Net Express

言語リファレンス　－　追加トピック

目　次

第1章：報告書作成

第2章：通信機能単位の概要

第3章：デバッグ機能単位

第4章：区分化

第5章：2バイト文字支援機能

第6章：2バイト文字支援機能の Micro Focus 拡張

第12章: Ryan McFarland COBOL V2.0 構文支援機能

第13章: Data General Interactive COBOL V1.3 構文支援機能

第1章: 報告書作成

報告書作成（report writer）は特殊な目的の機能であり、出力報告書の編成や形式や内容に焦点を当てている。もちろん、標準のCOBOL言語を使用して報告書を作成することもできるが、報告書作成言語機能を利用すると、もっと簡潔に報告書を構成し作成できる。報告書作成機能を使用すると、通常はプログラマが行う手続き部のプログラミングの多くが、自動的に報告書作成制御システム（report writer control system : RWCS）によって行われるようになる。したがって、プログラマは、いろいろな手続きを書く負担が少なくなる。たとえば、データの転記、印刷する行の編集、1ページ中の行数を数える、ページ番号の付番、見出し行や脚書き行の作成、論理的なデータの切れ目の判定、合計カウンタの更新などを、いちいちコーディングしなくて済むようになる。これらの操作はすべて、原始プログラムのデータ部の報告書節中に記述する文に基づいて、報告書作成制御システムによって行われる。

(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、報告書作成機能単位は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの機能単位を使用するべきではない。

(OSVS)OS/VS COBOLコンパイラは、ANSI'68標準に基いた報告書作成機能をサポートする（メインフレームLANGLVL(2)コンパイラオプションを使用する場合でも - これは普通ANSI'74サポートを必要とする)。しかし、コンパイラが２つのCOBOL標準をサポートしている場合が多くある。このため、他の方言でOS/VS COBOLをエミュレートするCOBOLシステムの使用を試みるべきではない。

(OSVS)この章では、これらのOS/VS COBOL拡張の多くはOSVSのマークで示してある。しかし、OS/VS COBOL 報告書作成の構文に関する本来の規則については、*IBM OS/VS COBOL Language Reference* を参照すること。

1.1 報告書節

COBOLデータ部の報告書節（report section）には、報告書記述項（report description entry）がいくつか含まれる。それらの報告書記述項が、全体で1つの報告書の記述を構成する。

報告書記述項中の報告書名は、出力ファイルには直接は割り当てられない。その代わりに、報告書名はファイル節中のファイル名に関連付けられ、そのファイル名を指定したOPEN文が実行されるときに、そのファイルが出力ファイルに関連付けられる。同じファイル名に、2つ以上の報告書を関連付けることができる。ファイル内で別々の報告書を区別するには、CODE句を使用する。ファイル名によって参照される外部ファイル結合子に関しては、同じファイル名に対してプログラムが別であれば、別の報告書を指定できる。ファイル記述項には、そのファイルを使用するプログラムによって作成される各報告書用の報告書記述項の名前を指定しなければならない。

報告書記述項には、報告書に名前を付け、印刷ページの形式および各種の報告集団を印字する行位置の範囲に関する情報を指定するための、一連の句を記述する。報告書記述項中に識別コードを指定して、中間出力ファイル中の各報告書を個別に識別するようにできる。

各報告書記述項に続けて、01レベルの記述項をいくつか記述する。さらに、各01レベルの記述項に続けて、レコード記述に似た階層を記述する。各01レベルの記述項とその下位の記述項は、報告集団（report group）を表わす。各報告集団は、何行かの印字行から構成される。 報告集団は、1つの単位として扱われる。つまり、1つの報告集団は完全に1論理ページ内に収まるように印字され、ページをまたがって印字されることはない。

1.1.1 報告書の構造

報告書の構造を決める際に検討する重要なこととして、縦および横の間隔の取り方、データの操作、報告集団の論理的および物理的な行位置の範囲が挙げられる。

1.1.1.1 縦の位置決め

1つの報告集団には、複数の行を含めることができる。ページ上の行の縦の位置は、各行に対応するLINE NUMBER句によって指定する。NEXT GROUP句は、ある報告集団の最後の行を印刷した後、次の報告集団との間を何行空けるかを指定する。次の報告集団の最初のLINE NUMBER句は、その報告集団を位置付けるのにさらに余分に空ける間隔を指定する。

1.1.1.2 横の位置決め

報告書を作成する際に、報告書行上のデータ・フィールドの位置を指定することができる。それには、COLUMN NUMBER句を使用する。定義したすべてのフィールドの間は、報告書作成制御システムによって空白にされる。

1.1.1.3 データの操作

報告書作成機能を使用するときは、報告集団へのデータの移送は手続き部の文ではなく、報告書節中の句によって行われる。このような、データの操作に影響を及ぼす報告書節中の句には、SOURCE, SUM, VALUEがある。

SOURCE（源）句は、暗黙のMOVE（転記）文の送出し側のデータ項目を指定する。受取り側の印字用項目は、SOURCE句の指定を含めた報告集団項目の記述によって定義する。

SUM（合計）句は、合計カウンタを自動的に設定する。SUM句の右辺には、GENERATE（作成）文が実行されたときに合計カウンタに加算するデータ項目を指名する。合計カウンタの内容を受取り側の印字用項目（SUM句を含める報告集団項目の記述によって定義する）に転記することは、報告集団を印字するときに自動的に行われる。

VALUE（値）句は、印字項目に定数を設定する。報告集団を印字するたびに、VALUE句を指定した印字項目の内容として、設定した定数が表示される。

まとめると、報告集団中のデータ項目は、表示位置を示すCOLUMN NUMBER句を指定した場合にだけ表示される。印字用項目中に入れる値は、報告集団記述項中に記述するSOURCE句、SUM句、VALUE句によって設定する。報告集団の印字項目が、手続き部の文によって直接的に値を設定されることはない。

1.1.2 報告集団の配置

ページ上に何を表示するかを決定するために、報告書の物理構成と論理構成とが相互に関係する。

1.1.2.1 物理的な報告集団の配置

PAGE句はページの長さ、見出し領域と脚書き領域の大きさ、明細行を表示する領域の大きさを指定する。報告書作成制御システムは、LINE NUMBER句とNEXT GROUP句に基づいて、報告集団を位置付ける。また、必要なときには、改ページを行い、ページ頭書き（page heading）報告集団およびページ脚書き（page footing）報告集団を自動的に作成する。

1.1.2.2 論理的な報告集団の配置

明細（detail）報告集団を、何レベルかの入れ子状の制御集団（control group）から構成することができる。各制御集団の先頭には制御頭書き（control heading）報告集団を配置し、末尾には制御脚書き（control footing）報告集団を配置できる。

入れ子状の制御集団を定義したとき、制御階層（control hierarchy）中の制御データ項目（control data item）の値の変化を認識することを、制御切れ（control break）と言い、制御データ名（control data-name）に関連する頭書きと脚書を、それぞれ制御頭書きと制御脚書きと言う。

GENERATE文の実行中に、報告書作成制御システム（RWCS）は制御階層に基づいて、制御切れの発生を自動的に検査する。制御切れが発生すると、それよりも低いレベルのすべての制御切れが発生したものとみなされる。実際には低いレベルの制御データ項目の値は変わっていなくても、制御切れが発生したことになる。制御切れが発生すると、下記の処理が順に行われる。

1. 最下位のレベルから制御切れが検出されたレベルまでの、すべての制御脚書き報告集団が順に表示される。
2. 制御切れが検出されたレベルから最下位のレベルまでの、すべての制御頭書き報告集団が順に表示される。
3. GENERATE文に指定した明細報告集団が表示される。

1.1.2.3 報告書作成手続き文

手続き部に記述する報告書作成文には、INITIATE, GENERATE, TERMINATE, SUPPRESS, USE BEFORE REPORTINGがある。

INITIATE（開始）文は、報告書作成制御システム（RWCS）に一連の初期化機能を自動的に実行させる。報告書の明細処理を行う前に、報告書の開始処理を行っておく。

データ名を指定したGENERATE文は、指定した明細報告集団の形式を整えて出力装置に書き出す。その他に、GENERATE文は上記の多くの暗黙の処理を報告書作成制御システム（RWCS）に実行させるきっかけとなる。

報告書名を指定したGENERATE文は、合計報告書を作成する。この型の文では、明細行はすべて自動的に抑制されて、 明細報告集団の処理の間に集計された合計だけが印字される。報告書名を指定したGENERATE文とデータ名を指定したGENERATE文は、報告書作成制御システム（RWCS）によってほぼ同様に処理される。違いは、報告書名を指定したGENERATE文では、明細行が印字されないことである。

TERMINATE（終了）文は、報告書の終了に関する機能をすべて自動的に実行するよう、報告書作成制御システム（RWCS）に指示する。報告書を収録したファイルを閉じる前に、TERMINATE文を実行しなければならない。.

SUPPRESS（抑制）文は、ある報告集団全体を実行時に印刷しないようにする。

BEFORE REPORTING指定をしたUSE（使用）文は、報告書作成制御システム（RWCS）によって自動的に実行される手続き内の特定の時点で、手続き部の文を実行できるようにする。USE BEFORE REPORTING手続き内の文は、SOURCE句によって参照されるデータ項目の内容を変更できる。したがって、自動的に作成される報告集団内で参照されるデータ項目の内容を制御することができる。

1.2 言語の概念

1.2.1 報告書ファイル

報告書ファイル（report file）は、順編成の出力ファイルである。報告書ファイルのファイル記述項には、REPORT句が含まれる。報告書ファイルへのレコードの書出しは、報告書作成制御システム（RWCS）によって行われる。

報告書ファイルは、ファイル管理記述項で名前を付け、ファイル記述項で仕様を記述する。報告書ファイル用のファイル記述には、REPORT句を含める。 報告書ファイルを参照し呼び出す文には、OPEN, GENERATE, INITIATE, SUPPRESS, TERMINATE, USE BEFORE REORTING, CLOSEがある。

1.2.2 特殊レジスタPAGE-COUNTER

予約語のPAGE-COUNTERは、ページカウンタの名前である。このページカウンタは、データ部の報告書節中の各報告書記述項ごとに生成される。このデータ項目は、暗黙のうちにPIC 9(6) BINARYと定義される。PAGE-COUNTERの値は報告書作成制御システム（RWCS）によって維持され、報告書のページに番号を振るためにプログラムによって使用される。PAGE-COUNTERを参照できるのは、報告書節のSOURCE句の中と手続き部の文の中だけである。(後述の*PAGE-COUNTERに関する規則*の節を参照。)

1.2.3 特殊レジスタLINE-COUNTER

予約語のLINE-COUNTERは、行数カウンタの名前である。この行数カウンタは、データ部の報告書節中の各報告書記述項ごとに生成される。このデータ項目は、暗黙のうちにPIC 9(6) BINARYと定義される。LINE-COUNTERの値は報告書作成制御システム（RWCS）によって維持され、報告書の 縦の位置付けを決めるために使用される。LINE-COUNTERを参照できるのは、報告書節のSOURCE句の中と手続き部の文の中だけである。LINE-COUNTERの値を変更できるのは、報告書作成制御システム（RWCS）だけである。(後述の*LINE-COUNTERに関する規則*の節を参照。)

1.2.4 特殊レジスタPRINT-SWITCH

(OSVS)予約語のPRINT-SWITCHは、レジスタの名前である。このレジスタの値を、USE BEFORE REPORTING宣言手続きの中でゼロ以外に設定できる。こうすると、対応する報告集団の印刷を抑制する効果がある。

1.2.5 添字付け

報告書節では、添字として、合計カウンタも特殊レジスタの LINE-COUNTER と PAGE-COU NTER も使用できない。

1.3 報告書作成機能単位における環境部

1.3.1 入出力節

入出力節に関する情報は、***言語リファレンス***の*プログラムの定義*の章を参照のこと。

1.3.2 ファイル管理段落

ファイル管理段落に関する情報は、***言語リファレンス***の*プログラムの定義*の章を参照のこと。

機能

ファイル管理記述項には、報告書ファイルに関連する物理属性を定義する。

追加の構文規則

1. OPTIONAL指定は、拡張モードで開いた報告書に対してだけ適用できる。実行用プログラムを実行するときに、毎回必ずしも存在するとは限らない報告書ファイルには、OPTIONAL指定をする必要がある。
2. 報告書ファイルは、順編成ファイルとする。SELECT句に指定した各報告書ファイルに関して、同じプログラムのデータ部のファイル記述項にREPORT句を含めなければならない。

1.3.3 入出力管理段落

入出力管理段落に関する情報は、 ***言語リファレンス***の*プログラムの定義*の章を参照のこと。

追加の構文規則

1. RECORD指定を伴うSAME句中に、報告書ファイルを含めてはならない。

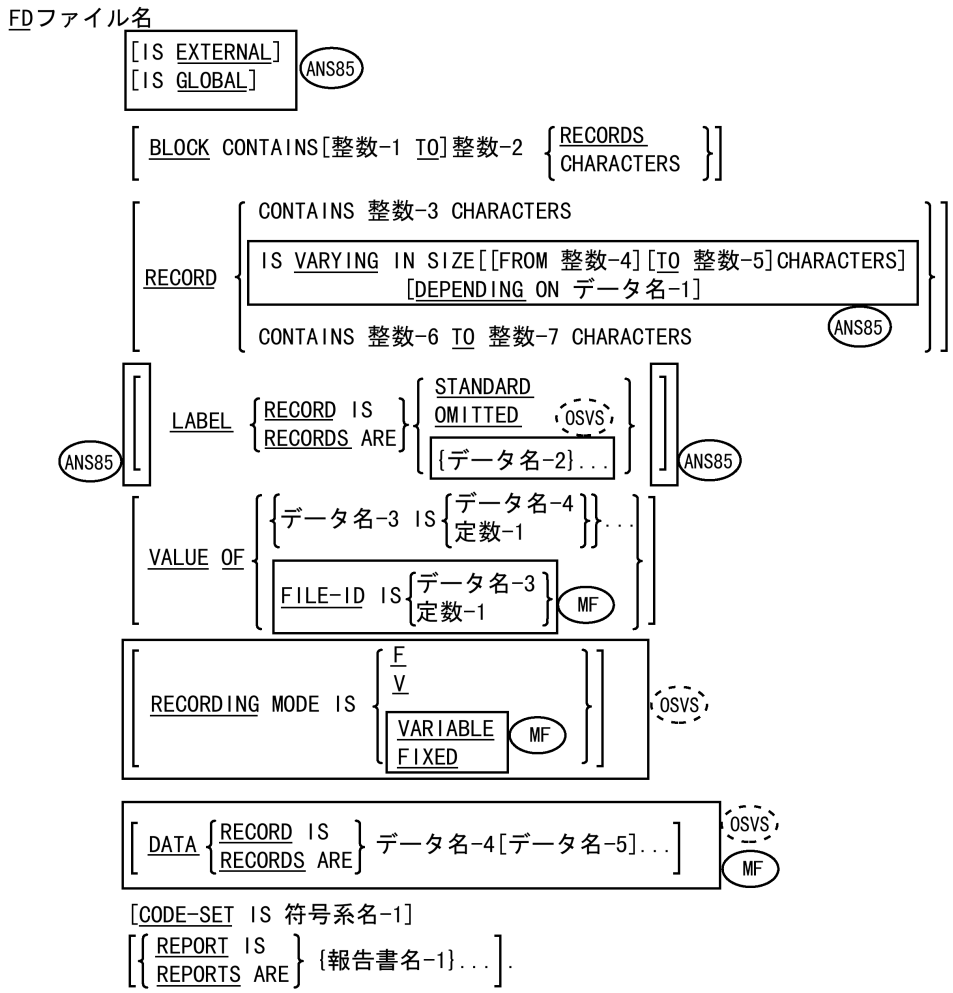
1.4 報告書作成機能単位におけるデータ部

1.4.1 ファイル記述項

ファイル記述項には、対象とするファイルに関する物理構造や識別やレコード名に関する情報を記述する。

(ANS85)ファイル記述項には、ファイル結合子や関連するデータ・レコードや関連するデータ項目の内部属性および外部属性を定義する。また、ファイル名が局所名であるか大域名であるかを、ファイル記述項に指定する。

一般形式



構文規則

1. レベル指示語のFDは、報告書ファイル用のファイル記述項の始まりを示す。報告書ファイルのファイル名の前に置く。
2. ファイル名-1に続く句は、どのような順序で書いてもよい。
3. ファイル名-1のファイルは、順ファイルとする。
4. 報告書ファイル用のファイル記述項の後ろに、レコード記述項を書くことはできない。(OSVS) (MF)レコード記述項を書いてもよい。
5. REPORT句が含まれるファイル記述項の左辺を参照できる手続き文は、USE文、CLOSE文、OUTPUT指定またはEXTEND指定を伴うOPEN文だけである。

一般規則

1. REPORT句を除いて、報告書ファイル用のファイル記述項内のすべての句は、順入出力モジュールで説明してある。
2. REPORT句については、次の項で説明する。

1.4.2 REPORT句

REPORT句は、報告書ファイルを構成する報告書の名前を指定する。

一般形式

```
{REPORT IS  }
{REPORTS ARE} {報告書名-1}...
```

構文規則

1. REPORT句中に指定した各報告書名を、同じプログラムの報告書節中の報告書記述項の左辺に指定しなければならない。報告書名を書く順序には意味はない。
2. 1つの報告書名は、ただ1つのREPORT句の中だけに現れなければならない。 (ANS85)1つの報告書名が、2つのREPORT句の中に現れてもよい。
3. REPORT句が含まれるファイル記述項の左辺を参照できる手続き文は、USE文、CLOSE文、OUTPUT指定またはEXTEND指定を伴うOPEN文だけである。

一般規則

1. 1つのREPORT句の中に複数の報告書名があることは、そのファイルに複数の報告書が含まれることを意味する。
2. INITIATE文を実行した後で同じ報告書ファイルに対してTERMINATE文を実行する前までの間、その報告書ファイルは報告書作成制御システム（RWCS）の制御下におかれる。報告書ファイルが報告書作成制御システム（RWCS）の制御下にある間は、その報告書ファイルを参照する入出力文を実行することはできない。

3. (ANS85)関連するファイル結合子が外部ファイル結合子である場合、そのファイル結合子に関連する実行単位中の各ファイル記述項において、そのファイル結合子を報告書ファイルとして記述する。

1.4.3 報告書節

報告書節は、原始プログラムのデータ部の中に位置する。報告書節には、報告書ファイルに書き出す報告書の仕様を記述する。各報告書の記述は報告書記述項で始め、その後ろに1つ以上の報告集団記述項を続ける。

一般形式

1.4.3.1 報告書記述項

報告書に名前を付けることに加えて、報告書記述項では報告書の各ページの 形式を定義する。具体的には、各型の報告集団を印字できる行位置の範囲を指定する。また、報告書記述項には制御データ項目も指定する。報告書を作成するときに、制御データ項目の値が変わるごとに、報告書の明細情報が制御集団という単位で処理される。

ファイル節のファイル記述項のREPORT句中の各報告書名は、報告書節の報告書記述項の左辺にする。さらに、報告書節中の各報告書はただ1つのファイル記述項だけに指名されているのでなければならない。

1.4.3.2 報告集団記述項

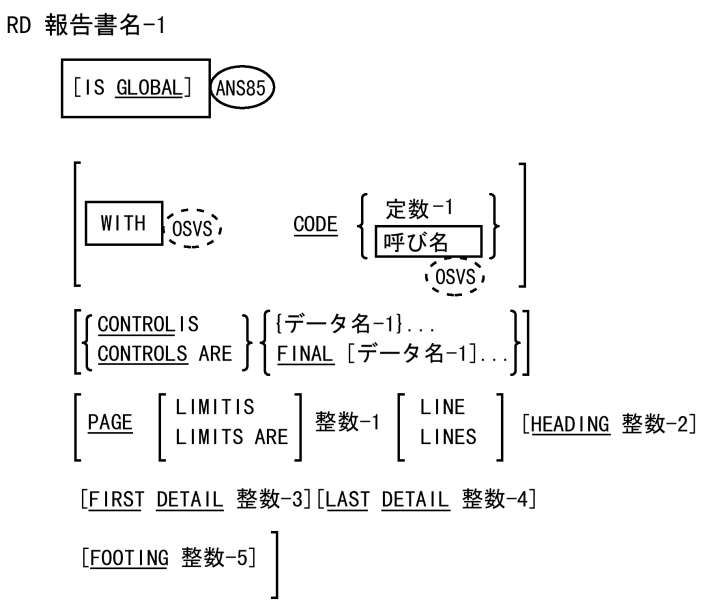
報告書記述項に続けて、 その報告書を構成する報告集団を記述する。各報告集団の先頭には、報告集団記述項を記述する。これは、レベル番号が01でTYPE句を含む記述項である。報告集団記述項の下位には、その報告集団の特性をさらに記述する、集団記述項および基本記述項を続けることができる。

1.4.4 報告書記述項

報告書記述項では、報告書に名前を付け、報告書中の各印刷行に付与する識別特性を指定し、報告書の物理構造と構成を記述する。

(ANS85)また報告書記述項には、報告書名が局所名であるか大域名であるかも指定する。

一般形式



構文規則

1. 報告書名-1は、1つのREPORT句の中だけに指定する。
2. 報告書名-1の後ろに続ける句の順序は重要ではない。
3. LINE-COUNTER、PAGE-COUNTER、報告書節の中で定義するすべてのデータ名に指定できる修飾語のうちで、報告書名は最高位を占める。

一般規則

1. CODE句、CONTROL句、PAGE句は、アルファベット順にそれぞれ説明する。
2. (ANS85)報告書記述項にGLOBAL句が含まれている場合、特殊レジスタのLINE-COUNTERとPAGE-COUNTERは大域名である。

1.4.5 PAGE-COUNTERに関する規則

1. PAGE-COUNTERは、ページ数を表わす特殊レジスタを指す予約語である。この特殊レジスタは、報告書節中に指定した各報告書に対して自動的に作成される。(前述の *特殊レジスタPAGE-COUNTER*の節を参照。)
2. 報告書節では、PAGE-COUNTERを参照できるのはSOURCE句の中でだけである。手続き部では、整数値をとるデータ項目を置くことのできる場所ならば、どこでもPAGE-COUNTERを使用できる。
3. 1つのプログラム中にPAGE-COUNTERが複数ある場合は、手続き部で参照するたびに、報告書名によってPAGE-COUNTERを修飾しなければならない。

報告書節では、修飾しないでPAGE-COUNTERを参照すると、その記述がされている報告書記述項に対応する報告書の名前によって、暗黙的に修飾される。別の報告書のPAGE-COUNTERを参照するときは、対象とする報告書名でPAGE-COUNTERを明示的に修飾しなければならない。

4. INITIAL文を実行すると、対象報告書のPAGE-COUNTERが報告書作成制御システム（RWCS）によって、1に設定される。
5. 報告書作成制御システム（RWCS）によって改ページされるたびに、PAGE-COUNTERは自動的に1繰り上げられる。
6. 手続き部の文によって、PAGE-COUNTERの値を変更できる。

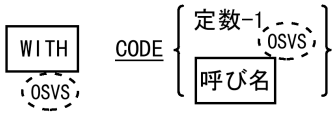
1.4.6 LINE-COUNTERに関する規則

1. LINE-COUNTERは、印字行の行位置（line number）を表わす特殊レジスタを指す予約語である。この特殊レジスタは、報告書節中に指定した各報告書に対して自動的に作成される。(前述の*特殊レジスタLINE-COUNTER* の節を参照。)
2. 報告書節においては、LINE-COUNTERを参照できるのは、SOURCE句の中でだけである。手続き部では、整数値をとるデータ項目を置くことのできる場所ならばどこでもLINE-COUNTERを使用できる。ただし、LINE-COUNTERの内容を変更できるのは、報告書作成制御システム（RWCS）だけである。
3. 1つのプログラム中にLINE-COUNTERが複数ある場合は、手続き部で参照するたびに、報告書名によってLINE-COUNTERを修飾しなければならない。
   報告書節では、修飾しないでLINE-COUNTERを参照すると、その記述がされている報告書記述項に対応する報告書の名前によって、暗黙的に修飾される。別の報告書のLINE-COUNTERを参照するときは、対象とする報告書名でLINE-COUNTERを明示的に修飾しなければならない。
4. INITIATE文を実行すると、対象報告書のLINE-COUNTERが報告書作成制御システム（RWCS）によってゼロに設定される。また、改ページされるたびに、LINE-COUNTERの値が報告書作成制御システムによって自動的にゼロに設定し直される。
5. LINE-COUNTERの値は、印字されることのない報告集団を処理しても、影響を受けることはない。また、本来なら印字されるがSUPPRESS文によって印字を抑制されている報告集団を処理しても、影響を受けることはない。
6. 報告書行が印字されるとき、LINE-COUNTERの値は、その位置を示す行位置を表わしている。報告集団を印字した後のLINE-COUNTERの値は、報告集団に関する表示規則によって規定される。(後述の*表示規則表*を参照。)

1.4.7 CODE（コード）句

CODE句は、各報告書行がどの報告書に属するかを示す、2文字の定数を指定する。

一般形式



構文規則

1. 定数-1は、2文字の英数字定数とする。
   (OSVS)定数-1は、特殊名段落中に指定した呼び名で置き換えられなければならない。詳細は、 *OS/VS COBOL Language Reference*を参照。
2. あるファイル中のどれかの報告書に対してCODE句を指定した場合、そのファイル中のすべての報告書にCODE句を指定しなければならない。

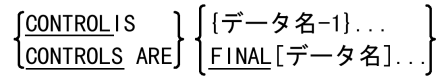
一般規則

1. CODE句を指定すると、各報告書の論理レコードの最初の2文字の文字位置に定数-1が自動的に書き込まれる。
2. 定数-1によって占められる文字位置は、印字行の記述には含まれないが、論理レコードの大きさには含まれる。

1.4.8 CONTROL（制御）句

CONTROL句は、報告書の制御階層のレベルを確立する。

一般形式



構文規則

1. データ名-1は、報告書節に定義してあってはならない。データ名-1は、修飾してもよい。
2. データ名-1をいくつか並べて指定する場合、それぞれを別のデータ項目にする。
3. データ名-1の下位に、可変反復データ項目が含まれていてはならない。

一般規則

1. データ名-1と語 FINALによって、制御階層のレベルを指定する。FINALを指定すると、それが最も高いレベルの制御であることを表わす。データ名-1をいくつか並べて指定すると、左から右に順々に制御レベルが低くなる。

2. ある報告書に関して、GENERATE文が最初に実行されると、その報告書に関するすべての制御データ項目の値が、報告書作成制御システム（RWCS）によって保存される。それ以降、GENERATE文が実行されるたびに、制御データ項目の値が変わったかどうかが報告書作成制御システム（RWCS）によって検査される。 どれか制御データ項目の値が変わると、制御切れが発生する。この制御切れのレベルは、値の変化があった最もレベルの高い制御データ項目に対応する。後述の*GENERATE*文の節を参照。)
3. 制御切れが発生したかどうか判定するために、報告書作成制御システム（RWCS）は、各制御データ項目の内容を、その報告書に関して前回GENERATE文が実行されたときに保存された各制御データ項目の内容と比較する。このとき、下記の規則が適用される。
    a. 制御データ項目が数字データ項目である場合、2つの数字作用対象に関する比較検査が適用される。
    b. 制御データ項目が指標付きデータ項目である場合、2つの指標付きデータ項目の比較検査が適用される。
    c. 制御データ項目が上記の3a.にも3b.にも該当しない場合、2つの文字作用対象に関する比較検査が適用される。

    (比較検査の詳細については、***言語リファレンス***の*プログラムの定義*の章を参照。)
4. FINALは、制御データ項目では表わせない報告書全体の制御切れを指定するときに、使用する。

1.4.9 PAGE（ページ）句

PAGE句は、ページの長さと 報告集団を表示する縦の範囲を指定する。

一般形式

```
PAGE [ LIMITIS      ] 整数-1 [ LINE  ] [HEADING 整数-2]
     [ LIMITS ARE   ]        [ LINES ]
     [FIRST DETAIL 整数-3][LAST DETAIL 整数-4]
     [FOOTING 整数-5]
```

構文規則

1. HEADING, FIRST DETAIL, LAST DETAIL, FOOTING の各指定は、どのような順序で書いてもよい。
2. 整数-1は、有効桁数が3桁を超えてはならない。
3. 整数-2の値は、0以上とする。
4. 整数-3の値は、整数-2の値以上とする。
5. 整数-4の値は、整数-3の値以上とする。
6. 整数-5の値は、整数-4の値以上とする。
7. 整数-1の値は、整数-5の値以上とする。

8. PAGE句を指定すると、各型の報告集団が表示される縦の範囲には、下記の規則が適用される。(後述の*ページ領域*の節を参照。)
    a. 報告書頭書き報告集団を1ページに排他的に（明細行と共存しないように）表示する場合、整数-2から整数-1までの行位置の範囲に、その報告書頭書き報告集団が収まるように定義する。
    報告書頭書き報告集団を1ページに非排他的に（明細行と共存するように）表示する場合、整数-2から整数-3の1行上までの行位置の範囲に、その報告書頭書き報告集団が収まるように定義する。
    b. ページ頭書き報告集団を表示する場合、整数-2から整数-3の1行上までの行位置の範囲に、そのページ頭書き報告集団が収まるように定義する。
    c. 制御頭書き報告集団または明細報告集団を表示する場合、整数-3から整数-4までの行位置の範囲に、それらの報告集団が収まるように定義する。
    d. 制御脚書き報告集団を表示する場合、整数-3から整数-5の1行上までの行位置の範囲に、その制御脚書き報告集団が収まるように定義する。
    e. ページ脚書き報告集団を表示する場合、整数-5の1行下から整数-1までの行位置の範囲に、そのページ脚書き報告集団が収まるように定義する。
    f. 報告書脚書き報告集団を1ページに排他的に（明細行と共存しないように）表示する場合、整数-2から整数-1までの行位置の範囲に、その報告書脚書き報告集団が収まるように定義する。
    報告書脚書き報告集団を1ページに非排他的に（明細行と共存するように）表示する場合、整数-5の1行下から整数-1までの行位置の範囲に、その報告書脚書き報告集団が収まるように定義する。
9. すべての報告集団は、1ページ内に表示しきれるように定義する。報告書作成制御システム（RWCS）は、複数行からなる報告集団をページをまたがって表示することは決してない。

一般規則

1. 報告書ページ中の各型の報告集団の縦の位置付けは、PAGE句中に指定した整数の値によって決められる。
    a. 整数-1は、報告書ページの大きさを決めるものであり、1ページ中の有効な行数を表わす。
    b. HEADING 整数-2は、報告書頭書き報告集団またはページ頭書き報告集団を表示する最初の行位置を定義する。
    c. FIRST DETAIL 整数-3は、本体集団を表示する最初の行位置を定義する。報告書頭書きもページ頭書きも、整数-3の行位置以降には表示できない。
    d. LAST DETAIL 整数-4は、制御頭書き報告集団または明細報告集団を表示する最後の行位置を定義する。
    e. FOOTING 整数-5は、 制御頭書き報告集団を表示する最後の行位置を定義する。ページ脚書き報告集団および報告書脚書き報告集団は、整数-5の行位置よりも後に続かなければならない。

2. PAGE句を指定するときにその指定要素を省略すると、下記のように暗黙的に値が設定される。
    a. HEADING指定を省略すると、整数-2の値として1が想定される。
    b. FIRST DETAIL指定を省略すると、整数-3の値は整数-2に等しいものとされる。
    c. LAST DETAIL指定とFOOTING指定を両方とも省略すると、整数-4および整数-5の値はともに整数-1に等しいものとされる。
    d. FOOTING指定をしLAST DETAIL指定を省略すると、整数-4の値は整数-5に等しいものとされる。
    e. LAST DETAIL指定をしFOOTING指定を省略すると、整数-5の値は整数-4に等しいものとされる。
3. PAGE句を省略すると、報告書は不定な長さの単一のページから構成されることになる。
4. 報告集団に関する表示規則は、その型ごとに分けて後述する。（後述の*表示規則表*を参照。）

1.4.10 ページ領域

PAGE句によって設定されるページ領域を、表1-1にまとめる。

表 1-1: ページ領域

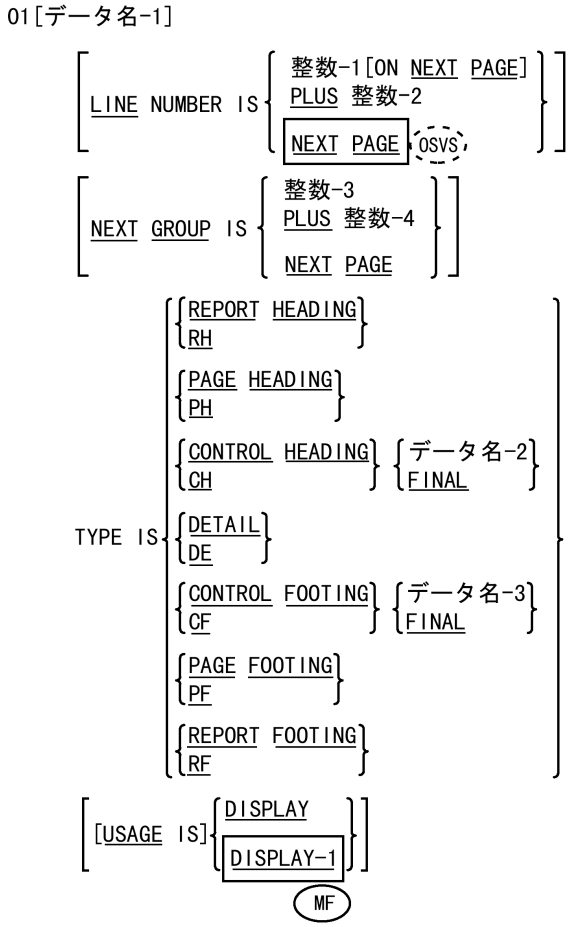
領域に表示される報告集団	領域の最初の行位置	領域の最後の行位置
NEXT GROUP NEXT PAGEを指定した報告書頭書き LINE 整数-1 NEXT PAGEを指定した報告書脚書き	整数-2	整数-1
NEXT GROUP NEXT PAGEを指定しない報告書頭書き ページ頭書き	整数-2	整数-3マイナス1
制御頭書き 明細	整数-3	整数-4
制御脚書き	整数-3	整数-5
ページ脚書き LINE 整数-1 NEXT PAGEを指定しない報告書脚書き	整数-5プラス1	整数-1

1.4.11 報告集団記述項

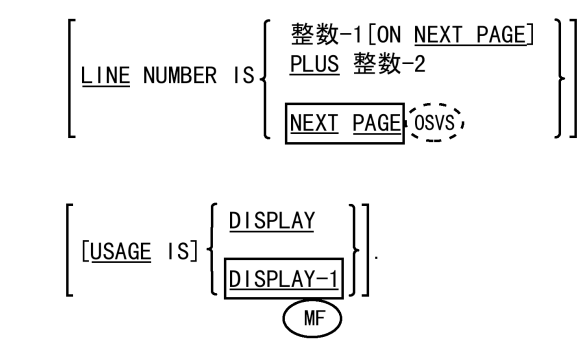
報告集団記述項は、報告集団およびその中の個々の項目の特性を指定する。

一般形式

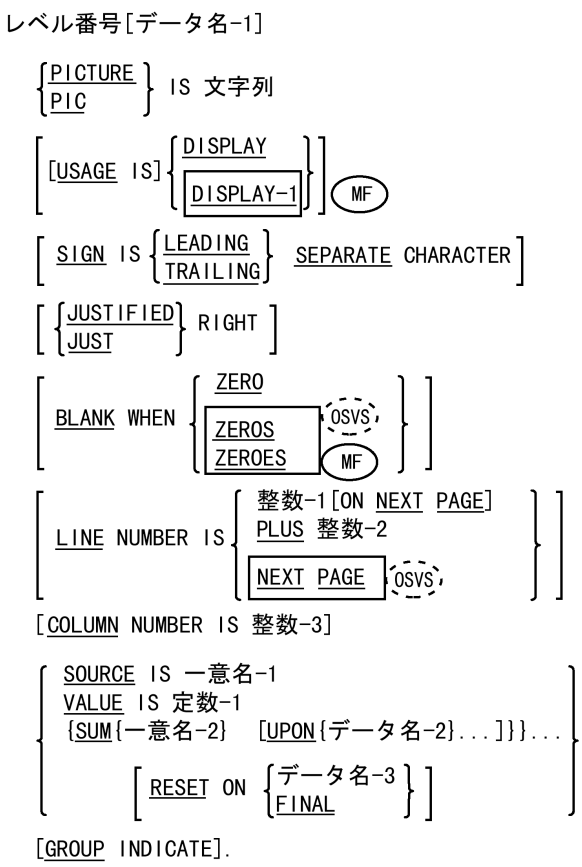
書き方 1



書き方 2



書き方 3



構文規則

すべての書き方

1. 報告集団記述項は、報告書節にだけ記述できる。
2. データ名句は、レベル番号の直後に書く。それ以外の句はどの順序で書いてもよい。
3. 報告集団の記述は4レベル以内の階層から構成される。
    a. 報告集団を記述する最初の記述項は、書き方1とする。
    b. 書き方1の記述項の直後に、書き方2または書き方3の記述項を書くことができる。
    c. 書き方2の記述項の直後に、書き方3の記述項を少なくとも1つ書く。
    d. 書き方3の記述項では、基本データ項目を定義する。
4. 報告書節では、USAGE句を使用するのは、印字項目の用途を宣言するためだけである。
    a. 書き方3の記述項内にUSAGE句を書く場合、その記述項では印字項目を定義する。

b. 書き方1または書き方2の記述項内にUSAGE句を書く場合、その下位の記述項の少なくとも1つで印字項目を定義する。

5. LINE NUMBER句が含まれる記述項の下位に、LINE NUMBER句を含む記述項を含めてはならない。
   (OSVS)しかし、LINE NUMBER NEXT PAGE句が含まれる記述項の下位に、NEXT PAGE指定を伴わないLINE NUMBER句を含む記述項を含めてもよい。

