数字は、ターミナルのキーボードの数字キーに対応します。例えば、**KEY 5 ON **の場合、キーボードの「**5**」のキーの入力時の割込みが有効になります。**<**キー式**> **を省略すると、割込みコントロールはすべてのキーに対して有効となります。** ON **は、キー割込みを許可（マスク解除）にします。この** KEY ON **命令から** KEY OFF **命令までを実行中に、キー割込みが発生すると、定義された 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降のすべてのキー割込みは、無視されます。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくだけで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンを実行します。**
参考：	KEY ON/OFF/STOP **命令は、**ON KEY **命令を実行した後にのみ実行可能です。**
例：	> 10 ON KEY 1 GOSUB 1000 > 20 KEY ON > 30 GOTO 30 > 1000 PRINT ”YOU PRESSED A KEY” > 1010 RETURN
参照：	ON KEY GOSUB/GOTO

LEFT$	
構文：	LEFT$(<**文字列式**>, <**数値式**>)
機能：	**関数。文字列の左端から指定された文字数の文字列を取り出します。**
説明：	<**文字列式**> **は、取り出し対象の文字列です。** <**数値式**> **は、取り出す文字数です。有効範囲は** 0～255**。** **取り出す文字数が** 0 **の場合、空白文字列が返されます。** **取り出す文字数が取り出し対象の文字列の文字数より大きい場合、取り出し対象の文字列全体が返されます。**
例：	> 10 A$ = LEFT$(”HELLO WORLD”, 5) > 20 PRINT A$ > RUN HELLO
参照：	MID$, RIGHT$

LEN	
構文：	LEN(<**文字列式**>)
機能：	**関数。文字列式の文字数を返します。**
参考：	カウントには出力されない文字と空白が含まれます。
例：	> 10 A = LEN(”HELLO WORLD AGAIN”) > 20 PRINT A > RUN 17
参照：	

LET	
構文：	[LET] <**変数**> ＝ <**式**>
機能：	**ステートメント。ステートメントの右側の** <式> **の数値を式の左側の変数に代入します。**
説明：	**キーワード：**LET **は、任意の代入文です。** **代入については、**”=”**を使用します。** ”LET”**は省略できます。**
参考：	**文字式を数値変数に代入するか、またはその逆を行うと、**”TYPE MISMATCH”**（型不一致エラー，エラーコード：**B013**）が発生します。代入の右側の数値式が左側の変数に対して異なる型であると、右側の式は、代入の前に左側の変数の型に変換されます。**
例：	> 10 LET A = 12.34 > 20 PRINT A > RUN 12.34 >
参照：	

LINE INPUT	
構文：	LINE INPUT [;] [# <**ポート式**>,] [<”**プロンプト文** ”> ; ,] <**文字変数**>
機能：	**ステートメント。指定ポートの受信バッファから、**CR **コードを区切りとして、データを** 1 **つの変数へ格納します。ターミナルからの** 1 **行（**CR **まで）のデータを、**1 **つに変数に格納するときなどに、使用します。**
説明：	<**ポート式**> **は、**1～3 **の範囲の整数を返す式です。**<**ポート式**> **を省略すると、デフォルトでターミナルポートと見なします。** <**プロンプト文**> **は、入力を促す文字列です。ターミナル上に表示されます。指定ポートを** TERM**、**COMU **以外のデバイスでオープンしている場合、プロンプトは表示されません。**INPUT **命令と異なり、**?**は表示されません。** **受信バッファからデータを** CR**（改行）コードを区切りとして、変数に格納します。** **文字列の最大長は、**255 **文字です。**CR**（改行）コードの入力までは、**BASIC **プログラムは待機し、ユニット前面の** BASIC LED **はゆっくりと点滅します。** **カンマ（**, **）とコロン（**: **）も、受信データの一部として取り扱われます。**
参考：	LINE INPUT **の後に；を付加すると、入力後その同じ行にカーソルが残ります。**
例：	> 10 OPEN #1,”TERM:” > 20 LINE INPUT #1, ”ENTER YOUR FULL NAME : ”; A$ > 30 PRINT ”HELLO ”;A$ > 40 END > RUN ENTER YOUR FULL NAME : JOE BLOGGS<CR> HELLO JOE BLOGGS >
参照：	INPUT

LIST	
構文：	LIST [<行番号> | •] [- [<行番号> | •]]
機能：	コマンド。現在の ASCII ユニット内の BASIC プログラムを、ターミナル画面にリスト表示します。
説明：	<行番号> は、任意の BASIC プログラム行番号です。有効範囲は、1〜65535 です。 <行番号> とダッシュ（-）を組み合わせることにより、リスト表示したい行範囲を指定します。 1. 単一の行番号を指定すると、その行だけが表示されます。 2. 行番号-行番号を指定すると、その行範囲全体が表示されます。 3. 行番号-を指定すると、指定した行番号からプログラムの終わりまでのすべてのプログラム行を表示します。 4. -行番号を指定すると、プログラムの始まりから指定した行番号までのすべてのプログラム行を表示します。 5. ピリオド（.）だけを指定すると、現在の行番号（最も新しく表示した行、または最も新しく入力した行）をリスト表示します。 リスト表示が画面に収まらない場合、リスト表示がスクロールされ、最後の複数行を表示します。
参考：	LIST コマンドによる出力は、[Ctrl]+[X]または[Ctrl]+[B]キーのいずれかにより、中断されます。[Ctrl]+[B]キーを使用する場合、[Ctrl]+[B]キーを再度押すと、出力を再開できます。 LIST と LLIST の違いは、次のとおりです。 ・LIST は、形 C200H-ASC11 と形 C200H-ASC21 のポート#1 および形 C200H-ASC31 のポート#3 に出力します。 ・LLIST は、ポート#2 に出力します。 ポートが OPEN 命令によって送信可能デバイスに設定されないと、”CANNOT WRITE TO READ-ONLY DEVICE”（読み取り専用デバイスへ書き込み不可エラー, エラーコード：0059）が発生します。 注：出力ポートは、形 C200H-ASC11 と形 C200H-ASC21 ではポート#1 であり、 形 C200H-ASC31 ではポート#3 です。
例：	>LIST 100-200 100 REM THIS IS THE BEGINNING OF A SECTION TO BE LISED 110 PRINT ”HELLO WORLD” 200 REM THIS IS THE END OF A SECTION TO BE LISTED >
参照：	

LLIST	
構文：	LLIST [<行番号> または .] [- [<行番号> または .]]
機能：	コマンド。現在の ASCII ユニット内の BASIC プログラムをプリンタに印字します。
説明：	<行番号> は、任意の BASIC プログラム行番号です。有効範囲は、1〜65535 です。 <行番号> とダッシュ（-）を組み合わせることにより、リスト表示したい行範囲を指定します。 1. 単一の行番号を指定すると、その行だけがリスト表示されます。 2. 行範囲-行番号を指定すると、その行範囲全体がリスト表示されます。 3. 行番号-を指定すると、指定した行番号からプログラムの終わりまでのすべてのプログラム行をリスト表示します。 4. -行番号を指定すると、プログラムの始まりから指定した行番号までのすべてのプログラム行をリスト表示します。 5. ピリオド（.）だけを指定すると、現在の行番号（最も新しく表示した行、または最も新しく入力した行）をリスト表示しす。
参考：	LIST コマンドによる出力は、[Ctrl]+[X]または[Ctrl]+[B]キーのいずれかにより、中断されます。[Ctrl]+[B]キーを使用する場合、[Ctrl]+[B]キーを再度押すと、出力を再開できます。 LLIST は、OPEN 命令の指定に関係なく、常に"CS_ON, RS_OFF, DS_ON"の条件で動作します。 ポートが OPEN 命令によって送信可能デバイスに設定されないと、"CANNOT WRITE TO READ-ONLY DEVICE"（読み取り専用デバイスへ書き込み不可エラー, エラーコード：0059）が発生します。 注：出力ポートは、形 C200H-ASC11、形 C200H-ASC21 および形 C200H-ASC31 でポート#2 になります。
例：	> LLIST 100-200 >
参照：	LIST

LOAD	
構文：	LOAD# <ポート式> [, " <デバイス記号>：[<通信条件> [, <データ長> [, <パリティ> [, <ストップビット> [, CS [, RS [, DS [, XN]]]]]]]] "]
機能：	コマンド。ASCII ユニットの ユーザメモリ（RAM）エリアへ、ターミナルから BASIC プログラムを転送します。
説明：	<ポート式> は、1〜3 の範囲内の整数を返す式です。 [, " <デバイス記号>： [<通信条件> [, <データ長> [, <パリティ> [, <ストップビット> [, CS [, RS [, DS [, XN]]]]]]] "]の説明については、OPE N命令を参照してください。 LOAD 命令実行中は、LOAD 命令で指定した条件（デバイス記号・通信条件・伝送制御信号の動作）が有効になります。LOAD 命令実行後、もとの指定にもどります。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. 形 C200H-ASC11 または形 C200H-ASC21 でポート#3 を指定すると、"ILLEGAL FUNCTION CALL"（関数コール不正エラー, エラーコード：B005）が発生します。 3. スタート/ストップスイッチをスタートからストップに切り替えることで、LOAD 動作中に転送を中断することができます。 本手順の詳細と ASCII ユニットに BASIC プログラムを転送（ロード）するための手順については、7-1 項「プログラミングの手順」を参照してください。
例：	> NEW > LOAD #1, "COMU:9600,8,N,2" > LIST 10 REM THIS IS THE SAMPLE PROGRAM THAT WAS LOADED 20 PRINT "HELLO WORLD" >
参照：	OPEN, SAVE

LOC	
構文：	LOC(<ポート式>)
機能：	関数。指定ポートの受信バッファ内にたまっているデータのバイト数を返します。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。
参考：	1. 指定ポートは、OPEN 命令でオープンされ、そのデバイス記号指定で、受信用に設定されている必要があります。 2. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 3. 形 C200H-ASC11 または形 C200H-ASC21 でポート#3 を指定すると、"ILLEGAL FUNCTION CALL"（関数コール不正エラー, エラーコード：B005）が発生します。
例：	> 10 OPEN #2, "COMU:" > 20 A = LOC(2) > 30 PRINT A
参照：	

LOG	
構文：	LOG(<**数値式**>)
機能：	**関数。数値式の** e **を底とする対数を計算します。**
説明：	<**数値式**> **は、**0 **より大きいことが必要です。** **入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。入力（引数）の型が、倍精度実数型の場合、出力（戻り）の型は倍精度実数型になります。**
例：	> 10 A = 0.4 > 20 PRINT LOG(A) > RUN -.916291
参照：	

LPRINT

構文：

LPRINT [# <ポート式>] [USING "<フォーマット文字列>" ;] <式> { [, / ; / <空白>] <式> }

機能：

ステートメント。指定ポートにデータを送信します。OPEN 命令に関係なく、デバイス記号が LPRT となります。

説明：

<ポート式> は、1〜2 の範囲の整数を返す式です。
ポート式を省略すると、デフォルトで#2となります。

<式> は、任意の有効な数値または文字式です。
<式> がセミコロン（;）または空白で区切られ、かつ、フォーマット文字列を使用しない場合、式（データ）は、送信バッファから連続で送信されます。<式> がカンマ（, ）で区切られた場合、式（データ）は、タブ文字で区切って送信されます。
終端のカンマ（, ）またはセミコロン（;）を省略すると、式（データ）の終わり CR（改行）が付加されます。
数値式の場合、式（データ）の前後に空白が挿入されて送信されます。数値式の前の空白は、マイナス符号用になっています。
式を指定しない場合、この命令は、送信バッファに、CR（改行）のみを送信します。
オプションの <フォーマット文字列> は、USING の後に、次のように指定します。
複数の <式> で指定する場合は、それぞれの <式> に対応するフォーマットをスペースで区切ってください。スペースがない場合は、正しくフォーマットに従って出力されないことがあります。

文字列	!	文字列の最初の 1 文字だけを送信します。
	& &	&と&に囲まれた文字数分を送信します。
	@	対応する文字列をそのまま出力します。
数値	#	数字の桁（負符号も含む）を示します。
	.	任意の位置に小数点 . を挿入します。
	+	数値（#）の直前または直後に、符号（+）を付けます。
	-	数値（#）の終わりに指定すると、数値（#）を負数の符号にすることを示します。値が正の場合、-を数値（#）の終わりに指定すると、*が上位空白桁に入ります。
	**	数値（#）の上位空白桁に*を付けます。
	¥¥	数値（#）の始まりに¥を接頭辞としてつけます。¥は 1 桁用の空白を割付けます。
	**¥	上記**と¥の組み合わせです。
	,	整数部を右側より 3 桁ごとにコンマで区切ります。
	^^^^	数値（#）の終わりに指定すると、指数形式であることを示します。　指数形式（E+nn,D+nn）で出力します。
	^^^^^	数値（＃）の終わりに指定すると、指定形式であることを示します。　指定形式（E+nnn, D＋nnn）で出力します。
	__	__は、__に続く 1 文字をそのまま送信します。特殊文字の前に"__"を置くと、文字列としてその特殊文字を出力できます。

次ページへ続く

前ページより

参考：	1. **ポートは、送信可能デバイスとして、**OPEN **命令によって開いておく必要があります。** 2. **フォーマット文字列（数値送信）で指定された桁数より、出力する式（データ）の桁数が大きい場合、**"%"**が数値の前に出力されます。** 3. **指定ポートを送信可能なデバイスとしてオープンしてしない場合、**"CANNOT WRITE TO READ-ONLY DEVICE"**（読み取り専用デバイスへ書き込み不可エラー，エラーコード：**0059**）が発生します。** 4. LPRINT **は、**OPEN **命令での指定に関係なく、常に**"CS_ON**、**RS_OFF**、**DS_ON"**の条件で実行されます。** 5. **式が** <**フォーマット文字列**> **に対応していない場合、**"TYPE MISMATCH"**（型不一致エラー，エラーコード：**B013**）が発生します。**
	> 10 LINE INPUT "ENTER A STRING : "; A$ > 20 OPEN #2, "LPRT: " > 30 LPRINT #2, A$ > 40 PRINT "THE STRING YOU ENTERED - ";A$; "- WAS SENT TO THE PRINTER" > RUN ENTER A STRING : TEST THE STRING YOU ENTERED - TEST - WAS TO THE PRINTER
参照：	PRINT, WAIT

MID$	
構文：	MID$(<**文字列式**>, <**数値式１**> [, <**数値式** 2>])
機能：	**関数。文字列の指定位置から指定文字数の文字列を取り出します。**
説明：	<**文字列式**> **は、取り出し対象の文字列です。** <**数値式** 1> **は、取り出される文字列内の開始位置です。有効範囲は** 1～255**。** <**数値式２**> **は、取り出す文字数です。有効範囲は** 0～255**。** **取り出す文字数が** 0 **の場合、または開始位置が検索文字長より大きい場合、空白文字が返されます。取り出す文字数が、取り出される文字列の残り文字数より大きい場合、開始位置からの取り出し残りの文字列全体が返されます。** <**数値式** 2> **を省略するか、または** <**数値式** 2> **が開始位置から右端までの文字数を超えると、開始位置から右にある文字がすべて、返されます。**
例：	> 10 A$ = "HELLO WORLD AGAIN" > 20 B$ = MID$(A$,7,5) > 30 PRINT B$ > RUN WORLD
参照：	LEFT$, RIGHT$

MID$	
構文：	MID$(<**文字列式** 1>, <**数値式** 1> [, <**数値式** 2>])=<**文字列式** 2>
機能：	**ステートメント。文字列の一部を置き換えます。文字列** 1 **の指定位置から指定文字数分の文字列を、文字列** 2 **の先頭から同じ指定文字数分の文字列で置き換えます。**
説明：	<**文字列式** 1> **は、置き換えられる文字列です 。** <**文字列式** 2> **は、置き換える文字列です** <**数値式** 1> **は、**<**文字列** 1> **の 置き換えられる文字の開始位置です。有効範囲：**1～255**。** <**数値式** 2> **は、**<**文字列** 2> **によって置き換える 文字数です。有効範囲：**0～255**。これを省略すると、**<**文字列** 1> **の開始位置からの文字列を、**<**文字列** 2> **の先頭からのすべての文字列で置き換えます。ただし、**<**数値式** 2> **を省略したときも省略しないときも、置き換え数は、**<**文字列** 1> **の開始位置から最終までの文字数以下です。** <**数値式** 1> **は**<**文字列** 1> **の文字数を超える文字位置は指定できません。**
例：	> 10 A$ = ”HELLO WORLD” > 20 MID$(A$, 7, 5) = ”AGAIN” > 30 PRINT A$ RUN HELLO AGAIN
参照：	LEFT$, RIGHT$, MID$**（関数**)

MODEL	
構文：	MODEL
機能：	**特殊関数。**ASCII **ユニットの形式の** C200H-**以下を示す番号を返します。**
説明：	**現在の** ASCII **ユニットの形式** C200H-**以下を返します。** 1：ASC11 2：ASC21 3：ASC31
例：	> 10 IF MODEL=1 THEN PRINT ”THIS UNIT IS AN ASC11” : GOTO 50 > 20 IF MODEL=2 THEN PRINT ”THIS UNIT IS AN ASC21” : GOTO 50 > 30 IF MODEL=3 THEN PRINT ”THIS UNIT IS AN ASC31” : GOTO 50 > 40 PRINT ”UNKNOWN MODEL OF ASCII UNIT ” > 50 END > RUN THIS UNIT IS AN ASC21
参照：	

NEW	
構文：	NEW
機能：	コマンド。ASCII ユニットのユーザメモリエリア内の現在利用している BASIC プログラム、すべての変数をクリアします。
説明：	このコマンドは、新しいプログラムを作成する前に現在利用しているユーザメモリをクリアするために使用します。 現在の BASIC プログラムが削除され、すべての変数はクリアされます。 開いているポートは閉じられ、CLOSE 命令と同様に、送信バッファに残っていたデータは、掃き出されてからクローズされ、受信バッファに残っていたデータは、クリアされてからクローズされます。
参考：	プログラムが PNAME 命令により保護されている場合は、NEW によりプログラムを削除できません。その場合、プログラム名を、PNAME” ”によりクリアする必要があります。PNAME 命令により保護されているプログラムをクリアしようとすると、”PROTECTED PROGRAM”（プログラム保護エラー，エラーコード：0062）が発生します。 ユーザメモリエリアが PMEM 命令により書き込み禁止となっている場合は、NEW によりユーザメモリエリアをクリアできません。その場合、書き込み禁止を PMEM OFF 命令により解除する必要があります。 ただし、プログラムが PWORD 命令によりパスワード保護されている場合は、NEW コマンドにより、プログラムは削除されます。
例：	> 10 REM DUMMY PROGRAM > 20 A = 5 > LIST 10 REM DUMMY PROGRAM 20 A = 5 > NEW > LIST >
参照：	PINF, PNAME, PMEM, PWORD

OCT$	
構文：	OCT$(<**数値式**>)
機能：	関数。〈数値式〉を 8 進数に変換します。
説明：	**数値式は、**-2147483648**～**2147483647 **の範囲であることが必要です。** **出力（戻り）の範囲は、**0**～**7777777777（8 **進数）です。**
参考：	**単精度または倍精度実数式は、変換の前に、小数点以下を切り捨てられ整数になります。**
例：例:	> 10 PRINT ”DEC”, ”OCTAL” > 20 FOR I = 1 TO 20 > 30 PRINT I, OCT$(I) > 40 NEXT I > RUN DEC　　OCTAL 1　　1 2　　2 3　　3 4　　4 5　　5 6　　6 7　　7 8　　10 9　　11 10　　12 11　　13 12　　14 13　　15 14　　16 15　　17 16　　20 17　　21 18　　22 19　　23 20　　24 >
参照：:	HEX$

ON ALARM	
構文：	ON ALARM <**時間式**> GOSUB | **<行番号>** | **<ラベル>** |
機能：	ステートメント。指定時間経過による割込みを行います。本命令実行から指定時間の後、GOSUB 先の行番号の割込みサブルーチンに分岐します。その割込み条件と割込み先を定義します。
説明：	<**時間式**> **は** 1〜864000 **を返す整数式です。** **時間式で指定する単位は、**0.1 **秒です。したがって、整数式の範囲は、**0.1 **秒** 〜24 **時間の範囲です。** <**行番号**> **は、割込みサブルーチンの開始位置を指定します。** <**ラベル**> **は、**GOSUB **命令の中で、行番号の代わりに使用されます。**
参考：	ALARM ON 命令によって、割込み許可となります。 ALARM OFF 命令によって、割込み禁止となります。 ALARM STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。 **タイマがアップした時に、実行中であった区間が、**ALARM ON **命令によってこの割込みが許可（マスク解除）されている場合、定義された割込みサブルーチンへ分岐します。** **割込みサブルーチンは、**RETURN **命令により終了してください。**RETURN **命令により、割込みで分岐時の命令に戻ります。** ON **アラーム割込みは、減算式タイマの割込みです。**ON ALARM **命令実行後、指定した設定時間から減算を開始します。**0 **に達したときに、定義された割込みサブルーチンへ分岐します。一度、**ON **アラーム割込みを実行（分岐処理）すると、**ON **アラーム割込みは無効になります。以降** ON **アラーム割込みは、** ON ALARM **命令により再度指定した設定時間から減算を開始します。つまり、この** ON **アラーム割込みは、ワンショットタイマ割込みです。この点が、インターバルタイマ割込みである、タイマ割込みとは異なります。** ON **アラーム割込みは、**1 **つのプログラム内で** 1 **つしか有効ではありません。複数の** ON ALARM **命令を１つのプログラム内に設定すると、最後の** ON ALARM **命令の割込みが有効になります。** ON **アラーム割込みのサブルーチンを実行中に、他の割込みがはいっても、その割込みはすぐに実行されず、**ON **アラーム割込みのサブルーチン実行終了後、割込み優先順位にしたがって実行されます。** **注：割込みを作成する場合の詳細については、**5-1 **項「各種の処理」の「■割込機能」を参照してください。**
例：	> 10 A=0 > 20 ON ALARM 30 GOSUB 1000 > 30 ALARM ON > 40 IF A=0 THEN GOTO 40 > 50 PRINT ”END” > 60 END > 1000 PRINT ”INTERRUPT OCCURRED” > 1010 A = 1 > 1020 RETURN > RUN INTERRUPT OCCURRED END
参照：	ALARM(ON|OFF|STOP), ON TIMER

ON COM	
構文：	ON COM [<ポート式>] GOSUB <行番号> | <ラベル> [, CODE [=<コード式>] | , BYTE=<バイト式> | , HEAD=<コード式>, TERM [=<コード式>] | , HEAD=<コード式>, BYTE=<バイト式>]
機能：	ステートメント。受信バッファに指定した形式でデータが受信されると、割込みサブルーチンが実行されます。その割込み条件と割込み先を定義します。 指定形式は、次の 5 種類が可能です。 1）何らかのデータを受信した場合、2）指定した 1 文字を受信した場合、3）指定バイト数を受信した場合、4）指定先頭文字の後、指定最終文字を受信した場合、5）指定先頭文字の後、指定バイト数を受信した場合
説明：	<ポート式> は、1〜2 の範囲の整数を返す式です。<ポート式> を省略すると、ON COM はすべてのポートで動作します。 <行番号> は、割込みサブルーチンの開始位置を指定します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。 <コード式> は、1 文字の ASCII コードを 10 進に換算して入力します。有効範囲は、0〜255 です。 <バイト式> は、バイト数を指定します。有効範囲は、1〜255 です。 ON COM 割込みは、次の 5 つのいずれかのフォームで使用できます。 フォーム(1)：ON COM[<ポート式>]GOSUB xxxx フォーム(2)：ON COM[<ポート式>]GOSUB xxxx, CODE=xx フォーム(3)：ON COM[<ポート式>]GOSUB xxxx, BYTE=yy フォーム(4)：ON COM[<ポート式>]GOSUB xxxx, HEAD=xx, TERM= yy フォーム(5)：ON COM[<ポート式>]GOSUB xxxx, HEAD=xx, BYTE=yy フォーム(1)： 受信バッファに、何らかのデータを受信すると、割込みが発生します。 フォーム(2)： 受信バッファに、CODE で指定した xx を ASCII コードとする文字を受信すると、割込みが発生します。xx を省略すると、CODE の文字列は CR になります。xx の有効範囲は、0〜255（ASCII コードの 10 進換算）です。 フォーム(3)： BYTE で指定した文字数 yy バイトのデータを受信すると、割込みが発生します。yy の有効範囲は、1〜255 です。 フォーム(4)： 受信バッファに、HEAD で指定した xx を ASCII コードとする文字を受信した後、TERM で指定した yy を ASCII コードとする文字を受信すると、割込みが発生します。xx、yy の有効範囲は、0〜255（ASCII コードの 10 進換算）です。yy を省略すると、TERM の文字列は CR になります。xx から yy までのすべての文字は、受信バッファに残ります。 フォーム(5)： 受信バッファに、HEAD で指定した xx を ASCII コードとする文字を受信した後、BYTE で指定した文字数 yy バイトのデータを受信すると、割込みが発生します。xx の有効範囲は、0〜255（ASCII コードの 10 進換算）です。yy の有効範囲は、1〜255 です。xx から yy バイトのすべての文字は、受信バッファに残ります。

次ページへ続く

前ページより

参考：	割込みサブルーチンは、RETURN 命令により終了してください。RETURN 命令により、割込みで分岐時の命令に戻ります。 割込みサブルーチン内で、受信バッファの内容をチェックした後、受信バッファ内のデータを処理してください。通信割込み（ON COM）が保留状態の間に、バッファデータが更新された可能性があります。 COM ON 命令によって、割込み許可となります。COM OFF 命令によって、割込み禁止となります。COM STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。通信割込みが入った時に、実行中であった区間が、COM ON 命令によってこの割込みが許可（マスク解除）されている場合、定義された割込みサブルーチンへ分岐します。 通信割込みのサブルーチンを実行中に、他の割込みがはいっても、その割込みはすぐに実行されず、通信割込みのサブルーチン実行終了後、割込み優先順位にしたがって実行されます。 BASIC で割込みを作成する場合の詳細については、5-1 項「各種の処理」の「■割込機能の ON COM と ON PC 例」を参照してください。 注 1： ポートは入力のために開いていなければなりません。 注 2： 通信条件は、通信速度が ポート 1＋ポート 2≦38.4k ビット/s となるようにしてください。
例：	> 10 ON COM 2 GOSUB 1000 > 20 COM ON > 30 GOTO 30 > 1000 PRINT ”COMMUNICATIONS INTERRUPT OCCURRED” > 1010 A$ = INPUT$(LOC(2), #2) > 1020 PRINT A$; ”RECEIVED” > 1030 RETURN
参照：	COM(ON|OFF|STOP)

ON ERROR	
構文：	ON ERROR GOTO ｜〈行番号〉 〈ラベル〉｜
機能：	**ステートメント。エラーが発生すると、**GOTO **先の行番号の割込みサブルーチンに分岐します。その割込み先を定義します。**
説明：	**<行番号> は、**0～65535 **の範囲の、任意の行番号です。これは、エラー割込みサブルーチンの最初の行を示します。**ON ERROR GOTO 0 **は、エラー割込みを無効にします（エラーが発生しても、何も処理をしません）。** **<ラベル> は、**GOTO **命令の中で、行番号の代わりに使用されます。**
参考：	**エラーが発生すると、エラーコードが、**ERR **関数に格納され、エラーを起こした行番号が、**ERL **関数に格納されます。これらの関数の内容をチェックすることで、エラー処理を行うことができます。** **エラーコード（**80～99**）のエラーでは** ON ERROR **命令により、割込みサブルーチンへ分岐させることはできません。** **エラー割込みサブルーチン実行中にエラーが発生した場合、そのエラーはエラー割込みの対象になりません。ただし、ターミナルにはエラーメッセージが表示され、**BASIC **プログラムの実行は終了します。** BASIC **で割込みを作成する場合の詳細については、**5-1 **項「各種の処理」の「■割込機能の** ON COM **と** ON PC **例」を参照してください。命令実行中は割込み待機となります。**
例：	> 10 ON ERROR GOTO 1000 > 20 ERROR 2 > 30 END > 1000 PRINT ”A SYNTAX ERROR HAS OCCURRED IN LINE ”;ERL > 1010 END > RUN A SYNTAX ERROR HAS OCCURRED IN LINE 20
参照：	ERC, ERL, ERR, ERROR, RESUME

ON <数値式>	
構文：	ON <数値式> GOTO | GOSUB <行番号> | <ラベル> { , <行番号> | , <ラベル> }
機能：	ステートメント。数値式の値に応じて、指定された行番号またはラベルに分岐します。
説明：	<数値式> は、1～255 を返す整数式です。 <行番号> は、サブルーチンの開始位置を指定します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。 この数値式によって、どのサブルーチンを実行するかを指定します。 ON<数値式>（GOTO｜GOSUB）行番号 1, 行番号 2･･･の場合、数値式の値が 1 の場合、行番号 1 へ分岐が行われ、数値式の値が 2 の場合、行番号 2 へ分岐が行われ、以下同様に、数値式の値が+1 される毎に右側の行番号が割付られます。 0 を指定するか、または行番号の個数が数値式で指定した数を超える場合、GOTO または GOSUB 命令は無視して、次の命令を実行します。
参考：	1．GOSUB 文を分岐に使用する場合、サブルーチンは、RETURN 命令により終了してください。RETURN 命令により、ON 数値式命令の次の命令にジャンプします。 2．式の値がこの有効範囲外であると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	> 10 FOR I = 1 TO 3 > 20 ON I GOSUB 100,200,300 > 30 NEXT I > 40 END > 100 PRINT ”1” > 110 RETURN > 200 PRINT ”2” > 210 RETURN > 300 PRINT ”3” > 310 RETURN > RUN 1 2 3
参照：	

ON KEY	
構文：	ON KEY [<キー式>] GOTO | GOSUB <行番号> | <ラベル>
機能：	ステートメント。ターミナルのキーボードから指定したキー（0〜9 のテンキー）を受信すると、割込みサブルーチンが実行されます。その割込み条件と割込み先を定義します。（ターミナル接続時のみ有効。）
説明：	<キー式> は、0〜9 の範囲の整数を返す整数式です。<キー式> ごとに ON KEY GOTO/GOSUB 命令を 1 つずつ対応させてください。<キー式> を省略すると、割込みがすべてのキー入力に対して有効になります。 <行番号> は、割込みサブルーチンの開始位置を指定します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。
参考：	KEY ON 命令によって、割込み許可となります。KEY OFF 命令によって、割込み禁止となります。KEY STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。キー割込みが入った時に、実行中であった区間が、 KEY ON 命令によってこの割込みが許可（マスク解除）状態だった場合、定義された割込みサブルーチンへ分岐します。 割込みサブルーチンは、RETURN 命令により終了してください。RETURN 命令により、割込みで分岐時の命令に戻ります。 キーボードを接続するポートは、最初に OPEN 命令を使ってデバイス記号 TERM で開く必要があります。 ON KEY 割込みのサブルーチンを実行中に、他の割込みがはいっても、その割込みはすぐに実行されず、 ON KEY 割込みのサブルーチン実行終了後、割込み優先順位にしたがって実行されます。 BASIC での割込みを作成する場合の詳細については、5-1「各種の処理」の「■割込機能の ON COM と ON PC 例」を参照してください。 この範囲外の値は、”FORMAT ERROR”（フォーマットエラー, エラーコード：B067）になります。 ON KEY GOTO 文の場合は、PC READ/PC WRITE（@付き含む）命令実行中は割込み待機中となります。
例：	> 10 ON KEY 1 GOSUB 60 > 20 ON KEY 2 GOSUB 80 > 30 ON KEY 3 GOSUB 100 > 40 KEY ON > 50 GOTO 50 > 60 PRINT ”1 PRESSED” > 70 RETURN > 80 PRINT ”2 PRESSED” > 90 RETURN > 100 PRINT ”3 PRESSED” > 110 RETURN > RUN
参照：	KEY(ON|OFF|STOP)

ON PC	
構文：	ON PC [<割込み番号>] GOSUB <行番号> <ラベル>
機能：	ステートメント。PC（CPU ユニット）からの割付 I/O または IOWR（#CC00）命令による割込みにより、指定 割込み番号の割込みサブルーチンを実行します。その割込み条件と割込み先を定義します。
説明：	<割込み番号> は、1～99 の範囲の値の整数式です。 <割込み番号> を省略すると、どの割込み番号が指定されても、この ON PC 命令による割込みサブルーチンへ分岐します。 割付 I/O の n+2CH での割込み番号のデフォルト値は 0 です。つまり、PC 割込みは ASCII ユニットに対して要求されません。 <行番号> は、割込みサブルーチンの最初の行を示します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。
参考：	PC ON 命令によって、割込み許可となります。PC OFF 命令によって、割込み禁止となります。PC STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。 PC 割込みが入った時に、実行中であった区間が、PC ON 命令によってこの割込みが許可（マスク解除）状態だった場合、定義された割込みサブルーチンへ分岐します。 割込みサブルーチンは、RETURN 命令により終了してください。RETURN 命令により、割込みで分岐時の命令に戻ります。 PC 割込みは、次のいずれかの方法で、割込みをかけることができます。 1. 割付 I/O （PC 割込みフラグ）による割込み CPU ユニットが、割付 I/O の n+2CH ビット 00～07 によって、PC 割込み番号をセット。同時に、割付 I/O の nCH ビット 04 によって、PC 割込みフラグを 0→1 に立ち上げることで、（PC 割込みが許可されていれば）指定 PC 割込み番号の PC 割込みが実行されます。 2. IOWR（#CC00）命令による割込み CPU ユニットが、IOWR（#CC00）命令によって、ASCII ユニットに対して、割込み番号を指示すると、（PC 割込みが許可されていれば）指定 PC 割込み番号の PC 割込みが実行されます。 PC 割込みは、ASCII ビジーフラグが 1（ビジー中）のときは実行されません。割込みをかけるときは、ASCII ビジーフラグをチェックしてください。 PC 割込みのサブルーチンを実行中に、他の割込みがはいっても、その割込みはすぐに実行されず、PC 割込みのサブルーチン実行終了後、割込み優先順位にしたがって実行されます。 PC 割込みの詳細については、6-3 項「データ交換方法の詳細」を参照してください。 BASIC で割込みを作成する場合の詳細については、5-1 項「各種の処理」の「■割込機能の ON COM と ON PC 例」を参照してください。

次ページへ続く

前ページより

例：	> 10 ON PC 1 GOSUB 70 > 20 ON PC 2 GOSUB 90 > 30 ON PC 3 GOSUB 110 > 50 PC ON > 60 GOTO 60 > 70 PRINT ”PC INTERRUPT #1” > 80 RETURN > 90 PRINT ”PC INTERRUPT #2” > 100 RETURN > 110 PRINT ”PC INTERRUPT #3” > 120 RETURN > 130 END
参照：	PC(ON｜OFF｜STOP)

