HEWLETT PACKARD

HP-16C

横河・ヒューレット・パッカード株式会社

HP-16C
操作ハンドブック

00016-90005 Rev.B

Printed in Canada

はじめに

この度は当社製品をお選び頂きましてありがとうございました。当社の計算機はスペースシャトルの宇宙飛行士を始め，数多くの分野で世界各国の技術者，科学者，教授や学生の方々にご愛用頂いております。

HP-16C は多くの機能を備えた高性能な計算機で，広くコンピュータやマイクロプロセッサに関連した仕事や研究に従事する方々，特にプログラマ，設計技術者に便利にご使用頂けるように設計してあります。HP-16C には次のような優れた機能があります。

- 4 種類の数値表記（16 進数，10 進数，8 進数，2 進数）の整数についてそれぞれ 1 の補数，2 の補数，符号なしの各状態で数学演算が出来る。
- ワード長が可変：最大 64 ビットまでユーザが自由に指定できる。
- 論理演算やビット操作も可能です。
- 203 バイトのユーザ用メモリーがあって，最大 203 行のプログラムが可能です。
- 実数の演算も出来ます。
- 不揮発性メモリーを採用したので，データやプログラム命令がスイッチを切っても消えない。
- 消費電力が非常に少く，電池の寿命が長くなった。

この本では読者の皆様がコンピュータの構造や 2 進数演算については既にある程度の基礎知識をお持ちのものとして説明を進めます。また広範囲な分野での説明をしますので，各キーの働きなどを簡単に見るには，巻末の機能の要点兼索引を見てください。

第 1 部は HP-16C の基礎知識で，当社の計算機独特の [ENTER] キーを使用する計算方式(RPN 方式)を始め，HP-16C の操作方法を説明します。第 2 部は HP-16C のプログラムで，プログラムの作成方法とその機能について説明します。プログラムの各章は最初に一般的な説明，

次にその例，最後に各機能に関する詳細な説明をするという構成で，HP-16C の動作内容を理解しやすく心掛けました。

第１部の第１章，第２章と第２部（HP-16C のプログラム）で説明する機能は他の HP 計算機にも使用しているものですが，HP-16C に独特の機能は第１部の第３章から第６章までにまとめて説明します。

まず 10 ページから始まる HP-16C を使ってみましょうを読みながら，実際に HP-16C を操作してみてください。大まかな使い方をお分かり頂けると思います。

付録の部分にエラーとフラグの状態，操作の分類，保証と修理についての説明をまとめました。また機能の要点兼索引には各キーの機能の簡単な説明と参照ページを付記しました。詳細な説明が必要な時にはそのページを見てください。

目　　次

HP-16C を使ってみましょう

HP-16C は整数演算状態でも実数演算状態でも計算出来る高性能な計算機です。整数演算状態では，整数の２進数演算，数値表示の基数変更，ビット操作，論理演算が出来，また実数演算状態では広範囲な10進数の実数計算が出来ます。プログラムはどちらの演算状態でも可能です。また HP-16C は不揮発性メモリーを採用したので，データやプログラム命令はキー操作でリセット（内容の消去）をしなければいつまでも（電源スイッチを切った後でも）記憶しています。

電力消費が非常に小さいのも HP-16C の大きな特長の一つで，充電アダプタが不要になり，軽量で薄型な計算機が実現しました。消費電力が少ないので，普通の使用法では電池の平均寿命が６～12か月と長持ちします。それに電圧低下の表示機能を付けたので，計算機が動作しなくなる前に電池交換の時期になったことを表示します。しかし長期間使用しないときは電池液漏れによる内部故障を防ぐために，電池を取り出しておくことをお勧めします。

また電力消費を節約して電池の寿命を少しでも延ばすために，HP-16C のキー操作を何もしない放置状態で約10分放置すると自動的にスイッチが切れるようにしてあります。

キー操作

当社の計算機はどれも他社の計算機とは異なる独特の計算方式（RPN―逆ポーランド法と呼び，[ENTER] キーを使用する方式）を採用しています。この計算方式は少しも難しくはなく，計算途中に括弧を使わなくてよいという利点があります。メモリー・スタック（X～T レジスタという演算レジスタ兼メモリーを組み合わせたもの）を使って計算を行います。ここで実際に四則演算キーを使った簡単な計算をしてみましょう。

計算機の電源スイッチを入れて（何も表示していなかったら[ON]を押してください），[HEX]，[DEC]，[OCT]，[BIN] のどれかのキーを

押し，整数演算状態の数値表示の基数（16 進，10 進，8 進，2 進）を選びます。これで表示する数値の表現法が決まり，表示の右側にはそれに対応して h，d，o，b のどれかの文字を表示します。無指定状態（工場出荷後に初めて [NO] を押したとき，または不揮発性メモリーをリセットしたとき）では 16 進数（整数）表示になっています。[g] [CLx]（青色の文字の機能）* を押すと表示をクリアして 0 に戻ります。

計算をするには，まず第一の数値をキーインして [ENTER] を押し，次に第二の数値をキーインして目的のキーを押します。演算キーを押すと計算を実行し，直ちに答を表示します。数字キーを間違えて押してしまったときには，[BSP] を押して間違えた数字を消してから正しい数字をキーインします。

整数の計算

数値表示の基数のどれかを選ぶと，計算機は整数演算状態になります。

問　題†	キー操作	表　示
（整数の 2 進数）	[g] [CLx] [BIN]	0 b
1111－1	1111 [ENTER] 1 [−]	1110 b
1111×11	1111 [ENTER] 11 [×]	101101 b

表示している数値をそのまま使って次のように計算することもできます。

問　題	キー操作	表　示
101101÷10	10 [÷]	10110 b
10110 AND 1111	1111 [f] [AND]	110 b

---

* 当社の計算機を始めてご使用になる方は，大部分のキーに 3 種の文字や記号が付いているのに注意してください。第一機能（キーの上面に白色の文字で記してある機能）を使用したいときには直接そのキーだけを押します。黄色または青色文字の第二機能を使用したいときには，まず黄色の [f] キー（黄色文字の機能を使うとき）または青色の [g] キー（青色文字の機能を使うとき）を押してからそのキーを押します。

† [f] [STATUS] と押すと 2-16-0000 と表示するはずです。このように表示しないときは 38 ページを見てください。

(+ の計算をすると C の文字を表示して，キャリー (carry) フラグをセットしたことを表示します。フラグは 37 ページで説明します。g CF 4 と押すとフラグと C の文字をクリアします。)
以上四つの計算で次の点に注意してください。

- 演算キーを押す前に 2 個の数値を両方ともキーインする。
- ENTER を使うのは 2 個の数値を続けてキーインするときに 2 数の間を区切るときだけです。
- 演算キー（この場合は −, ×, ÷, AND ）を押すと直ちに演算を実行して，結果を表示します。

実数の計算

HP-16Cを 10 進数の実数演算状態にすると実数演算ができます。FLOAT キーを押すと HP-16C は整数演算状態から実数演算算状能に切り替わって，指定した小数点以下の桁数を表示するようになります。

問　題	キー操作	表　示
(実数の 10 進数)	f FLOAT 4 BSP	0.0000
−4.9 ÷ 6	4.9 CHS ENTER 6 ÷	−0.8167
$\sqrt{60}$	60 g √x	7.7460

プログラムの利用

プログラムを書く　HP-16C はキー操作通りのプログラム，つまりプログラムは手操作で計算したときのキーの操作順序をそのまま書き出したものです。

例　表示の数値に次々と 1 を加えるプログラムを書いてみましょう。

キー操作	表　示*	
[g] [P/R]	000-	HP-16C をプログラム入力状態にする(PRGM の文字を表示する)。000 ライン。
[f] CLEAR [PRGM]	000-	プログラム用メモリをクリアする。
[g] [LBL] A	001-43,22, A	このプログラムのラベルを A にする。
1	002- 1	002 ライン：1
[+]	003- 40	003 ライン：プログラム走行時に表示している値に 1 を加える。
[f] SHOW [BIN]	004- 42 26	一時停止して，答を 2 進数で表示する。
[GTO] A	005- 22 A	ループの実行を続ける。
[g] [P/R]		HP-16C を計算状態に戻す。PRGM の文字が消える。表示は前の計算結果。

プログラムの走行　最初の値(下の例では 0)をキーインして表示させます。[ENTER] を押す必要はありません。プログラムの走行（実行とも言います）開始によって計算する 2 個の数値が区切られるからです。このプログラムはキーインした値に次々に 1 を加えて行きます。

キー操作	表　示	
[DEC]		10進数の整数演算状態に切り替える。(どの基数表示で始めてもよいが，プログラム中では 2 進数で表示します。)
16 [f] [WSIZE]		ワード長を 16 にする。
0	0 d	最初の値は 0。

*表示するのはライン番号とキーコードです。キーコードは 1 桁または 2 桁の数字で押したキーの上下左右位置を表すものです。

キー操作	表　示	
GSB A	1 b	ラベル A のプログラムを探し，走行を開始する。表示する数値は 2 進数。
	10 b	
	11 b	
	100 b	
	⋮	
R/S	22 d	無限ループになっているので，プログラムの走行を止めるには R/S (run/stop,走行兼停止) を押す。 R/S を押した瞬間の 2 進数に等しい 10 進数を表示する。

これまでの説明で HP-16C の使い方が少し分かったと思います。ただしこれは使い方のほんの一部だけで，外に数多くの優れた HP-16C の機能が揃っています。第 1 部の HP-16C の基礎知識に進んでください。

第 1 部
HP-16Cの
基礎知識

第1章　まず始めましょう

この章ではHP-16Cをごく普通に使うために必要な知識——数字の入力方法，表示をクリアする方法，[ENTER]を使った計算方式，不揮発性メモリーについて——を詳しく説明します。ここの例題は整数演算状態でやっていますが，機能は特に明記した場合を除いてすべて整数演算状態でも実数演算状態でも同じように動作します。この章の説明は主として，当社の計算機に慣れてないためその特性を良くご存知ない方に読んで頂きたいと思います。

スイッチの入り切り

HP-16Cのスイッチを入れたり切ったりするには [ON] キーを押します*。スイッチを入れてあってもキー操作などを何もしない放置状態で約10分放置すると，電池の寿命を延ばすために自動的にスイッチが切れます。

キー操作

キーの第一機能と第二機能

HP-16Cの大部分のキーは，第一機能と2種類の第二機能があります†。第一機能はキーの上面の文字や記号で，その機能を使いたいときはそのキーだけを押します（例：[+]）キーの向こうの黄色文字，また

---

* [ON] キーは他のキーより低くしてあります。これは他のキーの操作中に誤って [ON] キーを押してしまわないように配慮したためです。

† 本書の説明では，機能そのものを指すときには [MEM] のようにキーの記号通りに四角で囲み，キーの使用法を説明するときには [g] [MEM] のように前操作キーも付けます。またCLEAR，SET COMLP，SHOWとその角括弧の下の黄色文字の機能については，CLEAR [REG] や [f] SHOW [DEC] のようにCLEAR, SET COMPL, SHOWの記号を付けて記します。
前操作キーの後に幾つかのキーの一つを選んで押すときには，そのキー全体を中括弧で囲みました。例えば [f] [WINDOW] {0 − 7} です。

はキーの手前側の青色文字の第二機能を使いたいときには，対応する色の前操作キー（黄色文字の機能では黄色の f，青色文字の機能では青色の g）を押してからそのキーを押します(例: f XOR や g DBL+)。

前操作キーの f か g を押すと，次のキーを押すまでは f か g の文字を表示します。

前操作キーの取り消し

前操作キーは，その後に1個以上の別のキーを押さないと機能を指定できないキーです。前操作キーの一覧表は付録Bにまとめてあります。

STO や f などの前操作キーを押した後で，それを取り消したいときには f CLEAR PREFIX と押します。計算機は次の新しいキー操作を受け付ける状態になります。ただし f と g とを押し間違えたとき（g を押すときに f を押したとか，その逆操作）には直ぐ後に正しい方のキーを押すだけで訂正できます。

CLEAR操作

CLEAR操作を次の表にまとめました。レジスタをクリアするというのは，その内容を0に置き替えるということです。

クリア操作	動作の内容
f CLEAR PRGM 計算状態 プログラム入力状態 f CLEAR REG f CLEAR PREFIX	 プログラム用メモリ中の000ラインに戻る。 プログラム用メモリ全体をクリアする。 データ記憶レジスタ全体をクリアする。 キー操作途中にそれまでのキー操作を取り消す。

表示のクリア CLx と BSP

HP-16C には2種類の表示クリア用のキーがあります。 CLx (clear

X, X レジスタをクリアする) と [BSP] (back space, 1字取り消す) です。

計算状態では

- [CLx] を押すと表示だけをクリアして0にする。
- [BSP] を押すと, まだ数値入力が終ってなければ表示の右端の1個の数字だけを消します。([ENTER] を始め大部分のキーは数値入力を区切って, その次にキーインする数字を新しい数値の一部と見なします。)その後で, 消えた数字の代りに新しい数字をキーイン出来ます。数値入力を区切った後では, [BSP] は [CLx] と同じ働きです。

キー操作	表 示	
[HEX]		16進数表示。表示はそれまでの操作の結果。
1234	1234 h	数値入力を区切ってない。
[BSP]	123 h	最後に入れた数字だけを消す。
1	1231 h	
[ENTER]	1231 h	数値の入力を区切る。
[BSP]	0 h	全部の数字をクリアして0になった。
12	12 h	
[g] [CLx]	0 h	数値入力を区切ってなくても, 表示を全部クリアする。

プログラム入力状態では:

- [CLx] はプログラムの一部になり, [CLx] というプログラム命令を記憶して, 表示していた命令を消しません (削除しません)。
- [BSP] はプログラム中に入れられません。 [BSP] を押すと表示し

---

* [f] [STATUS] と押すと 2-16-0000 と表示するはずです。このように表示しないときは38ページを見てください。

てしまったプログラム命令を削除します。

単項演算機能

単項演算機能は，表示している1個の数値（Xレジスタ中の数値）だけを使って演算する機能です。この機能は数値を表示させてから押します。

キー操作	表　示
0	0 h
f NOT	FFFF h

二項演算機能と ENTER

二項演算機能は，計算機中に2個の数値が入っていなければ実行できない機能です。+，−，×，÷ は良く使う二項演算機能です。

数値入力の区切り　2個の数値を続けてキーインするときには，1個目の数値の数字入力が終ったことを計算機に知らせる必要があります。この2個の数値の区切りに ENTER キーを押します。ただし2個の数値の一方がその前の演算の結果で，既に計算機中にあれば，ENTER キーを押す必要はありません。数値入力用のキー以外の各機能は，数値の入力を区切る働きがあります。

連続計算　長い計算式の場合でも括弧を使う必要はありません。スタックに2個の数値を続けて入れるときに数値の区切りとして ENTER を使います。

例　(6 + 7) × (9 − 3) を10進数表記で計算。

キー操作	表　示	
DEC		10進数整数演算。前の結果を表示している。
6 ENTER	6 d	数値の入力を区切る。

• 数値入力用のキーは数字キーと BSP です。また実数演算状態では .，EEX，CHS も数値入力用のキーになります。

キー操作	表　示	
7 [+]	13 d	13 を中間結果として内部に記憶する。
9 [ENTER]	9 d	
3 [−]	6 d	6 も中間結果として記憶する。
[×]	78 d	(13 × 6) = 78

不揮発性メモリー

不揮発性メモリー中に記憶する情報

HP-16C は不揮発性メモリーを採用しているので，電源スイッチを切っても次の情報は消えずに記憶しています。

- 基数と演算状態（16 進数，10 進数，8 進数，2 進数の整数演算状態，または 10 進数の実数演算状態）。
- 補数の状態（1 の補数，2 の補数，符号なし）。
- ワード長。
- 計算機中に記憶している全数値。
- 計算機中に記憶している全プログラム。
- プログラム用メモリー中の位置。
- 保留しているサブルーチンのリターン情報。
- フラグのセット状態。
- 表示部分の移動（スクロール）情報。
- 桁区切り記号の種類。

計算機の電源スイッチを入れると，必ず計算状態（プログラム入力状態ではない）になります。

電池交換のために 5 分位電池を外しても不揮発性メモリーの内容は消えません。（このとき計算機の電源スイッチを切っておく必要があります。）電池交換については付録 C を見てください。

不揮発性メモリーのリセット

不揮発性メモリーをリセットする（メモリー内容を消す）には次のようにします：*

1. HP-16Cの電源スイッチを切る。
2. [ON] を押したまま [−] を押し続ける。
3. [ON] を先に放し，それから [−] を放す。（2と3の操作を [ON]/[−] と表記します。）

エラー表示　不揮発性メモリーをリセットするとPr Error（power error，電源のエラー）と表示します。どのキーでも押すとこのエラー表示が消えます。エラー表示とその原因の一覧表は付録Aにあります。

無指定状態　工場出荷後初めてHP-16Cのスイッチを入れたときや，不揮発性メモリーをリセットしたときには，次のような無指定状態になっています。

- 数値表示は16進数（整数演算状態）。
- 2の補数の状態。
- ワード長は16ビット。
- 全フラグをクリア。
- プログラム用メモリーと全レジスタもクリア。

---

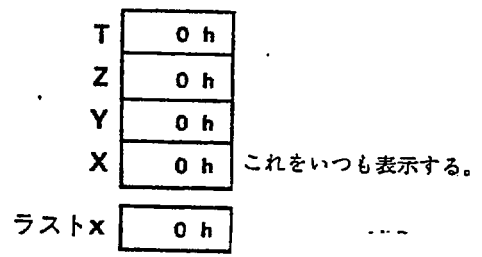
*計算機を落とすなどの誤操作をすると，リセットしなくても不揮発性メモリーの内容が全部消えてしまうことがあります。