書き方 1

6. データ名-1は以下の場合に必要である。
   a. GENERATE文によって、詳細報告集団を参照する。
   b. SUM文のUPON指定 によって、詳細報告集団を参照する。
   c. USE BEFORE REPORTING 文によって、詳細報告集団を参照する。
   d. 制御脚書き報告集団を、合計カウンタの修飾に使用する。

   データ名-1を書いた場合、それを参照できるのは、GENERATE文、SUM句のUPON指定、USE BEFORE REPORTING文、合計カウンタの修飾語だけである。

書き方 2

7. レベル番号には02以上48以下の任意の整数を使用できる。
8. 記述項には、少なくとも1つの選択句を含める。
9. データ名-1は書いても書かなくてもよい。データ名-1を書いた場合、それを使用できるのは、合計カウンタを修飾するためだけである。

書き方 3

10. レベル番号には02以上49以下の任意の整数を使用できる。
11. 書き方3では、以下の規則がある。
    a. GROUP INDICATE句は、明細報告集団中にだけ書ける。
    b. SUM句は、制御脚書き報告集団中にだけ書ける。
    c. COLUMN NUMBER句が含まれるがLINE NUMBER句は含まれない記述項は、LINE NUMBER句が含まれる記述項の下位に属していること。
    d. データ名-1は書いても書かなくてもよく、またどの記述項に書いてもよい。データ名-1を書いた場合、それを参照できるのは、その記述項で合計カウンタを定義している場合だけである。
    e. VALUE句が含まれる記述項には、COLUMN NUMBER句も含めること。
12. 書き方3の記述項で許されるすべての句の組合わせを、表に示す。この表は行に沿って左から右に読む。
    「必」は、その句が必須であることを示す。

「可」は、その句を書いてもよいが、必ずしも必要ではないことを示す。
「×」は、その句を書いてはならないことを示す。

表 1-2: 書き方3で許される句の組合せ

PIC	COLUMN	SOURCE	SUM	VALUE	JUST	BLANK WHEN ZERO	GROUP INDICATE	USAGE	SIGN	LINE
必	×	×	必	×	×	×	×	×	可	可
必	必	×	必	×	×	可	×	可	可	可
必	可	必	×	×	可	×	可	可	可	可
必	可	必	×	×	×	可	可	可	可	可
必	必	×	×	必	可	×	可	可	可	可

一般規則

1. 書き方1は、報告集団記述項用である。この記述項およびその下位のすべての記述項の内容によって、報告書が定義される。
2. 報告書機能機能単位におけるBLANK WHEN ZERO句、JUSTIFIED句、PICTURE句はそれぞれ、中核におけるものと同じである。（***言語リファレンス***の*プログラムの定義*の章を参照。）報告集団記述項のその他の句については、この後でアルファベット順に個々に説明する。

1.4.12 表示規則表

1.4.12.1 目的

下記の事項の仕様を表わす表と規則を以降に示す。

1. 報告集団の各型ごとに、許されるLINE NUMBER句とNEXT GROUP句の組合わせ。
2. それらの句を使用する際の必要条件。
3. 報告書作成制御システム（RWCS）によるそれらの句の解釈。

1.4.12.2 構成

報告書頭書き報告集団、ページ頭書き報告集団、ページ脚書き報告集団、報告書脚書き報告集団に関しては、個々に表示規則表を示す。明細報告集団、制御頭書き報告集団、制御脚書き報告集団に関しては、本体集団の表示規則表の中に一緒に示す。(後述の*本体集団の表示規則表*の節を参照。)

表示規則表の第1欄と第2欄には、該当する報告集団の型に関する、LINE NUMBER句とNEXT GROUP句の可能なすべての組合わせを示す。したがって、LINE NUMBER句とNEXT GROUP句の1つの組合わせに関しては、表示規則表の該当する行を読めばよい。

表示規則表の適用規則の欄は、大きく2つの部分に分けてある。最初の部分には、報告書記述にページ句が含まれる場合の規則を示す。2番目の部分には、報告書記述にページ句が含まれない場合の規則を示す。適用規則の具体的な項目名と目的を下に示す。

1. 上限と下限の規則
   これらの規則は、該当する報告集団を表示できるページ内の縦の範囲を規定する。
   ページ句を指定しなかった場合、印字される報告書は、報告集団別に縦の配置を規定しないものとみなされる。したがって、ページ句を省略した報告書記述に関しては、表の中に上限の規則も下限の規則も示されない。
2. 改ページの規則
   改ページの規則は、本体集団に対してだけ適用される。したがって、改ページの規則は、本体集団用の表示規則表の中にだけ示してある。報告書作成制御システム（RWCS）は、実行時に改ページの規則を適用して、その報告書の現在の位置から指定された本体集団をそのページ内に表示しきれるかどうかを判定する。
   報告書記述項にページ句を指定しなかった場合は、本体集団であっても改ページ規則は適用されない。
3. 印字開始行位置の規則
   印字開始行位置の規則は、報告書作成制御システム（RWCS）が、報告集団の最初の行を報告書媒体上のどこに表示するかを規定する。
   報告集団の2行目以降がある場合でも、報告書作成制御システム（RWCS）がそれを報告書媒体上のどこに表示するについては規定していない。報告集団の2行目以降をどこに表示するかは、一般的な規則によって定まる。（詳細については、LINE NUMBER句の一般規則を参照。）
4. 次の報告集団の規則
   次の報告集団の規則は、NEXT GROUP句の使用方法を規定する。
5. LINE-COUNTERの最終値設定の規則
   LINE-COUNTERの最終値設定の規則は、報告集団を表示した後で、報告書作成制御システム（RWCS）がLINE-COUNTERに設定する最終の値を規定する。

1.4.12.3 LINE NUMBER（行位置）句の表記法

表示規則表の第1欄には、LINE NUMBERの書き方を示す。これには記号を用いている。その意味は下記のとおり。

1. "A" はNEXT PAGE指定をしないで、絶対的な行位置を指定することを意味する。この形式のLINE NUMBER句は、1つの報告集団記述項の中で連続して書いてもよい。

2. "R" は相対的な行位置を指定することを意味する。この形式のLINE NUMBER句は、1つの報告集団記述項の中で連続して書いてもよい。
3. "NP" はNEXT PAGE指定をして絶対的な行位置を指定することを意味する。この形式のLINE NUMBER句は、1つの報告集団記述項の中で連続して書いてもよい。ただし、NEXT PAGE指定は最初のLINE NUMBER句だけに付けること。
   (OSVS)NEXT PAGEは、行位置を示さずに指定してもよい。

記号を2つ並べて示してある場合は、一連のLINE NUMBER句を2組続けて書くことを示す。たとえば、 "A R" は一連の "A" （上記の1に規定）の直後に一連の"R"（上記の2に規定）を続けることを意味する。

1.4.12.4 LINE NUMBER句の表記法の部分適用

表示規則表中に"A R" が許されるものとして示してある場合、"A" も許される。この場合、同じ表示規則が適用される。

表示規則表中に "NP R" が許されるものとして示してある場合、"NP"も許される。この場合、同じ表示規則が適用される。

1.4.12.5 次の報告集団用の整数待避項目

次の報告集団用の整数待避項目は、報告書作成制御システム（RWCS）だけが扱えるデータ項目である。NEXT GROUP句に現在のページに適用できない絶対的な行位置を指定すると、その値は報告書作成制御システム（RWCS）によって、次の報告集団用の整数待避項目に記憶される。改ページ処理が終わった後で、報告書作成制御システム（RWCS）は次の報告集団用の整数待避項目に記憶されている値を使用して、次の本体集団の位置付けを行う。

1.4.12.6 報告書頭書き報告集団の表示規則

報告書頭書き報告集団中で許されるLINE NUMBER句とNEXT GROUP句のすべての組合わせに適用される表示規則を、表1-3に示す。

1. 上限の規則
   報告書頭書き報告集団を表示できる最初の行位置は、ページ句の中のHEADING指定で設定した行位置である。
2. 下限の規則
   a. 報告書頭書き報告集団を表示できる最後の行位置は、ページ句の中のFIRST DETAIL指定の整数-3の値から1を引いた値となる。
   b. 報告書頭書き報告集団を表示できる最後の行位置は、ページ句の中の整数-1に指定した行位置である。

3. 印字開始行位置の規則
    a. 報告書頭書き報告集団の最初の印字行の行位置は、LINE NUMBER句の整数に指定した行位置である。
    b. 報告書頭書き報告集団の最初の印字行位置は、最初のLINE NUMBER句の整数に指定した値と、ページ句の中のHEADING指定の整数-2の値から1を引いた値とを加えた値となる。
    c. 報告書頭書き報告集団を表示しない。
    d. 報告書頭書き報告集団の最初の印字行位置は、LINE-COUNTERの内容（この場合は、ゼロ）と最初のLINE NUMBER句の整数の値とを加えた値となる。
4. 次の報告集団の規則
    a. NEXT GROUP句に指定する整数は、報告書頭書き報告集団の最後の印字行の行位置よりも大きく、ページ句のFIRST DETAIL指定の整数-3に指定した行位置よりも小さくなければならない。
    b. NEXT GROUP句に指定する整数と報告書頭書き報告集団の最後の印字行の行位置との和は、ページ句のFIRST DETAIL指定の整数-3に指定した行位置よりも小さくなければならない。
    c. NEXT GROUP句にNEXT PAGE指定をすると、報告書の最初のページには、報告書頭書き報告集団だけを表示することを意味する。報告書の最初のページに位置している間は、報告書作成制御システム（RWCS）は他の報告集団を処理しない。
5. LINE-COUNTERの最終値設定の規則
    a. 報告書頭書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、NEXT GROUP句の整数をLINE-COUNTERに入れる。
    b. 報告書頭書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、NEXT GROUP句の整数と報告書頭書き報告集団の最後の印字行の行位置との和をLINE-COUNTERに入れる。
    c. 報告書頭書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、ゼロをLINE-COUNTERに入れる。
    d. 報告書頭書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、報告書頭書き報告集団の最後の印字行の行位置をLINE-COUNTERに入れる。
    e. 印字項目が含まれない報告集団の処理によって、LINE-COUNTERが影響を受けることはない。

表1-3: 報告書頭書き報告集団の表示規則表

**		当てはまる規則***						
		PAGE句あり					PAGE句なし	
LINE NUMBER句*	NEXT GROUP句	上限	下限	印字開始	行位置	LINE-COUNTERの最終値設定	印字開始行位置	LINE-COUNTERの最終値設定
AR	絶対	1	2a	3a	4a	5a	無効な組合わせ	
AR	相対	1	2a	3a	4b	5b	無効な組合わせ	
AR	NEXT PAGE	1	2b	3a	4c	5c	無効な組合わせ	
AR	指定しない	1	2a	3a	規則なし	5d	無効な組合わせ	
R	絶対	1	2a	3b	4a	5a	無効な組合わせ	
R	相対	1	2a	3b	4b	5b	3b	5b
R	NEXT PAGE	1	2b	3b	4c	5c	無効な組合わせ	
R	指定しない	1	2a	3b	規則なし	5d	3d	5d
指定しない	指定しない	規則なし	規則なし	3c	規則なし	5e	3c	5e

* この欄での表記法については、前述の *LINE NUMBER句の表記法*

** 欄1または2 の「指定しない」は、指定した句が報告集団記述項にまったくないことを示す。

*** 規則の欄の「規則なし」は、LINE NUMBER句とNEXT GROUP 句の組み合わせに対する規則がないことを示す。

\+ *LINE NUMBER句*を参照。

++ *NEXT GROUP句*を参照。

1.4.12.7 ページ頭書き報告集団の表示規則

ページ頭書き報告集団中で許されるLINE NUMBER句とNEXT GROUP句のすべての組合わせに適用される表示規則を、表1-4に示す。

表1-4 ： ページ頭書き報告集団の表示規則表

**		当てはまる規則***				
		PAGE句あり****				
LINE NUMBER句*	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	LINE-COUNTERの最終値設定
AR	指定しない	1	2	3a	規則なし	4a
R	指定しない	1	2	3b	規則なし	4a
指定しない	指定しない	規則なし	規則なし	3c	規則なし	4b

* この欄での表記法については、前述の *LINE NUMBER句の表記法*を参照。

** 欄1または2 の「指定しない」は、指定した句が報告集団記述項にまったくないことを示す。

*** 規則の欄の「規則なし」は、LINE NUMBER句とNEXT GROUP 句の組み合わせに対する規則がないことを示す。

**** 報告書記述項にPAGE句を指定しない場合、ページ頭書き報告集団は定義できない。(後述の*TYPE*句の節を参照。)

表中の番号に対応させて、ページ頭書き報告集団の表示規則を以下に具体的に説明する。

1. 上限の規則
   報告書頭書き報告集団と同じページ上にページ頭書き報告集団を表示する場合は、ページ頭書き報告集団を表示できる最初の行位置は、報告書頭書き報告集団を表示したときに設定されたLINE-COUNTERの最終値に1を加えた値となる。
   これ以外の場合は、ページ頭書き報告集団を表示できる最初の行位置は、PAGE句のHEADING指定に指定した行位置となる。
2. 下限の規則
   ページ頭書き報告集団を表示できる最後の行位置は、ページ句の中のFIRST DETAIL指定の整数-3の値から1を引いた値となる。
3. 印字開始行位置の規則
   a. ページ頭書き報告集団の最初の印字行の行位置は、LINE NUMBER句の整数に指定した行位置である。
   b. 報告書頭書き報告集団と同じページ上にページ頭書き報告集団を表示する場合は、ページ頭書き報告集団の最初の印字行位置は、報告書頭書き報告集団を表示したときに設定されたLINE-COUNTERの最終値にページ頭書き報告集団の最初のLINE NUMBER句の整数の値を加えた値となる。
      これ以外の場合は、ページ頭書き報告集団の最初の印字行位置は、ページ頭書き報告集団の最初のLINE NUMBER句の整数の値と、ページ句の中のHEADING指定の整数-2の値から1を引いた値とを加えた値となる。
   c. ページ頭書き報告集団を表示しない。
4. LINE-COUNTERの最終値設定の規則

a. ページ頭書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値となる。
b. 印字項目が含まれない報告集団の処理によって、LINE-COUNTERが影響を受けることはない。

1.4.12.8 本体集団の表示規則

制御頭書き報告集団と明細報告集団と制御脚書き報告集団の中で許されるLINE NUMBER句とNEXT GROUP句のすべての組合わせに適用される表示規則を、表1-5に示す。表中の番号に対応させて、本体集団の表示規則を以下に具体的に説明する。

1. 上限の規則
   本体集団を表示できる最初の行位置は、PAGE句の中のFIRST DETAIL指定で設定した行位置である。
2. 下限の規則
   制御頭書き報告集団または明細報告集団を表示できる最後の行位置は、PAGE句の中のLAST DETAIL指定で設定した行位置である。
   制御脚書き報告集団を表示できる最後の行位置は、PAGE句の中のFOOTING指定で設定した行位置である。

3. 改ページの規則
   a. LINE-COUNTERの値が最初の絶対LINE NUMBER句の整数の値よりも小さい場合、報告書の現在のページ上に本体集団が表示される。
      LINE-COUNTERの値が最初の絶対LINE NUMBER句の整数の値以上である場合、報告書作成制御システム（RWCS）は改ページ処理を実行する。ページ頭書き報告集団が定義されていれば報告書作成制御システムはその処理を行った後で、前のページに最後に本体集団を表示したときに次の報告集団用の整数待避項目が設定されたか否かを判定する(LINE-COUNTERの最終値の設定規則6 a. を参照）。

      次の報告集団用の整数待避項目が設定されていなければ、報告書の現在のページ上に本体集団が表示される。次の報告集団用の整数待避項目が設定されていれば、報告書作成制御システム（RWCS）は次の報告集団用の整数待避項目をLINE-COUNTERに転記し、次の報告集団用の整数待避項目をゼロに設定し直してから、この規則を再び適用する。

b. 報告書の現在のページに既に本体集団が表示されている場合、報告書作成制御システム（RWCS）は試験的に合計値を算出する。この試験的な合計値は、LINE-COUNTERの値に報告集団のすべてのLINE NUMBER句の整数の値を加えたものである。試験的な合計値が本体集団の下限を表わす整数値を超えない場合は、現在のページ上に報告集団が表示される。試験的な合計値が本体集団の下限を表わす整数値を超える場合は、報告書作成制御システム（RWCS）は改ページ処理を実行する。ページ頭書き報告集団が定義されていれば、報告書作成制御システムはこれを処理してから、この規則を再び適用する。
報告書の現在のページにまだ本体集団が表示されていない場合、報告書作成制御システム（RWCS）は、前のページ上に最後に本体集団を表示したときに次の報告集団用の整数待避項目が設定されたか否かを判定する（LINE-COUNTERの最終値の設定規則6 a.を参照）。

次の報告集団用の整数待避項目が設定されていなければ、報告書の現在のページ上に本体集団が表示される。

次の報告集団用の整数待避項目が設定されていれば、報告書作成制御システム（RWCS）は次の報告集団用の整数待避項目をLINE-COUNTERに転記し、次の報告集団用の整数待避項目をゼロに設定し直してから、試験的な合計値を算出する。

この試験的な合計値は、LINE-COUNTERの値に1を加え、さらに本体集団のすべてのLINE NUMBER句のうち最初のものを除いた整数の値を加えたものである。試験的な合計値が本体集団の下限を表わす整数値を超えない場合は、現在のページ上に報告集団が表示される。試験的な合計値が本体集団の下限を表わす整数値を超える場合は、報告書作成制御システム（RWCS）は改ページ処理を実行する。ページ頭書き報告集団が定義されていれば、報告書作成制御システム（RWCS）はこれを処理してから、そのページに本体集団を表示する。

c. 報告書の現在のページに本体集団が既に表示されている場合、報告書作成制御システム（RWCS）は改ページ処理を実行する。ページ頭書き報告集団が定義されていれば、報告書作成制御システムはこれを処理してから、この規則を再び適用する。

d.
報告書の現在のページにまだ本体集団が表示されていない場合、報告書作成制御システム（RWCS）は、前のページ上に最後に本体集団を表示したときに次の報告集団用の整数待避項目が設定されたか否かを判定する（LINE-COUNTERの最終値の設定規則6 a.を参照）。

次の報告集団用の整数待避項目が設定されていなければ、報告書の現在のページ上に本体集団が表示される。次の報告集団用の整数待避項目が設定されていれば、報告書作成制御システム（RWCS）は次の報告集団用の整数待避項目をLINE-COUNTERに転記し、次の報告集団用の整数待避項目をゼロに設定し直す。

次いで、LINE-COUNTERの値が最初の絶対LINE NUMBER句の整数の値よりも小さければ、報告書の現在のページ上に本体集団が表示される。そうでなければ、報告書作成制御システムは改ページ処理を実行する。ページ頭書き報告集団が定義されていれば、それを処理してから、報告書作成制御システムはそのページ上に本体集団を表示する。

4. 印字開始行位置の規則
    a. 本体集団の最初の印字行の行位置は、LINE NUMBER句の整数に指定した行位置である。
    b. LINE-COUNTERの値がPAGE句のFIRST DETAIL指定の行位置以上であり、かつ報告書の現在のページにまだ本体集団が表示されていないと、現在の本体集団の最初の印字行はLINE-COUNTERの値によって示される行の次の行となる。
    LINE-COUNTERの値がPAGE句のFIRST DETAIL指定の行位置以上であり、かつ報告書の現在のページに既に本体集団が表示されていると、現在の本体集団の最初の印字行は、LINE-COUNTERの値と現在の本体集団の最初のLINE NUMBER句の整数の値とを加えた行位置となる。
    LINE-COUNTERの値がPAGE句のFIRST DETAIL指定の行位置よりも小さいと、本体集団の最初の印字行位置は、FIRST DETAILに指定した行位置となる。
    c. 本体集団を表示しない。
    d. 本体集団の最初の印字行の行位置は、LINE-COUNTERの内容と最初のLINE NUMBER句の整数の値とを加えた値となる。

5. 次の報告集団の規則
    絶対NEXT GROUP句に指定する整数は、PAGE句のFIRST DETAILに指定した整数の値以上であり、かつPAGE句のFOOTINGに指定した整数の値以下とする。

6. LINE-COUNTERの最終値設定の規則
    a. 制御脚書き報告集団を表示したところで、その制御脚書き報告集団が一番高いレベルの制御切れに対応していない場合、その制御脚書き報告集団の最後の印字行が表示された行位置がLINE-COUNTERの最終値として設定される。

これ以外の場合には、報告書作成制御システム（RWCS）は本体集団の最後の印字行の行位置とNEXT GROUP句の整数とを比較する。前者の方が後者よりも小さいと、報告書作成制御システム（RWCS）は最終値として、NEXT GROUP句の整数をLINE-COUNTERに入れる。前者の方が後者以上であると、報告書作成制御システム（RWCS）は最終値として、PAGE句のFOOTINGに指定された行位置をLINE-COUNTERに入れる。さらに、報告書作成制御システム（RWCS）はNEXT GROUP句の整数を次の報告集団用の整数待避項目に入れる。

b. 制御脚書き報告集団を表示したところで、その制御脚書き報告集団が一番高いレベルの制御切れに対応しているのではない場合、その制御脚書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値として設定される。

これ以外の場合には、報告書作成制御システム（RWCS）は試験的な合計値を算出する。この試験的な合計値はNEXT GROUP句の整数をその本体集団の最後の印字行の行位置に加えたものである。試験的な合計値がPAGE句のFOOTINGに指定された行位置よりも小さい場合は、報告書作成制御システム（RWCS）は最終値として、その試験的な合計値をLINE-COUNTERに入れる。試験的な合計値がPAGE句のFOOTINGに指定された行位置以上である場合は、報告書作成制御システム（RWCS）は最終値として、PAGE句のFOOTINGに指定された行位置をLINE-COUNTERに入れる。

c. 制御脚書き報告集団を表示したところで、その制御脚書き報告集団が一番高いレベルの制御切れに対応しているのではない場合、その制御脚書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値として設定される。

これ以外の場合には、報告書作成制御システム（RWCS）は最終値として、PAGE句のFOOTINGに指定された行位置をLINE-COUNTERに入れる。

d. 制御脚書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値として設定される。

e. 印字項目が含まれない本体集団の処理によってLINE-COUNTERが影響を受けることはない。

f. 制御脚書き報告集団を表示したところで、その制御脚書き報告集団が一番高いレベルの制御切れに対応しているのではない場合、その制御脚書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値として設定される。

これ以外の場合には、報告書作成制御システム（RWCS）は最終値として、最後の印字行が表示された行位置とNEXT GROUP句の整数との和を、LINE-COUNTERに入れる。

表 1-5: 本体集団の表示規則表

**		当てはまる規則***							
		PAGE句あり						PAGE句なし	
LINE NUMBER句*	NEXT GROUP句	上限	下限	改ページ	印字開始行位置	次の報告集団	LINE-COUNTERの最終値設定	印字開始行位置	LINE-COUNTERの最終値設定
AR	絶対	1	2	3a	4a	5	6a	無効な組合わせ	
AR	相対	1	2	3a	4a	規則なし	6b	無効な組合わせ+	
AR	NEXT PAGE	1	2	3a	4a	規則なし	6c	無効な組合わせ	
AR	指定しない	1	2	3a	4a	規則なし	6d	無効な組合わせ	
R	絶対	1	2	3b	4b	5	6a	無効な組合わせ	
R	相対	1	2	3b	4b	規則なし	6b	4d	6f
R	NEXT PAGE	1	2	3b	4b	規則なし	6c	無効な組合わせ	
R	指定しない	1	2	3b	4b	規則なし	6d	4d	6f
NP R	絶対	1	2	3c	4a	5	6a	無効な組合わせ+	
NP R	相対	1	2	3c	4a	規則なし	6b	無効な組合わせ+	
NP R	NEXT PAGE	1	2	3c	4a	規則なし	6c	無効な組合わせ+	
NP R	指定しない	1	2	3c	4a	規則なし	6d	無効な組合わせ+	
指定しない	指定しない	規則なし	規則なし	規則なし	4c	規則なし	6e	4c	6e

* この欄での表記法については、前述の *LINE NUMBER句の表記法*を参照。

** 欄1または2 の「指定しない」は、指定した句が報告集団記述項にまったくないことを示す。

*** 規則の欄の「規則なし」は、LINE NUMBER句とNEXT GROUP 句の組み合わせに対する規則がないことを示す。

\+ *LINE NUMBER句*を参照。

++ *NEXT GROUP句*を参照。

1.4.12.9 ページ脚書き報告集団の表示規則

ページ脚書き報告集団中で許されるLINE NUMBER句とNEXT GROUP句のすべての組合わせに適用される表示規則を、表1-6に示す。

表 1-6 ： ページ脚書き報告集団の表示規則表

**		当てはまる規則***				
		PAGE句あり****				
LINE NUMBER句*	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	LINE-COUNTERの最終値設定
AR	絶対	1	2	3a	4a	5a
AR	相対	1	2	3a	4b	5b
AR	指定しない	1	2	3a	規則なし	5c
指定しない	指定しない	規則なし	規則なし	3b	規則なし	5d

* この欄での表記法については、前述の *LINE NUMBER句の表記法*を参照。

** 欄1または2 の「指定しない」は、指定した句が報告集団記述項にまったくないことを示す。

*** 規則の欄の「規則なし」は、LINE NUMBER句とNEXT GROUP 句の組み合わせに対する規則がないことを示す。

**** 報告書記述項にPAGE句を指定しないと、ページ頭書き報告集団は定義できない。(*TYPE* 句の節を参照。)

表中の番号に対応させてページ脚書き報告集団の表示規則を以下に具体的に説明する。

1. 上限の規則
   ページ脚書き報告集団を表示できる最初の行位置は、PAGE句のFOOTING指定の整数-5に1を加えた値である。
2. 下限の規則
   ページ脚書き報告集団を表示できる最後の行位置は、PAGE句の整数-1の値である。
3. 印字開始行位置の規則
   a. ページ脚書き報告集団の最初の印字行の行位置は、そのLINE NUMBER句の整数に指定した行位置である。
   b. ページ脚書き報告集団を表示しない。
4. 次の報告集団の規則
   a. NEXT GROUP句の整数は、ページ脚書き報告集団の最後の印字行を表示した行位置よりも大きくなければならない。さらに、NEXT GROUP句の整数は、PAGE句の整数-1に指定した行位置を超えてはならない。
   b. NEXT GROUP句の整数とページ脚書き報告集団の最後の印字行を表示した行位置との和は、PAGE句の整数-1に指定した行位置を超えてはならない。
5. LINE-COUNTERの最終値設定の規則
   a. ページ脚書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、NEXT GROUP句の整数を、LINE-COUNTERに入れる。

b. ページ脚書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、NEXT GROUP句の整数とページ脚書き報告集団の最後の印字行を表示した行位置との和を、LINE-COUNTERに入れる。
c. ページ脚書き報告集団を表示した後で、報告書作成制御システム（RWCS）は最終値として、ページ脚書き報告集団の最後の印字行を表示した行位置を、LINE-COUNTERに入れる。
d. 印字項目が含まれない報告集団の処理によってLINE-COUNTERが影響を受けることはない。

1.4.12.10 報告書脚書き報告集団の表示規則

報告書脚書き報告集団中で許されるLINE NUMBER句とNEXT GROUP句のすべての組合わせに適用される表示規則を、表1-7に示す。表中の番号と対応させて、報告書脚書き報告集団の表示規則を、以下に具体的に説明する。

1. 上限の規則
   a. 報告書の現在のページ上にページ脚書き報告集団が表示されている場合、報告書脚書き報告集団を表示できる最初の行位置は、ページ脚書き報告集団を表示したときに設定されたLINE-COUNTERの最終値に1を加えたものとなる。
   これ以外の場合、報告書脚書き報告集団を表示できる最初の行位置は、PAGE句の整数-5に1を加えた値である。
   b. 報告書脚書き報告集団を表示できる最初の行位置は、PAGE句の中のHEADING指定で設定した行位置である。
2. 下限の規則:
   報告書脚書き報告集団を表示できる最後の行位置は、PAGE句の中の整数-1に指定した行位置である。
3. 印字開始行位置の規則
   a. 報告書脚書き報告集団の最初の印字行の行位置は、そのLINE NUMBER句の整数に指定した行位置である。
   b. 報告書の現在のページ上にページ脚書き報告集団が表示されている場合、報告書脚書き報告集団の最初の印字行の行位置は、ページ脚書き報告集団を表示したときに設定されたLINE-COUNTERの最終値と、報告書脚書き報告集団の最初のLINE NUMBER句の整数との和となる。これ以外の場合、報告書脚書き報告集団の最初の印字行の行位置は、報告書脚書き報告集団の最初のLINE NUMBER句の整数と、PAGE句のFOOTING指定の整数-5との和となる。
   c. 最初のLINE NUMBER句の中にNEXT PAGE指定があると、他の報告集団が表示されていないページ上に報告書脚書き報告集団を表示することを意味する。この場合、報告書脚書き報告集団の最初の印字行は、そのLINE NUMBER句の整数に指定した行位置に表示される。

d. LINE IS NEXT PAGEが報告集団中の唯一のLINE句である場合、報告書脚書き報告集団の最初の印字行は、PAGE句のHEADING指定に指定した整数-2の行位置に表示される。
e. 報告書脚書き報告集団の最初の印字行の行位置は、LINE-COUNTERの内容と最初のLINE NUMBER句の整数との和となる。
f. 報告書脚書き報告集団を表示しない。

4. LINE-COUNTERの最終値設定の規則
   a. 報告書脚書き報告集団の最後の印字行が表示された行位置が、LINE-COUNTERの最終値とし設定される。
   b. 印字項目が含まれない報告集団の処理によって、LINE-COUNTERが影響を受けることはない。

表 1-7: 報告書脚書き報告集団の表示規則表

**		当てはまる規則***						
		PAGE句あり					PAGE句なし	
LINE NUMBER句*	NEXT GROUP句	上限	下限	印字開始行位置	次の報告集団	LINE-COUNTERの最終値設定	印字開始行位置	LINE-COUNTERの最終値設定
AR	指定しない	1a	2	3a	規則なし	4a	無効な組合わせ	
R	指定しない	1b	2	3c	規則なし	4a	3d	4a
NP R	指定しない	1b	2	3c	規則なし	4a	無効な組合わせ	
指定しない	指定しない	規則なし	規則なし	3e	規則なし	4b	3e	4b

* この欄での表記法については、前述の *LINE NUMBER句の表記法*を参照。

** 欄1または2 の「指定しない」は、指定した句が報告集団記述項にまったくないことを示す。

*** 規則の欄の「規則なし」は、LINE NUMBER句とNEXT GROUP 句の組み合わせに対する規則がないことを示す。

\+ *LINE NUMBER句*の節を参照。

1.4.13 COLUMN NUMBER（印字位置）句

COLUMN NUMBER（印字位置）句は、印字項目を印字行上に表示する位置を指定する。

一般形式

COLUMN NUMBER IS 整数-1

構文規則

1. COLUMN NUMBER句は、報告集団内の基本項目に対してだけ、指定できる。COLUMN NUMBER句を指定する場合、LINE NUMBER句が含まれる記述項の中、またはその下位に属する記述項の中に書く。
2. 1つの印字行内では、印字項目は左のものから右のものへ順に定義しなければならない。このとき、印字項目どうしの印字位置が重ならないようにすること。
   (OSVS)COLUMN NUMBER句中に、印字項目を書く順序に制約を設けない。

一般規則

1. COLUMN NUMBER句は、SOURCE句の右辺、またはVALUE句の右辺、またはSUM句に定義した合計カウンタを印字行上に表示することを示す。COLUMN NUMBER句を指定しないと、項目は印字行上に表示されない。
2. 整数-1は、印字項目の左端の文字位置の印字位置を示す。
3. 報告書作成制御システム（RWCS）は、印字行上で印字項目が表示されないすべての印字位置を空白にする。
4. 印字行の左端を、印字位置1とする。

1.4.14 DATA-NAME（データ名）句

DATA-NAME（データ名）句は、記述対象のデータ項目に名前を付ける。

一般形式

データ名-1

構文規則

1. 報告書節では、データ記述項中に必ずしもデータ名-1を記述する必要はない。

一般規則

1. 報告書節において、下記の場合にはデータ名-1を指定しなければならない。
   a. 手続き部で、GENERATE文またはUSE文によって参照される報告集団を、データ名-1が表わすとき。
   b. 手続き部または報告書節で、合計カウンタを参照するとき。
   c. SUM句のUPON指定中で、明細報告集団を参照するとき。
   d. 合計カウンタを修飾するために、データ名-1が必要なとき。

1.4.15 GROUP INDICATE（集団表示）句

GROUP INDICATE（集団表示）句は、制御切れまたは改ページの後で最初に明細報告集団を表示するときだけ、該当する印字項目を印字することを示す。

一般形式

```
GROUP INDICATE
```

構文規則

1. GROUP INDICATE句を指定できるのは、印字項目を定義している明細報告集団の中だけである。

一般規則

1. GROUP INDICATE句を指定すると、下記の場合を除いて、SOURCE句またはVALUE句は無視されて空白にされる。
    a. 報告書の明細報告集団を最初に表示するとき。
    b. 改ページの後で、明細報告集団を最初に表示するとき。
    c. 制御切れの後で、明細報告集団を最初に表示するとき。
2. 報告書記述項にPAGE句もCONTROL句も指定しないと、GROUP INDICATEを指定した印字項目は、INITIATE文が実行された後で明細報告集団が最初に表示されるときに印字される。その後はSOURCE句またはVALUE句があっても、その項目は空白とされる。

1.4.16 レベル番号

レベル番号は、報告集団の階層構造内でデータ項目が占める位置を示す。

一般形式

```
レベル番号
```

構文規則

1. レベル番号は、各データ記述項の最初の要素として必要である。
2. RD記述項の下位に属するデータ記述項では、レベル番号は01から49の範囲内とする。

一般規則

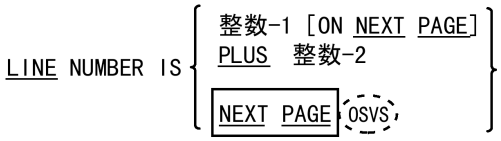
1. レベル番号01は、報告集団内の最初の記述項であることを示す。

2. レベル指示語RDをもつ報告書記述項の下位に01レベルの記述項が複数あっても、それは同じ領域を暗黙的に再定義することを表わさない。

1.4.17 LINE NUMBER（行位置）句

LINE NUMBER（行位置）句は、報告集団に関する縦の位置を指定する。

一般形式



構文規則

1. 整数-1および整数-2は、有効桁数が3桁を超えてはならない。
   PAGE句に指定した報告集団の型ごとの縦の範囲から外れるような値を、整数-1にも整数-2にも指定してはならない。(前述の*PAGE句*の節を参照。)
   整数-2の値は、ゼロであってもよい。
2. LINE NUMBER句が含まれる報告集団記述項の下位に、LINE NUMBER句が含まれる報告集団記述項が属していてはならない。
3. 1つの報告集団記述項の中では、すべての絶対LINE NUMBER句は、すべての相対LINE NUMBER句の前に置く。
4. 1つの報告集団記述項の中では、一連の絶対LINE NUMBER句に指定する整数は、昇順になっていること。これらの整数の値は、連続していなくてもよい。
5. 報告書記述項にPAGE句を指定しなかった報告書に関しては、その中のどの報告集団記述項においても、相対LINE NUMBER句だけしか指定できない。
6. 1つの報告集団記述項においては、 NEXT PAGEは1回だけ指定できる。NEXT PAGEを指定する場合は、その報告集団記述項中の最初のLINE NUMBER句の中に書く。
7. NEXT PAGE指定を伴うLINE NUMBER句は、本体集団または報告書脚書き報告集団の記述項中にだけ含めることができる。
8. 印字項目(前述の*COLUMN NUMBER句*の節を参照。) を定義する各記述項には、LINE NUMBER句を含めるかまたはLINE NUMBER句が含まれる記述項の下位に属させるかする。
9. ページ脚書き報告集団中に指定する最初のLINE NUMBER句は、絶対LINE NUMBER句とする。

一般規則

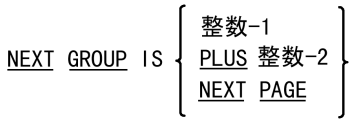
1. 報告集団の各印字行の行位置を指定するために、LINE NUMBER句を書く。

2. 報告書作成制御システム（RWCS）は、LINE NUMBER句によって指定された縦の位置付けを行ってから、印字行を表示する。
3. 整数-1は絶対行位置を表わす。絶対行位置は、印字行を表示する行位置を示す。
4. 整数-2は、相対行位置を表わす。LINE NUMBER句が報告集団記述項中の最初のLINE NUMBER句ではない場合、印字行を表示する行位置は、その報告集団の前の印字行が表示された行位置と相対LINE NUMBER句の整数-2とを加えた値となる。整数-2がゼロであると、前の印字行と同じ行に、次の行が印字される。
   相対LINE NUMBER句が報告集団記述項中の最初のLINE NUMBER句である場合、印字行を表示する行位置は、別途定められた規則によって決められる。（前述の*表示規則表*を参照。）
5. NEXT PAGE句は、次のページの指定した行位置から、該当する報告集団を表示することを表わす。（前述の*表示規則表*を参照。）NEXT PAGE指定には、必ずしも整数-1を指定しなくてよい。

1.4.18 NEXT GROUP（次の報告集団）句

NEXT GROUP（次の報告集団）句は、報告集団の最後の行を表示した後で行う行送りを指定する。

一般形式



構文規則

1. NEXT GROUP句は、少なくとも1つのLINE NUMBER句を含んでいる報告集団記述項に指定する。
2. 整数-1および整数-2は、有効桁数が3桁を超えてはならない。
3. 報告書記述項にPAGE句を指定しなかった報告書に関しては、その中のどの報告集団記述項においても、相対NEXT GROUP句だけしか指定できない。
4. ページ脚書き報告集団中に、NEXT GROUP句のNEXT PAGE指定を含めてはならない。
5. 報告書頭書き報告集団またはページ頭書き報告集団の中に、NEXT GROUP句を指定してはならない。

一般規則

1. NEXT GROUP句によって指定するページの位置付け操作は、その句が含まれる報告集団が表示された後で行われる。(前述の*表示規則表*を参照。)
2. 報告書作成制御システム（RWCS）は、NEXT GROUP句からの行送り情報に加えて、TYPE句とPAGE句の情報とLINE-COUNTERの値を使用して、新しいLINE-COUNTERの値を決定する。(前述の*表示規則表*を参照。)

3. 制御脚書き報告集団中に指定したNEXT GROUP句は、報告書作成制御システム（RWCS）によって無視される。ただし、最高のレベルの制御切れに対応する制御脚書き報告集団の場合は、無視されない。
4. 本体集団のNEXT GROUP句は次の本体集団を指すため、次の本体集団が表示される位置に影響を与える。 報告書頭書き報告集団のNEXT GROUP句は、ページ頭書き報告集団が表示される位置に影響を与える。ページ脚書き報告集団のNEXT GROUP句は、報告書脚書き報告集団が表示される位置に影響を与える。(前述の*表示規則表*を参照。)

1.4.19 SIGN（符号）句

SIGN（符号）句は、演算符号の特性を明示的に記述する必要があるときに、その位置と表現形式を指定する。

一般形式

```
[SIGN IS] { LEADING  } SEPARATE CHARACTER
          { TRAILING }
```

構文規則

1. SIGN句を指定できるのは、PICTURE句に文字"S"が含まれる数字データ記述項だけである。
2. SIGN句を適用する数字データ記述項は、明示的または暗黙的に、USAGE IS DISPLAYと記述されていること。
3. 報告集団記述項にSIGN句を含めるときは、 SEPARATE CHARACTERを指定する。

一般規則

1. SIGN句は、数字データ項目に対して演算符号の位置と表現形式を指定する。SIGN句は、PICTURE句に文字"S"が含まれる数字データ記述項に対してだけ適用される。PICTURE句内の"S" は演算符号が付いていることを表わすが、必ずしもその位置と表現形式を表わしているわけではない。
2. PICTURE句に文字"S" が含まれるがSIGN句が指定されていない数字データ記述項に記述されているデータ項目には、演算符号は付いているが、必ずしもその位置と表現形式が明示されているわけではない。この場合、そのような数字データ項目には、一般規則3は適用されない。（省略時解釈の演算符号の表現については、***言語リファレンス***の*COBOL言語の概念*の章の*文字の表現と基数の選定*の節を参照。）
3. 報告集団記述項中のSIGN句には、SEPARATE CHARACTERを指定する。符号位置は下記のようになる。
    a. 符号が付く位置は、LEADINGを指定したときは左端、TRAILINGを指定したときは右端となる。この演算符号の文字位置は、数字位置とは異なる。

b. PICTURE文字列内の文字"S"は、標準データ形式におけるデータ項目の大きさに含められる。
c. 正と負の演算符号は、それぞれ標準データ形式文字の"+" と "-" である。

4. PICTURE句に文字"S" が含まれる数字データ記述項は、符号付き数字データ項目を表わす。SIGN句の付いているデータ項目に対して計算や比較のために変換が必要になると、この変換は自動的に行われる。

1.4.20 SOURCE（源）句

SOURCE（源）句は、報告集団記述項内に定義されている印字項目に転記される、送り側データ項目を識別する。

一般形式

SOURCE IS 一意名-1

構文規則

1. 一意名-1はデータ部のどこに定義してあってもよい。一意名-1が報告書節内の項目である場合は、下記のどれかとする。
   a. PAGE-COUNTER
   b. LINE-COUNTER
   c. そのSOURCE句が含まれる報告書の一部を構成する合計カウンタ
2. 一意名-1は、報告書作成制御システム（RWCS）によって印字項目に暗黙的に転記される送り側データ項目を示す。一意名-1は、MOVE文の送り側項目に関する規則に適合するように定義されていること。（***言語リファレンス***の*プログラムの定義*の章の*MOVE文*の節を参照。）

一般規則

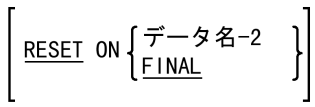
1. 報告書作成制御システム（RWCS）は、報告集団の印字行を編集してから表示する。(後述の*TYPE句*の節を参照。）この編集時に、SOURCE句によって指定された暗黙のMOVE文が、報告書作成制御システムによって実行される。

1.4.21 SUM（合計）句

SUM（合計）句は合計カウンタを設定し、合計の対象とするデータ項目を指定する。

一般形式

{SUM{一意名-1}...[UPON{データ名-1}...]}...