ON TIME$	
構文：	ON TIME$= <時刻文字列式> GOSUB <行番号> <ラベル>
機能：	ステートメント。ある時刻になると、割込みサブルーチンを実行します。その割込み条件と割込み先を定義します。
説明：	<時刻文字列式> は、「hh:mm:ss」,の絶対時間です。hh は時（00～23 の範囲）、mm は分（00～59 の範囲）、ss は秒（00～59 の範囲）です。 ワイルドカード*を使用して複数の時刻を一括で指定することもできます。 例えば、”02:30:**”は、2 時 30 分 00 秒から、1 秒間隔で 60 回の割込みが発生することを意味します。 ”02:3*:00”は、2 時 30 分から、1 分の間隔で 10 回の割込みが発生することを意味します。 <行番号> は、割込みサブルーチンの開始位置を指定します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。
参考：	TIME$ ON 命令によって、割込み許可となります。TIME$ OFF 命令によって、割込み禁止となります。TIME$ STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。 指定時刻になった時に、実行中であった区間が、TIME$ ON 命令によってこの割込みが許可（マスク解除）状態だった場合、定義された割込みサブルーチンへ分岐します。 割込みサブルーチンは、RETURN 命令により終了してください。RETURN 命令により、割込みで分岐時の命令に戻ります。 時刻割込みは、１つのプログラム内で１つしか有効ではありません。複数の ON TIME$命令を１つのプログラム内に設定すると、最後の ON TIME$命令が有効になります。 時刻割込みのサブルーチンを実行中に、他の割込みがはいっても、その割込みはすぐに実行されず、時刻割込みのサブルーチン実行終了後、割込み優先順位にしたがって実行されます。 BASIC で割込みを作成する場合の詳細については、5-1 項「各種の処理」の「■割込機能の ON COM と ON PC 例」を参照してください。
例：	> 10 ON TIME$ = ”08:00:00” GOSUB 100 > 20 TIME$ ON > 30 GOTO 30 > 100 PRINT ”WAKE-UP!!!” > 110 RETURN
参照：	TIME$(ON｜OFF｜STOP), ON ALARM

ON TIMER	
構文：	ON TIMER <時間式> GOSUB <行番号> <ラベル>
機能：	ステートメント。一定時間間隔ごとに割込みサブルーチンを実行します。その割込み条件と割込み先を定義します。 本命令実行から指定時間の後、定義した 割込みサブルーチンに分岐します。割込みサブルーチン実行後、指定時間が経過すると、再度割込みサブルーチンに分岐し、これを繰り返します。（TIMER OFF 命令によって割込みが禁止されるまで繰り返します。）
説明：	<時間式> は、1〜864000 を返す整数式です。 時間式で指定する単位は、0.1 秒です。したがって、整数式の範囲は、0.1 秒〜24 時間の範囲です。 <行番号> は、割込みサブルーチンの開始位置を指定します。 <ラベル> は、GOSUB 命令の中で、行番号の代わりに使用されます。
参考：	TIMER ON 命令によって、有効（割込み許可）となります。 TIMER OFF 命令によって、割込み禁止となります。 TIMER STOP 命令によって、割込み停止（割込みがあったことを記憶）されます。 タイムアップしたときに、実行中であった区間が、TIMER ON 命令によってこの割込みが許可（マスク解除）状態だった場合、定義された割込みサブルーチンへ分岐します。 割込みサブルーチンは RETURN 文により終了させてください。RETURN 命令により、割込みで分岐時の命令にもどります。 タイマ割込みは、減算式タイマの割込みです。ON TIMER 命令実行後、指定した設定時間から減算を開始します。0 に達した時に、この割込みが許可（マスク解除）状態だった場合、定義された割込みサブルーチンへ分岐します。一度、タイマ割込みを実行（分岐処理）後、 タイマ割込みは、有効のままのため、再度指定時間後、同じ割込みサブルーチンが繰り返し実行されます。タイマ割込みは、TIMER OFF 命令により禁止（マスク）されるまで、繰り返されます。つまり、インターバルタイマ割込みです。この点が、ワンショットタイマ割込みの ON アラーム割込みと異なります。 タイマ割込みの実行中、BASIC 割込みが停止されます。ALARM 用のタイマと TIMER 割込みは、BASIC 割込みの実行中継続します。
例：	> 10 T = 0 > 20 ON TIMER 30 GOSUB 100 > 30 TIMER ON > 40 GOTO 40 > 100 T = T+3 > 110 PRINT T; ”SECONDS HAVE ELAPSED SINCE THE START OF EXECUTION.” > 120 RETURN
参照：	TIMER(ON|OFF｜STOP)

OPEN

構文：

OPEN# <**ポート式**>, " <**デバイス記号**>：[<**通信条件**> [, <**データ長**> [, <**パリティ**> [, <**ストップビット**> [, CS [, RS [, DS [, XN]]]]]]]] "

注：形 C200H-ASC02 **の** OPEN **命令の構文でも動作します。**

機能：

ステートメント。ポートを指定の条件（デバイス記号、ポートの通信条件、伝送制御信号の動作指定）で開きます。

説明：

<**ポート式**> **は、**1〜3 **の範囲の整数を返す式です。**

ポート式を省略すると、デフォルトの#1 **となります。**

注：[ ] **内の要素を省略する場合、この文字列のデフォルト設定は次のようになります。**

"xxxx, 8, N, 2, yy, yy, yy, yy"

xxxx：DM **設定に従います**

yy, yy, yy, yy：7-88 **ページ参照**

<**デバイス記号**> **は、ポートに接続されるデバイスの種類を指定します。デバイス記号による送／受信バッファの使用、伝送制御信号（**RS, ER**）の状態は、以下のとおりです。**

デバイス記号	**周辺装置名**（外部機器名）	使用バッファ	**ポート** OPEN **時**			**入出力命令**（PRINT、INPUT 命令）実行中	
			RS		ER	RS	ER
			OPEN **命令で** RS_OFF **時**	OPEN 命令で RS_ON 時		OPEN 命令で RS_ON／OFF 時とも	
TERM	**ターミナル**	**送／受信バッファとも使用**	OFF	ON	**変化せず**	ON	**変化せず**
SCRN	ディスプレイ	**送信バッファのみ使用**	OFF	ON	**変化せず**	ON	**変化せず**
KYBD	**キーボード**	**受信バッファのみ使用**	OFF	ON	**変化せず**	ON	**変化せず**
COMU	汎用通信機器	**送／受信バッファとも使用**	OFF	ON	**変化せず**	ON	**変化せず**
LPRT	**プリンタ**	**送信バッファのみ使用**	OFF	ON	**変化せず**	ON	**変化せず**
EKPRT	**プリンタ**	**送信バッファのみ使用**	OFF	ON	**変化せず**	ON	**変化せず**
NKPRT	**プリンタ**	**送信バッファのみ使用**	OFF	ON	**変化せず**	ON	**変化せず**

次ページへ続く

前ページより

説明：	<通信速度> は、300、600、1200、2400、4800、9600、19200、38400 のいずれかです。省略すると、デフォルトの通信条件設定は、9600bps です。 <データ長> は、7、8 のいずれかです。省略すると、デフォルトのデータ長は 8 ビットです。 <パリティ> は、N（なし）、E（偶数）、O（奇数）のいずれかです。省略すると、デフォルトのパリティは、N（なし）です。 <ストップビット> 数は、1、2 のいずれかです。省略すると、デフォルトのストップビット数は 2 です。 CS、RS、DS、XN の各項目は、各信号線をモニタ、制御、チェックするかどうかを指定します。 CS=CS_ON: ASCII ユニットは、PRINT 命令実行中に CTS 信号をモニタします。 CS=CS_OFF: ASCII ユニットは、PRINT 命令実行中に CTS 信号をモニタしません。 RS=RS_ON: ASCII ユニットは、ポート通信ポートをオープンしてからクローズするまで、常に RTS 信号を ON にします。 RS=RS_OFF: ASCII ユニットは、入出力命令（PRINT、INPUT）命令実行間時にのみ、RTS 信号を ON とし、それ以外は、RTS 信号を OFF とします。 DS=DS_ON: ASCII ユニットは、出力命令（PRINT、LPRINT）実行開始時、DSR 信号をチェックし、DSR 信号が ON ならば、周辺機器（外部機器）は、通信準備完了と判断します。DSR 信号が OFF ならば、周辺機器（外部機器）は、通信不能として、DSR 信号が ON になるまで待ちます。 DS=DS_OFF: ASCII ユニットは、出力命令（PRINT、LPRINT）実行開始時、DSR 信号をチェックしません。 XN=XN_ON: ASCII ユニットの受信バッファおよび周辺機器（外部機器）の受信バッファの両方に対する Xon/Xoff フロー制御を行います。 ASCII ユニットの受信バッファに対して：ASCII ユニット自身の受信バッファが 3/4 （384 バイト）に達すると、Xoff コードを送信し、周辺機器（外部機器）に対してデータ送信停止を要求します。自身の受信バッファが 1/4 （128 バイト）以下に下がると、Xon コードを送信し、周辺機器（外部機器）に対して、データ送信再開を要求します。 周辺機器（外部機器）の受信バッファに対して：ASCII ユニットは、PRINT 命令実行中に、外部機器からの Xoff コードを受信すると、データの送信を停止します。周辺機器（外部機器）から Xon コードを受信すると、データの送信を再開します。 XN=XN_OFF: Xon/Xoff フロー制御を行いません。 CS、RS、DS、XN を省略すると、そのデフォルト値は、それぞれオープンしたデバイスの値となります。 このときすべて省略せず、3 項目以下の省略に関しては、カンマ指定が必要です。 例：XN_ON のみ指定する場合 OPEN #1, ”COMU:9600, 8, E, 2, XN_ON”では「コンフィグレーションエラー」となります。 OPEN #1, ”COMU:9600, 8, E, 2,,,,XN_ON”で正常に動作します。 必ず、カンマを入力してください。 LOAD 命令または SAVE 命令実行中は、LOAD 命令または SAVE 命令で指定した条件（デバイス記号・通信条件・伝送制御信号の動作）が有効になります。LOAD 命令または SAVE 命令実行後、指定は元の状態に戻ります。

次ページへ続く

前ページより

参考：	1. **形** C200H-ASC02 **のための通信設定は次のとおりです。** OPEN # <**ポート式**>, "<**デバイス記号**>**：**[(<**通信条件・有効信号線**>)]" **詳細は、形** C200H-ASC02 **のユーザズマニュアル**(SBCC-442)**を参照してください。** 2. **ポート**#3 **は、形** C200H-ASC31 **のターミナルポートに対応しています。** 3. **形** C200H-ASC11 **または形** C200H-ASC21 **でポート**#3 **を指定すると、**"ILLEGAL FUNCTION CALL"**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。**
例：	> 10 OPEN #2, "COMU:9600, 8, N, 2, CS_ON, RS_OFF, DS_OFF, XN_OFF" > 20 PRINT #2, "HELLO WORLD"
参照：	CLOSE, LOAD

･通信条件、有効信号を省略したときは、各デバイスのデフォルト値が設定されます。

デバイス名	**通信速度**	**データフォーマット**	**有効信号**
TERM	DM **設定に従う**	8 **ビット、パリティなし、** 2 **ストップビット**	CS_OFF、RS_ON、DS_OFF、XN_OFF
SCRN	DM **設定に従う**	8 **ビット、パリティなし、** 2 **ストップビット**	CS_ON、RS_ON、DS_OFF、XN_OFF
KYBD	DM **設定に従う**	8 **ビット、パリティなし、** 2 **ストップビット**	CS_ON、RS_ON、DS_OFF、XN_OFF
COMU	DM **設定に従う**	8 **ビット、パリティなし、** 2 **ストップビット**	CS_ON、RS_ON、DS_OFF、XN_OFF
LPRT EKPRT NKPRT	DM **設定に従う**	8 **ビット、パリティなし、** 2 **ストップビット**	CS_OFF、RS_ON、DS_ON、XN_OFF

注：ASC21 **のポート** 2 **では、**CS、RS、DS、XN **の項目は関係ありません。**

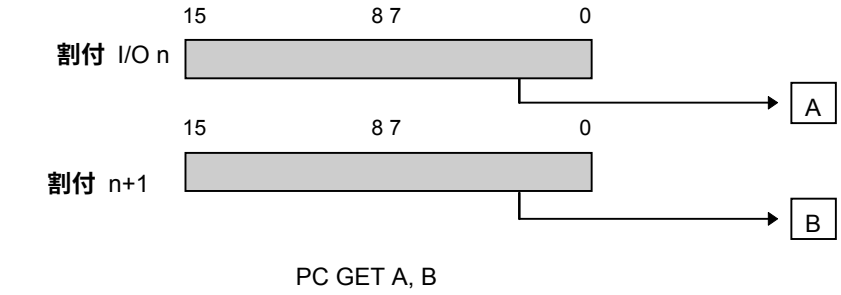
OPTION BASE	
構文：	OPTION BASE { 0 | 1 }
機能：	**ステートメント。配列変数の添字の始まりを** 1**（または** 0**）に指定します**
説明：	**配列変数の添字の始まりは通常** 0 **からですが、これを** 1 **から始めたいとき、この** OPTION BASE 1 **を使用します。**0 **にもどしたいとき、**OPTION BASE 0 **を実行します。**
例：	> 10 OPTION BASE 1 > 20 DIM A(2) > 30 FOR I = 0 TO 1 > 40 A(I) = I > 50 NEXT I > 60 FOR I = 0 TO 1 > 70 PRINT A(I) > 80 NEXT I > RUN BAD SUBSCRIPT ERROR IN LINE 40 > 10 OPTION BASE 1 > 20 DIM A(2) > 30 FOR I = 1 TO 2 > 40 A(I) = I > 50 NEXT I > 60 FOR I = 1 TO 2 > 70 PRINT A(I) > 80 NEXT I RUN 1 2
参照：	DIM

OPTION LENGTH	
構文：	OPTION LENGTH <**数値式**>
機能：	**ステートメント。文字変数の最大長を宣言し、変数へのメモリ割付け方式をスタティック方式とします。**
説明：	<**数値式**> **は、文字変数の内容の最大長を指定します。有効範囲は、**1～255 **です。** **プログラムの先頭行に入れるようにしてください。**
参考：	**文字変数へのメモリ割付けは、何も設定しない場合、ダイナミック方式となります。つまり、プログラム実行中に文字変数エリアの再割付けが発生します。この** OPTION LENGTH **命令で、必要な長さを宣言しておけば、プログラム実行前にメモリが割付けられるので、プログラム実行中に変数エリアの再割付け動作は発生しません。** PWORD **命令と組み合わせて使用する場合は、**PWORD **命令の後に入れてください。** OPTION LENGTH **命令は、１度だけ使用できます。文字変数を宣言する前にこの** OPTION LENGTH **命令を使用する必要があります。**
例：	> 10 OPTION LENGTH 32 > 20 A$ = ”THIS STRING IS LONGER THAN THE SPECIFIED LENGTH” > 30 PRINT A$ > RUN THIS STRING IS LONGER THAN THE S
参照：	DIM

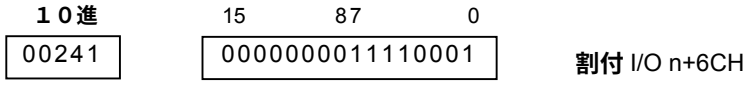
PC ON/OFF/STOP	
構文：	PC [<**割込み番号**>] | ON OFF STOP |
機能：	**ステートメント**。ON PC **命令による割込みを許可、禁止または停止します。**
説明：	<**割込み番号**> **は、どの** PC **割込み番号を** ON（**許可**）、OFF（**禁止**）**または** STOP（**停止**）**にするかを指定します。**1～99 **の範囲が可能です。**<**割込み番号**> **を省略すると、すべての** PC **割込み番号に適用されます。** ON **は、**PC **割込みを許可（マスク解除）にします。この** PC ON **命令から** PC OFF **命令までを実行中に、**PC **割込みが発生すると、定義した 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降** PC **割込みは無効となります。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくだけで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンを実行します。** **注：**PC ON/OFF/STOP **命令は、**ON PC **命令を実行した後にだけ実行可能です。**
例：	> 10 ON PC 1 GOSUB 60 > 20 ON PC 2 GOSUB 80 > 30 ON PC 3 GOSUB 100 > 40 PC ON > 50 GOTO 50 > 60 PRINT ”PC INTERRUPT #1” > 70 RETURN > 80 PRINT ”PC INTERRUPT #2” > 90 RETURN > 100 PRINT ”PC INTERRUPT #3” > 110 RETURN
参照：	ON PC

PC EGET	
構文：	[<読み出し結果> =] PC EGET# <共有メモリオフセットアドレス>,<読み出しチャネル数> , "<PC フォーマット> {, <PC フォーマット> } " ; <変数> {, <変数> }
機能：	ステートメント。ASCII ユニットが自身の共有メモリ（IOWR エリア）の指定アドレスからデータを読み出します。
説明：	<共有メモリオフセットアドレス> は、共有メモリの読み出し開始アドレスを示します。有効範囲は 0〜89 の整数です。 <読み出しチャネル数> は、共有メモリから読み出すチャネル（ワード）数です。有効範囲は、1〜90 の範囲の整数です。 <PC フォーマット> は、共有メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの概要は、PC READ 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーマットの説明」を参照してください。 <変数> は、任意の変数名です。<変数> は、読み出しチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。 PC EGET 命令は、<読み出し結果> を返すことができます。この <読み出し結果> は、PC EGET 命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<読み出し結果> =0 となり、共有メモリから読み出されたデータは有効です。命令実行が失敗した場合、<読み出し結果> =-1 となります。（命令実行失敗の原因は、別の PC EGET 命令が共有メモリを使用中のためなどです。） PC EGET 命令実行時に、すぐ内部の共有メモリからデータを読み込みます。
参考：	この PC EGET 命令を使用して、CPU ユニットから ASCII ユニットにデータを転送する場合、PC EGET 命令実行の前に、あらかじめ CPU ユニットのラダープログラム内の IOWR #00□□命令によって、ASCII ユニットの共有メモリにデータを書き込むことが必要です。BASIC プログラムの PC EGET 命令と CPU ユニットの IOWR #00□□命令との間には同期はありません。 <読み出しチャネル数> が共有メモリの IOWR エリアを超えるサイズを指定すると、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 PC フォーマットの指定に個数および型が対応していない場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 PC EGET 命令と IOWR #00□□命令の使用方法についての詳細は、6-3 項「データ変換方法の詳細」を参照してください。
例：	 PC EGET #0, 4, "1H4, 1H4, 2H4"; A, B, C$
参照：	PC EPUT, PC GET

PC EPUT	
構文：	[<書き込み結果>] = PC EPUT# <共有メモリオフセットアドレス>, <書き込みチャネル数>, " <PC フォーマット> {, <PC フォーマット> }", <数値式> {, <数値式> }
機能：	ステートメント。ASCII ユニットの共有メモリ（IORD エリア）の指定アドレスにデータを書き込みます。
説明：	<共有メモリオフセットアドレス> は、共有メモリの書き込み開始アドレスを示します。有効範囲は 0〜89 の整数です。 <書き込みチャネル数> は、共有メモリへ書き込むチャネル（ワード）数です。有効範囲は、1〜90 の範囲の整数です。<書き込みチャネル数> が共有メモリの IOWR エリアを超えるサイズを指定すると、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 <PC フォーマット>は、どのデータ形式で共有メモリに書き込むかを示します。PC フォーマットの概要は、PC READ 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーットの説明」を参照してください。 <数値式> は、任意の数値式で、有効範囲は-32768〜32767 です。数値式の値がこの範囲外の場合、命令は実行されません。<数値式> は、書き込みチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。対応していない場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC EPUT 命令は、<書き込み結果> を返すことができます。この <書き込み結果> は、PC EGET 命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<書き込み結果> =0 となり、共有メモリへのデータ書き込みは有効です。命令実行が失敗した場合、<書き込み結果> =-1 となります。（命令実行失敗の原因は、別の PC EPUT 命令または IORD#00□□命令が共有メモリを読み出し中のためなどです。）
参考：	PC EPUT 命令実行時に、すぐ内部の共有メモリへデータを書き込みます。 この PC EPUT 命令を使用して、ASCII ユニットから CPU ユニットにデータを転送する場合、PC EPUT 命令実行の後に、CPU ユニットのラダープログラム内の IORD #00□□命令によって、ASCII ユニットの共有メモリからデータを読み出すことが必要です。BASIC プログラムの PC EPUT 命令と CPU ユニットの IORD #00□□命令との間には同期はありません。 <書き込みチャネル数>が共有メモリの IOWR エリアを超えるサイズを指定すると、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC フォーマットの指定に個数および型が対応していない場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC EPUT 命令と IORD #00□□命令の使用方法についての詳細は、6-3 項「データ変換方法の詳細」を参照してください。
例：	共有メモリ 変数　アドレス 15　0 A → 50　16 進 4 桁 B → 51　16 進 4 桁 C$ → 52　16 進 4 桁 53　16 進 4 桁 4 ワード PC EPUT #0, 4, "1H4, 1H4, 2H4"; A, B, C$
参照：	PC EGET, PC PUT

PC GET	
構文：	PC GET <変数> [, <変数>]
機能：	ステートメント。割付 I/O の nCH のビット 0〜15 のデータを 1 番目の変数に読み込みます。同時に、割付 I/O の n+1CH のビット 0〜15 のデータを 2 番目の変数に読み込みます。いずれも、16 進数を 10 進数に変換して読み込みます。
説明：	<変数> は、任意の BASIC 変数名です。 1 番目の変数に、割付 I/O の nCH のビット 0〜15 が-32768〜32767 の範囲の整数で格納されます。 2 番目の変数に、割付 I/O の n+1CH のビット 0〜15 が-32768〜32767 の範囲の整数で格納されます 2 番目の <変数> を省略すると、割付 I/O の nCH のビット 0〜15 だけが読み込まれます。割付 I/O 内の 16 進数は、10 進値に変換され、変数に格納されます。
参考：	PC GET 命令は、命令実行直前の I/O リフレッシュ時のデータを読み込みます。 PC GET 文の使用方法の詳細については、第 6 章「CPU ユニットとのデータ交換」の各項を参照してください。
例：	 PC GET A, B
参照：	PC EGET, PC PUT

お願い PC GET では、配列変数は使用できません。

PC PUT	
構文：	PC PUT <数値式>
機能：	ステートメント。割付 I/O の n+6CH に、数値式（10 進数）を 16 進数に変換して格納します。
説明：	<数値式> は、-32768〜32767 の範囲の整数（任意の数値式）です。数値式が整数でない場合、割付 I/O に格納される前に、小数点以下が四捨五入され、整数に丸められます。
参考：	PC PUT 命令は、命令実行直後の I/O リフレッシュ時にデータを書き込みます。ASCII ユニットの BASIC プログラムは待機しません。
例：	 PC PUT 241
参照：	PC EPUT, PC GET

PC QREAD@

構文：

[<読み出し結果> =] PC QREAD ”<I/O メモリエリア種別>, <読み出し開始アドレス>, <読み出しチャネル数>, <PC フォーマット> {, <PC フォーマット> }” ; <変数> {, <変数> }

機能：

ステートメント。CPU ユニットに対して IOWR 命令実行要求を発行し、IOWR（#FD00）命令により書き込まれたデータを、指定変数に PC フォーマットにしたがって格納します。

説明：

<I/O メモリエリア種別> は、CPU ユニットのどの I/O メモリエリア種別からデータを読み出すかを指定します。I/O メモリエリア種別は以下のコードで指定します。

C200HE/HG/HX の場合

エリア種別	PC 機種	指定したいアドレス	PC QREAD@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	C200HE/HG/HX	D0000〜D6655	@D	0000〜6655
入出力リレー（CIO）		000〜511CH	@R	0000〜0511
リンクリレー（LR）		L00〜L63CH	@L	0000〜0063
保持リレー（HR）		H00〜H99CH	@H	0000〜0099
補助記憶リレー（AR）		AR00CH〜27CH	@A	0000〜0027
タイマ／カウンタ（T／C）		T/C000〜511	@G	0000〜0511

CS シリーズ の場合

エリア種別	PC 機種	指定したいアドレス*1	PC QREAD@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	CS シリーズ	D00000〜D06655	@D	0000〜6655
入出力リレー（CIO）		0000〜0511CH	@R	0000〜0511
データリンクリレー（CIO）		1000〜1063CH	@L	0000〜0063
保持リレー（HR）1		H000〜H099CH	@H	0000〜0099
保持リレー（HR）2		H100CH〜H127CH	@A	0000〜0027
タイマ（T）		T0000〜0511	@G	0000〜0511

＊1：これ以外の CS シリーズ CPU ユニットの I/O メモリアドレスを指定することはできません。CS シリーズ CPU ユニットでの指定方法の詳細については、6-6 項「CS シリーズを使用する場合の留意事項」を参照してください。

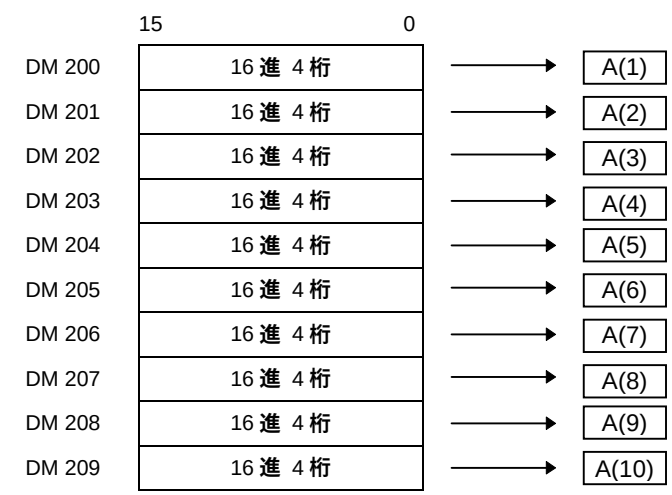
次ページへ続く

前ページより

説明：	<読み出し開始アドレス> は､指定 I/O メモリエリア種別内のデータ読み出し開始アドレスを指定します。開始アドレスは、指定 I/O メモリエリア種別のアドレス範囲内でなければなりません。 <読み出しチャネル数> は、指定開始アドレスから読み出すチャネル（ワード）数を指定します。有効範囲は、1 ～128 チャネル（ワード）です。 <PC フォーマット> は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの概要は、PC READ 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーットの説明」を参照してください。 <変数> は、任意の変数名です。<変数> は、読み出しチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。PC QREAD@命令は、<読み出し結果> を返すことができます。この <読み出し結果> は、PC QREAD @命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<読み出し結果> =0 となり、I/O メモリから読み出されたデータは有効です。命令実行が失敗した場合、<読み出し結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。） 注：PC QWRITE@命令によって指定される I/O メモリのエリアは、実際に転送されるエリアの指定ではなく、割付リレーエリア（n+8、n+9CH）に格納されるエリアの指定です。この割付リレーエリアは、CPU ユニットがモニタするためのエリアです。実際には、IORD 命令のオペランドによって指定されたエリアが、転送されます。この点は、PC QREAD@命令と異なります（PC QREAD@命令は、自身で指定した I/O メモリエリアからデータを転送します）。 したがって、CS シリーズの場合、PC QREAD@命令によって指定できるエリア範囲と IORD 命令によって指定できるエリア範囲が異なるため、PC QREAD@命令によっては指定できないエリアを転送するときは、PC QREAD@命令にダミーのエリアを指定してください（その場合、割付リレーエリアを使用した CPU ユニットでのモニタは行えません）。

次ページへ続く

前ページより

参考：	PC QREAD@命令は、CPU ユニットの IOWR(#FD0)命令とセットで使用してください。 ASCII ユニットは、PC QREAD@命令実行により、CPU ユニットに対して、IOWR 命令実行要求（割付 I/O の n+5CH のビット 15）を発行し、IOWR（#FD00）命令実行時にデータ転送されると、そのデータを指定変数に、PC フォーマットにしたがって格納します。ASCII ユニットは、PC QREAD 命令実行から IOWR(#FD0)命令実行によるデータ転送まで待機します。 読み出し開始アドレスが範囲外の場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC フォーマットの指定に個数が対応していない場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 I/O メモリのエリア種別>、<読み出し開始アドレス>、<読み出しチャネル数> は、IOWR 命令と PC QREAD@命令の両方で同じ設定にする必要があります。もし読み出し開始アドレスが異なっていた場合、IOWR 命令の設定にしたがって、読み出されます。もし読み出しチャネル数が異なっていた場合、エラーコード：0081 となります。PC QREAD@命令の使用方法の詳細については、第 6 章「CPU ユニットとのデータ交換」の各項を参照してください。
例：	 PC QREAD "@D, 200, 10, S10H4"; A(1)
参照：	PC QWRITE, PC READ, WAIT

PC QWRITE@

構文：

[<書き込み結果> =] PC QWRITE ”<I/O メモリエリア種別>, <書き込み開始アドレス>, <書き込みチャネル数>, <PC フォーマット> {, <PC フォーマット> }” ; <数値式> {, <数値式> }

機能：

ステートメント。CPU ユニットに対して IORD 命令実行要求を発行し、指定変数のデータを PC フォーマットにしたがって転送する用意をします。CPU ユニットの IORD（#FD00）命令により、そのデータが転送されます。

説明：

<I/O メモリエリア種別>は、CPU ユニットのどの I/O メモリエリア種別からデータを読み出すかを指定します。I/O メモリエリア種別は以下のコードで指定します。

C200HE/HG/HX の場合

エリア種別	PC 機種	指定したいアドレス	PC QWRITE@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	C200HS/HE/HG/HX	D0000〜D6143	@D	0000〜6143
入出力リレー（CIO）		000〜511CH	@R	0000〜0511
リンクリレー（LR）		L00〜L63CH	@L	0000〜0063
保持リレー（HR）		H00〜H99CH	@H	0000〜0099
補助記憶リレー（AR）		AR00CH〜27CH	@A	0000〜0027
タイマ／カウンタ（T／C）		T/C000〜511	@G	0000〜0511

CS シリーズ の場合

エリア種別	PC 機種	指定したいアドレス*1	PC QWRITE@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	CS シリーズ	D00000〜D06655	@D	0000〜6655
入出力リレー（CIO）		0000〜0511CH	@R	0000〜0511
データリンクリレー（CIO）		1000〜1063CH	@L	0000〜0063
保持リレー（HR）1		H000〜H099CH	@H	0000〜0099
保持リレー（HR）2		H100CH〜H127CH	@A	0000〜0027
タイマ（T）		T0000〜0511	@G	0000〜0511

＊1：これ以外の CS シリーズ CPU ユニットの I/O メモリアドレスを指定することはできません。CS シリーズ CPU ユニットでの指定方法の詳細については、6-6 項「CS シリーズを使用する場合の留意事項」を参照してください。

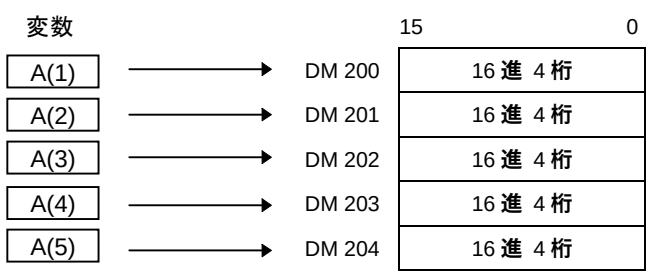
<書き込み開始アドレス> は、指定 I/O メモリエリア種別内のデータ書き込み開始アドレスを指定します。開始アドレスは、指定 I/O メモリエリア種別のアドレス範囲内でなければなりません。

<書き込みチャネル数> は、指定開始アドレスから書き込むチャネル（ワード）数を指定します。有効範囲は、1 〜128 チャネル（ワード）です。書き込みチャネル数が範囲外の場合、”FORMAT ERROR”（フォーマットエラー, エラーコード：B067 ）が発生します。

<PC フォーマット>は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの概要は、PC WRITE 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーットの説明」を参照してください。

次ページへ続く

前ページより

説明：	<数値式> は、任意の変数名です。<数値式> は、書き込みチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。対応していない場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC QWRITE@命令は、<書き込み結果> を返すことができます。この <書き込み結果> は、PC QWRITE@命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<書き込み結果> =0 となり、I/O メモリへ書き込まれたデータは有効です。命令実行が失敗した場合、<書き込み結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。） 注：PC QWRITE@命令によって指定される I/O メモリのエリアは、実際に転送されるエリアの指定ではなく、割付リレーエリア（n+8、n+9CH）に格納されるエリアの指定です。この割付リレーエリアは、CPU ユニットがモニタするためのエリアです。実際には、IORD 命令のオペランドによって指定されたエリアが、転送されます。この点は、PC WRITE@命令と異なります（PC WRITE@命令は、自身で指定した I/O メモリエリアにデータを転送します）。 したがって、CS シリーズの場合、PC QWRITE@命令によって指定できるエリア範囲と IORD 命令によって指定できるエリア範囲が異なるため、PC QWRITE@命令によっては指定できないエリアを転送するときは、 PC QWRITE@命令にダミーのエリアを指定してください（その場合、割付リレーエリアを使用した CPU ユニットでのモニタは行えません）。
参考：	PC QWRITE@命令は、CPU ユニットの IORD(#FD0)命令とセットで使用してください。ASCII ユニットは、PC QWRITE@命令実行により、CPU ユニットに対して、IORD 命令実行要求（割付 I/O の n+5CH のビット 14)を発行し、PC フォーマットにしたがって指定変数からデータを送信する準備をします。IORD(#FD0)命令が実行され、ASCII ユニットからデータが読み出されます 。ASCII ユニットは、PC QWRITE 命令実行から、IORD(#FD0) 命令実行によるデータ転送まで待機します。 書き込み開始アドレス範囲外の場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 書き込みチャネル数が範囲外の場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 PC フォーマットの指定に個数が対応していない場合、"FORMAT ERROR"（フォーマットエラー，エラーコード：B067）が発生します。 <I/O メモリのエリア種別>、<書き込み開始アドレス>、<書き込みチャネル数>は、IORD 命令と PC QWRITE@命令の両方で同じ設定にする必要があります。もし書き込み開始アドレスが異なっていた場合、IORD 命令の設定にしたがって、書き込まれます。もし <書き込みチャネル数> が異なっていたエラーコード：0081 となります。PC QREAD@命令の使用方法の詳細については、第 6 章「CPU ユニットとのデータ交換」の各項を参照してください。
例：	 PC QWRITE "@D, 200, 5, 1H4, 1H4, 1H4, 1H4, 1H4"; A(1), A(2), A(3), A(4), A(5)
参照：	PC QREAD、PC WRITE、WAIT

PC READ

構文：

[<読み出し結果> =] PC READ "<PC フォーマット> {, <PC フォーマット> }"; <変数> {, <変数> }

機能：

ステートメント。CPU ユニットからの PC READ 要求を受けると、割付 I/O で指定された I/O メモリからデータを読み出し、PC フォーマットにしたがって変数に格納します。

説明：

<PC フォーマット> は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの詳細は、付-3「PC フォーマットの説明」を参照してください。

PC フォーマット名称	フォーマット	説明
I 型(BCD) フォーマット	m I n	m チャネル分のデータを BCD データとみなして、各チャネルの下位 1 〜n 桁のデータを各変数に格納します。 m： チャネル数 n： チャネル内の指定桁数（1〜4） n：指定桁 1：桁 1（ビット 0〜3） 2：桁 1〜2（ビット 0〜7） 3：桁 1〜3（ビット 0〜11） 4：桁 1〜4（ビット 0〜15）
H 型(16 進) フォーマット	m H n	m チャネル分のデータを 16 進データとみなして、各チャネルの下位 1 〜n 桁のデータを各変数に格納します。 m：チャネル数 n： チャネル内の指定桁数（1〜4） n：上位桁 1：桁 1（ビット 0〜3） 2：桁 1〜2（ビット 0〜7） 3：桁 1〜3（ビット 0〜11） 4：桁 1〜4（ビット 0〜15）
O 型(8 進) フォーマット	m O n	m チャネル分のデータを 8 進データとみなして、各チャネルの下位１〜n 桁のデータを各変数に格納します。 m： チャネル数 n： チャネル内の指定桁数（1〜4） n：上位桁 1：桁 1（ビット 0〜3） 2：桁 1〜2（ビット 0〜7） 3：桁 1〜3（ビット 0〜11） 4：桁 1〜4（ビット 0〜15）