第2章　自動メモリー・スタック

メモリー・スタックとスタック操作

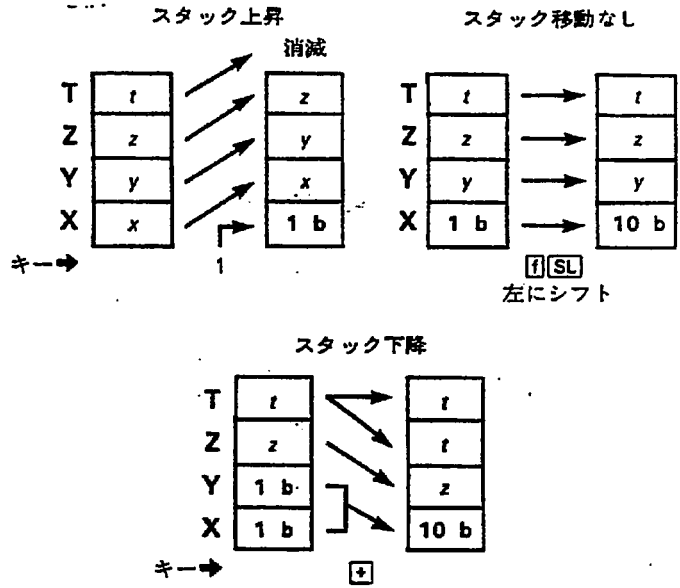
HP-16Cの計算方式（RPN方式）では複雑な計算式も括弧を使わずに解くことができ，キーを押す回数が最少ですみます。メモリー・スタック（X－Tレジスタを組み合わせたもの）と [ENTER] キーを採用していて，中間結果を自動的に記憶してすぐに使えるようにしてあるからです。この章では，プログラムまで含めてあらゆる状態でHP-16Cを使用するための基礎になるスタックの動作について説明します。

自動スタック・レジスタ



計算機がプログラム入力状態でないとき（PRGMの文字を表示してないとき）に表示しているのは，Xレジスタ中の数値です。

スタック中の数値は後入れ先出し方式で記憶します。次の三つの図は，3種類のスタック移動の説明図です。x，y，z，tはそれぞれスタック中の任意の数値で，ここでは計算機が2進数表示になっているものとします。

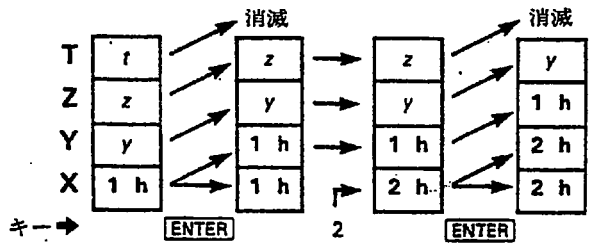


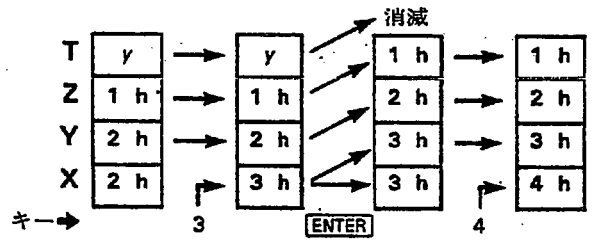
普通は，単項演算（第1章を参照してください）ではスタック移動はありませんが，二項演算ではスタック下降になります。

スタック下降時にはTレジスタの内容がZレジスタに移ると同時にTレジスタにも残るので，この数値を自動的に発生する定数として繰り返し使用することも可能です。

スタック操作

ENTER（入れる）を押すと，続けてキーインする2個の数値を区切ることができます。ENTERを押すとスタックが上昇し，表示（Xレジスタ）している数値をYレジスタにもコピーするからです。その後に入力する値がXレジスタにあった値を書き替え，そのときにはスタック上昇はありません。次例はスタックに16進数の1，2，3，4が入る様子を示したものです。（淡網の部分は，そのレジスタの内容は次にキーインしたり呼び出した数値によって書き替わることを表します。）





スタック移動のキーには [ENTER] 以外に次の3種類があります。

- [R↓] (roll down, 下方に回転移動) ——スタック・レジスタの内容が1レジスタずつ下降します。Xレジスタの内容はTレジスタに移ります。
- [R↑] (roll up, 上方に回転移動) ——スタックの内容が1レジスタずつ上昇します。Tレジスタの内容はXレジスタに移ります。
- [x≷y] (X exchange Y, XとYの入れ替え) —XとYレジスタ内の数値を入れ替えます。

ラストXレジスタ

上記のレジスタ以外にもう一つラストxレジスタがあって，数値演算の実行直前に表示していた数値を記憶します。* [9] [LSTx] (last x, 計算直前のx) キーを押すと，ラストxレジスタの内容を表示 (Xレジスタ) にコピーします。

---

*演算のときにxをラストxに記憶する演算の種類の一覧表は付録Bにあります。

[LSTx]機能を使うと，定数をもう一度キーインしないで計算に使うことが出来ます（27ページの定数を使用した計算を見てください）。また，最後の演算直前に表示していた数値を呼び戻せるので，演算に誤りがあっても元通りに訂正できます。

例えば，長い計算の途中で11を足すときに誤って10を足してしまったときには，次のように訂正します。

キー操作	表　示	
[BIN]		2進数表示。それまでの操作の結果。
1010 [ENTER]	1010 b	
10 [+]	1100 b	間違ってキーインした数を使ってしまった！
[9] [LSTx]	10 b	[+]を押す前にXレジスタにあった数値（誤ってキーインした値）をラストxレジスタから呼び戻す。
[−]	1010 b	[+]を押したのでその逆の[−]を押して値を元に戻す。
11 [+]	1101 b	正しい答。

数値演算とスタック

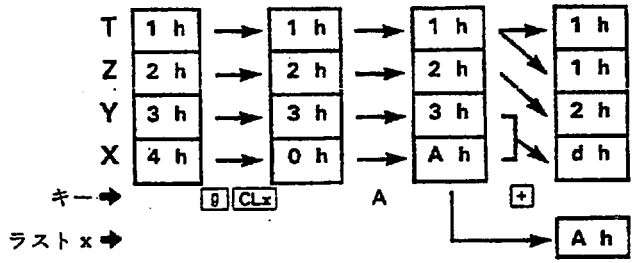
スタックの移動

2個の数値を続けてキーインしたいときは，1個目の数値の後で[ENTER]を押す必要があります。しかし何か機能を実行した（[R↓]のようなスタック操作も含む）直後に新しい数値をキーインするときには，キーインする前に[ENTER]を押す必要はありません。HP-16Cの大部分の機能を実行したときには，本来の機能の他に次の動作も同時に行なっています。

- 自動メモリー・スタックの上昇を可能にする——つまり，次の数値をキーインしたり記憶レジスタから表示に呼び戻したりすると，スタック内容が自動的に上昇するようになります。

- 数値入力を区切る。次にキーインする数字を別の数値と見なします。

ただし例外として，ENTER と CLx はスタック上昇が可能になりません。つまり，このキーを押した後に次の数値をキーインしたり呼び戻したりしたときはスタック上昇が発生しません。それまで表示していた数値を新しい数値に書き替えるだけです。(ENTER キーは，押したときにスタック内容が上昇するのであって，次の数値をキーインしたり呼び戻したときにはスタック上昇はありません。)



図示のように，X レジスタと Y レジスタ中にあるデータ（A_{16} と 3_{16}）で数値演算をすると，スタック内容が下降して演算の答（D_{16}）が X レジスタに入ります。

各機能を実行したときのスタック上昇の影響（スタック上昇が可能か，可能でないか，あるいは中立的なのか）と数値入力についての詳細な一覧表は付録 B にあります。

括弧がある計算

自動的にスタック内容が上下するので，括弧がある複雑な計算も括弧を使わずに出来ます。中間結果を自動的にスタックに記憶し，必要に応じて使用できます。どんな複雑な計算も一つ一つに分解すると，単項演算や二項演算を連続したものになるので簡単に解けます。紙と鉛

• 数値の入力が既に終了している（区切ってある）ときには，BSP は表示をクリアしてスタック上昇が可能にはなりません。数値入力の途中ではスタック上昇には中立的なキーです。

筆で普通に計算するときのように，内側の括弧から外側の括弧に向かって順番に計算すれば，中間結果をわざわざ記憶レジスタ中に記憶する必要は滅多にありません。

例えば整数演算状態の10進数表示で3[4+5(6+7)]の計算をしてみましょう。

キー操作	表　示	
DEC		表示はそれまでの操作の結果。
6 ENTER 7 +	13 d	中間結果。
5 ×	65 d	中間結果。
4 +	69 d	中間結果。
3 ×	207 d	最終結果。

この例を見て分かるように，二項演算後はスタックが自動的に下降し，その後で新しい数値をキーインすると自動的にスタックが上昇します。

定数計算

定数を使用した連続計算をするには記憶レジスタを使う方法以外に次の2通りの方法があります。

- ラストxレジスタを使う。
- 計算前にスタック全体に定数をキーインする。

例　3個の8ビット2進数，10001001，10101111，11110101から上位の4ビットを取り除いて下位の4ビットだけにしましょう。定数1111をマスクとして使います。

ラストxレジスタを使う方法　定数をYレジスタではなくXレジスタに入れて計算します。定数がラストxレジスタに入るようにするためです。その定数は g LSTx を押すと呼び出せます。

キー操作	表　示	
BIN		2進数表示。表示はそれまでの操作の結果。
10001001 ENTER	10001001 b	最初の値。
1111	1111 b	マスク（定数）。

キー操作	表 示	
f AND	1001 b	下位の4ビット。
10101111	10101111 b	2番目の値。
g LSTx	1111 b	定数を呼び出す。
f AND	1111 b	下位の4ビット。
11110101 g LSTx	1111 b	
f AND	101 b	下位の4ビット。

スタックを使う方法　定数をキーインしてから ENTER を3回押し，スタック中の各レジスタに入れます。演算（ここでは AND）のたびにスタック内容が下降し（Yレジスタ内にいつも定数が入ります），またTレジスタにも定数が残ります。 BSP または CLx を押してから次の数値をキーインするとスタックが上昇しないので，新しい数値がその前の結果を書き替えて，Yより上のレジスタには定数だけが残ります。

キー操作	表 示	
1111 ENTER	1111 b	マスク（定数）。
ENTER ENTER	1111 b	スタックの全レジスタに1111を入れる。
10001001 f AND	1001 b	最初の値の下位4ビット。
BSP	0 b	スタック上昇が可能ではない。
10101111 f AND	1111 b	2番目の値の下位4ビット。
BSP	0 b	スタック上昇が可能ではない。
11110101 f AND	101 b	3番目の値の下位4ビット。

第3章　数値と表示の形式と指定

HP-16C の数値表記法は，他の計算機に比べて非常に多様性をもっています。この章では整数の指定と表示について色々な面から説明します。数値表記の基数，ワード長，補数，数値の範囲，結果の表示，等が主な内容です。（実数については第5章　実数で説明します。）指定した形式は不揮発性メモリー中に記憶しています。

整数演算状態

数値表示の基数（ [HEX]，[DEC]，[OCT]，[BIN] ）は整数演算だけに有効です。小数部分のある10進数は第5章で説明する実数演算状態で計算します。上記の4個の基数キーのどれかを押すと整数演算状態になります。

基数の指定

HP-16C の整数演算状態では，表示や数値入力に使う基数は HEX（16進数），DEC（10進数），OCT（8進数），BIN（2進数）の4種類です。そのときの基数は，8桁の表示の右側にそれぞれ **h**，**d**，**o**，**b** の文字で示します。工場出荷後始めて HP-16C のスイッチを入れたとき，または不揮発性メモリーをリセットしたときには自動的に HEX（16進）になっています。

[HEX]，[DEC]，[OCT]，[BIN] のどれかのキーを押すと，表示はそのキーに対応する基数の表示になり右揃えの整数表示に変わります。数字キーを押すとそのときの基数に従って解釈しますが，基数に合わない数字をキーインする（例えば2進数になっているときに3を押す）と HP-16C は何の反応もしません。16進数では [0]～[9] の数字キー以外に [A]～[F] のキーも使用し，その表示はそれぞれ **A**, **b**, **C**, **d**, **E**, **F** になります。

注 そのときの基数表示とは無関係に，計算機内部では常に2進数で表現しています。基数のキーを押して基数を切り替えても表示が変わるだけで，計算機の内部表現は変わりません。

一時的な表示(SHOW)

表示している数値を一時的に他の基数で見たいときには，[f] SHOW | [HEX]，[DEC]，[OCT]，[BIN] | と押します。基数のキーを押し続けている間は，その基数で数値を表示します。

キー操作	表　示
[HEX] F	F h
[BIN]	1111 b
[f] SHOW [OCT] (押したまま)	17 o
(放す)	1111 b

補数の状態と符号なしの状態

HP-16C は1の補数，2の補数，符号なしの3種類の数値表現ができます。工場出荷後に計算機のスイッチを始めて入れたときや不揮発性メモリーをリセットしたときには自動的に2の補数の状態になっています。一度指定した状態は，キー操作（またはプログラム中の命令）で変更するか不揮発性メモリーをリセットするまでそのまま有効です。（この本では，特に明記したとき以外は2の補数状態で使用します。）

符号付きの数値を2進数で表わすときにワード長の一番左のビット（最上位のビット，MSB）は符号ビットです（正の時0，負の時1）。10進数では負の値は－符号付きで表示します。

---

*36ページのキー操作例で，数値表示の基数，ワード長，補数状態の指定によって表示が変わっても整数演算状態の内部の2進数表現が変化しないことが分かると思います。

1の補数状態

[f] SET COMPL [1's] と押すと，1の補数状態になります。1の補数状態のときに [CHS] (change sign, 符号変更) を押すと，Xレジスタ中の数値の1の補数になって全ビットが逆になります。

1の補数では正の数値表現と負の数値表現が同じ個数だけあり，0には0と－0の二つの表現があります。

2の補数状態

[f] SET COMPL [2's] と押すと，2の補数状態になります。2の補数状態のときに [CHS] を押すと表示していた数値の2の補数（Xレジスタ中の全ビットを逆にして1を加えたもの）になります。

2の補数では0には一つの表現しかありませんが，正の数値表現より負の数値表現の方が1個多くなります。

符号なしの状態

[f] SET COMPL [UNSGN] と押すと，符号なしの状態になり，符号ビットを使わなくなります。最上位のビットも符号ではなくて数値を表すので，8ビット長で表現できる最大値は 127_{10} から 255_{10} に変わります。

符号なしの状態で符号を変えても意味がありません。符号なしの状態で [CHS] を押すと，Xレジスタ中の数値の2の補数になります。それと同時にフラグ5 (Gの文字表示) をセットし，本当の結果は負の値でそれは符号なしの状態の範囲外であることを示します。

次表は，(ワード長が4) で表現できる4ビットの全数値が3種類の補数状態によって10進数の何に相当するかを示したものです。

4ビット2進数に対応する10進数

2進数	1の補数 状態	2の補数 状態	符号なしの 状態
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	−0	−1	15
1110	−1	−2	14
1101	−2	−3	13
1100	−3	−4	12
1011	−4	−5	11
1010	−5	−6	10
1001	−6	−7	9
1000	−7	−8	8

ワード長と表示部分

HP-16Cはデータの単位としてワードを使っていて，1ワード長を64ビットまで自由に指定できます。工場出荷後始めてHP-16Cのスイッチを入れたときや不揮発性メモリーをリセットしたときには，ワード長は自動的に16ビットになります。計算機の表示部には1度に8桁まで表示し，左側の不要の0は表示しません*。表示部にそのとき表示

*37ページの説明のように，フラグ3をセットすると左側の不要の0も全部表示するようになります。

している桁の左または右にまだ表示できない桁があると，**h**, **d**, **o**, **b** の左か右，または両方に・を表示してそれを知らせます。

ワード長

ワード長を指定するには，まずXレジスタ中に指定したい値（1_{10}～64_{10}）をキーインしてから [f] [WSIZE] (word size, ワード長）と押します。その絶対値をワード長として使い，0は64と見なします。[WSIZE] を実行するとスタックが下降します。

ワード長が7ビット以下になっていると，新しくワード長を指定しようとすると入力できる値に制限を受けます。そのようなときには0 [f] [WSIZE] と押せばワード長が64になるので，それから希望するワード長を指定します。65以上のワード長を指定しようとすると，**Error 2** を表示します。

キー操作	表　示	
[DEC] 16 [f] [WSIZE]		10進数;ワード長は16。
[f] SET COMPL [2's]		2の補数状態。
32767 [ENTER]	32767 d	ワード長16で2の補数のときに表示できる最大の正の値。
8 [f] [WSIZE]	-1 d	0111111111111111_2 (16ビット) から $1111\ 1111_2$ (8ビット) に変わる。
16 [f] [WSIZE]	255 d	11111111_2 から $0000\ 0000\ 11111111_2$ に変わる。

注　ワード長を変えると，メモリー・スタックに記憶している値が変化することがあります。ワード長を短くすると切り捨てが発生し

---

*1または2の補数状態で，そのときのワード長で表せる最大の正の値より大きい値を入力しようとすると，負の値になることがあります。それは最上位のビット（符号ビット）が1になるからで，37ページのキー操作例の最後がこれです。ワード長が3以下では，最初に入力した数字がそのときの基数では正しいものでもそのワード長での最大値より大きいと0を入力したのと同じになります。

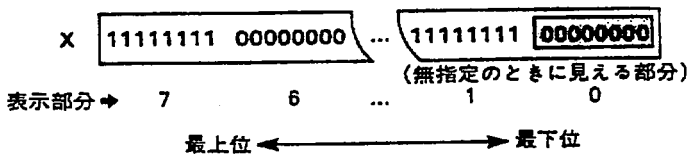
て，下位のビットだけが残ります。ワード長を長くすると，負の数値の符号ビットが符号でなくなります。ワード長を一たん短くしてから元の長さに戻しても，元のスタック内容には戻りません。(記憶レジスタへの影響はこれとは異なり 71 ページで説明します。)

表示部分

表示部は X レジスタ中の数値の内の 8 桁を見せる窓 (window) と考えることができます。他のレジスタと同様に X レジスタは，ワード長に従って最大 64 桁の 2 進数を記憶できます。普通見えるのは表示部分 0 で，X レジスタ中の数値の最下位の 8 桁です。9 桁以上の数字をキーインすると，最上位の桁は表示の左側に移動して消え，表示部分 1 に移ります。

f WINDOW |0 ～ 7| と押すと，X レジスタ中のワードの別の 8 桁部分を見ることが出来ます。X レジスタに新しい数値を入れるたびに，表示は表示部分 0 (ワードの最下位の 8 桁) に戻ります。ワード長の最大が 64 なので表示部分の番号は 0 から 7 までです。(ワード長が短いときや数値が小さいときには，上位の表示部分は空白になります。) 8 以上の表示部分 (WINDOW) を指定すると Error 1 を表示します。