構文規則

1. SUM句が含まれる報告集団記述項の左辺のデータ項目は、数字として定義されていること。一意名-1は、数字データ項目とする。一意名-1が報告書節内で定義されている場合、合計カウンタを参照しなければならない。
   UPON指定を省略した場合、SUM句の中にあり、それ自体が 合計カウンタである一意名は、そのSUM句が含まれる報告集団中またはその報告書の制御階層の中で、それよりも低いレベルの報告集団に定義されていること。
   UPON指定をした場合は、そのSUM句に含まれる一意名は、合計カウンタではならない。
2. データ名-1は、そのSUM句が含まれる 制御脚書き報告集団が含まれる報告書中に記述された、明細報告集団の名前とする。データ名-1を報告書名で修飾してもよい。
3. SUM句は、制御脚書き報告集団の記述中にだけ指定できる。
4. データ名-2は、その報告書のCONTROL句中に指定したデータ名のどれかとする。データ名-2は、RESET指定が含まれる報告集団よりも低い制御階層に属してはならない。
   RESET指定中にFINALを含める場合、その報告書のCONTROL句にもFINALを指定しておく。
   (OSVS)この規則は強制しない。
5. 合計カウンタの修飾語として最もレベルが高いのは、報告書名である。

一般規則

1. SUM句は、合計カウンタを設定する。合計カウンタは、演算符号付きの数字データ項目である。報告書作成制御システム（RWCS）は、実行時に一意名-1の各データ項目の値を合計カウンタに加算する。この加算は算術文の規則に従って行われる。(***言語リファレンス***の*プログラムの定義*の章の*算術文*と*作用対象の重なり*の各節を参照。)
2. 合計カウンタの大きさは、そのSUM句が含まれる基本項目の記述中のPICTURE句によって指定される、受取り側項目の文字数に等しい。
3. 報告書の1つの基本項目には、合計カウンタは1つだけ設定される。1つの基本項目の記述中に何回SUM句を記述しても、設定される合計カウンタは1つである。
4. 報告書の印字用の基本項目の記述項にSUM句を記述すると、合計カウンタが源データ項目となる。報告書作成制御システム（RWCS）は、MOVE文の規則に従って合計カウンタの内容を印字項目に転記してから、その印字項目を表示する。
5. 報告書の基本項目の記述項にSUM句が含まれているとき、その左辺にデータ名を書くと、そのデータ名は合計カウンタの名前となる。このデータ名は、印字項目の名前ではない。

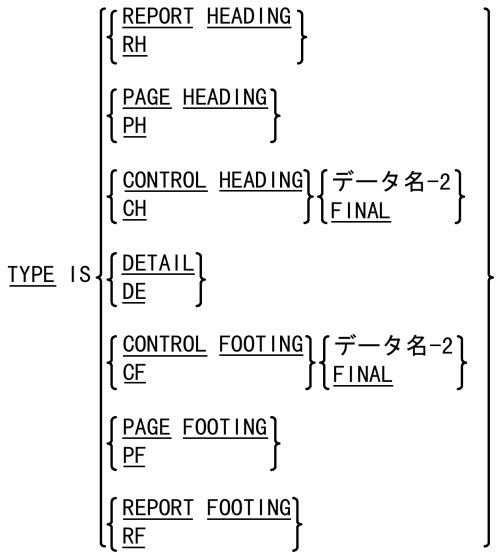
6. 手続き部の文によって、合計カウンタの内容を変更してもよい。
7. 一意名のデータ項目の値を合計カウンタに加える処理は、GENERATE文およびTERMINATE文の実行時に報告書作成制御システム（RWCS）によって行われる。合計カウンタへの加算には、縦の集計（subtotalling）、横の集計（crossfooting）、繰上げ（rolling forward）の3種類ある。縦の集計は、GENERATE文の実行中に制御切れの処理をした後で、明細報告集団を表示する前に行われる。(後述の*GENERATE*文の節を参照。）横の集計と繰上げは、制御脚書き報告集団の処理中に行われる。（後述の*TYPE句*の節を参照。）
8. UPON指定をすると、明細報告集団を指名して、選択的に縦の集計を行うことができる。
9. 報告書作成制御システム（RWCS）は、各加数を合計カウンタに加算する。加算が行われる時点は、加数の種類によって異なる。
    a. 加数が同じ 制御脚書き報告集団内に定義されている合計カウンタであるとき、合計カウンタにその加数を加算することを、横の集計と呼ぶ。
       横の集計は、制御切れが発生し制御脚書き報告集団が処理されるときに行われる。
       横の集計は、制御脚書き報告集団内で合計カウンタが定義されている順序に従って行われる。まず最初に、制御脚書き報告集団内に定義されている、最初の合計カウンタへの加数の加算が行われる。次いで、制御脚書き報告集団内に定義されている、2番目の合計カウンタへの加数の加算が行われる、というぐあいである。
       加数の1つが、そのSUM句が含まれるデータ記述項によって定義された合計カウンタである場合、加算にはその合計カウンタの加算前の値が使用される。
    b. 加数が、下位のレベルの制御脚書き報告集団で定義された合計カウンタであるとき、合計カウンタにその加数を加算することを、繰上げと呼ぶ。制御切れが発生し下位レベルの制御脚書き報告集団が処理されるときに、下位レベルの制御脚書き報告集団中の合計カウンタが、上位レベルの合計カウンタに足し込まれる。
    c. 加数が合計カウンタではないとき、合計カウンタにその加数を加算することを、縦の集計と呼ぶ。SUM句にUPON指定をした場合、指定した明細報告集団用のGENERATE文が実行されるときに、集計が行われる。SUM句にUPON指定をしなかった場合、そのSUM句が含まれる報告書に関してデータ名を指定したすべてのGENERATE文が実行されるたびに、合計カウンタではない加数が集計される。
10. 2つ以上の一意名に同じ加数が指定されていると、その加数はSUM句の中で参照される回数だけ合計カウンタに加算される。2つ以上のデータ名に同じ明細報告集団を指定しても構わない。このような明細報告集団に対するデータ名を指定したGENERATE文が実行されるときに、UPON指定にそのデータ名が指定されている分だけ集計が行われる。
11. 報告書名を指定したGENERATE文が実行されるときに行われる縦の集計については、別途説明する。(後述の*GENERATE*文の節を参照。)

12. RESET指定を省略すると、合計カウンタが定義されている制御脚書き報告集団の処理を行ったきに、報告書作成制御システム（RWCS）はその合計カウンタの値をゼロに設定し直す。 RESET指定をすると、指定されたレベルの制御階層の処理を行ったきに、報告書作成制御システム（RWCS）はその合計カウンタの値をゼロに設定し直す。(後述の*TYPE句*の節を参照。)
SUM句の値は初めに、その合計カウンタが含まれる報告書用のINITIATE文が実行される間に、報告書作成制御システムによってゼロに設定される。

1.4.22 TYPE（報告集団の型）句

TYPE（報告集団の型）句は、報告集団の種類とそれを編集表示する時点を指定する。

一般形式



構文規則

1. RHは、REPORT HEADINGの省略形である。
   PHは、PAGE HEADINGの省略形である。
   CHは、CONTROL HEADINGの省略形である。
   DEは、DETAILの省略形である。
   CFは、CONTROL FOOTINGの省略形である。
   PFは、PAGE FOOTINGの省略形である。
   RFは、REPORT FOOTINGの省略形である。
2. 報告書頭書き報告集団、ページ頭書き報告集団、 FINALを指定した制御頭書き報告集団、 FINALを指定した制御脚書き報告集団、ページ脚書き報告集団、報告書脚書き報告集団は、それぞれ、2回以上書くことはできない。

3. ページ頭書き報告集団およびページ脚書き報告集団を指定できるのは、該当する報告書記述項にPAGE句を指定してある場合だけである。
4. データ名-1、データ名-2、FINALを指定する場合、対応する報告書記述項の CONTROL句の中にも指定しておく。
   (OSVS)FINALに関しては、この規則を強制しない。
   報告書記述項のCONTROL句の中の各データ名およびFINALに、制御頭書き報告集団と制御脚書き報告集団を1つずつ指定できる。しかし、この指定はしなくても構わない。
5. 制御脚書き報告集団、ページ頭書き報告集団、ページ脚書き報告集団、報告書脚書き報告集団では、SOURCE文およびUSE文によって、下記のどれかを参照してはならない。
   a. 制御データ項目が含まれる集団データ項目。
   b. 制御データ項目の下位に属するデータ項目。
   c. 制御データ項目の何らかの部分を再定義または再命名したデータ項目。

   ページ頭書き報告集団およびページ脚書き報告集団では、SOURCE文およびUSE文によって、制御データ名を参照してはならない。
6. 手続き部に報告書名を指定したGENERATE文を書くときは、対応する報告書記述項には、明細報告集団を2つ以上含めてはならない。データ名を指定したGENERATE文を書かないときは、明細報告集団を記述しなくてもよい。
7. 1つの報告書の記述の中には、最低1つの本体集団を記述する。

一般規則

1. 明細報告集団は、GENERATE文による指示に基づいて、報告書作成制御システム（RWCS）によって処理される。明細報告集団以外の報告集団は、報告書作成制御システム（RWCS）によって自動的に処理される。
2. REPORT HEADING指定は、 該当する報告書の最初の報告集団として、報告書作成制御システム（RWCS）によって処理される報告集団を1報告書につき1回だけ指定する。報告書頭書き報告集団は、その報告書の最初のGENERATE文の実行中に処理される。
3. PAGE HEADING指定は、該当する報告書の各ページ上の最初の報告集団として、報告書作成制御システム（RWCS）によって処理される報告集団を指定する。ただし、下記の例外がある。
   a. 報告書頭書き報告集団または報告書脚書き報告集団を、排他的に表示するように指定されている場合、そのページ上では、ページ頭書き報告集団は処理されない。
   b. 報告書頭書き報告集団を、排他的に表示するように指定されていない場合、ページ頭書き報告集団は2番目の報告集団として処理される。

   (前述の*表示規則表*を参照。)
4. CONTROL HEADING指定は、指定された制御データ名用の制御集団の開始時に、報告書作成制御システム（RWCS）によって処理される報告集団を指定する。ただし、FINALを指定した場合は、報告書に関する最初のGENERTAE文が実行される間に処理される。GENERATE文の実行中に制御切れが検出されると、報告書作成制御システム（RWCS）は、最も高いレベルの制御切れ以下のすべての制御頭書き報告集団を処理する。

5. DETAIL指定は、対応するGENERTAE文が実行されるときに、報告書作成制御システム（RWCS）によって処理される報告集団を指定する。
6. CONTROL FOOTING指定は、指定された制御データ名の制御集団の終わりに、報告書作成制御システム（RWCS）によって処理される報告集団を指定する。
   FINALを指定した場合、制御脚書き報告集団は該当する報告書の最後の本体集団として、報告書作成制御システム（RWCS）によって1報告書につき1回だけ処理される。
   GENERATE文の実行中に制御切れが検出されると、報告書作成制御システム（RWCS）は、最も高いレベルの制御切れ以下のすべての制御脚書き報告集団を処理する。報告書に関して最低1回GENERATE文が実行されていれば、TERMINATE文の実行中にすべての制御脚書き報告集団が表示される。(後述の*TERMINATE文*の節を参照。)
7. PAGE FOOTING指定は、該当する報告書の各ページ上の最後の報告集団として、報告書作成制御システム（RWCS）によって処理される報告集団を指定する。ただし、下記の例外がある。
   a. 報告書頭書き報告集団または報告書脚書き報告集団を、排他的に表示するように指定されている場合、そのページ上では、ページ脚書き報告集団は処理されない。
   b. 報告書脚書き報告集団を、排他的に表示するように指定されていない場合、ページ脚書き報告集団は最後から2番目の報告集団として処理される。

   (前述の*表示規則表*の節を参照。)
8. REPORT FOOTING指定は、該当する報告書の最後の報告集団として、報告書作成制御システム（RWCS）によって処理される報告集団を、1報告書につき1回だけ指定する。報告書に関して最低1回GENERATE文が実行されていれば、TERMINATE文の実行中に報告書脚書き報告集団が表示される。（後述の*TERMINATE文*の節を参照。）
9. 報告書頭書き報告集団、ページ頭書き報告集団、制御頭書き報告集団、ページ脚書き報告集団、報告書脚書き報告集団を処理するときに、報告書作成制御システム（RWCS）が実行する処理手順を、下に説明する。
   a. 報告集団のデータ名を参照するUSE BEFORE REPORTING手続きがあれば、そのUSE手続きが実行される。
   b. SUPRESS文が実行された状態にあるかまたは報告集団に印字項目が含まれていない場合、その報告集団の処理はそれ以上は行われない。
   c. SUPRESS文が実行されてなく報告集団に印字項目が含まれている場合、報告書作成制御システム（RWCS）は、印字行を編集して、該当する報告集団の型の表示規則に従ってその報告集団を表示する。(前述の*表示規則表*の節を参照。)
10. 制御脚書き報告集団を処理するときに、報告書作成制御システム（RWCS）が実行する処理手順を、下に説明する。

制御切れが検出されると、報告書作成制御システム（RWCS）は、一連の制御脚書き報告集団を作成する。これは、制御階層の一番低いレベルの報告集団から、制御切れが検出された最も高いレベルの報告集団に至る。ここで注意することは、ある制御データ名に関して制御脚書き報告集団が定義されていなくても、その報告書のSUM句のRESET指定の対象にその制御データ名が指定されていれば、報告書作成制御システム（RWCS）は下記の10f.の処理を実行する必要がある、ということである。

a. 合計カウンタを横に集計する。具体的には、その報告集団において定義されている合計カウンタで、同じ報告集団の中のSUM句の作用対象にもなっているものをすべて、そのSUM句の合計カウンタに加算する。(前述の*SUM句*の節を参照。)

b. 合計カウンタを繰上げ集計する。 具体的には、その報告集団において定義されている合計カウンタであって、それより上位レベルの制御脚書き報告集団の中のSUM句の作用対象にもなっているものをすべて、その上位レベルのSUM句の合計カウンタに加算する。(前述の*SUM句*の節を参照。)

c. その報告集団のデータ名を参照するUSE BEFORE REPORTING手続きがあれば、そのUSE手続きを実行する。

d. SUPRESS文が実行された状態にあるかまたはその報告集団に印字項目が含まれていない場合、報告書作成制御システム（RWCS）は次に下記の手順10e.を実行する。

e. SUPRESS文が実行されてなく、その報告集団に印字項目が含まれている場合、報告書作成制御システム（RWCS）は、印字行を編集して、制御脚書き報告集団の表示規則に従ってその報告集団を表示する。

f. 報告書作成制御システム（RWCS）は、制御階層のそのレベルを処理したときに設定し直す合計カウンタを、ゼロに設定する。(前述の*SUM句*の節を参照。)

11. データ名を指定したGENERATE文に応じて報告書作成制御システム（RWCS）が実行する明細報告集団の処理を、以下に説明する。

報告書に明細報告集団を1つだけ記述した場合に、データ名を指定したGENERATE文に応じて報告書作成制御システム（RWCS）が実行する明細報告集団の処理を、下記の11a.から11e.に示す。これらの手順は、データ名を指定したGENERATE文が実行されているように実行される。

報告書に明細報告集団を1つも記述しなかった場合に、データ名を指定したGENERATE文に応じて報告書作成制御システム（RWCS）が実行する明細報告集団の処理を、下記の11a.に示す。この手順は、報告書に明細報告集団を1つだけ記述した場合と同じように、データ名を指定したGENERATE文が実行されているように実行される。

a. 報告書作成制御システム（RWCS）は、 明細報告集団用に指定されている、縦の集計を行う。(前述の*SUM句*の節を参照。)

b. その報告集団のデータ名を参照するUSE BEFORE REPORTING手続きがあれば、そのUSE手続きを実行する。

c. SUPRESS文が実行された状態にあるかまたはその報告集団に印字項目が含まれていない場合、その報告集団に関してはそれ以上何も行わない。

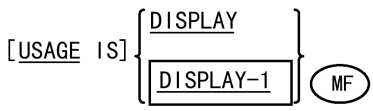
d. データ名を指定したGENERATE文に基づいて明細報告集団を実行した場合、その報告集団に関しては、それ以上何も行わない。
e. 上記の11c.も11d.も当てはまらない場合、報告書作成制御システム（RWCS）は印字行を編集して、明細報告集団の表示規則に従ってその報告集団を表示する。（前述の*表示規則表*を参照。）

12. 一般規則9，10，11に記したように、報告書作成制御システム（RWCS）は、制御頭書き報告集団、制御脚書き報告集団、明細報告集団を処理しているときに、それに割り込んで改ページ処理をすることができなければならない。改ページが必要と判断されたときは、報告書作成制御システム（RWCS）は改ページとそれに併うページ脚書き報告集団およびページ頭書き報告集団の処理を行ってから、その本体集団を表示する。
13. 制御切れ処理の間に、報告書作成制御システム（RWCS）が制御切れを検出するために保存した制御データ項目を 「古い値」と呼ぶ。
    a. 制御切れに伴う制御脚書き報告集団の処理の間に、その制御脚書き報告集団に関連するUSE手続きまたはSOURCE句の中で制御データ項目を参照すると、古い値が使用される。
    b. TERMINATE文が実行されると、報告書作成制御システム（RWCS）は最も高いレベルの制御切れが検出されたとみなす。この時点で制御脚書き報告集団または報告書脚書き報告集団に関連するUSE手続きまたはSOURCE句の中で制御データ項目を参照すると、古い値が使用される。
    c. 上記以外の場合に、報告集団内またはそれに関連するUSE手続きでデータ項目を参照した場合はすべて、その報告集団を処理している時点でのそのデータ項目の現在の値が使用される。

1.4.23 USAGE（用途）句

USAGE（用途）句は、計算機の記憶領域内におけるデータ項目の形式を指定する。

一般形式



構文規則

1. USAGE句は、任意のデータ記述項に書くことができる。
2. USAGE句を集団項目用のデータ記述項の中に書いた場合、その下位に属する基本項目または集団項目用のデータ記述項の中にも、USAGE句を書くことができる。
3. 報告集団項目用のUSAGE句には、USAGE IS DISPLAYだけが指定できる。

一般規則

1. 集団項目のレベルにUSAGE句を書いた場合、その集団に属する各基本項目に対してそのUSAGE句が適用される。
2. USAGE句は、計算機の記憶領域内でデータ項目が表現される形式を指定する。 USAGE句は、データ項目の使い方を制限するものではない。しかし、手続き部の文によっては、参照する作用対象の用途を制限するものがある。USAGE句は、データ項目の基数または文字表現の種類に影響を及ぼすことがある。
3. USAGE DISPLAY句は、データの形式が標準データ形式であることを示す。
4. 基本項目または基本項目を含む集団項目にUSAGE句を指定しないと、用途はDISPLAYとみなされる。

1.4.24 VALUE（値）句

VALUE（値）句は、報告書節中の印字項目の値を定義する。

一般形式

VALUE IS 定数-1

構文規則

1. 符号付き数字定数に指定するPICTURE文字列は、符号付き数字用のものとする。
2. VALUE句中の数字定数は、その項目のPICTURE句によって示される値の範囲に収まっていること。また、ゼロ以外の数字が切り捨てられるようではならない。VALUE句中の文字定数は、その項目のPICTURE句によって示される長さを超えてはならない。

一般規則

1. VALUE句は、その項目またはその上位の項目に関するデータ記述中の他の句と矛盾するものであってはならない。さらに、下記の規則が適用される。
    a. 項目の項類が数字である場合、VALUE句中の定数-1は数字とする。
    b. 項目の項類が英字、英数字、英数字編集、数字編集である場合、VALUE句中の定数-1は文字定数とする。定数は、データ項目が英数字と定義されているものとして収められる。（***言語リファレンス***の*COBOL言語の概念*の章の*標準桁寄せ規則*の節を参照。）PICTURE句中の編集文字は、データ項目の長さを決定するためには用いられるが、データ項目の初期化には影響を及ぼさない。（***言語リファレンス***の*プログラムの定義*の章の*PICTURE句*の節を参照。）したがって、編集項目用の値は、編集された形で指定すること。

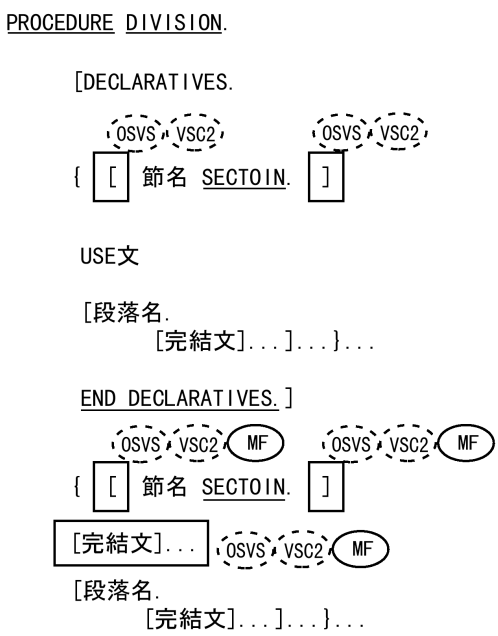
c. 項目にBLANK WHEN ZERO句やJUSTIFIED句が指定されていても、初期化には影響しない。

2. 報告書節で、報告書の基本項目記述項にVALUE句が含まれているがGROUP INDICATE句は含まれていない場合、その印字項目は、その報告集団が表示されるたびに指定された値を取ることになる。GROUP INDICATE句も含まれている場合、その印字項目は、GROUP INDICATE句の条件が成り立つときだけ、指定された値で表示される。(*区分化*の章の*GROUP INDICATE句*の節を参照。)

1.5 報告書作成機能単位における手続き部

1.5.1 一般説明

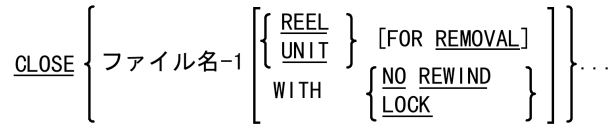
COBOL原始プログラム中に、報告書作成機能のUSE BEFORE REPORTING文が含まれていると、手続き部には 宣言手続きが含まれる。USE BEFORE REPORTING文が含まれているときの、手続き部の一般形式を下に示す。



1.5.2 CLOSE（閉じる）文

CLOSE（閉じる）文は、リール/ユニットおよびファイルの処理を終了させる。さらに、適用可能な場合、巻戻しやロックや取外しを行うこともできる。

一般形式



構文規則

CLOSE文の対象とするファイルは、すべて編成や呼出し法が同じである必要はない。

1. CLOSE文に適用できる指定は、実現されている順ファイル機能の水準に依存する。(***言語リファレンス***の*プログラムの定義*の章を参照。)

一般規則

以降の記述において別途断わらないかぎり、「リール」と「ユニット」は同じ意味であり、CLOSE文の中で同じように使用できる。順大容量記憶ファイルの扱いは、論理的にはテープやアナログ順媒体上のファイルと同じである。複数のテープ環境を持つファイルの扱いは、ファイルが1つのリールに完全に含まれている場合、論理的には順単一リール/ユニットファイルと同じである。

1. CLOSE文は、開かれているファイルに対してだけ実行できる。
2. 様々な記憶媒体に適用される種々のCLOSE文の効果を示すために、すべての 報告書ファイルを下記の種類に区分する。
    a. 非リール/ユニット: 巻戻しやリール/ユニットの概念が意味をもたない入出力媒体を使用したファイル。
    b. 順単一リール/ユニット: 1つのリール/ユニットに完全に格納される順ファイル。
    c. 順複数リール/ユニット: 複数のリール/ユニットにまたがって格納される順ファイル。
3. CLOSE文の書き方とファイルの種類ごとに、実行結果をまとめて、表に示す。

表 1-8 ： CLOSE文の書き方とファイルの種類の関係

CLOSE文の書き方	ファイルの種類		
	非リール/ユニット	順単一リール/ユニット	順複数リール/ユニット
CLOSE	C	C,G	A,C,G
CLOSE WITH LOCK	C,E	C,E,G	A,C,E,G
CLOSE WITH NO REWIND	C,H	B,C	A,B,C
CLOSE WITH REEL/UNIT	F	F,G	F,G
CLOSE REEL/UNIT FOR REMOVAL	F	D,F,G	D,F,G

上の表の中に示した記号の意味を、以降に説明する。

A. **出力報告書ファイルに対する前のリール/ユニットの影響**
報告書ファイル中の現在のリール/ユニットよりも前にあるリール/ユニットが、すべて閉じられる。ただし、既にCLOSE REEL/UNIT文が実行されているものを除く。

B. **現在のリールを巻き戻さない**
現在のリール/ユニットは、そのままの位置にとどめられる。

C. **出力報告書ファイルを閉じる**
報告書ファイルにラベル・レコードを指定してある場合、作成者の標準ラベル方式に従ってラベルの処理が行われるラベル・レコードを指定しているのにラベル・レコードが存在しない場合、またはラベル・レコードを指定していないのにラベル・レコードが存在する場合、CLOSE文の動作はどうなるかわからない。作成者によって指定された終了手続きが実行される。報告書ファイルにラベル・レコードが指定されていない場合、ラベルの処理は行われないが、作成者によって指定された他の終了手続きは実行される。

D. **リール/ユニットを取り外す**
可能な場合は、現在のリールまたはユニットが巻き戻され、そのリールまたはユニットは実行単位から論理的に取り外される。しかし、その後でその報告書ファイルに対してREELまたはUNITを指定しないでCLOSE文を実行し、続いてその報告書ファイルに対してOPEN文を実行すると、その報告書ファイル内のリールまたはユニットを適切な順番で再び呼び出せる。

E. **ファイルをロックする**
この実行単位の実行中は、その報告書ファイルを再び開くことはできない。

F. **リール/ユニットを閉じる**
出力報告書ファイル（リール/ユニット媒体）
下記の操作が行われる。

1. 標準終了リール/ユニット・ラベル手続きが実行される。
2. リール/ユニットの交換。現在のボリューム・ポインタは新しいリール/ユニットを指すように更新される。
3. 標準開始リール/ユニット・ラベル手続きが実行される。

4. そのファイルに対して次にWRITE文を実行すると、次の論理レコードはそのファイルの次のリール/ユニットに書き出される。

出力報告書ファイル（非リール/ユニット媒体）

このCLOSE文の実行は正常に終了したものと扱われる。実際には、ファイルは開かれたままであり、一般規則4に示された以外の処理は何も行われない。

G. **巻き戻す**

現在のリールまたは類似の装置は、物理的始点に位置付けられる。

H. **選択指定は無視される**

何も選択指定がされていないように、CLOSE文は実行される。

4. CLOSE文が実行されると、ファイル名-1に関する入出力状態の値が更新される。
5. 報告書ファイルの処理を終了するには、まずその報告書ファイルに関連する処理を開始されたすべての報告書をTERMINATE文を用いて終了させてから、CLOSE文を用いてその報告書ファイルを閉じる。
6. REELもUNITも指定されていないCLOSE文の実行が正常に終了すると、報告書ファイルは開かれた状態ではなくなり、ファイル結合子との結び付きも切れる。
7. 1つのCLOSE文に複数のファイル名-1を指定すると、その実行結果は、個々のファイル名-1に対して同じ順序でCLOSE文を別々に実行したのと同じことになる。

1.5.3 GENERATE（作成）文

GENERATE（作成）文は、 データ部の報告書節中に指定された報告書記述に従って報告書を作成するよう、報告書作成制御システム（RWCS）に指示する。

一般形式

```
GENERATE { データ名-1   }
         { 報告書名-1 }
```

構文規則

データ名-1には、 明細報告集団の名前を指定する。この名前は、報告書名で修飾してもよい。

1. 報告書名-1を指定できるのは、報告書の記述が下記の条件をすべて満たす場合だけである。
   a. CONTROL句を含む。
   b. 2つ以上の明細報告集団を含まない。
   c. 本体集団を最低1つ含む。

一般規則

1. 報告書名-1を指定したGENERATE文に応じて、報告書作成制御システム（RWCS）は合計報告の処理を行う。ある報告書用に実行されるGENERATE文がすべて報告書名-1を指定した形のものである場合、作成される報告書を 合計報告書と呼ぶ。
2. データ名-1を指定したGENERATE文に応じて、報告書作成制御システムは明細報告の処理を行う。その中には、指定された明細報告集団に特有の処理も含まれる。通常、データ名-1を指定したGENERATE文を実行することによって、報告書作成制御システム（RWCS）は指定された明細報告集団を表示する。
3. 報告書に対する最初のGENERATE文を実行する間に、報告書作成制御システム（RWCS）は制御データ項目の値を保存する。同じ報告書に対する2回目以降のGENERATE文を実行する間に、報告書作成制御システム（RWCS）は保存した制御データ項目の値を使用して、制御切れの発生を検出する。制御切れの発生が検出されると、報告書作成制御システム（RWCS）は新しい制御データ項目の値を保存し、その値を使用して、さらに制御切れの発生を検出する。
4. 報告書を表示する間、本体集団を表示するために改ページが必要になったときに、ページ頭書き報告集団およびページ脚書き報告集団が定義されていると、報告書作成制御システム（RWCS）によってそのための処理が自動的に行われる。(前述の*表示規則表*の節を参照。)
5. 報告書に対する最初のGENERATE文を実行するときに、報告書作成制御システム（RWCS）は下記の報告集団を記してある順に処理する。ただし、報告書記述項中にそれらの報告集団が定義されていることを前提とする。報告書作成制御システム（RWCS）はまた、一般規則4.に記したように、ページ頭書き報告集団およびページ脚書き報告集団の処理も行う。各型の報告集団を処理するときに報告書作成制御システム（RWCS）が行う具体的な処理については、前述の*TYPE 句*の節を参照。
    a. 報告書頭書き報告集団を処理する。
    b. ページ頭書き報告集団を処理する。
    c. 制御頭書き報告集団を、最高レベルから最低レベルまですべて処理する。
    d. データ名-1を指定したGENERATE文が実行される場合は、指定された明細報告集団用の処理を実行する。報告書名-1を指定したGENERATE文が実行される場合は、明細報告集団用の処理に係わる手順の一部を実行する。(前述の*TYPE句*の節を参照。)
6. 報告書に対する2番目以降のGENERATE文を実行するときに、報告書作成制御システム（RWCS）は下記の手順のうち適用できるものを実行する。報告書作成制御システム（RWCS）はまた、一般規則4.に記したように、ページ頭書き報告集団およびページ脚書き報告集団の処理も行う。各型の報告集団を処理するときに報告書作成制御システム（RWCS）が行う具体的な処理については、前述の*TYPE句*の節を参照。)
    a. 制御切れの検出。制御データ項目の値が変わったか否かを判定する規則は、比較条件の場合と同じである。制御切れが発生すると、下記の処理を行う。
        1. 制御切れを検出するために報告書作成制御システム（RWCS）が使用した制御データ項目の値を、制御脚書きに関連するUSE手続きおよびSOURCE句で使用できるようにする。(前述の*TYPE句*の節を参照。)

2. 制御脚書き報告集団を、低いレベルから高いレベルへ向けて処理する。その対象となるレベルは、発生している制御切れの中の最も高いレベル以下のものである。
3. 制御頭書き報告集団を高いレベルから低いレベルへ向けて処理する。その対象となるレベルは、発生している制御切れの中の最も高いレベル以下のものである。

b. データ名-1を指定したGENERATE文が実行される場合は、指定された明細報告集団用の処理を実行する。報告書名-1を指定したGENERATE文が実行される場合は、明細報告集団用の処理に係わる手順の一部を実行する。（前述の*TYPE句*の節を参照。）

7. 報告書に対してGENERATE文を実行できるのは、その報告書に対するINITIATE文が実行された後で、TERMINATE文が実行される前までの間だけである。

1.5.4 INITIATE（開始）文

INITIATE（開始）文は、報告書作成制御システム（RWCS）に 報告書の処理を開始させる。

一般形式

INITIATE {報告書名-1}...