次ページへ続く

前ページより

説明：

B 型(ビット) フォーマット	m B n	m チャネル分のデータをバイナリデータとみなして、各チャネルの n ビット目のデータのみ（他を 0 とみなして）を 10 進データに変換して各数値変数に格納します。 m： チャネル数 n： チャネル内の指定ビット（0〜15）
A 型(ASCII) フォーマット	m A n	m チャネル分のデータを ASCII データとみなして、各チャネルの n で指定する上位／下位データを１個の文字変数に格納します。 m： チャネル数 n： チャネルの上位／下位バイト指定
S型(配列変数) フォーマット	S m X n	m チャネル分のデータを X で指定するデータ形式（I 型 、H 型、O 型、B 型のいずれか）とみなして、各チャネルのデータを n 個の各配列変数に格納します。 m： チャネル数 X： データ形式（I、H、O または B） n： Ｉ 、H、O または B の各指定フォーマットと同様

n	指定ビット
0	ビット 0
1	ビット 1
2	ビット 2
3	ビット 3
4	ビット 4
5	ビット 5
6	ビット 6
7	ビット 7
8	ビット 8
9	ビット 9
10	ビット 10
11	ビット 11
12	ビット 12
13	ビット 13
14	ビット 14
15	ビット 15

n：	バイト指定
1	下位バイト（ビット 0〜7）
2	上位バイト 1（ビット 8〜15）
3	下位および上位バイト 0〜1（ビット 0〜15）

次ページへ続く

前ページより

説明：	1．A 型フォーマットでは、ASCII ユニットの 1 変数に PC 側の複数チャネルを対応することができます。ただし、1 度に最大 127 チャネルまでです。 A 型、S 型フォーマット、および変数が文字型変数以外は、1 チャネルごとに１つの変数を対応させる必要があります。 A 型フォーマットは、1 つのフォーマットに対して 1 つの変数を対応させます。 S 型フォーマットは、1 つのフォーマットに対して 1 つの配列変数を対応させます。 A 型、S 型フォーマット、および変数が文字型変数以外では 1 つのフォーマットに対して複数の変数を対応させることができます。 2．m を省略したときは 1 チャネル分となります。 1 度に扱えるチャネル数は数値型の場合 1 ～255、文字型の場合 1～64 チャネルです。 フォーマットおよびサブコマンドは、文字変数でもかまいません。 フォーマットおよびサブコマンドは大文字で記述してください。 <変数> は、任意の変数名です。<変数> は、読み出しチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。 PC READ 命令は、オプションで、<読み出し結果> を返すことができます。この <読み出し結果> は、PC READ 命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<読み出し結果> =0 となり、I/O メモリから読み出されたデータは有効です。命令実行が失敗した場合、<読み出し結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。例えば、メインルーチン内で PC READ 命令でデータ転送を実行中に、割込みプログラム内でも PC READ 命令でデータ転送を実行した場合などです。） ASCII ユニットは、CPU ユニットからの PC READ 要求（割付 I/O nCH のビット 01）を受けると、割付 I/O （n+3CH, n+4CH）で指定された I/O メモリを読み出し、PC フォーマットにしたがって、指定した変数に格納します。 CPU ユニットからの PC READ 要求の次の I/O リフレッシュ時に、データ転送されます。 ASCII ユニットは、PC READ 命令を実行してから、データ転送終了まで、BASIC プログラムの実行を待機します。 PC READ 文の使用方法については、6-3 項「データ変換方法の詳細」の各節を参照してください。
参考：	1. PC の出力データ長が変数データ長より大きい場合、余分な出力データ部分は無視されます。 2. PC の出力データ長が変数データ長より小さい場合、不足が発生した変数への読み込みは実行されません。 3. PC フォーマットの指定に個数が対応していない場合、”FORMAT ERROR”（フォーマットエラー, エラーコード：B067）が発生します。 <変数> に文字列型変数を使用した場合、従来の形 C200H-ASC02 と動作が異なりますで、付-1「ご使用上の留意事項」、付-3「PC フォーマットの説明」をよくお読みになってからご使用ください。
例：	CPU ユニット　　ASCII ユニット 15　　0　　変数 注 [16 進 4 桁] → A(1) [16 進 4 桁] → A(2) [16 進 4 桁] → A(3) 注：I/O メモリの先頭アドレスと転送チャネル数は、割付 I/O の n+3, n+4CH での指定によります。 PC READ ”S3H4”; A(1)
参照：	PC QREAD, PC WRITE, WAIT

PC READ@

構文：

[<読み出し結果> =] PC READ "<I/O メモリエリア種別>, <読み出し開始アドレス>, <読み出しチャネル数>, <PC フォーマット> {, <PC フォーマット> }, " ; <変数> {, <変数> }

機能：

ステートメント。ASCII ユニット単独で、指定 I/O メモリからデータを読み出します。ASCII ユニットが、I/O メモリの種別、読み出し開始アドレス、読み出しチャネル数をすべて指定します。

説明：

<I/O メモリエリア種別> は、CPU ユニットのどの I/O メモリエリア種別からデータを読み出すかを指定します。I/O メモリエリア種別は以下のコードで指定します。

C200H/HS/HE/HG/HX の場合

エリア種別	PC 機種	指定したいアドレス	PC READ@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	C200H	D0000〜D1999	@D	0000〜1999
	C200HS/HE/HG/HX	D0000〜D6655		0000〜6655
入出力リレー（CIO）	C200H	000〜255CH	@R	0000〜0255
	C200HS	000〜299CH		0000〜0299
	C200HE/HG/HX	000〜511CH		0000〜0511
リンクリレー（LR）	C200H/HS/HE/HG/HX	L00〜L63CH	@L	0000〜0063
保持リレー（HR）		H00〜H99CH	@H	0000〜0099
補助記憶リレー（AR）		AR00CH〜27CH	@A	0000〜0027
拡張データメモリ（EM） バンク No.0	C200HG/HX	E0_0000〜E0_6143	@E	0000〜6143
タイマ／カウンタ（T／C）	C200H/HS/HE/HG/HX	T/C000〜511	@G	0000〜0511

CS シリーズの場合

エリア種別	PC 機種	指定したいアドレス*1	PC READ@命令での指定方法	
			コード	先頭アドレス
データメモリ（DM）	CS シリーズ	D00000〜D06655	@D	0000〜6655
入出力リレー（CIO）		0000〜0511CH 注：0281CH は指定不可	@R	0000〜0511 注：0281 は指定不可
データリンクリレー（CIO）		1000〜1063CH	@L	0000〜0063
保持リレー（HR）1		H000〜H099CH	@H	0000〜0099
保持リレー（HR）2		H100CH〜H127CH 注：H101CH は指定不可	@A	0000〜0027 注：0001 は指定不可
拡張データメモリ（EM）バンク No.0		E0_00000〜E0_06143	@E	0000〜6143
タイマ（T）		T0000〜0511	@G	0000〜0511

＊1：これ以外の CS シリーズ CPU ユニットの I/O メモリアドレスを指定することはできません。CS シリーズ CPU ユニットでの指定方法の詳細については、6-6 項「CS シリーズを使用する場合の留意事項」を参照してください。

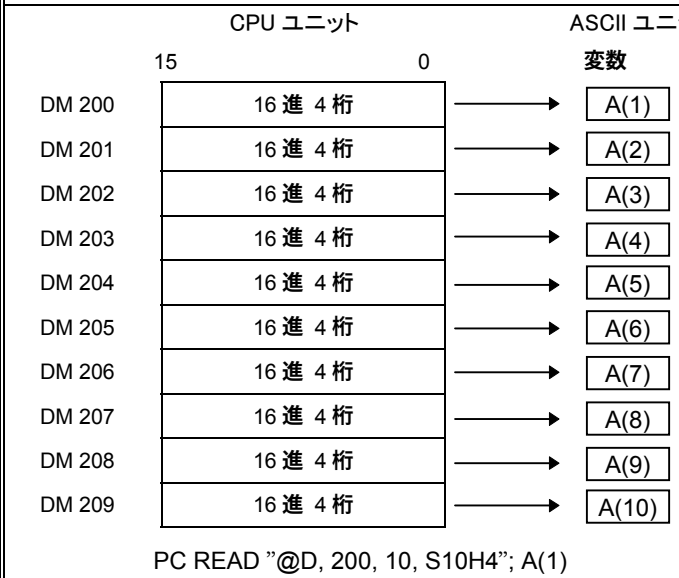
注：EM は、C200HG/HX だけで使用できます。

<読み出し開始アドレス> は、指定 I/O メモリエリア種別内のデータ読み出し開始アドレスを指定します。開始アドレスは、指定 I/O メモリエリア種別のアドレス範囲内でなければなりません。

<読み出しチャネル数> は、指定開始アドレスから読み出すチャネル（ワード）数を指定します。有効範囲は、1 〜255 チャネル（ワード）です。

次ページへ続く

前ページより

説明：	<PC フォーマット> は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの概要は、PC READ 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーットの説明」を参照してください。 <変数> は、任意の変数名です。<変数> は、読み出しチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。 PC READ@命令は、<読み出し結果> を返すことができます。この<読み出し結果> は、PC READ @命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<読み出し結果> =0 となり、I/O メモリから読み出されたデータは有効です。命令実行が失敗した場合、<読み出し結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。例えば、メインルーチン内で PC READ@命令でデータ転送を実行中に、割込みプログラム内でも PC READ@命令でデータ転送を実行した場合などです。）
参考：	ASCII ユニットは、CPU ユニットからの指示なく、単独で、CPU ユニットの I/O メモリエリアを指定して、データを読み出し、それを PC フォーマットによって指定したデータ形式とみなして、指定した変数に格納します。 PC READ@命令実行直後の I/O リフレッシュ時に、データが転送されます。ASCII ユニットは、PC READ@命令実行からデータ転送まで、BASIC プログラムの実行を待機します。 範囲外の場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 読み出しチャネル数が範囲外の場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 PC フォーマットの指定に個数が対応していない場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 PC READ@命令の使用方法については、6-3 項「データ変換方法の詳細」の各節を参照してください。 <変数> に文字列型変数を使用した場合、従来の形 C200H-ASC02 と動作が異なりますで、付-1「ご使用上の留意事項」、付-3「PC フォーマットの説明」をよくお読みになってからご使用ください。
例：	 PC READ "@D, 200, 10, S10H4"; A(1)
参照：	PC QREAD, PC WRITE, WAIT

PC WRITE

構文： [<書き込み結果> =] PC WRITE ”<PC フォーマット> {, <PC フォーマット> }”; <数値式>{, <数値式> }

機能： ステートメント。CPU ユニットからの PC WRITE 要求を受けると、変数のデータを割付 I/O で指定された I/O メモリへ PC フォーマットにしたがって書き込みます。

説明： <PC フォーマット> は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの詳細は、付-3「PC フォーマットの説明」を参照してください。

PC フォーマット名称	フォーマット	説明
I 型（BCD）フォーマット	m I n	m 個の各変数の下位 1 〜n 桁のデータを BCD データに変換して m チャネル分の各チャネル下位 1 〜n 桁に格納します。 m： チャネル数 n： チャネル内の指定桁数（1〜4） n： 指定桁 1： 桁 1（ビット 0〜3） 2： 桁 1〜2（ビット 0〜7） 3： 桁 1〜3（ビット 0〜11） 4： 桁 1〜4（ビット 0〜15）
H 型(16 進)フォーマット	m H n	m 個の各変数の下位 1 〜n 桁のデータを 16 進データに変換して m チャネル分の各チャネル下位 1 〜n 桁に格納します。 m：チャネル数 n： チャネル内の指定桁数（1〜4） n： 上位桁 1： 桁 1（ビット 0〜3） 2： 桁 1〜2（ビット 0〜7） 3： 桁 1〜3（ビット 0〜11） 4： 桁 1〜4（ビット 0〜15）
O 型(8 進)フォーマット	m O n	m 個の各変数の下位 1 〜n 桁のデータを 8 進データに変換して m チャネル分の各チャネル下位 1 〜n 桁に格納します。 m：チャネル数 n： チャネル内の指定桁数（1〜4） n： 上位桁 1： 桁 1（ビット 0〜3） 2： 桁 1〜2（ビット 0〜7） 3： 桁 1〜3（ビット 0〜11） 4： 桁 1〜4（ビット 0〜15）

次ページへ続く

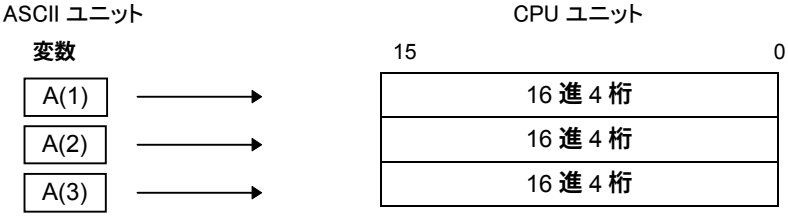
前ページより

説明：

B 型(ビット) フォーマット	m B n	**m 個の各数値変数を** 10 **進とみなしてバイナリデータに変換し、各** n **ビット目のデータのみを（他を 0 として）** m **チャネル分の各チャネルに格納します。** m**：チャネル数** n**： チャネル内の指定ビット（**0～15**）** n：指定ビット 0：ビット 0 1：ビット 1 2：ビット 2 3：ビット 3 4：ビット 4 5：ビット 5 6：ビット 6 7：ビット 7 8：ビット 8 9：ビット 9 10：ビット 10 11：ビット 11 12：ビット 12 13：ビット 13 14：ビット 14 15：ビット 15
A **型**(ASCII) **フォーマット**	m A n	1 **個の文字変数の先頭から** 2×m **文字分を** ASCII **データに変換して（**2 **文字ずつを** 1 **チャネルに対応させ）、**m **チャネル分の各チャネルの、**n **で指定する上位／下位バイト位置に格納します。** m**：チャネル数** n**： チャネルの上位／下位バイト指定** n：：バイト指定 1：下位バイト（ビット 0～7） 2：上位バイト 1（ビット 8～15） 3：下位および上位バイト 0～1（ビット 0～15）
S型(配列変数) フォーマット	S m X n	n **個の各配列変数のデータを、**X **で指定するデータ形式（**I **型 、**H **型 、**O **型 、**B **型のいずれか）に変換して、** m **チャネル分の各チャネルに格納します。** m**：チャネル数** X**： データ形式（**I、H、O **または** B**）** n**：** I、H、O **または** B **の各指定フォーマットと同様**

次ページへ続く

前ページより

説明：	1．A 型フォーマットでは、ASCII ユニットの変数に PC 側の複数チャネルを対応することができます。ただし、1 度に最大 127 チャネルまでです。 A 型、S 型フォーマット、および変数が文字型変数以外は１チャネルごとに 1 つの変数を対応させる必要があります。 A 型フォーマットは１つのフォーマットに対して 1 つの変数を対応させる。 S 型フォーマットは１つのフォーマットに対して 1 つの配列変数を対応させます。 A 型、S 型フォーマット、および変数が文字型変数以外では 1 つのフォーマットに対して複数の変数を対応させることができます。 2．m を省略したときは 1 チャネル分となります。 1 度に扱えるチャネル数は数値型の場合 1～255、文字型の場合 1～64 チャネルです。 フォーマットおよびサブコマンドは、文字変数でもかまいません。 フォーマットおよびサブコマンドは大文字で記述してください。 <数値式> は、任意の変数名です。<数値式> は、書き込みチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。数値式の値が、小数部分を含む場合は、小数点以下を切り捨てた値となります。単精度実数型および倍精度実数型の数値式は、内部で整数型に変換されます。 PC WRITE 命令は、<読み出し結果> を返すことができます。この <読み出し結果> は、PC WRITE 命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<書き込み結果> =0 となり、I/O メモリから読み出されたデータは有効です。命令実行が失敗した場合、<書き込み結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。例えば、メインルーチン内で PC WRITE 命令でデータ転送を実行中に、割込みプログラム内でも PC WRITE 命令でデータ転送を実行した場合などです。）
参考：	ASCII ユニットは、CPU ユニットからの PC WRITE 要求（割付 I/O nCH のビット 02）を受けると、指定した変数のデータを、割付 I/O（n+3CH, n+4CH）で指定された I/O メモリへ、PC フォーマットにしたがって書き込みます。 対応していない場合、”FORMAT ERROR”（フォーマットエラー, エラーコード：B067）が発生します。 <変数> に文字列型変数を使用した場合、従来の形 C200H-ASC02 と動作が異なりますで、付-1「ご使用上の留意事項」、付-3「PC フォーマットの説明」をよくお読みになってからご使用ください。 CPU ユニットからの PC WRITE 要求の次の I/O リフレッシュ時に、データ転送されます。ASCII ユニットは、PC WRITE 命令を実行してから、データ転送終了まで、BASIC プログラムの実行を待機します。
例：	 注：I/O メモリの先頭アドレスと転送チャネル数は、割付 I/O の n+3, n+4CH での指定によります。 PC WRITE ”S3H4”; A(1)
参照：	RC READ, PC QWRITE, WAIT

PC WRITE@

構文：

[<**書き込み結果**> =] PC WRITE ”<I/O **メモリエリア種別**>, <**書き込み開始アドレス**>, <**書き込みチャネル数**>, <PC **フォーマット**> {, <PC **フォーマット**> } ”, ; <**数値式**> { , <**数値式**> }

機能：

ステートメント。ASCII ユニット単独で、指定 I/O メモリデータを書き込みます。ASCII ユニットが、I/O メモリの種別、読み出し開始アドレス、書き込みチャネル数をすべて指定します。

説明：

<I/O **メモリエリア種別**> **は、**CPU **ユニットのどの** I/O **メモリエリア種別へデータを書き込むかを指定します。**I/O **メモリエリア種別は以下のコードで指定します。**

C200H/HS/HE/HG/HX **の場合**

エリア種別	PC **機種**	**指定したいアドレス**	PC WRITE@**命令での指定方法**	
			コード	**先頭アドレス**
データメモリ（DM**）**	C200H	D0000〜D1999	@D	0000〜1999
	C200HS/HE/HG/HX	D0000〜D6143		0000〜6143
入出力リレー（CIO**）**	C200H	000〜255CH	@R	0000〜0255
	C200HS	000〜299CH		0000〜0299
	C200HE/HG/HX	000〜511CH		0000〜0511
リンクリレー（LR**）**	C200H/HS/HE/HG/HX	L00〜L63CH	@L	0000〜0063
保持リレー（HR**）**		H00〜H99CH	@H	0000〜0099
補助記憶リレー（AR）		AR00CH〜27CH	@A	0000〜0027
拡張データメモリ（EM）バンク No.0	C200HG/HX	E0_0000〜E0_6143	@E	0000〜6143
タイマ／カウンタ（T／C)	C200H/HS/HE/HG/HX	T/C000〜511	@G	0000〜0511

CS **シリーズ の場合**

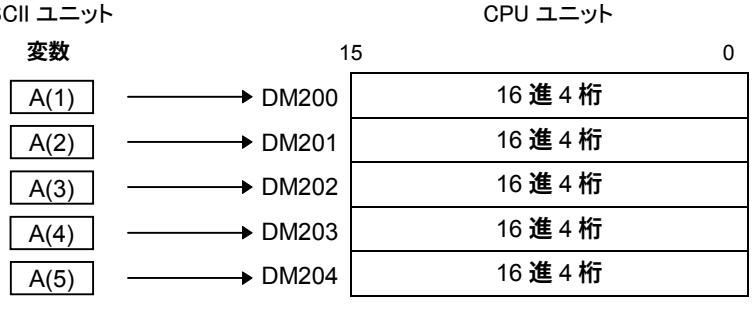
エリア種別	PC **機種**	**指定したいアドレス***1	PC WRITE@**命令での指定方法**	
			コード	**先頭アドレス**
データメモリ（DM**）**	CS **シリーズ**	D00000〜D06655	@D	0000〜6655
入出力リレー（CIO**）**		0000〜0511CH **注：**0281CH **は指定不可**	@R	0000〜0511 **注:**0281 **は指定不可**
データリンクリレー（CIO**）**		1000〜1063CH	@L	0000〜0063
保持リレー（HR**）**1		H000〜H099CH	@H	0000〜0099
保持リレー（HR**）**2		H100CH〜H127CH **注：**H101CH **は指定不可**	@A	0000〜0027 **注:**0001 **は指定不可**
拡張データメモリ（EM）バンク No.0		E0_00000〜E0_06143	@E	0000〜6143
タイマ（T**）**		T0000〜0511	@G	0000〜0511

＊1：**これ以外の** CS **シリーズ** CPU **ユニットの** I/O **メモリアドレスを指定することはできません。**CS **シリーズ** CPU **ユニットでの指定方法の詳細については、**6-6 **項「**CS **シリーズを使用する場合の留意事項」を参照してください。**

注：EM **は、**C200HE/HG/HX **だけで使用できます。**

次ページへ続く

前ページより

説明：	<書き込み開始アドレス> は、指定 I/O メモリエリア種別内のデータ書き込み開始アドレスを指定します。開始アドレスは、指定 I/O メモリエリア種別のアドレス範囲内でなければなりません。 <書き込みチャネル数> は、指定開始アドレスから書き込むチャネル（ワード）数を指定します。有効範囲は、1〜255 チャネル（ワード）です。書き込みチャネル数が範囲外の場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 <PC フォーマット> は、I/O メモリの指定データを、どのデータ形式と見なすかを示します。PC フォーマットの概要は、PC WRITE 命令の項を参照ください。PC フォーマットの詳細は、付-3「PC フォーマットの説明」を参照してください。 <数値式> は、任意の変数名です。<数値式> は、書き込みチャネル数に個数が対応していることが必要です。また、PC フォーマットの指定に個数および型が対応していることが必要です。 PC WRITE@命令は、<書き込み結果> を返すことができます。この <書き込み結果> は、PC WRITE @命令の実行が成功したか失敗したかを示します。命令実行が成功した場合、<書き込み結果> =0 となり、I/O メモリへ書き込まれたデータは有効です。命令実行が失敗した場合、<書き込み結果> =-1 となります。（命令実行失敗の原因は、他のデータ転送を実行中のためなどです。例えば、メインルーチン内で PC WRITE@命令でデータ転送を実行中に、割込みプログラム内でも PC WRITE@命令でデータ転送を実行した場合などです。）
参考：	ASCII ユニットは、CPU ユニットからの指示なく、単独で、CPU ユニットの I/O メモリエリアを指定して、指定した変数のデータを PC フォーマットに従って書き込みます。 PC WRITE @命令実行直後の I/O リフレッシュ時に、データが転送されます。ASCII ユニットは、PC WRITE @命令実行からデータ転送まで、BASIC プログラムの実行を待機します。 範囲外の場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 書き込みチャネル数が範囲外の場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 対応していない場合、"FORMAT ERROR"（フォーマットエラー, エラーコード：B067）が発生します。 <変数> に文字列型変数を使用した場合、従来の形 C200H-ASC02 と動作が異なりますで、付-1「ご使用上の留意事項」、付-3「PC フォーマットの説明」をよくお読みになってからご使用ください。 PC WRITE @命令の使用方法については、6-3 項「データ交換方法の詳細」の各節を参照してください。
例：	 PC WRITE "@D,200,5,1H4,1H4,1H4,1H4,1H4"; A(1),A(2),A(3),A(4),A(5)
参照：	PC QREAD, PC WRITE, WAIT

PEEK	
構文：	PEEK（ <アドレス式> ）
機能：	関数。ユーザメモリエリア内での指定番地（アドレス）の内容（ASCII コード→10 進数変換値）を読み出して返します。
説明：	<アドレス式> は、内容を読み出されるユーザメモリエリアの番地 （&H0E000〜&H3FFFF ）です。16 進数で入力します。有効範囲は 16 進：&H0E000〜&H3FFFF です。 出力（戻り）は、<アドレス式> で指定された番地（アドレス）の内容 1 バイトの ASCII コードを、10 進数に変換した整数値です。返される整数は、0〜255 の範囲内です。
参考：	ユーザメモリエリア 内には、変数エリアの他にプログラム実行用メモリエリアなどもありますので、取扱いに注意してください。
例：	> 10 A = PEEK(&H30000) > 20 PRINT A > RUN 32
参照：	POKE

PGEN	
構文：	PGEN <プログラム式>
機能：	コマンド。4 つの BASIC プログラムエリアのいずれかを選択します。ターミナルからのすべてのコマンドは、この選択されたアクティブなプログラムエリアに対して働きます。
説明：	<プログラム番号> は、1〜4 の範囲内にある整数です。
参考：	<プログラム番号> が許容範囲外である場合、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。 ターミナルから入力されるすべてのコマンドは、この PGEN コマンドによって選択されたアクティブなプログラムエリアに対して働きます。
例：	> PGEN 1 > LIST 10 REM THIS IS THE PROGRAM IN MEMORY AREA #1 20 PRINT ”HELLO WORLD” > PGEN 2 > LIST 10 REM THIS IS THE PROGRAM IN MEMORY AREA #2 20 PRINT ”HELLO AGAIN WORLD” >
参照：	PNAME, PINF, PMEM

PINF	
構文：	PINF [ALL | <プログラム番号>]
機能：	コマンド。ユーザメモリ エリア（指定される BAIC プログラム番号、中間コード）についての情報を表示します。
説明：	<プログラム番号>は、1〜4 の範囲内の整数です。 ALL は、すべてのユーザメモリエリア（BASIC プログラム番号 1〜4、中間コード）についての情報を 表示します。 <プログラム番号>は、指定したプログラム番号についての情報を表示します。 <プログラム番号>を省略した場合、現在のプログラム番号についての情報を表示します。 指定プログラム番号のプログラムエリアの名前 [書き込み禁止ステータス]、使用バイト数、中間コードの使用バイト数、ユーザメモリの残りの未使用バイト数が、次のように表示されます。 P1 ：名前 [R/W] 使用バイト数 P2 ：名前 [R/W] 使用バイト数 P3 ：名前 [R/W] 使用バイト数 P4 ：名前 [R/W] 使用バイト数 LIB ： 使用バイト数 I_CODE ： 使用バイト数 未使用バイト数 注： P1〜P4：プログラム番号 1〜4、 書き込み禁止ステータスは、R：書き込み禁止オン、R/W：書き込み禁止オフ LIB：システム予約 I_CODE：中間コード
参考：	本 ASCII ユニットは起動（RUN）させたときに、実行するプログラムの中間コードを生成します。その時、中間コードを生成するだけのメモリが足りなくなることがありますので、プログラム作成の際には、RUN コマンドを実行して I_CODE のサイズと未使用バイト数を確認してください。中間コード生成後の未使用メモリは、変数エリアなどに使用されます。 <プログラム番号> が許容範囲外である場合、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	> PINF ALL P1 : TEST [R/W] 6523 BYTES P2 : [R/W] 0 BYTES P3 : [R] 2876 BYTES P4 : MY_PROG [R] 23231 BYTES LIB : 0 BYTES I_CODE : 13046 BYTES 157890 BYTES FREE >
参照：	PGEN, PMEM, PNAME

PMEM	
構文：	PMEM | ON OFF |
機能：	**コマンド。現在のプログラムの書き込み禁止をオンまたはオフにします。**
説明：	PMEM **文は、現在の** BASIC **プログラム を書き込み禁止にします。**PMEM ON **により書き込み禁止を設定すると、現在の** BASIC **プログラム 、変更または削除できません。**
参考：	BASIC **プログラムの書き込み禁止を解除するためには、**PMEM OFF **を実行します。** RENUM **の禁止は** PMEN ON **では設定できません。**
例：	> 10 REM THIS PROGRAM WILL BE PROTECTED > PREM ON > DEL 10 PROTECTED PROGRAM > PMEM OFF > DEL 10 > LIST >
参照：	DEL, EDIT, NEW PGEN, PINF, PNAME

PNAME	
構文：	PNAME <**文字列式**>
機能：	**コマンド。現在の** BASIC **プログラムエリア（**PGEN **コマンドによってプログラム番号を選択します）に名前を割り当てます。**
説明：	<**文字列式**> **は、最長** 8 **文字の、任意の 文字列です。名前が** 8 **文字を超える場合は、以降が切り捨てられます。名前では、大文字と小文字が区別されます。** PNAME **コマンドによりプログラムに名前が割り当てられた場合、**NEW **コマンドからは保護されます（** NEW **コマンドによってはプログラムは削除されなくなります）。この場合、**PNAME ” ”**を実行し、プログラム名を削除してから、**NEW **コマンドを実行することで、プログラムは削除されます。** **なお、**EDIT **コマンドと** DEL **コマンドを使って、プログラムを編集することはできます。**
参考：	**プログラムエリアに割り当てられた名前は、**PNAME **コマンドに名前として空の文字列を割り当てると削除できます。（**PNAME” ”**）**
例：	> 10 REM THIS PROGRAM HAS A NAME > PNAME ”PROG1” > PINF P1 : PROG1 [R/W] 30 BYTES >
参照：	DEL, EDIT, NEW PGEN, PINF, LOAD

POKE	
構文：	POKE <アドレス式>, <数値式>
機能：	ステートメント。ユーザメモリエリア内での指定番地（&H0E000～＆H3FFFF ）に 1 バイトのデータを書き込みます。
説明：	<アドレス式> は、内容を書き込むユーザメモリエリアの番地（アドレス）です。16 進数で入力します。有効範囲は 16 進：&H0E000～&H3FFFF　です。 <数値 式> は、指定される番地（アドレス）に書き込まれる 1 バイトのデータです。有効範囲は 16 進：&H00～FF（10 進：0～255）です。
参考：	ユーザメモリエリア内には変数エリアの他にプログラム実行用メモリエリアなども含まれていますので、データの書込みについては十分注意してください。システムの暴走につながる場合があります。POKE 命令で変数以外のアドレスにデータ書込みは行わないでください。
例：	POKE &H25000, &H65（&H25000 は変数のアドレス）
参照：	PEEK

PRINT

構文：

PRINT [# <ポート式>]
[USING "<フォーマット文字列>" ;] <式> { [, | ; | <空白>] <式> }

機能：

ステートメント。指定ポートにデータを送信します。

説明：

<ポート式> は、1〜3 の範囲の整数を返す式です。
ポート式を省略すると、ターミナルポート No.（形 C200H-ASC11／21 の場合は＃1 、形 C200H-ASC31 の場合 は＃3 ）となります。

<式> は、任意の有効な数値または文字式です。式が <フォーマット文字列> に対応していない場合、"TYPE MISMATCH"（型不一致エラー, エラーコード：B013）が発生します。

<式> がセミコロン（;）または空白で区切られ、かつ、フォーマット文字列を使用しない場合、式（データ）は、送信バッファから連続で送信されます。<式> がカンマ（, ）で区切られた場合、式（データ）は、タブ文字で区切って送信されます。

終端のカンマ（, ）またはセミコロン（;）を省略すると、式（データ）の終わり CR（改行）が付加されます。

数値式の場合、式（データ）の前後に空白が挿入されて送信されます。数値式の前の空白は、マイナス符号用になっています。

式を指定しない場合、この命令は、送信バッファに、CR（改行）のみを送信します。

オプションの <フォーマット文字列> は、USING の後に、つぎのように指定します。PRINT 命令では、表示データを見やすくするためなどに、この USING <フォーマット文字列> を指定して、表示書式を指定できます。
複数の<式>で指定する場合は、それぞれの<式>に対応するフォーマットをスペースで区切ってください。スペースがない場合は、正しくフォーマットに従って出力されないことがあります。

送信データ種類	コード	説明
文字列	!	文字列の最初の 1 文字だけを送信します。
	& &	&と&に囲まれた文字数分を送信します。（文字数に&も含みます。）
	@	対応する文字列をそのまま出力します。
数値	#	数字の桁（負符号も含む）を示します。
	.	任意の位置に小数点．を挿入します。
	+	数値（#）の直前または直後に、符号（+）を付けます。
	-	数値（#）の終わりに指定すると、数値（#）を負数の符号にすることを示します。値が正の場合、-を数値（#）の終わりに指定すると、*が上位空白桁に入ります。
	**	数値（#）の上位空白桁に*を付けます。
	¥¥	数値（#）の始まりに¥を接頭辞として付けます。¥は 1 桁用の空白を割付けます。
	**¥	上記**と¥の組み合わせです。
	,	整数部を右側より 3 桁ごとにコンマで区切ります。
	^^^^	数値（#）の終わりに指定すると、指数形式であることを示します。　指定形式（E+nn, D+nn）で出力します。
	^^^^^	数値（＃）の終わり指定すると、指数形式であることを示します。　指定形式（E+nnn,D+nnn）で出力します。
	__	__は、__に続く 1 文字をそのまま送信します。特殊文字の前に__を置くと、文字列として特殊文字を出力できます。

符号を含む数字の桁数を、＃の個数で指定します。（＃で指定した桁数より、データの桁数が小さい場合、右詰めで表示されます。大きいとき、数値の前に"%"が出力されます。）

次ページへ続く

前ページより

参考：	**形** C200H-ASC31 **のポート**#1 **および、形** C200H-ASC11**、形** C200H-ASC21**、形** C200H-ASC31 **のポート**#2 **の場合、**DMA**（ダイレクトメモリアクセス）データ送信が可能です。**DMA **送信を使用するかどうかは、ポート**#1 **の場合、**DM m+2 **のビット** 08～15 **およびポート**#2 **の場合、**DM m+3 **のビット** 08～15 **で指定します。最大** 255 **バイトのデータが** DMA **送信可能です。** **注：式が** <**フォーマット文字列**> **に対応していない場合、**"TYPE MISMATCH"**（型不一致エラー**, **エラーコード：**B013**）が発生します。** 1. **ポートは、送信可能デバイスとして、**OPEN **命令によって開いておく必要があります。** 2. **ポート**#3 **は、形** C200H-ASC31 **のターミナルポートに対応しています。** 3. **形** C200H-ASC11 **または形** C200H-ASC21 **でポート**#3 **を指定すると、**"ILLEGAL FUNCTION CALL"**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。** 4. **フォーマット文字列（数値送信）で指定された桁数より、送信する式（データ）の桁数が大きい場合、**"%"**が数値の前に送信されます。**
例：1	> 10 PRINT "HELLO WORLD" > RUN HELLO WORLD
例：2	**例：** >10 X=1234.56 >20 PRINT USING "####.### ";X >30 PRINT USING "+####.### ";X >40 PRINT USING "X=######.## ";X >50 PRINT USING "+###.# ";1234.5 >60 END >RUN 1234.560 +1234.560 X=1234.56 %+1234.5
例：3	**例：**"&　　&"**フォーマットのスペースが** 6 **文字（**&**を含めて** 8 **文字（**8 **バイト））出力指定時** >10 A$="123"+SPACE$(8) >20 B$="1234567890" >30 PRINT USING "&　　&";A$, "abc" >40 PRINT USING "&　　&";B$, "abc" >RUN 123　　abc 12345678abc
参照：	LPRINT, WAIT

PWORD	
構文：	PWORD <**文字列式**>
機能：	**ステートメント。**BASIC **プログラムエリア（**PGEN **コマンドによってプログラム番号を選択）をパスワードにて読出禁止にできます。**
説明：	**<文字列式> は、最大** 32 **文字の、任意の有効な 文字列です。名前が** 32 **文字より長い場合、以降が切り捨てられます。パスワードは、大文字/小文字を使用できます。** PWORD **命令は、**BASIC **プログラムの最初の行に記述する必要があります。プログラムの先頭にないと、パスワードが有効になりません。**
参考：	PWORD **の設定は、プログラム入力の最後に行ってください。最初に入力するとパスワードを要求します。** BASIC **プログラムが** PWORD **命令により保護されている場合、入力されるコマンドに対してパスワードが要求されます。パスワードが正しくない場合、**”PASSWORD INCORRECT” **（パスワード不正エラー**, **エラーコード：**0069**）が発生します。パスワードが正しい場合、コマンドは通常どおり実行されます。** PWORD **命令でパスワードを設定してあっても、**NEW **命令からはプロテクトされません。**
例：	> 1 PWORD ”ABRACADABRA” > 20 PRINT ”PROTECTED BY A PASSWORD” > RUN > PROTECTED BY A PASSWORD > LIST PASSWORD : *************** 1 PWORD ”ABRACADABRA” 20 PRINT ”PROTECTED BY A PASSWORD” >
参照：	PNAME, PMEM