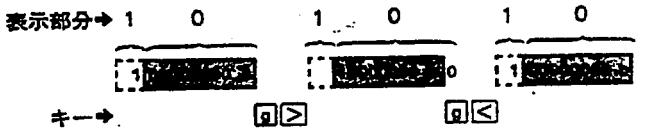
例　16 桁の 16 進数 FF00 FF00 FF00 FF00 を 2 進数で表わすと 64 桁なります (8 個の 1 と 8 個の 0 を交互に並べたものです)。2 進数では，f WINDOW 0 から f WINDOW 7 まで順に押するとその全体を見ることができます。



表示の移動

< か > 機能を使うと，表示している数値を 1 桁ずつ左右に移動することが出来ます。これは数値そのものを変えるのではなく，見える数字を動かすだけです。

表示をどちらに動かすとXレジスタ中の数値の残りの部分が見られるかは，基数表示（h, d, o, b）の左右の・で分かります。例えば基数の表示の左側に・があると（・b）表示している部分より左側にまだ桁があるので，[g][>]と押して数値を右方向に移動すると見えなかった桁が見えます。表示している部分が数値の中間であると，基数表示の両側に・を表示します。



例　[>]と[WINDOW]を使ってXレジスタの内容全体を見てみましょう。ワード長は16ビットです。

キー操作	表　示	
[BIN]		2進数表示にする。
11111111 [ENTER]	11111111 b	表示がいっぱいになった（8桁）。
1 [+]	00000000 .b	左側に・があるので数値は左側に続いている。
[g][>]	10000000 b.	数値を1桁右方向に移動する（右側に・が移って数値が右側に続くことを示す）。
[f][WINDOW] 1	1 b.	表示部分1の内容：最上位の桁。
[f][WINDOW] 0	00000000 .b	表示部分0の内容：最下位の桁。

[>]，[<]による表示部分の移動は，ビット操作または数値演算を実行すると元に戻って表示部分0を表示します。表示部分0に戻らない機能の一覧表は付録Bにあります。

表示と HP-16C の内部表現

次のキー操作は各種の機能（基数の指定，ワード長指定，補数の状態指定）を実行したときに表示と内部の2進数表現がどう変わるかを説明するためのものです。

キー操作	表　示	内部の2進数表現
[HEX]		
8 [f] [WSIZE]		
[BSP]	0 h	00000000
62	62 h	01100010
[OCT]	142 o	01100010
[BIN]	1100010 b	01100010
[DEC]	98 d	01100010
62	62 d	00111110
[OCT]	76 o	00111110
62	62 o	00110010
[HEX]	32 h	00110010
[f] SET COMPL [2's]	32 h	00110010
[CHS]	CE h	11001110
		ワード長が8で2の補数のときは負の値。
[OCT]	316 o	11001110
[BIN]	11001110 b	11001110
[DEC]	-50 d	11001110
[f] SET COMPL [1's]	-49 d	11001110
		内部表現は変化しない。
[f] SET COMPL [UNSGN]	206 d	11001110
[f] SET COMPL [1's]	-49 d	11001110
		1の補数状態ではこれを負の値と解訳する。
[CHS]	49 d	00110001

キー操作	表　示	内部の2進数表現
2	2 d	00000010
5	25 d	00011001
4	-001 d	11111110
		1の補数状態では-1_{10}に対応。0は数値入力の訂正用に表示している。
f SET COMPL UNSGN	254 d	11111110

フ　ラ　グ

HP-16Cにはプログラム実行の制御に使用できる3個のユーザ用フラグ（0, 1, 2）と，HP-16Cの状態を知らせるための3個のシステム用フラグ（3, 4, 5）があります。

プログラム中でフラグを使用する方法については，第9章　プログラムのジャンプと制御で説明します。

フラグ3, 4, 5の3個は単にHP-16Cの状態を知らせる役割を持っているだけで，HP-16Cの操作には関係ありません（ただしこれをプログラム実行の制御用に使用するときを除きます）。

- フラグ3——有効数字左側の0を表示するかしないかに関係します。これをセットしておくと，有効数字よりも左側の0（これを左側の0と呼ぶ）も表示します。クリアしておくと左側の0を表示しません。無指定状態ではクリアしてあります。（10進数の整数演算状態と実数演算状態では左側の0はいつも表示しないので注意してください。）
- フラグ4——キャリー（carry，繰り上がり）が発生したときにセットします（セットするとCの文字を表示します。）
- フラグ5——求めた値が範囲外（out-of-range，つまり表現できる最大値より大きい，またはそのときの状態では表現できない）のときにセットしCの文字を表示します。

キャリーと範囲外がどういうときに発生するかは第4章で説明します。

各フラグのクリア，セット，および判定は次のようにします*:

- [g] [SF] n——フラグn (0~5) をセットする。
- [g] [CF] n——フラグnをクリアする。
- [g] [F?] n——フラグnをセットしてあるかどうか判定する。

フラグの状態とそれに対応する文字（があれば）は，次のどれかの方法で変更するまではそのまま変わりません。

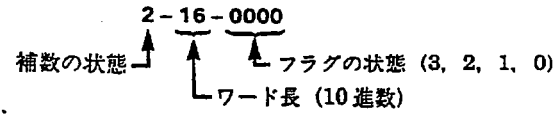
- 不揮発性メモリーをリセットする（メモリー内容を消す）。
- フラグに影響のある機能を実行する（フラグ4と5だけ）。
- [CF] でフラグをクリアまたは [SF] でフラグをセットする。

プログラム中でフラグを使用するのは，後でフラグの状態を判定できるように記録しておくためです。第9章で，フラグをジャンプに使う方法を説明します。

HP-16C の状態確認

[f] [STATUS] を押すと，一時的に次の三つの情報を表示します。(1)そのときの補数の状態，(2)その時のワード長，(3)フラグ4と5以外のどのフラグをセットしてあるか（フラグ4と5をセットするとCとGの文字を表示します）。この情報は [STATUS] キーを押している間は表示し続けます。状態を変更する方法は，30ページ（補数の状態），32ページ（ワード長），37ページ（フラグ）を見てください。

最初に [STATUS] を押したときの表示†



---

* フラグ番号は10進数で押してください。フラグ番号はスタックには入りません。

† 工場出荷後にHP-16Cのスイッチを初めて入れたときや不揮発性メモリーをリセットした後のことです。

補数の状態表示　補数の状態によって0（符号なし），1（1の補数），2（2の補数）のいずれか。

フラグの状態表示　フラグの状態表示位置に0または1を使った4桁の数字を表示します。この各桁は右から順にフラグ0，1，2，3に相当し，1を表示している桁のフラグをセットしてあります。例えば STATUS 表示の次のようなフラグの状態表示を考えてみましょう。

フラグ3210
−0100　　フラグ2をセットしてある。

フラグ3210
−1101　　フラグ0，2，3をセットしてある。

特殊な表示

状態表示記号

HP-16Cには6種類の状態表示記号があって，それぞれHP-16Cの各種状態を表します。各状態表示の意味と使用法については，それぞれのページで説明します。

*	電圧が低下。40ページ。
f と g	前操作キー（f または g）を押した。17ページ。
C	フラグ4（キャリー）をセット。41ページ。
G	フラグ5（範囲外）をセット。42ページ。
PRGM	プログラム入力状態。76ページ。

エラー表示

誤った操作（例えばワード長に64より大きい値を指定）をするとエラー表示になります。エラー表示とその原因の一覧表は付録Aにあります。

Error 表示を消すには，任意のキー1個を押すとエラー発生直前の状態に戻ります。その後は普通に操作出来ます。

電圧低下の表示

表示の左下隅に＊の記号が点滅し始めたら電池の電圧が低下しています。この時点ではプログラムを連続実行すると 15 分以上，手操作で 1 時間以上は使えるでしょう（電池によってはもっと使えることもあります。）電池の交換方法については付録 C（107 ページ）を見てください。

第４章　数学演算とビット操作

整数の数学演算とビット操作は，整数演算状態のときだけ出来ます。この操作をするとキャリーと範囲外の状態が発生することがあるので，操作の説明前にキャリーと範囲外の状態について説明します。

実数演算状態での数学演算やその他の機能については第５章実数で説明します。

キャリーと範囲外の状態

数学演算やビット操作の中にはそれを実行した時にキャリーと範囲外の状態（片方または両方）が発生するものがあります。この状態が発生するとそのフラグをセットし（後で状態判定に使用できる），同時に目で確認できるように文字を表示します。キャリーと範囲外の定義は実行する機能によって異なります。

フラグをキー操作でセットしたりクリアしたりする方法は第３章のフラグ（38ページ）で説明しました。

フラグ４　キャリー（C）

整数演算状態で下記のビットのシフト，回転，数学演算を実行すると，フラグ４とＣの文字をセットするかクリアします。フラグ４はキャリーフラグで，キャリービットが１のときにセットし，キャリービットが０のときにクリアします。

SL	RL	RLn	＋（キャリー，繰り上がり）
SR	RLC	RLCn	－（ボロー，借り）
ASR	RR	RRn	÷（余りが≠０）
	RRC	RRCn	DBL÷（余りが≠０）
			√x（余りが≠０）

（各機能についてはこの章で後に説明します。）

＊付録Ａにこの状態に関連のある機能と，それがキャリーフラグと範囲外フラグにどう影響するかの一覧表があります。

例 次の簡単な足し算でキャリーフラグ (4) をセットし，次にクリアしてみましょう。

キー操作	表 示	([STATUS]: 2-16-0000)*
[HEX] FFFF [ENTER]	FFFF h	16進数表示。
1 [+]	0 h	Cを表示。キャリーが発生したのでフラグ4をセットした。
1 [+]	1 h	キャリーが発生しないのでキャリーフラグをクリアした。

フラグ5 範囲外 (G)

そのときのワード長や補数状態では演算の結果を正しく表すことが出来ないときにフラグ5とGの文字をセットします。[+]と[−]の演算ではこれは普通のコンピュータの“オーバーフロー”に相当します。

整数演算状態で下記の各機能を実行すると，フラグ5とGの文字をセットするかクリアします。

[+] [−] [×] [÷] [ABS] [CHS]
[DBLX]† [DBL÷]†

四則演算の[+]，[−]，[×]，[÷]は実数演算状態でもフラグ5に影響することがあります。[FLOAT]機能もフラグ5に影響します。詳細については第5章を見てください。

結果が範囲外になると，そのときのワード長に収まる範囲の下位のビットだけを表示します。1または2の補数状態での[×]または[+]では，答の最上位のビット（符号ビット）は正しい結果の符号ビットに一致します。

---

* この本でこのように最初に状態表示を明示したのは，HP-16Cの状態（38ページ参照）をそのようにしてないと，その例の通りにならないからです。
† 常にフラグ5をクリアします。

キー操作	表　示	(STATUS: 2-16-0000)
DEC 32767 ENTER	32767 d	
2 ×	32766 d	G を表示しフラグ5をセットする，オーバーフロー。左端の桁が0なので正数。
g CF 5	32766 d	フラグ5をクリア。

フラグ5をプログラム走行中にセットすることがありますが，そのためにプログラム走行が停止することはありません。

数学演算

加減乗除算

+，−，×，÷ の整数の加減乗除算は4種の基数のどれを使っても実行できます。演算データは異なる基数で入力しても良いけれども，必ずXとYレジスタに入れておく必要があります。演算が終るとスタック下降し，演算の結果がXレジスタに入ります。

整数演算状態では，÷は整数の除算を行ない，商の小数部分は切り捨てます。

×以外の演算を実行すると，フラグ4とフラグ5を必ずセットまたはクリアします。×はフラグ5だけに影響します。

例　$(5A0_{16}) \div (177764_8)$ の計算。

キー操作	表　示	(STATUS: 2-16-0000)
HEX 5A0 ENTER	5A0 h	1番目の値を入力。
OCT 177764	177764 o	8進数表示に変更し2番目の値を入力。
÷	177610 o	結果は8進数。キャリーが発生しなかったので，答は正確。
HEX	FF88 h	16進数に変換。

1の補数状態の加減算 2の補数状態と符号なしの状態では，加算や減算の結果は単にXとYレジスタの二つのビット・パターンの合計または差になります。しかし1の補数状態では，加算の結果はキャリーの有無によって影響を受け，減算の結果はボローの有無によって影響を受けます。最上位のビットでキャリーがあると，ビット・パターンの合計よりも1だけ多くなります。また最上位のビットにボローがあると，ビット・パターンの差よりも1だけ少なくなります。どちらの場合もフラグ4をセットします。

(STATUS: 1-04-1000)

キャリーがある

```
                 111
   -1           1110
 +(-1)         +1110
 ------        -----
  -2₁₀          1100
               +   1
               -----
               1101₂
```

キャリーがない

```
   -3           1100
 +  3          +0011
 ------        -----
  -0₁₀         1111₂
```

ボローがある

```
                 1
    3           Ø011
   -4          -0100
 ------        -----
  -1₁₀          1111
               -   1
               -----
               1110₂
```

ボローがない

```
                  0
    6           01X0
   -5          -0101
 ------        -----
   1₁₀         0001₂
```

加算のときのキャリーフラグ 2進数の加算のときに最上位のビットからさらに上の桁へ繰り上がりがあるとキャリーフラグ（フラグ4とCの表示）をセットします。加算のときにこのような最上位ビットでの繰り上がりがなければ，キャリー・フラグをクリアします。これはどの補数状態でも同じです。

(STATUS: 2-04-1000)

キャリーをセット

```
                 ¹1010
   -6            +1100
 +(-4)           -----
 ------          0110₂
   6₁₀
```

（範囲外フラグもセットしたので答は正しくない）

キャリーをクリア

```
    6             0110
  + 1            +0001
 ------          -----
   7₁₀           0111₂
```

減算のときのキャリーフラグ　2進数の減算のときに最上位のビットでさらに上位の桁からのボローがあるとキャリー・フラグ（フラグ4とCの表示）をセットします。最上位ビットにボローがなければ，キャリーフラグをクリアします。これはどの補数状態でも同じです。(HP-16Cの減算は負数の加算として計算するのではないため，キャリーの発生に影響するので注意が必要です。)

([STATUS]: 2-04-1000)

キャリー・フラグをセット		キャリー・フラグをクリア	
	0,		0,
-6	1010	6	0110
-(-4)	−1100	−1	−0001
-2_{10}	1110_2	5_{10}	0101_2

範囲外フラグ　演算の結果がそのときのワード長または補数状態で表せなくなると，範囲外フラグをセットします。[+]では，2の補数状態でそのときに表現できる最大の負数を −1 で割った時にだけ発生します。

例　ワード長が4ビットのとき，2進数で (7+6) を計算してフラグ4と5の状態を調べてみましょう。

キー操作	表　示	([STATUS]:2-04-0000)
[BIN]		2進数演算。
111 [ENTER]	111 b	−3。フラグ5(範囲外)
110	110 b	をセットし，フラグ4
[+]	1101 b	(キャリー)をクリアする。

除算の余り　[RMD]

除算では結果の整数部分だけがXレジスタに入ります。余りが0でないときはフラグ4（キャリー）をセットしCの文字を表示します。余りが0ならフラグ4をクリアします。

商ではなくて余りの方を求めたいときには，[+] の代りに [f] [RMD] (remainder，余り) と押します。この操作では |y| MOD |x| を計算します。結果の符号は被除数の符号（yの符号）と同じになります。

キー操作	表 示	([STATUS]: 2-16-0000)
[HEX] 66 [ENTER]	66 h	16進数表示。
7 [÷]	-E h	66/7は余りがあるのでCの文字を表示しフラグ4をセットする。
2 [÷]	7 h	Cの文字が消えたのでフラグ4をクリアした。E/2は余りがない。
4 [f] [RMD]	3 h	7/4の余り。

平 方 根

[√x] はXレジスタ中の数値の平方根を求めます。平方根の小数部分は切り捨ててしまいます。小数部分が0でなければフラグ4（キャリーフラグ）をセットし，0のときはフラグ4をクリアします。

負数と補数

符号の変換 [CHS] (change sign, 符号を変える) を使うと，Xレジスタ中の数値の（1または2の）補数になって符号が変わります。この補数状態でXレジスタ中の数値がそのとき表現できる最大の負数のときに [CHS] を押すとフラグ5（範囲外フラグ）をセットするだけで値は変化しません。

符号なしの状態で [CHS] を押すと，2の補数になってフラグ5（Gの文字）をセットしますが，これは符号なしの状態では負数は範囲外になるからです。

負数をキーインしたいときには，数字の全桁をキーインしてから [CHS] を押します。整数演算状態では [CHS] を押すと数値の入力を区切ったことになります。

絶対値 [g] [ABS] (absolute value, 絶対値) を押すと，Xレジスタ中の数値が負数のときは絶対値（それの1または2の補数）を求めます。符号なしの状態のときと，Xレジスタ中の数値が正数のときには変化しません。

2の補数状態でXレジスタ中の数値がそのとき表現できる最大の負数のときに [ABS] を使用するとフラグ5（範囲外フラグ）をセットするだけで値は変化しません。

論理演算

論理演算（ブール代数）のNOT, OR, AND, 排他的ORを実行すると，2進数のビットごとの分析結果を求められます。[OR]，[AND]，[XOR] の各機能を実行すると，XとYレジスタ中のワードの対応するビットごとに演算を実行し，スタック下降して結果がXレジスタに入ります。[NOT] はXレジスタ中のワードについて実行し，スタック下降はありません。

NOT

[NOT] を実行するとXレジスタ中の2進数の全ビットを逆にした値になります。これは1の補数状態で [CHS] を使って元の数の1の補数を求めたのと同じことです。Xレジスタ中の数値だけが変化します。

キー操作	表　示	([STATUS]:2-16-1000)
[BIN] 11111111	11111111 b	2進数状態。
[f] [NOT]	00000000.b	0000000011111111_2
[f] [WINDOW] 1	11111111 b.	の1の補数は 111111 1100000000_2。

AND

[AND] は2個のワードの対応するビットごとにAND（論理積）を求めます。対応するビットが両方とも1の時だけ，結果のそのビット位置が1になります。片方または両方が0のときは結果のそのビット位置が0になります。

[AND] の使用例は54ページのマスクにあります。

OR

[OR] は2個のワードの対応するビットごとにOR（論理和）を求めます。対応するビットが両方とも0のときだけ，結果のそのビット位置が0になります。片方または両方が1のときは結果のそのビット位置が1になります。