構文規則

1. 報告書名-1は、データ部の報告書節中の報告書記述項に定義されていること。

一般規則

1. INITIATE文は、指定された各報告書に対して、下記の初期化処理を行う。
   a. すべての 合計カウンタをゼロに設定する。
   b. LINE-COUNTERをゼロに設定する。
   c. PAGE-COUNTERをゼロに設定する。
2. INITIATE文は、報告書に関連するファイルを開くわけではない。したがって、INITIATE文を実行する前に、OPEN文をOUTPUTまたはEXTENDを指定して実行しなければならない。
3. 報告書名-1に対するINITIATE文を実行した場合、対応してその報告書に対してTERMINATE文を実行する。その後でなければ、報告書名-1に対して次のINITIATE文は実行できない。
4. 1つのINITIATE文に複数の報告書名-1を指定すると、その実行結果は、個々の報告書名-1に対して同じ順序でINITIATE文を別々に実行したのと同じことになる。

1.5.5 OPEN（開く）文

言語リファレンスのプログラムの定義の章を参照のこと。

追加の構文規則

1. 報告書ファイルは、順編成ファイルとする。
2. 報告書ファイルに対するOPEN文に指定できる入出力モードは、OUTPUTまたはEXTENDだけである。

追加の一般規則

1. 報告書ファイルに含まれる報告書に対するINITIATE文を実行する前に、その報告書ファイルに対してOPEN文を実行しなければならない。

1.5.6 SUPPRESS（抑制）文

SUPPRESS（抑制）文は、報告書作成制御システム（RWCS）に報告集団の表示を抑制させる。

一般形式

```
SUPPRESS PRINTING
```

構文規則

1. SUPPRESS文は、USE BEFORE REPORTING手続き内にだけ指定できる。

一般規則

1. SUPPRESS文によって表示が抑制される報告集団は、USE手続きの中で指定されたものだけである。
2. SUPPRESS文は、報告集団の表示を抑制する必要があるたびに実行しなければならない。
3. SUPPRESS文が実行されると、報告書作成制御システム（RWCS）は下記の報告集団機能の処理を、しないようにする。
    a. 報告集団の印字行の表示。
    b. 報告集団中のすべてのLINE句の処理。
    c. 報告集団中のNEXT GROUP句の処理。
    d. LINE-COUNTERの更新。
4. (OSVS)特殊レジスタのPRINT-SWITCHに1を設定することによっても、 SUPPRESS PRINTINGの機能を実現できる。

つまり、下記の2つの文は機能的に等しい。

```
MOVE 1 TO PRINT-SWITCH
SUPPRESS PRINTING
```

1.5.7 TERMINATE（終了）文

TERMINATE（終了）文は、報告書作成制御システム（RWCS）に 報告書の処理を終了させる。

一般形式

TERMINATE {報告書名-1}...

構文規則

1. 報告書名-1は、データ部の報告書節中の報告書記述項に定義されていなければならない。

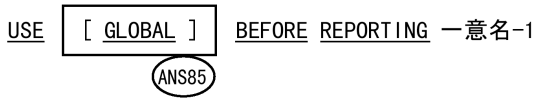
一般規則

1. TERMINATE文は、報告書作成制御システム（RWCS）にすべての制御脚書き報告集団を作成させる。この処理は低いレベルから高いレベルへと進められる。次いで、報告書脚書き報告集団が作成される。このとき、報告書作成制御システム（RWCS）は、最高レベルの制御データ項目の制御切れが発生したものとして、制御脚書き報告集団および報告書脚書き報告集団のSOURCE句およびUSE手続きで、一連の制御データ項目の古い値を使用できるようにする。
2. INITIATE文とTERMINATE文の間にGENERATE文が実行されなかった場合、その報告書に関してTERMINATE文は、報告書作成制御システム（RWCS）に報告集団を何も作成させず、またそれに関連する処理を何もさせない。
3. 報告書を表示する間、本体集団を表示するために改ページが必要になったときに、ページ頭書き報告集団およびページ脚書き報告集団が定義されていると、報告書作成制御システム（RWCS）によってそのための処理が自動的に行われる。(前述の*表示規則表*を参照。)
4. 報告書に対してTERMINATE文を実行できるのは、その報告書に対するINITIATE文が実行された後、かつTERMINATE文が実行される前だけである。
5. 1つのTERMINATE文に複数の報告書名-1を指定すると、その実行結果は、個々の報告書名-1に対して同じ順序でTERMINATE文を別々に実行したのと同じことになる。
6. TERMINATE文は、報告書が関連するファイルを閉じるわけではない。したがって、報告書ファイルを閉じるために、CLOSE文を実行しなければならない。報告書の作成を開始したならば、対応する報告書ファイルをTERMINATE文で終了させてから、CLOSE文を実行しなければならない。

1.5.8 USE BEFORE REPORTING（報告の前に使用）文

USE BEFORE REPORTING（報告の前に使用）文は、データ部の報告書節中に指定された報告集団が表示される直前に実行する手続き部の文を指定する。

一般形式



構文規則

1. USE BEFORE REPORTING文を記述する場合は、手続き部の宣言部分の節の見出しに続けて、それ自体で1つの完結文として書く。その節の後ろには、何も記述しなくてもよいし、使用する手続きを定義する段落をいくつか記述してもよい。
2. 一意名-1は、報告集団を指していること。同じ手続き部内の複数のUSE BEFORE REPORTING文の中に、同じ一意名-1が現れてはならない。
3. USE BEFORE REPORTING手続き中の段落内にGENERATE文、INITIATE文、TERMINATE文が現れてはならない。また、USE BEFORE REPORTING手続き中のPERFORM文の範囲内にGENERATE文、INITIATE文、TERMINATE文が現れてはならない。
4. USE BEFORE REPORTING手続きは、どの制御データ項目の値を変えてもならない。
5. USE BEFORE REPORTING文自体は、実行されない。この文は、USE手続きを呼び出す条件を定義するだけである。

一般規則

1. プログラムの実行中に指定された報告集団が作成される直前に、 宣言手続きが呼び出される。該当する報告集団は、宣言の先頭のUSE BEFORE REPORTING文中の一意名-1に指定する。
2. 宣言手続きの中から、通常の手続きを参照してはならない。
3. 他の宣言節または通常の手続きからは、PERFORM文によってだけ、USE BEFORE REPORTING文に関連する手続き名を参照できる。
4. USE BEFORE REPORTING文に指定した手続きは、該当する報告集団が作成される直前に、報告書作成制御システム（RWCS）によって実行される。(前述の*TYPE句*の節を参照。)
5. 既に呼び出されているUSE手続きが、まだ呼び出し元の手順に制御を返さないうちに、そのUSE手続きを起動する文が実行されてはならない。
6. プログラムが他のプログラム中に含まれている場合、以下に示す特別な規則が適用される。これらの規則の適用では、最初の限定的な指令だけが実行用に選択される。実行用に選択された指令は、その指令を実行するための規則に従っていなくてはならない。指令を選択する順序は、次の通りである。
    a. 限定的な条件を引き起こす文を持つプログラム中の指令。

b. GLOBAL指定が指定され、限定的な指令を最後に実行したプログラムを直接含むプログラム中の指令。
c. 規則6b を適用して選択された指令(最も外側のプログラムに適用されたものまでを含む)。限定的な指令が見つからない場合、何も実行されない。

第2章：通信機能単位の概要

通信機能単位（communication module）は、通信文（message）またはその一部を呼び出し、処理し、生成する機能を提供する。この機能によって、通信管理システム（message control system）を通じて、ローカル（局所）およびリモート（遠隔地）の通信装置と通信することができる。

(MF) この章でANS85と印されている機能は、COMS85指令の影響を受ける。（実行時に通信機能単位がどのようにサポートされるかの詳細については、『COBOLシステムリファレンス』を参照）

(XOPEN) 標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、通信機能単位は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの機能単位を使用するべきではない。

2.1 通信機能単位におけるデータ部

2.1.1 通信節

COBOLプログラムにおいては、通信記述項（communication description entry）が通信節（communication section）中で階層的に最も高い位置にある。通信節の見出しに続けて、通信記述項を書く。　この通信記述項は、レベル指示語のCD、通信記述名、一連の句からなる。入力通信記述項には、主待行列（queue）、副待行列（sub-queue）、受信日付け（message date）、受信時刻（message time）、文字数（text length）、状態キー（status key）、終了キー（end key）、通信文の個数（message count）などを指定する。出力通信記述項には、宛先の個数（destination count）、文字数、状態キー、誤りキー（error key）、宛先（destination）などを指定する。

(ANS85) 入出力通信記述項には、受信日付け、受信時刻、通信装置、文字数、終了キー、状態キーを指定する。

通信記述項は終止符（.）で止める。通信記述項のレコード領域は、その後ろに利用者がレコード記述項を定義することによって、暗黙的に再定義される。

2.1.2 通信記述項の全体的な骨組み

通信記述項は、通信管理システムとCOBOLシステムとの間のインタフェースを指定する。

一般形式

書き方1

```
CD 通信記述名
                                  [                                          ]
                                  [[SYMBOLIC QUEUE IS データ名-1]            ]
                                  [                                          ]
                                  [   [SYMBOLIC SUB-QUEUE-1 IS データ名-2]   ]
                                  [   [SYMBOLIC SUB-QUEUE-2 IS データ名-3]   ]
                                  [   [SYMBOLIC SUB-QUEUE-3 IS データ名-4]   ]
   FOR[INITIAL]INPUT              [   [MESSAGE DATE IS データ名-5]           ]
                                  [   [MESSAGE TIME IS データ名-6]           ]
                                  [   [SYMBOLIC SOURCE IS データ名-7]        ]
                                  [   [TEXT LENGTH IS データ名-8]            ]
                                  [   [END KEY IS データ名-9]                ]
                                  [   [STATUS KEY IS データ名-10]            ]
                                  [   [MESSAGE COUNT IS データ名-11]]        ]
                                  [[データ名-1 データ名-2 ...データ名-11]    ]
                                  [                                          ]
```

書き方2

```
CD 通信記述名  FOR OUTPUT

 [DESTINATION COUNT IS データ名-1]
       [TEXT LENGTH IS データ名-2]
       [STATUS KEY IS データ名-3]
       [DESTINATION TABLE OCCURS 整数-2 TIMES  ]
       [   [INDEXED BY 指標名-1[指標名-2]...   ]
[ERROR KEY IS データ名-4]
       [SYMBOLIC DESTINATION IS データ名-5]
```

書き方3

```
CD 通信記述名
                                  [                                          ]
                                  [  [MESSAGE DATE IS データ名-1]            ]
                                  [  [MESSAGE TIME IS データ名-2]            ]
                                  [  [SYMBOLIC TERMINAL IS データ名-3]       ]
   FOR [INITIAL]I-O               [  [TEXT LENGTH IS データ名-4]             ]
                                  [  [END KEY IS データ名-5]                 ]
                                  [  [STATUS KEY IS データ名-6]              ]
                                  [[データ名-1 データ名-2...データ名-6]      ]
                                  [                                          ]
```

構文規則

すべての書き方

1. 通信記述項は、通信節にだけ書くことができる。 .

書き方1 と 3

2. 単一のプログラム内では、INITIAL句は1つの通信記述項だけに指定できる。手続き部の見出しにUSING指定をしているプログラムの中で、INITIAL句を使用してはならない。（第3章「*中核*」の「*3.6-1　手続き部の見出し*」の節を参照）
3. INITIAL句以外の選択句は、どんな順序で書いてもよい。
4. この書き方の選択句をすべて省略した場合は、通信記述項の直後に01レベルのデータ記述項を書く。どの選択句の後に01レベルのデータ記述項を書いてもよい。
5. 入力通信記述項の直後に書いたレコード記述項は、暗黙的に通信レコードを再定義することになる。このレコード記述項は、標準データ形式で数えて87文字ちょうどのレコードとして記述する。　通信レコードは、何回も再定義してかまわない。ただし、VALUE句を指定できるのは、最初の再定義だけである。通信管理システムは、暗黙の再定義に関係なく、つねに一般規則4の規則に従って通信レコードを参照する。
6. データ名-1、データ名-2、……、データ名-11は、通信記述項内で一意とする。このようにデータ名を連続して書くとき、どのデータ名を予約語のFILLERとしてもよい。

書き方2

7. (ANS85)選択句はどんな順序で書いてもよい。
8. この書き方の選択句をすべて省略した場合、通信記述項の直後に01レベルのデータ記述項を書く。
9. 出力通信記述項の直後に書いたレコード記述項は、暗黙的に通信レコードを再定義することになる。通信レコードを何回も再定義してかまわない。ただし、VALUE句を指定できるのは、最初の再定義だけである。通信管理システムは、暗黙の再定義に関係なく、つねに一般規則18に従って通信レコードを参照する。
10. データ名-1、データ名-2、……、データ名-5は、通信記述項内で一意とする。
11. DESTINATION TABLE OCCURS句を指定しないと、ERRORキーとSYMBOLIC DESTINATIONがそれぞれ1つずつ確保される。この場合、それらのデータ項目を参照するときに、添字も指標も使用することはできない。
12. DESTINATION TABLE OCCURS句を指定すると、データ名-4およびデータ名-5を参照するときには、添字付けまたは指標付けをしなければならない。
13. データ名-1および整数-2のデータ項目の値に制限はない。

書き方3

14. (ANS85)入出力通信記述項の直後に書いたレコード記述項は、暗黙的に通信レコードを再定義することになる。このレコード記述項は、標準データ形式で数えて33文字ちょうどのレコードとして記述する。　通信レコードを何回も再定義してかまわない。ただし、VALUE句を指定できるのは、最初の再定義だけである。通信管理システムは、暗黙の再定義に関係なく、つねに一般規則26に従って通信レコードを参照する。

一般規則

すべての書き方

1. 各文の実行が終了した時点で、書き方1のデータ名-10、書き方2のデータ名-3、
(ANS85)書き方3のデータ名-6
の状態キーが取る値を、表2-1に示す。表中の○は、その欄の文がその行の状態キーの値を取ることを示す。

表2-1　通信文と状態キーの対応

RECEIVE文												
	SEND文 (ANS85)（入出力用）											
		(ANS85)SEND文（出力用）										
			(ANS85)PURGE文									
				ACCEPT MESSAGE COUNT文								
					ENABLE INPUT文							
						ENABLE INPUT/I-O TERMINAL文						
							ENABLE OUTPUT文					
								(ANS85)DISABLE INPUT文				
									(ANS85)DISABLE INPUT /I-O TERMINAL文			
										DISABLE OUTPUT文		
											状態キーの値	
○	○	○	○	○	○	○	○	○	○	○	00	誤りは検出されない。動作は完了した。
		○	○								10	1つ以上の宛先が切り離されている。動作は完了した。（一般規則2を参照）
	○										10	宛先が切り離されている。動作しなかった。
					○	○	○	○	○	○	15	(ANS85)発信源または1つ以上の待行列か宛先が既に切り離されているか、または接続されている。（一般規則2を参照）
	○	○	○				○			○	20	1つ以上の宛先が未知である。既知な宛先に対しては、動作が完了した。（一般規則2を参照）
○				○	○			○			20	1つ以上の主待ち行列または副待ち行列が未知である。動作しなかった。
○						○			○		21	(ANS85)発信源が未知である。動作しなかった。
		○	○				○			○	30	宛先個数が正しくない。動作しなかった。
					○	○	○	○	○	○	40	パスワードが正しくない。接続または切り離しの動作はしなかった。
	○	○									50	文字数が一意名-1の大きさよりも多い。
	○	○									60	送信を要求された部分の文字数がゼロであるかまたは一意名-1が抜けている。動作しなかった。
		○									65	(ANS85)出力待ち行列の容量を超えた。（一般規則2を参照）
			○								70	(ANS85)1つ以上の宛先で対象が存在しなかった。他の宛先への動作は完了した。
		○	○		○		○	○		○	80	(ANS85)10,15,20の状態キー条件のうち、2つ以上が発生した。（一般規則2を参照）
											9x	(ANS85)オペレーティングシステムによって定義される状態。

書き方1

2. 入力通信記述項情報は、通信管理システムとプログラムとの間で処理される通信文に関する連絡用に使用される。この情報は、通信装置から通信文の一部として入ってくるわけではない。
3. 各入力通信記述項に、標準データ形式で数えて87文字分のレコード領域が割り当てられる。このレコード領域は、通信管理システムに対して下記のように定義される。
    a. SYMBOLIC QUEUE句を書くと、データ名-1が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は入力レコード中の文字位置1-12を占める。
    b. SYMBOLIC SUB-QUEUE-1句を書くと、データ名-2が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置13-24を占める。
    c. SYMBOLIC SUB-QUEUE-2句を書くと、データ名-3が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置25-36を占める。
    d. SYMBOLIC SUB-QUEUE-3句を書くと、データ名-4が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置37-48を占める。
    e. MESSAGE DATE句を書くと、データ名-5が長さ6文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置49-54を占める。
    f. MESSAGE TIME句を書くと、データ名-6が長さ8文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置55-62を占める。
    g. SYMBOLIC SOURCE句を書くと、データ名-7が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置63-74を占める。
    h. TEXT LENGTH句を書くと、データ名-8が長さ4文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置75-78を占める。
    i. END KEY句を書くと、データ名-9が長さ1文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置79を占める。
    j. STATUS KEY句を書くと、データ名-10が長さ2文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置80-81を占める。
    k. MESSAGE COUNT句を書くと、データ名-11が長さ6文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置82-87を占める。

もう1つ別の指定方法がある。データ名を連続して書くと、上記の順番に対応する句を指定したことになる。

どちらの指定方法を採っても、下記のレコード記述を暗黙的に定義したことになる。見出し「注記」の下の情報は説明であって、記述の一部ではない。

暗黙の記述		注記
01 データ名-0.		
02 データ名-1	PIC X(12).	SYMBOLIC QUEUE
02 データ名-2	PIC X(12).	SYMBOLIC SUB-QUEUE-1
02 データ名-3	PIC X(12).	SYMBOLIC SUB-QUEUE-2
02 データ名-4	PIC X(12).	SYMBOLIC SUB-QUEUE-3
02 データ名-5	PIC 9(06).	MESSAGE DATE
02 データ名-6	PIC 9(08).	MESSAGE TIME
02 データ名-7	PIC X(12).	SYMBOLIC SOURCE
02 データ名-8	PIC 9(04).	TEXT LENGTH
02 データ名-9	PIC X.	END KEY
02 データ名-10	PIC XX.	STATUS KEY
02 データ名-11	PIC 9(06).	MESSAGE COUNT

4. 使用しないときのデータ名-2、データ名-3、データ名-4のデータ項目の内容は、空白にしておく。
5. データ名-1のデータ項目には、主待ち行列の記号名を入れる。データ名-2、データ名-3、データ名-4のデータ項目には、副待ち行列の記号名を入れる。　これらの記号名はすべて、システム名の構成規則に従い、かつ、あらかじめ通信管理システムに対して定義済みでなければならない。
6. RECEIVE文は、通信記述項中に指定された待ち行列から次の通信文またはその一部を順次引き取る。

   RECEIVE文の実行中に、どの待ち行列から通信文を取り出すかをより具体的に指定する必要があるときは、データ名-2、データ名-3、データ名-4のデータ項目の内容を使用する。待ち行列構造のあるレベルを指定したときは、それより上位のレベルも指定しなければならない。

   待ち行列を最下位のレベルまで詳しく指定しなかったときは、呼び出す次の通信文またはその一部は、通信管理システムによって決定される。

   RECEIVE文の実行が終了した時点では、データ名-1からデータ名-4のデータ項目には、待ち行列構造の記号名がすべてのレベルにわたって入れられている。
7. 通信文の処理を行うプログラムが通信管理システムによって起動されると、そのプログラムは実行単位を確立する。そのとき、そのプログラムの実行を要求する基となった待ち行列構造の記号名が、INITIAL句を伴う通信記述項のデータ名-1からデータ名-4のデータ項目に入れられている。これ以外の場合には、それらのデータ項目には空白が入れられている。

   待ち行列データ項目に記号名または空白を入れる処理は手続き部の最初の文が実行される前に行われる。

次いで、データ名-1からデータ名-4のデータ項目の内容をそのままにしてRECEIVE文を実行することによって、そのプログラムを実行するきっかけとなった実際の通信文を引き取ることができる。通信記述項の残りの項目は、そのときに更新される。

8. INITIAL句のないプログラムを通信管理システムが起動しようとした場合、結果はどうなるかわからない。
9. データ名-5の形式は、YYMMDD（年、月、日）である。これは、通信管理システムが、通信文の受信が完了したことを認識した日付を表わす。

   データ名-5のデータ項目の内容は、RECEIVE文の実行の一環として、通信管理システムによってだけ更新される。
10. データ名-6の形式は、HHMMSSTT（時、分、秒、百分の一秒）である。これは、通信文の受信が完了したことを通信管理システムが認識した時刻を表わす。

    データ名-6のデータ項目の内容は、RECEIVE文の実行の一環として、通信管理システムによってだけ更新される。
11. RECEIVE文の実行中に、通信管理システムは、その通信文を発信した通信装置の記号名をデータ名-7のデータ項目に入れる。この記号名は、システム名の構成規則に従っていなければならない。ただし、その通信装置の記号名が通信管理システム側でわかっていないときは、このデータ項目に空白を入れる。
12. 通信管理システムは、RECEIVE文を実行した結果として受信された文字数を、データ名-8のデータ項目に入れる。（この章で後述する「2.3-5　*RECEIVE*文」の節を参照）
13. データ名-9のデータ項目の内容は、RECEIVE文の実行の一環として、下記の規則に従って、通信管理システムによってだけ更新される。
    a. RECEIVE MESSAGE指定をしてあると、データ名-9は下記のどれかに設定される。
       - 通信群の終了が検出されていると、データ名-9の内容は3に設定される。
       - 通信文の終了が検出されていると、データ名-9の内容は2に設定される。
       - 通信文の部分終了が検出されていると、データ名-9の内容は0に設定される。

    b. RECEIVE SEGMENT指定をしてあると、データ名-9は下記のどれかに設定される。
       - 通信群の終了が検出されていると、データ名-9の内容は3に設定される。
       - 通信文の終了が検出されていると、データ名-9の内容は2に設定される。
       - 通信行の終了が検出されていると、データ名-9の内容は1に設定される。
       - 通信行の部分終了が検出されていると、データ名-9の内容は0に設定される。

    c. 上記の条件がいくつか同時に満たされる場合は、上記の順番に従って最初に該当した条件によって、データ名-9のデータ項目の内容が決定される。

14. データ名-10のデータ項目の内容は、前に実行されたRECEIVE, ACCEPT MESSAGE COUNT, ENABLE INPUT, DISABLE INPUTの各文の状態条件を表わす。
    データ名-10のデータ項目の内容と状態条件との実際の関係は、表2-1を参照。
15. データ名-11のデータ項目の内容は、主待ち行列、副待ち行列-1などにある通信文の数を表わす。このデータ項目の内容は、COUNT指定を伴うACCEPT文を実行する一環としてだけ、通信管理システムによって更新される。

書き方2

16. 出力通信記述項の情報は通信装置に送られる性質のものではなく、処理する通信文に関して、プログラムと通信管理システムの間での連絡に使用されるものである。
17. 各出力通信記述項に、標準データ形式で数えて（10 + 13 * 整数-2）の計算式で求められる文字数分の、連続するレコード領域が割り当てられる。このレコード領域は、通信管理システムに対して下記のように定義される。
    a. DESTINATION COUNT句を書くと、データ名-1が長さ4文字の符号なし整数データ項目の名前として定義される。このデータ項目は、このレコード中の文字位置1-4を占める。
    b. TEXT LENGTH句を書くと、データ名-2が長さ4文字の符号なし整数データ項目の名前として定義される。このデータ項目は、このレコード中の文字位置5-8を占める。
    c. STATUS KEY句を書くと、データ名-3が長さ2文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置9-10を占める。
    d. 文字位置11以降は表となる。この表は、下記の2つの要素から構成される。これらの要素の文字数の合計は、13文字である。
        - ERROR KEY句を書くと、データ名-4が長さ1文字の基本英数字データ項目の名前として定義される。
        - SYMBOLIC DESTINATION句を書くと、データ名-5が長さ12文字の基本英数字データ項目の名前として定義される。

    上記の句を書くと、下記のレコード記述を暗黙的に定義したことになる。見出し「注記」の下の情報は説明であって、記述の一部ではない。

暗黙の記述		注記
01 データ名-0.		
02 データ名-1	PIC 9(04).	DESTINATION COUNT
02 データ名-2	PIC 9(04).	TEXT LENGTH
02 データ名-3	PIC XX.	STATUS KEY
02 データ名 OCCURS 整数-2 TIMES.		DESTINATION TABLE
03 データ名-4	PIC X.	ERROR KEY
03 データ名-5	PIC X(12).	SYMBOLIC DESTINATION

18. データ名-1のデータ項目の内容は、宛先の数を示す。この宛先の数は、SEND，ENABLE OUTPUT，DISABLE OUTPUTの各文を実行する際に、通信管理システムがデータ名-5のデータ項目の中から取り出す宛先の数として使用される。

    通信管理システムは、データ名-5のデータ項目の最初の出現番号のものを最初の宛先とし、次の出現番号のものを次の宛先とするといったように、データ名-1のデータ項目によって示される数の宛先を処理する。

    SEND，ENABLE OUTPUT，DISABLE OUTPUTの各文を実行している間に、データ名-1のデータ項目の値が整数-2を超えると、誤り条件が発生し、それらの文の実行は停止される。

19. SEND，ENABLE OUTPUT，DISABLE OUTPUTの各文を実行するときに、データ名-1のデータ項目の値の妥当性を保証するのは、プログラマの責任である。

20. SEND文を実行する一環として、通信管理システムはデータ名-2のデータ項目の内容を、送り出す通信文の文字数と解釈する。そして、SEND文に指定された一意名の領域の左端の文字からこの文字数に達するまでを順に送信する。（この章で後述する「2.3-6 *SEND*文」の節を参照）

21. データ名-5のデータ項目の各反復要素には、通信管理システムに既知の宛先が入れられている。この宛先はシステム名の構成規則に従っていなければならない。

22. データ名-3のデータ項目の内容は、前に実行されたSEND，ENABLE OUTPUT，DISABLE OUTPUTの各文の状態条件を表わす。

    データ名-3のデータ項目の内容と状態条件との実際の関係は、表2-1を参照。

23. SEND，ENABLE OUTPUT，DISABLE OUTPUTの各文の実行中に、指定された宛先の中に通信管理システムにとって未知なものがあると、データ名-3のデータ項目の内容とデータ名-4のデータ項目のすべての反復要素が更新される。

    データ名-4のデータ項目の内容が1であるときは、対応するデータ名-5のデータ項目の値が通信管理システムにとって未知であることを示す。これ以外の場合は、データ名-4のデータ項目の内容はゼロに設定される。

24. 表2-2は、各文の実行完了後のデータ名-4のデータ項目の内容を示す。"○" は対応する文で対応する値を取り得ることを示す。

表2-2　誤りキーの値

SEND文					
	PURGE文				
		ENABLE OUTPUT文			
			DISABLE OUTPUT文		
				誤りキーの値	
○	○	○	○	0	誤りは検出されない。
○	○	○	○	1	宛先が未知である。
○	○			2	宛先が切り離されている。
	○			4	参照する宛先の通信文の一部がない。
		○	○	5	宛先が既に接続されているかまたは切り離されている。
○				6	出力待ち行列の容量を超えた。
				7-9	将来の予備。
				A-Z	オペレーティングシステムによって定義される条件。

書き方3

(ANS85)

25. 入出力通信記述情報は、取り扱う通信文に関して、通信管理システムとプログラムとの間で交換される通信内容に係わるものである。この情報は、通信文の一部として端末から送られてくるものではない。
26. 各入出力通信記述項に対して、連続する33文字分のレコード領域が割り当てられる。このレコード領域は、通信管理システムに対して、下記のように定義される。
    a. MESSAGE DATE句を書くと、データ名-1が長さ6文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置1-6を占める。
    b. MESSAGE TIME句を書くと、データ名-2が長さ8文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置7-14を占める。
    c. SYMBOLIC TERMINAL句を書くと、データ名-3が長さ12文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置15-26を占める。
    d. TEXT LENGTH句を書くと、データ名-4が長さ4文字の符号なし整数データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置27-30を占める。
    e. END KEY句を書くと、データ名-5が長さ1文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置31を占める。

f. STATUS KEY句を書くと、データ名-6が長さ2文字の基本英数字データ項目の名前として定義される。このデータ項目は、入力レコード中の文字位置32-33を占める。

もう1つ別の指定方法がある。データ名を連続して書くと、上記の順番に対応する句を指定したことになる。

どちらの指定方法を採っても、下記のレコード記述を暗黙的に定義したことになる。見出し「注記」の下の情報は説明であって、記述の一部ではない。

暗黙の記述	注記
01 データ名-0.	
02 データ名-1 PIC 9(6).	MESSAGE DATE
02 データ名-2 PIC 9(8).	MESSAGE TIME
02 データ名-3 PIC X(12).	SYMBOLIC TERMINAL
02 データ名-4 PIC 9(4).	TEXT LENGTH
02 データ名-5 PIC X.	END KEY
02 データ名-6 PIC XX.	STATUS KEY

27. 通信文を処理するプログラムが通信管理システムによって起動されると、INITIAL句を伴う入出力通信記述項を参照する最初のRECEIVE文によって、プログラムを起動するきっかけとなった実際の通信文が返される。
28. データ名-1の形式は、YYMMDD（年、月、日）である。これは、通信管理システムが、通信文の受信が完了したことを認識した日付を表わす。

    データ名-1のデータ項目の内容は、RECEIVE文の実行の一環として、通信管理システムによってだけ更新される。
29. データ名-2の形式は、HHMMSSTT（時、分、秒、百分の一秒）である。これは、通信文の受信が完了したことを通信管理システムが認識した時刻を表わす。

    T　データ名-2のデータ項目の内容は、RECEIVE文の実行の一環として、通信管理システムによってだけ更新される。
30. 通信文の処理を行うプログラムが通信管理システムによって起動されると、そのプログラムは実行単位を確立する。このとき、そのプログラムの実行を要求する基となった通信端末の記号名が、INITIAL句を伴う入出力通信記述項のデータ名-3のデータ項目に入れられている。この記号名は、システム名の構成規則に従っていなければならない。

    それ以外の場合には、そのデータ項目には空白が入れられている。

    このデータ項目に記号名または空白を入れる処理は、手続き部の最初の文が実行される前に行われる。
31. INITIAL句のないプログラムを通信管理システムが起動しようとした場合、結果はどうなるかわからない。
32. 入出力通信記述項にINITIAL句を指定してある場合、通信管理システムによってプログラムが起動されたときに、データ名-3のデータ項目の内容は、そのプログラムによって変更されてはならない。データ名-3のデータ項目の内容が変更された場合、通信記述名-1を参照するどの文の実行も不成功に終わり、データ名-6のデータ項目に、状況に応じて、発信源または宛先が不明であることを示す値が設定される（一般規則1.を参照）。

33. INITIAL句を伴わない入出力通信記述項またはINITIAL句を伴う入出力通信記述項で、プログラムが通信管理システムによって起動されたのではない場合、通信記述名-1を参照する最初の文は、プログラムの中でデータ名-3に発信源または宛先を設定してからでなければ実行してはならない。

    通信記述名-1を参照する文を最初に実行した後では、データ名-3のデータ項目の内容をそのプログラムの中で変更してはならない。データ名-3のデータ項目の内容が変更された場合、通信記述名-1を参照するどの文の実行も不成功に終わり、データ名-6のデータ項目に、状況に応じて、発信源または宛先が不明であることを示す値が設定される（一般規則1.を参照）。

34. 通信管理システムは、RECEIVE文を実行した結果として受信された文字数を、データ名-4のデータ項目に入れる。（この章で後述する「2.3-5　*RECEIVE*文」の節を参照）

    SEND文を実行する一環として、通信管理システムは、データ名-4のデータ項目の内容を、送り出す通信文の文字数と解釈する。そして、SEND文に指定された一意名の領域の左端の文字からこの文字数に達するまでを、順に送信する。（この章で後述する「2.3-6　*SEND*文」の節を参照）

35. データ名-5のデータ項目の内容は、RECEIVE文の実行の一環として、下記の規則に従って、通信管理システムによってだけ更新される。

    a. RECEIVE MESSAGE指定をしてあると、データ名-5は下記のどれかに設定される。
        - 通信群の終了が検出されていると、データ名-5の内容は3に設定される。
        - 通信文の終了が検出されていると、データ名-5の内容は2に設定される。
        - 通信文の部分終了が検出されていると、データ名-5の内容は0に設定される。

    b. RECEIVE SEGMENT指定をしてあると、データ名-5は下記のどれかに設定される。
        - 通信群の終了が検出されていると、データ名-5の内容は3に設定される。
        - 通信文の終了が検出されていると、データ名-5の内容は2に設定される。
        - 通信行の終了が検出されていると、データ名-5の内容は1に設定される。
        - 通信行の部分終了が検出されていると、データ名-5の内容は0に設定される。

    c. 上記の条件がいくつか同時に満たされる場合は、上記の順番に従って最初に該当した条件によって、データ名-5のデータ項目の内容が決定される。

36. データ名-6のデータ項目の内容は、前に実行されたDISABLE, ENABLE, PURGE, RECEIVE, SENDの各文の状態条件を表わす。

    データ名-6のデータ項目の内容と状態条件との実際の関係は、一般規則1.に示した。

2.2 通信機能単位における手続き部

2.2.1 ACCEPT MESSAGE COUNT（通信文個数の入力）文

ACCEPT MESSAGE COUNT（通信文個数の入力）文は、待ち行列中の通信文の個数を入手する。

一般形式

```
ACCEPT 通信記述名 MESSAGE COUNT
```

構文規則

1. 通信記述名は、入力通信記述項を参照しなければならない。

一般規則

1. TACCEPT MESSAGE COUNT文は、通信記述名用に指定されているMESSAGE COUNTフィールドを、主待ち行列や副待ち行列-1などの中に存在する通信文の個数を反映するように更新する。
2. ACCEPT MESSAGE COUNT文を実行するときには、通信記述項の各項目のうち、最低限、通信文の個数を調べる対象の主待ち行列の名前が設定されていること。この文が実行されると、データ名-10のデータ項目（STATUS KEY）とデータ名-11のデータ項目（MESSAGE COUNT）が適宜更新される。（この章で前述した「2.2-2 *通信記述項の全体的な骨組み* 」を参照）

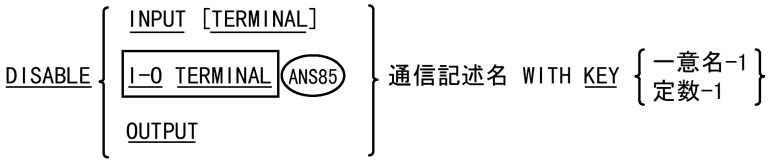
2.2.2 DISABLE（切離し）文

DISABLE（切離し）文は、指定された出力待ち行列と出力宛先との間または指定された入力源と入力待ち行列との間の、データの転送を停止するよう、通信管理システムに通知する。

(ANS85) DISABLE文のKEY指定は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に削除される予定である。

(MF) この構文は、Micro Focus COBOLに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

一般形式



構文規則

1. INPUT指定をしたときは、通信記述名は入力通信記述項を参照すること。
2. (ANS85)I-O TERMINAL指定をしたときは、通信記述名は入出力通信記述項を参照すること。
3. OUTPUT指定をしたときは、通信記述名は出力通信記述項を参照すること。
4. 定数-1または一意名-1のデータ項目の内容は、英数字と定義されていること。

一般規則

1. DISABLE文は、通信管理システムと指定された発信源または宛先との間を、論理的に切り離す。既に論理的な切り離しが行われている場合、またはプログラムの外部で論理的な切り離しが行われる場合、そのプログラム内ではDISABLE文は必要ない。COBOLプログラムと通信管理システムとの間のデータ転送のための論理経路は、DISABLE文の影響を受けない。
2. INPUT指定に補助語のTERMINALを付けると、発信源とすべての主待ち行列および副待ち行列との間の論理経路が閉鎖される。すると、通信記述項の項目のうちデータ名-7のデータ項目（SYMBOLIC SOURCE）の内容だけが意味を持つ。
3. INPUT指定に補助語のTERMINALを付けないと、通信記述項の項目のうちデータ名-1（SYMBOLIC QUEUE）からデータ名-4（SYMBOLIC SUB-QUEUE-3）までの主待ち行列、および副待ち行列に関連するすべての発信源との論理経路が閉鎖される。
4. (ANS85)I-O TERMINAL指定をすると、発信源（データ名-3のデータ項目（SYMBOLIC TERMINAL）の内容によって定義される）とプログラムとの間の論理経路が閉鎖される。
5. OUTPUT指定をすると、通信記述項の項目のうちデータ名-5のデータ項目（SYMBOLIC DESTINATION）の内容によって定義される、すべての宛先との論理経路が閉鎖される。
6. 定数-1または一意名-1のデータ項目の内容はシステムに組み込まれているパスワードと照合される。両者が一致した場合にだけ、DISABLE文は受け付けられる。両者が一致しないと、通信記述項の項目のうちのSTATUS KEYの値が更新される。

   通信管理システムは、1文字以上10文字以内のパスワードを取り扱えなければならない。
7. 通信管理システムは、発信源または宛先がアクティブでない最も早い段階でDISABLE文を実行するように保証する。しかし、DISABLE文を実行することによって、端末との間で送信または受信中の通信文の残りの部分が送られなくなることはない。