RANDOM	
構文：	RANDOM [<**元値**>]
機能：	**ステートメント。元値より乱数の系列を変更します。**
説明：	**<元値> は、乱数の元値として使用される数値式です。有効な範囲は、-**32768**～**32767 **です。** RANDOM **命令が<元値>なしで使用されると、はプロンプトにより元値の入力を促します。** **乱数の系列が変更されると、**BASIC **プログラムが実行されるたびに、**RND **関数によって異なる乱数シーケンスが実行されます。乱数発生パターンを変えるには** RANDOM **命令によって、乱数の系列を変更してください。**
例：	> 10 RANDOM 654 > 20 PRINT RND > RUN .234831 > PRINT RND .969226 > RUN .234831
参照：	RND

READ	
構文：	READ <変数> { , <変数> }
機能：	ステートメント。DATA 命令によって定義された定数を変数に取り込みます。
説明：	<変数> は、任意の有効な 数値または文字変数名です。 変数の型、および順序は、DATA 命令で指定した、定数の型および順序と一致していることが必要です。
参考：	READ 命令は常に DATA 命令とセットで使用します。対応する DATA 命令が存在しないか、または DATA 命令のすべての定数が READ 命令により読み出された場合、”OUT OF DATA”（データエラー，エラーコード：B004）が発生します。 READ 命令で指定する変数は、任意の型であることが可能ですが、この型は、対応する DATA 命令の定数の型と一致する必要があります。READ 命令の変数の型が、DATA 命令の定数の型と一致しない場合、”SYNTAX ERROR IN LINE xxxxx”（構文エラー，エラーコード：B002）が発生します。 READ 命令は、数値定数を、文字変数または数値変数に読み出すことができます。しかし、文字定数は、常に文字変数に読み出す必要があります。 DATA 命令が複数ある場合、それらを一連の定数とみなして、順番に定数を読み出していきます。 RESTORE 命令が実行された場合、最初の DATA 命令から再び読み出していきます。 READ 命令によってすべての定数が読み出され、かつ RESTORE 命令が実行されなかった場合、次の READ 命令で、”OUT OF DATA”（データエラー，エラーコード：B004）が発生します。
例：	> 10 DATA 10, ”HELLO”, 1.6 ”WORLD” > 20 READ A, B$, C, D$ > 30 PRINT B$+” ”+D$ > 40 PRINT A+C > RUN HELLO WORLD 11.6
参照：	DATA, RESTORE

REM	
構文：	REM [<コメント文字列>]
機能：	ステートメント。プログラム中にコメント（非実行文字列）を挿入します。
説明：	<コメント文字列> は、引用符で囲む必要のない、自由に定義可能な非実行文字列です。これは、最長行（行番号、スペース含め最大 255 文字）まで可能です。REM 命令は、プログラムを理解しやすくするために使用してください。 REM 命令の後のすべての文字列はプログラムの実行上無視されます。 REM とコメント文字列の間に空白を挿入する必要はありません。
参考：	コロン（：）の後にＲＥＭ命令を記述して、行の終わりコメントを追加することができます。コロン（：）の後に、引用符文字（ ’ ）を使用して REM の代わりにすることもできます。 REM を DATA 命令の中で使用すると、文字定数とみなされるため、DATA 命令の中では使用できません。
例：	> 10 REM TEST PROGRAM
参照：	

RENUM	
構文：	RENUM [<**新しい行番号**>] [, <**古い行番号**>] [, <**増分**>]
機能：	**コマンド。プログラム行番号を付け直します。**
説明：	**<新しい行番号> は、新しいシーケンスで使用される最初の行番号です。これは、**1〜65535**の範囲にある、任意の有効な行番号です。省略すると、デフォルト値** 10 **をとることになります。** **<古い行番号> は、そこから番号の付け直しを開始する、現在のプログラム内の最初の行番号です。省略すると、これはプログラムの最初の行にデフォルト設定されます。** **<増分> は、新しいシーケンスで使用する増分です。これは、**1〜65535 ***の範囲***にある、任意の有効な整数です。省略すると、デフォルト値** 10 **をとることになります。**
参考：	RENUM **命令はまた、**GOTO**、**GOSUB**、**IF...THEN...ELSE **および** RESTORE **命令の、プログラムで使用する行番号参照子を変更します。** RENUM **命令は無効なプログラム番号の原因となる場合、実行されません。**”ILLEGAL FUNCTION CALL”**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。**
例：	> 1 REM FIRST LINE > 2 REM SECOND LINE > 3 GOTO 3 > RENUM 100, 1 > LIST 100 REM FIRST LINE 110 REM SECOND LINE 120 GOTO 120 >
参照：	

RESTORE	
構文：	RESTORE [<行番号> | <ラベル>]
機能：	**ステートメント。**READ **命令が読み始める** DATA **命令の行番号を指定します。**
説明：	**<行番号> は、**DATA **命令が存在する 行番号です。有効な範囲は、**1 **～**65535 **です。** **<ラベル> は、行番号の代わりに使用されます。** **<行番号> または <ラベル> を省略すると、**READ **命令は、プログラム内の 最初の** DATA **命令の定数を読み込みます。** **行番号またはラベルを省略したときは、プログラム中の最初の** DATA **命令を指定したことになります。**
参考：	**指定された <行番号> が** BASIC **プログラム内に存在しない場合、**"UNDEFINED LINE NUMBER"**（行番号未定義エラー，エラーコード：**B008**）が発生します。**
例：	> 10 DATA 10, "HELLO", 1.6, "WORLD" > 20 READ A, B$, C, D$ > 30 PRINT B$+" "+D$ > 40 PRINT A+C > 50 RESTORE 10 > 60 READ A, B$, C, D$ > 70 PRINT A+C > 80 PRINT B$+" "+D$ HELLO WORLD 11.6 11.6 HELLO WORLD
参照：	DATA, READ

RESUME	
構文：	RESUME [<**行番号**> | <NEXT >]
機能：	**ステートメント。エラー処理終了後、プログラム実行を再開します。**
説明：	<**行番号**> **は、**BASIC **プログラムの任意の行番号で、ここから実行が再開します。有効な範囲は、**0～65535 **です。** **エラー処理ルーチンからの復帰は、**RETURN **命令でなく、この** RESUME **命令を使用します。** **行番号を** 0 **に指定または省略すると、エラーの発生した行を再度実行します。** NEXT **を指定すると、エラーの発生した行の次の行から実行が再開されます。**
例：	> 10 B = 6 > 20 ON ERROR GOTO 80 > 30 FOR A = 1 TO 10 > 40 PRINT A, B, A/B > 50 B = B-1 > 60 NEXT A > 70 END > 80 PRINT ”ERROR HAS OCCURRED” > 90 B = 5 > 100 RESUME 1 6 .166667 2 5 .4 3 4 .75 4 3 1.33333 5 2 2.5 6 1 6 ERROR HAS OCCURRED 7 5 1.4 8 4 2 9 3 3 10 2 5 >
参照：	ERROR, ON ERROR

RIGHT$	
構文：	RIGHT$(<**文字列式**>,<**数値式**>)
機能：	**関数。文字列の右端から指定された文字数の文字列を取り出します。**
説明：	<**文字列式**> **は、取り出し対象の文字列です。** <**数値式**> **は、取り出す文字数です。有効範囲は** 1～255 **です。** **取り出される文字数が** 0 **の場合、空白文字列が返されます。** **取り出される文字数が取り出し対象の文字数より大きい場合、取り出し対象の文字列全体が返されます。**
例：	> 10 A$ = RIGHT$(”HELLO WORLD”, 5) > 20 PRINT A$ > RUN WORLD
参照：	LEFT$, MID$

RND	
構文：	RND [<数値式>]
機能：	関数。0 と１の間の乱数を返します。
説明：	<数値式> の有効範囲は、-2147483648～2147483647 です。 <数値式> が負の場合、乱数発生器の最初の番号が返されます。 <数値式> を省略するかまたは正の場合、乱数発生器の次の乱数が返されます。 <数値式> が 0 の場合、乱数発生器の最後の乱数が繰り返されます。 注：乱数発生器は、RANDOM 関数により変更できます。
例：	> 10 RANDOM 654 > 20 PRINT RND > RUN .234831 > PRINT RND .969226 > RUN .234831
参照：	RANDOM

ROMLOAD	
構文：	ROMLOAD
機能：	コマンド。フラッシュ ROM 内の BASIC プログラムをユーザメモリ（RAM）に読み出します。
説明：	フラッシュ ROM 内の 4 つのすべての BASIC プログラムを、フラッシュ ROM からユーザメモリ（RAM）に読み出します。
参考：	読み出し中の注意事項については、7-1 項「プログラミングの手順」を参照してください。 プログラム中に使用すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。 ROMSAVE していないフラッシュ ROM から ROMLOAD しようとすると”DATA TRANSFER CHECK SUM ERROR”（データ転送エラー，エラーコード：0070）が発生します。
例：	> 10 REM THIS PROGRAM WILL BE SAVED > ROMSAVE > 10 REM THIS WILL BE OVERWRITTEN > LIST 10 REM THIS WILL BE OVERWRITTEN > ROMLOAD > LIST 10 REM THIS PROGRAM WILL BE SAVED >
参照：	ROMSAVE, ROMVERIFY

ROMSAVE	
構文：	ROMSAVE
機能：	コマンド。ユーザメモリ（RAM）内の BASIC プログラムをフラッシュ ROM に保存し（書き込み）ます。
説明：	ユーザメモリ（RAM）内の 4 つのすべての BASIC プログラムを、ユーザメモリ（RAM）からフラッシュ ROM に保存し（書き込み）ます。
参考：	書き込み中の注意事項については、7-1 項「プログラミングの手順」を参照してください。 ROMSAVE 中は、ASCII ビジーフラグが ON になります。また、ROMSAVE 中（フラッシュ ROM への書込み中）は、ポートからの受信処理、PC とのインタフェース処理（I/O リフレッシュなど）は行われません。 プログラム中に使用すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。
例：	> 10 REM THIS PROGRAM WILL BE SAVED > ROMSAVE > 10 REM THIS WILL BE OVERWRITTEN > LIST 10 REM THIS WILL BE OVERWITTEN > ROMLOAD > LIST 10 REM THIS PROGRAM WILL BE SAVED >
参照：	ROMLOAD, ROMVERIFY

ROMVERIFY	
構文：	ROMVERIFY
機能：	コマンド。フラッシュ ROM 内のプログラムと、RAM 内のプログラムとを照合します。
説明：	フラッシュ ROM 内と RAM 内の すべての（4 つの）BASIC プログラム を 比較します。 内容が一致しない場合、”VERIFY ERROR”（照合エラー，エラーコード：0066）が発生します。内容が一致している場合、出力はなく、ASCII ユニットはコマンドモードに戻ります。
参考：	プログラム中に使用すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー，エラーコード：B005）が発生します。
例：	> 10 REM THIS PROGRAM WILL BE SAVED > ROMSAVE > VERIFY > 10 REM THIS WILL BE OVERWRITTEN > VERIFY VERIFY ERROR > ROMLOAD > VERIFY > LIST 10 REM THIS PROGRAM WILL BE SAVED
参照：	ROMLOAD, ROMSAVE

RTS	
構文：	RTS #<ポート式> | SET RESET |
機能：	ステートメント。指定されたポートの RS 信号線をセット（ON）またはリセット（OFF）します。
説明：	<ポート式> は、1〜3 の範囲の整数を返す式です。 SET を指定すると、RS 信号線が ON になります。RESET を指定すると、RS 信号線が OFF になります。ポートは OPEN 状態でないと使用できません。
参考：	1. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 2. ASC11 にポート#3 を指定するか、または形 C200H-ASC21 にポート#2（RS-422A/485）またはポート#3 を指定すると、”ILLGEAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	RTS #2 SET
参照：	CTS, DSR, DTR

RUN	
構文：	RUN [<開始行番号>] [TO <最終行番号>]
機能：	コマンド。プログラム実行を開始します。
説明：	<開始行番号> は、1〜65535 の範囲の任意の有効な行番号です。省略すると、最初の行から実行します。<開始行番号> を指定すると、指定行番号から実行します。 <最終行番号> 1〜65535 の範囲の任意の有効な行番号です。省略すると、最終の行まで実行します。<最終行番号> を指定すると、プログラムは指定行番号の実行前で停止（ブレイク）します。
参考：	プログラム内に存在しない行番号を指定した場合、”UNDEFINED LINE NUMBER”（行番号未定義エラー, エラーコード：B008）が発生します。 RUN コマンドは、すべての変数をクリアして、プログラムの実行前に開いていたポートを閉じます。RUN コマンドを実行するときに、ユニット前面のスタート/ストップスイッチがすでにストップに設定されていると、プログラムは実行できません。 実行中のプログラムは、ターミナルから [Ctrl] + [X] キーを押すか、またはユニット前面のスタート/ストップスイッチをストップに設定すると、停止（ブレイク）します。または、プログラム内の END 命令または STOP 命令により、停止（ブレイク）します。 ・プログラミング直後に RUN コマンドを実行すると、”Please wait compiling.....”が表示され、RUN コンパイル実行中であることを示します。 ・RUN コマンドの実行からプログラムがスタートするまでは、BASIC プログラムのコンパイル、ガベージコレクションの所要時間により、長くかかることがあります。 ・<開始行番号>、<最終行番号>にプログラム中にない行番号を指定すると、”UNDEFINED LINE NUMBER”（行番号未定義エラー, エラーコード：B008）となります。 ・RUN コンパイル時間は、53Kbyte（約 1300 行）のプログラムで約 10 秒かかります。また、プログラム中で使用している命令により、この時間は変わりますので、ご注意ください。 ・プログラム編集を繰り返した後で RUN した場合、ASCII ユニット内のメモリ状態が複雑になっていることがありますので、コンパイルに数分かかる場合があります。
例：	RUN
参照：	END, STOP

SAVE	
構文：	SAVE #<ポート式> [, ”TERM ：[<通信条件> [, <データ長> [, <パリティ> [, <ストップビット> [, CS [, RS [, DS [, XN]]]]]]]] ”]
機能：	コマンド。ASCII ユニットのユーザメモリ（RAM）エリア内の BASIC プログラムをターミナルに読み出します。
説明：	<ポート式> は、1〜3 の範囲内の整数を返す式です。 [, ”TERM >：[<通信条件> [, <データ長> [, <パリティ> [, <ストップビット> [, CS[, RS [, DS [, XN]]]]]]] ”] の説明については、OPEN 命令を参照してください。 SAVE 命令実行中は、SAVE 命令で指定した条件（デバイス記号・通信条件・伝送制御信号の動作）が有効になります。SAVE 命令実行後、もとの設定状態 に戻ります。
参考：	1. ASCII ユニットに格納されている BASIC プログラムをターミナルに読み出す手順は、ポートに接続されているターミナルモードが VT100 か FIT10 かによって異なります。いずれの場合でも、ターミナルからの入力は、転送中は無効になります。BASIC プログラムを保存する場合の詳細については、7-1 項「プログラミングの手順」を参照してください。 2. ポート#3 は、形 C200H-ASC31 のターミナルポートに対応しています。 3. 形 C200H-ASC11 または形 C200H-ASC21 でポート#3 を指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。
例：	> SAVE #1, “TERM：9600, 8, N, 2, CS_OFF, RS_OFF, DS_OFF, XN_ON”
参照：	LOAD

SEARCH	
構文：	SEARCH(<整数型配列変数>, <数値式 1> [, <数値式 2>] [, <数値式 3>])
機能：	関数。指定配列変数の中に、指定数値があるもっとも小さい添字番号を返します。
説明：	<整数型配列変数> は、検索される整数型の 1 次元の配列変数です。 <数値式 1> は、指定配列変数の中で探したい数値式（整数）です。これが整数でない場合、小数点以下は四捨五入されます。 <数値式 2> は、指定配列変数の中で検索を始める開始添字番号（整数）です。省略されると、開始添字番号は、配列内の先頭と見なされます。なお、OPTION BASE 命令によって指定する配列変数の始めの添字の数が 1 で、かつ、この <数値式 2> を 0 と指定すると、検索は添字 1 から開始します。 <数値式 3> は、順に検索するときの添字の増分を示します。例えば、2 を指定すると、+2 した添字番号の配列変数に対して、順に検索します。省略すると、値は１と見なされます。 指定配列変数に、指定数値がない場合、-1 が返されます。
例：	> 10 DIM ARRINT%(100) > 20 FOR I = 1 TO 100 > 30 ARRINT%(I) = I MOD 10 > 40 NEXT I > 50 A = SEARCH(ARRINT%, 7, 12, 1) > 60 PRINT ”7 IS FIRST FOUND AT POSITION ”; A > RUN 7 IS FIRST FOUND AT POSITION 17
参照：	

SGN	
構文：	SGN(<**数値式**>)
機能：	**関数。数値式の符号を返します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **SGN は、数値式が正の場合** 1 **を、また数値式が負の場合**-1 **を返します。数値式が** 0 **の場合、SGN は** 0 **を返します。**
例：	> 10 INPUT A% > 20 IF SGN(A%) = 1 THEN PRINT ”POSITIVE” > 30 IF SGN(A%) = -1 THEN PRINT ”NEGATIVE” > 40 IF SGN(A%) = 0 THEN PRINT ”ZERO” > 50 GOTO 10 > RUN ? 1 <CR> POSITIVE ? -34 <CR> NEGATIVE ? 0 <CR> ZERO
参照：	

SIN	
構文：	SIN(<**数値式**>)
機能：	**関数。数値式の正弦 を計算します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。入力（引数）の型が、倍精度実数型の場合、出力（戻り）の型は倍精度実数型になります。**
参考：	**入力（引数）はラジアン単位で指定してください。**
例：	> 10 A = 0.4 > 20 PRINT SIN(A) > RUN .389418
参照：	COS, TAN

SPACE$	
構文：	SPACE$(<**数値式**>)
機能：	**関数。指定した数の空白文字列を返します。**
説明：	**<数値式> は、**0**～**255 **の範囲の、任意の整数値です。非整数値は、小数点以下を切り捨てられ整数になります。** **数値式が** 0 **の場合、空白文字列が返されます。**
例：	> 10 A$ = ”HELLO” + SPACE$(5) + ”WORLD” > 20 PRINT A$ > RUN HELLO　　WORLD
参照：	SPC

SPC	
構文：	SPC(<**数値式**>)
機能：	**関数。**PRINT **または** LPRINT **命令内で使用され、空白を返します。**
説明：	**<数値式> は、**0**〜**255 **の範囲の整数値を返す式です。指定数の空白を返します。** **非整数値は、小数点以下切り捨てで丸められます。式が負の場合、**0 **と見なされます。** PRINT **命令または** LPRINT **命令内で、空白を出力するために使用されます。** **空白がターミナルの列幅を超える場合、出力は次の行に折り返されます。**
参考：	SPC **は、**PRINT **命令または** LPRINT **命令とセットで使用する必要があります。これは単独の命令では使用できません。** **数値式が** 255 **より大きい場合、**”ILLEGAL FUNCTION CALL”**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。**
例：	> 10 PRINT ”HELLO” ; ”WORLD” > 20 PRINT ”HELLO”;SPC(5) ; ”WORLD” > RUN HELLOWORLD HELLO　　WORLD
参照：	TAB

SQR	
構文：	SQR(<**数値式**>)
機能：	**関数。数値式の平方根を計算します。**
説明：	**<数値式> は、任意の整数、単精度実数式または倍精度実数式です。** **引数が整数型または単精度実数型である場合、戻り型は単精度実数です。引数が倍精度実数型である場合、戻り型もまた倍精度実数です。**
参考：	**<数値式> が負の場合、**”ILLEGAL FUNCTION CALL”**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。**
例：	> 10 A = 4 > 20 PRINT SQR(A) > RUN 2
参照：	

STEP	
構文：	STEP [<**数値式**>]
機能：	**コマンド。停止（ブレイク）位置から、指定行数分を実行します。**
説明：	**<数値式> は、実行するプログラム行数を指定します。これは、*0*〜*255* の*範囲*の整数値を返す式です。省略すると、値** 1 **を返します。**
参考：	STEP **コマンドは、停止（ブレイク）位置から、指定行数を実行した後、停止（ブレイク）します。継続起動させる場合、別の** STEP **コマンドまたは** CONT **コマンドを実行します。** **停止（ブレイク）位置がない場合（**END、LOAD **または** EDIT **コマンドの後で** STEP **を実行する場合など）、**STEP **コマンドによって、プログラムは最初の行から、指定行数分実行します。**
例：	> 10 PRINT ”STARTED” > 20 STOP > 30 PRINT ”CONTINUING” > 40 PRINT ”CONTINUING” > 50 PRINT ”FINISHED” > RUN STARTED EXECUTED BREAK IN LINE 20 > STEP 2 CONTINUING CONTINUING BREAK IN LINE 40 > CONT FINISHED >
参照：	CONT, RUN

STOP	
構文：	STOP
機能：	**ステートメント。**BASIC **プログラムの実行を停止し、コマンドモードに戻ります。**
説明：	STOP **命令を実行すると、**BASIC **プログラムの実行は停止（ブレイク）し、「**BREAK EXECUTED IN LINE <**行番号**> **」というメッセージがターミナルに表示されます。**
参考：	CONT **コマンドまたは** STEP **コマンドを使って、停止（ブレイク）している現在行から実行を再開させることができます。** **変数エリアはクリアされず、また開いているポートは閉じません。**
例：	STOP
参照：	CONT, RUN, STEP

STR$	
構文：	STR$(<**数値式**>)
機能：	**関数。数値式を文字列に変換します。**
説明：	**<数値式> は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **<数値式> が正の場合、接頭辞として空白が文字列の前に付けられます。** **<数値式> が負の場合、接頭辞としてマイナス符号（-）が、文字列の前に付けられます。** **数値を文字として扱ったほうが、プログラミングしやすいことがあります。その場合、この関数を使用して、数値を文字列に変換します。**
参考：	**この逆の文字列を数値に変換する関数が、VAL 関数です。**
例：	> 10 FOR A = 1 TO 5 > 20 A$ = STR$(A) > 30 PRINT A$ > 40 NEXT A > RUN 1 2 3 4 5
参照：	VAL

STRING$	
構文：	STRING$(<**数値式** 1>, |<**数値式** 2>|) |<**文字列式**>|
機能：	**関数。数値式１で指定した個数の文字列の繰り返しを返します。**
説明：	**<数値式** 1> **は、文字列の個数を示します。任意の整数、倍長整数、単精度実数、倍精度実数です。非整数値式は小数点以下は四捨五入されます。有効範囲：は** 0～255**。この値が** 0 **であると、空の文字列が返されます。** **<数値式** 2> **は、繰り返し対象の文字列（**1 **文字）の** ASCII **文字コードです。任意の整数、倍長整数、単精度実数、倍精度実数です。有効範囲：**0～255**。** **<文字列式> は、繰り返し対象の文字列（**1 **文字）です。複数の文字の場合、最初の** 1 **文字目が対象となります。** **同じ文字を複数個並べた文字列をつくるときに使用します。**
例：	> 10 FOR A = 1 TO 5 > 20 A$ = STRING$(A, ”*****”) > 30 PRINT A$ > 40 NEXT A > RUN * ** *** **** *****
参照：	SPACE$

TAB	
構文：	TAB(<**数値式**>)
機能：	**関数。**PRINT **または** LPRINT **命令でターミナルディスプレイ上の指定位置にカーソルを移動します。**
説明：	<**数値式**> **は、**1**～**255 **の有効範囲の、任意の整数値式です。非整数値式は丸められます。数値式の値が１行の表示文字数を超えた場合、は、１行表示文字数で割った余りとなります。** **現在のカーソル位置が指定されたタブ位置をすでに超えている場合、カーソルは次の行の指定された位置に移動します。**
例：	> 10 PRINT ”H”; TAB(0) ; ”W” > 20 PRINT ”H”; TAB(1) ; ”W” > 30 PRINT ”H”; TAB(2) ; ”W” > 40 PRINT ”H”; TAB(3) ; ”W” > 50 PRINT ”HELL”; TAB(4) ;”O” > RUN H W HW H W H　W HELLO
参照：	SPC

TAN	
構文：	TAN(<**数値式**>)
機能：	**関数。数値式の正接（タンジェント）を計算します。**
説明：	<**数値式**> **は、整数、倍長整数、単精度実数、倍精度実数のいずれかの数値式です。** **入力（引数）の型が整数型または倍長整数型または単精度実数型の場合、出力（戻り）の型は単精度実数型になります。入力（引数）の型が、倍精度実数型の場合、出力（戻り）の型は倍精度実数型になります。** **注：入力（引数）はラジアン単位で指定してください。**
例：	> 10 A = 0.4 > 20 PRINT TAN(A) > RUN 0.422793
参照：	SIN, COS

TIME$	
構文：	TIME$ [= <**時刻文字列式**>]
機能：	**特殊関数。内部クロックの時刻を返すか、または時刻を設定します。**
説明：	<**時刻文字列式**> **は、「**hh:mm:ss**」形式です。**hh **が時で、有効範囲は** 00〜23 **です。**mm **は、分で、有効範囲は** 00〜59 **です。**ss **は秒で、有効範囲は** 00〜59 **です。**
参考：	<**時刻文字列式**> **が有効時刻でない場合、**"ILLEGAL FUNCTION CALL"**（関数コール不正エラー**, **エラーコード：**B005**）が発生します。** **一度、**TIME$ [= <**時刻文字列式**>] **で設定すれば、**TIME$**は、現在の時刻を返します。**
例：	TIME$ = "12:10:05"
参照：	DATE$, DAY

TIME$ ON/OFF/STOP	
構文：	TIME$ | ON OFF STOP |
機能：	**ステートメント。**ON TIME$**命令による時刻割込みを許可、禁止または停止します。**
説明：	ON **は、時刻割込みを許可（マスク解除）にします。この** TIME$ ON **命令から** TIME$ OFF **命令までを実行中に、時刻割込みが発生すると、定義された 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降時刻割込みは、無効になります。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくだけで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンを実行します。**
参考：	TIME$ ON/OFF/STOP **命令は、**ON TIME$**命令を実行した後にだけ実行可能です。**
例：	> 10 ON TIME$ = "08:00:00" GOSUB 100 > 20 TIME$ ON > 30 GOTO 30 > 100 PRINT "WAKE-UP!!!" > 110 RETURN
参照：	ON TIME$

TIMER ON/OFF/STOP	
構文：	TIMER | ON OFF STOP |
機能：	**ステートメント。タイマ割込みを許可、禁止または停止します。**
説明：	ON **は、タイマ割込みを許可（マスク解除）にします。この** TIMER ON **命令から** TIMER OFF **命令までを実行中にタイマがアップし、割込みが発生すると、定義された 割込みサブルーチンに分岐します。** OFF **は、割込みを禁止（マスク）にします。以降タイマ割込みは、無効となります。** STOP **は、割込みを停止（保留）にします。割込みが入っても、割込みが入ったことを記憶しておくだけで、処理分岐しません。**ON **により、割込みのマスクが解除されると、記憶していた割込みサブルーチンを実行します。**
参考：	TIMER ON/OFF/STOP **命令は、**ON TIMER **命令を実行した後にだけ実行可能です。**
例：	> 10 T = 0 > 20 ON TIMER 30 GOSUB 100 > 30 TIMER ON > 40 GOTO 40 > 100 T = T+3 > 110 PRINT T; "SECONDS HAVE ELAPSED SINCE TH START OF EXECUTION." > 120 RETURN
参照：	ON TIMER

TRACE	
構文：	TRACE
機能：	**コマンド。トレースバッファの内容をターミナルに表示します。**
説明：	TRON M **命令を実行すると、実行されたプログラムの行番号は、最大** 255 **行分のトレースバッファに格納されます。トレースバッファに格納されたトレース内容は、電源** OFF **後も、バッテリバックアップされます。したがって、必要時に、この** TRACE **コマンドにより、記憶されたトレース内容をターミナルに表示させることが可能です。** NEW **コマンド、**TROFF **コマンド を実行した場合、トレースバッファの内容もクリアされます。**
例：	> TRON M > 10 FOR A = 1 TO 5 > 20 PRINT A > 30 NEXT A > RUN 1 2 3 4 5 > TRACE [10] [20] [30] [10] [20] [30] [10] [20] [30] [10] [20] [30] [10] [20] [30] [10] [30]
参照：	TROFF, TRON

TROFF	
構文：	TROFF
機能：	**コマンド。以降の行番号のトレースをオフに（終了）します。**
説明：	**プログラム実行中、トレース用の行番号の出力を停止します。**
参考：	ASCII **ユニットがリセットされた場合、または電源を** OFF **にした場合、または** NEW **コマンドを実行した場合、行番号のトレースは自動的にオフ（終了）になります。**
例：	> 10 TORN > 20 FOR A = 1 TO 5 > 30 NEXT A > 40 TROFF > 50 FOR A = 1 TO 5 > 60 NEXT A > RUN [20] [30] [20] [30] [20] [30] [20] [30] [20] [30] [20] [40] >
参照：	TRACE, TRON

TRON	
構文：	TRON [M]
機能：	**コマンド。以降の行番号のトレースをオン（開始）にします。M オプションを指定すると、行番号がトレースバッファに格納されます。**
説明：	TRON **命令を実行すると、実行されたプログラムの行番号が変わるタイミングで、実行された順に、ターミナルに表示されます 。** M **オプションを指定すると、実行された行番号は、最大** 255 **行分のトレースバッファに格納されます。トレースバッファに格納されたトレース内容は、電源** OFF **後も、バッテリバックアップされます。したがって、必要時に、**TRACE **コマンドにより、記憶されたトレース内容をターミナルに表示させことが可能です。**
参考：	**行番号のトレースは、**TROFF **命令によりオフ（終了）にすることが可能です。** ASCII **ユニットがリセットされた場合、または電源を** OFF **にした場合、または** NEW **コマンドを実行した場合、行番号のトレースは自動的にオフ（終了）になります。**
例：	> 10 TORN > 20 FOR A = 1 TO 5 > 30 NEXT A > 40 TROFF > 50 FOR A = 1 TO 5 > 60 NEXT A > RUN [20] [30] [20] [30] [20] [30] [20] [30] [20] [30] [20] [40] >
参照：	TRACE, TROFF

VAL	
構文：	VAL(<**文字列式**>)
機能：	**関数。文字列を数値式に変換します。**
説明：	<**文字列式**> **は、数値の任意の文字表記です。** **文字列式は、**10 **進形式、**16 **進形式（**&H **を先頭に指定）、**8 **進形式（**&**または**&O **を先頭に指定）、単精度実数、倍精度実数が可能です。（指数形式の実数の場合、単精度実数の場合** E**、倍精度実数の場合** D **を用いて指定する必要があります。）** **数値式内の空白は無視されます。** **文字表記が数値を表記しない場合（文字列の最初が＋、－、**&**、.、数字でない場合）、**0 **が返されます。数値以外の文字が現れた場合、それ以降の文字は無視されます。** **空白が先行している場合は、変換の前に文字列から削除されます。**
参考：	**この逆の数値を文字列に変換する関数が、**STR$**関数です。**
例：	> 10 A$ = ”2” > 20 B$ = ”3” > 30 PRINT A$+B$ > 40 PRINT VAL(A$)+VAL(B$) > RUN 23 5 >
参照：	STR$

VARPTR	
構文：	VARPTR(<**変数**>)
機能：	**関数。変数の値が格納されているメモリ番地（アドレス）を返します。**
説明：	**<変数> は、割り当てられた値をすでに持っている 変数である必要があります。** **変数の値が格納されているユーザメモリエリアの番地（アドレス）が、10 進数で返されます。** **変数が配列変数の場合、配列のメモリブロックのアドレスの最上位が格納されているメモリ番地（アドレス）が、10 進数で返されます。**
参考：	
例：	> 10 A = 5 > 20 PRINT VARPTR(A) > RUN 12321
参照：	

WAIT	
構文：	WAIT <待ち時間> [, <行番号>]
機能：	ステートメント。WAIT 命令に続く特定命令の実行を監視し、その特定命令が監視時間を超えても実行終了しない場合、タイムアウトを発生させ、指定行番号、または監視対象命令のつぎの命令を実行します。
説明：	<待ち時間> は待ち時間を指定します。この時間を超えても、監視対象の命令が実行終了しない場合、タイムアウトが発生し、次の行または指定行に実行が移ります。<待ち時間> は「[<mm>:]<ss>.<t>」の形式をもちます。 <mm> は、分を指定します。有効範囲は 00〜59 です。この値はオプション（省略可）です。 <nn> は、秒を指定します。有効範囲は、00〜59 です。この値は必須です。 <t> は、1/10 秒を指定します。有効単位は、0〜9 です。この値は必須です。 <行番号> は、タイムアウトが発生した後実行を継続する行を指定します。これは、1〜65535 の範囲の 行番号です。指定された行番号が存在しない場合、”UNDEFINED LINE NUMBER ”（行番号未定義エラー, エラーコード：B008）が発生します。
参考：	WAIT 命令は、この命令の直後の命令を監視します。この直後の監視対象の命令が、指定時間内に実行を終了した場合、プログラムは通常どおり実行を継続します。そうでない場合（タイムアウトした場合）、監視対象の命令の実行は終了し、指定行番号がある場合、指定行番号へ分岐します。指定行番号を省略した場合、監視対象の次の命令を実行します。 WAIT 命令によって監視できる命令は、次のとおりです。 INPUT, NPUT$, LINE INPUT, PC READ, PC WRITE, PC QREAD, PCQWRITE, PRINT, LPRINT, FOR,および WHILE 不正な待ち時間を指定すると、”ILLEGAL FUNCTION CALL”（関数コール不正エラー, エラーコード：B005）が発生します。 指定された行番号が存在しない場合、”UNDEFINED LINE NUMBER”（行番号未定義エラー, エラーコード：B008）が発生します。 WAIT 命令が監視中に他の BASIC 割込みが発生するとその BASIC 割込みサブルーチンが実行されます。その間 WAIT 命令の監視タイマは一時停止しています。
例：	> 10 WAIT ”00:1.5”, 100 > 20 INPUT A$ > 30 PRINT ”HELLO ”;A$ > 40 END > 100 PRINT ”TOO LATE...” > 110 END > RUN ? TOO LATE...
参照：	

WATCH	
構文：	WATCH [SET<**変数**> | DEL<**変数**]
機能：	**コマンド。指定変数をモニタ対象として設定（**SET**）または削除（**DEL**）またはリスト表示します。プログラムが停止（ブレイク）した時、その時点での変数の値を知ることができます。**
説明：	**<変数> は、割り当てられた値をすでに持っている変数である必要があります。最大** 16 **個の変数を同時にモニタ可能です。<変数>に配列変数を指定することはできません。** SET **は、指定変数をモニタ対象としてセットします。** DEL **は、指定変数をモニタ対象から削除します。** SET **も** DEL **を指定しないと、セットしてあるモニタ対象の変数をすべてリスト表示します。**
参考：	WATCH **コマンドを入力しておくと、**BASIC **プログラムが停止（ブレイク）したときに、ターミナル上に、モニタ対象の変数のその時点の値が、変数＝□□と表示されます。** RUN **コンパイルを実行した後は、**WATCH **コマンドによる変数指定を再度行ってください。**
例：	> WATCH SET A$ > WATCH SET B > 10 A$ = ” ” > 20 FOR B = 1 TO 9 > 30 A$ = A$ + STR$(B) > 40 STOP > 50 NEXT B > RUN BREAK IN LINE 40 A$ = 1 B = 1 > CONT BREAK IN LINE 40 A$ = 1 2 B = 2 >
参照：	BRKPT, CONT, STEP, STOP