例 10101_2 と 10011_2 について OR を実行して，両方とも 0 のビットを調べましょう。

キー操作	表 示	(STATUS: 2-16-0000)
10101 ENTER	10101 b	
10011	10011 b	
f OR	10111 b	両方のビット 3（結果のビットが 0 になっている）とビット 4 よりも左側の全ビットが 0。

排他的 OR

XOR は 2 個のワードの対応するビットごとに排他的 OR (EXCLUSIVE OR，論理差）を求めます。対応するビットが互いに異なる時だけ，結果のそのビット位置が 1 になります。それ以外のビット位置は 0 になります。

例 2 個の 2 進数（01010101_2 と 01011101_2）について XOR を実行して両方の値が等しいかどうか調べましょう。結果に 1 が含まれていればそのビット位置で 2 数が異なっていることが分かります。

キー操作	表 示	(STATUS: 2-16-0000)
1010101 ENTER	1010101 b	
1011101	1011101 b	
f XOR	1000 b	2 個の値は右側から 4 番目のビットが違う。

ビットのシフトと回転

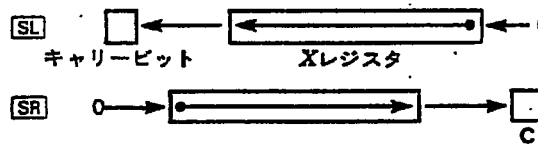
ビットのシフトと回転はワード中の各ビットを右また左に順に移動させる機能です。移動の結果ワードの端から出たビットがどうなるか，また反対側の端に新しく入るビットの値が何か，は実行するシフトまたは回転の種類によって異なります。

LJ を除いて，どのシフトまたは回転を実行してもそれぞれの図のように必ずフラグ 4（キャリー）をセットまたはクリアします。

ビットのシフト

HP-16C はXレジスタ中の値について2種類のシフトを実行できます。それは普通のシフト（ SL と SR ）と符号ビットの残るシフト（ ASR ）です。またXレジスタ内容の左揃え（ LJ ）も出来ます。

普通のシフト　f SL (shift left, 左にシフト) または f SR (shift right, 右にシフト) を押すと，Xレジスタ中のワードの全ビットが1ビットずつ左または右に移動します。ワードの端から出たビットはキャリー (C) ビットに入り，それまでのキャリービットの状態は消失します。ワードの反対側の端に入る新しいビットは常に0です。



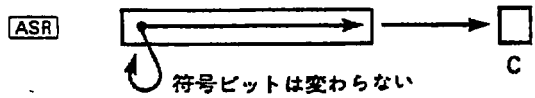
左揃え　ビット・パターンをそのときのワード長内で左揃えするには g LJ (left-justify) を押します。スタックは上昇して左揃えされたワードがYレジスタに入り，カウント数（左揃えするために必要なビットの移動数）がXレジスタに入ります。 LJ ではキャリー・フラグに影響はありません。

例　2進数 1111 をワード長8で左揃えしてみましょう。

キー操作	表　示	(STATUS: 2-08-0000)
1111	1111 b	
g LJ	100 b	カウント数，ワードの左揃えに4ビットシフトした。
R↓	11110000 b	左揃えしたワード。

数学的右シフト　g ASR (arithmetic shift right) を押すと，Xレジスタ中のワードが SR のときと同様に1ビットずつ右に移動します。ただしワードの左端の新しいビットは0ではなく，符号ビットのままです。（符号なしの状態では符号ビットがないので， ASR は SR と同じ働きです。）シフトの結果右端から出たビットが1の時にはキ

キャリー (C) ビットをセットし，0のときにはクリアします。



例　正の2進数を右にn桁シフトする操作は，その数を 2^n で割ったのと同じ意味です。数学的右シフトでは符号ビットがそのまま変わらないので，偶数の負の整数を2で割る計算にも使えます*。01111111（ワード長8）を 2^3 で割る計算と，10000000を 2^3 で割る計算をやってみましょう。

キー操作	表　示	（STATUS: 2-08-0000）
g SF 3		左側の0も表示させる。
1111111	01111111 b	
f SHOW DEC	127 d	この操作で数値入力を区切るので ENTER は不要。
f SR f SR f SR	00001111 b	1回のシフトごとに整数を2で割り，キャリービットに1が入るのでフラグ4をセットする。
f SHOW DEC（押したまま）	15 d	
（放す）	00001111 b	
10000000	10000000 b	
f SHOW DEC	-128 d	
g ASR g ASR	11100000 b	
g ASR	11110000 b	1回のシフトごとに符号ビットはそのまま残り，キャリーフラグをクリアする。
f SHOW DEC	-16 d	
（放す）	11110000 b	

---

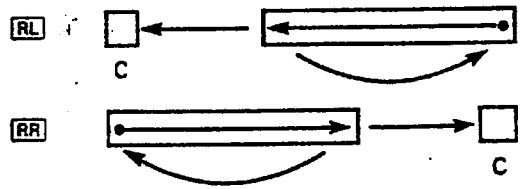
*2の補数状態で奇数の負の整数のときに，ASR を使うとそれよりも1小さい数を2で割ったのと同じになります。

ビットの回転

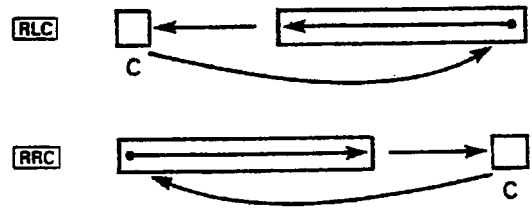
HP-16Cには3種類，全部で8種のビット回転機能があります。

- 左方向または右方向の回転（ RL ， RR ）
- キャリービットも含めた左方向または右方向の回転（ RLC ， RRC ）
- n桁の回転（ RLn ， RRn ， RLCn ， RRCn ）

回転　f RL (rotate left, 左回転) または f RR (rotate right, 右回転) を押すと，レジスタ中のワードが1ビットずつ左または右方向に回転します。ワードの端から出たビットは反対端のビットに入ります。反対側に入ったビットが1の時にはキャリーフラグをセットし，0の時にはクリアします。



キャリービットも含めた回転　g RLC (rotate left through carry, キャリーを通る左回転) または g RRC (rotate right through carry, キャリーを通る右回転) を押すと，左端または右端のビットがキャリービットに入り，反対端のビットにはそれまでのキャリービットが入ります。



1度に2ビット以上の回転　ビット・パターンがYレジスタにあって，nがXレジスタにあるときに，f RLn ， f RRn ， g RLCn ，

[g] [RRCn] のどれかを押すとビット・パターンが|n|ビットだけ回転します。スタック下降して，結果がXレジスタに入ります。

キャリーフラグ（フラグ4）の状態は [RL]，[RR]，[RLC]，[RRC] を|n|回実行した場合と同じになります。例えばn=3のときに [RRn] を実行すると，右から3番目のビット（ビット2）が1のときだけキャリーフラグをセットします。

例 8ビットの2ワードとして別々のレジスタに入れたデータを，16ビットの1ワードとして左方向に回転するキー操作を研究してみましょう。ここではワード長8ビットのまま 00011100 11100111 のワードを回転してみます。

キー操作	表　示	([STATUS]:2-08-1000)
11100	00011100 b	16ビットのワードの上位8ビット。
[f] [SL]	00111000 b	最上位のビットがキャリービットに入る。
[g] [LSTx]	00011100 b	上位8ビットをまたXレジスタに入れる。
11100111	11100111 b	16ビットのワードの下位8ビット。
[g] [RLC]	11001110 b	キャリービット（上位8ビットの最上位のビット）を下位の8ビットの最下位のビットに入れる。
[x⇄y]	00011100 b	XとYレジスタの内容を交換。
[g] [RLC]	00111001 b	キャリービットがワードの上位の1部として入り，0がキャリービットに入るのでCが消える。
[x⇄y]	11001110 b	新しいワードは 00111001 11001110。
[g] [CF] 3	11001110 b	左側の0を消す。

ビットのセット，クリア，判定

ワード中の個々のビットを1にセットしたり0にクリアするには，[SB] (set bit) または [CB] (clear bit) 機能を使います。またフラグの状態を判定するのと同様に，[B?] を使ってビットをセットしてあるかどうかを調べることもできます。プログラム中で判定すると，その結果によってプログラムの実行順序を変化させることができます。

ワード中の特定のビットをセット，クリア，または判定する方法は次のとおりです。

- 特定のビットを含んでいるワードをYレジスタに入れる。
- セット，クリア，または判定したいビットの番号をXレジスタに入れる。

キー（[SB] または [CB]）を押すとスタック下降して，結果のワードがXレジスタに入ります。

ビット番号は0からワード長よりも1小さい数までで，最下位のビットがビット番号0，それから左に向って順にビット番号が1ずつ大きくなります。

キー操作	表　示	([STATUS]:2-16-0000)
11111111 [ENTER]	11111111 b	ワードをキーインして，Yレジスタにコピーする。
11	11 b	ビット番号は3。
[f][CB]	11110111 b	スタック下降して，結果のワードがXレジスタに入る。

指定したビットをセットしてあるかどうかの判定（[f][B?]）は，プログラムの条件判断（そのビット・パターンによってプログラムの実行をどうするかを決める）として役立ちます。(この場合XとYレジスタ中に前記のように正しい値が入っていることが必要です。)条件ジャンプについては第9章で説明します。

マ ス ク

[MASKL] (mask left) や [MASKR] (mask right) の機能を使うと，ビットが1だけの左揃えまたは右揃えのワードを作れます。Xレジスタ内の数値でマスクを構成する1の個数を指定します。実行後はマスク・パターンがXレジスタ中に入ります（スタックは移動しません）。

マスクはワード長と同じ大きさまで作れます。マスクを途中だけに置きたいときは [MASKL] か [MASKR] とシフト機能を使用します。

例 2桁の数値をASCIIで表現すると1桁について8ビットなので合計16ビットを使います。ASCII 16進表現の65（00110110 0011 0101）の上位桁の6を取り出してそのASCIIコードの下半分を2進化10進数（BCD）に変換すると次のようになります。

```
      0011 0110 0011 0101  ASCIIの65 (3, 6, 3, 5)
[AND] 0000 1111 0000 0000  マスク
      0000 0110 0000 0000  取り出した上位の桁の6。
```

これをマスク操作前にシフト機能を使って次のようなキー操作をするとキーを押す回数が少なくて済みます。

キー操作	表 示	([STATUS]: 2-16-0000)
[HEX] 3635 [ENTER]	3635 h	
8 [f] [RRn]	3536 h	右方向に8ビット回転して使いたい桁の6を右揃えする。
4 [f] [MASKR]	F h	16ビットの右端4ビットを1のマスク(1111)にする。
[f] [AND]	6 h	右端の4ビットの6を取り出す。

ビットの合計

[g] [#B] (number of bits, ビット数) を押すとXレジスタ中のビット

を合計してその結果がXレジスタに入ります。元のビット・パターンはラストxレジスタに入ります。スタックは下降しません。（ワード長が1か2のときにはその結果は符号なしの状態で解釈する必要があります。）

2倍長の演算

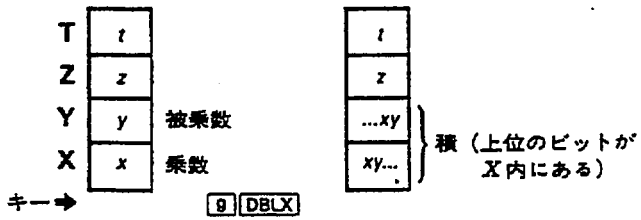
HP-16Cには2倍長の演算機能が3種類あります。 [DBLX] (double multiply)， [DBL÷] (double divide)， [DBLR] (double remainder)。この機能を使うと乗算のときは積のワード長を2倍にして，除算と剰余のときは被除数のワード長を2倍にして，正確な計算ができます。

16進数または8進数で2倍長の演算のときに結果を意味のある数値にするには，ワードの境界（ビット数の倍数）で1個の数字を分割しないように注意する必要があります。従ってワード長を指定するときは16進数演算では4の倍数，8進数演算では3の倍数にします。*

2倍長の乗算

[DBLX] はXとYレジスタ中にある同じワード長の2個の数値の乗算をして，その結果をXとYレジスタに2倍長で入れます。（スタックは下降しません。）下位のビットがYレジスタに，上位のビットがXレジスタに，それぞれ右揃えで入ります。

この演算のときのスタックの内容は下図の通りです。スタックにはt, z, y, xの各値が1個のレジスタに1ワードずつ入っています。



*第7章（プログラムの基礎）に10進数の整数状態で [DBLX] を使うプログラムがあります。83ページを見てください。

例 2進数表示，ワード長が5，2の補数状態で7×6を2倍長で求めてみましょう。

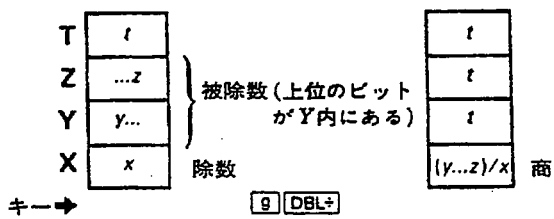
7	00111	Y中に5ビット。
×6	× 00110	X中に5ビット。
42_{10}	00001 01010$_2$	42_{10} の10ビット表現がXレジスタとYレジスタに分割されて入る。
	X Y	

キー操作	表 示	([STATUS]: 2-05-1000)
[BIN] 111 [ENTER]	00111 b	2進数表示。
110 [g] [DBLX]	00001 b	上位のビットがXレジスタに入る。
[x≷y]	01010 b	下位のビットがYレジスタに入る。従って結果は 00001 01010$_2$。

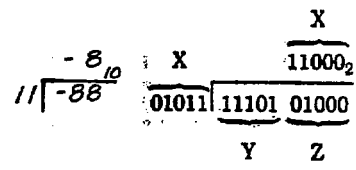
2倍長の除算

[DBL÷] はYとZレジスタ中にある2倍長の被除数(上位のビットがYレジスタ，下位のビットがZレジスタ) を，Xレジスタ中の1ワードの除数で割った商を求めます。スタックは2回下降して，結果の1ワードがXレジスタに入ります。

結果がそのときのワード長で表せない値のときは Error 0 を表示します。余りが0でないときにはフラグ4 (キャリーフラグ) をセットします。この演算のときのスタック内容は下図の通りです。



例　2進数表示，ワード長が5，2の補数状態のとき，(−88÷11) を2倍長で計算してみましょう。



X中に5ビットの結果。

-88_{10} を10ビットで表したものをY，Zの両レジスタに分割。

キー操作	表　示	(STATUS: 2-05-1000)
1000 ENTER	01000 b	10ビットの被除数の内下位5ビットをZレジスタに入れる。
11101 ENTER	11101 b	10ビットの被除数の内上位5ビットをYレジスタに入れる。
1011 g DBL÷	11000 b	商。
g CF 3	11000 b	左側の0を表示しないようにする。

2倍長の剰余

DBLR は DBL÷ と同じ計算をしますが，このときは商でなく剰余を求めます。商が64ビットを超えると Error 0 を表示します。

剰余は RMD (45ページ) と同様に求め，結果の符号は被除数の符号と同じになります。

例　2倍長の除算の応用

$\frac{5714AF2_{16}}{7E14684_{16}}$ の商を16進数で16桁まで求めてみましょう。

結果は分数になりますが，まず

$$\frac{5714AF2\overbrace{000\cdots0}^{0\text{が}16\text{個}}{}_{16}}{7E14684_{16}}$$

の商を整数で求めてからその左側に小数点を付ける（これで結果を 2^{64} で割ったことになります）ようにすると，整数演算状態でこの問題を解くことができます。このように長い分子のときには [DBL÷] を使います。

キー操作	表　示	([STATUS]: 2-64-0000)
[HEX]		16進数演算。
[f] SET COMPL [UNSGN]		符号なしの状態にするとより大きい結果まで範囲外にならずに求めることができる。
0 [ENTER]	0 h	2倍長の被除数は 5714AF2×2^{64}
5714AF2 [ENTER]	5714AF2 h	
7E14684 [g] [DBL+]	7E985d8C .h	キャリービットをセットした。
[f] [WINDOW] 1	b0d06F6A h.	結果は B0D06F6A 7E985D8C$_{16}$，従って問題の答は 0.B0D06F6A 7E985D8C$_{16}$。

第5章　実　　数

これまで説明してきた計算とは異なる分野の計算も出来るように，HP-16C に実数（10 進数だけ）の演算機能も用意してあります。実数演算状態（Floating-Point Decimal mode）では，ワード長は自動的に 56 ビットになります。

> 注　実数演算状態と整数演算状態とでは数値は互換性のない異なる形式で表現しています*。従って片方の形式で記憶レジスタに記憶した値は，HP-16C を他方の形式に切り替えたときに正しく変換できません。ただしまた元の形式に戻すと正しい値として使えます。

実数演算状態への切り替え

[FLOAT] (floating point)機能は実数演算状態に切り替え，X と Y レジスタの内容を（後記の通り）実数に変換します。

[f] [FLOAT] {0～9, [.]} を押すと実数演算状態に切り替わります。[FLOAT] の後に押す数字で表示したい小数点以下の桁数を指定します。数字の代わりに [.] キーを押すと浮動小数点表示（指数部あり表示）になります。（このキーを押す前に実数演算状態になっているときにはスタック中の数値変換はありません。）

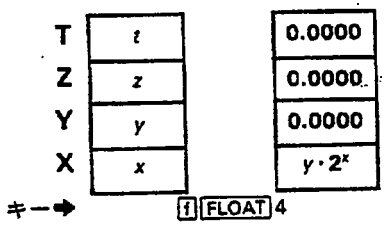
スタック中の変換

実数演算状態に切り替えるときに既にスタックに入っている値をそのまま残してその後の計算に使いたいことがあると思います。そのため HP-16C には整数演算状態から実数演算状態に切り替えるときにスタック中の値を変換して残しておくルーチンを組み込んであります。整数演算状態のときに [FLOAT] を実行すると Y と X レジスタ中の値は

---

* 実数演算状態では BCD(Binary Coded Decimal，2 進化 10 進数）形式，整数演算状態では 2 進数形式。

(y)(2ˣ)に等しい実数に変換されてXレジスタに入ります。またY,Z,Tの各レジスタをクリアします。



$y \cdot 2^x$が$9.999999999 \times 10^{99}$より大きいときにはフラグ5(範囲外)をセットし，オーバーフローの表示（全桁に9を表示）になります。オーバーフローがなければフラグ5をクリアします。