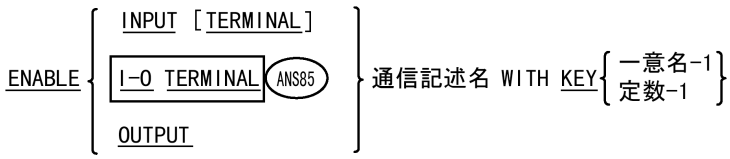
2.2.3 TENABLE（接続）文

ENABLE文のKEY指定は、ANS '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に削除される予定である。

(ANS85) ENABLE（接続）文は、指定された出力待ち行列と出力宛先との間、または指定された入力源と入力待ち行列との間のデータの転送を可能にするよう、通信管理システムに通知する。

(MF) この構文は、Micro Focus COBOLに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。（詳細については、『COBOLシステムリファレンス』を参照）

一般形式



構文規則

1. INPUT指定をしたときは、通信記述名は入力通信記述項を参照すること。
   (ANS85)I-O TERMINAL指定をしたときは、通信記述名は入出力通信記述項を参照すること。
2. OUTPUT指定をしたときは、通信記述名は出力通信記述項を参照すること。
3. 定数-1または一意名-1のデータ項目の内容は英数字と定義されていること。

一般規則

1. ENABLE文は、通信管理システムと指定された発信源または宛先との間を、論理的に接続する。既に論理的な接続が行われている場合、またはプログラムの外部で論理的な接続が行われる場合、そのプログラム内ではENABLE文は必要ない。COBOLプログラムと通信管理システムとの間のデータ転送のための論理経路は、ENABLE文の影響を受けない。
2. INPUT指定に補助語のTERMINALを付けると、発信源とすべての主待ち行列および副待ち行列との間の論理経路のうちで既に接続されているものが開通される。こうすると、通信記述項の項目のうちデータ名-7のデータ項目（SYMBOLIC SOURCE）の内容だけが、通信管理システムに対して意味を持つ。
3. INPUT指定に補助語のTERMINALを付けないと、通信記述項の項目のうちデータ名-1（SYMBOLIC QUEUE）からデータ名-4（SYMBOLIC SUB-QUEUE-3）までの主待ち行列、および副待ち行列に関連するすべての発信源との論理経路が開通される。

4. (ANS85)I-O TERMINAL指定をすると、発信源（データ名-3のデータ項目（SYMBOLIC TERMINAL）の内容によって定義される）とプログラムとの間の論理経路が開通される。
5. OUTPUT指定をすると、通信記述項の項目のうちデータ名-5のデータ項目（SYMBOLIC DESTINATION）の内容によって定義される、すべての宛先との論理経路が開通される。
6. 定数-1または一意名-1のデータ項目の内容は、システムに組み込まれているパスワードと照合される。両者が一致した場合にだけ、ENABLE文は受け付けられる。両者が一致しないと、通信記述項の項目のうちのSTATUS KEYの値が更新される。
   通信管理システムは、1文字以上10文字以内のパスワードを取り扱えなければならない。

2.2.4 PURGE（取消し）文

(ANS85)

PURGE（取消し）文は、いくつかのSEND文によって引き渡された部分的な通信文を通信管理システムから除去する。

一般形式

PURGE 通信記述名

構文規則

1. 通信記述名は、出力通信記述項または入力通信記述項を参照しなければならない。

一般規則

1. PURGE文を実行すると、通信記述名の通信記述項中に指定されている宛先への伝送待ちの部分的な通信文が、通信管理システムによって除去される。
2. EMIまたはEGIに関連する通信文は、いずれもPURGE文の実行によって影響されることはない。
3. 通信記述名-1によって参照され領域の状態キー・データ項目の内容および誤りキー・データ項目の内容は、（該当すれば）通信管理システムによって更新される。（この章で前述した「2.2-2 *通信記述項の全体的な骨組み*」を参照）

2.2.5 RECEIVE（受信）文

RECEIVE（受信）文は、通信文、通信行、通信文または通信行の一部、およびそれらのデータに関する情報を、通信管理システムによって保持されている待ち行列からCOBOLプログラムに引き渡す。該当するデータが何もないときは、無条件文を実行することができる。

一般形式

```
RECEIVE 通信記述名 { MESSAGE / SEGMENT } INTO 一意名-1

    [NO DATA 無条件文-1]

    [WITH DATA 無条件文-2]                (ANS85)
    [END-RECEIVE]
```

構文規則

1. 通信記述名は、入力通信記述項 (ANS85)または入出力通信記述項 を参照しなければならない。

一般規則

1. 通信記述名が入力通信記述項を参照する場合、その中のデータ名-1（SYMBOLIC QUEUE）からデータ名-4（SYMBOLIC SUB-QUEUE-3）までのデータ項目の内容は、通信文を収録している待ち行列の構造を表わす。（この章で前述した「2.2-2　*通信記述項の全体的な骨組み*n」を参照）
2. (ANS85)通信記述名が入出力通信記述項を参照する場合、その中のデータ名-3（SYMBOLIC TERMINAL）のデータ項目の内容は、通信文の発信源を表わす。
3. 通信文、通信行、通信文または通信行の一部は、一意名-1の領域の受信文字位置に送られる。送られたデータは左詰めにされる。後部に空白は埋められない。
4. RECEIVE文を実行した結果、利用可能なデータが通信管理システムによって一意名-1のデータ項目に収められた場合は、NO DATAが指定されていても無視される。そして、WITH DATAが指定されていなければ、制御はRECEIVE文の末尾に移される。
   (ANS85)WITH DATAが指定されていれば、制御は無条件文-2に移される。制御が無条件文-2に移された場合、その中に記述されている個々の文に関する規則に従って、プログラムの実行が続けられる。このとき、明示的に制御を移転させる手続きの分岐または条件文が出てきた場合は、その文の規則に従って制御が移される。そうでない場合は、無条件文-2の実行が終了すると、制御はRECEIVE文の末尾に移される。
5. RECEIVE文を実行した結果、利用可能なデータが通信管理システムによって一意名-1のデータ項目に収められなかった場合は、下記の3つの処理のうちのどれかが行われる。どのような条件のときにデータが利用可能ではないかは、作成者が定義する。
   a. RECEIVE文中にNO DATAが指定されている場合、制御は無条件文-1に移される。この場合、無条件文-1の中に記述されている個々の文に関する規則に従って、プログラムの実行が続けられる。このとき、明示的に制御を移転させる手続きの分岐または条件文が出てきた場合は、その文の規則に従って制御が移される。そうでない場合は、無条件文-1の実行が終了すると、制御はRECEIVE文の末尾に移される。
      (ANS85)WITH DATAは、指定されていても無視される。

b. RECEIVE文中にNO DATA指定がなされていない場合、一意名-1のデータ項目中に利用可能なデータが収められるまで、実行用プログラムの処理は中断される。

c. いくつかの主待ち行列
(ANS85)または副待ち行列
が通信管理システムにとって未知であると、適切な状態コードが設定され、次いでデータが利用可能になった場合と同様に制御が移される。

6. RECEIVE文が実行されるたびに、通信記述名の通信記述項に含まれるデータ項目が、通信管理システムによって適切に更新される。
7. RECEIVE文を1回実行することによって、一意名-1のデータ項目に、2件以上の通信文（MESSAGE指定をした場合）または2件以上の通信行（SEGMENT指定をした場合）が返されることは決してない。しかし、入力待ち行列中で通信文全体が利用可能になるまで、通信管理システムが実行用プログラムに通信文の一部を引き渡すことはない。このことは、RECEIVE文にSEGMENT指定をした場合でも当てはまる。
8. MESSAGE指定をした場合、通信行の終了記号は無視されて、下記の規則に従ってデータが転送される。
    a. 通信文の大きさが一意名-1の領域の大きさと等しいと、その通信文はその領域に格納される。
    b. 通信文の大きさが一意名-1の領域の大きさよりも小さいと、その通信文はその領域に左詰めで格納される。後ろに空白は埋められない。
    c. 通信文の大きさが一意名-1の領域の大きさよりも大きいと、その通信文は左端からその領域にいっぱいになるまで左詰めで格納される。その通信文の残りの部分は、同じ主待ち行列および副待ち行列を指定してRECEIVE文を繰り返し実行することによって、一意名-1の領域に転送することができる。　上記の8a，8b，8cの規則を適用するに際しては、通信文の残りの部分は新しい通信文と同様に扱われる。
9. SEGMENT指定をした場合、下記の規則に従ってデータが転送される。
    a. 通信行の大きさが一意名-1の領域の大きさと等しいと、その通信行はその領域に格納される。
    b. 通信行の大きさが一意名-1の領域の大きさよりも小さいと、その通信行はその領域に左詰めで格納される。後ろに空白は埋められない。
    c. 通信行の大きさが一意名-1の領域の大きさよりも大きいと、その通信行は左端からその領域にいっぱいになるまで左詰めで格納される。その通信行の残りの部分は、同じ主待ち行列および副待ち行列を指定してRECEIVE文を繰り返し実行することによって、一意名-1の領域に転送することができる。　上記の9a，9b，9cの規則を適用するに際しては、通信行の残りの部分は新しい通信行と同様に扱われる。
    d. RECEIVE文によって呼び出されるテキスト中に通信文の終了記号または通信群の終了記号があると、暗黙の通信行の終了記号があるものと解釈されて、そのテキストは通信行として扱われる。

10. RECEIVE文の実行によって一度通信文の一部が引き渡されると、その実行単位内で残りの部分を受け取るには、引続きRECEIVE文を実行しなければならない。
11. (ANS85)END-RECEIVE指定は、RECEIVE文の範囲を区切る。（第2章「*COBOLの概念*」の2.9-2中「*範囲明示文*」の節を参照）

2.2.6 SEND（送信）文

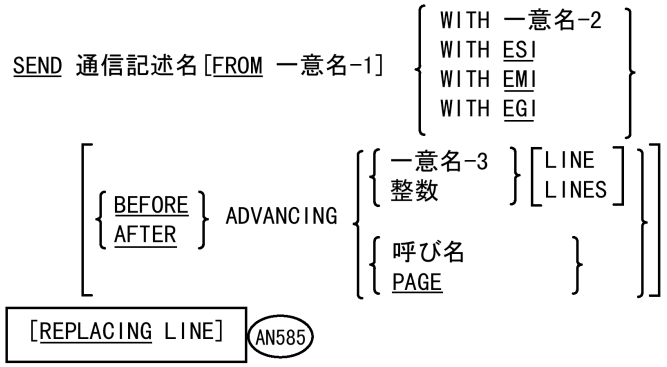
SEND（送信）文は、通信文、通信行、通信文または通信行の一部を、通信管理システムによって保持されているいくつかの出力待ち行列へ引き渡す。

一般形式

書き方1

```
SEND 通信記述名 FROM 一意名-1
```

書き方2



構文規則

1. 通信記述名は、出力通信記述項
   (ANS85)または入出力通信記述項 を参照すること。
2. 一意名-2は、演算符号の付かない1文字の整数とする。
3. ADVANCING指定の中で使用するときは、一意名-3は基本整数項目の名前とする。
4. W呼び名を指定すると、その名前は特別の機能を持つものとして、識別される。呼び名は、環境部の特殊名段落に定義されている。
5. 整数または一意名-3のデータ項目の値は、ゼロであってもよい。
6. (ANS85)一意名-1が関数一意名であるときは、英数字関数を参照しなければならない。

一般規則

すべての書き方

1. 受信側の通信装置の行の長さが固定の場合（プリンタ、表示画面、カード・パンチなど）
    a. 各通信文または通信行は、物理行の左端の文字位置から開始される。
    b. 物理行の長さよりも短い通信文または通信行は、右側に空白を埋められる。
    c. 物理行の長さを超える通信文または通信行が切り捨てられることはない。送信テキストは、物理行と等しい長さにパックされてから受信装置に送られる。物理行を超えるテキストは、次の行に継続して送られる。
2. 受信側の通信装置の行の長さが可変の場合（紙テープ・パンチ、他のコンピュータなど）、各通信文または通信行は、通信装置の次の利用可能な文字位置から開始される。
3. SEND文を実行する一環として、通信管理システムは通信記述名の領域中のデータ名-2のデータ項目（TEXT LENGTH）の内容を、送り出す通信文の文字数と解釈する。そして、SEND文に指定された一意名の領域の左端の文字からこの文字数に達するまでを、順に送信する。

   データ名-2のデータ項目（TEXT LENGTH）の内容がゼロである場合、一意名-1のデータ項目の内容は送信されない。

   データ名-2のデータ項目（TEXT LENGTH）の長さがゼロから一意名-1のデータ項目の長さのまで範囲を超える場合、データ名-3のデータ項目（STATUS KEY）に誤りの理由を示す値が設定され、データは送信されない。（状態については、表2-1を参照）
4. SEND文を実行する一環として、通信記述名の領域中のデータ名-3のデータ項目（STATUS KEY）の内容は、通信管理システムによって更新される。（この章で前述した「2.2-2 *通信記述項の全体的な骨組み*」を参照）
5. 一意名-1のデータ項目の内容に特殊な制御文字が含まれている場合、結果はどうなるかわからない。
6. 書き方1のSEND文を1回実行すると、通信文または通信行の一つの部分だけが通信管理システムに引き渡される。

   書き方2のSEND文を1回実行することによって、一意名-2のデータ項目の内容、または ESI，EMI，EGIによって指定される2件以上の通信文または2件以上の通信行が引き渡されることは決してない。

   しかし、通信文全体が出力待ち行列中に入り切るまでは、通信管理システムは通信文のどの部分も通信装置に送ることはない。
7. 実行単位の実行中に、EMIまたはEGIによって末尾を区切られていないか、
   (ANS85)またはPURGE文を実行することによって除去されていない
   通信文の一部は、どう処置されるかわからない。しかし、その通信文は論理的には存在しないため、宛先に送ることはできない。
8. SEND文の実行によって一度通信文の一部が通信管理システムに引き渡されると、その実行単位内で残りの部分を引き渡すには、引続きSEND文を実行しなければならない。

書き方2

9. 一意名-2のデータ項目の内容は、一意名-1のデータ項目の内容に終了記号を付けることを指定する。この終了記号には、通信行の終わりを示すものと、通信文の終わりを示すものと、伝送の終わりを示すものとがある。詳細については、下の表にまとめる。

一意名-2の データ項目の内容	一意名-1のデータ項目に 付け加えられるもの	意 味
"0"	インディケータなし	インディケータなし
"1"	通信行終了記号（ESI）	通信行の終わり
"2"	通信文終了記号（EMI）	通信文の終わり
"3"	通信群終了記号（EGI）	通信群の終わり
"1","2","3"以外の文字は"0"として解釈される		

一意名-2のデータ項目の内容が"1"でも"2"でも"3"でもなく、一意名-1が指定されていないと、データ名-3のデータ項目（STATUS KEY）に誤りの原因を示す値が設定され、データは送られない。

10. ESIは、通信行の終わりを通信管理システムに知らせる。
    EMIは、通信文の終わりを通信管理システムに知らせる。
    EGIは、通信群の終わりを通信管理システムに知らせる。通信管理システムがEGIをどう解釈するかは、ランタイムシステムによって指定される。　通信管理システムはこれらの終了記号を認識して、通信群、通信文、通信行を制御するために必要な措置をとる。
11. これらの終了記号は、EGI，EMI，ESIの順に階層を構成している。 EGIの前には必ずしもESIやEMIがある必要はない。EMIの前には必ずしもESIがある必要はない。
12. ADVANCING指定をすると、縦方向の位置決めができる通信装置上で、各通信文または通信行の縦方向の位置付けを制御できる。　縦方向の位置決めができない通信装置に関しては、通信管理システムは明示的または暗黙的に指定された縦方向の位置付けを無視する。
13. 指定した一意名-2のデータ項目の内容がゼロである場合、ADVANCING指定 (ANS85)およびREPLACING指定 をしたとしても、通信管理システムによって無視される。
14. 縦方向の位置付けができる装置に対してADVANCING指定をしなかった場合、AFTER ADVANCING 1 LINEを指定したものと、自動的に想定される。
15. 縦方向の位置付けができる装置に対して明示的または暗黙的にADVANCING指定をした場合、下記の規則が適用される。
    a. 一意名-3または整数を指定すると、その値と等しい行数、通信装置に送られるテキストの前または後で改行される。（下記のd.およびe.を参照）
    b. (ANS85)一意名-3のデータ項目の値が負であると、結果はどうなるかわからない。
    c. 呼び名を指定すると、通信装置に送られたテキストは、該当する通信装置用に定義されている規則に従って位置付けられる。

d. BEFORE指定をすると、上記の一般規則15a.
(ANS85)および15b.
に従って縦方向の位置付けが行われる前に、通信文または通信行が通信装置上に表示される。

e. AFTER指定をすると、上記の一般規則15a.
(ANS85)および15b.
に従って縦方向の位置付けが行われた後で、通信文または通信行が通信装置上に表示される。

f. PAGE指定をすると、通信装置に送られたテキストは、装置が次のページに位置付けられる前または後（BEFOREとAFTERのどちらを指定したかによる）に表示される。　改ページ機能のない装置にPAGE指定をすると、ADVANCING 1 LINEを指定したものと想定される（BEFOREかAFTERかは、その指定による）。

16. (ANS85)同じ位置に2つ以上の文字を表示できる文字・画像通信装置上で、後から送られた文字を既に送られている文字に重ねるか置き換えるかについては、下記の規則を適用する。

a. REPLACING指定をすると、SEND文によって送信された文字はすべて、同じ行に既に送信されている文字があれば、その行の左端から開始して上書きされる。

b. REPLACING指定をしないと、SEND文によって送信された文字はすべて、同じ行に既に送信されている文字があれば、その行の左端から開始して重ね合わされる。

17. (ANS85)受信側の通信装置が文字の置き換えられない種類のものである場合は、REPLACING指定の有無にかかわらず、SEND文によって送信された文字はすべて、同じ行に既に送信されている文字があればその行の左端から開始して重ね合わされる。

18. (ANS85)受信側の通信装置が2つ以上の文字を重ね合わすことのできない種類のものである場合は、REPLACING指定の有無にかかわらず、SEND文によって送信された文字はすべて、同じ行に前に送信されている文字があれば、その行の左端から開始して上書きされる。

第3章: デバッグ機能単位

デバッグ機能単位（debug module）は、実行用プログラムを実行中に手続きを監視する条件を表現する手段を利用者に提供する。
(ANS85)ANSIデバッグ機能単位は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、Micro Focus COBOLに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、デバッグ機能単位は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの機能単位を使用するべきではない。

3.1 標準ANSI COBOL デバッグ

何を監視しどのような情報を表示するかは利用者が決めることである。COBOLのデバッグ機能は、単に関連する情報を容易に取り出す手段を提供するだけである。
COBOLのデバッグ機能単位をサポートする言語機能には、下記のものがある。

- WITH DEBUGGING MODEスイッチ - 実行用プログラムを生成するときに使用する
- ランタイムスイッチ
- USE FOR DEBUGGING文
- 特殊レジスタ - DEBUG-ITEM
- デバッグ行

予約語の DEBUG-ITEMは、デバッグ機能をサポートするCOBOLシステムによって自動的に生成される、特殊レジスタの名前である。DEBUG-ITEMは1プログラムあたりに1つだけ割り当てられる。DEBUG-ITEMの下位のデータ項目の名前も、予約語である。

3.1.1ランタイムスイッチ

環境部の翻訳用計算機段落の中に、DEBUGGING MODE句を書く。これはプログラム中に記述されているデバッグ文に関する、ランタイムスイッチとしての働きをする。
プログラム中に WITH DEBUGGING MODE句を指定すると、すべてのデバッグ節とすべてのデバッグ行が、このマニュアルのこの節に記載したとおりに機能する。
プログラム中にDEBUGGING MODE句を指定しないと、すべてのデバッグ節とすべてのデバッグ行は、注記行のように扱われ、その構文のチェックも行われない。

3.1.2 COBOL デバッグランタイムスイッチ

ランタイムスイッチは、COBOLシステムによって挿入されたデバッグ用のコードを動的に起動する。このスイッチは、プログラムの中からは操作できない。このスイッチは、COBOL環境の外部で制御される。このスイッチが「オン」であると、原始プログラム中に記述されているUSE FOR DEBUGGING文が効力を発揮する。このスイッチが「オフ」であると、原始プログラム中に記述されているUSE FOR DEBUGGING文の効力はすべて停止される。USE FOR DEBUGGING文の効力を発揮させたり停止したりするために、実行用プログラムを生成し直す必要はない。
原始プログラム中にWITH DEBUGGING MODE句を指定しないと、ランタイムスイッチは実行用プログラムの実行に何の影響も及ぼさない。

3.2 COBOL デバッグにおける環境部

3.2.1 WITH DEBUGGING MODE 句

WITH DEBUGGING MODE句は、すべてのデバッグ節とすべてのデバッグ行を実行用プログラムに含めるように指示する。この句を指定しないと、すべてのデバッグ節とすべてのデバッグ行は、注記行のように扱われる。
WITH DEBUGGING MODE句は中核でも使用でき、ANSI '85の廃要素には指定されていない。

一般形式

一般規則

1. 環境部の 翻訳用計算機段落の中にWITH DEBUGGING MODE句を指定すると、すべてのUSE FOR DEBUGGING文とすべてのデバッグ行が実行用プログラムに含められる。
2. 環境部の翻訳用計算機段落の中にWITH DEBUGGING MODE句を指定しないと、どのUSE FOR DEBUGGING文も、それに関連するすべてのデバッグ節もデバッグ行も、注記文であるように扱われる。

注: WITH DEBUGGING MODE句はANSIデバッグ機能単位と同様に、中核の一部である。ANSI標準の次回の全面改訂の際にも、削除される予定はない。

3.3 COBOL デバッグにおける手続き部

3.3.1 READY TRACE（追跡開始）文

(OSVS)(VSC2)

READY TRACE（追跡開始）文は、デバッグ用の機能を果たす文である。処理の流れが経過した順に、節名と段落名を画面に表示するようにする。

一般形式

```
READY TRACE.
```

一般規則

1. この文の実行を可能にするためには、COBOLシステムのTRACE指令をオンに設定しておく。TRACE指令をオンに設定しておかないと、READY TRACE文は注記になる。
2. (VSC2)READY TRACEは受け付けられるが、注記としてだけ扱われる。

3.3.2 RESET TRACE（追跡停止）文

(OSVS)(VSC2)

RESET TRACE（追跡停止）文は、節名および段落名を追跡して画面に表示する機能を、停止する。

一般形式

```
RESET TRACE.
```

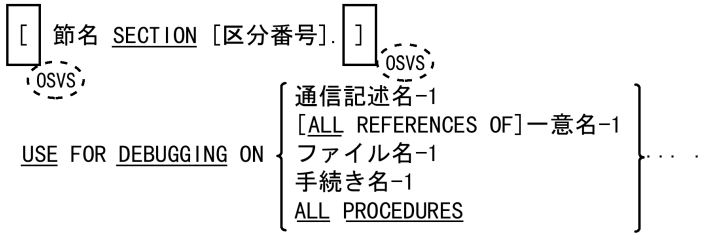
一般規則

1. この文の実行を可能にするためには、COBOLシステムの TRACE指令をオンに設定しておく。TRACE指令をオンに設定していないかREADY TRACEが起動されていないと、RESET TRACE文は注記になる。
2. (VSC2)RESET TRACEは受け付けられるが、注記としてだけ扱われる。

3.3.3 USE FOR DEBUGGING（デバッグ用使用）文

USE FOR DEBUGGING（デバッグ用使用）文は、関連するデバッグ節によって監視する、利用者項目を指定する。

一般形式



ランタイムスイッチ

1. 次のランタイムスイッチは、この節で説明した構文や意味に影響することがある。
    - D - ANSIデバッグ機能の起動

構文規則

1. デバッグ節を指定する場合は、 DECLARATIVESヘッダ（宣言部分の見出し）の直後にまとめて置く。
2. USE FOR DEBUGGING文そのものを除いて、デバッグ節の中から非宣言手続きを参照してはならない。
   (OSVS)(VSC2)(MF)この制限は強制しない。
3. デバッグ節の外に記述した文から、デバッグ節の中に定義した手続き名を参照してはならない。
4. USE FOR DEBUGGING文そのものを除いて、あるデバッグ節の中に記述した文から別のUSE手続き内に定義されている手続き名を参照するために使用できる文は、PERFORM文だけである。
5. デバッグ節内に定義した手続き名を、USE FOR DEBUGGING文の中に指定してはならない。
6. どの識別子、通信記述名、ファイル名、手続き名も、1つのUSE FOR DEBUGGING文の中では1回だけ指定することができる。
7. ALL PROCEDURES（すべての手続き）指定は、同一のプログラム中に1回だけ書くことができる。
8. ALL PROCEDURES指定をした場合、どのUSE FOR DEBUGGING文の中にも手続き名-1、手続き名-2などを指定してはならない。
9. 一意名-1、一意名-2などによって参照されるデータ項目のデータ記述項が、OCCURS句を含んでいるかまたはOCCURS句を含むデータ記述項の下位に属する場合、通常は、一意名-1、一意名-2などには必要な添字または指標を付けてはならない。
10. 特殊レジスタである DEBUG-ITEM DEBUG-ITEMは、デバッグ節からだけ参照できる。

一般規則

1. デバッグ節の中に記述した文によって、デバッグ節が自動的に起動されることはない。
2. USE FOR DEBUGGING文の中にファイル名-1を指定すると、下記の場合に、該当するデバッグ節が実行される。
    a. ファイル名-1を参照するOPEN文またはCLOSE文が実行された後
    b. （USE手続きの後に指定されている）READ文が実行された後で、付随するAT END指定またはINVALID KEY指定の無条件文が実行されない場合
    c. ファイル名-1を参照するDELETE文またはSTART文が実行された後
3. USE FOR DEBUGGING文の中に手続き名-1を指定すると、下記の場合に、該当するデバッグ節が実行される。
    a. 指定した手続きが実行される直前
    b. 手続き名-1を参照するALTER文が実行された直後
4. ALL PROCEDURE指定をすると、上記の一般規則4に記述した事柄が、該当するプログラム中のすべての手続き名に適用される。ただし、デバッグ節の中に指定した手続き名を除く。
5. ALL REFERENCES OF（すべての参照）句を付けて一意名-1を指定すると、識別子-1を明示的に参照する下記の文の実行に関連して、該当するデバッグ節が実行される。
    a. WRITE文またはREWRITE文が実行される直前。ただし、FROM句が指定されている場合は、暗黙の転記が実行された後。
    b. DEPENDING ON指定を伴うGO TO文に関して、制御が移される直前、および制御が移される先の手続き名に関連するデバッグ節が実行される前。
    c. PERFORM文の中で一意名-1を参照するVARYING指定かAFTER指定かUNTIL指定をしている場合、その一意名-1によって参照されるデータ項目の内容が初期化または変更または評価された直後。
    d. その他のCOBOL文が実行された直後。

    実行も評価もされない指定の中に一意名-1を書いた場合、関連するデバッグ節は実行されない。
6. ALL REFERENCES OF指定をしないで一意名-1を指定すると、下記のそれぞれの時点で、該当するデバッグ節が実行される。
    a. 一意名-1を明示的に参照するWRITE文またはREWRITE文が実行される直前。ただし、FROM句が指定されている場合は、暗黙の転記が実行された後。
    b. PERFORM文の中で一意名-1を参照するVARYING指定かAFTER指定かUNTIL指定をしている場合、その一意名-1によって参照されるデータ項目の内容が初期化または変更または評価された直後。
    c. 一意名-1を明示的に参照しその一意名によって示されるデータ項目の内容を変更する、その他のCOBOL文が実行された直後。

    実行も評価もされない指定の中に一意名-1を書いた場合、関連するデバッグ節は実行されない。
7. 1つの文に関しては、同じ作用対象を何回指定しても、その作用対象に関するデバッグ節は1回しか実行されない。対象とする手続きを繰り返し実行するPERFORM文の場合は、各繰り返しのたびに、関連するデバッグ節が実行される。

1つの無条件文の中では、個々の無条件動詞は、デバッグの目的上は別々の文とみなされる。

8. USE FOR DEBUGGING文中に通信記述名-1を指定した場合、デバッグ節は下記の場合に実行される。
    a. 通信記述名-1を参照するENABLE文かDISABLE文かSEND文が実行された後。
    b. 通信記述名-1を参照するRECEIVE文が実行された後。ただし、NO DATA無条件文が実行された場合を除く。
    c. 通信記述名-1を参照するACCEPT MESSAGE COUNT文が実行された後。
9. 修飾語としてファイル名-1、一意名-1、手続き名-1、通信記述名-1を使用した場合は、上記の一般規則に記述されているデバッグのための参照にはあたらない。
10. デバッグ節が実行されるたびに、特殊レジスタの DEBUG-ITEMに、その実行の基となった条件に関する情報が記録される。 DEBUG-ITEMは、暗黙的に下記のように記述されている。

```
 01 debug-item.
     02 debug-line      pic x(6).
     02 filler          pic x value space.
     02 debug-name      pic x(30).
     02 filler          pic x value space.
     02 debug-sub-1     pic s9999 sign is leading separate character.
     02 filler          pic x value space
     02 debug-sub-2     pic s9999 sign is leading separate character.
     02 filler          pic x value space.
     02 debug-sub-3     pic s9999 sign is leading separate character.
     02 filler          pic x value space.
     02 debug-contents  pic x(n).
```

11. 毎回デバッグ節が実行される前に、データ項目DEBUG-ITEMの内容が空白で埋められる。その後で、DEBUG-ITEMの下位項目の内容が下記の一般規則に従って更新され、それから直ちに制御がデバッグ節に移される。下記の一般規則に触れられていないデータ項目の内容は、空白のまま残る。
    更新はMOVE（転記）文の規則に従って行われる。ただし、DEBUG-CONTENTSへの転記は唯一の例外で、データの内部表現の型に違いがあっても変換することなく、英数字基本項目間の転記として扱われる。
12. DEBUG-LINEの内容は、対応するCOBOL原始プログラムの行番号である。これは原始プログラム中の該当行を識別する手段となる。
13. DEBUG-NAMEの内容は、デバッグ節を起動した要因の名前の上30文字である。
    DEBUG-NAME中では、すべての修飾語は語INまたはOFで区切られている。
    添字や指標は、付けられていても、DEBUG-NAMEには入れられない。
14. デバッグ節を起動する要因となったデータ項目が添字付けまたは指標付けされている場合、各レベルの出現番号がそれぞれDEBUG-SUB-1とDEBUG-SUB-2とDEBUG-SUB-3に入れられる。
15. DEBUG-CONTENTSは、下記の一般規則によって必要となるデータを記憶しておくのに十分な大きさのデータ項目である。
16. プログラム中の宣言手続きではない最初の手続きが最初に実行されたときに、デバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、その手続きの最初の文を識別する情報が入れられる。

b. DEBUG-NAMEには、その手続きの名前が入れられる。
c. DEBUG-CONTENTSには、"START PROGRAM" が入れられる。

17. ALTER文中で手続き名-1を参照したことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、手続き名-1を参照したALTER文を識別する情報が入れられる。
    b. DEBUG-NAMEには、その手続き名-1が入れられる。
    c. DEBUG-CONTENTSには、ALTER文のTO指定の対象となった手続き名が入れられる。
18. GO TO文によって制御が移されたことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、手続き名-1に制御を移したGO TO文を識別する情報が入れられる。
    b. DEBUG-NAMEには、その手続き名-1が入れられる。
19. SORT文またはMERGE文のINPUT指定またはOUTPUT指定中の手続き名-1を参照したことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、手続き名-1を参照したSORT文またはMERGE文を識別する情報が入れられる。
    b. DEBUG-NAMEには、その手続き名-1が入れられる。
    c. DEBUG-CONTENTSには、下記のデータが入れられる。
        1. SORT文のINPUT指定において手続き名-1が参照された場合は、"SORT INPUT"。
        2. SORT文のOUTPUT指定において手続き名-1が参照された場合は、"SORT OUTPUT"。
        3. MERGE文のOUTPUT指定において手続き名-1が参照された場合は、"MERGE OUTPUT"。
20. PERFORM文の制御機構によって制御が移されたことによって、手続き名-1に関連するデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、手続き名-1を参照したPERFORM文を識別する情報が入れられる。
    b. DEBUG-NAMEには、その手続き名-1が入れられる。
    c. DEBUG-CONTENTSには、"PERFORM LOOP" が入れられる。
21. USE文によって手続き名-1が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、USE手続が実行される基となった文を識別する情報が入れられる。
    b. DEBUG-NAMEには、その手続き名-1が入れられる。
    c. DEBUG-CONTENTSには、"USE PROCEDURE" が入れられる。
22. 先行する段落から手続き名-1に暗黙的に制御が移されたことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、前の文を識別する情報が入れられる。
    b. DEBUG-NAMEには、手続き名-1が入れられる。

c. DEBUG-CONTENTSには、"FALL THROUGH" が入れられる。

23. ファイル名-1と通信記述名-1を参照したことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、ファイル名-1と通信記述名-1を参照した文を識別する情報が入れられる。
    b. DEBUG-NAMEには、ファイル名-1と通信記述名-1が入れられる。
    c. READ文でファイル名-1を参照した場合、DEBUG-CONTENTSには読み込まれたレコード全体が入れられる。
    d. READ文以外の文でファイル名-1を参照した場合、DEBUG-CONTENTSには空白が入れられる。
    e. 通信記述名-1を参照した場合、DEBUG-CONTENTSには通信記述名に関連する領域の内容が入れられる。
24. 一意名-1を参照したことによってデバッグ節が実行されると、DEBUG-ITEMの内容は下記のようになる。
    a. DEBUG-LINEには、一意名-1を参照した文を識別する情報が入れられる。
    b. DEBUG-NAMEには、一意名-1が入れられる。
    c. DEBUG-CONTENTSには、デバッグ節に制御が移されたときの一意名-1のデータ項目の内容が入れられる（一般規則の5と6を参照）。

3.3.4 デバッグ行

デバッグ行は、標識領域に"D" または "d" と書かれている行である。境界Aから境界Rまでがすべて空白であるデバッグ行は、空白行とみなされる。
デバッグ行の内容は、それが注記行とみなされるか否かにかかわらず、文法的に正しいプログラムを構成するようでなければならない。
翻訳用計算機段落中にWITH DEBUGGING MODE句を指定していないと、デバッグ行はすべて注記行とみなされる。
デバッグ行を連続して書いてもよい。デバッグ行を次の行に継続することもできる。ただし、後ろの各行の標識領域にも"D" または "d" を書く。行をまたがって文字列を書くことはできない。
デバッグ行を書くことができるのは、 実行用計算機段落よりも後ろだけである。

注: デバッグ行はANSI中核の一部であり、デバッグ機能単位に含まれるのではない。したがって、デバッグ行は廃要素には指定されておらず、次のANSI標準の全面改訂の際に削除される予定もない。

第4章：区分化

区分化機能単位（segmentation module）は、実行用プログラムのオーバレイに必要な情報をコンパイラに与える。

区分化（segmentation）機能を利用することによって、常駐区分（permanent segment）および独立区分（independent segment）を指定できる。また、区分番号（segment-number）が異なる節を混在させたり、原始プログラムの固定部分（fixed portion）にオーバレイされてもよい区分（segment）を含めたりすることができる。
(ANS85)区分化機能単位は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に削除される予定である。
(MF)この構文は、Micro Focus COBOLに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、区分化機能単位は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの機能単位を使用するべきではない。

4.1 区分化の一般説明

COBOLの区分化は、ユーザーが実行用プログラムのオーバレイに必要な情報をコンパイラに与えることができるようにする機能である。
COBOLでの区分化は、手続きだけを対象にしている。したがって、実行用プログラムを区分化するのに必要な事項は、手続き部に関してだけ規定する。

4.1.1 構成

4.1.1.1 プログラムの区分

原始プログラムの手続き部は、必ずしも節に分けて構成する必要はない。しかし、手続き部をいくつかの節の集団を連続して記述したものとして構成することはよく行われる。この場合、各集団は、密接に関連する一連の処理を集めて、全体として所定の機能を果たすように設計される。一方、区分化を適用するときは、手続き部全体をいくつかの節に分け、各節が実行用プログラムの固定部分または独立区分のどちらに入るのかを区別しておかなければならない。区分化をしても、手続き名を一意にするための修飾の必要性は変わらない。

4.1.1.2 固定部分

固定部分とは、実行用プログラムのうちの、論理的に常に記憶領域にあるように扱われる部分である、と定義される。固定部分は、常駐区分とオーバレイ可能な固定区分（fixed overlayable segment）から成る。
常駐区分とは、固定部分のうちで、プログラムの他のどの部分によってもオーバレイされない区分を言う。
オーバレイ可能な固定区分とは、固定部分のうちで、論理的には常に記憶領域中にあるように扱われるが、記憶領域の利用を最適化するために他の区分によってオーバレイされてもよいものを言う。

4.1.1.3 独立区分

独立区分とは、実行用プログラムのうちで、オーバレイ可能な独立区分や他の独立区分と、オーバレイしたりオーバレイされたりする部分である、と定義される。独立区分は、プログラムの実行中に初めて（明示的または暗示的に）その区分に制御が移されたときには、初期状態にある。また、以降その区分に制御が移されたときにも、下記の場合には独立区分は初期状態にある。

- 連続する別の区分番号の区分から、暗黙的に制御が移された場合。
- 別の区分番号の区分中のSORT（整列）文またはMERGE（併合）文から、独立区分中の入力手続きまたは出力手続きに暗黙的に制御が移された場合。
- 別の区分番号の区分から、明示的に制御が移された場合。ただし、下記の2.項は例外とする。