お願い　WATCH **では、配列変数は使用できません。**

WHILE/WEND	
構文：	WHILE <数値式> <プログラム> WEND
機能：	ステートメント。条件成立の間、WHILE～WEND 間のプログラム（複数行）を繰り返し実行します。条件が不成立になると、繰り返し処理を終了し、WEND 命令に続く行番号から実行します。
説明：	<数値式> の結果が真（0 以外）の間、WHILE～WEND 間のループを繰り返します。<数値式> の結果が偽（0 ）の場合、ループは終了し、WEND 命令に続く行番号から実行します。
参考：	WHILE/WEND は、FOR/NEXT での繰り返し回数指定の代わりに、繰り返し条件を指定したときに、使用します。例えば、X=0 になるまで、繰り返し演算処理をしたい場合で、繰り返し回数が、やってみないとわからない場合などに、使用します。 WHILE/WEND は常に対で使用する必要があります。WHILE と WEND が対応していないと、”MISMATCH:WHILE/WEND”（WHILE/WEND 不一致エラー, エラーコード：B024）が発生します。 WHILE/WEND 命令は、ネスティング可能です。ネスティング数の制限はありません（ただし、ユーザメモリの容量を超えないこと）。
例：	> 10 A = 0 > 20 WHILE A<10 > 30 PRINT A > 40 A = A+1 > 50 WEND
参照：	@IF, FOR

7−7　ユーザ定義関数

形C200H-ASC11/21/31では、ユーザ定義の関数を作成することができます。
・BASIC関数

■BASIC関数

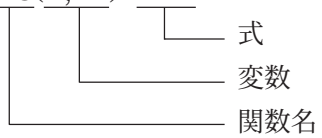
BASIC関数とは、ユーザがBASICで式を作成し、それをユーザ定義の関数として登録します。BASIC内で、他の関数と同様に、定義した関数名で呼び出して実行できる機能です。
関数名、入力変数、式を、BASICのDEF FN命令で宣言します。

●BASIC関数の使用手順

1)　DEF FN命令で、次の内容を宣言
・呼び出し時のBASIC関数名
・BASIC関数へ入力される変数名
・式

2)　BASIC関数名で呼び出し、実行
BASIC関数の定義(DEF FN命令による)
構文：DEF FN関数名　(変数名[, 変数名])＝式
意味：ユーザ作成のBASIC関数を定義(宣言)します。関数名(変数名の条件で自由)、関数へ入力する変数名(複数可)、式を指定します。

例：DEF FNABC(A, B)=A+B



●BASIC関数の呼び出し

通常の関数と同様に、関数名と、(　)内に変数を指定して、呼び出します。
例：C=FNABC(A, B)

お願い　**以下の点に注意しくてください。**
- **関数名には予約語から始まる文字列は使用できません。**
- **関数名の最後に型宣言文字($ % & ! #)を使用できません。**

7－8　デバッグ機能

■概要

プログラムを各種の方法で指定行にてブレイクさせ、そのときの変数内容をモニタできます。また、実行した行番号の経緯を知ることも可能です。

注：ブレイク機能のコマンドを使って割込サブルーチン中でもブレイクをかけることはできますが、メインから継続して実行を再開させることはできません。

●ブレイク

指定行でブレイク	**ターミナルからのコマンド入力による**	**複数の指定行でブレイク（最大255個）**	BRKPT**コマンド入力**	**プログラムエラー**
		1行または指定行数分だけ実行してブレイク	STEP**コマンド入力**	
		指定行まで実行	RUN　TO**行番号入力**	
	プログラム内の命令による	**指定行でブレイク**	STOP**命令**	

ブレイクした状態で ⇓　　**ブレイクした状態で** ⇓

●変数モニタ

変数の内容モニタ（最大16変数）	**ターミナルからのコマンド入力による**	WATCH**コマンド入力**

●実行行番号をトレース

プログラム内の命令による	TRON**命令**
トレースバッファに保存し、後でモニタ	TRON [M]**命令→** TRACE**コマンド入力**

以下に各デバッグ機能を説明します。

1)ブレイクポイント機能

2)ステップ機能

3)変数モニタ機能

4)トレース機能

■ブレイクポイント機能（BRKPTコマンドによる）

指定行番号にBRKPTコマンドによってブレイクポイントを設定すると、プログラムはその指定行番号の位置で止まります。行番号xxxで止まると、"Break executed in line xxx."と、ターミナルに表示されます。同時に、ASCIIユニットは、コマンドモードに切り替わります。ブレイクポイントは、最大255個設定することが可能です。ブレイクした後、その行からプログラムを再起動させるには、ターミナルからCONTコマンドを入力します。

構文：BRKPT （SET｜DEL）行番号

意味：ブレイクポイントの設定(SET)、削除(DEL)します。指定がないときは、現在セットされているブレイクポイントをリスト表示します。

例：

BRKPT　SET　200　行番号200にブレイクポイントをセットします。

BRKPT　DEL　100　行番号100のブレイクポイントを削除します。

BRKPT　　現在セットされているブレイクポイントのリストを表示します。

BRKPT　DEL　ALL　現在セットされているブレイクポイントをすべて削除します。

■ステップ機能(STEPコマンドによる)

STEPコマンド実行ごとに、1行ずつまたは指定数の行ずつプログラムを実行します。

構文：STEP ［数値式］

意味：指定数の行を実行します。指定がないときは、1行のみ実行します。

例：　STEP　5　　5行のプログラムを実行します。

　　STEP　　1行のプログラムを実行します。

■変数モニタ機能(WATCHコマンドによる)

WATCHコマンドによって、指定変数(配列変数を除く)の内容をモニタすることができます。WATCHコマンドを入力しておくと、STOP命令、STEPコマンド、ブレイクポントまたはプログラムエラーによってBASICプログラムがブレイクしたとき、"Break executed in line xxx."と表示された後、ターミナル上に、モニタ対象の変数の値が、"変数＝□□"と表示されます。表示の順は、WATCHコマンドの入力順です。同時に最大16変数をモニタ可能です。

構文：WATCH ［(SET｜DEL)　変数名

意味：指定変数をモニタ(WATCH)対象としてセット(SET)、または削除(DEL)します。指定がないときは、モニタ対象の変数をリスト表示します。同時に16変数をモニタ可能。変数名に配列変数は指定できません。

例：

WATCH　SET　A$　A$をモニタ(WATCH)対象としてセットします。

WATCH　DEL　B　Bをモニタ(WATCH)対象から削除します。

WATCH　　現在セットされているモニタ(WATCH)対象の変数のリストを表示します。

変数モニタ機能実行例：

WATCH　SET　B$　B$をモニタ(WATCH)対象としてセットします。

WATCH　SET　A　Aをモニタ(WATCH)対象としてセットします。

RUN後、行番号xxxでブレイクした場合、

BREAK　IN　LINExxx

B$= ”ABCD ”

A=3

参　考　**RUNコンパイルを実行した後は、WATCHコマンドによる変数指定を再度行ってください。**

■トレース機能（TRON命令、TRACEコマンドによる）

TRON命令を実行すると、実行されたプログラムの行番号が、実行された順に、ターミナルに表示されます。これにより、IF　GOTOやGOSUBサブルーチンによる分岐や、FOR～NEXTによる繰り返しなどの実行経緯を知ることができます。

Mオプション付きのTRON命令の場合、トレースデータをトレースバッファに保存しておき、後でTRACEコマンドを使用して、ターミナルに表示することができます。

トレースバッファは、最大255行を記憶できます。（255行を超える場合は、最初に実行した行番号からまた上書きしていきます。）

注：トレースバッファに格納されたトレース内容は、電源OFF後もバッテリバックアップされています。そのため、後で自由にTRACE命令で読み出すことが可能です。なお、トレースバッファのクリアの方法は、NEW、TROFF命令によります。

構文：TRON［M］

意味：以降の行番号のトレースを開始します。RUN後、ターミナルに、実行された行番号が［　］で囲まれて、連続表示されます。
オプション［M］付きの場合、トレースバッファにトレース内容を出力します。最大255行をトレース。

例：

TRON　　以降の行番号のトレースを開始します。

TRON［M］　以降の行番号のトレースを開始し、トレースバッファに格納します。

トレース機能実行例：

RUN

［10］［20］［30］［20］［40］・・・・行番号［10］［20］［30］［20］［40］の順で実行されました。

構文：TRACE

意味：トレースバッファ内の内容をターミナルに表示します。実行した行番号を［］で囲んで、実行順に連続表示します。

トレース機能実行例：

TRON M

TRACE

［20］［30］［20］［40］・・・・行番号［10］［20］［30］［20］［40］の順で実行されました。

参　考　実行行番号が変わったときに、トレースとして行番号がセットされます。よって、100 GOTO 100 のような場合には、トレースは[100]のみとなります。

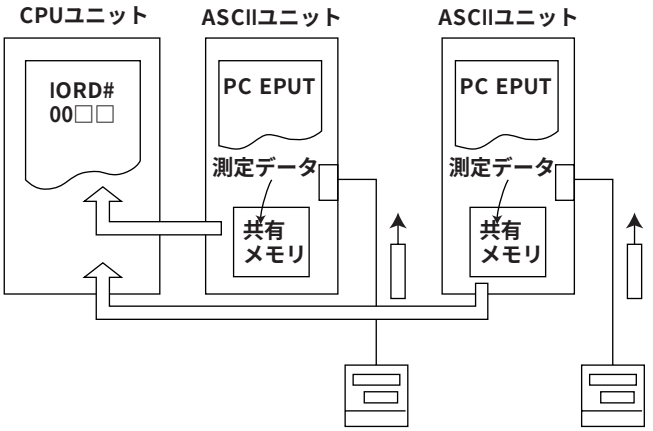
注：割込サブルーチンを実行した場合、プログラムのトレース内容をターミナルに表示させると、割込サブルーチンの行番号が重複したときは、“ DEVICE UNAVAILABLE IN PORT □ ”エラーが発生します。このような場合、Mオプション付きのTRON命令を実行して、トレースデータをトレースバッファに格納するようにしてください。

第8章
プログラム例

8−1　データ交換の応用例

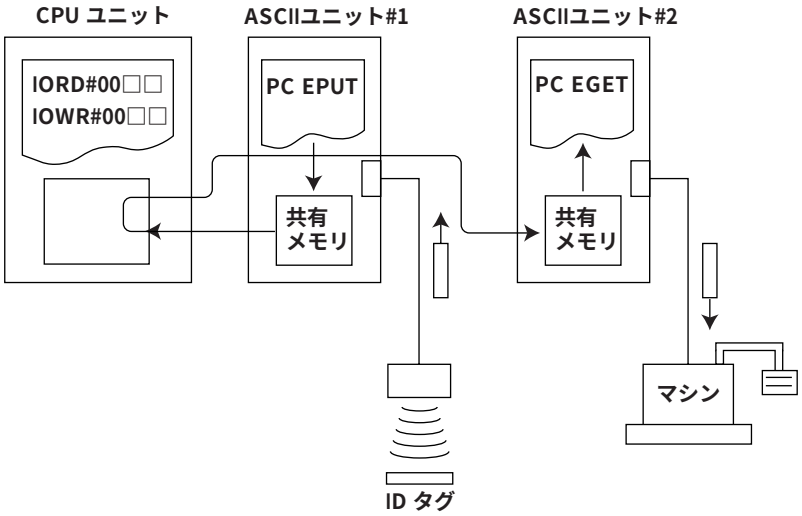
■CPUユニットとASCIIユニットが非同期で各々の処理をする例

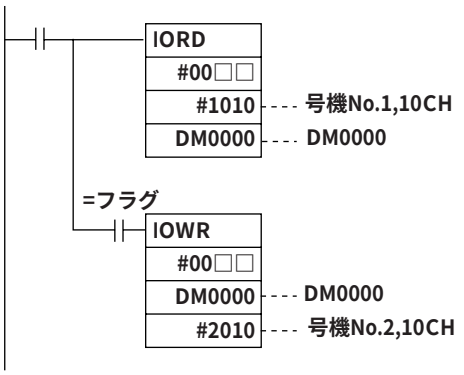
例. 監視システムにおいて、調節計からの測定データ(PV)をASCIIユニット内の共有メモリに都度セット。CPUユニットは、そのデータをモニタするため、必要に応じて、ASCIIユニットの共有メモリから読み出す。



ASCIIユニット	CPUユニット
PC EPUT 命令を使用して、→ 調節計からの測定データを ASCII ユニットの共有メモリへ高速リフレッシュ	必要に応じて、IORD(#00□□)命令を使用して、ASCIIユニットのもつそのときの測定データを読み出す

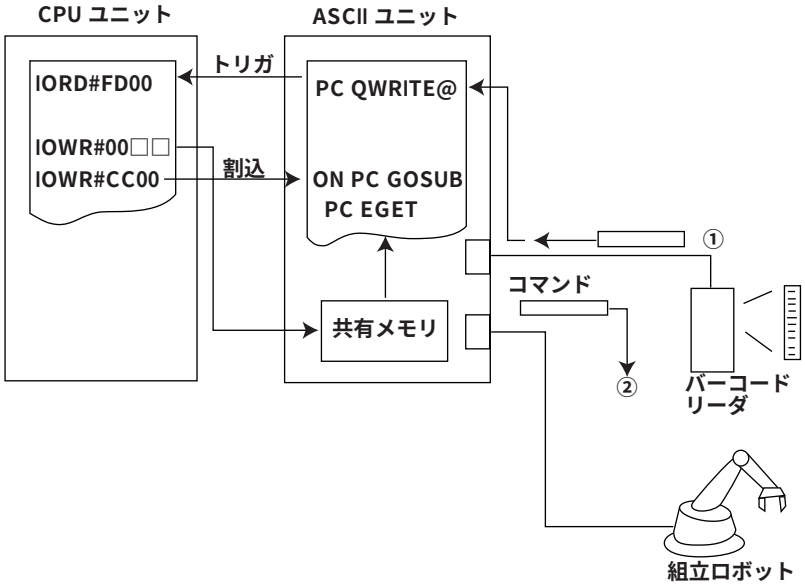
例. 複数のASCIIユニット間で、データ交換を行う。CPUユニットはその仲介をするのみ。



CPU ユニット	ASCII ユニット
 ASCII ユニット#1 の共有メモリから 読み出し、DM0000～ DM0009へ ASCII ユニット#2 の共有メモリへ ＤM0000～DM0009 から書き込み IORD(#00□□)とIOWR(#00□□)を続けて実行	PC EPUT命令（ASCIIユニット#1） を使用して、共有メモリへ書き込み。 PC EGET 命令（ASCII ユニット#2） を使用して、共有メモリから読み出し。

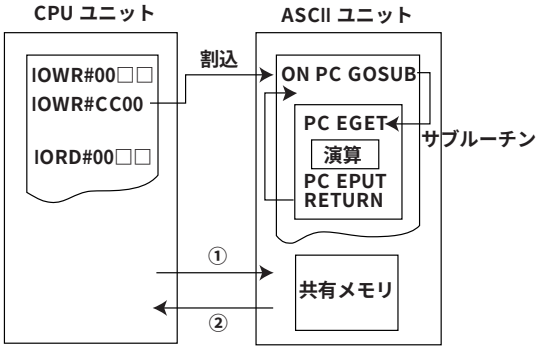
■CPUユニットとASCIIユニット間で高速のデータ交換をする例

例. ASCIIユニットが、通信インタフェースとして機能する場合
（ASCIIからのトリガ→CPUユニットの処理→CPUユニットからのトリガの順）
バーコードリーダからのデータに応じて、CPUユニットの処理により、組立ロボットを制御する。



	ASCII ユニット	CPU ユニット
①	バーコードリーダのデータを入手すると、 IORD命令実行要求フラグを出す。	IORD命令実行要求フラグを受けて、 IORD (#FD00) 命令でデータを入手
②	割込を受けて、割込サブルーチン内 のPC EGET命令で、共有メモリ内に 書き込まれているデータを読み出し、 組立ロボットにコマンドを発行する	ある処理の上、IOWR (#00□□) 命令で、ASCIIユニットの共有メモリへ データを書き込む、その上で、IOWR (#CC00) 命令で、ASCIIユニットへの割込

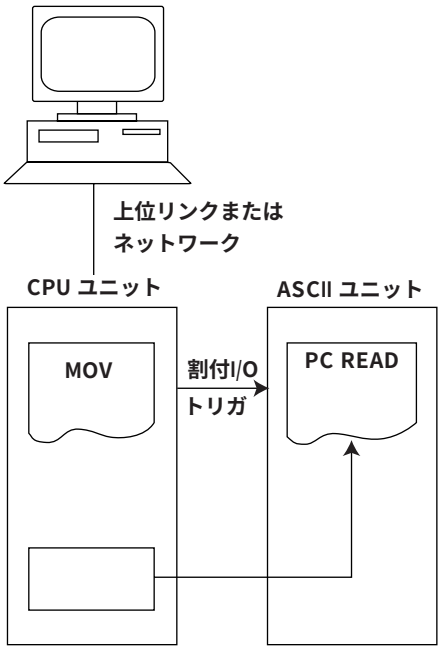
例. ASCIIユニットが特殊演算(BASIC)ユニットとして機能する場合
(CPUユニットからのトリガ→ASCIIユニットの演算→ASCIIユニットからのトリガの順)
ASCIIユニットに、CPUユニットが不得意とするような演算処理をまかせ、処理を分散することで、CPUユニット、ASCIIユニットともに処理を高速化する。



	CPU ユニット	ASCII ユニット
①	IOWR (#00□□) 命令で共有メモリにデータ書き込み、その上で、IOWR (#CC00) 命令でASCII ユニットへ割込 →	割込を受けて、割込サブルーチン内の PC EGET命令で、共有メモリ内に書き込まれているデータを読み出し、特殊演算を行います
②	割込サブルーチン終了で（割込サブルーチン完了フラグの変化を見て）IORD (#00□□) 命令で共有メモリから読み出し	← PC EPUT 命令で、演算結果を共有メモリに書き込み、割込サブルーチン終了

■CPUユニットとASCIIユニット間で、大量のデータを交換する例

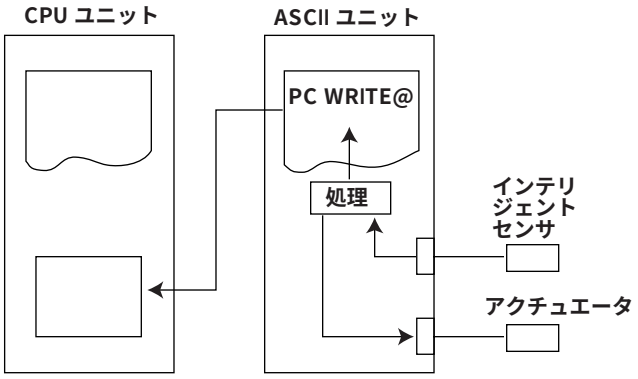
例. 上位パソコンからの生産指示データを各種マシンに対して、レシピ(設定／パラメータ)データとして与える。大量のデータ転送。さほど高速でなくてよい。



CPU ユニット	ASCII ユニット
割付 I/O を介して、ASCII ユニットに対してREAD要求 →	PC READ命令でCPU ユニットのI/Oメモリを読み出し、最大255CHを読み出せます。ただし、I/O リフレッシュ時、しかも1サイクルで最大127CH

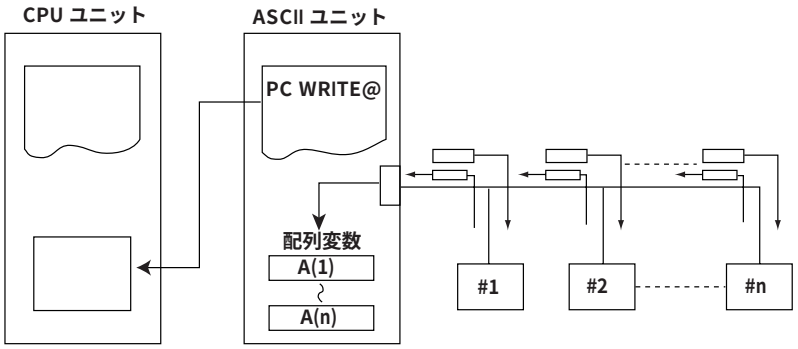
■ASCIIユニットが主体でCPUユニットへ伝える例

例. ASCIIユニットを制御ユニットとして使用し、状態モニタ用データをASCIIユニット側から手上げ的に伝える。



ASCII ユニット	CPU ユニット
センサからのデータをもとに処理し、アクチュエーターへ出力 必要に応じ、ASCIIユニットから、手上げ的に、PC WRITE@命令で、状態モニタ用データを書き込む	→ 状態モニタ用データを読み出す

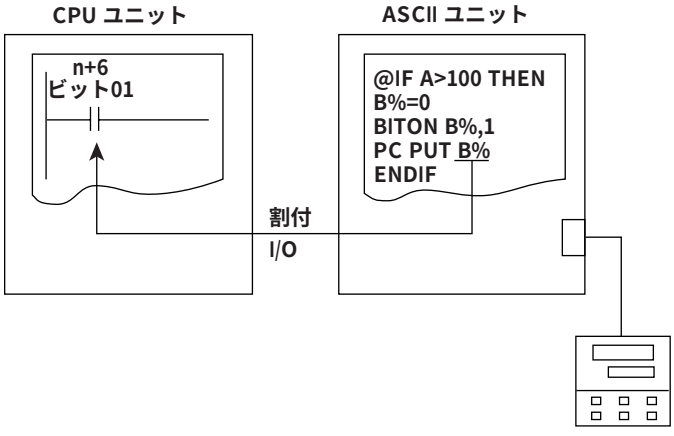
例. 複数台の外部機器をRS-422/485で接続し、ポーリングしてコマンドを発行し、各レスポンスデータを続けて配列変数に格納。必要に応じ、CPUユニットへ手上げ的に伝える。



ASCII ユニット	CPU ユニット
各レスポンスデータを配列変数に格納 必要に応じ、ASCIIユニットから手上げ的にPC WRITE@命令で、データを渡す	→ レスポンスデータを読み出す

■CPUユニットとASCIIユニット間で、ビットデータを変換する例

例. 温調器から入手した測定温度が、ある値以上でONとなる信号をCPUユニットへ伝える



ASCII ユニット	CPU ユニット
数値変数B%のビット1をONした上、PC PUT命令でBをn+6CHにセット	n+6CHビット01を読み出す

第9章
エラーとその処置

9−1　エラーメッセージ一覧

BASIC命令実行中にエラー(BASICエラー)が発生、またはシステムエラーが発生すると、CPUユニット内の割付I/OエリアのビットのCHにエラー種別、ビット0〜11にエラーコードが格納されます。同時に、ターミナルに、対応エラーメッセージが表示されます。(ただし、CPUユニットとのインタフェースから発生するエラー0080〜0099は、n+7CHへのエラーコードが格納されるだけで、エラーメッセージは表示されません。)

■割付I/Oエリア

割付I/O CH	ビット	内　容	値
n+7CH	0〜11	エラーコード	000〜255(BCD)
	12〜15	エラー種別	B：BASICエラー 0：システムエラー 1：ポート#1通信エラー 2：ポート#2通信エラー 3：ターミナルポート通信エラー （ASC31のみ） } 通信ポートエラー

■エラー種別／コード

□□□□

- 下位2桁：エラーコード000〜255
- 上位1桁：エラー種別
  - B：BASICエラー
  - 0：システムエラー
  - 1：ポート#1通信エラー
  - 2：ポート#2通信エラー
  - 3：ターミナルポート通信エラー（ASC31のみ）

  } 通信ポートエラー

■ターミナル上での表示例

SYNTAX ERROR IN LINE XXXX

└ 行番号ありで実行したときに表示されます。

注：ターミナル以外でポートが使用されているときに、BASICプログラムの実行が停止すると、ASCIIユニットはメッセージの表示待ちとなります。この場合、ターミナルポートにターミナルを接続して[Ctrl]+[X]を入力すると、各種メッセージがターミナルに表示されます。

エラー種別／コード・エラーメッセージおよび各処置は、次のとおりです。

■エラー種別／コード・エラーメッセージの内容および各処置の一覧

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
BASICエラー	B001	NEXT WITHOUT FOR	FOR命令が不足しています。	FOR～NEXTの組み合わせを確認してください。
	B002	SYNTAX ERROR IN LINE xxxxx	プログラムが文法にそっていません。変数名、ラベル名に予約語(コマンド名)と同じ文字から始まるものがあります。	プログラムを修正してください。予約語と異なる変数名、ラベル名としてください。
	B003	RETURN WITHOUT GOSUB	GOSUB命令の実行以前にRETURN命令に出会いました。	GOSUB～RETURNの構文を確認してください。
	B004	OUT OF DATA	READ命令で読むべきデータがありません。 READ命令の変数の個数・型と、DATA命令の定数の個数・型が一致していません。またはRESTORE命令が欠けています。	READ命令の変数の個数・型と、DATA命令の定数の個数・型を確認してください。
	B005	ILLEGAL FUNCTION CALL	ステートメントや関数の呼び方が間違っています。	ステートメントや関数の呼び方(構文)を確認してください。
	B006	OVERFLOW	数値が許される範囲を超えています。	数値の範囲、変数の型を確認してください。
	B007	OUT OF MEMORY	メモリ容量が不足しています。	ユーザメモリ容量が足りなくなりました。PINFコマンドなどで、メモリの使用状況を確認し、プログラムの見直しを行ってください。
	B008	UNDEFINED LINE NUMBER	行番号の指定が間違っています。	分岐先等の行番号を確認してください。
	B009	BAD SUBSCRIPT	配列変数の添字がDIM命令で指定した最大値の範囲外です。	DIM命令で指定した添字最大値内の値を使用してください。
	B010	DUPLICATE DEFINITION	配列変数、ユーザ定義BASIC関数を二重に定義しました。	配列変数、ユーザ定義BASIC関数で、二重定義したどちらかを変更してください。
	B011	DIVISION BY ZERO	0で除算を行っています。	除算の確認をし、プログラムを修正してください。
	B012	ILLEGAL DIRECT	コマンドモードでは実行できないステートメントを、コマンドモードで実行しようとしました。	BASICプログラムの中(プログラムモード)で実行してください。
	B013	TYPE MISMATCH	変数の型が合っていません。	変数の型を確認してください。
	B014	未使用	－	－

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
BASICエラー	B015	未使用	―	―
	B016	未使用	―	―
	B017	CANNOT CONTINUE	プログラムの実行が再開できません。	RUN コマンドで、プログラムを継続起動またははじめから起動させてください。
	B018	UNDEFINED USER FUNCTION	ユーザ関数が定義されていません。	ユーザ関数をDEF FNで定義してください。
	B019	NO RESUME	ON ERROR GOTO命令によるエラー割込サブルーチン内にRESUME命令が不足しています。	RESUME命令を追加してください。
	B020	RESUME WITHOUT ERROR	エラーが発生していないのに、RESUME命令を実行しました。	プログラムを確認してください。
	0021	未使用	―	―
	B022	MISSING OPERAND	必要なパラメータ(オペランド)がありません。	プログラムの構文を確認してください。
	B023	FOR WITHOUT NEXT	NEXT命令が不足しています。プログラム構造上で、WHILE命令の不足で発生することもあります。	FOR～NEXTあるいはWHILE～WENDの組み合わせを確認してください。
	B024	MISMATCH: WHILE/WEND	WEND、またはWHILE命令が不足しています。	WEND、またはWHILE命令を追加してください。
	B025	UNDEFINED SYMBOL	文字が使用可能な範囲外です。または、飛び先のラベルがありません。	文字を使用可能範囲にしてください。または、飛び先のラベルを確認してください。
	B026	INVALID EXPRESSION	不正な式を使用しています。	式を修正してください。
	B027	INVALID STATEMENT	不正なステートメントを使用しています。	ステートメントを修正してください。
	B028	未使用	―	―
	B029	MISMATCH: @IF/ENDIF	@IFとENDIFが対応していません。	@IFとENDIFの対応を確認してください。
システムエラー	0030	FATAL ERROR: BUS ERROR	ASCIIユニットが正常に動作できない状態が発生しています。	電源をON／OFFしてください。それでも、エラーが出る場合は、ユニットを交換してください。
	0031	FATAL ERROR: ADDRESS ERROR		
	0032	FATAL ERROR: PRIVILEGE VIOLATION		

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
システムエラー	0033	FATAL ERROR: UNINITIALISED INTERRUPT	システムエラー	電源をON／OFFしてください。それでも、エラーが出る場合は、ユニットを交換してください。
	0034	FATAL ERROR: SPURIOUS INTERRUPT		
	0035	FATAL ERROR: ILLEGAL INSTRUCTION		
	0036	RTC ACCESS ERROR		日付、時刻データが消失している可能性がありますので、再度設定してください。
	0037	FATAL ERROR: MPU IN LIMP MODE ERROR		電源をON／OFFしてください。それでも、エラーが出る場合は、ユニットを交換してください。
	0038	FATAL DEBUG ERROR: SYSTEM DEBUG ERROR IN PORT xxxxx	システムエラー	
	0039	未使用	－	－
オペレーションエラー	0040	BREAK EXECUTED IN LINE xxxxx	ユーザ設定により停止されました。	－
	0041	CANNOT RUN WITH SWITCH IN STOP POSITION	スイッチがSTOPの位置で、RUNしようとしました。	スイッチをSTARTの位置にしてからRUNしてください。
	0042	USER BREAK	ユーザにより停止されました。	－
	0043	FILE TRANSFER INTERRUPTED	プログラム転送中に停止されました。	－
	0044	PROGRAM MEMORY ERROR	ユーザプログラムが壊れています。	START/STOPスイッチを操作してメモリをイニシャルして、電源を再投入してください。
	0045	未使用	－	－
	0046	未使用	－	－
	0047	未使用	－	－
	0048	未使用	－	－
	0049	未使用	－	－
通信ポートエラー 注．□はエラーが発生しているポート番号です。	0050	BAD PORT NUMBER	ポート番号が間違っています。	ポート番号を確認してください。
	0051	CANNOT READ FROM WRITE-ONLY DEVICE	出力用外部機器に対して入力を行っています。	ポートの指定または外部機器の指定を確認してください。
	0052	PORT ALREADY OPEN	その番号のポートはすでに開かれています。OPEN命令で同じポートを続けて2回開いています。	OPEN命令を１つ削除してください。

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
通信ポートエラー 注．□はエラーが発生しているポート番号です。	□053	PARITY ERROR COMMUNICATING WITH DEVICE	外部機器との通信で、エラーが発生しました。	通信回線を確認してください。
	□054	未使用	−	−
	0055	PORT CONFIGURATION ERROR	ポートの通信条件設定が間違っていいます。 COM ON割込を使用している場合、ポートの通信条件が制限を越えています。	ポートの通信条件設定を確認してください。 または、ポート1のボーレートと通信割込使用条件が制限を越えていないか確認してください。
	□056	DIRECT STATEMENT	LOADコマンドを実行した際、ASCIIユニットに転送するプログラム内に行番号のない行が発見されました。	ASCIIユニットに転送するプログラムを確認してください。
	0057	PORT IS NOT OPEN	オープンしていないポート番号を使用しました。または、デバイス記号による入出力(送受信)の指定と異なった入出力(送受信)を行おうとしました。	適切なデバイス記号のOPEN命令で、ポートを開いてください。
	□058	FRAMING ERROR COMMUNICATING WITH DEVICE	ポートから読み出された受信データのフレーム形式(スタートビット＋データ＋パリティ＋ストップビットのフレーム形式)が不正です。	OPEN命令で指定した通信条件が、外部機器のフレーム形式と一致しているか否かを確認してください。
	0059	CANNOT WRITE TO READ-ONLY DEVICE	入力専用の外部機器に対して、出力を行おうとしました。	OPEN命令で指定したデバイス記号が､外部機器の仕様と一致しているか否かを確認してください。
	0060	DEVICE UNAVAILABLE	OPEN命令で、不正なデバイス記号を使用しています。	OPEN命令内のデバイス記号を確認してください。
	□060	DEVICE UNAVAILABLE IN PORT□	割込みによりPRINT命令など入出力命令どうしの競合が発生しています。	入出力命令どうしが競合しないプログラムとしてください。
	0061	未使用	−	−
実行エラー	0062	PROTECTED PROGRAM	NEWコマンドからプログラムが保護されています。	NEWコマンドでプログラムを新規作成する場合は、PNAME" "コマンドでプログラム名を消してください。 PMEM ONが設定されている場合は、PMEM OFFしてからNEWを実行してください。
	□063	INCOMING DATA HAS BEEN LOST	受信データが消失しました。	通信速度を下げて使用してください。または、フロー制御を使用してください。
	0064	NO SUPPORT	サポートしていません。 フラッシュROMにプログラムがセーブされていないのに自動転送設定となっています。	OPTION LENGTH命令と他の変数定義、命令の位置を見直してください。
	0065	FLASHROM	装着されているフラッシュROMが異常です。	再度実行してください。 それでもエラーが出る場合は、ユニットを交換してください。

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
実行エラー	0066	VERIFY ERROR	フラッシュROMの照合時に、エラーが発生しました。	RAM上のプログラムに問題がない場合は、もう1度ROMSAVEで正しいデータをフラッシュROMに書き込んでください。
	B067	FORMAT ERROR	PC READ／PC WRITE命令のフォーマットまたは変数指定が間違っているか、またはCPUユニット側の割付Ｉ/Oの転送チャネル数、転送元先頭チャネルNo.指定が不正です。	ASCIIユニット側のPC READ／PC WRITE命令の内容、または割付Ｉ/Oのn+3,n+4CHを確認してください。
	0068	未使用	－	－
	0069	PASSWORD INCORRECT	パスワードが間違っています。	正しいパスワードを入力してください。
	0070	DATA TRANSFER CHECK SUM ERROR	データ転送中にデータ化けが発生しました。または、フラッシュROMに何も書き込まれていません。	データ転送をやり直してください。
	0071	未使用	－	－
	0072	未使用	－	－
	0073	未使用	－	－
	0074	未使用	－	－
	0075	未使用	－	－
	0076	未使用	－	－
	0077	未使用	－	－
	0078	未使用	－	－
	0079	未使用	－	－
PCインタフェースエラー	0080	メッセージなし	IOWR／IORD命令が不正です。	IOWR／IORD命令のオペランドを確認してください。
	0081	メッセージなし	IOWR／IORD命令の転送（読み書き）チャネル数が不正です。	IOWR／IORD（#00□□）命令：最大90CH、IOWR／IORD（#FD00）命令：最大128CHの範囲内にしてください。
	0082	メッセージなし	IOWR（#CC00）命令によるASCIIユニットへの割込みが不正です。	IOWR（#CC00）命令の割込番号がON PC（割込番号）命令と一致しているか否かを確認してください。