例　$25E47_{16}$を実数に変換しましょう。

キー操作	表　示	(STATUS: 2-64-0000)
HEX		16進数表示。
25E47 ENTER	25E47 h	仮数。
0	0 h	2の指数。
f FLOAT 2	155,207.00	実数演算状態に切り替えて小数点以下2桁まで表示。この値は$(25E47_{16}) \cdot 2^0$に等しい。

実数演算状態への切り替え時に発生する他の影響

整数演算状態（16進数，10進数，8進数，2進数表示のどれか）から実数演算状態（10進数だけ）に切り替えると，上記以外に次の状態に影響があります。

- ワード長は56ビットになります。

• 付録Dに IEEE 規格案の実数2進数表現と HP-16C の実数表現との変換プログラムがあります。

- スタック（Xレジスタを除く）とラストXレジスタをクリアします。スタック上昇が可能な状態になります。
- 記憶レジスタはクリアしません。ただし，実数演算状態で記憶したのではないレジスタの内容（インデクス・レジスタも含む）を呼び出そうとすると，普通は Error 6 を表示します。
- 補数状態はそのまま残りますが，演算機能や数値表示には関係しません。補数状態は，整数演算状態に戻るときのXレジスタの変換に関係します。

数値入力と表示指定

符号の変更　CHS (change sign)を押すと表示している数値の符号が変わります。負の値をキーインしたいときには数字部分をキーインしてから CHS を押します。実数演算状態ではこれで数値の入力を区切ったことになりません。

浮動小数点表示　f FLOAT . と押すと実数演算状態になると同時に，表示が浮動小数点表示（指数部あり表示）になります。仮数は小数点以下6桁まで表示します。

指数　指数部分の数値は EEX (enter exponent, 指数部の入力)を使って入力します。最初に仮数部分をキーインしてから f EEX を押し，その後で指数部分をキーインします。（仮数部分が負のときには EEX の前に CHS を押します。）指数が負のときには指数部分をキーインした後に CHS を押します。

仮数の桁が多過ぎて指数表示部分に入ってしまうときには，その桁は EEX を押したときに表示から消えますが，HP-16C の内部ではそのまま記憶しています。†

有効数字の表示　実数演算状態では表示指定とは無関係に HP-16C 内

• 詳細は 72 ページを見てください。

† 表示の読み違いを防ぐために，EEX は小数点より左の桁が8桁以上のときや 0.000001 より小さい仮数には働きません。そのような数をキーインしたいときは指数部分を正数か負数で大きくなるように仮数部分を変更してください。

部の全数値は10桁の有効数字と2桁の(10の)指数で表現しています。Xレジスタ内の数値の有効数字全10桁を見たいときは f CLEAR PREFIX を押します。PREFIX キーを押していると有効数字全体を表示します。

キー操作	表 示	(STATUS: 2-56-0000)
f FLOAT 3		
45 g √x	6.708	
f CLEAR PREFIX		
(押したまま)	6708203932	
(放す)	6.708	

オーバーフローとアンダーフロー　実数演算の結果Xレジスタ内の数値の絶対値が 9.999999999×10^{99} より大きくなると代りに $\pm9.999999999\times10^{99}$ が入ります(有効数字の最後の3桁は表示しません)。フラグ5(範囲外フラグ)をセットし，Gの文字を表示します。

実数演算の結果Xレジスタ内の数値の絶対値が $1.000000000\times10^{-99}$ より小さくなると，代りに0が入ります。オーバーフローまたはアンダーフローでプログラムの走行が止まることはありません。

整数演算状態への切り替え

基数キー(HEX , DEC , OCT , BIN)のどれかを押すと整数演算状態に戻ります(実数演算状態でなくなります)。

スタック中の変換

実数演算状態でなくなるときには，XとYレジスタの内容を実数演算状態に切り替えたときとは逆に変換します。つまり，Xレジスタ内の数値を$(y)(2^x)$の形式と考えて整数部分のyがYレジスタに，2の指数部分のxがXレジスタに入ります。yの値が0でないときには $2^{31}\leq|y|<2^{32}$ になるように定義します。つまり仮数yを四捨五入して32ビットの整数にします。指数部分のxの値は状態切り替え前のXレジスタ中の値が変換後の$(y)(2^x)$に等しくなるように変換します。

新しいxとyの値は指定した基数で表示します。(符号なしの状態のときにはxとyの絶対値がそれぞれXとYに入ります。)

整数演算状態から実数演算状態へ切り替えてから整数演算状態に戻すと，初めのy,xの組み合わせに戻らないのが普通ですが，変換前と変換後は等価です。

例　1.284×10^{-17} を整数の10進数表示にしてみましょう。

キー操作	表　示	([STATUS]:2-56-0000)
1.284	1.284	
[f][EEX]	1.284　00	
17[CHS]	1.284　-17	
[DEC]	-88 d	整数モードに切換え，Xレジスタには2の指数部分が入っている。
[x≷y]	73787526 .d	仮数
[f][WINDOW]1	39 d.	答は$(3973787526\times2^{-88})_{10}$。

例　次のキー操作は整数演算状態の1($1\cdot2^0=1$)を実数演算状態に変換してから整数演算状態に戻すと($80000000_{16}\times2^{-31}$)，同じ1が別の表現になる例です。

キー操作	表　示	([STATUS]:2-56-0000)
[HEX]		
1[ENTER]	1 h	y=1。|y|は$2^{31}\leq\|y\|<2^{32}$の整数ではない。
0	0 h	x=0
[f][FLOAT]4	1.0000	$(y)\cdot(2^x)=(1)\cdot(2^0)=1$
[HEX]	FFFFFFE1 .h	整数演算状態に戻す。
[f]SHOW[DEC]	-31 d	$x=-31_{10}$
[x≷y]	80000000 h	$y=80000000_{16}=2^{31}$。yはここでは$2^{31}\leq\|y\|<2^{32}$で$y\cdot2^{-31}=1$になる整数。

整数演算状態への切り替え時に発生する他の影響

実数演算状態から整数演算状態に切り替えると，上記以外に次のよう

な影響があります。

- ワード長は56ビットのままです。
- スタックと記憶レジスタはクリアしません。ただし整数演算状態で記憶したのではない記憶レジスタ内の値を本来の意味とは異なって解釈します。
- 補数状態は変化しません。

実数演算状態での数学演算

機　　能

整数演算状態で使用できる数学演算機能（[+]，[−]，[×]，[÷]，[√x]）はどれも実数演算状態でも使用できます。実数演算状態では[+]と[√x]の結果は整数演算状態のときのように整数の答に限られることはありません（小数部分まで計算します）。

[1/x]は実数演算状態のときだけ働き，Xレジスタ内の値の逆数を求めます。

範囲外フラグ

実数演算状態でフラグ5（範囲外）に影響する数学演算機能は[+]，[−]，[×]，[÷]だけです。この機能を実行すると必ずフラグ5をセットまたはクリアします。[f][FLOAT](0～9，[·])を実行したときも同様にフラグ5に影響があります。

キャリーフラグ（フラグ4）は実数演算状態では影響はありません。

実数演算状態では使用できない機能

一般的に表示と基数指定機能それにビット操作機能は，実数演算状態では使用できません。付録Bに実数演算状態で使用できない機能の一覧表があります。

・そのような値はレジスタの内容を変えずに実数演算状態に戻すと，本来の値通りに使用することが出来ます。

桁の区切り記号

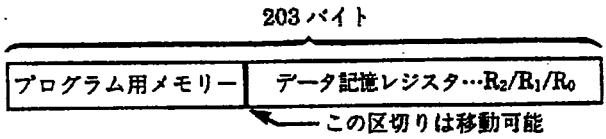
HP-16C は整数部分と小数部分の区切りに小数点(.)，整数部分の 3 桁ごとの位取りにコンマ(,)を表示するようにしてあります。しかしヨーロッパのようにこの記号が逆の国もあるので，次の操作（ON/.）をすると記号が逆になり，もう一度操作すると元に戻ります。この操作を繰り返すと記号が交互に変わります。

1. HP-16C の電源スイッチを切る。
2. ON を押したままにする。
3. . も押したままにする。
4. 先に ON を放し，次に . を放す。

第6章　メモリーと記憶レジスタ

メモリーの配分

HP-16C ではレジスタの形のデータ記憶部分とプログラム・ラインの形式のプログラム用メモリーとが同じ記憶領域を共用しています。この記憶領域は 203 バイト（1 バイトは 8 ビット）で，最初は全部がデータ記憶用のレジスタに配分してあります*。しかしプログラム命令を入力すると（第7章　プログラムの基礎を見てください），記憶領域の一部が自動的にプログラム用メモリーに変わります。



記憶レジスタからプログラム用メモリーへの変更

データ記憶レジスタからプログラム用メモリーへの自動的な変更は 7 バイト単位ずつで，それは 7 行分のプログラム命令に相当します。最初のプログラム命令を記録してから 7 行分を記録するごとに，7 バイトのメモリーがデータ記憶レジスタからプログラム用メモリーに切り替わります。

自動的なメモリー配分

記憶したプログラム命令*	プログラム用メモリーの配分	記憶レジスタの配分
0 バイト	0 バイト	203 バイト
1～7	7	196
8～14	14	189
15～21	21	182
⋮	⋮	⋮
190～196	196	7
197～203	203	0

* 1 行の命令＝1 バイト

* ここで最初と言うのは，工場出荷後に初めて HP-16C のスイッチを入れた状態または不揮発性メモリーをリセットしたときの状態を指します。

従って記憶レジスタ用のバイト数とプログラム用メモリーのバイト数の合計はいつも7の倍数です。

注　HP-16Cがデータ記憶レジスタをプログラム用メモリーに変更するのは，最も番号の大きいレジスタから番号の小さいレジスタに向って逆の順番です。また記憶レジスタに入っているデータは，そのレジスタがプログラム用メモリーに変わるときに消滅します。

プログラム用メモリーから記憶レジスタへの変更

一たんプログラム命令を記録すると，それをキー操作で削除するまでは記憶レジスタに戻ることはありません。プログラム用メモリーに配分した領域がデータ記憶レジスタに戻るのは，キー操作でプログラム命令を個々に削除，または全部一度に削除した(CLEAR [PRGM] または不揮発性メモリーのリセットで）ときだけです。プログラム用メモリーの命令削除部分は7バイト単位でデータ記憶レジスタに戻ります。

従って知らないうちにプログラム命令が消えてしまうことは絶対にありません。プログラム用メモリーに切り替わっているレジスタを利用しようとするとError 3（記憶レジスタが存在しない）を表示します。

記憶レジスタのビット長

1個のレジスタが1ワード分なので，レジスタのビット長はそのときのワード長によって変わります。記憶レジスタのビット長は必ず4ビット（1/2バイト）の倍数で，そのときのワード長と同じか，それより大きい最小の4の倍数です。例えばワード長が13，14，15，16のときには記憶レジスタ1個は16ビット（2バイト）です。

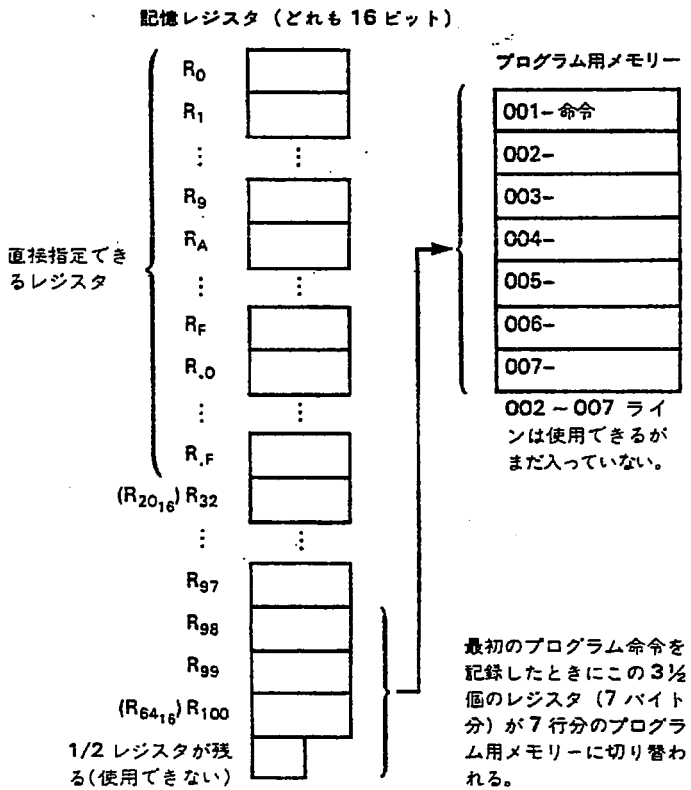
実数演算状態ではワード長と，記憶レジスタのビット長は自動的に56ビット（7バイト）になります。

データ記憶レジスタの最大個数は，データ記憶に使用できるバイト数（203からプログラム用メモリーのバイト数を差し引いたもの）を1レジスタあたりのバイト数で割った答です。例えばワード長が16ビットのとき，1レジスタは2バイト（16バイト）です。プログラム用メモリーをクリアしてあると（データ記憶に203バイトが使用可能），203/2＝101.5なので101個（R_0～$R_{100_{10}}$）の記憶レジスタが使用で

＊インデクス・レジスタ（後記）は例外です。インデクス・レジスタのビット長は一定（68ビット）で，プログラム用メモリーには切り替わりません。

きることになります(1/2個のレジスタはデータ記憶には使えません)。

ワード長が16ビットのときに，最初のプログラム命令を記録したときの記憶領域の配分は次のようになります。



8番目のプログラム命令を記録すると次の3½個のレジスタ(7バイト分)がプログラム用メモリーに切り替わります。その結果94½個のレジスタが残り，実際にデータ記憶レジスタとして使用できるのは94

個（R_0～$R_{99_{10}}$ または $R_{5D_{16}}$）だけになります。（残っている 1/2 個は次にプログラム用メモリーへ切り替わるときに使います。）

メモリー配分の状態確認（MEM）

f MEM (memory)を押すとそのときのメモリー配分状態を一時的に表示します（ MEM を押している間は表示します）。表示する内容は次の通りです。

P—*B*　　r—*RRR*

B　次の7バイトを切り替える（次の7バイト分の記憶レジスタの減少）までにまだプログラム用メモリーに追加できるバイト数（プログラム命令数）。$0 \leq B \leq 6$

RRR　そのときデータ記憶に使用できる記憶レジスタの個数。$0 \leq RRR \leq 406$ で，*RRR* は必ず 10 進数で表示。（レジスタの最小ビット長は 4 ビットなのでレジスタの最大数は 406 個になる。）

前ページの図の状態（プログラム命令を一つ入れ，ワード長 16 ビット）で MEM を押すと，次のように表示します。

キー操作	表　示		
ON/- BSP		0 h	不揮発性メモリーをリセット：ワード長が 16 ビットになりプログラム用メモリーをクリアした。
f MEM（押したまま） （放す）	P-0	r-101 0 h	7バイトをプログラム用メモリーに切り替えるまでに記録できるライン数は 0。101 個の記憶レジスタが使用可能。
g P/R	000-		HP-16C をプログラム入力状態に切り替える，000 ライン。
1	001-	1	プログラム命令を 1 行分記録する。

キー操作	表示		
f MEM	P-6	r-098	次の7バイトをプログラム用メモリーに切り替えるまでに，6行（バイト）を記憶できる。98個の記憶レジスタが使用可能。
	001-	1	
f CLEAR PRGM	000-		プログラム用メモリーをクリア。
g P/R		0 h	プログラム入力状態を終りにする。

記憶レジスタの操作

数値を記憶したり呼び出したりするときには，その数値は表示（Xレジスタ）とデータ記憶レジスタとの間でコピーします。前記のように使用できる記憶レジスタの最大個数はワード長とそのとき使用可能な記憶領域の大きさによって決まります。

その使用できる記憶レジスタの内，最大32個は記号(0～9,A～F,.0～.9,.A～.F)で直接指定（アドレス）できます。残りの部分（ワード長が16ビットのとき最大69個）は，インデクス・レジスタ（後記）を使って間接的にしか利用できません。番号の一番大きいレジスタから先にプログラム用メモリーに切り替わるので，データは番号が一番小さいレジスタから入れていくのが最良の方法です。(裏表紙内側の図も見てください。)

数値の直接記憶（ストア）と呼び出し（リコール）

直接利用できるデータ記憶レジスタとXレジスタとの間で数値を記憶したり（store，ストア）呼び出したり（recall，リコール）するには，STO |0～F,.0～.F, I| または RCL |0～F,.0～.F, I| と押します。STOを押したときには指定したレジスタにXレジスタ内の値をコピーし（レジスタに記憶します），RCLを押したときにはスタックが上昇し（上昇が可能な状態のとき），指定したレジスタ内の値をXレジスタにコピーします（Xレジスタに呼び出します）。

存在しないレジスタ（プログラム命令用に使っている記憶領域の場合

も含む）を指定しようとすると，Error 3 の表示になります。データ記憶レジスタが自動的にプログラム用メモリーに切り替わったときには，そこに記憶していたデータが消滅することと，また最も番号が大きいレジスタが最初に切り替わることを忘れないでください。

例　R_0 に 10^8 を記憶し，それを呼び出してから2倍してみましょう。

キー操作	表　示	
f FLOAT 0		表示はそれまでの X,Y の内容によって異なる。
f EEX 8	1　08	
STO 0	100,000,000.	
BSP	0.	
RCL 0	100,000,000.	
2 ×	200,000,000.	