独立区分に2回目移行に制御が移されたとき、下記の場合は、最後に使用された状態になっている。

- 別の区分番号の区分から、暗黙的に制御が移された場合。ただし、上記の1.項を除く。
- EXIT PROGRAM（プログラムの出口）文
  (OSVS)(VSC2)(MF)またはGOBACK（復帰）文
  の結果として、明示的に制御が移された場合。

4.1.2 区分化の分類

区分化される節は、節番号および下記の基準に基づいて、分類される。

- 論理的要件
  常に参照可能でなければならない節、または頻繁に参照される節は、オーバレイ可能な固定区分または常駐区分とするのが普通である。あまり使用されない節は、独立区分にするのが普通である。
- 使用頻度
  一般に、節を参照する頻度が高ければ高いほど節番号を小さくし、参照する頻度が低ければ低いほど節番号を大きくする。

- 他の節との関係
  頻繁に連絡をとる節同士には、同じ節番号を与える。

4.1.3 区分化の制御

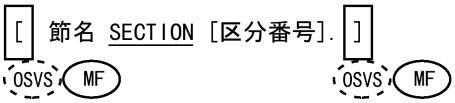
プログラムの論理的な順序は、通常は物理的な順序と同じである。ただし、制御の移行が行われたときは、両者の間にずれが生じる。原始プログラム中で制御を移す先は、節内の任意の段落とすることができる。これは、必ずしも節の先頭へ制御を移す必要はない、ということである。

4.2 プログラム区分の構造

4.2.1 区分番号

節の分類は、区分番号によって行われる。区分番号は、 節の見出しに含まれる。

一般形式



構文規則

1. 区分番号は、0から99までの整数とする。
2. 節の見出しに区分番号を書かなかったときは、区分番号は0であるとみなされる。
3. 宣言部分中の節の区分番号は、49以下とする。
4. (OSVS)(VSC2)区分番号は、符号付き整数とする。浮動小数点数定数であってはならない。

一般規則

1. 同じ区分番号のすべての節が、1つの区分を構成する。同じ区分番号を持つすべての節が、原始プログラム中で物理的に隣接していなくてもよい。
2. 区分番号が0から49までの区分は、実行用プログラムの固定部分を構成する。
3. 区分番号が50から99までの区分は、独立区分である。

4.2.2 常駐区分の範囲

一般形式

SEGMENT-LIMIT（常駐区分の範囲）句は、実行用計算機段落中に下記の形式で書く。

SEGMENT-LIMIT IS 区分番号

構文規則

1. 区分番号は、1から49までの整数とする。

一般規則

1. (MF)SEGMENT-LIMIT句は注記にとどまる。

4.3 プログラムの流れに関する制限

区分化を行うときは、ALTER，PERFORM，MERGE，SORTの各文に関して、下記の制限が課される。

4.3.1 ALTER文

区分番号が50以上の節内にあるGO TO文は、別の区分番号に属する節内のALTER文によって参照されてはならない。
それ以外のALTER文の使用法はすべて有効であり、ALTER文が参照するGO TO文がオーバレイ可能な固定区分中にあっても実行される。

4.3.2 PERFORM文

PERFORM文が固定区分中の節にある場合、これは範囲を持つ。その範囲内の文を実行することによって実行される宣言節の他に、下記の範囲がある。

- 固定部分中のいくつかの節や段落（複数の区分にまたがってもよい）。
- 単一の独立区分に完全に含まれている、いくつかの節や段落。

PERFORM文が独立区分中の節にある場合、これはその中に範囲を持つ。その範囲内の文を実行することによって実行される宣言節の他に、下記の範囲がある。

- 固定部分中のいくつかの節や段落（複数の区分にまたがってもよい）。
- そのPERFORM文と同じ独立区分に完全に含まれている、いくつかの節や段落。

4.3.3 MERGE文

固定区分中の節に書いたMERGE文によって参照される出力手続きは、下記の範囲になければならない。

- 固定区分に完全に含まれる。
- 単一の独立区分に完全に含まれる。

独立区分中に書いたMERGE文によって参照される出力手続きは、下記の範囲になければならない。

- 固定区分に完全に含まれる。
- そのMERGE文と同じ独立区分に完全に含まれる。

4.3.4 SORT文

固定区分中の節に書いたSORT文によって参照される出力手続きは、下記の範囲になければならない。

- 固定区分に完全に含まれる。
- 単一の独立区分に完全に含まれる。

独立区分中に書いたSORT文によって参照される出力手続きは、下記の範囲になければならない。

- 固定区分に完全に含まれる。
- そのSORT文と同じ独立区分に完全に含まれる。

第5章：2バイト文字支援機能

(VSC2)(MF)
世界の言語の中には、日本語のように文字の数が数千にも及ぶ文字集合を使用しているものが数多くある。一方、たいていの計算機では、文字を表わすのに8ビットを使用し、8ビット・コードのそれぞれの値に別々の文字を割り当てている。この方式だと、256種類の文字までしか表わすことができない。
理想的には、COBOLのプログラマは文字を表わすための内部コードを意識する必要がないことが望ましい。しかし、実際には内部コードの特性がプログラマに影響を与えることがある。中でも、表わせる文字の数が256しかないということは、特に大きな制約である。
この問題に対処するため、2バイト文字集合（double-byte character set:DBCS）が用意されている。この方式では、各文字は隣接する2バイトからなる16ビットのコードで表わされる。この結果、数千種類の文字を表わすことが可能となる。
2バイト・コードを具体的にどのように文字に割り当てるかは、国によって異なっている。
COBOLシステムで使用されている8ビット・コードは、情報交換用米国標準コード（ASCII）である。この章では、ASCIIコードのことを、1バイト文字集合（single-byte character set）と呼ぶことにする。
(MF)2バイト文字集合はDBCS指令の影響を受ける。
日本語支援の詳細については、*2バイト文字支援機能のMicro Focus 拡張*の章を参照。

5.1 2バイト文字（DBCS）データ

DBCS指令を使用すると、COBOLシステムは2種類のコード体系を認識するようになる。これによって、データを2バイト文字形式で格納できるようになる。しかし、DBCS指令は他のコード体系を使用できないようにするわけではない。したがって、DBCS指令を使用しても、1バイト文字形式でデータを格納することができる。
必要なハードウェアを備えていれば、2バイト文字の形式で入力や出力に使用されるデータ項目が認識されて、画面やキーボードやプリンタなどの装置にそれらのデータを適正に表示したり、データを受け取ったりすることができる。

5.2 2バイト文字集合中のローマ字

1バイト文字集合には、ローマ字のアルファベットとその他のいくつかの文字が含まれる。国によっては、2バイト文字集合の中に1バイト文字集合の中の文字の多くが含まれる。
ハードウェアによっては、データの格納方式が1バイトか2バイトかで、表示される文字の形態がはっきり異なるものがある。たとえば、画面によっては、2バイト・コードの文字の方が、1バイト・コードの文字よりも大きく表示される。

5.3 マルチベンダー統合体系支援機能

NTTのマルチベンダー統合体系（MIA）に合わせて作成したプログラムは、COBOLコンパイラによってコピーできる。このためには、DBCS指令およびCURRENCY-SIGN"92"指令を使用する。

5.4 原始プログラム

2バイト・コード文字は、定数（定数はデータであるため）、注記、注記項、利用者語に使用できる。これ以外については、DBCS指令は原始プログラム中で使用できる文字の範囲を変更しない。つまり、プログラムは依然としてCOBOLの文字集合(***言語リファレンス***の*COBOL言語の概念*の章を参照。）を使用して書かれている。

5.5 言語の拡張

2バイト文字のデータを含むデータ項目を定義するために、PICTURE句とUSAGE句が拡張されている。2バイト文字のデータ用には、新しい形式の定数が必要である。
2バイト文字のデータの取扱いを定義するために、各種の指定や句や文に規則が追加されている。
特に断わりがないかぎり、COBOLのすべての機能と規則は、2バイト文字を使用する際にも適用される。 以下に記述するのは、2バイト文字に関して追加されている規則と書き方だけである。

5.6 注記と注記項

注記および注記項の中では、1バイト文字と2バイト文字を自由に混ぜて書いてよい。

5.7 利用者語

利用者語の中では、1バイト文字か2バイト文字のどちらかを使用できる。利用者語を使用するものには、符号系名、字類名、条件名、データ名/一意名、レコード名、ファイル名、指標名、呼び名、段落名、節名、記号文字がある。
Ⓜ利用者語の中では、1バイト文字と2バイト文字を自由に混ぜて書いてよい。
ある文字が1バイト文字集合と2バイト文字集合の両方の中に存在するときは、それぞれに属する文字は異なるものとみなされる。(前述の*2バイト文字集合とローマ字*の節を参照。)

5.8 空白

字類が2バイト文字のデータ中の空白は、2バイト・コードで表わされる。2バイト・コードで表わされた空白文字を、「全角の空白」と呼ぶ。
Ⓜ全角の空白に割り当てられる値は、DBCS指令およびDBSPACE指令の影響を受ける。

5.9 データ項目

5.9.1 2バイト文字データ項目

言語リファレンスの*COBOL言語の概念*の章で説明した字類の他に「2バイト文字」を追加する。
字類の2バイト文字には、項類として、2バイト文字と2バイト文字編集の2つが含まれる。
字類が2バイト文字のデータ項目を表わすには、USAGE DISPLAY-1句を使用する。この句を指定したデータ項目のPICTURE文字列には、文字"G"と"B" だけを書くことができる。" G" は2バイト文字の文字位置を表わす。"B" は編集文字であり、全角の空白を挿入する文字位置を表わす。
PICTURE文字列がすべて"G" であるデータ項目の項類は、2バイト文字である。PICTURE文字列中に"G"と"B" が混在するデータ項目の項類は、2バイト文字編集である。
PICTURE文字の"G" と "B" は、どちらも1つの2バイト文字の文字位置を表わす。2バイト文字のデータ項目の長さは、そのPICTURE文字列中の"G" と "B" の数で表わす。ただし、例外として別途規定している場合もある。
部分参照では、最左端文字位置と長さはバイト数ではなく、2バイト文字の数で指定する。
字類が英数字のデータ項目を使用できる箇所であれば、どこでも字類が2バイト文字のデータ項目を使用してよい。ただし、この章で後述する該当する規則に従うものとする。また、これには例外もある。

5.9.2 混合データ項目

項類が英数字のデータ項目中に収めたデータの中に、2バイト文字を含めることができる。このようなデータにおいては、1バイト文字は1バイト・コードで表わされ、2バイト文字は2バイト・コードで表わされる。空白は半角で表わされる。
入力および出力のとき、1バイト文字も2バイト文字も認識される。2バイト文字の最初のバイトは1バイト文字では使われないコードであるので、両者を一緒に使用しても混同されることはない。しかし、プログラム内でデータを操作する際には、2バイト文字も通常の英数字と同様に扱われる。したがって、2バイト文字の2つのバイトが分割されないようにすることは、プログラマの責任である。
この型のデータ項目の長さは、すべての場合において、バイト単位で数える。

5.10 定数

5.10.1 2バイト文字定数

言語リファレンスの*COBOL言語の概念*の章で説明した文字定数と数字定数の他に、3番目の定数として2バイト文字定数が加えられている。
2バイト文字定数は、両端を引用符またはアポストロフィで囲み、その前に"G" を付けた文字列である。2バイト文字定数には、計算機の2バイト文字集合中の任意の文字を含めることができる。2バイト文字定数の長さは、2バイト文字で最長28文字までである。2バイト文字定数を行をまたがって継続することはできない。

引用符またはアポストロフィを区切り文字として使用した場合、2バイト文字定数の中にその区切り文字を含めるには、その文字を2つ続けて書く。区切り文字として使用していない引用符またはアポストロフィを2バイト文字定数の中に含める場合は、その文字を1つだけ書く。実行用プログラム内での2バイト文字定数の値は、文字列に下記の解釈を加えたものである。

1. 先頭の"G"と両端の区切り文字を除く。
2. 文字列中に含まれる2つの連続した区切り文字は、1つの文字として扱う。

5.10.1.1 2バイト文字定数の項類

文字定数を使用できる箇所であれば、どこでもすべての2バイト文字定数を使用することができる。ただし、この章で後述する該当する規則に従うものとする。また、これには例外もある。

5.10.2 混合定数

文字定数の中に2バイト文字を含めることができる。2バイト文字が含まれる文字定数を「混合定数」と呼ぶ。このような定数においては、1バイト文字は1バイト・コードで表わされ、2バイト文字は2バイト・コードで表わされる。空白は半角で表わされる。
出力のとき、1バイト文字も2バイト文字も認識される。2バイト文字の最初のバイトは1バイト文字では使われないコードであるので、両者を一緒に使用しても混同されることはない。しかし、プログラム内でデータを操作する際には、2バイト文字定数も通常の文字定数と同様に扱われる。したがって、2バイト文字の2つのバイトが分割されないようにすることは、プログラマの責任である。
文字定数の項類は英数字であり、2バイト文字ではない。このことは、2バイト文字が含まれるか否かに左右されない。
混合定数を、行をまたがって継続することはできない。
(MF)この制限は解除されている。

5.10.3 表意定数

（この章で記述する字類および項類に関する該当する規則に従って、）2バイト文字定数だけを使用できる箇所で表意定数を使用した場合、これは2バイト文字定数となる。この定数内の各空白は、全角である。
2バイト文字定数として使用できる表意定数は、SPACE（S）だけである。

5.10.4 "N"定数

(MF)(COB370)
COBOL/370 および MIA COBOL仕様においては、2バイト文字定数に相当する、別の形式の定数が使用されている。

一般形式

```
N"DBCS文字-1..."
```

構文規則

1. N定数には、最長18文字までの2バイト文字を含めることができる。N定数を行をまたがって継続することはできない。
2. N定数には、計算機の2バイト文字集合中の2バイト文字だけを含めることができる。
3. 1バイト文字と2バイト文字の引用符を同等視するとき、2バイト文字の引用符を定数中に含める場合は、その引用符を2つ続けて書かなければならない。たとえば、2バイト文字定数中に2バイト文字の引用符を含めるには、下記のように書く。
   N"ABC""DEF"
4. APOST指令を使用すると、N定数の仕様と動作を、G定数と同様に変更できる。つまり、引用符（2行）を書く代わりに、アポストロフィ（1行）を使用することができる。

一般規則

1. N定数をALLと共に使用して、表意定数を作成できる。(***言語リファレンス***の *COBOL言語の概念*の章を参照。)
2. すべての文字は、2バイト文字であること。

5.11 プログラム構造

5.11.1 プログラム終了見出し

構文規則

1. プログラム名に2バイト文字を含めてはならない。
   ㊂この制限は解除された。

5.12 2バイト文字機能単位における見出し部

5.12.1 プログラム名段落

構文規則

- プログラム名に2バイト文字を含めてはならない。
  ㊂この制限は解除された。
- 注記項には2バイト文字を含めてもよい。

5.13 2バイト文字機能単位における環境部

5.13.1 翻訳用計算機段落

構文規則

1. 翻訳用計算機名には、2バイト文字を含めてもよい。

5.13.2 実行用計算機段落

構文規則

1. 実行用計算機名には、2バイト文字を含めてもよい。

5.13.3 特殊名段落

構文規則

1. CURRENCY SIGN句において、定数-6は2バイト文字定数であってはならない。また、"G" (MF)(COB370)または "N" であってもならない。
2. ALPHABET句において、定数-1、定数-2、定数-3は2バイト文字定数であってはならない。
3. CLASS句において、定数-4および定数-5は2バイト文字定数であってはならない。

5.13.4 ファイル管理段落

構文規則

1. ASSIGN句において、定数-1は2バイト文字定数であってはならない。また、外部ファイル参照には、2バイト文字が含まれてはならない。
   (MF)この制限は解除された。

5.14 2バイト文字機能単位におけるデータ部

5.14.1 JUSTIFIED（けたよせ）句

一般規則

1. 2バイト文字のデータ項目にも、JUSTIFIED句を適用できる。

5.14.2 PICTURE（形式）句

一般規則

PICTURE句で表せる項類が、2バイト文字と2バイト文字編集の2種類追加されている。この2つの項類は、共にUSAGE IS DISPLAY-1と記述しなければならない。
(MF)この2つの項類は、必ずしもUSAGE IS DISPLAY-1と記述しなくてもよい。
(COB370)DISPLAY-1 は PIC N項目には任意であるが、PIC G 項目では必ず必要である。

5.14.3 2バイト文字データに関する規則

1. PICTURE文字列に含めることのできる文字は、"G"
   (MF)(COB370)と "N" だけである。
2. 内容としては、2バイト文字集合中の任意の文字を含めることができる。

5.14.4 2バイト文字編集データに関する規則

PICTURE文字列には、記号"G"と"B" を任意に組み合わせて含めることができる。

5.14.4.1 使用する記号

使用する記号の働きは下記のとおり。

G - 各"G"が、2バイト文字または全角の空白だけを含むことができる文字位置を表わす。

B - 各"B"が、全角の空白を挿入する文字位置を表わす

(MF)(COB370)N - 各"N"が、2バイト文字または全角の空白だけを含むことができる文字位置を表わす

1つの"G" または "B"
(ANS85)または "N"
は、2バイト文字の1つの文字位置を表わすことに注意。

5.14.5 編集規則

データ項目にどのような編集をすることができるかは、そのデータ項目がどの項類に属するかによって決まる。「表 4-1 データの項類に適用できる編集の種類」(***言語リファレンス***の *プログラムの定義*の章を参照) に、下記の拡張を加える。

表 5-1 ： 2バイト文字の項類に適用できる編集

項　類	編集の種類
2バイト文字	なし
2バイト文字編集	単に"B" を挿入するだけ

5.14.5.1 固定挿入編集

1バイト文字データ項目中で使用した場合、"B"（空白）は半角の空白を表わす。　2バイト文字データ項目中で使用した場合は全角の空白を表わす。

5.14.6 REDEFINES（再定義）句

構文規則

1. データ名-1またはデータ名-2のどちらかの字類が2バイト文字である場合、両方の字類とも2バイト文字とする。
   (MF)この制限は解除された。

5.14.7 RENAMES（再命名）句

構文規則

1. データ名-1またはデータ名-2のどちらかの字類が2バイト文字である場合、両方の字類とも2バイト文字とする。THROUGH句は、使用できない。

5.14.8 USAGE（用途）句

一般形式

下記の書き方が追加されている。

```
[USAGE IS] DISPLAY-1
```

構文規則

1. DISPLAY-1項目のPICTURE文字列には、 "G" と "B" だけを含めることができる。PICTURE文字列に"G" を指定した項目には、USAGE IS DISPLAY-1句を記述しなければならない。したがって、USAGE IS DISPLAY-1句は、項目の字類を2バイト文字とする働きがあることに注意。
   (MF)(COB370)PICTURE文字列に
   (MF)"G" または

(MF)(COBOL370)"N" を指定した項目に、USAGE IS DISPLAY-1句を記述する必要はない。
(MF)PICTURE文字列に"G" が含まれていると、対応する項目の字類は2バイト文字とみなされる。

2. USAGE IS DISPLAY-1と記述した集団項目または基本項目に対して、BLANK WHEN ZERO句は使用できない。SYNCHRONIZED句は、無視される。
3. 画面節データ項目の用途は、暗黙的または明示的にUSAGE DISPLAY-1と定義してよい。

一般規則

1. USAGE IS DISPLAY-1句は、データの形式が2バイト文字であることを示す。

5.14.9 VALUE（値）句

構文規則

1. データ記述項において、項目の項類が2バイト文字である場合、VALUE句中の定数の項類は2バイト文字とする。2バイト文字定数を使用できるのは、項目の項類が2バイト文字または2バイト文字編集であるときだけである。
2. VALUE句中の2バイト文字定数の長さは、PICTURE文字列によって指定された大きさを超えてはならない。

5.14.10 条件名に関する規則

1. 条件名記述項において、該当するデータ項目の項類が2バイト文字である場合、対応するVALUE句中の定数の項類は2バイト文字とする。2バイト文字定数を使用できるのは、対応するデータ項目の項類が2バイト文字であるときだけである。

5.15 2バイト文字機能単位における手続き部

5.15.1 条件式

5.15.1.1 比較条件

字類が2バイト文字に属するデータ項目および定数を 比較条件中に使用し、任意の比較演算子を適用できる。各作用対象は、集団項目または2バイト文字とする。データの変換、編集、逆編集は行われない。2バイト文字の項目と2バイト文字編集の項目とは、区別されない。
行われる比較は、文字の比較である。2バイト文字集合中の文字の間には、一般に文字の大小順序は付けられない。したがって、2バイト文字を2進数のようにみなして、その文字を表わすビット・パターンの数値に基づいて文字の大小順序を決定する。
2バイト文字の中に1バイト文字に相当する文字が含まれている場合、2バイト文字の大小順序に従って文字を並べたときに、1バイト文字に相当する文字が1バイト文字の文字の大小順序どおりに並べられる保証はないので注意すること。

(MF)ある文字が2バイト文字集合にも1バイト文字集合にも含まれている場合、その2バイト文字と1バイト文字は等しいとはみなされない。（前述の*2バイト文字集合中のローマ字*の節を参照。）
PROGRAM COLLATING SEQUENCE句は、字類が2バイト文字または2バイト文字定数のデータ項目の比較に関しては効力がない。
作用対象の長さが等しくない場合、長い方と同じになるまで短い方の後ろに全角の空白を付け加えたものとして、比較が行われる。

5.15.1.2 字類条件

2バイト文字データ項目に対する字類検査は、2バイト文字および漢字である。
(MF)2バイト文字および漢字の字類検査はサポートされていない。

5.15.2 転記操作

字類が2バイト文字のデータ項目同士の間、またはデータ項目と定数の間でデータを転記することに係わるすべての文は、この章で後述するMOVE文の2バイト文字の転記に関する一般規則に従う。

5.15.3 ACCEPT（受取り）文

構文規則

1. 書き方2において、一意名は字類が2バイト文字のものであってはならない。

一般規則

1. 書き方1において、一意名の字類が2バイト文字である場合にだけ、2バイト文字データを入力できる。

5.15.4 CALL（呼ぶ）文

構文規則

1. 一意名-1は、字類が2バイト文字のデータ項目であってはならない。定数-1は、2バイト文字定数であってはならない。
   (MF)この制限は解除された。
2. 一意名-1または定数-1によって指定するプログラム名には、2バイト文字が含まれていてはならない。
   (MF)この制限は解除された。

一般規則

1. USING句の中で参照される字類が2バイト文字の各項目は、呼ぶプログラムの手続き部の見出しのUSING句の中の、字類が2バイト文字の項目と対応させること。

5.15.5 CANCEL（取消し）文

構文規則

1. 一意名-1は、字類が2バイト文字のデータ項目であってはならない。定数-1は、2バイト文字定数であってはならない。
   (MF)この制限は解除された。
2. 一意名-1または定数-1によって指定するプログラム名には、2バイト文字が含まれていてはならない。
   (MF)この制限は解除された。

5.15.6 INITIALIZE（初期化）文

一般形式

一般形式が拡張されて、ALPHABETICやALPHANUMERICなどのオプションの他に、DBCS,
(MF)NATIONAL, NATIONAL-EDITED
が加えられている。

一般規則

1. DBCS,
   (MF)NATIONAL, NATIONAL-EDITEDのいずれかのオプションを指定すると、
   項類が2バイト文字または2バイト文字編集のデータ項目が初期化される。

5.15.7 INSPECT（文字列検査）文

構文規則

1. 一意名-2を除く一意名および定数の中に、どれか1つでも字類が2バイト文字のものがあれば、それらはすべて字類が2バイト文字のものでなければならない。

一般規則

1. 一意名-2に記録される計数は、2バイト文字の文字数である。バイト数ではない。

5.15.8 MOVE（転記）文

構文規則

1. 一意名および定数の中に、どれか1つでも字類が2バイト文字のものがあれば、それらはすべて字類が2バイト文字のものでなければならない。
   (MF)字類が2バイト文字の項目、項類が英数字の項目、項類が英字の項目を自由に混在させてよい。
2. 書き方2のMOVE CORRESPONDINGにおいて、1組の項目のどちらか一方の字類が2バイト文字であるならば、両方とも字類が2バイト文字でなければならない。

一般規則

1. 送出し側項目の字類が2バイト文字であり、受取り側項目の項類が2バイト文字編集であると、受取り側項目上で編集が行われる。字類が2バイト文字の項目のこれ以外の組合わせでは、変換も編集も逆編集も行われない。
   (MF)この規則は、次のように緩められている。受取り側項目の項類が2バイト文字編集であれば、編集が行われる。字類が2バイト文字の項目が含まれるそれ以外の組合わせでは、変換も編集も逆編集も行われない。
2. 受取り側項目が送出し側項目よりも短い場合は、受取り側項目に収まりきらないデータ部分は切り捨てられる。受取り側項目が送出し側項目よりも長い場合は、受取り側項目の右側部分に全角の空白が埋められる。

5.15.9 SEARCH（表引き）文

構文規則7

1. 書き方2において、一意名-1の字類が2バイト文字であるならば、OCCURS句中に定義されているASCENDING/DESCENDING KEY項目の字類も、2バイト文字でなければならない。

5.15.10 STOP（停止）文

構文規則

1. 定数は、2バイト文字定数であってはならない。
   (MF)この制限は解除された。

5.15.11 STRING（連結）文

構文規則

1. 一意名-4を除く一意名および定数の中に、どれか1つでも字類が2バイト文字のものがあれば、それらはすべて字類が2バイト文字のものでなければならない。

一般規則

1. 一意名-4によって示される相対位置は、2バイト文字での位置である。

5.15.12 UNSTRING（分解）文

構文規則

1. 一意名-1、一意名-2、一意名-3、定数-1、定数-2の中に、どれか1つでも字類が2バイト文字のものがあれば、それらはすべて字類が2バイト文字のものでなければならない。

一般規則

1. 一意名-7によって示される相対位置は、2バイト文字での位置である。
2. 一意名-6中に記録される計数は、2バイト文字でのものである。

第6章: 2バイト文字支援機能の Micro Focus 拡張

(MF)

2バイト文字支援機能に関するMicro Focusの拡張機能は、16ビット・コーディング方式（2バイト文字集合）を使用している環境向けに、Micro Focus社が提供する追加のプログラミング機能である。この機能には、以前のMicro Focus製品で提供されていた日本語支援機能がすべて組み込まれている。

プログラムをマルチベンダー統合体系（MIA）標準に準拠させたいか、またはプログラムにIBM VS COBOL IIまたはIBM SAAとの互換性をもたせたい場合は、*2バイト文字支援機能*の章を参照のこと。

COBOLシステムに用いられている8ビット・コードで表わした文字を、 1バイト文字と呼ぶ。隣接した2バイトを使用した16ビット・コードで表わした文字を、2バイト文字と呼ぶ。
2バイト文字支援機能に関するMicro Focusの拡張機能を使用するには、 NCHAR指令、またはJAPANESE指令 を使用する。

2バイト文字支援機能に関するMicro Focusの拡張機能を使うと、*2バイト文字支援機能*の章 に定義した2バイト文字支援機能は、一部変更される。たとえば、1バイト文字のデータ項目から2バイト文字のデータ項目への転記操作に際して、1バイト文字から2バイト文字への変換が行われる。

字類NCHARとJAPANESE、および字類NCHAR-EDITEDとJAPANESE-EDITEDは同義語であり、どちらを使用してもよい。この章で、字類または項類NCHARまたは項類NCHAR-EDITEDと言うときは、それぞれ、字類または項類JAPANESEまたは項類JAPANESE-EDITEDと言うのと等しい。

6.1 NCHARデータ

NCHAR指令またはJAPANESE指令を設定すると、COBOLコンパイラは、項類がNCHARであるデータ、つまり2バイト文字が格納されているデータを認識するようになる。しかし、NCHAR指令またはJAPANESE指令は、項類が1バイト文字のデータの使用を妨げるわけではない。したがって、NCHAR指令またはJAPANESE指令を設定しても、1バイト文字のデータを使用することができる。
必要なハードウェアを備えていれば、入力や出力に使用されるNCHARデータ項目が認識されて、画面やキーボードやプリンタなどの装置にそれらのデータを適正に表示したり、データを受け取ったりすることができる。

6.2 原始プログラム

2バイト・コード文字は、定数、注記、注記項、利用者語に使用できる。これ以外については、NCHAR指令またはJAPANESE指令は、原始プログラム中で使用できる文字の範囲を変更しない。つまり、プログラムは依然としてCOBOLの文字集合(***言語リファレンス***の*COBOL言語の概念*の章を参照) を使用して書かれている。

6.3 言語の拡張

NCHARデータを含むデータ項目を定義するために、PICTURE句とUSAGE句が拡張されている。
NCHARデータの取扱いを定義するために、各種の指定や句や文に規則が追加されている。
特に断わりがないかぎり、COBOLのすべての機能と規則は、2バイト文字支援機能に関するMicro Focusの拡張機能を使用しているときでも適用される。以下に記述するのは、この拡張機能に関して追加されている規則と書き方だけである。

6.4 注記と注記項

注記および注記項の中では、1バイト文字と2バイト文字を自由に混ぜて書いてよい。

6.5 利用者語

下記の利用者語の中では、1バイト文字と2バイト文字を自由に混ぜて書いてよい。

符号系名、	通信記述名、	字類名、
条件名、	定数名、	データ名/一意名、
ファイル名、	指標名、	レベル番号、
登録集名、	呼び名、	実行用計算機名、
段落名、	プログラム名、	レコード名、
報告書名、	画面名、	節名、
区分番号、	翻訳用計算機名、	記号文字、
原文名		

上記の利用者語に関しては、構文規則が追加されたものと考えた方がよい。ある文字が1バイト文字集合と2バイト文字集合の両方に存在するときは、両方の文字が等しいとはみなされない。
オペレーティングシステムによっては、下記の利用者語には、ASCII文字しか使用できないものがある。

外部ファイル参照、	登録集名、	プログラム名

6.6 空白

字類がNCHARのデータ中の空白は、2バイト・コードで表わされる。 2バイト・コードで表わされた空白文字を、「全角の空白」と呼ぶ。

全角の空白に割り当てられる値は、 NCHAR指令、 JAPANESE指令、DBSPACE指令の影響を受ける。VALUE句をもたないすべてのデータ項目に共通するが、字類がNCHARのデータ項目の初期値は半角の空白となっている。

6.7 データ項目

6.7.1 NCHAR データ項目

言語リファレンスの*COBOL言語の概念*の章で説明した字類の他にNCHARを追加する。この字類には、項類としてNCHARとNCHAR-EDITEDの2つが含まれる。
字類がNCHARのデータ項目を表わすには、USAGE NCHAR句またはUSAGE JAPANESE句を使用する。この句を指定したデータ項目のPICTURE文字列には、文字"N", "B", "/" , "0" だけを書くことができる。
PICTURE文字列がすべて"N" であるデータ項目の項類は、NCHARである。PICTURE文字列中に"N","B", "/" , "0" が混在するデータ項目の項類は、NCHAR-EDITEDである。
PICTURE文字の"N", "B", "/" , "0" は、どちらも1つの2バイト文字の文字位置を表わす。NCHARデータ項目の長さは、そのPICTURE文字列中の "N", "B","/","0" の数で表わす。ただし、例外として別途規定している場合がある。
部分参照では、最左端文字位置と長さは、バイト数ではなく2バイト文字の数を指定する。
字類が英数字のデータ項目を使用できる箇所であれば、どこでも字類がNCHARのデータ項目を使用してよい。ただし、この章で後述する該当する規則に従うものとする。また、これには例外もある。

6.7.2 混合データ項目

項類が英数字のデータ項目中に収めたデータの中に、2バイト文字を含めることができる。このようなデータにおいては、1バイト文字は1バイト・コードで表わされ、2バイト文字は2バイト・コードで表わされる。空白は半角で表わされる。
プログラム内でデータを操作する際には、2バイト文字も通常の英数字と同様に扱われる。したがって、2バイト文字の2つのバイトが分割されないようにすることは、プログラマの責任である。
この型のデータ項目の長さは、計算機の記憶領域に格納したときは、すべての場合において、バイト単位で数える。

6.8 定数

6.8.1 NCHAR 定数

言語リファレンスの*COBOL言語の概念*の章で説明した文字定数と数字定数の他に、3番目の定数としてNCHAR定数が加えられている。
NCHAR定数は、両端を引用符またはアポストロフィで囲んだ文字列である。この文字列には、

計算機の2バイト文字集合中の任意の文字を含めることができる。

6.8.2 NCHAR 定数の項類

文字定数を使用できる箇所ならばどこでも、すべての2バイト文字定数を使用できる。ただし、この章で後述する該当する規則に従うものとする。また、これには例外もある。

6.8.3 混合定数

文字定数の中に、2バイト文字を含めることができる。1バイト文字と2バイト文字が両方含まれる文字定数を「混合定数」と呼ぶ。このような定数においては、1バイト文字は1バイト・コードで表わされ、2バイト文字は2バイト・コードで表わされる。空白は半角で表わされる。
出力のとき、1バイト文字も2バイト文字も認識される。プログラム内でデータを操作する際には、2バイト文字定数も通常の文字定数と同様に扱われる。したがって、2バイト文字の2つのバイトが分割されないようにすることは、プログラマの責任である。
混合定数の項類は英数字であり、NCHARではない。
文字列の区切り文字として使用する引用符またはアポストロフィを混合定数中に含める場合は、その引用符を2つ続けて書かなければならない。区切り文字として使用しない文字を含める場合は、1つ書くだけでよい。実行用プログラム内での混合定数の値は、文字列そのものである。ただし、中に出てくる2つの連続した区切り文字は1つの文字として扱われる。

6.8.4 表意定数

（この章で記述する字類および項類に関する該当する規則に従って、）NCHAR字類定数だけを使用できる箇所で表意定数を使用した場合、これはNCHAR定数となる。

6.8.5 表意定数の値

		NCHAR値の例	
定　数	内　容	シフトJIS	EUC
ZERO ZEROS ZEROES	文脈に応じて、1つ以上の2バイト文字の"0" を表わす。	x"824F"	x"A3B0"
SPACE SPACES	計算機の文字集合の中の1つ以上の全角の空白を表わす。	x"8140"[1]	x"A1A1"[1]
HIGH-VALUE HIGH-VALUES	プログラムの文字の大小順序の中で順序番号が最大の文字を1つ以上表わす。	x"FFFF"	x"FFFF"
LOW-VALUE LOW-VALUES	プログラムの文字の大小順序の中で順序番号が最小の文字を1つ以上表わす。	x"0000"	x"0000"
QUOTE QUOTES	1つ以上の2バイト文字の「"」を表わす。	x"818D"[2]	x"A1ED"[2]

[1] この値はDBSPACE指令の影響を受ける。

[2] この値はAPOST指令の影響を受ける。

6.9 環境部

6.9.1 実行用計算機段落

一般規則

コンパイラは、PROGRAM COLLATING SEQUENCE指定の中では、8ビット文字だけが使用できるように設計されている。

ALPHABET句を使用して、2バイト文字定数が含まれる文字の大小順序を定義することは無意味である。これは、2バイト文字が2つの別々の1バイト文字として扱われてしまうためである。このような形で2バイト文字を使用すると、2バイト文字は1バイト文字として並べられ、2バイト文字の意味は無視される。

文字の大小順序を定義するときに、ALPHABET句内にASCII文字以外の文字を含めることができる。たとえば、日本語環境において、シフトJISコードの半角の片仮名が使用できる。半角の片仮名は、1バイトの文字として格納され表示される。

EUCを使用している場合は、ALPHABET句内に半角の片仮名を使用することはできない。これらはシフト・コードと組み合わせて2バイトで使用される。

6.9.2 特殊名段落

構文規則

1. CURRENCY SIGN句において、定数-6は、NCHAR定数であっても、"N" であってもならない。
2. ALPHABET句において、定数-1、定数-2、定数-3は、NCHAR定数であってはならない。
3. CLASS句において、定数-4および定数-5は、NCHAR定数であってはならない。

6.9.3 ファイル管理段落

構文規則

1. ASSIGN句において、定数-1はNCHAR定数でもよい。外部ファイル参照に、2バイト文字が含まれてもよい。

6.10 データ部

6.10.1 JUSTIFIED（けたよせ）句

一般規則

1. NCHARデータ項目にも、JUSTIFIED句を適用できる。

6.10.2 PICTURE（形式）句

一般規則

1. PICTURE句で表わせる項類が、NCHARとNCHAR-EDITEDの2種類追加されている。この2つの項類は、共にUSAGE IS NCHARまたはUSAGE IS JAPANESEと記述できる。

6.10.3 NCHAR データに関する規則

1. PICTURE文字列に含めることのできる文字は、"N" だけである。
2. 内容としては、2バイト文字集合中の任意の文字を含めることができる。

6.10.4 NCHAR-EDITED データに関する規則

1. PICTURE文字列には、記号"N","B", "/", "0" を任意に組み合わせて含めることができる。

6.10.4.1 使用する記号

使用する記号の働きは下記のとおり。

		NCHAR値の例	
記　号	内　容	シフトJIS	EUC
N	各"N"は、2バイト文字または全角の空白だけを含むことのできる文字位置を表わす。		
B	各"B"は、全角の空白を挿入する文字位置を表わす。	x"8140"[1]	x"A1A1"[1]
/	各"/"は、全角のスラッシュを挿入する文字位置を表わす。	x"851E"	x"A1BF"
0	各"0"は、 全角のゼロを挿入する文字位置を表わす。	x"824F"	x"A3B0"