9－1　エラーメッセージ一覧

エラー種別	エラー種別＋エラーコード	エラーメッセージ	エラー内容	処置
PCインタフェースエラー	0083	メッセージなし	IOWR／IORD(#00□□)命令による共有メモリの読み書き時、読み書き先頭チャネル(IOWRエリア／IORDエリアの先頭からのオフセットチャネル□□)が不正です。	IOWR／IORD(#00□□)命令のオフセットチャネル□□、および IOWRエリア／IORDエリアの先頭アドレス(割付DMシステム設定m+6,m+7CHで設定)を確認してください。
	0084	未使用	－	－
	0085	メッセージなし	PC割込No.が不正です。	1～99を使用してください。
	0086	メッセージなし	PC割込No.がBCD値でありません。	BCD値としてください。
	0087	未使用	－	－
	0088	未使用	－	－
	0089	未使用	－	－
DM設定エラー	0090	メッセージなし	起動時のプログラムNo.設定が不正です。(DM割付エリア)	00～04(BCD)のいずれかを設定してください。
	0091	メッセージなし	フラッシュROMからの自動転送設定が不正です。(DM割付エリア)	00、5AHexのいずれかを設定してください。
	0092	メッセージなし	スタートモード設定が不正です。(DM割付エリア)	00、5AHexのいずれかを設定してください。
	0093	メッセージなし	初期値転送設定が不正です。(DM割付エリア)	0、1(BCD)のいずれかを設定してください。
	0094	メッセージなし	システム予約(DM割付エリア)	0を設定してください。
	0095	メッセージなし	ポート#1,2,ターミナルのいずれかの通信速度設定が不正です。(DM割付エリア)	00～08(BCD)いずれかを設定してください。
	0096	メッセージなし	1サイクル当たり転送チャネル数が不正です。(DM割付エリア)	PCがSYSMACα、またはCSシリーズであることを確認の上、00、5AHexのいずれかを設定してください。
	0097	メッセージなし	スクリーンサイズ、またはターミナルモードが不正です。	設定値を範囲内の値に設定してください。
	0098	メッセージなし	IORDエリアまたはIOWRエリアの先頭アドレス設定が不正です。(DM割付エリア)	00、10～99(BCD)のいずれかを設定してください。
	0099	メッセージなし	DMA送信モードの設定が不正です。	00、5AHexのいずれかを設定してください。

参　考　**エラー発生時に、BASICプログラム内で、エラー処理のプログラムを組むときは、次の命令を使用します。**

こんなとき	命　　令	備　　考
エラー発生時に、割込サブルーチンに分岐するとき	ON ERROR GOTO命令	割込サブルーチン内に、エラー処理のプログラムを組みます。
発生したエラーコードを知りたいとき	ERR関数	ON ERROR GOTO先の割込サブルーチン内で使用します。
発生した行番号を知りたいとき	ERL関数	ON ERROR GOTO先の割込サブルーチン内で使用します。
エラー発生をシミュレートしたいとき	ERROR命令	指定したエラーコードのエラーが発生したと見なした処理が行われます。

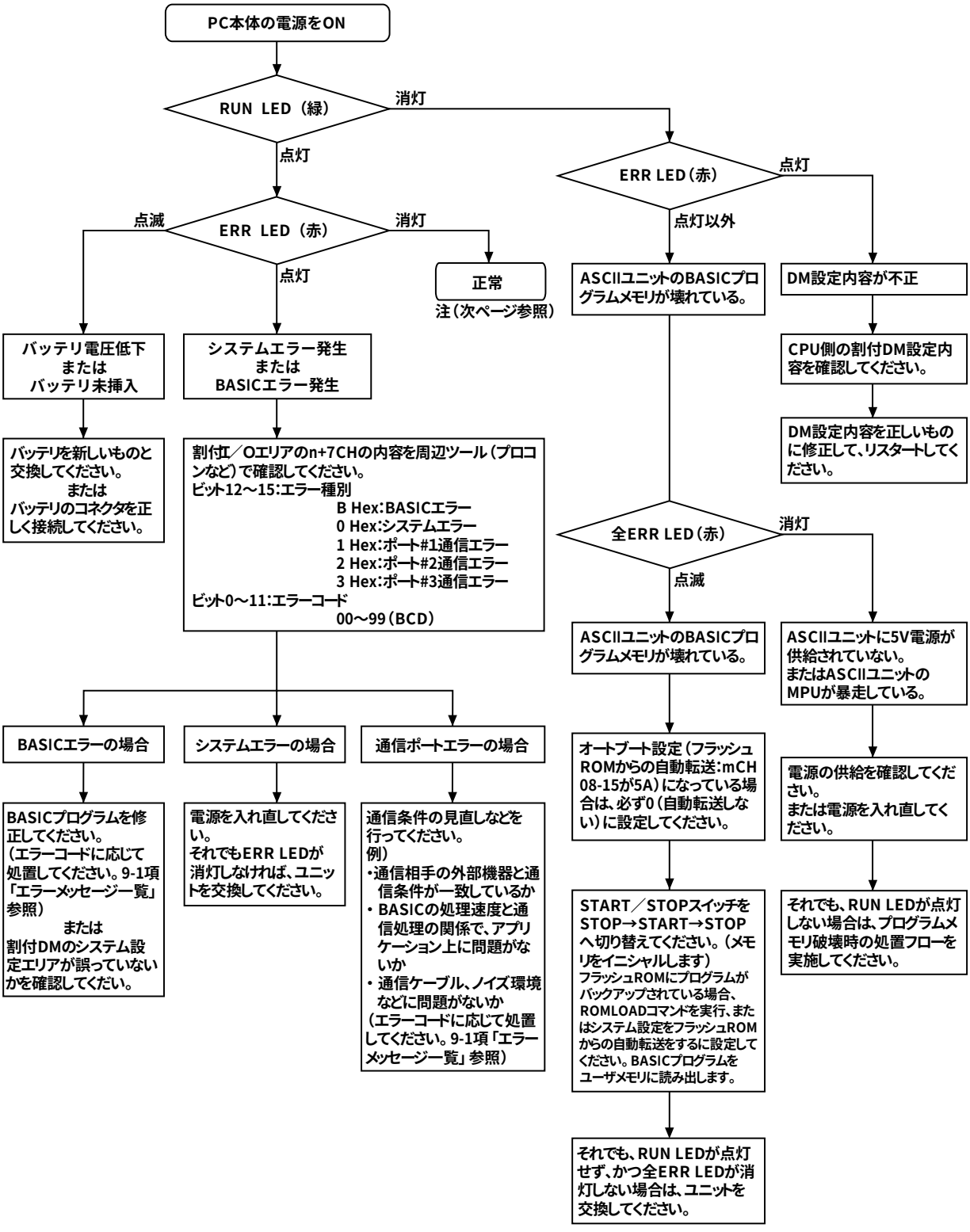
9−2　異常とその処置

■異常一覧

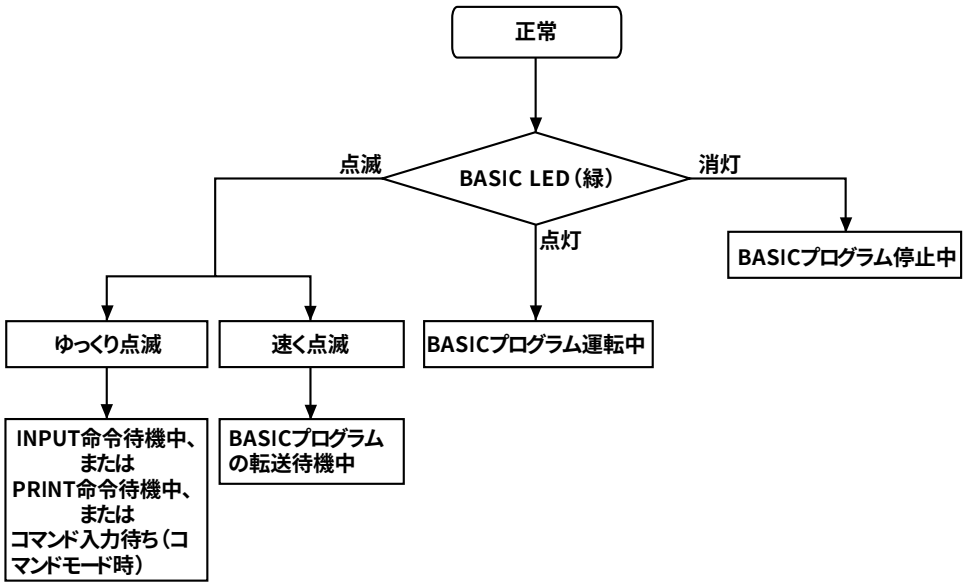
異常項目	原　　因	処　　置
LED表示が全て点灯していない	PC本体の電源がOFF となっている。	PC本体の電源をONにしてください。
	ASCIIユニットがしっかりと固定されていない。	ASCIIユニットを正しく装着してください。
	高機能I/Oユニットのいずれかが不良（高機能I/Oユニットの立上り待ち）。 このときCPUユニットは運転を開始しません。	異常のある高機能I/Oユニットを交換してください。 （異常ユニットは「I/Oテーブル読出し」操作で「$」のみの表示になっています。） ユニット交換の前に下の欄の処置*1を実施してください。
	高機能I/Oユニットの号機No．設定が重複している（I/Oユニットオーバー）。 このときCPUユニットは運転を開始しません。	号機No．が重複しないように設定し直してください。 号機No．は「I/Oテーブル読出し」操作で見ることができます。
	CPUユニット⇔本体 ASCIIユニット間のリフレッシュが正常に行われなかった（高機能ユニット異常）。 このときASCIIユニットのみ運転を停止します。	原因解除後、号機No．に応じたリスタートフラグでリスタート（OFF→ON→OFF）をかけてください。 リスタートをかけても復帰しない場合は、下の欄の処置＊1を実施してください。
	BASICプログラムメモリが破壊されているため、「高機能ユニット異常」や「CPU待機中」が発生している。	（処置＊1） ASCIIユニット裏面のユーザメモリデフォルトスイッチをON（右）にして電源を再投入してください。ユニットの全LEDが点滅したら電源をOFFし、スイッチを元にもどして電源を再投入してください。この状態でも復帰しない場合は、ユニットを交換してください。 詳細は「プログラムメモリ破壊時の処置フロー」に従ってください。
「ERR1／2／T」LED点灯（通信ポート#1/#2/ターミナルポートエラー）	接続機器の電源がOFFとなっている。	接続機器の電源をONにしてください。
	接続機器への接続ケーブルが外れている。	接続ケーブルを正しく接続し、ねじ止めして固定してください。
	接続ケーブルの断線、コネクタの接触不良。	接続ケーブルの修理、または交換をしてください。
	接続機器とASCIIユニットの伝送速度、通信条件があっていない。	接続機器とASCIIユニットの伝送速度、通信条件を合わせてください。
「ERR」LED点灯（システム／BASICエラー）	PCのDM設定に不正データが設定されている。 BASICプログラムの実行上、問題が発生している。	エラーコード／エラーメッセージを確認し、そのエラーに応じた処置を行ってください。
「ERR」LED点滅（バッテリエラー）	バッテリのコネクタが外れている。	バッテリのコネクタを正しく接続してください。
	バッテリの電圧が低下している。	バッテリを新しいものと交換してください。
全ERR LED表示が点滅し、初期画面で "PROGRAM MEMORY ERROR"が表示される。 [CTRL]＋[X]*1 も効かない。	BASICプログラムメモリが破壊されている。	DM設定のフラッシュROMからの自動転送を「自動転送しない」に設定した後、ASCIIユニット前面のSTART／STOPスイッチをSTOP→START→STOP操作してください。 BASICプログラムメモリがクリアされます。その後、電源を再投入してください。 フラッシュROMにプログラムがバックアップされていれば、この後ROMLOADコマンドで復旧することができます。

＊1：[CTRL]＋[X]は、[CTRL]キーを押しながら[X]キーを押すことを示します。

■トラブルシューティングフロー



注：正常時のBASIC LED**の動作**



■プログラムメモリ破壊時の処置フロー

1　PC本体の電源をOFFにして、ASCIIユニットをベースから取り外します。

2　ASCIIユニット裏面のユーザメモリデフォルトスイッチをON（右側）にして、ベースに取り付けます。

3　PC本体の電源をONにして、ASCIIユニットの全LED（T/Rを除く）が点滅するのを確認します。PC本体は「CPU待機中」となります。このとき、DM設定がフラッシュROMから自動転送する（mCH 08-15：5A）になっている場合は、0（自動転送しない）に設定してください。
（LEDが点滅しない場合は、ユニットを交換してください。）

4　全LEDが点滅するのを確認したら、PC本体の電源をOFFにしてASCIIユニットをベースから取り外します。

5　ASCIIユニット裏面のユーザメモリデフォルトスイッチをOFF（左側）にして、ベースに取り付けます。

6　PC本体の電源をONにして、RUN LEDが点灯するのを確認します。
（RUN LEDが点灯しない場合は、ユニットを交換してください。）

7　プログラムメモリはデフォルト状態（オールクリア状態）となっていますので、RAM上に再度プログラムを転送（DM設定をフラッシュROMから自動転送する［mCH 08-15：5A］など）してください。

お願い　**プログラムメモリ破壊時の処置を行う場合は、必ずDM設定のフラッシュROMからの自動転送が「自動転送しない（mCH 08-15：00）」になっていることを確認してください。「自動転送する（mCH 08-15：5A）」に設定されている場合は、0に設定してから処置を行ってください。**

9−3　CPU側の異常表示

C200H/HS、C200HX/HG/HE(-Z)、CSシリーズのCPU側(CPUユニット)では、ASCIIユニットの異常に対して、次の監視を行います。ASCIIユニットは、高機能I/Oユニットの分類として監視されます。

■高機能I/Oユニットの異常一覧

異常項目	異常要因および動作	処　置
高機能I/O立ち上がり待ち	・ASCIIユニットのプログラムメモリが破壊されているか、または高機能I/Oユニットがハード的に不良です。 ・このとき、PCは運転を開始しません。	・プログラムメモリ破壊時の処置フローを実施してください。それでも復帰しない場合は、異常のある高機能I/Oユニットを交換してください。 ・異常ユニットは「I/Oテーブル読み出し」操作を行うと「$＊＊」だけの表示になっています。
I/Oユニットオーバー	・高機能I/Oユニットの号機No.の設定が重複しています。 ・このとき、PCは運転を開始しません。 ・また、特殊補助リレー25415がONします。	・号機No.が重複しないように設定し直してください。 ・号機No.は「I/Oテーブル読み出し」操作で確認することができます。
高機能ユニット異常	・CPU⇔高機能ユニット間のリフレッシュが正常に行われませんでした。 ・このとき、異常ユニットだけ運転を停止します。 ・また、特殊補助リレー25415(CS1の場合、A40206)がONします。	・C200H/HSの場合は、補助記憶リレーAR0000〜0009で異常ユニットの号機No.を調べ、異常要因を取り除いたあと、AR0100〜0109でリスタート(OFF→ON→OFF)してください。 ・C200HX/HG/HE(-Z)の場合は、リレー番号28200〜28215で異常ユニットの号機No.を調べ、異常要因を取り除いたあと、28100〜28115でリスタート(OFF→ON→OFF)してください。 ・CS1の場合は、リレー番号A41800〜A41815で異常ユニットの号機No.を調べ異常要因を取り除いたあと、A50200〜A50215でリスタート(OFF→ON→OFF)してください。 ・リスタートをかけても復帰しない場合は、プログラムメモリ破壊時の処置フローを実施してください。それでも復帰しない場合は、ユニットを交換してください。

■高機能I/Oユニット用補助記憶リレー／特殊補助リレーの機能

●高機能I/Oユニットのエラー表示

以下に示す異常が発生した場合、下表に示すリレーがONします。

・高機能I/Oユニットの号機No.が重複して設定されている場合

・CPUユニット⇔高機能I/Oユニット間のリフレッシュが正常に行われなかった場合

リレー番号				機　能
C200H/HS	C200HX/HG/HE(-Z)	CSシリーズ		
		異常時	号機No.重複時	
AR0000	28200	A41800	A41100	0号機が異常時ONします。
AR0001	28201	A41801	A41101	1号機が異常時ONします。
AR0002	28202	A41802	A41102	2号機が異常時ONします。
AR0003	28203	A41803	A41103	3号機が異常時ONします。
AR0004	28204	A41804	A41104	4号機が異常時ONします。
AR0005	28205	A41805	A41105	5号機が異常時ONします。
AR0006	28206	A41806	A41106	6号機が異常時ONします。
AR0007	28207	A41807	A41107	7号機が異常時ONします。
AR0008	28208	A41808	A41108	8号機が異常時ONします。
AR0009	28209	A41809	A41109	9号機が異常時ONします。
－	28210	A41810	A41110	A(10)号機が異常時ONします。
－	28211	A41811	A41111	B(11)号機が異常時ONします。
－	28212	A41812	A41112	C(12)号機が異常時ONします。
－	28213	A41813	A41113	D(13)号機が異常時ONします。
－	28214	A41814	A41114	E(14)号機が異常時ONします。
－	28215	A41815	A41115	F(15)号機が異常時ONします。

SYSMACαは、C200HX/HG-CPU5□(-Z)/6□(-Z)、C200HX-CPU85-Zのみ

機　種	リレー番号	機　能
C200HX/HG/HE(-Z) C200H/HS	25415	高機能I/Oに異常があった場合ONします。
CS1	A40206	
	A40113	高機能I/Oユニットの号機No.が重複している場合ONします。

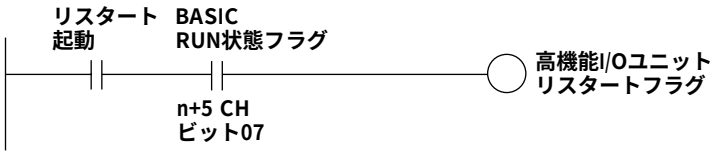
●高機能I/Oユニットリスタートフラグ

高機能I/Oユニットをリスタートする場合、下表のリレーをOFF→ON→OFFしてください。このフラグを使用すると、電源をOFFせずにリスタートすることができます。

リレー番号			機　能	
C200H/HS	C200HX/HG/HE(-Z)	CS1		
AR0100	28100	A50200	0号機リスタートフラグ	
AR0101	28101	A50201	1号機リスタートフラグ	
AR0102	28102	A50202	2号機リスタートフラグ	
AR0103	28103	A50203	3号機リスタートフラグ	
AR0104	28104	A50204	4号機リスタートフラグ	
AR0105	28105	A50205	5号機リスタートフラグ	
AR0106	28106	A50206	6号機リスタートフラグ	
AR0107	28107	A50207	7号機リスタートフラグ	
AR0108	28108	A50208	8号機リスタートフラグ	
AR0109	28109	A50209	9号機リスタートフラグ	
－	28110	A50210	A(10)号機リスタートフラグ	SYSMACαは、C200HX/HG-CPU5□(-Z)/6□(-Z)、C200HX-CPU85-Zのみ
－	28111	A50211	B(11)号機リスタートフラグ	
－	28112	A50212	C(12)号機リスタートフラグ	
－	28113	A50213	D(13)号機リスタートフラグ	
－	28114	A50214	E(14)号機リスタートフラグ	
－	28115	A50215	F(15)号機リスタートフラグ	

お願い　**電源ON後、ASCIIユニットが立ち上がっている最中にリスタートをかけると、ASCIIユニット内のプログラムメモリが壊れることがあります。リスタートは、ASCIIユニットが完全に立ち上がってからかけるようにしてください。**

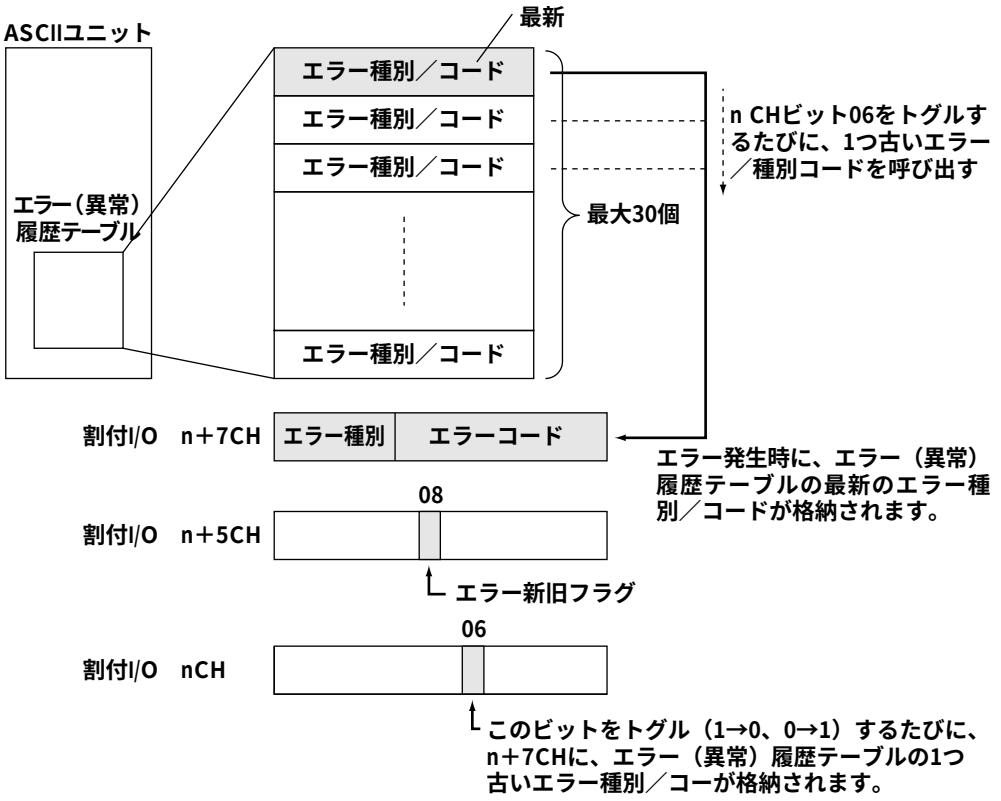
例



9−4　異常履歴の読出し／クリア

■エラー(異常)履歴テーブルの概要

ASCIIユニット内には、最大30個のエラー(異常)履歴テーブルがあります。
エラーが発生すると、エラーコードおよびエラー種別が、この履歴テーブルに格納されます。



■エラー(異常)履歴テーブルの読出し方

エラー(異常)履歴テーブルは、次のように割付I/Oに反映されるため、CPUユニットから割付I/Oを介して読み出すことが可能です。

1. エラー発生時には、最新のエラーコードおよびエラー種別が、割付I/Oのn+7CHにセットされます。

2. 過去のエラーコードおよびエラー種別に遡りたいときは、割付I/OのnCHのビット06をトグル(1→0､0→1)するたびに、エラー(異常)履歴テーブルを参照して、1つ古い(直前の)エラーコードおよびエラー種別がn+7CHにセットされます。遡っていって、それ以上古いエラーがなければ、一度n+7CHが00Hexになった後、最新のエラーに戻ります。
   注：エラー発生時にそのエラーコードをBASICプログラム内で読み出したいときは、ERR関数を使用します。

3. 読み出したエラーが、プログラムRUNしてからのものかどうか(エラー情報の新、旧)を割付I/Oのn+5CHのビット08でモニタすることができます。このビットは、1つ1つの異常履歴のエラー種別／コードに対応しています。

■エラー(異常)をクリアする方法

●BASIC命令で、エラー(異常)履歴テーブル内のエラーコードをクリアする場合

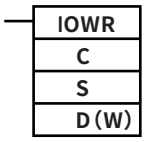
ERC命令を使用して、エラーをクリアします。

コマンドモードでERCを実行すると、そのとき発生していたエラー状態を解除し、かつ異常履歴テーブルにあるエラー種別／コードもすべてクリアします。

●CPUユニットから、ASCIIユニットのエラーをクリアする場合

CPUユニットから、発生しているASCIIユニットのエラー状態を解除するには、コントロールコードを#EC00としたインテリジェントI/O書込命令(IOWR命令)を実行します。

なお、異常履歴テーブルに発生したエラー種別／コードは残ります。



C：#EC00
S：任意(データ内容は0000としてください)
SYSMACαシリーズの場合
D：ビット12～15：号機No.(0～F)
およびビット00～11：001Hex
CS1シリーズの場合
W：ビット0～3：号機No.(0～F)
W+1：0001Hex

●BASIC命令プログラム内で、エラーをクリアする場合

プログラム内でERCを実行すると、そのとき発生しているエラー状態を解除します。解除されたエラー種別／コードは、異常履歴テーブルに残っています。

9−5　保守

■取り扱いの際のお願い

・ASCIIユニットの交換は、電源を切った状態で、行ってください。
・ASCIIユニットを交換した場合、再度ユニットが異常でないかを確認してください。
・ASCIIユニットが故障した場合、ASCIIユニット内のユーザメモリエリア(RAM)またはフラッシュROMのBASICプログラムをまったく読み出せないことがあります。作成したBASICプログラムを、外部機器(パソコンなど)に保存することをお勧めします。

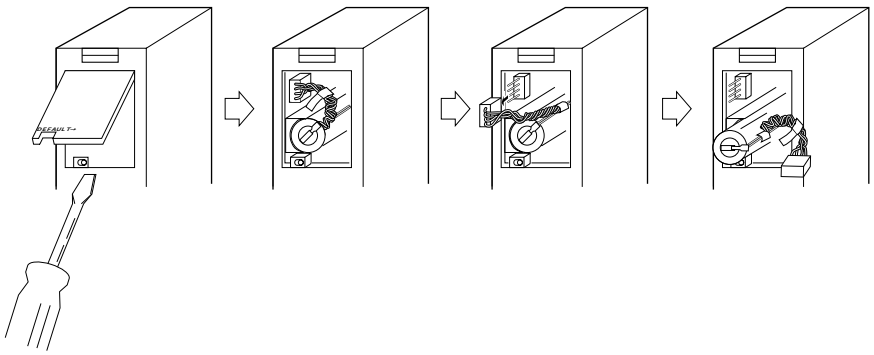
■バッテリの交換方法

1　電源をOFFにします。(最初から電源が入っていないときは、いったん1分以上電源を投入した後、電源をOFFにしてください。)

2　ベースユニットのロックレバーをドライバなどで下に押しながら、ASCIIユニットを取り外します。

3　ASCIIユニット裏面のバッテリ収納部のカバーを、マイナスドライバを使用して、取り外します。

4　バッテリをコネクタごと引き抜き、新しいバッテリと交換します。交換は5分以内に行ってください。



5　手順4の逆の作業で新しいバッテリを取り付けます。

6　バッテリ収納部のカバーを取り付けます。

7　ASCIIユニットをベースユニットに取り付けます。

●交換用電池

名称：バッテリセット　　形式：形C200H-BAT09

●バッテリ寿命と交換時期

電池の有効期間(最大寿命時間)は、ASCIIユニットへの通電、無通電の時間によらず、周囲温度25℃の条件で5年です。周囲温度が高いと、寿命は短くなります。

前面のERROR LEDが点滅(赤)してから、1週間以内に、バッテリを交換してください。なお、前面のERROR LEDが点滅(赤)すると、割付I/Oのn＋5CHのビット06がONします。このビットをバッテリのメンテナンス用に使用してください。

お願い **故障発生時、すみやかな修復作業を行うために、できるだけ予備のASCIIユニット、バッテリを用意してください。**

付録

付－1 ご使用上の留意事項

■形C200H-ASC02から置き換える場合

●実行時間

各命令語の実行時間が、形C200H-ASC11/ASC21/ASC31と形C200H-ASC02では異なります。そのため、BASICプログラムの実行時間が変わりますので、ご注意ください。

●機械語プログラムの使用

形C200H-ASC02で作成した機械語プログラムは、形C200H-ASC11/ASC21/ASC31では使用できません。

●実数のフォーマット

形C200H-ASC11/ASC21/ASC31では、実数(単精度・倍精度)の浮動小数点フォーマットが、IEEE-754準拠に変更されたため、実数を使用した演算結果が変わってくることがありますので、ご注意ください。

演算結果に精度を要する場合は、引数に倍精度実数(数値)を使用してください。

例：

```
10  A#=0.9+0.1
20  A=ACOS(A#)
```

●割込処理における同一命令の競合

形C200H-ASC02では、BAISCプログラムの1つの命令実行中は割込が入ることがありませんでした。これに対して、形C200H-ASC11/ASC21/ASC31では、1つの命令実行中でも割込が入ります。したがって、ある命令実行中に割込がはいって、割込サブルーチン内でそれと同じ命令を実行したときに、命令間の競合が発生し、その結果うまく動作しないことがあります(例：PRINT命令実行中に割込が入って、割込サブルーチン内でPRINT命令、INPUT､LINE INPUT命令を同じポートに対して実行したとき)。その場合、各種割込の制御(許可／禁止／停止)命令を使用して、命令間の競合は発生しないように、排他制御を行ってください。

●通信関連エラーの取り扱い

形C200H-ASC02では、通信上でエラーが発生した場合、CPUユニット側のエラーフラグがONするだけでBASICプログラムの実行は停止しませんでした。

形C200H-ASC11/21/31では、通信上でエラーが発生した場合、ON ERROR命令を使用しない場合、形C200H-ASC02と同様にPC側のエラーフラグがONするだけで、BASICプログラムの実行は停止しません。ただし、ON ERROR命令を使用した場合、通信エラー時にも、指定したサブルーチンへ分岐します。しかし、ON ERROR命令を使用する場合、エラー割込処理のサブルーチンへ分岐中に新たなエラー(通信エラーは除く)が発生すると、BASICプログラムの実行は停止されますので、ご注意ください。

●内部補助リレーの割付の変更

内部補助リレーの各割付チャネル／ビットの機能が変更されています。CPUユニット側のラダープログラムを流用する場合は、割付内容を再度チェックしてください。

また、ASCIIビジーフラグの機能・動作が変更されていますので、このフラグを使用する際にはご注意ください。

●ASCIIユニットのシステム設定の変更

ASCIIユニットのシステム設定方法が、ユニット裏面のディップスイッチから、割付DMエリアの先頭10CHに変更されています。PC本体用周辺ツールを使用して、CPUユニットの割付DMの設定を行ってください。

●通信時の送信コード、伝送制御信号の変更

OPEN命令実行時のデバイス記号指定に応じた、PRINT命令実行時に出力するコードの制約が、変更されています(ポート2のTERM指定時を除いて、すべて送信時コードを出力に変更)。同時に、OPEN命令時のデバイス記号指定に応じた、各伝送制御信号の動き(タイミングチャート)も、変更されています。詳細については、第4章「外部汎用機器とのデータ交換」を参照して、外部機器とのインタフェースを確認してください。

●ターミナルのエミュレーションモードの選択

形C200H-ASC02で使用していたターミナルをそのまま形C200H-ASC11/ASC21/ASC31でも使用する場合、そのターミナルが、「VT100」モードをサポートしているときは、「VT100」モードを使用してください。「VT100」モードをサポートしていないときは、「FIT10」モードを使用してください。

●PC READ／WRITE(@付きを含む)命令の文字列変数の使用時

形C200H-ASC02では、 PC READ／WRITE(@付きを含む)命令において、変数に文字列変数を使用した場合、文字列の先頭から4文字をWRITE(書込み)したり、4文字だけ変数へREAD (読出し)するようになっていました。

これに対して、形C200H-ASC11/21/31では、変数に格納されている分をWRITE(書込み)したり、最大255文字までチャネルデータをREAD(読出し)することができます。

例：

形C200H-ASC02の場合

PC READ "@D, 0, 5, 5H4"；A$, B$, C$, D$, E$

形C200H-ASC11/21/31の場合

PC READ "@D, 0, 5, 5H4"；A$

A$に5CHのデータがすべてREADされます。

したがって、下記フォーマットで、形C200H-ASC02と形C200H-ASC11/21/31とでは動作が異なります。

例：

PC READ "@D, 0, 5, 5H4"；A$(0)

形C200H-ASC02の場合

最初の1CH分のデータが、 A$(0)へ格納されます。

形C200H-ASC11/21/31の場合

5CH分のデータが、すべてA$(0)へ格納されます。

従来の形C200H-ASC02で下記のようにご使用になっている場合、形C200H-ASC11/21/31では動作が異なりますので、プログラムを変更してからご使用ください。

例1）

変更前：

```
10 PC READ "@D, 0, 10, S10H4"；A$(0)
```

←形C200H-ASC11/21/31ではフォーマットエラーとなります。

↓

変更後：

```
10 PC READ "@D, 0, 10, S10H4"；A (0)
20 FOR I= 0 TO 9 ：A$(I)=RIGHT$("0000"＋HEX$(A(I)), 4)：NEXT I
```

例2)

変更前：

```
10 PC READ "@D, 0, 5, 5H4"；A0$, A1$, A2$, A3$, A4$
```

↓

変更後：

```
10 PC READ "@D, 0, 5, 5H4"；A$
20 A0$=MID$( A$, 1, 4)
30 A1$=MID$( A$, 5, 4)
40 A2$=MID$( A$, 9, 4)
50 A3$=MID$( A$, 13, 4)
60 A4$=MID$( A$, 17, 4)
```

形C200H-ASC11/21/31では、文字列変数を使用すると、PCのDMデータを外部機器へ送ったり、外部機器からのデータをDMへ送ったりすることが簡単になります。

例）

DM0000	1 2 3 4
DM0001	5 6 7 8
DM0002	9 A B C
：	：
：	：

→100 PC READ "@D, 0, 3, 3H4"；A$ →A$="123456789ABC"
110 SD$=HEAD$＋A$＋TERM$
120 PRINT SD$

例3）

変更前：

```
10 PC READ "@D, 0, 255, 100A3, 100A3, 55A3"；A$, B$, C$
```

←形C200H-ASC11/21/31ではフォーマットエラーとなります。

↓

変更後：

```
10 PC READ "@D, 0, 100, 100A3"；A$
20 PC READ "@D, 100, 100, 100A3"；B$
30 PC READ "@D, 200, 55, 55A3"；C$
```

転送チャネル数が127CH以下の場合で、複数の変数に入れたい場合は、例2)のようにMID$関数で転送後分けてください。

変更前：

```
10  PC  READ  "@D, 0, 4, 2A3, 2A3"；A$, B$
                          ↓
```

変更後：

```
10  PC  READ "@D, 0, 4, 4A3"；A$
20  B$=MID$(A$, 5, 4)
30  A$=MID$(A$, 1, 4)
```

形C200H-ASC11/21/31では、変数に文字列変数は1つまで使用可能となっています。従来の形C200H-ASC02のように複数指定できませんので、ご注意ください。
その他のPCフォーマットの詳細については、付－3「PCフォーマットの説明」を参照してください。

●PRINT USING命令の&フォーマット使用時

PRINT USING命令において、"&　　&"フォーマットで指定したフォーマットの文字数よりも短い文字列を出力する場合、そのフォーマットに対して足りない文字数分のスペースは自動で付加されません。スペースを必要とする場合は、出力する文字列にSPACE$命令などでスペースを付加してからPRINT USING命令で出力するようにしてください。

■ターミナルの操作時

●ターミナルソフトウェアによる制約

形C200H-ASC11/ASC21/ASC31では、ターミナルエミュレーションモードとして、「VT100」モードが使用可能です。ただし、ご使用のターミナルソフトウェアがVT100モードをサポートしても、ターミナルソフトウェア自身の機能上の制約によって、ASCIIユニットの機能が一部使用できないことがあります(例：日本語版Windows95付属のHyper Terminalでは、矢印キーが機能しないなど)。

●EDIT時による制約

EDITにおいて、編集行を↑、↓キーにて移動する場合、キーを押し続けての移動はしないでください。編集行に、↑、↓キーコードが付加されてしまうことがあります。

■プログラムの実行時

形C200H-ASC02では、入力されたBASICソースプログラムをそのまま実行します。これに対して、形C200H-ASC11/ASC21/ASC31では、BASICプログラムの実行速度を上げるために、入力されたBASICソースプログラムをいったん中間コード(命令内容を意味する簡易コード)にコンパイルしてから実行します。そのため、形C200H-ASC11/ASC21/ASC31では、RUNコマンドを実行しても、このコンパイル処理のため、実際にプログラムを実行するまで、時間を要することがあります。RUNコンパイル時間は、53Kbyte(約1300行)のプログラムで約10秒かかります。
また、プログラム中で使用している命令により、この時間は変わりますので、ご注意ください。
なお、中間コードは、プログラムエリアが変更後のRUN時に、再生成されます。

■プログラムの保存時

ROMSAVEコマンドによって、フラッシュROMにプログラムを書込み中(ターミナルに終了表示が出るまで)に、PCの電源を絶対に切らないでください。フラッシュROMに書込み中に、PCの電源を切ると、フラッシュROMが電気的に壊れ、セーブできなくなる可能性があります。したがって、ROMSAVEコマンドを実行するときは、PCの電源環境に注意してください。