レジスタ内容の変化

ワード長を変更すると，スタックの内容は影響を受けますが（第3章のワード長で説明しました），記憶レジスタの内容は影響を受けません。ただしワード長の変更によって記憶レジスタのビット長と境界（従ってレジスタの合計数）が変わります。その結果記憶しているビットの一部が本来の形式通りに利用できないようになることがあります（これは元のワード長に戻すとまた本来の形式通りに利用できるようになります）。このため計算などで一時的にワード長を変更しても，元のワード長に戻すと記憶したデータを再び使用することが可能です。

例　次のキー操作はワード長を2倍（16から32）にして，それから元に戻したときの R_0 と R_1 の内容の変化の説明用です。

キー操作	表　示	
HEX 10 f WSIZE		16進数（整数）表示。ワード長は16ビット。
f CLEAR REG		記憶レジスタをクリア。
1234 STO 0	1234 h	R_0 中に記憶。
5678 STO 1	5678 h	R_1 中に記憶。
20 f WSIZE	5678 h	ワード長を2倍にする。
RCL 0	56781234 h	R_0 には以前の R_0 と R_1 の内容がつながって入っている。

キー操作	表 示	
[RCL] 1	0 h	このR₁はクリアされている。
10 [f] [WSIZE]	0 h	元にワード長に戻す。
[RCL] 0	1234 h	このR₀とR₁の内容は元の通りです。
[RCL] 1	5678 h	

HP-16Cを整数演算状態と実数演算状態との間で切り替えてもレジスタの内容は変化しません。しかし両状態では内部表現が違うので，整数演算状態で記憶した内容は実数演算状態にすると前と同じ値にならず，逆の場合も同様です。ただし元の状態とワード長に戻すと，元の内容に戻ります。

データ記憶レジスタのクリア

[f] CLEAR [REG] (clear registers)を押すと，全データ記憶レジスタの内容をクリアして0にします。(スタックとラストxレジスタは変化しません。) データ記憶レジスタを1個だけクリアするにはそのレジスタに0を記憶させます。不揮発性メモリーをリセットすると全記憶レジスタやスタックもクリアします。

インデクス・レジスタ

インデクス・レジスタ(R_I)は専用の記憶レジスタで，他の記憶レジスタへの間接的な指定，プログラムのラベルへの間接的なジャンプ，およびプログラムのループを制御するときのループ・カウンタとして利用できます。(ジャンプとループ・カウンタとしての利用法は第9章で説明します。) 他の記憶レジスタとは異なって，インデクス・レジスタはワード長に関係なくいつも68ビットで，プログラム用メモリーに切り替わることはありません。

キー操作の省略

[I] または [(i)] キーを他のキー ([STO] , [RCL] , [GTO] , [GSB] など) の後に押すときには，[I] または [(i)] の前に押すはずの前操作キーの [f] は省略できます。省略してもHP-16Cは正しく理解できるためです。例えば [STO] [I] は [STO] [f] [I] と同じ意味で，しかも操作が簡単です。

インデクス・レジスタ中の数値の記憶と呼び出し

インデクス・レジスタの内容を直接利用するには [I] を使用します [STO] [I]，[RCL] [I]，[x≷I]。R_I 中に記憶する数値は 68 ビット表現ですが，値はXレジスタ内の値と等しくなります。R_I からXにコピーする数値はその時のワード長に合うように上位を切り捨てて，下位のビットだけになります。

記憶と呼び出し　Xレジスタ内の値をインデクス・レジスタにコピー（記憶）したり，インデクス・レジスタの値をXレジスタにコピーする（呼び出す）には [STO] [I]，[RCL] [I] を使いますが，これは他のレジスタのときと同じ働きです。

***X*と*I*の内容の交換**　[x≷y] と同じように，[x≷I] はXレジスタの内容とインデクス・レジスタの内容を交換します。

間接的な数値の記憶と呼び出し

[(i)] を使うと [STO] [(i)]，[RCL] [(i)]，[x≷(i)] で記憶レジスタを間接的に利用できます。R_I 中の数値の絶対値が記憶レジスタの番号になります。†（実数演算状態のときには整数部だけを使います。）下表は R_I と記憶レジスタの番号との関係を示したものです。

間接指定

R_I の内容		[(i)] で指定するレジスタ
0	(0_{16})	R_0
⋮	⋮	⋮
9	(9_{16})	R_9
10	(A_{16})	R_A
⋮	⋮	⋮
15	(F_{16})	R_F
16	(10_{16})	$R_{.0}$
⋮	⋮	⋮
31	($1F_{16}$)	$R_{.F}$
32	(20_{16})	R_{32} ($=R_{20_{16}}$)
33	(21_{16})	R_{33} ($=R_{21_{16}}$)
⋮	⋮	⋮

• R_I の内容を呼び出すときのワード長がそれを記憶したときのワード長よりも小さいと，呼び出した値が R_I 中の値と等しくならないことがあります。

† この絶対値は 68 ビットのワード長とそのときの整数状態を使って計算します。

最初の32個の記憶レジスタ(0～.F)よりも後の記憶レジスタは [STO] [(i)]，[RCL] [(i)]，[x≷(i)] を使用しなければ利用できませんが，上表のように最初の32個も同様に利用できます。

例 R_{326} に3を記憶してみましょう。データ記憶レジスタに R_{326} という大きい番号を使うには，ワード長を4ビットにする必要があります(203バイト/0.5バイト＝406個のレジスタ)。しかしインデクス・レジスタに326という値を記憶するにはワード長がそれよりも大きくなければなりません。(R_{31}($R_{.F}$) よりも後のレジスタを使うにはインデクス・レジスタを使います。) 従って R_{326} に3を記憶させるにはワード長の操作が2回必要になります。計算機がワード長4になっているときには，次の操作をします。

キー操作	表　示	([STATUS]: 2-04-0000)
[DEC] 0 [f] [WSIZE]		ワード長を大きく(64)する。
326	326 d	
[STO] [I]	326 d	R_I に 326_{10} を記憶。
4 [f] [WSIZE]	6 d	これで4ビットのレジスタ406個が使用可能。
3 [STO] [(i)]	3 d	R_{326} に3を記憶。
[BSP]	0 d	表示をクリア。
[RCL] [(i)]	3 d	R_{326} の内容を呼び出す。

第2部
HP-16Cの
プログラム

第7章　プログラムの基礎

第7～9章でHP-16Cのプログラムなどについて説明します。どの章も最初に基本的な技術を説明し（仕組み），次にその技術を実際に利用している例を示し（例），最後に詳しい各種の知識を説明する（詳細説明）という構成にしてあります。HP-16Cを利用する程度によって必要なところを読んでください。

仕　　組

プログラムの作成

HP-16Cのプログラム作成は非常に簡単で，これまで説明してきた手計算に使用するキー操作の順序をそのまま記録するだけです。(これをキー操作通りのプログラムと呼びます。) ただし一連の計算手順からプログラムを作成するときに，データの入力をどこにどのようにするかを決めることと，HP-16Cへのプログラムの入力つまり記録という手計算にはない2種類の仕事が必要です。またプログラム中で判断したり，条件または無条件ジャンプを使って反復計算することも可能です。

ここではXとYレジスタにある2個の16ビット長のデータを32ビット長のデータに連結してXレジスタに入れるプログラムを作りながら，プログラム作成と利用の基礎を学んでいくことにしましょう。

プログラムの入力

プログラム入力状態　[g] [P/R] (program/run, プログラムと計算状態の切り替え) を押すと，HP-16Cがプログラム入力状態になります (**PRGM** の文字を表示します)。プログラム入力状態のときには，大部分のキーを押すとプログラム命令として記憶し，演算などの実行はしません。

キー操作	表　示	
[g] [P/R]	000-	プログラム入力状態に切り替える，**PRGM** の

文字とライン番号を表示する。

プログラム入力状態のときのキー操作は，プログラム命令としてプログラム・ラインに入ります。ライン番号はプログラム用メモリー中の位置を示すものです。000 ラインはプログラム用メモリーの開始位置を示すだけのもので命令の記憶には使用できず，命令を記憶できる最初のラインは 001 ラインです。000 以外のプログラム・ラインは，命令を入力するまではメモリー中に存在しません。

プログラムはどのラインから始めることもできますが，普通は 001 ラインから始めます。命令を入力するとそれまでに入力したプログラムはそのままとまってプログラム用メモリーの下方に移動してその分だけライン番号が増加します。

プログラムの始め　プログラム用メモリーをクリアするとメモリー中の全プログラム命令が消え，HP-16C が 000 ラインに戻ります。それには，プログラム入力状態のときに [f] CLEAR [PRGM] と押します。

HP-16C が 000 ラインでなくて，しかもプログラム用メモリーをクリアしたくないときには，計算状態で [f] CLEAR [PRGM] または [GTO] [.] 000 と押すか，プログラム入力状態で [GTO] [.] 000 と押すかのどれかで，プログラム用メモリーの 000 ラインに戻ります。([GTO] [.] 命令を記録することはできません。)

[LBL] (label)命令——[g] [LBL] の後に数字または英字 [0～9, A～F] のラベルを続けたもの——は，プログラムやルーチンの開始位置を定義するのに使います。ラベルを使用すると，特定のプログラムやルーチンを選んで実行するのが簡単になります。

キー操作	表　示	
[f] CLEAR [PRGM]	000-	プログラム用メモリーをクリアして，プログラム用メモリーの 000 ライン（プログラム用メモリーの開始点）に戻る。
[g] [LBL] A	001-43,22, A	ラベル A のキーコード。

プログラムの記録 プログラムは手操作で問題を解くときに使用するのと同じキー操作を組み合わせたものです。プログラム入力状態のときに押したキーはプログラム命令としてメモリー中に記録します。表示にはライン番号とキーコードが現われますが，キーコードは1~2桁の数値で，キー位置を表すものです（後で説明します）。

例 ワードを連結するプログラムの本体部分は次の通りです。2個のデータをXとYレジスタ中に入れておくと，このプログラムの002~008ラインでこの2個の16ビット長のワードを1個の32ビット長のワードに連結します。初めにXレジスタ中にあったデータが，答では上位のビットになります。

キー操作	表示		
[HEX]	002-	23	
2	003-	2	ワード長を16から32へ2倍し，XとYの数値の左側に16ビット分を追加。
0	004-	0	
[f][WSIZE]	005-	42 44	
[g][LSTx]	006-	43 36	ワード長(32)を呼び戻す。
[f][SR]	007-	42 b	ワード長の1/2(16)を計算。
[f][RLn]	008-	42 E	数値を左方向に16ビットだけシフト。
[f][OR]	009-	42 40	このOR演算でXとYの内容を連結。

プログラムの終り

- [g][RTN] (return, 戻る) 命令はプログラムの終りで，000ラインに戻って停止します。†ただしプログラム用メモリーの終りには自動的な [RTN] があるので，メモリー中の最後のプログラムではこの命令を省略できます。

---

*プログラム中に入れられない機能（86ページに一覧表があります）を除きます。

†ただし，第9章 99ページの説明のようにリターン条件付きのサブルーチンを実行中の場合を除きます。

- [R/S] (run/stop, 走行と停止) 命令でプログラムが停止しますが, HP-16C は 000 ラインに戻りません。

キー操作	表　示	
[g][RTN]	010-　43 21	これがプログラム用メモリー中の最後のプログラムなら省略できる。

プログラムの走行

計算状態　プログラム入力が終ったら [g][P/R] と押すと, 計算状態に戻ります (**PRGM** の文字が消えます)。プログラムの走行 (実行) は計算状態のときだけ可能です。

キー操作	表　示	
[g][P/R]		計算状態 **PRGM** の文字が消える。以前の結果を表示。

計算状態とプログラム入力状態を切り替えても, プログラム用メモリー中の位置は変化しません。HP-16C の電源スイッチを切る前にプログラム入力状態であってもスイッチを入れると必ず計算状態になります。

プログラムの実行　計算状態で [GSB] の後に目的のラベルのキーを押します。この操作でラベルのあるプログラム・ラインからそのプログラムの実行を開始します。プログラム実行中は **running** の文字を点滅表示します。([GSB] はサブルーチンへのジャンプ一別の使用状態一にも使います。)

キー操作	表　示	([STATUS]: 2-16-0000)
[HEX]FFFE[ENTER]	FFFE h	最初の数値を X と Y レジスタに入れる。
DDDC	dddC h	2番目の数値を X レジスタに入れる。これが上位の桁になる。
[GSB]A	dddCFFFE h	連結された結果の 16 進数。
[f][STATUS]	2-32-0000	ワード長が 32 になっている。

これとは別に，[GTO] [.] nnn（3桁のライン番号）または [GTO] とラベルキーを押して実行したいライン位置に合わせてから [R/S] を押して実行を開始する方法もあります。

プログラム走行途中の一時休止

プログラム命令として [g] [PSE] (pause, 一時休止) を使うと，プログラムの実行を一時休止して中間結果を表示することが出来ます。休止時間を長くしたいときには [PSE] を2個以上使います。

[R/S] 命令は [R/S] の次のラインでプログラムの実行を完全に止めます。計算状態で（つまりキー操作で）[R/S] を押すと停止したラインからプログラムの実行を再開出来ます。

データの入力

プログラムを作成するときにデータをどのようにしていつ入力するかということを必ず考える必要があります。データの入力方法には，プログラム実行前にまとめて入力しておく方法と，プログラム実行途中に一時停止させて入力する方法の2通りがあります。

実行前に入力しておく方法

1. [STO] でデータをデータ記憶レジスタに記憶しておいて，プログラム中にプログラム命令の [RCL] でそれを呼び出すことができます。

   例えば前記の連結のプログラムでは，ワード長の値をプログラム中に書かずに次のようにレジスタから呼び出してもよいわけです。

キー操作

[g] [LBL] A
[HEX]
[RCL] 1　　ワード長を R_1 から呼び出す。
[f] [WSIZE]
⋮

2. プログラムの始めのラインでデータを使うのなら，プログラム実行前にスタック中に入れておくことが出来ます。プログラム

の始めに [ENTER] はいりません── [LBL]，[GSB]，[GTO] の各機能で数値の入力を区切ってスタックの上昇が可能な状態にします。この方法を前例で使いました。

スタックがあるので，プログラム実行前に2個以上の数値を入れておくことが可能です。それ以降の操作によるスタック内容の移動とスタックの操作方法（[x⇄y]，[R↓]など）に気を付けると，スタックの4個のレジスタ全体に数値をキーインしてそれを使うプログラムを作ることも出来ます。

これは79ページで使った方法で，そこではプログラムの実行前に2個の数値をX,Yレジスタに入れました。実際には下位と上位のワードをZとYレジスタに入れ，ワード長をXレジスタに入れておく方法もあります。もしこのように3個の値をスタックに入れておくならプログラムの始めの部分は次のようになります。

キー操作	
[g] [LBL] A	ワード長がXレジスタにある。
[HEX]	
[f] [WSIZE]	スタックが下降する。上位のワード（Yにあった）はXレジスタに移り，下位のワード（Zにあった）はYレジスタに移る。
⋮	

実行中に直接入力する方法　プログラム実行中に必要に応じてデータを入力します。プログラム中でデータが必要な位置に[R/S](run/stop)命令を入れ，プログラムの実行をそこで止めます。データを入力したら，[R/S] を押してプログラムを続行します。

プログラム用メモリー

HP-16Cにはデータ記憶とプログラム・ラインに使用できる記憶領域が203バイトあります。プログラム用メモリーは1度に7バイト（7ライン分）ずつ，データ記憶領域から配分するのでそのたびにデータ記憶レジスタが減少します。詳しくは66～70ページのメモリーの配分を見てください。

---

*ただし [R/S] はスタック上昇には無関係です。次にスタック上昇が可能にならない操作とスタック上昇に影響しない操作の一覧表は付録Bにあります。

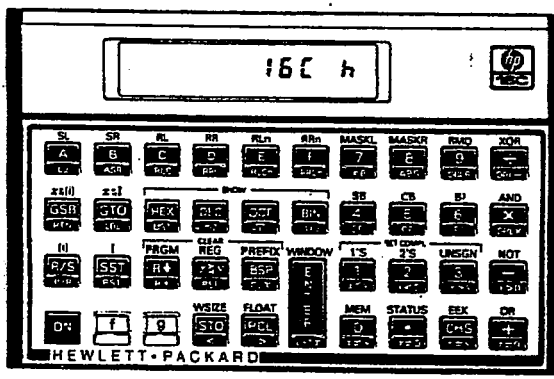
プログラム命令とキーコード

数字，小数点，演算キーはどれも1個が1命令としてプログラム用メモリーの1行（1バイト）に記録します。前操作キー（ f ，STO ，GTO ，LBL など）を含んだ命令も全体で1行に入ります。

HP-16Cの各キーは，プログラム入力状態ではそれぞれ1または2桁のキーコードで区別します。

2桁のキーコードの左の桁はキーの横行（一番上が1，...，一番下が4），右の桁は縦列（一番左から1，2，...，9，0）で，数字キー（A～Fも含む）のキーコードは数字そのものの1桁だけです。

命　令	コ ー ド
9 LBL 1	001-43,22, 1
f SET COMPL UNSGN	002- 42 3 SET COMPL UNSGN は"3"



42: 4行目の2列目

例

次のプログラムは2倍長の演算(55～58ページ参照)を使って，HP-16C

で可能な任意の基数の大きい値の乗算をして少くとも38桁までの正確な答を10進数で求める（つまり $10^{19}\times 2^{64}$ 未満）ものです。2倍長演算の結果はXとYレジスタ（Xに上位の桁，Yに下位の桁）に入ります。

2倍長の演算はHP-16C内部では2進数で行うので，意味のある10進数の答を得るためには次のような余分な操作（1個のレジスタ中に入れることが出来る最大の10の指数である 10^{19} で割る）が必要です。

キー操作	表　示	
[f] CLEAR [PRGM]		プログラム用メモリーの000ラインに合わせるが，クリアはしない。（これはプログラム入力状態のときだけメモリーをクリアする。）
[g] [P/R]	000–	プログラム入力状態（PRGMの文字を表示する。）
[g] [LBL] 1	001–43,22, 1	
[f] SET COMPL [UNSGN]	002– 42 3	符号ビットがないので大きい答まで求められる。
[g] [DBL×]	003– 43 20	X，Yレジスタの内容で2倍長の乗算。
[STO] 1	004– 44 1	結果の上位の桁を R_1 に記憶。
[x⇄y]	005– 34	
[STO] 2	006– 44 2	結果の下位の桁を R_2 に記憶。
[x⇄y]	007– 34	
[RCL] 0	008– 45 0	10の最大の指数（除数）を呼出す。
[g] [DBLR]	009– 43 9	
[RCL] 2	010– 45 2	積の下位の桁。
[RCL] 1	011– 45 1	上位の桁。
[RCL] 0	012– 45 0	除数。
[g] [DBL÷]	013– 43 10	

キー操作	表 示	
[DEC]	014- 24	答を10進数で表わすようにする。
[g] [RTN]	015- 43 21	

このプログラムを実行するには，ワード長を64にし，10^{19}（符号なしの状態で最大の10の指数）を R_0 に記憶します。次に 12345678987654 と 987654321234567 をXとYレジスタにキーインします。