[1] この値はDBSPACE指令の影響を受ける。

"N", "B", "/", "0" は、それぞれ2バイト文字の1つの文字位置を表わすので注意すること。

6.10.5 編集規則

データ項目にどのような編集をすることができるかは、そのデータ項目がどの項類に属するかによって決まる。表 4-1 データの項類に適用できる編集の種類(***言語リファレンス***の *プログラムの定義*の章を参照) に下記の拡張が加えられている。

表 6-1: データの項類と施せる編集

項　類	施せる編集
NCHAR	なし
NCHAR-EDITED	単に"B", "/", "0" を挿入するだけ

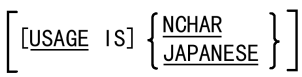
6.10.6 固定挿入編集

"B" (空白) は、1バイト文字データ項目中で使用した場合、半角の空白を表わす。NCHARデータ項目中で使用した場合、全角の空白を表わす。
"/" (スラッシュ)は、 1バイト文字データ項目中で使用した場合、半角のスラッシュを表わす。NCHARデータ項目中で使用した場合、全角のスラッシュを表わす。
"0" (ゼロ) は、1バイト文字データ項目中で使用した場合、半角のゼロを表わす。NCHARデータ項目中で使用した場合、全角のゼロを表わす。

6.10.7 USAGE（用途）句

一般形式

下記の書き方が追加されている。



構文規則

1. PICTURE文字列には、"N", "B","/", "0" だけを含めることができる。
   PICTURE文字列に"N" が含まれていると、対応する項目の字類は、NCHARまたは日本語とみなされる。
2. USAGE IS NCHARまたはUSAGE IS JAPANESEと記述した集団項目または基本項目に対しては、BLANK WHEN ZERE句を適用できない。SYNCHRONIZED句は無視される。

一般規則

1. USAGE IS NCHARまたはUSAGE IS JAPANESE句は、データの形式がNCHARであることを示す。

6.10.8 VALUE（値）句

構文規則

1. データ記述項において、項目の項類がNCHARであるならば、VALUE句中の定数の項類はNCHARでなければならない。NCHAR定数を使用できるのは、項目の項類がNCHARまたはNCHAR-EDITEDであるときだけである。
   VALUE句中のNCHAR定数の長さは、PICTURE文字列によって指定された大きさを超えてはならない。

6.11 手続き部

6.11.1 条件式

6.11.1.1 条件名

字類がNCHARの定数を含む条件名が、字類がNCHARでない基本項目と共に使用されると、定数は1バイト文字定数とみなされる。

6.11.1.2 比較条件

字類がNCHARに属するデータ項目および定数を比較条件中に使用し、任意の比較演算子を適用できる。データの変換、編集、逆編集は行われない。項類NCHARの項目と項類NCHAR-EDITEDの項目は、区別されない。

行われる比較は、文字の比較である。2バイト文字集合中の文字の間には、一般に文字の大小順序は付けられない。したがって、2バイト文字を2進数のようにみなして、その文字を表わすビット・パターンの数値に基づいて、文字の大小順序を決定する。

2バイト文字の中に1バイト文字に相当する文字が含まれている場合、2バイト文字の大小順序に従って文字を並べたときに、1バイト文字に相当する文字が1バイト文字の文字の大小順序どおりに並べられる保証はないので注意すること。

ある文字が2バイト文字集合にも1バイト文字集合にも含まれている場合、その2バイト文字と1バイト文字は等しいとはみなされない。

PROGRAM COLLATING SEQUENCE句は、字類がNCHARまたはNCHAR定数のデータ項目の比較に関しては効力がない。

作用対象の長さが等しくない場合、長い方と同じになるまで短い方の後ろに全角の空白を付け加えたものとして、比較が行われる。

6.11.1.3 字類条件

字類検査にJAPANESEが追加される。この字類検査は、検査対象のデータ項目中のすべての文字が有効な半角の片仮名か2バイト文字であるか、すべてが空白のときに、真となる。

6.11.2 ACCEPT（受取り）文

一般規則

1. 書き方1において、1バイト文字と2バイト文字の両方のデータを入力できる。1バイト文字データは、2バイト文字データに変換されない。
   書き方1でNCHARデータ項目を受け取る場合、入力されたデータの妥当性検査も変換も文字の桁寄せも行われない。この結果、データが壊れることもあり得る。したがって、この方法は使用しない方がよい。使用する場合は、十分に注意すること。
   書き方4と5では、NCHARデータ項目に入力する文字として、1バイト文字も2バイト文字も有効である。ただし、1バイト文字は、対応する2バイト文字に変換される。
   　NCHARデータ項目を受け取る際に、通常の編集機能がすべて文字単位でサポートされている（バックスペース、打ち直し、削除、リストア、重ね打ち、挿入）。

注: 日本語EUCなどの環境では、それに固有の動作が起こる場合がある。

6.11.3 INITIALIZE（初期化）文

一般形式

一般形式が拡張されて、ALPHABETICやALPHANUMERICなどのオプションの他に、NCHARおよびJAPANESEが加えられている。

構文規則

1. NCHARとJAPANESEは、一緒に指定できない。DBCSまたはNATIONALと一緒にNCHARまたはJAPANESEは指定できない。

一般規則

1. NCHARまたはJAPANESEを指定すると、項類がNCHARのデータ項目が初期化される。

6.11.4 INSPECT（文字列検査）文

一般規則

1. 一意名-1の字類がNCHARであるならば、一意名-2に記録される計数は2バイト文字の文字数であって、バイト数ではない。

構文規則

1. 一意名-2を除くすべての一意名と定数は、そのうちのどれかが字類NCHARの場合、すべて字類NCHARであること。

6.11.5 MOVE（転記）文

字類がNCHARのデータ項目どうしの間、またはデータ項目と定数の間でデータを転記することに係わるすべての文は、下記の2バイト文字の転記に関する一般規則に従う。

受取り側項目の項類	送り出し側項目の項類			
	英数字	NCHAR	NCHAR-EDITED	桁寄せ付きNCHAR
英数字	注を参照	可/G1	可/G4	不可/S2
NCHAR	可/G2	可/G3	可/G4	可/G5
NCHAR-EDITED	不可/S1	不可/S1	不可/S1	不可/S1
桁寄せ付きNCHAR	可/G2	可/G3	可/G4	可/G5

注: この表に字類が英数字のものも含めてあるのは、1バイト文字データ項目の用途を示すためである。 1バイト文字データ項目どうしの転記処理の詳細については、***言語リファレンス***の*プログラムの定義*の章を参照。 1バイト文字と2バイト文字が混在するデータ項目の転記処理については、上の表と下記の説明を参照。

構文規則

1. 送出し側のデータ項目の字類は、NCHAR-EDITEDであってはならない。
2. 受取り側のデータ項目の字類がNCHARであり、JUSTIFIED句が指定されている場合、送出し側のデータ項目の字類はNCHARとする。

一般規則

1. 送出し側データ項目の字類が英数字または英字であり、受取り側データ項目の字類がNCHARであると、送出し側データ項目中の1バイト文字は、対応する2バイト文字に変換されて受取り側データ項目に転記される。送出し側データ項目中の2バイト文字は、そのまま受取り側データ項目に転記される。
   受取り側データ項目が送出し側データ項目よりも短い場合は、受取り側データ項目に収まりきらないデータ部分は切り捨てられる。受取り側データ項目が送出し側データ項目よりも長い場合は、受取り側データ項目の右側部分に全角の空白が埋められる。
2. 送出し側データ項目の字類がNCHARであり、受取り側データ項目の字類が英数字であると、送出し側データ項目中の16進値が受取り側データ項目にバイト単位でそのまま転記される。
   受取り側データ項目が送出し側データ項目よりも短い場合は、受取り側データ項目に収まりきらないデータ部分は切り捨てられる。
3. 受取り側データ項目が送出し側データ項目よりも長い場合は、受取り側データ項目の右側部分に半角の空白が埋められる。
   送出し側データ項目の字類がNCHARであり受取り側データ項目の字類もNCHARであると、送出し側データ項目中の値が受取り側データ項目にバイト単位でそのまま転記される。
4. 送出し側データ項目の字類が英数字か英字かNCHARであり、受取り側データ項目の項類NCHAR-EDITEDであると、受取り側データ項目上で編集が行われる。
   受取り側データ項目が送出し側データ項目よりも短い場合は、受取り側データ項目に収まりきらないデータ部分は切り捨てられる。受取り側データ項目が送出し側データ項目よりも長い場合は、受取り側データ項目の右側部分に全角の空白が埋められる。
   送出し側データ項目の字類が英数字か英字であると、送出し側データ項目の1バイト文字16進値は対応する2バイト文字に変換されて、受取り側データ項目に転記される。
   送出し側データ項目中の2バイト文字は、そのまま受取り側データ項目に転記される。
5. 送出し側データ項目の字類がNCHARであり、受取り側データ項目の字類がJUSTIFIED句付のNCHARであると、送出し側データ項目中の値が受取り側データ項目にバイト単位でそのまま転記される。

受取り側データ項目にJUSTIFIED句が指定されており、送出し側データ項目の方が受取り側データ項目よりも長い場合は、受取り側データ項目の左側の収まりきらない部分が切り捨てられる。受取り側データ項目にJUSTIFIED句が指定されており、受取り側データ項目が送出し側データ項目よりも長い場合は、データは右側に桁寄せして転記され、受取り側データ項目の左側部分に全角の空白が埋められる。

6.11.6 SEARCH（表引き）文

一般規則

1. 書き方1において、一意名-1の字類がNCHARである場合、一意名-2に記録される値は2バイト文字で構成される。

6.11.7 STRING（連結）文

一般規則

1. 一意名-3の字類がNCHARである場合、一意名-4によって示される相対位置は2バイト文字での位置である。

構文規則

1. 一意名-4を除くすべての一意名と定数は、そのうちのどれかが字類NCHARの場合、すべて字類NCHARであること。

6.11.8 UNSTRING（分解）文

一般規則

1. 一意名-1の字類がNCHARであるならば、一意名-7によって示される相対位置は2バイト文字での位置であり、一意名-6中に記録される計数は2バイト文字でのものである。

構文規則

1. 一意名-1，一意名-2，一意名-3，定数-1，定数-2は、そのうちのどれかが字類NCHARの場合、すべて字類NCHARであること。

第7章：例

この章では、このマニュアルおよび『言語リファレンス』の随所で詳細に説明した原始コードの句および文のいくつかについて、例を示して注解する。この章で取り上げる事項の構文および一般規則の詳細については、参照マニュアル中の該当する説明を参照。
この章で提示するサンプル・コードは完全なCOBOLプログラムではなく、具体的なコーディングの課題を例解するのに十分な程度の部分的なものである。完全なCOBOL原始プログラムの例が多数、COBOLシステム中にコンピュータで読める形式で収められている。ディスク上のドキュメント*"Products Contents Checklist"*の中から"sample"という言葉を探すと、それらのサンプル・プログラムを見ることができる。

注意： この章では、ボックスやバブルのような方言を示す目印を表示することはしない。この章で使用するCOBOLオプションに存在する可能性のある構文および意味の違いに関する情報については、リファレンス・マニュアルの関連する説明事項およびCOBOLシステム・ドキュメンテーションを参照。

この章に示す例は、主なテーマとする句や指定や文のアルファベット順に配列してある。もちろん、ほとんどの例の中に、主なテーマ以外のCOBOLの構文も出てくることがある。ここに示す例は部やモジュール別に分割してはいない。

7.1 呼出しプロトタイプ

下のプログラムはCALLプロトタイプの使い方を例示する。CALLプロトタイプを下記のように定義したとする。

```
identification division.
program-id.  callsub is external.
environment division.
configuration section.
special-names.
    call-convention 3 is some-language.
data division.
linkage section.
01  x1      pic 9(4) comp-5.
01  x2      pic xx.
01  x3      pic 9(8).
01  x7      pic x.
procedure division some-language using by value      x1
                                       by reference x2
                                       by reference x3.
entry "callsub2" using x2 delimited
                       any
                       x1.
entry "printf" using x7 delimited
                     any repeated.
end program callsub.
```

上のCALLプロトタイプと同じ原始ファイル中に、下記の「実際の」原始コード"real"があるとする。

```
identification division.
program-id.  prog-1.
data division.
working-storage section.
01  x1      pic 9(4) comp-5.
01  x2.
    05      pic 9(4) comp-5.
    05      pic x(20).
01  x3      pic 9(8).
01  x4      pic 9(9) comp-5.
01  x5      pic x.
01  x6      pic x(20).
procedure division.
mainline.
    call "callsub" using x1 x2 x3
```

上のCALL文は下記に相当する。

```
                          by value x1
                          by reference x2
                          by reference x3
```

いろいろなCALL文とその結果を以下に例示する。

例1

```
    call "callsub" using x1 x2
```

上のCALL文はエラーとなる。パラメータの数が合わないからである。

例2

```
    call other-language "callsub" using x1 x2 x3
```

上のCALL文はエラーとなる。呼出し方式が正しくないからである。

例3

```
    call "callsub" using by reference x1 x2 x3
```

上のCALL文はエラーとなる。x1は値で渡されなければならないからである。

例4

```
    call "callsub" using 99 x2 x3
```

上のCALL文は下記のように使用したのに等しい。

```
                by value 99 size 2
                by reference x2
                by reference x3
```

例5

```
        call "callsub" using x4 x2 x3
```

上のCALL文はエラーとなる。x4の長さが合わないからである。

例6

```
        call "callsub" using x1 x5 x3
```

上のCALL文はエラーとなる。x5が小さすぎるからである。

例7

```
        call "printf" using "A long %l\n" x4
```

上のCALL文において、x4はANY REPEATEDでカバーされるパラメータである。

例8

```
        call "callsub2" using "Hello" x2 x1
```

上のCALL文は下記に等しい。

```
        move "Hello" & x"00" to temp
        call "callsub2" using temp x2 x1
```

例9

```
        call "callsub2" using x6 x2 x1
```

上のCALL文は下記に等しい。

```
        move x6 to temp
        move x"00" to temp (21:1)
        call "callsub2" using temp x2 x1
```

例10

```
        call "callsub2" using x6 x2 x1 x4
```

上のCALL文はエラーとなる。渡されるパラメータが多すぎるからである。

7.1.1 CALLプロトタイプの使用例

COBOLのアプリケーション・プログラマがCOBOLプログラムの中からCの関数を呼び出したい場合、以下のことを行う必要がある。

- COBOLのアプリケーション・プログラムの中でCの関数のパラメータを定義する必要がある。このことは、Cのデータ型とCOBOLのデータ型とを対応させることを意味する。
- Cの関数を実際に呼び出すさいには、パラメータの数と各パラメータの型を合致させなければならない。
- Cの関数に渡す文字列は終端を空文字で区切ったCの文字列に変換する必要がある。

上記の過程を自動化するために、COBOL TYPEDEFSおよびCOBOL CALLプロトタイプを使用することができる。そうすると、終端を空文字で区切ったCの文字列に文字列を変換する処理も自動的に行われる。上記のすべてを行う方法を示す例を下に示す。

呼び出したいCの関数を**my_C_function**とする。この関数のCのコードの一部を下に示す。

```
sample.c
------------------------------------------------------------------
/*** start of source module sample.c ***/
/*------------------------*/
/*  Include Header Files  */
/*------------------------*/
#include <stdio.h>
#include "sample.h"
/*-------------------*/
/*  Sample Function  */
/*-------------------*/
int my_C_function (parm_1, parm_2, parm_3)
num_type parm_1;
unsigned char *parm_2;
complex_type *parm_3;
{
    int rtn_code = 0;
    printf("    my-C_function: invoked\n");
    printf("    my-C_function: parm_1 = %d\n", parm_1);
    if (parm_2 == NULL) {
        printf("    my_C_function: parm_2 = IS NULL\n", parm_2);
        rtn_code = -1;
    } else {
        printf("    my_C_function: parm_2 = %s\n", parm_2);
    }
    if (parm_3 == NULL ) {
        printf("    my_C_function: parm_3 = IS NULL\n", parm_3);
        rtn_code = -1;
    } else {
        printf("    my_C_function: parm_3\n");
        printf("                   (num1) = %d\n", parm_3->num1);
        printf("                   (num2) = %d\n", parm_3->num2);
    }
    printf("    my_C_function: completed\n");
    return(rtn_code);
}
/*** end of source module sample.c ***/
------------------------------------------------------------------
```

この例では、Ｃの関数に下記の３つのパラメータが使用されている。

- 型定義された値
- Ｃの文字列
- 構造化されたデータ型

Ｃの型定義と関数プロトタイプが収められたヘッダー・ファイルがある。それは下記のようになっている。

```
sample.h
-----------------------------------------------------------------
/*** start of source module sample.h ***/
#ifndef     SAMPLE
#define     SAMPLE
/*------------*/
/*  Typedefs  */
/*------------*/
typedef int num_type;
typedef struct {
        int num1;
        long num2;
} complex_type;
/*----------------------*/
/*  Function Prototype  */
/*----------------------*/
extern int my_C_function (
        num_type parm_1,
        unsigned char *parm_2,
        complex_type *parm_3
);
#endif      /* SAMPLE */
/*** end of source module sample.h ***/
-----------------------------------------------------------------
```

最初のステップは、Cの型定義と関数プロトタイプをCOBOLのTYPEDEFSとCALLプロトタイプに変換することである。それは、Micro Focus COBOLに付随して提供されている**h2cpy**ユーティリティを使用して行うことができる。

```
h2cpy sample.h
```

は下記のコピーブックを出力する。

```
sample.cpy
-----------------------------------------------------------------
 program-id. "c_typedefs" is external.
 77  char                  pic s9(2)  comp-5 is typedef.
 77  uns-char              pic  9(2)  comp-5 is typedef.
 77  short                 pic s9(4)  comp-5 is typedef.
 77  uns-short             pic  9(4)  comp-5 is typedef.
 77  int                   pic s9(9)  comp-5 is typedef.
 77  uns-int               pic  9(9)  comp-5 is typedef.
 77  long                  pic s9(9)  comp-5 is typedef.
 77  uns-long              pic  9(9)  comp-5 is typedef.
 77  d-l-float                        comp-2 is typedef.
 77  d-float                          comp-2 is typedef.
 77  float                            comp-1 is typedef.
 77  proc-pointer          procedure-pointer is typedef.
 77  data-pointer                     pointer is typedef.
 77  void                  pic  9(2)  comp-5 is typedef.
 01  num-type          is typedef       usage int.
 01 complex-type       is typedef.
     02 num1             usage int.
     02 num2             usage long.
 entry "my_C_function" using
         by value      int
         by reference  uns-char
         by reference  complex-type
     returning         int
     .
 end program "c-typedefs".
-----------------------------------------------------------------
```

上記の中に下記のものが含まれている。

- Ｃの基本的なデータ型をCOBOL TYPEDEFSに対応づける、レベル77のデータ項目
- Ｃのヘッダー・ファイル中のtypedefに合致するTYPEDEFSである、レベル01のデータ項目
- Ｃの関数用のCOBOLのCALLプロトタイプの働きをする、entry文

テキスト・エディタを用いて、このファイルに下記の変更を加える。

- 必要のない77レベルのデータ項目を削除する
- uns-charの対応づけを、数字フィールドではなく、"pic x"に対応するように変更する。この変更を行わないと、 PIC Xのフィールドをパラメータとして渡そうとすると、プロトタイプのパラメータ仕様に合わないとして、エラーとされる。
- CALLプロトタイプ中のuns-charの横に必要語のuns-charを追加する。これによって、実行時に呼出し側のために、パラメータとして渡されるTEXT文字列が終端を空文字で区切ったＣの文字列に変換されるようになる。

上記の編集を行った結果を下に示す。

sample.cpy

```
------------------------------------------------------------------
 program-id. "c_typedefs" is external.

 77  uns-char              pic x            is typedef.
 77  int                   pic s9(9)  comp-5 is typedef.
 77  long                  pic s9(9)  comp-5 is typedef.
 77  data-pointer                    pointer is typedef.

 01 num-type           is typedef       usage int.
 01 complex-type       is typedef.
     02 num1             usage int.
     02 num2             usage long.

 entry "my_C_function" using
         by value      int
         by reference  uns-char delimited
         by reference  complex-type
     returning         int
     .

 end program "c_typedefs".
------------------------------------------------------------------
```

my_C_function 関数を呼び出すCOBOLのアプリケーション・プログラムの例を下に示す。

```
------------------------------------------------------------------
copy 'sample.cpy'.

identification division.
program-id.  prog.
working-storage section.
01  ws-parm-1                        usage num-type.
01  ws-parm-2                        pic x(50)
                   value "This is a PIC X string from COBOL".
01  ws-parm-3                        usage complex-type.
01  ws-return-code                   usage int.

procedure division.
main-code section.
    display "prog: started"

    move 123     to ws-parm-1
    move 1       to num1 IN ws-parm-3
    move 2       to num2 IN ws-parm -3

    display " "
    display "prog: call 'my_C_function' with ALL parameters"
    call "my_C_function" using ws-parm-1
                               ws-parm-2
                               ws-parm-3
                         returning ws-return-code
    end-call
    display "prog: 'my_C_function' return code = "
            ws-return-code

    display " "
    display "prog: call 'my_C_function' with NULL parameters"
    call "my_C_function" using 0
                               OMITTED
                               OMITTED
                         returning ws-return-code
    end-call
    display "prog: 'my_C_function' return code = "
            ws-return-code

    display " "
    display "prog: completed"
    exit program
    stop run.
------------------------------------------------------------------
```

上記の例において、下記のコーディングがなされている。

- 原始プログラム中の最初の文として、コピーファイル **sample.cpy**が組み込まれている。このファイルはCOBOLのTYPEDEFSおよびCALLプロトタイプが収められているコピーブックである。

  型定義とプロトタイプは完全な「外部」プログラムとして定義されている。それらは実際の原始プログラムの前に置かれている。その形は複数のプログラムが収められた原始ファイルに似ている。
- Ｃの関数用のパラメータは、COBOLのTYPEDEFSを使用して、作業場所節中に定義されている。
- この例では、Ｃの関数の呼出しが２回行われている。このことから、下記の点に注意すべきである。

- BY REFERENCE/VALUEは指定されていない。プロトタイプから正しい省略時解釈の値が引き出される。
- パラメータの数と型が正しく指定されていることが、プロトタイプによって保証される。
- ２番目の呼出しでは、OMITTEDという語が特別に指定されている。それは、BY REFERENCEパラメータに、パラメータ値の代わりに、NULLを渡すためである。このようにする必要がある理由は、単にBY VALUE 0とすることはできないからである。それは無効とされる。なぜならば、BY REFERENCEはそのパラメータとして必須であるからである。OMITTEDと指定することによって、Ｃの関数にパラメータへのポインタが渡される代わりに、NULLが渡されるようになる。

上の例を実行した結果の出力を下に示す。

```
----------------------------------------------------------------
%prog
prog: started
prog: call 'my_C_function' with ALL parameters
      my_C_function: invoked
      my_C_function: parm_1 = 123
      my_C_function: parm_2 = This is a PIC X string from COBOL
      my_C_function: parm_3
                     (num1) = 1
                     (num2) = 2
      my_C_function: completed
prog: 'my_C_function' return code = +0000000000
prog: call 'my_C_function' with NULL parameters
      my_C_function: invoked
      my_C_function: parm_1 = 0
      my_C_function: parm_2 = IS NULL
      my_C_function: parm_3 = IS NULL
      my_C_function: completed
prog: 'my_C_function' return code = -0000000001
prog: completed
%
----------------------------------------------------------------
```

7.2 手続きポインタの呼出しと設定

```
* Calling program:
 program-id. startup.
 working-storage section.
 01 start-point  usage procedure-pointer.
 procedure-division.
     set start-point to entry "menu"
     call "controller" using start-point
     display "End of run"
     stop run.
 entry "illegal"
* Recursive calls invalid without local-storage section.
     stop run.
 end program startup.
* Called program:
 program-id. controller.
 working-storage section.
 01 next-option  pic x.
 linkage section.
 01 current-proc usage procedure-pointer.
 procedure division using current-proc.
     perform until current-proc = NULL
         call current-proc returning next-option
*        Note program-id must be called before any entry point
         evaluate next-option
          when "a"    set current-proc to entry "sub1"
          when other  set current-proc to NULL
         end evaluate
     end-perform
     exit program.
 end program controller.
 program-id. menu.
 working-storage section.
 01 exit-option  pic x.
 procedure division.
     display "In menu"
     move "a" to exit-option
     exit program returning exit-option.
*    Note that the maximum size of returned value is 4 bytes
 entry "sub1"
     display "In sub1"
     exit program returning 1.
```

7.3 動的に割り当てられたデータ領域をサブプログラムから引き取る呼出し

```
* Calling program:
 program-id. calling.
 working-storage section.
 01 call-size   pic x(4) comp-5 value 64.
 linkage section.
 01 call-area   pic x.
 procedure division.
     call "sub2" using call-size
                 returning address of call-area
     if address of call-area not = null
         display "Contents of new area: " call-area(1:call-size)
     end-if
     stop run.
end program calling.
* Called program:
 program-id. sub2.
 working-storage section.
 01 sub-pointer usage pointer.
 linkage section.
 01 link-size   pic x(4) comp-5.
 01 link-area   pic x.
 procedure division using link-size.
     call "CBL_ALLOC_MEM" using sub-pointer
                                by value link-size
                                         0 size is 4
     if return-code = 0
         set address of link-area to sub-pointer
         move "Hello!" to link-area(1:call-size)
     else         set sub-pointer to null
     end-if
     exit program returning sub-pointer.
```

7.4 COPY (ANSI'68またはLANGLVL(1)の変形)

OLDCOPYコンパイラ指令を設定すると、COPY文の動作が少し変更される。具体的には、ANSI'74およびANSI'85の標準に定義されているようにではなく、ANSI'68の標準に定義されているように動作するようになる。この変更された動作は、IBMのメインフレーム上でLANGLVL(1)コンパイラ・オプションを使用したときの、OS/VS COBOLおよびDOS/VS COBOLの動作とも一貫性がある。

OLDCOPYコンパイラ指令を設定し、かつ01レベルのデータ記述全体をコピー・メンバーに含めたい場合は、COPY文とコピーされるデータ記述の両方を01レベルのデータ項目として定義すべきである。ただし、COBOLプログラムの残りの部分では、コピー文から得られたデータ名しか利用できない。例を下に示す。

原始ファイルのコード

```
    01   PRODUCT-CODE COPY COPYPROD.
```

コピーファイル"COPYPROD"のコード

```
    01   PROD-CD.
         05   ITEM-NAME      PIC X(30).
         05   ITEM-NUMBER    PIC X(5).
```

結果のCOBOLコード

```
01   PRODUCT-CODE.
     05   ITEM-NAME      PIC X(30).
     05   ITEM-NUMBER    PIC X(5).
```

7.5 COPY（語の部分的な置換）

ANSI'85に準拠するコンパイラ中のCOPY文を使用すると、コピー・メンバーの原文中の語を部分的に変更することができる。ただし、この構文が効力を発揮するのは、ある種の方式（および特殊な文字）を使用した場合だけであるので、十分に注意を要する。この技法を応用するときには、プログラマは変更可能な節をコピー・メンバーにあらかじめ設定しておかなければならない。実際、いったんこの技法を使用すると、置換が行われなかったときには、コピー・メンバーは正しくコンピュータされなくなる。例を下に示す。

原始ファイルのコード

```
copy Payroll
     replacing ==(TAG)== by ==Payroll==.
```

コピー・ファイルのコード

```
01  (TAG).
    05 (TAG)-Week      pic s99.
    05 (TAG)-Gross-Pay pic s9(5)v99.
    05 (TAG)-Hours     pic s9(3)
                          occurs 1 to 52 times
                          depending on (TAG)-Week of (TAG).
```

置換しながらコピーした結果は下記のようになる。

```
01  Payroll.
    05 Payroll-Week      pic s99.
    05 Payroll-Gross-Pay pic s9(5)v99.
    05 Payroll-Hours     pic s9(3)
                          occurs 1 to 52 times
                          depending on Payroll-Week of Payroll.
```

7.6 特殊名段落のCRT状態句

特殊名段落のCRT状態句は下記の構成のデータ項目を提示する。

- 状態キー１ - ACCEPT操作の終了条件を示す。
- 状態キー２ - ACCEPT操作の終了条件をさらに細かく区別する。
- 状態キー３ - ACCEPT操作を終了させたキーの生のキーボード・コードを保持する。

CRT状態キーをコーディングおよび参照する方法を示す例を下に示す。

```
*************************************************************
*                                                           *
*   The following shows how the special-names paragraph     *
*   sets up both a cursor position field and a CRT status   *
*   key field.                                              *
*                                                           *
*************************************************************
 special names.
     cursor is cursor-position
     crt status is crt-status.
...
```

```
 working-storage section.
 01 cursor-position                  pic 9(4).

*************************************************************
*   The following group item defines the CRT status key     *
*   field and establishes certain 78-level condition-names  *
*   associated with key fields.                             *
*                                                           *
*   Programs using these definitions should be compiled     *
*   with NOIBMCOMP and MF to function as expected.          *
*                                                           *
*************************************************************

 01  crt-status.
     05 crt-status-1                pic 9.
        88  terminate-key                value 0.
        88  function-key                 value 1.
        88  adis-key                     value 2.
        88  status-1-error               value 9.
    05  crt-status-2                 pic 99 comp-x.
        88  esc-key                      value 0.
        88  f1-key                       value 1.
        88  enter-key                    value 1.
        88  fun-key-num-user             values 0 thru 127.
        88  fun-key-num-system           values 0 thru 26.
    05  crt-status-3                 pic 99 comp-x.
        88 raw-key-code                 values 0 thru 255.
             ...
 procedure-division.
             ...
*************************************************************
*                                                           *
*   The following shows the procedural code that would      *
*   evaluate the CRT status keys and direct processing      *
*   accordingly.                                            *
*                                                           *
*************************************************************

     evaluate terminate-key also function-key also adis-key
      when true  also any  also any
         if esc-key
             evaluate crt-status-3
               when 0  perform raw-key-0
               when 1  perform raw-key-1
               when 2  perform raw-key-2
               when 3  perform raw-key-3
                  ...
             end-evaluate
        else
             perform logic-for-terminator-key
        end-if

      when any  also true  also any
         evaluate crt-status-2
           when 1  perform user-function-1
           when 2  perform user-function-2
           when 3  perform user-function-3
           when 4  perform user-function-4
           when 5  perform user-function-5
            ...
         end-evaluate
      when any  also any  also true
```

```
            evaluate crt-status-2
              when 1 perform sys-function-1
              when 2 perform sys-function-2
              when 3 perform sys-function-3
              when 4 perform sys-function-4
              when 5 perform sys-function-5
                 ...
               end-evaluate
        end-evaluate
```

7.7 IF文（条件付きコンパイル）

原始プログラムのコードの一部を「条件付きでコンパイルする」ために、&IF文を使用することができる。下の例では、プログラムに下記の指令を指定してコンパイルする。

```
    CONSTANT WHERE "PC"
```

すると、コンパイル時に、"NO"という語が表示されて、実行用プログラム・コードにはGO TO文ではなく、EVALUATEが含まれる。

```
    $if WHERE = "PC"
         evaluate test-field
           when 5  perform test-a
         end-evaluate
    $if other-constant defined
    $display Program compiled with other-constant set
    $else
    $display NO
    $end
    $else
         go to test-a test-b depending on test-field
    $end
```

7.8 INSPECT文（計数、置換、変換）

INSPECT文を使用すると、指定した文字列が出現する回数を数え、その文字列を別の文字列で置き換えたり、文字の組合せを別の文字の組合せに変換したりすることができる。この文字列検査の条件の設定は非常に複雑になりうる。この動詞の変形および用途をいくつか下に例示する。

下に示すINSPECT文の各例において、この文を実行する直前のCOUNT-nはゼロであるとする。各例の結果は、その上の２つの連続するINSPECT文を実行した結果である。

例１

```
inspect item tallying
    count-0 for all "AB", all "D"
    count-1 for all "BC"
    count-2 for leading "EF"
    count-3 for leading "B"
    count-4 for characters;

inspect item replacing
    all "AB" by "XY", "D" by "X"
    all "BC" by "VW"
    leading "EF" by "TU"
    leading "B" by "S"
    first "G" by "R"
    first "G" by "P"
    characters by "Z"
```

ITEMの初期値	COUNT-0	COUNT-1	COUNT-2	COUNT-3	COUNT-4	ITEMの最終値
EFABDBCGABEFGG	3	1	1	0	5	TUXYXVWRXYZZPZ
BABABC	2	0	0	1	1	SXYXYZ
BBBC	0	1	0	2	0	SSVW

例２

```
inspect item tallying
        count-0 for characters
        count-1 for all "A";

    inspect item replacing
        characters by "Z"
        all "A" by "X"
```

ITEMの初期値	COUNT-0	COUNT-1	ITEMの最終値
BBB	3	0	ZZZ
ABA	3	0	ZZZ

例３

```
inspect item tallying
    count-0 for all "AB" before "BC"
    count-1 for leading "B" after "D"
    count-2 for characters after "A" before "C"

inspect item replacing
    all "AB" by "XY" before "BC"
    leading "B" by "W" after "D"
    first "E" by "V" after "D"
    characters by "Z" after "A" before "C"
```

ITEMの初期値	COUNT-0	COUNT-1	COUNT-2	ITEMの最終値
BBEABDABABBCABEE	3	0	2	BBEXYZXYXYZCABVE
ADDDDC	0	0	4	AZZZZC
ADDDDA	0	0	5	AZZZZZ
CDDDDC	0	0	0	CDDDDC
BDBBBDB	0	3	0	BDWWWDB

例４

```
inspect item tallying
    count-0 for all "AB" after "BA" before "BC";

inspect item replacing
    all "AB" by "XY" after "BA" before "BC"
```

ITEMの初期値	COUNT-0	ITEMの最終値
ABABABABC	1	ABABXYABC

例５

```
inspect item converting
    "ABCD" to "XYZX" after quote before "#".
```

The above INSPECT is equivalent to the following INSPECT:

```
inspect item replacing
    all "A" by "X" after quote before "#"
    all "B" by "Y" after quote before "#"
    all "C" by "Z" after quote before "#"
    all "D" by "X" after quote before "#".
```

ITEMの初期値	ITEMの最終値
AC"AEBDFBCD#AB"D	AC"XEYXFYZX#AB"D

7.9 定数名のNEXT指定

VALUE句の定数名形式のNEXT指定は常に、記憶領域の次のバイトが前のデータ宣言のどれだけ後ろにあるかを示すオフセットを指す。例を下に示す。

```
01  x1              pic x(10).
01  x2 redefines x1    pic x.
    78  next-offset    value next.
01  x3              pic xx.
```

next-offset中の値は、*x3*の開始ロケーションではなくて、*x1*の２番目のバイトである。
このことはOCCURS句に関して混乱を招く原因となりうる。たとえば、下記の例がある。

```
01  group-item.
    05  tabl occurs 10 times.
            78  offset-a   value next.
        10 elem            pic x.
            78  offset-b   value next.
    05  after-tabl        pic x(02).
```

*offset-a*は*elem*の最初の反復の開始点でのオフセットを指す。それに対して、 *offset-b* は、２番目の表要素*elem*の２番目の反復要素の開始ロケーションを指すのであって、*after-tabl*の開始ロケーションを指すのではない。*after-tabl*の開始ロケーションを得たい場合には、原文を下記のように記述すべきである。

```
01  group-item.
    05  dummy-item     pic x(10).
        78 offset-c            value next.
    05  tabl redefines dummy-item
             occurs 10 times.
           78 offset-a     value next.
        10 elem            pic x.
           78 offset-b     value next.
    05  after-tabl     pic x (02).
```

この例では、*offset-c*は*after-tabl*の開始オフセットを指す。

7.10 入力および出力の手続きを使用しての整列

SORT文を使用して、ファイル中のレコードを整列させることができる。下記のプログラムは入力ファイルおよび出力ファイルの名前をコマンド行から受け取る。レコードを入力し、そのレコードを整列し、その結果を出力する入力および出力の手続きにおいて、RELEASE文とRETURN文がそれぞれ使用される。

```
$SET ANS85
select ifile assign to ipf
    organization is line sequential
    file status is ipstat.
select sfile assign to "temp.dat".
select ofile assign to opf
    organization is line sequential.
fd ifile.
01 irec     pic x(80).
fd ofile.
01 orec     pic x(80).
sd sfile.
01 srec     pic x(80).
working-storage section.
01 ipstat.
  03 iskey1 pic x.
  03 iskey2 pic x.
01 ipf  pic x(20).
01 opf  pic x(20).
01 ext  pic x(20).
01 clin     pic x(132).
01 len  pic 9(2) comp-x.
01 a    pic 9(2) comp-x value 0.
procedure division.
    accept clin from command-line
    unstring clin delimited by space into ipf, opf, ext
    if ext not = space
        display "too many arguements
    end if
    sort sfile on ascending srec
        input procedure sortin
        output procedure sortout
    stop run.
sortin section.
    open input ifile
    read ifile
    perform until ipstat not = "00"
        move irec to srec
        release srec
        read ifile
    end-perform
    close ifile.
sortout section.
    return sfile at end go to sortout-exit
    display srec
    go to sortout.
sortout-exit.
    display "Done".
```

7.11 表の項目の整列

整列するために、OCCURS句中にキーを指定して表を下記のように定義する。

```
working-storage section.
01 group-item.
    05   tabl   occurs 10 times
                    ascending elem-item2
                    descending elem-item1.
         10 elem-item1 pic x.
         10 elem-item2 pic x.
    . . .
procedure division.
    . . .
    sort tabl.
    if tabl (1) . . .
```