■RS-422A/485ポートについて

形C200H-ASC21のポート#2(RS-422A/485)は、1:N通信として、使用できます。ただし、このポートでは、全二重通信はできませんので、ご注意ください。

■FCS命令による誤りチェックコードの算出方式について

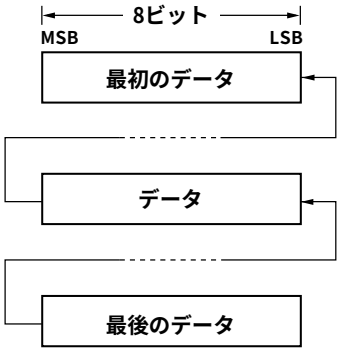
FCS命令による文字列の誤りチェックコードの算出における、CRC-CCITTおよびCRC-16の算出の方式は、以下のとおりです。通信相手となる外部機器の算出方式と一致しているかどうかを確認してください。

●CRC-CCITT

XMODEM方式を採用しています。

全8ビットデータを以下のように、直列につなぎ、このデータを決められた17ビットの2進数(1　0001　0000　0010　0001)で除算し、その余り(16ビット)の値を誤りチェックコードとします。

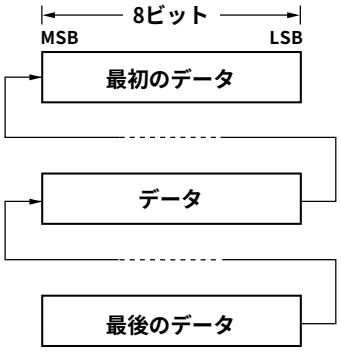
演算の初期値は、0(16ビットすべて0)としています。



●CRC-16

MODBUS方式を採用しています。

全8ビットデータを以下のように、直列につなぎ、このデータを決められた17ビットの2進数(1　1000　0000　0000　0101)で除算し、その余り(16ビット)の値を誤りチェックコードとします。演算の初期値は、－1(16ビットすべて1)としています。



■受信割込使用時の通信速度について

ON COM割込を使用する場合は、各ポートの通信速度は以下となるようにしてください。

ポート1＋ポート2≦38.4kビット/s

この条件を越えて設定してあると、「PORT CONFIGURATION ERROR」となります。

■PCとのデータ授受と通信速度について

1サイクル当たりのデータ転送チャネル数を最大127CHとして、比較的大きいサイズのチャネル数(例えば127CH)でPC READ/PC WRITEを行っている場合で、通信速度の速いもの(19.2kビット/s以上)で通信を行っていると、PCとのデータ授受、データ受信が重なると、PCとのデータ授受時間延びますので、結果的にPC側のスキャンタイムが延びることがあります。

PC側のスキャンタイムの影響を最小にするには、1サイクル当たりのデータ転送チャネル数を最大20CHにするなどしてください。

■プログラム実行中断後の操作について

通信ポートをOPENしているようなBASICプログラムの実行を、[Ctrl]＋[X]キーなどで中断したとき、外部から通信ポートにデータが受信された状態になっていると、BASICプログラムの実行中断後に、LIST表示後のプロンプトの戻りなどが遅くなることがあります。このような場合は、実行中断後、CLOSEコマンドを入力してから、LIST表示などを行ってください。

■命令実行中のBASIC割込実行について

命令内で完了待ちが発生するような命令(WAIT命令での監視対象命令)で、その命令内の待ち状態でBASICプログラムの実行が止まっているとき、BASIC割込(ON COM、ON PCなど)が発生した場合、全BASIC割込に対して1つだけ割込サブルーチンが実行されます。その他発生した割込は保留扱いとなり、それぞれ1つまで記憶されます。

■サブルーチンからの分岐の繰り返しによるユーザメモリの消費

以下の①と②の命令を組み合わせて使用するときは、次に示す点に注意してください。

① GOSUB命令によるサブルーチン、または割込命令(ON COM、ON KEY、ON PC、ON TIMER、ON ALARM、ON TIME$、ON ERRORのいずれか)によるサブルーチン

② GOTO命令、またはON ERROR GOTO／RESUME命令、またはON KEY GOTO命令、またはWAIT命令

上記①のGOSUB命令または割込によるサブルーチン内から、上記②のGOTO命令などによって強制分岐すると、スタックエリアにデータが残ったままとなり、ASCIIユニット内のユーザメモリエリアを消費します。これを繰り返すと。BASICエラーの「OUT OF MEMORY」(メモリ容量の不足、エラーコード：B007)が発生、またはスタックエリアがオーバーフローし、システムが正常に動作しなくなる恐れがあります。

必ず、GOTO命令などによる分岐後も、サブルーチン内に戻り、RETURN命令によってサブルーチン内から出るように、プログラムを作成してください。
以下に注意が必要な例を示します。
例：

```
10  ON  ERROR  GOTO  100
20  PRINT  FRE
30  GOSUB  1000
90  END
100  PRINT  ERR, ERL;
110  RESUME  20
1000  A=1/0  'Divide by zero
1010  RETURN
```

上記例では、サブルーチン内の1000行から、（エラーが発生すると）ON ERROR GOTO命令により100行に分岐します。
このような場合は、110行をRESUME 20（エラー処理後、行番号20からの再開）ではなく、RESUME NEXT（エラーが発生した行の次の行からの実行開始）にするなどして、サブルーチンから出るようにしてください。

■プログラムメモリ破壊時のユニット動作について

プログラムメモリが破壊されると、PROGRAM MEMORY ERRORやPC本体上で高機能ユニット異常、CPU待機中が発生することがあります。
この異常状態からはASCIIユニットの裏面にあるユーザメモリデフォルトスイッチを使用することで復帰させることができます。このとき、プログラムメモリは強制的に初期化されますので、プログラム入力後はROM SAVEでプログラムをフラッシュROMへセーブするようにしてください。
フラッシュROMへセーブしたプログラムのLOADは「7－1 プログラミングの手順」の「■フラッシュROMへの書込み／読出し」を参照して行ってください。
また、POKE命令によるユーザプログラムエリアへのデータ書込みは、注意して行ってください。プログラムメモリ破壊の原因となることがあります。POKE命令によるプログラム破壊は、すぐにメモリ異常として現象が出ない場合があります。

■プログラムの実行までの時間について（立ち上がり時間）

デバッグを行ったままROMSAVEすると、プログラムメモリ上にゴミが残ったままとなるので、電源ON時の立ち上がりが遅くなることがあります。
このような場合には、プログラムを一度ターミナルにてSAVEし、ターミナルからLOADし直してから、ROMSAVEしてください。

■プログラムメモリ操作中の電源断、リスタートについて

プログラムメモリを操作する処理中（LOAD/SAVE、ROMLOAD/ROMSAVE、オートブート、RUNコンパイル）中にPC本体からのリスタートや電源断を行わないでください。プログラムメモリが破壊されると正常に動作しなくなる場合があります。

■ ユーザプログラムメモリ不足時の動作について

ユーザメモリエリアにおけるBASICプログラムの占める容量が大きくなるとプログラム実行中にメモリ不足状態となり、「FATAL DEBUG ERROR：SYSTEM DEBUG ERROR IN PORT 104」などが表示され、プログラム実行を停止することがあります。

メモリ不足は、そのプログラム内容によってその対処方法は異なりますが、以下のようなBASICプログラムの見直しを行ってください。

1)BASICプログラムサイズを縮小する

プログラム内容で共通化できるところをサブルーチン化したり、処理をまとめることが可能な箇所(例えば、いくつかの異なる条件での共通結果の変数への代入など)は一括処理するなどしてプログラム行数を減らしてください。

【BASICプログラムサイズの確認】

```
>PINFALL
P1      ：TEST    [R/W]  73231 BYTES     ←BASICプログラムサイズ
P2      ：TEST2   [R/W]  56224 BYTES     ←BASICプログラムサイズ
P3      ：                0 BYTES
P4      ：                0 BYTES
LIB     ：                0 BYTES
I_CODE  ：                0 BYTES

74111 BYTES FREE
```

2)プログラム中の変数で変数名が同じ名称で型が異なるものは変数名を変更する

例えば、変数Aと変数A$を使用している場合などは、同じ名称とならないように“A$”→“A1$”にするなどして変数名を変更してください。

変数名称が同じにならないようにしておけば、コンパイル時にメモリ不足のチェックが可能となります。

ただし、通常文字変数の場合は、プログラム実行中にサイズが変動するため、メモリ不足チェックは完全ではありません。チェックを必要とする場合には、OPTION LENGTH命令を使用すれば文字変数のサイズを固定長としチェックすることができます。

付－2 従来機種との比較の詳細

形C200H-ASC11/ASC21/ASC31と、従来機種の形C200H-ASC02との比較の詳細を以下に示します。比較の概要については、1-5項「従来機種との比較の概要」を参照してください。

項目		形C200H-ASC02	形C200H-ASC11/ASC21/ASC31
プログラムの実行		入力したソースプログラムをそのまま実行するため、RUN時すぐに実行する	入力したソースプログラムを中間コードにコンパイルしてから実行するため、RUN後しばらくして実行する
メモリ	文字変数エリアの割付方式	ダイナミック方式	スタティック(固定)方式、ダイナミック(自由)方式を選択可(OPTION　LENGTH命令の使用／非使用による)
	ガーベジコレクション(未使用領域の再編成による領域確保)	プログラム実行中に実行	RUNコマンド実行時に実行
	メモリエリアの指定	FRE命令でメモリエリアの指定をする	なし
通信	通信バッファ	受信バッファ：256バイト 送信バッファ：256バイト	受信バッファ：512バイト 送信バッファ：512バイト
	ER(DTR)信号の動き	システムで制御される	システムで制御されず、プログラム上のDTR命令によっても制御できる。
	データ送信時に送信可条件待ちのときのLED	ERR　LEDが点灯	BASIC　LEDがゆっくり点滅
	OPEN命令実行時の指定デバイス記号による、PRINT命令実行時に出力するコードの制約	コードによって出力しないコードや、LFを付加して出力するコードがあり	通信ポート2にてTERMは使用不可。 それ以外はすべて出力する(制約なし)
	PRINT命令によるデータ送信時のDMA(ダイレクトメモリアクセス)データ送信機能	なし(DMA送信なし：ソフトウェアにより、データ送信し、その間BASICプログラムは実行停止)	あり(DMA送信する／しないを選択可能：ソフトウェアによるデータ送信と、ソフトウェアはトリガのみでハードウェアによる並行データ送信可)
CPUユニットとのデータ交換	CPUユニットの1サイクル当たりの、PC　READ／PC　WRITE(@付き含む)によるデータ交換数	最大20CH	最大20CHまたは最大127CH選択可(SYSMACα、CS1のみ)
	割付I/Oによるデータ交換数(PC　PUT／PC　GET命令を使用)	CPUユニット→ASCIIユニット：8ビット ASCIIユニット→CPUユニット：8ビット	CPUユニット→ASCIIユニット：32ビット ASCIIユニット→CPUユニット：16ビット
	CPUユニット側でASCIIユニットで発生したエラーをモニタ	不可	可 ASCIIユニット→CPUユニットへの割付I/Oに、発生中または過去(最大30個をトグル切替)のエラーコードおよび種別が格納される

項目		形C200H-ASC02	形C200H-ASC11/ASC21/ASC31
CPUユニットとのデータ交換	PC READ/PC WRITEなどに文字型変数を使用したときのデータの格納方法	1文字変数に対して4文字までしかREADされない、またはWRITEしない	1文字変数に対して255文字の範囲内でREAD、またはWRITEされる
フラグ	ソフトリスタートフラグ	あり	なし
	PC　READ要求とPC割込	Writeフラグで共用	PC　READ要求フラグとPC割込フラグは別フラグ
	ASCIIビジーフラグ	PC割込時、レスポンスとして変化する	PC割込時に変化しない。代わりに、PC割込失敗のレスポンスとして、PC割込failフラグが変化する
割込	多重割込	する（割込サブルーチン実行中に、×××ON命令があると、他の割込が入ったとき、実行中の割込サブルーチンを途中で終了し、他の割込サブルーチンへ分岐）	しない（割込サブルーチン実行中に、他の割込が入ったとき、実行中の割込サブルーチンをすべて実行後、次の割込サブルーチン実行。ただしON ERROR割込時は即実行）
		多重割込の実行順は、割込発生順にしたがう	次の割込サブルーチンの実行順は、割込優先順位にしたがう
	割込の実行	1命令実行中は、BASIC割込が発生しても割込サブルーチンへ分岐しない。命令の実行完了後に分岐する	1命令実行中であっても、BASIC割込で割込サブルーチンへ分岐する。ただし、1度だけ
処理速度	GOSUB、GOTO命令の飛び先による実行時間	飛び先により変化する	どこへ飛んでも一定
命令	RND命令	乱数発生器ロジックが異なる	
	WAIT命令	割込サブルーチン中もタイマは止まらない	割込サブルーチン実行中はタイマ停止
	PRINT　USING命令	USINGフォーマットによる表示が異なるケースあり	
数値の範囲	整数の範囲	－32768～32767	－2147483648～2147483647
	単精度実数の範囲	－1.70141E+38～1.70141E+38	$\pm 1.2\times10^{-38}$～3.4×10^{38}
	倍精度実数の範囲	－1.701411834604692D+38～－1.701411834604692D+38	$\pm 2.2\times10^{-308}$～1.79×10^{308}
	算術関数の結果の精度	単精度	倍精度まで可
ROMへの保存／読出		EEPROMからの読み出しはLOADを使用	ROMLOAD、ROMSAVEを使用して、フラッシュROMとユーザメモリ（RAM）との読み書き可能

付－3 PCフォーマットの説明

■PCフォーマットの説明

ASCIIユニットが、CPUユニットのI/Oメモリのデータまたは ASCIIユニット自身の共有メモリを下記の命令で読み書きする場合、I/Oメモリまたは共有メモリのデータの形式をどういう形式と見なしてASCIIユニットの変数に読み出し、 ASCIIユニットの変数からI/Oメモリまたは共有メモリへどういうデータ形式で格納するかを指定する必要があります。そのデータ形式の指定フォーマットを、PCフォーマットと呼びます。

対象命令：

CPUユニットのI/Oメモリのデータに対して：PC READ／PC WRITE、PC READ@／PC WRITE@、PC QREAD@／PC QWRITE@命令

ASCIIユニットの自身の共有メモリのデータに対して：PC EPUT／PC EGET命令

1. A型フォーマットでは、ASCIIユニットの1変数にPC側の複数チャネルを対応することができます。ただし、1度に最大127チャネルまでです。
   A型、S型フォーマット、および変数が文字列変数以外は、1チャネルごとに1つの変数を対応させる必要があります。
   A型フォーマットは、1つのフォーマットに対して1つの変数を対応させます。
   S型フォーマットは、1つのフォーマットに対して1つの配列変数を対応させます。
   A型、S型フォーマット、および変数が文字列変数以外では1つのフォーマットに対して複数の変数を対応させることができます。

2. mを省略したときは1チャネル分となります。
   1度に扱えるチャネル数は整数型：1～255、文字型：1～64(Aフォーマットは1～127)チャネルです。
   フォーマットおよびサブコマンドは、文字変数でもかまいません。
   フォーマットおよびサブコマンドは大文字で記述してください。

●CPUユニット・I/OメモリまたはASCIIユニット・共有メモリ→ASCIIユニット・変数へのデータ読み出し時

PCフォーマット名称	フォーマット		意　味	例
I型(BCD) フォーマット	mIn	変数が数値型：m=1～255 n=1～4 変数が文字型：m=1～255 （注1） n=1～4 ただし変数が文字型の場合 m×n＜256	mチャネル分のデータをBCDデータとみなして、各チャネルの下位1～n桁のデータを各変数に格納します。	2I3： 2チャネル分のデータをBCDとみなして、各チャネルの下位1～3桁(ビット00～11)のデータを2個の変数に格納します。
H型(16進) フォーマット	mHn	変数が数値型：m=1～255 n=1～4 変数が文字型：m=1～255 （注1） n=1～4 ただし変数が文字型の場合 m×n＜256	mチャネル分のデータを16進データとみなして、各チャネルの下位1～n桁のデータを各変数に格納します。	2H3： 2チャネル分のデータを16進とみなして、各チャネルの下位1～3桁(ビット00～11)のデータを2個の変数に格納します。
O型(8進) フォーマット	mOn	変数が数値型：m=1～255 n=1～4 変数が文字型：m=1～255 （注1） n=1～4 ただし変数が文字型の場合 m×n＜256	mチャネル分のデータを8進データとみなして、各チャネルの下位1～n桁のデータを各変数に格納します。	2O3： 2チャネル分のデータを8進とみなして、各チャネルの下位1～3桁(ビット00～11)のデータを2個の変数に格納します。
B型(ビット) フォーマット	mBn	変数が数値型：m=1～255 n=0～15 変数が文字型：m=1～255 （注1） n=1～15 ただし変数が文字型の場合、文字数が255以下となること	mチャネル分のデータをバイナリデータとみなして、各チャネルのnビット目のデータのみ(他を0とみなして)を10進データに変換して各数値変数に格納します。	2B3： 2チャネル分のデータをバイナリとみなして、各チャネルの3ビット目のデータのみ(他を0とみなして)を10進データに変換して各数値変数に格納します。
A型(ASCII) フォーマット	mAn	m=1～255(注2) n=1～3 ただし n＝1～2のときは m＝127以下	mチャネル分のデータをASCIIデータとみなして、各チャネルのnで指定する上位／下位データを1個の文字変数に格納します。 n=1のとき：下位バイト n=2のとき：上位バイト n=3のとき：上位および下位バイト	2A3： 2チャネル分のデータをASCIIデータとみなして、各チャネルの上位および下位バイト(計4文字)を1個の文字変数に格納します。
S型(配列変数) フォーマット	SmXn 注:Xは、I,H,O,Bのいずれか	m=1～255 n=各指定フォーマットと同様	mチャネル分のデータをXで指定するデータ形式(BCD、16進、8進、ビットのいずれか)のデータとみなして、各チャネルのデータをn個の配列変数に格納します。 変数に文字配列型は使用できません。	S2I2： 2チャネル分のデータをBCDデータとみなして、各チャネルのデータを2個の配列変数に格納します。

注1：V1.02までのバージョンではm＝1～64です。
　　バージョンはターミナル接続時の初期画面でご確認ください。

注2：V1.02までのバージョンではm＝1～127です。
　　バージョンはターミナル接続時の初期画面でご確認ください。

●ASCIIユニット・変数→CPUユニット・I/OメモリまたはASCIIユニット・共有メモリへのデータ読み出し時

PCフォーマット名称	フォーマット		意　味	例
I型(BCD)フォーマット	mIn	変数が数値型： m=1～255 n=1～4 変数が文字型： m=1～64 n=1～4	m個の各変数の下位1～n桁のデータをBCDデータに変換してmチャネル分の各チャネル下位1～n 桁に格納します。	2I3： 2個の変数の下位1～3桁のデータをBCDデータに変換して2チャネル分の各チャネルの下位1～3桁に格納します。
H型(16進)フォーマット	mHn	変数が数値型： m=1～255 n=1～4 変数が文字型： m=1～64 n=1～4	m個の各変数の下位1～n桁のデータを16進データに変換してmチャネル分の各チャネル下位1～n 桁に格納します。	2H3： 2個の変数の下位1～3桁のデータを16進データに変換して2チャネル分の各チャネルの下位1～3桁に格納します。
O型(8進)フォーマット	mOn	変数が数値型： m=1～255 n=1～4 変数が文字型： m=1～64 n=1～4	m個の各変数の下位1～n桁のデータを8進データに変換してmチャネル分の各チャネル下位1～n 桁に格納します。	2O3： 2個の変数の下位1～3桁のデータを8進データに変換して2チャネル分の各チャネルの下位1～3桁に格納します。
B型(ビット)フォーマット	mBn	変数が数値型： m=1～255 n=0～15	m個の各数値変数を10進とみなしてバイナリデータに変換し、各nビット目のデータのみを(他を0として)mチャネル分の各チャネルに格納します。	2B3： 2個の各数値変数を10進とみなしてバイナリデータに変換し、各3ビット目のデータのみを(他を0として)2チャネル分の各チャネルに格納します。
A型(ASCII)フォーマット	mAn	m=1～127 n=1～3	1個の文字変数の先頭から2×m文字分をASCIIデータに変換して(2文字ずつを1チャネルに対応させ)、mチャネル分の各チャネルの、nで指定する上位／下位バイト位置に格納します。 n=1のとき：下位バイト(上位バイトは00Hex) n=2のとき：上位バイト(下位バイトは00Hex) n=3のとき：上位および下位バイト	2A3： 1個の文字変数の先頭から4文字分をASCIIデータに変換して、2チャネル分の各チャネルの、上位および下位バイトに格納します。
S型(配列変数)フォーマット	SmXn 注:Xは、I,H,O,Bのいずれか	m=1～255 n=各指定フォーマットと同様	n個の各配列変数のデータを、Xで指定するデータ形式(BCD、16進、8進、ビットのいずれか)に変換して、 mチャネル分の各チャネルに格納します。変数に文字配列型は使用できません。	S2H2： 2個の配列変数のデータを16進に変換して、2チャネル分の各チャネルに格納します。

1度に転送できるチャネル数の最大は255チャネルですが、以下の制限事項があるため整数型変数では実際には50～60チャネルが限度となります。
1)1行に書くことのできる文字数は最大255文字
2)整数型変数では、1チャネルごとに1変数を対応させる必要があります。

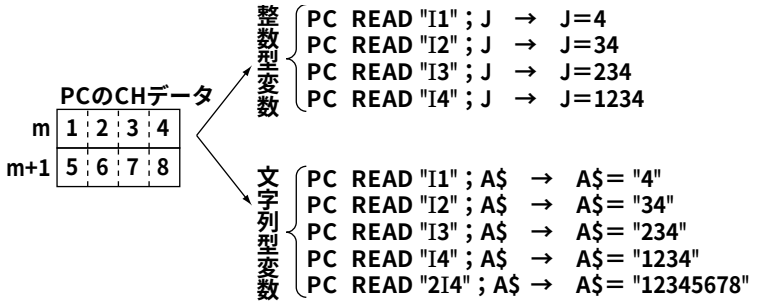
(例)H型フォーマットの場合

```
1000  PC READ  "59H4"; A0, A1, A2, …, B0,  B1, …,
└──────────────┘└──────────────────────┘
      19文字              最大236文字
```

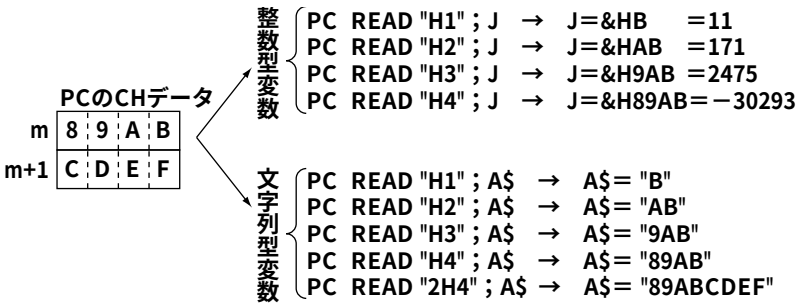
1変数に3文字を使用するとして書き込むことができる変数の最大個数は236／3＝79
したがって、最大79チャネル分の転送となります。

■PC READフォーマット変数列

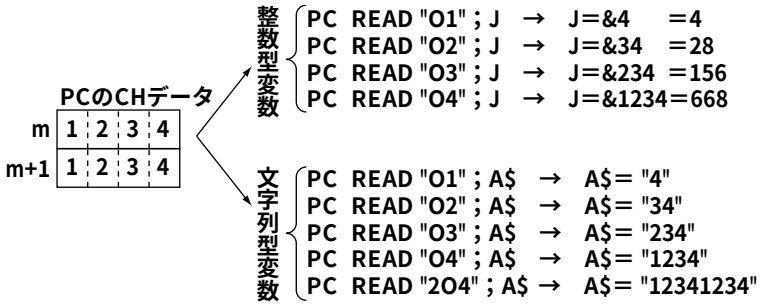
(1)I型フォーマット



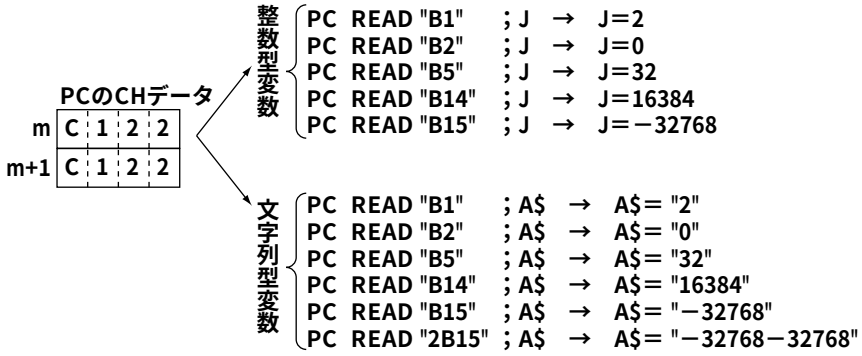
(2)H型フォーマット



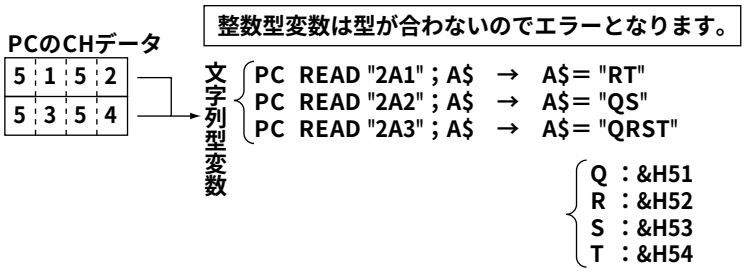
(3)O型フォーマット



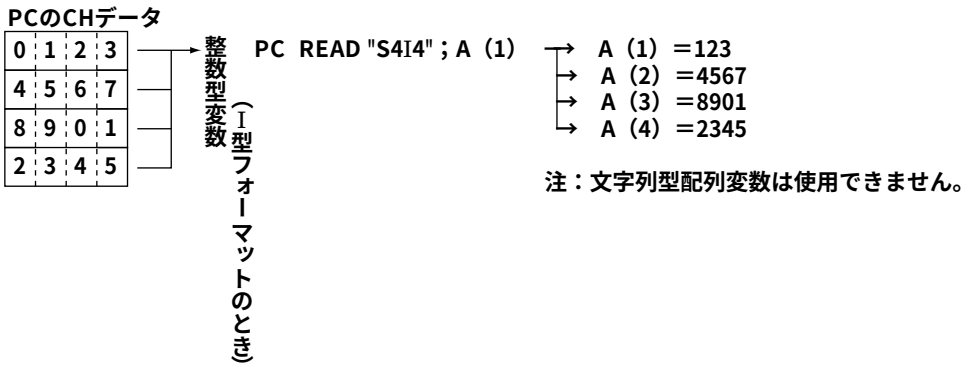
(4)B型フォーマット



(5)A型フォーマット

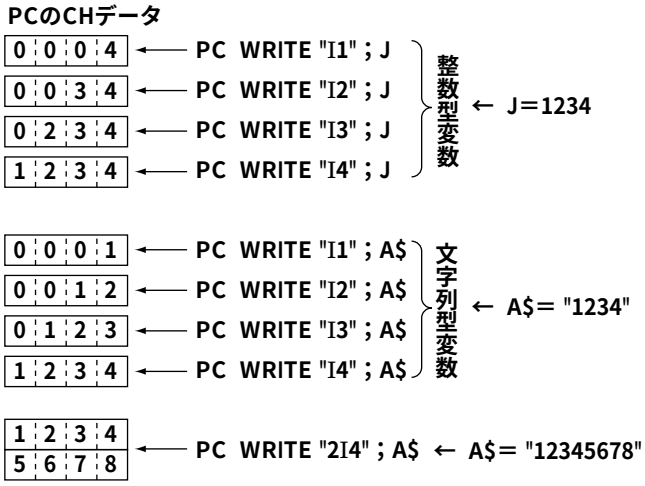


(6)S型フォーマット

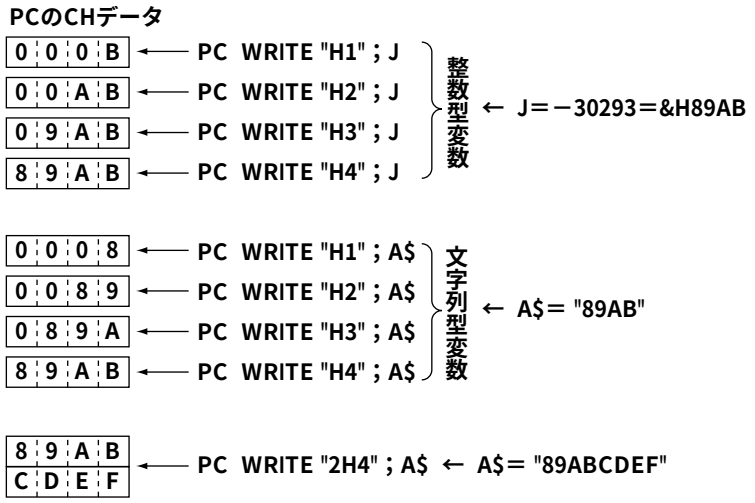


■PC WRITEフォーマット変数列

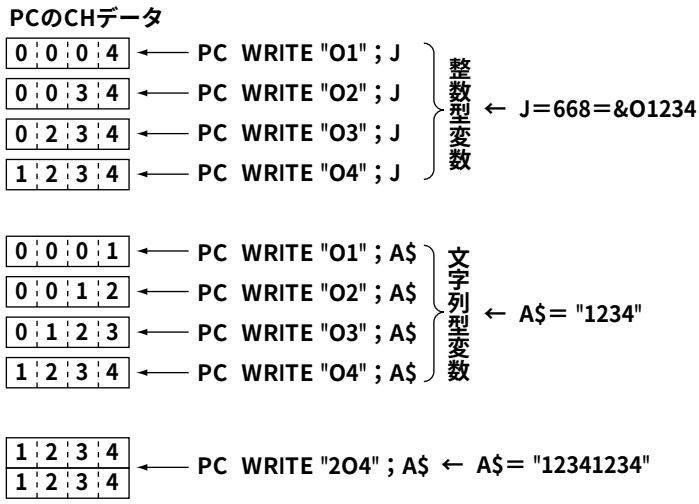
(1)I型フォーマット



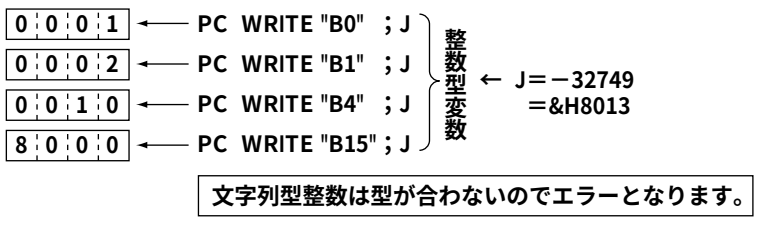
(2)H型フォーマット



(3)O型フォーマット



(4)B型フォーマット



文字列型整数は型が合わないのでエラーとなります。

●フォーマットとの変数の関係

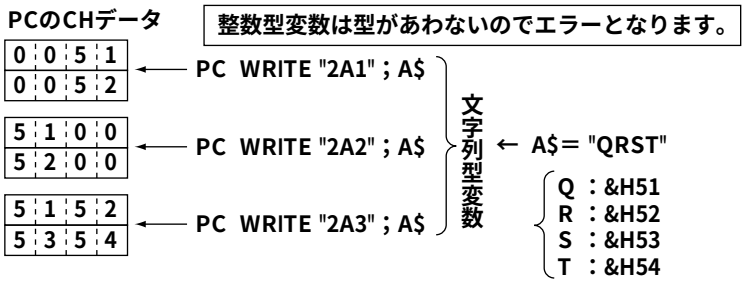
Bnのとき　変数を2進数で表したときnビットがONしていれば、PCのnビットをONし他のビットをOFFします。変数のnビットがOFFしていれば、PCのすべてのビットをOFFします。

(例)－32749＝&H8013=1000　0000　0001　0011

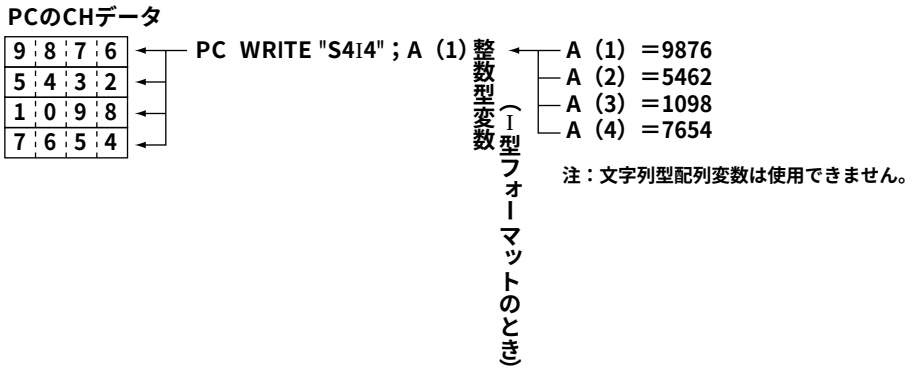
B0、B1、B4、B15……ON

B2、B3、B5～B14……OFF

(5)A型フォーマット



(6)S型フォーマット



お願い

1．PC READ/PC WRITEで、整数型変数と文字型変数を混在で使用するには、下記のように文字型変数を最後にしてください。PC WRITEで変数に配列変数を使用した場合は、下記の組合せは使用できません。

例：

PC READ "@D, 0, 2, H4, H4"；B, A$

PC READ "@D, 0, 2, H4, H4"；A$, BとするとFORMAT ERRORとなります。

2．S型フォーマットでは文字列型配列変数は使用できません。文字型データとしてREAD/WRITEする場合には、下記のようなフォーマットで行ってください。

PC READ "@D, 0, 10, 10H4"；A$

3．変数に文字列変数を使用する場合には1つまでとなります。複数の指定はできません。

4．A型フォーマットでASCIIユニット変換→CPUユニットI/Oメモリへの書き込みを行うときは、mA1、mA2を使用する場合でも、文字列型変数の文字数は2×m文字としてください。

例1：PC WRITE "@D, 0, 4, 4A1"；("1234"+"1234")

例2：A$="1234"　　└ダミーデータ

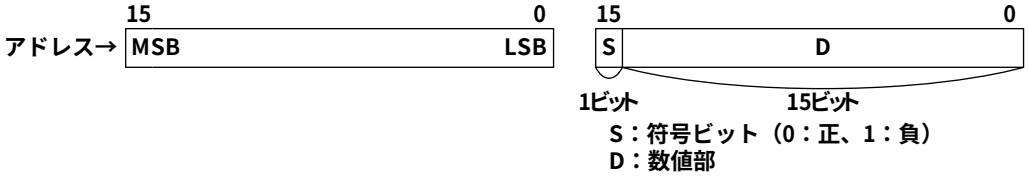
PC WRITE "@D, 0, 4, 4A1"；(A$+A$)