キー操作	表 示	
[g] [P/R]		計算状態に戻す（PRGMの文字が消える）。それ以前の操作の結果を表示。
0 [f] [WSIZE] [DEC]		ワード長を最大の64にする。
[f] SET COMPL [UNSGN]		
10000000 00000000	00000000 .d	
0000 [STO] 0	00000000 .d	R_0 に 10^{19} を記憶。
12345678987654		
[ENTER]	78987654 .d	乗算する2個の数値を入力。
987654321234567	21234567 .d	
[GSB] 1	19326320 .d	ラベル1のプログラムを実行。結果の積はX，Yレジスタ。上位のワードは1,219,326,320。
[f] [WINDOW] 1	12 d.	
[エミュ]	31035818 .d	下位のワードは731,267,636,031,035,818。正確な答は1,219,326,320,731,267,636,031,035,818$_{10}$。
[f] [WINDOW] 1	12676360 .d.	
[f] [WINDOW] 2	73 d.	
	有効数字より前の0は表示しない。	

別の2個の数値でこのプログラムを繰り返すには，XとYレジスタにその値をキーインして [GSB] 1 と押します。（ [DBL÷] の余りが0でないのとこのプログラムの実行中にフラグ4をセットします。しかし009ラインで余りを計算するのでこれは別に問題にはなりません。）

詳細説明

プログラムのラベル

プログラム（またはサブルーチン）のラベルはHP-16Cの実行開始位置を探すときの目印です。プログラムやサブルーチンで使用できるラベルは全部で16個（0–9, A–F）です。

GSB ラベルのような探索命令があると，HP-16Cはそのラベルを探すためにプログラム用メモリーを下の方に向かって調べます。必要があればプログラム用メモリーの最後から001ラインに戻って探索を続けます。求めるラベルが見つかるとそこで探索を終えて実行を開始します。

計算機はそのときの位置から下方にだけ探索するので，同じプログラム中に同じラベルを2個以上使うことも可能です（あまりお勧めしません）。実行は最初に見つけたラベルのあるラインから開始します。

予定してなかったプログラムの停止

キーに触わった時 実行中にどれかのキーに触わるとプログラムの実行を停止します。ただし演算の途中ではそれが終るまで停止しません。

エラーによる停止 正しくない演算をしようとすると，プログラムが直ちに停止してErrorの表示になります。

エラー原因の命令(プログラムが停止したラインにある)のライン番号とキーコードを見たいときは，まず任意のキー1個を押してError表示を消してからプログラム入力状態にします。

プログラム実行中に範囲外の状態（オーバーフローを含む）が発生すると，フラグ5をセットしてGの文字を表示しますが，プログラムの実行は停止しません。

プログラム中に入れられない機能

HP-16Cがプログラム入力状態のとき，大部分の機能（キー操作）をプログラム用メモリー中に命令として記録できます。ただし下記の機

能はプログラム用メモリー中に命令として記録することが出来ません
(これをプログラム不能と呼びます)。

[f] CLEAR [PREFIX]
[f] CLEAR [PRGM]
[ON] / [−]
[ON] / [·]
[ON] / [×]

[g] [P/R]
[f] [MEM]
[f] [STATUS]
[ON] / [D]
[ON] / [+]

[SST]
[g] [BST]
[BSP]
[GTO] [·] *nnn*
[ON] / [÷]

第8章　プログラムの編集

HP-16C には既に記憶している命令やプログラムを変更したいときに使う幾つかの編集機能があります。

仕　組　み

プログラムを修正する作業には，まず変更が必要な位置に移動しそれから削除や挿入するという二つの段階があります。

プログラム用メモリー中のライン位置への移動

[GTO] (go to, ジャンプ) 命令　[GTO] [·] nnn と押すと，計算状態のときもプログラム入力状態のとき (PRGM を表示している) もプログラム用メモリーのライン番号 nnn まで移動します。このプログラム不能の機能は手操作でプログラム用メモリー中の希望する位置に移動するときに使います。nnn は 000≦nnn≦203 の 3 桁の数値で，実際にあるプログラムのライン番号に合わせる必要があります。

[SST] (single step, 1 ステップ) 命令　[SST] を押すとプログラム用メモリー中のラインを 1 回につき 1 行ずつ進みます。このプログラム不能の機能はプログラムのトレース (trace, 追跡) に使います。トレースする時にプログラムを実行しながら進むことも実行しないで進むことも出来ます。

プログラム入力状態のときに [SST] を押すと，メモリ中を 1 行だけ前進してその命令を表示します。命令を実行しません。キーを押し続けるとプログラム用メモリー中のラインを次々に表示します。

計算状態のときに [SST] を押すと，押している間はそのプログラム・ラインを表示します。キーを放すとその命令を実行して，結果を表示すると同時に次に実行するラインに進みます。これはプログラムを 1 回に 1 行ずつ実行してその内容を見るトレース (追跡) で，プログラムのチェックにとても便利です。

[BST] (back step, 後戻り) 命令 プログラム入力状態または計算状態で [g] [BST] を押すと，プログラム用メモリー中のラインを1行後退します。この機能もプログラム不能です。プログラム入力状態で [BST] を押し続けると次々に後退を続けられます。どちらの状態でもプログラム命令を実行しません。

プログラム命令の削除

プログラム命令を削除するにはプログラム入力状態で [BSP] (back space) を使います。削除したいラインに移動してから [BSP] を押します。削除するとそれ以降のプログラムの順序はそのままでライン番号だけが小さくなります。

計算状態で [BSP] を押すとプログラム用メモリーには何の影響もありませんが，表示をクリアします (18ページ参照)。

プログラム命令の挿入

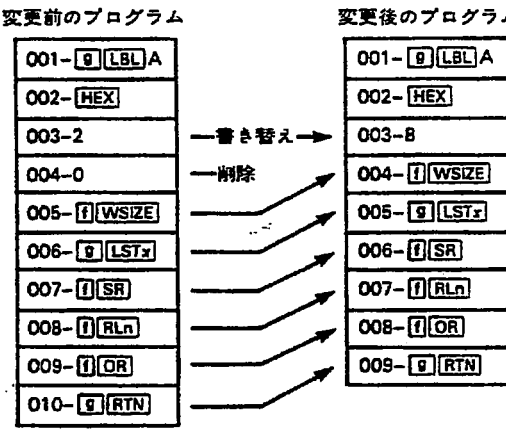
プログラム・ラインを追加するには，まずメモリー中の挿入したいラインの一つ前まで移動します。そこで新しい命令をキーインするとその命令がそれまで表示していたラインの次に入ります。既にある命令を変更したり書き替えたりしたいときにはそれを削除してから新しい命令を挿入します。

メモリー領域を全部使っているとHP-16Cは新しいプログラム命令を受け入れずに **Error 4** を表示します。

例

前章の連結プログラム (77ページから) を使ってHP-16Cの編集機能を説明します。まず最初にワード長を32 (20_{16}) から8 (8_{16}) に変更し，変更したプログラムを1行ずつ進めてその実行をチェックしてみましょう。

編集過程は下図の通りです。このプログラムをメモリーの001−010ラインに記憶しているものとします。



プログラム・ラインを削除または追加すると，それ以降のライン番号が変わります。終りの方から順に編集するとまだ編集してない部分のライン番号は変わりません。それで元の番号を使って目的のラインに行けます。下記の編集手順は連結プログラムが001－010ラインにあるものとして説明します。

キー操作	表　示		
g P/R			プログラム入力状態。（ライン位置はそれまでの操作によって異なる。）
GTO · 004（または SST を使う）	004-	0	004ライン（命令は数字の0）に移動。
BSP BSP	002-	23	004と003ラインを削除。
8	003-	8	新しい003ラインに8を挿入。この3操作で20が8に変わった。

変更したプログラムをトレースするために，計算状態に戻し，ワード長を4にし，Yレジスタに 7_{16}，Xレジスタに 6_{16} を入れます。SST

を押して1回に1行ずつプログラムを実行してみましょう。2倍長の連結された結果が 67_{16} になるはずです。

キー操作	表　示	(STATUS: 2-04-0000)
g P/R		計算状態。
HEX		
7 ENTER	7 h	4ビットの下位のワード。
6	6 h	4ビットの上位のワード。
GTO A	6 h	プログラム用メモリーのラベルAの位置にジャンプ。
SST (押したまま)	001-43,22, A	プログラムの001ライン：ラベルA。
(放す)	6 h	Xレジスタの内容。
SST	002- 23 6 h	002ライン：16進数表示にする。
SST	003- 8 8 h	003ライン：8
SST	004- 42 44 6 h	004ライン: f WSIZE ワード長が8になった。
SST	005- 43 36 8 h	005ライン: g LSTx ワード長を呼び戻す。
SST	006- 42 b 4 h	006ライン：f SR ワード長の½を計算。
SST	007- 42 E 60 h	007ライン：f RLn 左方向に4ビットのシフト。
SST	008- 42 40 67 h	008ライン：f OR 8ビットに連結されたワード。
SST	009- 43 21 67 h	最後の命令(g RTN)。

詳細説明

ラインの位置

プログラム用メモリー中のHP-16Cライン位置は電源スイッチの入り切りやプログラム入力状態と計算状態の切り替えでは変わりません。従ってプログラム編集中に電源スイッチが自動的に切れても，スイッチを入れてプログラム入力状態に切り替える（スイッチを入れると必ず計算状態になります）だけで元の位置に戻れます。

HP-16Cは命令を入れてない（まだ記録してない）プログラム・ラインに移動することはできません。[SST]を使ってプログラム用メモリーのそのときの終端まで行くと次は000ラインに戻ってしまいます。

HP-16Cの初期化

記憶レジスタを使用したり，ある状態によって実行順序を変えたりするプログラムでは，記憶レジスタの内容と計算機の状態がプログラムに影響します。実行中のプログラムに不適当な状態になっていると誤った結果になります。そこでプログラムを実行する直前か，プログラム中の先頭部分でHP-16Cを初期化する（レジスタをクリアしたり，適正な状態に切り替える）のが良い方法です。あるプログラム中で切り替えた状態がそれ以降のプログラム実行に影響することも忘れないでください。

HP-16Cを初期化するのは次の各機能です。CLEARの各機能，SET COMPLの各機能，整数演算の基数指定，実数演算状態，[WSIZE]，[SF]，[CF]

第9章　プログラムのジャンプと制御

HP-16Cのジャンプ機能には，無条件ジャンプ，条件ジャンプ（ある状態によってプログラムの実行順序を変える），とサブルーチンがあります。またカウンタの値を記憶できるインデクス・レジスタを使用すると，ジャンプやループを広範囲に制御できるようになります。

仕組み

ジャンプ

[GTO] (GO TO) 命令　[GTO] ラベル命令で無条件ジャンプが出来ます。実行中のプログラムに [GTO] |0−9, A−F, [I]| 命令があると，実行は指定したプログラムまたはルーチンに移ります。（ライン番号によるジャンプは出来ません。）HP-16Cは指定したラベルをメモリーの下方向に探します。必要があれば001ラインに戻って再び下方を探します。

計算状態のときにキー操作で [GTO] ラベルを使うと，プログラム用メモリー中の指定したラベル位置まで移動します。プログラムの実行はしません。

[GSB] (go to subroutine) 命令　[GSB] ラベル命令で指定したラベルから始まるサブルーチンに実行が移ります。それから [RTN] 命令に出会うと，自動的に元のメインプログラムに戻ります。†サブルーチンの実行については後で説明します (99ページ)。

• [GTO] [I] と [GSB] [I] はキーを省略した（[f] キーを押さない）操作です。73ページを見てください。

† [RTN] 条件付きのサブルーチンを実行中でないときには [RTN] 命令で実行を停止して000ラインに戻ります。

インデクス・レジスタを使った間接ジャンプ

R_I（インデクス・レジスタ）にインデクスの値を入れておくと，それに応じたラベルに間接的にジャンプしたり（GTO I），間接的にサブルーチンにジャンプしたり（GSB I）出来ます。

この機能は下表のようにインデクス・レジスタ内の値の絶対値に対応するラベルにジャンプします。（実数演算状態ではR_I中の値の整数部だけを使います。）ラベルには0～9，A～Fの16種類あります。

例えば，インデクス・レジスタ内の値が-14_{10}（STATUS：2—08—0000）のときGTO I命令でプログラムがジャンプする先はLBL Eです（$|-14_{10}| = E_{16}$）。

間接的なジャンプ

R_I内の値		GTO I または GSB I で実行が移るラベル
0	(0_{16})	LBL 0
⋮	⋮	⋮
9	(9_{16})	LBL 9
10	(A_{16})	LBL A
⋮	⋮	⋮
15	(F_{16})	LBL F

条件判断

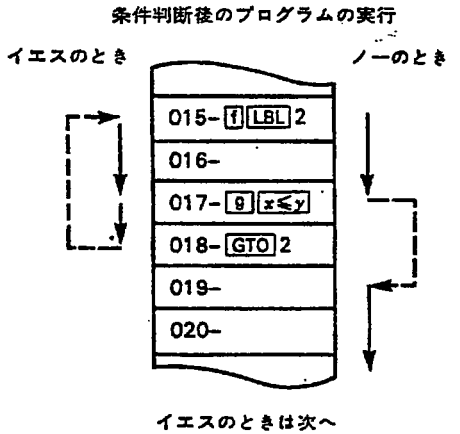
プログラムの実行順序を変えるもう一つの方法は条件判断によるもので，Xレジスタの数値と0またはYレジスタの数値を比較してイエスかノーの判断を行ないます。（1の補数状態で判断するときは$-0=0$と見なします。）

HP-16Cの条件判断用に次の8種類（どれも9を使う機能）があります。

x≤y　x<0　x>y　x>0　x≠y　x≠0　x=y　x=0

＊絶対値はワード長68とそのときの補数状態で計算します。

条件判断をすると，プログラムの実行はイエスのときには次へという規則に従って進み，条件がイエスのときは次のラインに進み，条件がノーのときには次の1ラインだけ飛びます。条件判断のすぐ後に GTO 命令を置いて，条件が合ったときだけ GTO でジャンプするような条件ジャンプにするのが普通です。



フラグとビットのセットの判定

条件ジャンプのための条件判断は F? (flag set ?) と B? (bit set ?) を使っても出来ます。この機能の実行後も前の条件判断と同様にプログラムはイエスのときは次への規則に従います（上図）。フラグ（またはビット）をセットしてあると次のラインに進み，フラグ（またはビット）をクリアしてあると次の1ラインだけを飛びます。

第3章で説明したように各フラグの番号と意味は次の通りです。

0
1
2
} ユーザ用フラグ（プログラムの制御に使用）

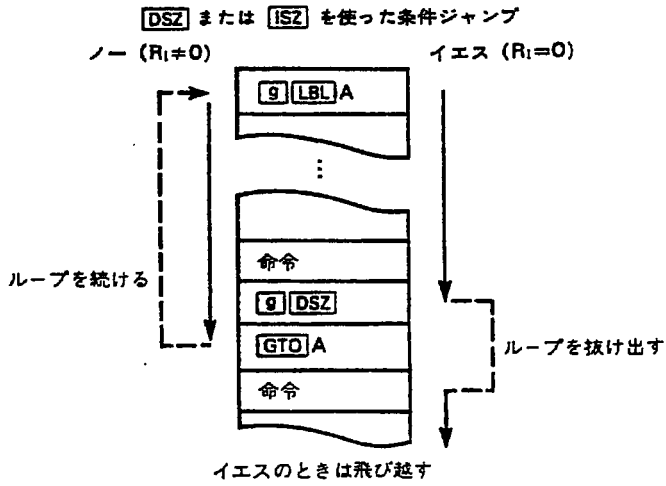
3　左側の0を表示するかしないか
4　キャリーまたはボローの状態
5　範囲外の状態

フラグ4と5はHP-16Cが自動的にセットしますが，ユーザがそれをセットすることも出来ます。フラグ4をセットするとキャリービットが1になります。フラグのセットについては第3章（38ページ），ビットのセットについては第4章（53ページ）を見てください。

カウンタを使ったループの制御：[DSZ]と[ISZ]

[DSZ]：(decrement and skip next line if counter equals zero，カウンタを1減少し，もしそれが0だったら次のラインを飛び越す)と[ISZ](increment and skip if zero，カンンタを1増加し，もしそれが0だったら次のラインを飛び越す）機能はインデクス・レジスタ中のカウンタの値を増減（増やす／減らす）し，その値によってループの実行を制御します。カウンタの値が0になるとプログラムは次の1行だけを飛び越して実行を続けます。

実行中のプログラムにこのどちらかの機能があるとインデクス・レジスタ中のカウンタの値が1だけ減少（[DSZ]）または増加（[ISZ]）し，その結果の値が0のときは次の1ラインだけを飛びます。その飛び越すラインをループにジャンプする命令にしておくと0になったときにループから抜け出すことが出来ます。



R_i 中の値はそのときの補数状態に従って解釈します。正数でも負数でも良く，また整数でも実数でも構いません。 [DSZ] と [ISZ] はキャリーフラグと範囲外フラグの状態には影響を与えません。

例

記憶しているデータの値が正しいかどうかを調べるためにチェックサムのルーチンを使うことがあります。 [#B] を使うとビット・パターンの合計値を求めておき，後で同じビット・パターンの合計値と比較します。

次のプログラムは指定した記憶レジスタ内のビット・パターンの全ビットを合計してチェックサムを求めます。記憶レジスタ R_A～R_I の内容のチェックサムを順次求めます。ビットを合計すると，レジスタ Y と Z に入っているそのときの2倍長のチェックサムに加算します。下記はライン 012 の直前にスタックに入っている内容です。

T		
Z		それまでのチェックサム：上位のワード
Y		それまでのチェックサム：下位のワード
X		この数値のビットを合計して，Y と Z 内にあるそれまでの2倍長のチェックサムに加算する。

結果のチェックサムは X と Y レジスタに入ります。

このプログラムは [DSZ] をインデクス・レジスタ内のレジスタ・ポインタを減少させてループの条件ジャンプを制御するために使っています。

キー操作	表示			
[g] [P/R]	000-			
[f] CLEAR [PRGM]	000-			
[g] [LBL] D	001-43,22, d			
[f] SET COMPL [UNSGN]	002-	42	3	ビットの合計用に符号なし状態にする。
4	003-		4	
[f] [WSIZE]	004-	42	44	ワード長は 4。