これは単純な整列である。データ項目TablのOCCURS句内のキー定義に基づいて順序を判定して、表を昇順に整列している。具体的には、Elem-Item2が第１キー（昇順）であり、Elem-Item1が第２キー（降順）である。

要素全体を使用して表を整列するには、下記のようにする。

```
working-storage section.
01 group-item.
    05 tabl occurs 10 times
        10 elem-item1 pic x.
        10 elem-item2 pic x.
    . . .
procedure division.
    . . .
    sort tabl ascending.
    if tabl (1) ...
```

これは、表のすべての要素を使用して順序を判定して昇順に整列する、単純な整列である。

順序を決定するキーを指定して表を整列するには、下記のようにする。

```
working-storage section.
01 group-item.
    05  tabl    occurs 10 times
                    ascending elem-item3
                    descending elem-item1.
        10 elem-item1 pic x.
        10 elem-item2 pic x.
        10 elem-item3 pic x.
    . . .
procedure division.
    . . .
    sort tabl descending elem-item2 elem-item3
    if tabl (1) ...
```

この処理では、指定されたデータ項目をキーとして、表を整列する。OCCURS句の中でキーとして指定されていないが、Elem-Item2が第１キーとなる。Elem-Item3は第２キーとなる。Elem-Item3はこのソートに関してはDESCENDINGキーとして扱われる。なぜならば、SORT文に指定されたDESCENDING（キー・データ項目全体に影響）の方がOCCURS句内に指定されたASCENDINGよりも優先されるからである。

入れ子になった表を整列するに下記のようにする。

```
working-storage section.
01 group-item.
    05  tabl1   occurs 10 times
                    indexed by t1-ind t2-ind.
        10  tabl2 occur 5 times.
            15 group1.
                20 elem-item1 pic x.
            15 group2.
                20 elem-item1 pic 9.
    . . .
procedure division.
    . . .
    set t1-ind to 3
    sort tabl2 descending elem-item1 of group2
    if group1 (3 1) ...
```

この整列処理ではTabl2の３番目のインスタンス、つまりTabl2(3)だけが整列される。修飾されたデータ項目のGroup2のElem-Item1がキーに使用されている。手続き部で参照するには通常、Group2のElem-Item1は２レベルの添字付け／指標付けを必要とする。しかし、ここでは添字付けも指標付けも行われていない。（同様に、Tabl2は通常は１レベルの添字付けを必要とする。しかし、SORT文中のデータ名-2を添字付けすることはできない。その代わりに、整列対象のインスタンスを判定するために、T1-Indの値が使用されている。）

7.12 分割キー

プログラムに下記の定義が含まれているとする。

```
01 rec.
    03 forename     pic X(10).
    03 personnel-no pic X(4).
    03 surname      pic X(15).
```

the syntax:

```
    record key is fullname =
        surname forename
```

すると、COBOLシステムは下記のデータ項目から構成される明示的に定義された集団項目であるかのように、 fullname を扱う。

```
    03 surname      pic X(15).
    03 forename     pic X(10).
```

7.13 利用者による型定義ないし構造体

このCOBOLシステムでは、下記のようにデータを記述することができる。

```
01 struct-1        TYPEDEF.
   05  part-1  pic x (20).
   05  part-2  pic x(10).
01 USHORT  pic 9 (4)   comp-5 typedef.
```

これは、struct-1とUSHORTを新しいデータ型として定義している。それを次のように使用することができる。

```
01  a.
    05  b  struct-1.
    05  x  USHORT.
```

これは下記のように記述されているかのように解釈される。

```
01  a.
    05  b.
        10  part-1 pic x(20).
        10  part-2     pic x(10).
    05  x              pic 9(4) comp-5.
```

第8章：廃言語要素

この章には、ANSI X3.23-1985　COBOL仕様中に廃要素と指定されているものを列挙する。この章に列挙した廃要素は、以下のようである。

- 標準COBOL仕様の次回の改訂時に、削除されることになっている。
- 今後、機能拡張や変更や保守が行われない。
- X/Openに準拠する原始プログラム中では、使用するべきではない。

8.1 廃要素の一覧

- 長さが2桁以上の*定数* にALLを付けた表意定数を、数字または数字編集のデータ項目と関連付けること。
- 見出し部内の、AUTHOR，INSTALLATION，DATE-WRITTEN，DATE-COMPILED，SECURITYの各段落。
- 実行用計算機段落のMEMORY SIZE句。
- 入出力管理段落のRERUN句。
- 入出力管理段落のMULTIPLE FILE TAPE句。
- ファイル記述項内のLABEL RECORDS句。
- ファイル記述項内のVALUE OF句。
- ファイル記述項内のDATA RECORDS句。
- ALTER文。
- DISABLE文内のキー指定。
- ENABLE文内のキー指定。
- ENTER文。
- GO TO文中で*手続き名* を除外するオプション。
- OPEN文中のREVERSED指定。
- *定数* を指定したSTOP文。
- 区分化機能単位。
- デバッグ機能単位。

第9章：VS COBOL IIとの互換性

VS COBOL IIとの互換性をとる目的でこのCOBOLシステムに組み込まれている機能は、下記の2つの指令を用いて制御する。

FLAG "VSC2"

VSC2 "*整数*"

ここで、*整数*の値は下記のように互換性の水準を表わす。

VSC2 (1) VS COBOL II リリース 1.x

VSC2 (2) VS COBOL II リリース 2

VSC2 (3) VS COBOL II リリース 3.x

VSC2 (4) VS COBOL II リリース 4.x

上記の違いは、FLAG "VSC2" 指令を指定したときの、フラグの付けられ方にある。COBOLシステムは、これらの水準の内容の違いを示すことはない。フラグの形で示すだけである。

要素	**VSC2(1)**	**VSC2(2)**	**VSC2(3)**	**VSC2(4)**
ADD..TO..GIVING	ANSI'85のフラグが付けられる	ANSI'85のフラグが付けられる	サポートされている	サポートされている
ALPHABET 英数字字類検査	特殊名の中では必要語は禁止	特殊名の中では必要語は禁止	特殊名の中では必要語が必要	特殊名の中では必要語が必要
160文字の英数字定数	ANSI'85のフラグが付けられる	ANSI'85のフラグが付けられる	サポートされている	サポートされている
BINARY用途	受け入れられない	受け入れられない	サポートされている	サポートされている
CALL BY CONTENT	ANSI'85のフラグが付けられる	ANSI'85のフラグが付けられる	サポートされている	サポートされている
CALL ON EXCEPTION	ANSI'85のフラグが付けられる	ANSI'85のフラグが付けられる	サポートされている	サポートされている
字類名	利用者語の字類名は受け入れられない。大文字だけ可	利用者語の字類名は受け入れられない。大文字だけ可	特殊名段落の中に利用者語の字類名を追加定義できる。大文字と小文字の両方とも可	特殊名段落の中に利用者語の字類名を追加定義できる。大文字と小文字の両方とも可

要素	VSC2(1)	VSC2(2)	VSC2(3)	VSC2(4)
COMMON	受け入れられない	受け入れられない	サポートされている	サポートされている
DATA - レベル番号に続くデータ名をA領域に書く	ANSI'85のフラグが付けられる	ANSI'85のフラグが付けられる	サポートされている	サポートされている
DAY - OF - WEEK	受け入れられない	受け入れられない	サポートされている	サポートされている
DISPLAY WITH NO ADVANCING	ANSI'85 拡張機能のフラグが付けられる	ANSI'85 拡張機能のフラグが付けられる	サポートされている	サポートされている
END PROGRAM	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
ALSOを省略したEVALUATE	サポートされている	MF 拡張機能のフラグが付けられる	MF 拡張機能のフラグが付けられる	MF 拡張機能のフラグが付けられる
段落中に単独でないEXIT PROGRAM	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
EXTERNAL	受け入れられない	受け入れられない	サポートされている	サポートされている
ファイル状態コード	ANSI 74 コード	ANSI 74 コード	ANSI'85 コード	ANSI'85 コード
GLOBAL句	受け入れられない	受け入れられない	サポートされている	サポートされている
入れ子が7段階のOCCURS	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
相対編成と索引編成に対するOPEN EXTEND	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
任意のFILLERまたはデータ名	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
INITIAL プログラム名	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
2バイト文字を置換するINITIALIZE	受け入れられない	受け入れられない	サポートされている	サポートされている
INSPECT CONVERTING	受け入れられない	受け入れられない	サポートされている	サポートされている
LESS OR EQUAL, GREATER OR EQUAL, <=, >=	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている

要素	VSC2(1)	VSC2(2)	VSC2(3)	VSC2(4)
小文字	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
複数ファイルへのMERGE GIVING	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
NOT 範囲符	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
OPTIONAL RELATIVEファイル	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
PACKED - DECIMAL	受け入れられない	受け入れられない	サポートされている	サポートされている
PADDING CHARACTER	受け入れられない	受け入れられない	サポートされている	サポートされている
PIC の継続	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
利用者語と同じプログラム名	受け入れられる	受け入れられる	受け入れられない	受け入れられない
RECORD DELIMITER	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
RECORD IS VARYING	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
部分参照	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
REPLACE	受け入れられない	受け入れられない	サポートされている	サポートされている
SET TO ON/OFF	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
入れ子のSIGN	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
SORT DUPLICATES 指定	ORDERを使うと受け入れられない	ORDERを使うと受け入れられない	サポートされている	サポートされている
STANDARD -2	受け入れられない	受け入れられない	サポートされている	サポートされている
記号文字	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている
OCCURSとVALUEの併用	ANSI'85 のフラグが付けられる	ANSI'85 のフラグが付けられる	サポートされている	サポートされている

第10章：DOS/VS COBOL支援機能

Micro Focus COBOLシステムが、DOS/VS COBOLの構文をサポートする範囲は限られている。MVSおよびCMSに対してはVSE/ESAで実現されるので、このCOBOLシステムではVS COBOL II方言に関して区別をしていない。

DOS/VSのすべての予約語を使用できるようにするために、DOS/VS指令が用意されている。FLAG"DOSVS" 指令を使用すると、DOS/VS COBOLのものではない構文にフラグを付けることができる。

DOS/VS COBOLはOS/VS COBOLとほとんど同じであるが、下記の違いがある。

ASSIGN (EXTERNAL) 指令 を設定してあると、SET命令中で定義される変数は下記のどれかである。

1. 最後のハイフンに続くすべてのテキスト。ただし、P, R, S, SR, ASのどれかであるものを除く。
2. 最初のハイフンに先行する名前の部分。

特殊レジスタ(***言語リファレンス*** の *COBOL の概念* を参照) の中に、DOS/VS用の登録をする必要がある。

特殊レジスタ:

COM-REG
暗黙のデータ記述:

PIC X(11)
使用法:

DOS/VS COBOLにおいては、この項目はDOS通信領域の12バイト目から22バイト目までに相当する。しかし、Micro Focus COBOLシステムでは、この通信領域は割り当てられて利用されるだけで、プログラムによってその中に入れられるデータに対して特別な意味をもつことはない。

第11章: Microsoft COBOL V1.0および V2.0の構文支援機能

ここでは、IBM 1.0の構文およびMicrosoft V2.0の構文との互換性をとる目的で、このCOBOLシステムが受け入れるMicrosoftの構文を列挙する。Microsoft COBOLの機能の多くは、既にこのCOBOLシステムに組み入れられている。これらは、***言語リファレンス***の本文に記載してある。 この付録に列挙する互換性をもたせるための構文の中には、標準COBOL構文と全く同じであるが、原始コードをコンパイルするときにMSシステム指令を設定すると、違った動作をするものがある。他のMicrosoftの構文は、このMS指令を設定した場合にだけ、利用できる。Microsoftの構文の詳細については、***IBM COBOL 1.0***マニュアル、および***Microsoft COBOL V2.0***マニュアルを参照。ここでは、IBM COBOL V1.0を対象とする。Microsoft V2.0 の拡張構文に触れるときは、その旨を明記してある。

11.1 特殊レジスタのLINとCOL

LIN とはそれぞれ、画面上の現在のカーソルの行位置と文字位置を示すために使用する。この2つは、ACCEPT文、DISPLAY文、EXHIBIT文でサポートされている位置指定の一部を構成する。これらの文については、この章の中で後述する。この2つのレジスタをプログラムに含めるためには、原始コードをコンパイルするときに、MSシステム指令を設定しなければならない。
LINとCOLの形式は、共にPIC S9(4) COMPである。この2つの特殊レジスタは、通常のデータ項目と同じように使用できる。ただし、これらを自分で定義してはならない。これらはシステムによって自動的に定義される。位置指定の中で使用するのに先立って、この2つのレジスタの値を適切に設定しなければならない。
MS指令を設定したときは、COLUMNの省略形としてのCOLは、COBOLの予約語としては使用できなくなるので注意すること。

11.2 環境部

11.2.1 特殊名段落

下記の2つの句がサポートされている。

1. PRINTER IS 呼び名

2. SWITCH-n 注釈名 { ON / OFF } IS 条件名

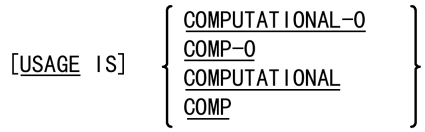
nは0から8の範囲の整数である。省略時解釈の設定は、OFFである。

11.3 データ部

11.3.1 USAGE句

一般形式

下記の書き方のUSAGE句がサポートされている。



構文規則

1. COMPと COMPUTATIONALは同義語である。
2. COMP-0と COMPUTATIONAL-0は同義語である。

一般規則

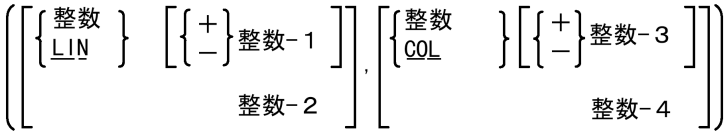
1. 原始コードをコンパイルするときにMSシステム指令を設定すると、USAGE IS COMPUTATIONALはUSAGE IS DISPLAYと同じ意味となる。
2. USAGE IS COMP-0は、数字データ項目に対してだけ適用できる。該当する項目のPICTUREにS9(5)を超えないものを指定すると、その項目に PIC S9(4) USAGE COMP（つまり、2バイトの符号つき2進数）を指定したことになる。これ以外の場合は、この指定の働きは、実際に指定されたPICTURE句にUSAGE DISPLAYを付加することである。

11.4 手続き部

11.4.1 位置指定

位置指定は、この付録に記述するACCEPT文、DISPLAY文、EXHIBIT文における、画面位置を指定するものである。この指定を行う場合は、原始コードをコンパイルするときに、MSシステム指令を設定しなければならない。

一般形式



構文規則

1. 上記の書き方中のコンマは、必須である。

11.4.2 ACCEPT文

下記の書き方のACCEPT文がサポートされている。

一般形式

ACCEPT[位置指定]一意名[WITH句]

一般規則

1. 原始コードをコンパイルするときにMSシステム指令を設定すると、ACCEPT文は集団項目を基本項目として扱う。つまり、ACCEPT文は対象項目を1つのものとして受け取るのであり、その下位に属する基本項目に分けては取り扱わない。
2. 原始コードをコンパイルに投入するときにMSシステム指令を設定すると、最初の作用対象が現れるべき画面上の位置の指定を省いた場合、省略時解釈として現在のカーソルの位置が採られる。
3. ***言語リファレンス***のプログラムの定義の章中のACCEPT文の節に記載した書き方2にも、上記の規則1と2は当てはまる。
4. WITH指定のオプションと 同義語を下に示す。

   AUTO/AUTO-SKIP
   BACKGROUND-COLOR
   BELL/BEEP
   BLINK
   REVERSE-VIDEO
   RIGHT-JUSTIFY
   SECURE/NO-ECHO
   SIZE
   SPACE-FILL
   TRAILING-SIGN
   UNDERLINE
   UPDATE
   ZERO-FILL

   これらのオプションは、***言語リファレンス***のプログラムの定義の章中の ACCEPT ACCEPT文の節に記載した書き方2にも当てはまる。
   標準COBOLとは異なるオプションを以下に説明する。

a. UPDATE: 原始コードをコンパイルするときにMS指令を設定すると、UPDATEオプションを指定しなかった場合、入力用のデータ項目は最初は空白で表示される。（MS指令を設定しないと、データを最初どう表示するかは構成オプションとなる）。
UPDATEオプションを指定した場合は、入力用のデータ項目は最初に表示される。

b. PROMPT: 原始コードをコンパイルするときにMS指令を設定すると、このオプションを、プロンプト文字を表示させるために指定しなくてもよい。

c. LENGTH-CHECK: このオプションを指定すると、 EMPTY-CHECKオプションが暗黙のうちに設定される。したがって、操作員は何かを入力しなければならない。このオプションを指定した場合、原始コードをコンパイルするときにMS指令を設定しなければならない。

d. FOREGROUND-COLOR ： これらの句中の整数-1は、それぞれ、画面項目の前景色 と背景色を指定するものである。この値は0から15であり、下記のように定義されている。

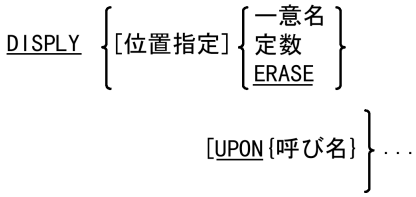
0	黒（black）	8	灰色（grey）
1	青（blue）	9	淡青色（light blue）
2	緑（green）	10	淡緑色（light green）
3	藍（cyan）	11	淡藍色（light cyan）
4	赤（red）	12	淡赤色（light red）
5	赤紫（magenta）	13	淡紫色（light magenta）
6	茶（brown）	14	黄色（yellow）
7	白（white）	15	高輝度の白（high intensity white）

カラー画面上で、前景色に値8から15の整数を指定することは、値0から7の整数と HIGHLIGHTを指定することに相当する。白黒画面上では、このことは単にHIGHLIGHTを指定することに相当する。
カラー画面上で、背景色に値8から15の整数を指定することは、値0から7の整数とBLINKを指定することに相当する。白黒画面上では、このことは単にBLINKを指定することに相当する。

11.4.3 DISPLAY文

一般形式

下記の書き方のDISPLAY文がサポートされている。



一般規則

1. 原始コードをコンパイルするときにMSシステム指令を設定すると、DISPLAY文は集団項目を基本項目として扱う。つまり、DISPLAY文は対象項目を1つのものとして表示するのであり、その下位に属する基本項目に分けては取り扱わない。
2. 原始コードをコンパイルするときにMSシステム指令を設定すると、最初の作用対象が現れる画面上の位置の指定を省いた場合、省略時解釈として現在のカーソルの位置が採られる。
3. ***言語リファレンス***の*プログラムの定義*の章中のDISPLAY文の節に記載した書き方2にも、上記の規則1と2は当てはまる。
4. ERASEを指定すると、画面上の現在のカーソルの位置から先が消去される。

以下の2つの規則は、***言語リファレンス***の*プログラムの定義*の章中のDISPLAY文の書き方2への追加事項である。

5. WITH指定のオプションと同義語を下に示す。

BACKGROUND-COLOR
BELL/BEEP
BLANK
BLINK
FOREGROUND-COLOR
HIGHLIGHT
REVERSE-VIDEO
SIZE
UNDERLINE

6. FOREGROUND-COLORとBACKGROUND-COLOR: これらの句中の整数-1は、それぞれ、画面項目の前景色と背景色を指定する。この値は0から15であり、下記のように定義されている。

0	黒（black）	8	灰色（grey）
1	青（blue）	9	淡青色（light blue）
2	緑（green）	10	淡緑色（light green）
3	藍（cyan）	11	淡藍色（light cyan）
4	赤（red）	12	淡赤色（light red）
5	赤紫（magenta）	13	淡紫色（light magenta）
6	茶（brown）	14	黄色（yellow）
7	白（white）	15	高輝度の白（high intensity white）

カラー画面上で、 前景色に値8から15の整数を指定することは、値0から7の整数とHIGHLIGHTを指定することに相当する。白黒画面上では、このことは単に HIGHLIGHTを指定することに相当する。
カラー画面上で、 背景色に値8から15の整数を指定することは、値0から7の整数とBLINKを指定することに相当する。白黒画面上では、このことは単に BLINKを指定することに相当する。

11.4.4 EXHIBIT文

CHANGEDもNAMEDも指定しないで EXHIBIT文を使用すると、 NAMEDを指定したように動作する。

一般形式

下記の書き方のEXHIBIT文がサポートされている。

```
EXHIBIT[CHANGED][NAMED]      {[位置指定]{一意名 / 定数 / ERASE}}...
                [UPON 呼び名]
```

構文規則

1. 呼び名は、sysout，syslist，syslst，syspunch，syspch、操作卓、プリンタに関連するものとする。

一般規則

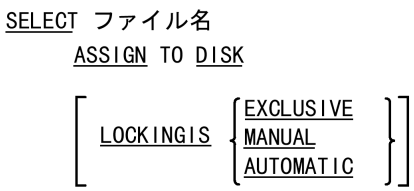
1. ERASEを指定すると、画面上の現在のカーソルの位置から先が消去される。

11.5 Microsoft V2.0 追加構文支援機能

11.5.1 レコードのロック

MS指令を設定すると、 SELECT文中でLOCKING句が使用できるようになる。

一般形式



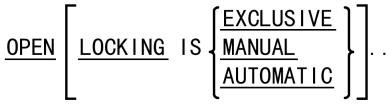
一般規則

1. 順ファイルと行順ファイルに関しては、 EXCLUSIVE（排他的なロック）だけが許される。
2. MANUALを指定すると、対象ファイルに複数のロックを適用できる。
3. AUTOMATICを指定すると、現在レコードに単一のロックが自動的に適用される。
4. 順ファイルと行順ファイルに関する省略時解釈のロック方式は、EXCLUSIVEである。
5. 相対ファイルと索引ファイルに関する省略時解釈のロック方式は、AUTOMATICである。
6. MANUALを指定すると、UNLOCK文によってすべてのロックが解除される。

11.5.2 OPEN LOCKING文

MS指令を設定すると、 OPEN文中でLOCKING句が使えるようになる。

一般形式



一般規則

1. 順ファイルと行順ファイルに関しては、EXCLUSIVEロックだけが許される。
2. NO LOCKING指定をすると、暗黙的にEXCLUSIVEと想定される。

11.5.3 READ文（マニュアルモード）

一般形式

READファイル名[NEXT]RECORD[LOCK][WAIT][INTO一意名]

一般規則

1. MS指令を設定すると、READ文中でWAIT句が使えるようになる。これは注記として扱われる。

11.5.4 START文

一般形式

STARTファイル名[LOCK][WAIT]

一般規則

1. MS指令を設定すると、START文中でLOCK句およびWAIT句が使えるようになる。これは注記として扱われる。

第12章: Ryan McFarland COBOL V2.0 構文支援機能

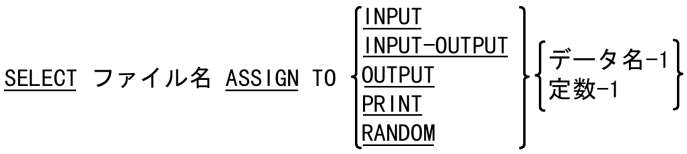
ここでは、Ryan McFarland COBOL V2.0との互換性をとる目的で、このCOBOLシステムが受け入れる構文を列挙する。Ryan McFarland COBOLの機能の多くは既にこのCOBOLシステムに組み入れられている。これらは、***言語リファレンス***の本文に記載してある。 ここに列挙する互換性をもたせるための構文の中には、標準COBOL構文と全く同じであるが、原始コードをコンパイルするときにRMシステム指令を設定すると、違った動作をするものがある。 Ryan McFarland 構文の詳細については、***RM/COBOL Language Reference Manual*** バージョン 2.0 のマニュアルを参照。

12.1 環境部

12.1.1 ASSIGN句

一般形式

ASSIGN句では、下記の書き方がサポートされている。



一般規則

1. 語INPUT, OUTPUT, INPUT-OUTPUT, PRINT, RANDOMは注記として扱われる。

12.1.2 ORGANIZATION句

一般形式

ORGANIZATION句では、下記の書き方がサポートされている。

```
ORGANIZATION IS BINARY SEQUENTIAL
```

一般規則

1. この句は、ORGANIZATION IS SEQUENTIALと等しいものとみなされる。
   ただし、原始コードをコンパイルするときにRM"ANSI" 指令を指定すると、この句はORGANIZATION IS LINE SEQUENTIALと等しいものとして扱われる。
2. ファイルに対してORGANIZATION句を指定しないと、ORGANIZATION IS LINE SEQUENTIALであるものと想定される。ただし、原始コードをコンパイルするときに、RMシステム指令を設定していることを前提とする。
   ただし、原始コードをコンパイルするときにRM"ANSI" 指令を指定すると、ORGANIZATION IS SEQUENTIALであるものと想定される。

12.2 データ部

12.2.1 VALUE OF LABEL句

一般形式

FD記述項に、下記の句が受け入れられる。

```
VALUE OF LABEL IS 定数
```

一般規則

1. これは注記として扱われる。

12.2.2 文字定数の長さ

原始コードをコンパイルするときにRMシステム指令を設定すると、データ部内で2047文字までの長さの文字定数が許される。

12.2.3 省略時解釈の符号の表現

構文:

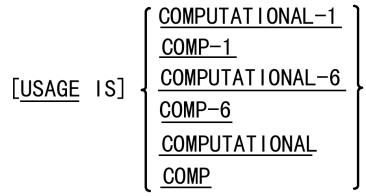
```
NUMERIC SIGN IS TRAILING SEPARATE
```

原始コードをコンパイルするときにRMシステム指令を設定すると、特殊名段落において、上記の句が指定されているものと想定される。 ただし、原始コードをコンパイルするときにRM"ANSI" 指令を指定すると、この句は指定されていないものとされる。

12.2.4 USAGE句

一般形式

下記の書き方のUSAGE句がサポートされている。



構文規則

1. COMP と COMPUTATIONAL は同義語である。
2. COMP-1 と COMPUTATIONAL-1 は同義語である。
3. COMP-6 と COMPUTATIONAL-6 は同義語である。

一般規則

1. 原始プログラム中でUSAGE COMP-1と定義された各データ項目に対して、COBOLシステムは2バイトの符号付き2進データ項目を割り当てる。このデータ項目は、-32Kから+32Kの範囲の値を保持できる。このデータ項目の割当ては、PICTURE文字列の値に左右されない。結果として、USAGE COMP-1データ項目は、標準COBOLでPIC S9(4) COMPと指定したのと同様に扱われる。
2. COMP-6データ項目は、COBOL COMP形式に変換される。その結果として、各データ項目に割り当てられる大きさがRM/COBOLシステムの場合よりも小さくなると、空き部分に空(X"00") が埋められる。RM/COBOL でのUSAGE 句の詳細については、***互換性ガイド*** を参照。
3. USAGE IS COMPUTATIONALは、USAGE IS DISPLAYと同様に扱われる。

12.3 手続き部

12.3.1 CALLパラメータとしての定数

定数を、CALL文のパラメータに使用できる。

12.3.2 EXIT PROGRAM文

原始コードをコンパイルするときにRMシステム指令を設定すると、EXIT PROGRAM文は、終了させようとする副プログラムによって開かれていたすべてのファイルを閉じる。
ただし、RM"ANSI" 指令を指定すると、EXIT PROGRAM文はその処理を行わない。

12.3.3 境界検査

原始コードをコンパイルするときにRMシステム指令を設定すると、添字の境界は検査されない。

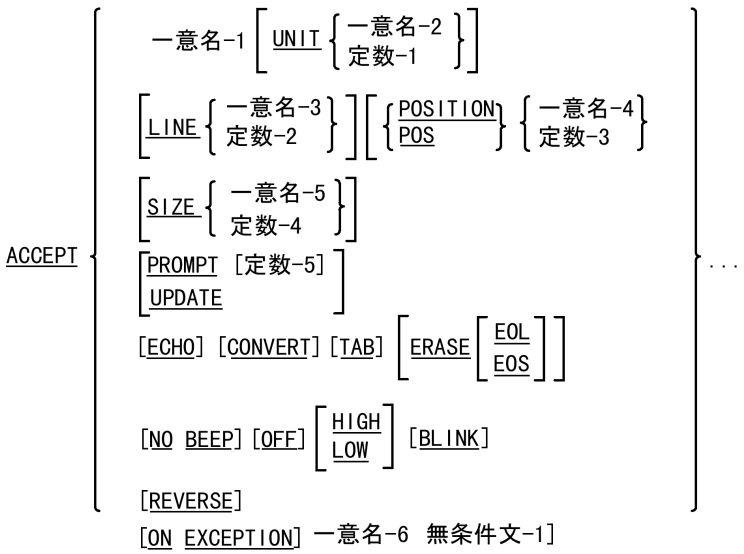
12.3.4 指標データ項目の大きさの割当て

原始コードをコンパイルするときにRMシステム指令を設定すると、指標データ項目には、通常の4バイトの代わりに2バイトが割り当てられる。

12.3.5 ACCEPT文

一般形式

下記の書き方の ACCEPT文がサポートされている。



構文規則

1. 一意名-3、一意名-4、一意名-5を指定する場合、整数の一意名とする。定数-2、定数-3、定数-4を指定する場合、整数の定数とする。
2. 定数-5は1文字とする。

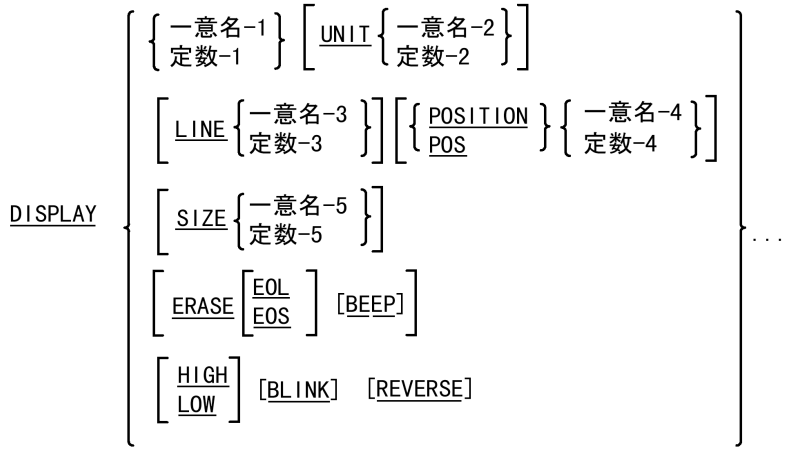
一般規則

1. この書き方は、***言語リファレンス***の*プログラムの定義*の章のACCEPT文の書き方5と等しいものとして扱われる。ただし、作用対象は、複数指定できることに注意。

12.3.6 DISPLAY文

一般形式

下記の書き方のDISPLAY文がサポートされている。



構文規則

1. 一意名-3、一意名-4、一意名-5を指定する場合、整数の一意名とする。 定数-2、定数-3、定数-4を指定する場合、整数の定数とする。
2. 定数-5は1文字とする。

一般規則

1. この書き方は、***言語リファレンス***の*プログラムの定義*の章のDISPLAY文の書き方3と等しいものとして扱われる。

12.3.7 英数字データ項目に対する非標準操作

英数字の値を数字データ項目中に入れることができる。ただし、右側に桁寄せされ（受取り側項目が数字である場合と同じ）、左側に空白が埋められる。たとえば、"AB" をPIC 9(5)の項目へ転記すると、結果は"　　AB"となる。

12.3.8 順ファイルに対するOPENとCLOSE

RM/COBOLでは、順ファイルに対する OPEN文と CLOSE文にNO REWINDを指定できる。標準COBOLでは、これらの文は注記にとどまる。

12.3.9 PERFORM文

通常のCOBOLシステムでは、スタックを使用してPERFORM文を処理している。これに対してRM/COBOLシステムでは、個々の手続き名に戻り番地を結び付ける。この結果、RM/COBOLシステムでは、PERFORM文のすべての終了点が、実際に使用されるまで有効となる。これに対して通常のCOBOLシステムでは、ある1時点では、実行範囲内に入っている最も内側のPERFORM文の終了点だけが有効である。
ただし、 PERFORM-TYPE"RM" 指令を設定すると、このCOBOLシステムはRM/COBOLと同じ方法でPERFORM文を処理するようになる。

12.3.10 手続き名

データ名と同じ名前の手続き名を使用できる。

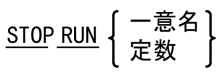
12.3.11 行順ファイルに対するREWRITE

行順ファイルに対して REWRITE文を適用できる。ただし、新しいレコードの長さは、元のレコードの長さと同じであることを前提とする。

12.3.12 STOP RUN文

一般形式

下記の書き方のSTOP RUN文がサポートされている。



構文規則

1. 一意名は、整数データ項目とする。
2. 定数は、整数とする。

一般規則

1. 整数データ項目または定数の値は、特殊レジスタのRETURN-CODEに入れられる。

12.3.13 ファイル入出力状態コード

原始コードをコンパイルするときに RMシステム指令を設定すると、COBOL入出力状態コードは、RMの同等のものと対応付けられる。

12.3.14 ロックされたレコード

原始コードをコンパイルするときにRMシステム指令を設定すると、レコードがロックされていることが判明した場合に、下記の処理が行われる。

- 適用可能な 宣言節が該当するファイルにあり、そのファイル状態値がその宣言節の実行のきっかけとなるものである場合、
    - 制御は宣言節に移される。
    - レコード領域には、読み込まれたレコードが正しく入れられる。
    - 入出力状態コードには、「レコードがロックされている」状態が示される。
- 適用可能な宣言節が該当するファイルにないか、またはそのファイル状態値がその宣言節の実行のきっかけとなるものではない場合、ロックが解除されるまで、そのレコードの読込みが繰り返し試みられる。

第13章: Data General Interactive COBOL V1.3 構文支援機能

この章には、Data General Interactive COBOL V1.3との互換性をとる目的で、このCOBOLシステムが受け入れる構文を列挙する。Data General Interactive COBOLの機能の多くは、既にこのCOBOL製品に取り入れられている。これらは、この***言語リファレンス***に記述してある。この章に記載した互換性をもたせるための構文の中には、形の上では標準COBOLと全く同じであるが、原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、違った動作を示すものがある。Data Generalの構文全般の記述については、***Data General Interactive Programmer's Reference Manual*** を参照のこと。

注: DG指令を指定しても、予約語に変動はない。

13.1 環境部

13.1.1 スイッチ名

COBOLでサポートされているスイッチ0から8に加えて、別の形のスイッチ名としてAからZの範囲内の大文字を使用できる。

これらの文字は、スイッチ0からスイッチ25に対応している。実行時に命令行上に指定するのは該当する数字であって、文字ではない。

たとえば、スイッチJをオンにするには、実行命令の後ろに次のように指定する。

```
+9 ファイル名
```

しかし下記のように、CALL文の中でランタイムスイッチを指定することはできない。

```
CALL "PROG.INT/A"
```

このような指定方法は、サポートされていないからである。

13.1.2 ディスク上のファイル名

ファイルをディスクに割り当てると、ディスク上のファイル名は大文字で付けられる。ファイル名を小文字で指定しても、大文字に変換される。

13.1.3 DATA SIZE 句

SELECT文の中で下記の句を使用できる。

```
[DATA SIZE IS 整数]
```

これは注記にとどまる。

13.1.4 INDEX SIZE句

相対ファイルまたは索引ファイル用のSELECT文の中で、下記の句を使用できる。

```
[INDEX SIZE IS 整数]
```

これは注記にとどまる。

13.1.5 副レコードキーの重複

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、DUPLICATES指定をしてもしなくても、索引ファイル中でレコードキーの重複が許される。

13.1.6 副レコードキー

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、 副レコードキーに主レコードキーと同じ領域を割り当てることができる。

UNIX:
UNIXでは、START文とREAD文、またはどちらか一方を設定すると、エラーメッセージが表示されるので注意する必要がある。これによってプログラムの実行が妨げられることはない。これは単に、コーディングに問題があることを知らせるものである。

13.1.7 入出力管理段落

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、SAME AREA指定はSAME RECORD AREA指定と等しいものとして扱われる。

13.2 データ部

13.2.1 VALUE 句

文字データ項目のデータ記述中のVALUE句の中で、数字定数を使用できる。

13.2.2 画面節

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、TO項目またはUSING項目と併用したHIGHLIGHT指定は、画面上のすべての非保護の領域を強輝度で表示する。

13.3 手続き部

13.3.1 CALL 文

一般形式

下記の書き方の CALL文がサポートされている。

```
CALL PROGRAM プログラム名定数[USING{データ名} ]
     [ON EXCEPTION 無条件文]
```

一般規則

この書き方はCHAIN文と同等に扱われる。

13.3.2 COPY INDEXED 文

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、動詞COPYの後ろに語INDEXEDを続けて書くことができる。語INDEXEDは、注記として扱われる

13.3.3 DISPLAY 文

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、WITH NO ADVANCINGを指定をしたDISPLAY文の動作は、Data General Interactive COBOLと同様のものに変更される

13.3.4 ファイル共有の構文

原始プログラムをコンパイルするときにDGコンパイラ指令を設定すると、索引ファイルおよび相対ファイル用の省略時解釈のロック方式は、ともにMANUAL WITH LOCK ON MULTIPLE RECORDSとなる

13.3.5 OPEN 文

動詞OPENの後ろに語EXCLUSIVEを続けて書くことができる。これは、WITH LOCK指定と等しいものとして扱われる

13.3.6 READ 文

ファイル名と補助語のRECORDの後ろに、必要語LOCKを続けて書くことができる。こうすると、READ文によって、読み込んだレコードがロックされる。

この書き方は、行順ファイルには適用されない。