5．PC WRITEのI型フォーマットでは、マイナス値を書き込むことができません。フォーマットエラーとなります。

付－4 変数のメモリ上の格納形式

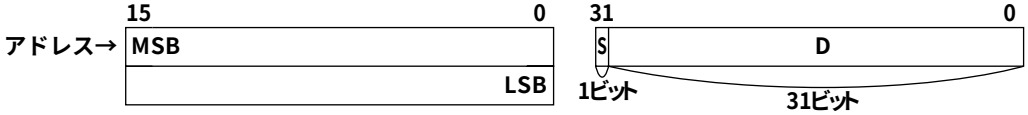
変数は、変数の型によって以下のようにメモリに格納されます。

●整数型変数



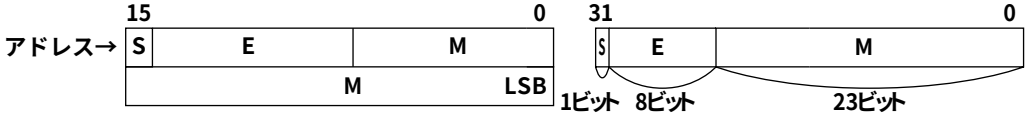
S：符号ビット（0：正、1：負）
D：数値部

●倍長整数型変数



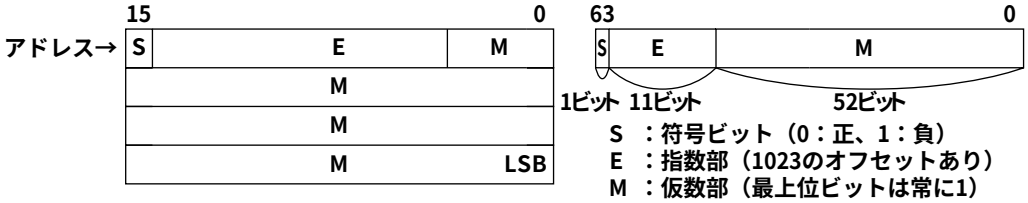
S：符号ビット（0：正、1：負）
D：数値部

●単精度実数型変数



S ：符号ビット（0：正、1：負）
E ：指数部（127のオフセットあり）
M ：仮数部（最上位ビットは常に1）

●倍精度実数型変数



S ：符号ビット（0：正、1：負）
E ：指数部（1023のオフセットあり）
M ：仮数部（最上位ビットは常に1）

●文字列変数



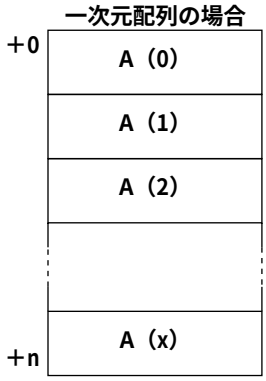
●1変数ごとのメモリバイト数が偶数になるように、空きバイトを付加して調整されます。

●現在長さ以降の使用しない文字データ部は、不定値となります。

付－４　変数のメモリ上の格納形式

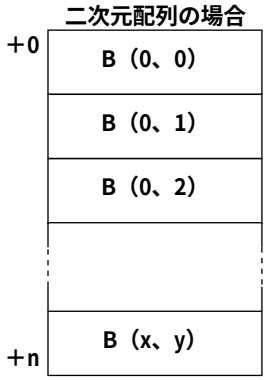
●配列変数

一次元配列の場合



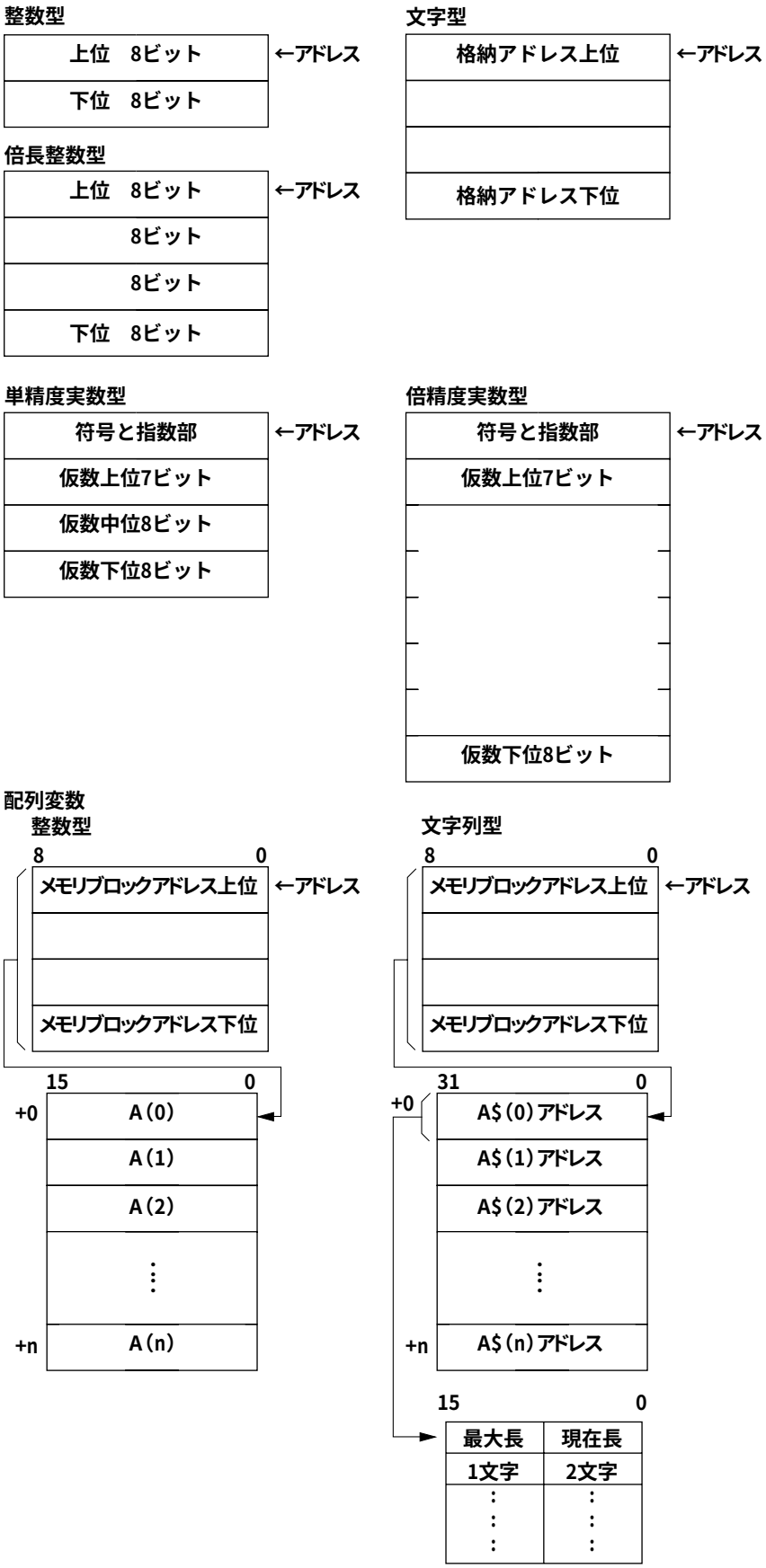
●配列の要素として、各変数が順に、メモリ上に格納されます。

二次元配列の場合



●多次元配列の場合は、一番右側の添字が先に変化する順に、メモリ上に格納されます。

〔変数の型とVARPTRで示されるアドレスとの関係〕



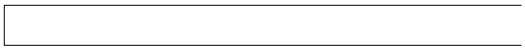
付－5 ケーブルの作成

RS-232CまたはRS-422A/485ケーブルを作成する場合は、次の手順でコネクタ配線を行ってください。

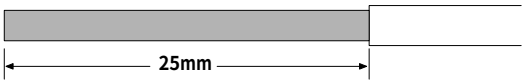
■ケーブル処理（シールド線をFGに接続する側）

各処理における長さは下図を参考にしてください。

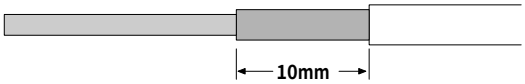
1 ケーブルを必要な長さに切断します。



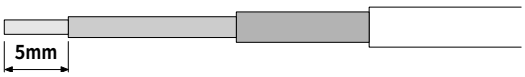
2 カミソリを用いてシースをむきます。
このときシールド線（編組）を傷付けないように注意してください。



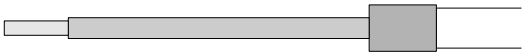
3 ハサミを用いてシールド線を切断します。



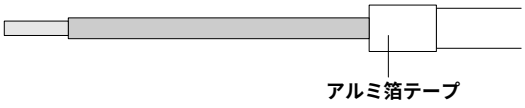
4 ストリッパを用いて各線の芯線をむきます。



5 シールド線を折り返します。



6 シールド線の折り返しの上からアルミ箔テープを巻き付けます。

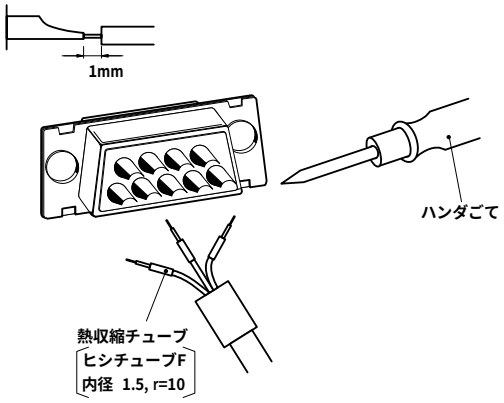


■ハンダ付け

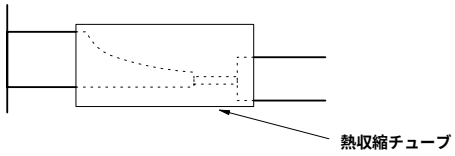
1　各線に熱収縮チューブを通します。

2　各線およびコネクタ端子に予備ハンダをします。

3　各線のハンダ付けをします。

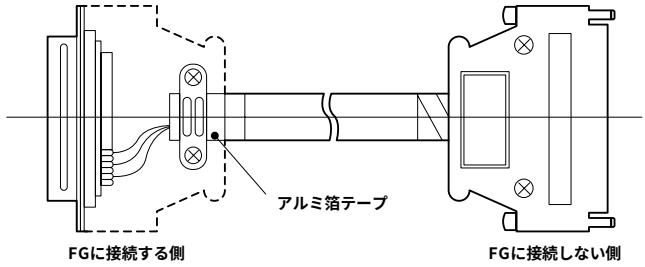


4　熱収縮チューブをハンダ部まで戻し、ジェッタでチューブを加熱して収縮させます。



■フード組み立て

コネクタフードを組み立てます。



付－6 命令実行時間サンプル

	命令	実行時間[ms]
	A=1	0.19
	A%=1	0.14
	A=A+1	0.41
	A=A－1	0.41
	A=A*1	0.39
	A=A/1	0.41
	A=A\1	0.36
	A=A MOD 2	0.35
	A=A^2	0.52
	A%=NOT A%	0.20
	A%=A% AND 1	0.28
	A%=A% OR 1	0.28
	A%=A% XOR 1	0.28
	A%=A% EQV 1	0.28
	A%=A% IMP 1	0.28
ステートメント	@IF A=0 THEN B=0 ELSE B=1 ENDIF （真、偽共）	0.45
	BITON C%, 0／BITOFF C%, 0	0.12
	CLOSE	0.60
	FOR I=1 TO 10000：NEXT I （ループ1回当たり）	0.14
	FOR I%=1 TO 10000：NEXT I% （ループ1回当たり）	0.07
	GOSUB *TEST *TEST：RETURN	0.13
	GOTO	0.03
	IF A=0 THEN B=0 ELSE B=1 （真、偽共）	0.45
	INPUT #2, A$ 200文字	約20.0
	LET	0.19
	LINE INPUT #2, A$ 200文字	約30.0
	OPEN #2, "COMU："	1.06
	PC EGET #0, 90, "S90I4；A(0)	7.8
	PC EPUT #0, 90, "S90I4；A(0)	8.4
	PC GET I, J	0.34
	PC PUT I	0.17
	PC QREAD"@D, 0, 10, 10I4"; A, B, C, D, E, F, G, H, I, J (注2)	11.5
	PC QWRITE"@D, 0, 10, 10I4"; A, B, C, D, E, F, G, H, I, J (注2)	8.8

	命令	実行時間[ms]
ステートメント	PC READ "I4"；A (注1)	7.5
	PC READ "5I4"；A, B, C, D, E (注1)	7.6
	PC READ "10I4"；A, B, C, D, E, F, G, H, I, J (注1)	7.7
	PC READ "63H4"；A$ (注1)	41.9
	PC WRITE "I4"；A (注1)	3.9
	PC WRITE "5I4"；A, B, C, D, E (注1)	4.2
	PC WRITE "10I4"；A, B, C, D, E, F, G, H, I, J (注1)	7.7
	PC WRITE "63H4"；A$ (注1)	20.7
	PC READ "@D, 0, 1, I4"；A (注1)	5.8
	PC READ "@D, 0, 5, 5I4"；A, B, C, D, E (注1)	5.9
	PC READ "@D, 0, 10, 10I4"；A, B, C, D, E, F, G, H, I, J (注1)	7.8
	PC READ "@D, 0, 63, 63H4"；A$ (注1)	32.7
	PC READ "@D, 0, 63, 63H4"；A$ （αモード） (注1)	30.8
	PC WRITE "@D, 0, 1, I4"；A (注1)	4.0
	PC WRITE "@D, 0, 5, 5I4"；A, B, C, D, E (注1)	4.0
	PC WRITE "@D, 0, 10, 10I4"；A, B, C, D, E, F, G, H, I, J (注1)	7.7
	PC WRITE "@D, 0, 63, 63H4"；A$ (注1)	20.7
	PC WRITE "@D, 0, 63, 63H4"；A$ （αモード） (注1)	9.0
	PRINT A$(9600ビット/s, 255文字)	ノーマルモード：約300 DMAモード：約5
	REM	0.03
	WHILE A<100：A=A+1：WEND （1ループ当たり）	0.67

注1：実行時間は、PCとのハンドシェイクを含めた時間です(スキャンタイム約2msで測定)。

・転送チャネル数20チャネル以下のとき

MAX　PCの2スキャンタイム

@形式　MAX　PCの1スキャンタイム

・転送チャネル数20チャネルを超えるとき(C200Hモード)

MAX　INT((転送チャネル数－1)／20)＋2スキャンタイム

@形式　MAX　INT((転送チャネル数－1)／20)＋1スキャンタイム

・転送チャネル数127チャネルを超えるとき(SYSMACαモード)

MAX　INT((転送チャネル数－1)／127)＋2スキャンタイム

@形式　MAX　INT((転送チャネル数－1)／127)＋1スキャンタイム

注2：実行時間は、PCとのハンドシェイク(IORD、IOWR命令の実行)を含めた時間です。

付－6　命令実行時間サンプル

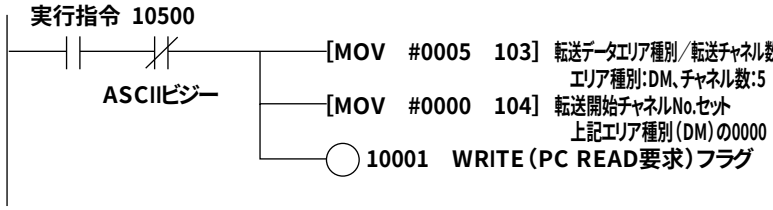
	命令	実行時間[ms]
関数	A=ABS(－1.5)	0.22
	A=ACOS(0.5)	1.72
	A%=ASC("A")	0.16
	A=ASIN(0.5)	1.7
	A=ATN(0.5)	0.74
	B#=CDBL(1)	4.7
	D$=CHR$(&H41)	0.45
	C%=CINT(0.5)	0.24
	A=COS(0.5)	0.92
	A=CSNG(0.1#)	0.22
	A=EXP(5)	1.05
	A$=STRING$(250, "A")として B$=FCS(A$, 1)　水平パリティ	1.0
	C%=FIX(1.1)	0.26
	D$=HEX$(1)	0.5
	A$=INPUT$(100, #2)	3.0
	C%=INT(1.1)	0.25
	D$=LEFT$("ASCII", 2)	0.51
	A%=LEN("ASCII")	0.17
	A%=LOC(2)	0.22
	A=LOG(5)	1.0
	D$=MID$("ASCII", 1, 1)	0.62
	D$=OCT$(&051)	0.54
	D$=RIGHT$("ASCII", 2)	0.5
	A=SIN(0.5)	0.73
	A=SQR(2)	0.66
	D$=STR$(1)	0.64
	A=TAN(3)	1.01
	A%=VAL("A")	0.2

付－7 サンプルプログラム（基本例）

号機No.：0の場合（n=100CH、内部補助リレー100～109CHに割付）

1）PCのデータをASCIIユニットへ転送する

＜ラダープログラム＞



＜BASICプログラム＞

```
100 PC READ "5I4" ; A, B, C, D, E
```

左記転送開始チャネル（DM0000）を先頭とした5CH分のデータをBCDデータとみなして、5個の変数A,B,C,D,Eに格納（読出し）

2）PCのデータをASCIIユニットへ読み出す

＜ラダープログラム＞

必要ありません

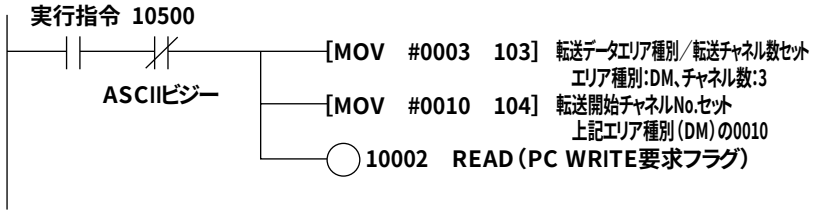
＜BASICプログラム＞

```
100 PC READ "@D, 0, 5, 5I4" ; A, B, C, D, E
```

CPUユニットのDM0000を先頭とした5CH分のデータをBCDデータとみなして、各4桁（16ビット）のデータを、5個の変数A,B,C,D,Eに格納（読出し）

3）ASCIIユニットのデータをPCへ転送する

＜ラダープログラム＞



＜BASICプログラム＞

```
100 PC WRITE "3I4" ; A, B, C
```

A,B,Cの3個の変数の4桁のデータをBCDデータに変換して、CPUユニットの左記転送開始チャネル（DM0010）を先頭とした3CHの各4桁（16ビット）に転送（書込み）

4）ASCIIユニットのデータをPCへ書き込む

＜ラダープログラム＞

必要ありません

＜BASICプログラム＞

```
100 PC WRITE "@D, 10, 3, 3I4" ; A, B, C
```

3個の変数A,B,Cの各4桁のデータをBCDデータに変換して、 CPUユニットのDM0010を先頭とした3CHの各4桁（16ビット）に転送（書込み）

5）PCとASCIIユニットで同期をとってデータをPCからASCIIユニットへ転送する（SYSMACαのみ）

＜ラダープログラム＞



＜BASICプログラム＞

```
→ 100 PC QREAD "@D, 0, 5, 5I4" ; A, B, C,D, E
```

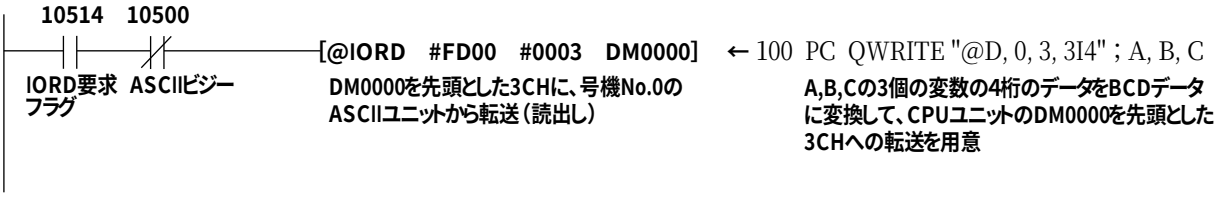
IOWR（#FD00）命令により転送された5CHのデータをBCDデータとみなして、各4桁（16ビット）のデータを5個の変数A,B,C,D,Eに格納

付

付録

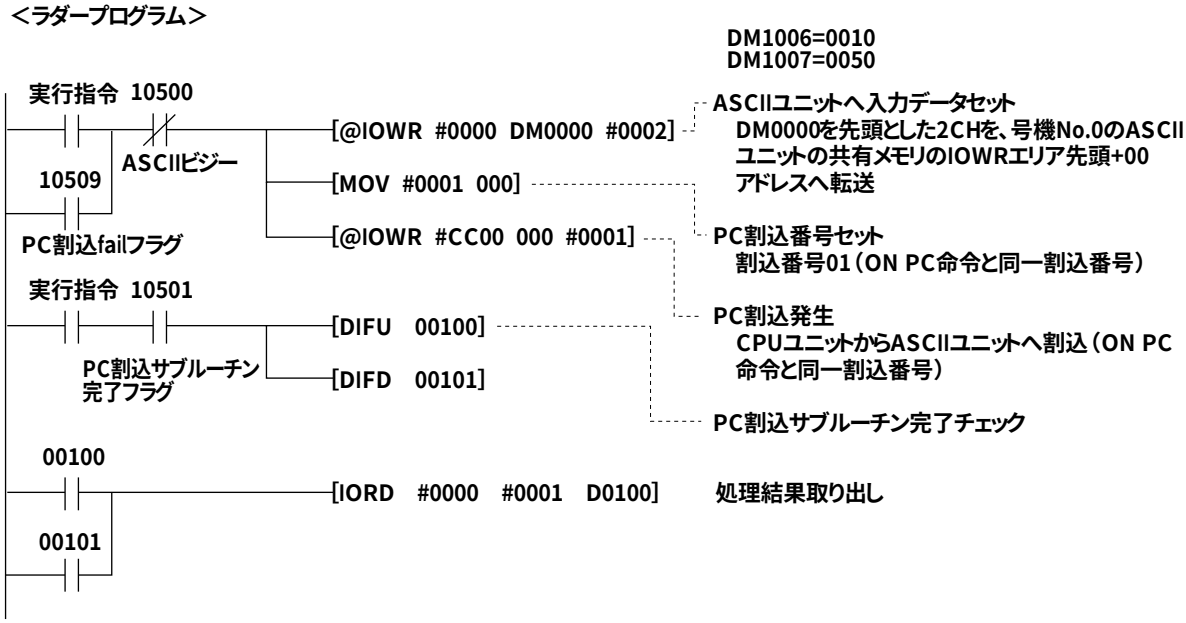
6)PCとASCIIユニットで同期をとってデータをASCIIユニットからPCへ転送する

＜ラダープログラム＞　　＜BASICプログラム＞



7)ASCIIユニットをPCのサブセットとして処理させる

＜ラダープログラム＞



＜BASICプログラム＞

```
100 ON PC 1 GOSUB *PINT
110 PC 1 ON
 :
 :
190 END
200 *PINT
210 PC EGET #0, 2, "2I4" ; A, B
220 C%=A+B
230 PC EPUT #0, 1, "I4" ; C%
240 RETURN
```

付－8 予約語一覧

No.	予約語	No.	予約語	No.	予約語	No.	予約語
0	@IF	43	LINE	86	SYSLOAD	129	ERR
1	ALARM	44	LIST	87	SYSMOVE	130	EXP
2	ALL	45	LLIST	88	THEN	131	FCS
3	AUTO	46	LOAD	89	TIMER	132	FLX
4	BASE	47	LPRINT	90	TO	133	FRE
5	BITOFF	48	MAP	91	TROFF	134	HEX$
6	BITON	49	NEW	92	TRON	135	INKEY$
7	BRKPT	50	NEXT	93	TRACE	136	INOUT$
8	CLEAR	51	OFF	94	USING	137	INSTR
9	CLOSE	52	ON	95	VAL	138	INT
10	CLS	53	OPEN	96	VARPTR	139	INTRB
11	COM	54	OPTION	97	WAIT	140	INTRS
12	CONT	55	PC	98	WATCH	141	INTRR
13	DATA	56	PGEN	99	WEND	142	LEFT$
14	DEF	57	PINF	100	WHILE	143	LEN
15	DEFDBL	58	PMEM	101	WRITE	144	LOC
16	DEFINT	59	PNAME	102		145	LOG
17	DEFLNG	60	POKE	103		146	MID$
18	DEFSNG	61	PRG	104	AND	147	MODEL
19	DEFSTR	62	PRINT	105	EQV	148	OCT$
20	DEL	63	PUT	106	IMP	149	PEEK
21	DIM	64	PWORD	107	MOD	150	RIGHT$
22	DTR	65	QREAD	108	NOT	151	RND
23	EDIT	66	QWRITE	109	OR	152	SEARCH
24	EGET	67	RANDOM	110	XOR	153	SGN
25	ELSE	68	READ	111	ABS	154	SIN
26	END	69	REM	112	ACOS	155	SPACE$
27	ENDIF	70	RENUM	113	ASC	156	SPC
28	EPUT	71	RESET	114	ASIN	157	SQR
29	ERASE	72	RESTORE	115	ATN	158	SSTACK
30	ERC	73	RESUME	116	CDBL	159	STR$
31	ERROR	74	RETURN	117	CHR$	160	STRING$
32	FN	75	ROMLOAD	118	CINT	161	TAB
33	FOR	76	ROMSAVE	119	CLNG	162	TAN
34	GET	77	ROMVERIFY	120	COS	163	TIME$
35	GO	78	RTS	121	CSNG	164	TIME
36	HELP	79	RUN	122	CTS	165	STRING
37	IF	80	SAVE	123	DATA$	166	STACK
38	INPUT	81	SET	124	DAY	167	WDT
39	KEY	82	STEP	125	DMA	168	BYTE
40	LENGTH	83	STOP	126	DSR	169	CODE
41	LET	84	SUB	127	EOF	170	HEAD
42	LIB	85	SYSBACK	128	ERL	171	TERM

付－9 索引

記号

2線／4線選択スイッチ 2-4
@IF 7-35

A

ABS 7-35
ACOS 7-36
ALARM ON/OFF/STOP 7-36
ASC 7-37
ASCIIビジーフラグ 3-8
ASIN 7-37
ATN 7-37
AUTO 7-38

B

BASIC LED 2-3
BASIC RUN状態フラグ 3-8
BASICで使う文字 7-18
BITON/BITOFF 7-38
BRKPT 7-39

C

CDBL 7-39
CHR$ 7-40
CINT 7-40
CLEAR 7-41
CLNG 7-41
CLOSE 7-42
CLS 7-42
COM ON/OFF/STOP 7-43
CONT 7-43
COS 7-44
CSNG 7-44
CTS 7-44

D

DATA 7-45
DATE$ 7-45
DAY 7-46
DEF FN 7-46
DEFINT/DEFLNG/DEFSNG/DEFDBL/DEFSTR ... 7-47
DEL 7-47
DIM 7-48
DMA 7-49
DMAデータ送信 4-6
DSR 7-49
DTR 7-49

E

EDIT 7-50
EERR LED 2-3
END 7-51
EOF 7-51
ERC 7-52
ERL 7-52
ERR 7-53
ERROR 7-53
ERR1／2／T LED 2-3
EXP 7-54

F

FCS 7-54
FIX 7-55
FOR TO STEP/NEXT 7-56
FRE 7-57

G

GOSUB 7-57
GOTO 7-58

H

HELP 7-58
HEX$ 7-59

I

IF 7-60
INKEY$ 7-60
INPUT$関数 4-10
INPUT命令 4-8
INPUT 7-61
INPUT$ 7-62
INSTR 7-62
INT 7-63
INTRB 7-63
INTRR 7-64
INTRS 7-64
IORD／IOWRエリア ..... 1-7、3-12
IORD(#00□□) ..... 6-16、6-29
IORD(#FD00) ..... 6-14、6-19、6-29
IORD命令 ..... 6-29、6-31

IORD要求フラグ 3-8
IOWR(#00□□) ..... 6-15、6-18、6-26
IOWR(#CC00) ..... 6-15、6-16、6-21、6-23、6-27
IOWR(#EC00) ..... 6-28、9-17
IOWR(#FD00) ..... 6-13、6-19、6-25
IOWR命令 ..... 6-25、6-31
IOWR要求フラグ 3-9

K

KEY ON/OFF/STOP 7-65

L

LEFT$ 7-65
LEN 7-65
LET 7-66
LINE INPUT 7-66
LIST 7-67
LLIST 7-68
LOAD 7-69
LOC 7-7-69
LOG 7-7-70
LPRINT 7-71

M

MID$ ..... 7-72、7-73
MODEL 7-73

N

NEW 7-74

O

OCT$ 7-75
ON ALARM 7-76
ON COM 7-77
ON <数値式> 7-80
ON ERROR命令 ..... 4-18、7-79
ON KEY 7-81
ON PC 7-82
ON TIME$ 7-84
ON TIMER 7-85
OPEN命令 ..... 4-2、7-86
OPTION BASE 7-89
OPTION LENGTH 7-89

P

PC EGET ..... 6-11、6-18、7-91
PC EPUT ..... 6-12、6-18、7-92
PC GET ..... 6-10、6-18、7-93
PC ON/OFF/STOP 7-90
PC PUT ..... 6-10、6-18、7-93
PC QREAD ..... 6-13、6-19
PC QREAD@ 7-94
PC QWRITE ..... 6-14、6-19
PC QWRITE@ 7-97
PC READ ..... 6-6、6-17、7-99
PC READ@ 7-102
PC READ/PCWRITE命令使用時の転送チャネル数 3-6
PC READ/PCWRITE命令使用時の転送データエリア種別 3-6
PC READ/PCWRITE命令使用時の転送開始チャネル 3-6
PC READ@ ..... 6-8、6-17
PC READ要求フラグ 3-5
PC WRITE ..... 6-7、6-17、7-104
PC WRITE@ ..... 6-9、6-17、7-107
PC WRITE要求フラグ 3-5
PC割込 6-20
PC割込みfailフラグ 3-8
PC割込サブルーチン完了フラグ 3-8
PC割込フラグ ..... 3-5、6-20、6-22
PEEK 7-109
PGEN 7-109
PINF 7-110
PMEM 7-111
PNAME 7-111
POKE 7-112
PRINT 7-113
PRINT命令 ..... 4-5、7-11
PWORD 7-115

R

RANDOM 7-115
READ 7-116
REM 7-116
RENUM 7-117
RESTORE 7-118
RESUME 7-119
RIGHT$ 7-119
RND 7-120
ROMLOAD 7-120
ROMSAVE 7-121
ROMVERIFY 7-121

RS-232C 2-8
RS-422A／485 2-10
RTS 7-122
RUN 7-122
RUN LED 2-3

S

SAVE 7-123
SEARCH 7-123
SGN 7-124
SIN 7-124
SPACE$ 7-124
SPC 7-125
SQR 7-125
STEP 7-126
STOP 7-126
STR$ 7-127
STRING$ 7-127

T

TAB 7-128
TAN 7-128
TIME$ 7-129
TIME$ ON/OFF/STOP 7-129
TIMER ON/OFF/STOP 7-129
TRACE 7-130
TROFF 7-130
TRON 7-131
T／R1／2／T LED 2-3

V

VAL 7-131
VARPTR 7-132

W

WAIT 7-133
WATCH 7-134
WHILE/WEND 7-135

ア

エラー(異常)履歴 9-16
エラーコード新旧種別 3-8
エラーコード読み出し 3-5
エラー種別／コード ..... 9-2～9-8

カ

関係式 7-23
関数 7-16
型宣言文字 7-20
共有メモリ 1-7
行番号 7-17
号機No.スイッチ 2-4
コマンド 7-16
コマンドモード 7-16
コメント 7-18

サ

算術式 7-23
式 7-22
時刻合わせフラグ 3-5
システム／BASICエラー 3-8
システム設定エリア ..... 1-7、3-10
実数型 7-19
終端抵抗スイッチ 2-4
数値式 7-22
数値定数 7-18
数値変数 7-20
スタート／ストップスイッチ 2-4
スタティック方式 7-12
ステートメント 7-16
ステップ機能 7-138
整数型 7-18

タ

ターミナルとの接続 2-11
ターミナルポート異常フラグ 3-8
ダイナミック方式 7-13
単精度型 7-19
定数 7-18
デバイス記号(周辺装置名) 4-4
データメモリへの割付 3-10
伝送制御信号 ..... 4-4、4-6、4-9、4-11～4-17
トレース機能 7-139

ナ

内部補助リレーへの割付け 3-2

ハ

倍精度型 7-19
配列変数 7-20
バッテリ ..... 2-5、9-18
バッテリ異常フラグ 3-8
フラッシュROM 7-9
ブレイクポイント機能 7-137
プログラムモード 7-16
文 7-18
変数 7-20
変数名 7-20
変数モニタ機能 7-138
ポート1/2異常フラグ 3-8

マ

文字式 7-23
文字定数 7-18
文字変数 7-20

ヤ

ユーザ定義関数 7-136
ユーザメモリデフォルトスイッチ 2-5

ラ

ラベル 7-17
論理式 7-24

ワ

割込機能 5-4
割込番号 3-6

生産中止(予定)機種一覧

アプリケーションの重複する機種や、ニーズに合わなくなり、ここ数年ご注文がない機種につきましては整理・統合を実施させていただいています。今回生産中止対象機種をご紹介します。

■生産中止(予定)機種

商品名称	商品形式
C200H CPU**ユニット**	**形**C200H-CPU21
	形C200H-CPU22
	形C200H-CPU23
	形C200H-CPU31
C200HS CPU**ユニット**	**形**C200HS-CPU01
	形C200HS-CPU01-C
	形C200HS-CPU01-E
	形C200HS-CPU03
	形C200HS-CPU21
	形C200HS-CPU21-C
	形C200HS-CPU21-E
	形C200HS-CPU23
	形C200HS-CPU31
	形C200HS-CPU33
CS1 CPU**ユニット**	**形**CS1G-CPU42
	形CS1G-CPU43
	形CS1G-CPU44
	形CS1G-CPU45
	形CS1H-CPU63
	形CS1H-CPU64
	形CS1H-CPU65
	形CS1H-CPU66

代替機種につきましては、別途お問い合わせください。

●本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途への使用をご検討の場合は、定格・性能に対し余裕を持った使い方やフェールセイフ等の安全対策へのご配慮をいただくとともに、当社営業担当者までご相談いただき仕様書等による確認をお願いします。
●本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物（又は技術）に該当するものを輸出（又は非居住者に提供）する場合は同法に基づく輸出許可、承認（又は役務取引許可）が必要です。

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー 営業統轄事業部
東京都品川区大崎1-11-1 ゲートシティ大崎ウエストタワー14F（〒141-0032）

●営業にご用の方も、技術お問い合わせの方も、フリーコールにお電話ください。
音声ガイダンスが流れますので、案内に従って操作ください。

フリーコール **0120-919-066**（クイック / オムロン）

携帯電話・PHSなどではご利用いただけませんので、その場合は下記電話番号へおかけください。
電話 055-982-5015（通話料がかかります）

【技術のお問い合わせ時間】
■営業時間：9:00～12:00/13:00～19:00
（土・日・祝祭日は9:00～12:00/13:00～17:00）
■営 業 日：年末年始を除く
上記フリーコール以外に、055-977-6389（通話料がかかります）
におかけいただくことにより、直接FAシステム機器の技術窓口につながります。

【営業のお問い合わせ時間】
■営業時間：9:00～12:00/13:00～17:30（土・日・祝祭日は休業）
■営 業 日：土・日・祝祭日/春期・夏期・年末年始休暇を除く

●FAXによるお問い合わせは下記をご利用ください。
お客様相談室 FAX 055-982-5051

●その他のお問い合わせ先
納期・価格・修理・サンプル・承認図は貴社のお取引先、
または貴社担当オムロン営業員にご相談ください。

インターネット情報サービス
オムロンFA機器の最新情報がご覧いただけます。
Industrial Webホームページ http://www.fa.omron.co.jp/

標準在庫機種の緊急ご購入の際にご利用ください。
オムロンツーフォーサービス株式会社
受注センター TEL:03-5825-2324 http://www.omron24.co.jp/

オムロン商品のご用命は

このマニュアルは古紙配合率100%の再生紙を使用しています

Man. No. SCCC-324H 2006年 3月現在

© OMRON Corporation 1998 All Rights Reserved.
お断りなく仕様などを変更することがありますのでご了承ください