キー操作	表　示		
[HEX]	005–	23	
A	006–	A	
[STO] [I]	007–	44 32	最初のレジスタ（R_A）の番号を R_I に記憶。
0	008–	0	
[ENTER]	009–	36	チェックサムを0にする。
[g] [LBL] 0	010–	43,22, 0	合計を求めるループの開始（スタック上昇が可能になる）。
[RCL] [(i)]	011–	45 31	R_I に記憶されている番号のレジスタの内容を呼び戻す。
[g] [#B]	012–	43 7	X レジスタ中のビットを合計する。
[+]	013–	40	この合計をそれまでのチェックサムの下位のワードに加算。キャリーフラグをセットすることがある。
[x≷y]	014–	34	それまでのチェックサムの上位のワードを X に入れる。
0	015–	0	X に0を入れる。
[g] [RLC]	016–	43 C	前の加算でキャリーがあったら X に1を入れる。
[+]	017–	40	キャリービットをチェックサムの上位のワードに加算。
[x≷y]	018–	34	チェックサムの下位のワードを X に戻す。
[g] [DSZ]	019–	43 23	R_I 内のレジスタ番号を1減らす。
[GTO] 0	020–	22 0	R_I 内のレジスタ番号がまだ0でなければループを続ける。
[g] [RTN]	021–	43 21	

それでは R_1〜R_A に次の4ビットの16進数を入れて最新のチェックサム（ビットの合計）を計算しましょう。

R_1: A	R_3: B	R_5: 3	R_7: A	R_9: D
R_2: 7	R_4: 1	R_6: D	R_8: 2	R_A: 6

キー操作	表　示	(STATUS: 0-04-0000)
g P/R		計算状態に戻す。
HEX		
A STO 1	A h	上記の値を R_1〜R_A に入れる。
⋮	⋮	
6 STO A	6 h	
GSB D	6 h	2倍長のチェックサムの下位のビット。
x≷y	1 h	上位のビット：前記ビットパターンの合計は 16_{16} (22_{10})。

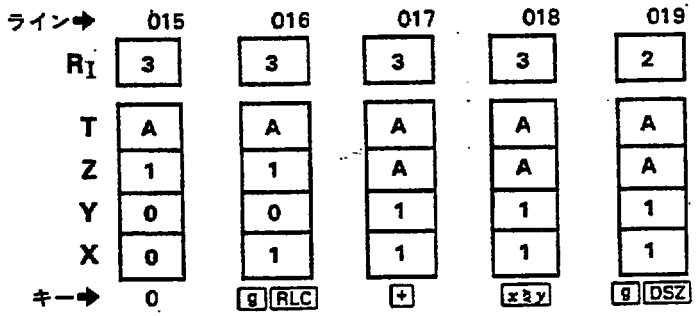
プログラムを作ったり分析するときには各命令の前後のスタックの内容を図に表すと便利です。下のスタック図は上記のプログラムのループ部分（ LBL 0:010〜019 ライン）でのスタックの内容の変化を示したものです。

このループの8回目の繰り返しのときに013ラインで R_3 の内容のチェックサムをそれまでのチェックサム（E_{16}）に加えるとワード長を超えるので、キャリーフラグをセットします。この繰り返し部分が下図です。(T と Z レジスタの A は006と007ラインの残り物です。)

ライン➡	010	011	012	013	014
R_I	3	3	3	3	3
T	A	A	A	A	A
Z	A	0	0	A	A
Y	0	E	E	0	1
X	E	b	3	1	0
キー➡	g LBL 0	RCL (i)	g #B	+	x≷y

012ラインでそのとき R_I で間接指定したレジスタの内容のチェックサ

ムを求め，013ラインでこのチェックサムをそれまでのチェックサムの下位のワードに加算します。014～017ラインで前回の加算のキャリービットをチェックサムの上位のワードに加えます。

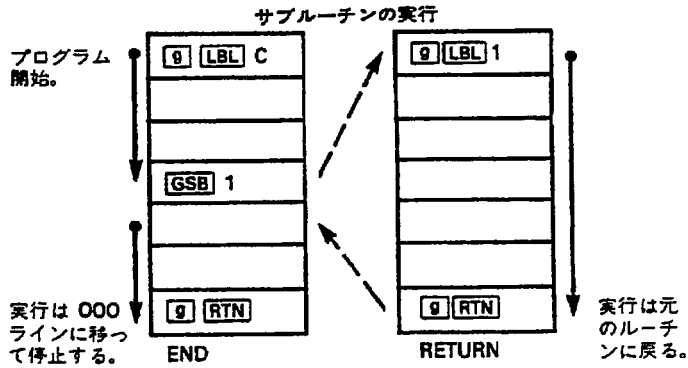


019ラインでR_i内の記憶レジスタの番号を1だけ減らすので，次のループで別のレジスタ内のビットの計を求めます。

詳細説明

サブルーチン

実行　サブルーチンにジャンプする（go to subroutine）命令の GSB ラベルは，RTN までサブルーチンを実行するという条件付きのジャンプ命令です。プログラムの実行は一たん指定したラベルがあるラインに移り，次の RTN 命令に出会うまでそのまま続け，RTN で元の GSB 命令のすぐ次の命令に戻ります。



内側のサブルーチン　内側のサブルーチンとはあるサブルーチンの中で別のサブルーチンを使うことで，ネスティングとも呼び，HP-16Cでは4重まで可能です（主体のプログラムは数えません）。

5重以上に重なったサブルーチンを使おうとすると，5重目の GSB の位置で停止して Error 5 を表示します。内側のサブルーチンがないサブルーチンや，内側にサブルーチンがあるサブルーチンを何組使用出来るかについては，メモリー容量以外には制限がありません。

プログラム中とキー操作の GSB

GSB キーには次の二つの異なる機能があります。1) キー操作でプログラムを実行する，2) 実行中のプログラム中でサブルーチンにジャンプする。キー操作でプログラムを実行するために GSB を使ったときには RTN までサブルーチンを実行するという条件が付かないので注意してください。次の RTN に出会うと，途中に GSB のプログラム命令（サブルーチンにジャンプ）が無ければプログラムの実行はライン 000 に戻って停止します。

付録A　エラーとフラグ

エラー発生原因

正しくないパラメータを使った操作をしょうとすると（例えば存在していないフラグ番号を指定した），Errorの文字と数字を表示します。このエラー表示を消すには任意の1個のキーを押します。こうするとエラー表示になる直前の表示に戻ります。番号別のエラー発生原因は次の通りです。

Error 0　数学演算が正しくない

[+]	x=0だった。
[RMD]	x=0だった。
[DBL+]	•x=0だった。 •商がワード長に収まらなかった。
[DBLR]	•x=0だった。 •商が64ビットを超えた。
[√x]	x<0だった。
[1/x]	x=0だった（実数演算状態のときだけ）。

Error 1　フラグ番号，[WINDOW]の番号，[FLOAT]の桁数，[GTO][.]の数字が正しくない

フラグ番号が6以上だった。
[WINDOW]の番号が8以上だった。
[FLOAT]にA～Fを使った。
[GTO][.]にA～Fを使った。

Error 2 ビット数が正しくない

[MASKL] または [MASKR]	|x|>そのときのワード長
[RLn] または [RRn]	|x|>そのときのワード長
[RLCn] と [RRCn]	|x|> (そのときのワード長 +1)
[SB], [CB], [B?]	|x|≧そのときのワード長
[WSIZE]	|x|>64

Error 3 レジスタ番号が正しくない

指定した記憶レジスタが存在しない。

Error 4 ラベルまたはライン番号の指定が正しくない

指定したラベルまたはライン番号が存在しない。204 行以上のプログラムを記録しようとした。

Error 5 サブルーチンのネスティングが大き過ぎる

サブルーチンのネスティングが 4 重を超えた。

Error 6 レジスタの内容が無効

実数演算状態のときに，実数演算状態で記憶したのではない記憶レジスタ（R_I を含む）の内容を呼び出そうとした。

実数演算状態のときに，R_L の内容が実数ではないのに [DSZ] または [ISZ] を使用した。

Error 9 要修理

自己診断で回路の故障を発見した，キーの動作試験中に違うキーを押した。付録 C を見てください。

Pr Error (Power Error)

電源異常のため不揮発性メモリーの電源が切れてリセットしてしまった。

フラグに影響する機能

二つの重要なフラグのキャリーフラグ（4）と範囲外フラグ（5）は，整数演算状態での数学演算とシフト機能によって影響（セットまたはクリア）を受けます。影響する機能は下表の通りです。

×＝セットまたはクリア　　0＝常にクリア　　－＝影響しない

機能	影響を受けるフラグ		使用するレジスタ	
	キャリー(4)	範囲外(5)	演算に使用するデータ	結果
[+]	×	×*	X, Y	X
[-]	×	×*	X, Y	X
[×]	－	×*	X, Y	X
[÷]	×	×*	X, Y	X
[√x]	×	－	X	X
[DBL×]	－	0	X, Y	X, Y
[DBL÷]	×	0	X, Y, Z	X
[ABS]	－	×	X	X
[CHS]	－	×	X	X
[SL]	×	－	X	X
[SR]	×	－	X	X
[ASR]	×	－	X	X
[RL]	×	－	X	X
[RR]	×	－	X	X
[RLC]	×	－	X	X
[RRC]	×	－	X	X
[RLn]	×	－	X, Y	X
[RRn]	×	－	X, Y	X
[RLCn]	×	－	X, Y	X
[RRCn]	×	－	X, Y	X

＊実数演算状態でも影響を受けます。

付録B　操作の分類

ここでは操作と略記しましたが，プログラム可能の機能をプログラム中に命令として入れておいたのを実行したときにも，キー操作だけのときと同じ効果があります。

数値入力を区切る操作

HP-16Cの大部分の操作は数値入力を区切ります。つまりこの操作の後でキーインした数字を別の数値の一部と見なします。

ただし次の数値入力操作は数値入力を区切ったことになりません。

[0]～[9]　[·]　[CHS]（実数演算状態のときだけ）
[A]～[F]　[EEX]　[BSP]

スタック上昇に影響する操作

スタック上昇に与える影響からHP-16Cの操作を3種に分類できます。次にスタック上昇が可能にならない操作，次にスタック上昇が可能になる操作，スタック上昇に影響しない操作です。

次にスタック上昇が可能にならない操作

HP-16Cで次にスタック上昇が可能にならない操作は二つです。このどちらかの操作後にキーインした数値はXレジスタのそれまでの内容を書き替えるだけで，スタック上昇はしません。

[ENTER]　[CLx]*

---

* 数値入力を区切った後の [BSP] も同じです。

スタック上昇に影響しない操作

影響しない操作はたくさんあって，スタック上昇についてのそれまでの状態（次にスタック上昇が可能になるか可能にならないか）は変わりません。

下表はスタック上昇に影響しない操作です：

HEX	WINDOW	MEM	PSE
DEC	<, >	STATUS	R/S
OCT	GTO . *nnn*	CLEAR PREFIX	SST
BIN	P/R	CLEAR REG	BST

SHOW {HEX, DEC, OCT, BIN}
SET COMPL {1's, 2's, UNSGN}
FLOAT（実数演算状態のときだけ）

次にスタック上昇が可能になる操作

大部分の操作は次にスタック上昇が可能になります。この操作の後で数値をXレジスタに入れるとスタック内容が上昇します。

上記の二つの表にない操作は全部次にスタック上昇が可能になる操作です。

ラストxレジスタに影響する操作

下表の操作をするとそれまでのXレジスタ中のxがラストxレジスタに入ります。

−	RMD	XOR	ABS	LJ	RLCn	RLC	SB	RRn
+	DBL×	NOT	$\sqrt{x}$	ASR	RRCn	RRC	CB	#B
×	DBL÷	OR	1/x	RL	SL	MASKL	B?	CHS †
÷	DBLR	AND	WSIZE	RR	SR	MASKR	RLn	

＊数値入力操作も数値の入力が終る（区切る）までではこれにあたります。
†整数演算状態のときだけです。

整数演算状態で [FLOAT] を実行するとラストxレジスタをクリアします。

表示部分の移動（スクロール）に影響する操作

下表の操作は表示部分の移動（スクロール）を元に戻さない，つまり表示を [WINDOW] 0 に戻しません。これ以外の操作はどれも表示部分を元に戻します。

[<], [>]	[P/R]	[PSE]	[SST]
[ON]	[LBL]	[R/S]	[BST]
[STO]	[RTN]	[GSB]	[DSZ]
[MEM]	[SF], [CF], [F?]	[GTO]	[ISZ]
[STATUS]			

SHOW {(そのときの基数)} CLEAR {[PRGM], [REG], [PREFIX]}

[x≤y], [x<0], [x≥y], [x>0], [x≠y], [x≠0] [x=y], [x=0]

前操作キー

HP-16C の前操作キーは次の通りです。

[f]	[STO]	[SF]	[LBL]	[GTO] [.]
[g]	[RCL]	[CF]	[GSB]	
[FLOAT]	[WINDOW]	[F?]	[GTO]	

実数演算状態では使用できない機能

[SL]	[RR]	[MASKL]	[SB]	[NOT]
[LJ]	[RRC]	[MASKR]	[CB]	[OR]
[SR]	[RLn]	[RMD]	[B?]	[<]
[ASR]	[RLCn]	[DBLR]	[WSIZE]	[>]
[RL]	[RRn]	[DBL÷]	[XOR]	[A] to [F]
[RLC]	[RRCn]	[DBL×]	[AND]	

SHOW {[HEX], [DEC], [OCT], [BIN]}

整数演算状態で使用できない機能は [.]，[EEX]，[1/x] だけです。

付　　録 C

電池，保証と修理について

電　　池

HP-16C は電池を 3 個使っています。アルカリ電池でごく普通の使いかたでは 6 か月以上使えます。HP-16C に付属の電池はアルカリ電池ですが，酸化銀電池（寿命が約 2 倍になります）も使用可能です。

新品のアルカリ電池ではプログラムの連続計算（電気を一番多く使っている状態です*）で 80 時間以上使えます。新品の酸化銀電池では同じ状態で 180 時間以上使えます。普通のプログラム計算ではプログラムの連続計算のときよりも電池の消耗が少くなります。数字などを表示しているだけのとき（キーを押したり，プログラム計算の途中でないとき）には電気を少ししか使わないからです。

HP-16C のスイッチを切っておくと不揮発性メモリーの内容が消えないようにほんのごく僅かに電流を流すだけなので，新品のアルカリ電池で約 1 年半，新品の酸化銀電池で約 2 年は持ちます。しかし HP-16C を長期間使わないときには電池の液漏れによる故障を防ぐために電池を外しておくことをお勧めします。

実際の電池の寿命は HP-16C をどう使うか，つまりプログラム計算の比率が多いかとか，手操作の比率が多いか，さらにどんな計算に使うのかによってかなり変化します。

HP-16C に付属の電池も，次表の交換用電池も充電できません。

---

* HP-16C の電流消費量はそのときの状態，つまりスイッチを切ってあるとき（不揮発性メモリーにだけ電流を流す），待機状態（表示しているだけ）のとき，動作中（プログラム計算や手計算中，あるいはキーを押した）のときによって変わります。スイッチを入れてあるときには待機中と動作中が混じりあった状態になります。そのため実際の電池の寿命は HP-16C がどの状態だったかによってかなり大幅に変わります。

ご 注 意

電池を充電しないでください。電池は高温のところに置かないでください。こうすると電池から液が漏れ出し，そのまま HP-16C 中に入れると故障の原因になります。電池を燃えている火の中に投げ込まないでください。こうすると爆発することがあります。同様に燃やすごみの中に入れないでください。

HP-16C 用の交換用電池は次の通りです。この記号は IEC（国際電気標準会議）の小型電池規格に入っているので，ヨーロッパ諸国でも同じ記号で通用します。

アルカリ電池	酸化銀電池
LR44 型	SR44 型（G-13 型とも言います）

次の電池はアメリカ製なので上の記号と異なりますが相当品です。

Eveready A76	Eveready 357
UCAR A76	UCAR 357
RAY-O-VAC RW82	RAY-O-VAC RS76 または RW42
Varta 4276	Duracell MS76 または 10L14
	Varta 541

電圧低下の症状

電池の電圧が低下すると，表示部の左下部分に＊印が点滅します。

アルカリ電池を使用のときは

- ＊印が見え始めてから連続で 2 時間以上使用できます*。
- スイッチを切っておけば＊印が見え始めてから約 1 か月間は不揮発性メモリーの内容が消えません。

---

* これは連続してプログラム計算をしているときの最低使用可能時間です。手操作で HP-16C を使っているときは，動作と待機が組み合わさっている状態なので，＊印が見え始めてからの使用可能時間はこれよりも長くなります。

酸化銀電池を使用のときは

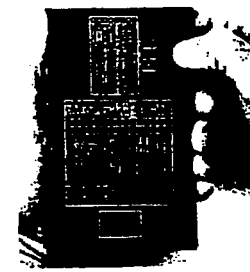
- ＊印が見え始めてから連続で15分以上使用できます。
- スイッチを切っておけば＊印が見え始めてから約1週間は不揮発性メモリーの内容が消えません。

新しい電池との交換

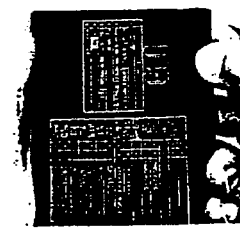
電池交換などで電池を外してもしばらくの間はHP-16Cの不揮発性メモリーの内容は消えません。（ただし電池を外す前に必ずスイッチを切ってください。）これは電池交換の間にプログラムやデータが消えないようにするためです。しかしあまり長時間電池を外したままにしておくと不揮発性メモリーの内容が消えてしまいます。

新しい電池と交換するには次のようにしてください。

1. HP-16Cのスイッチを切る。
2. HP-16Cを写真のように持って電池蓋の部分を外側に押しだして少しあけます。



3. 電池蓋の外側をつまんで全部あけます。



ご 注 意

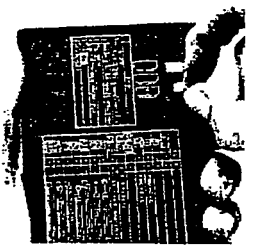
次の4～5で電池を外している間はキーにさわらないようにしてください。そうしないと不揮発性メモリーの内容が消えてしまったり，キーが働かなくなってしまうことがあります。

4. 電池側を下に向けて軽く振って，電池を手のひらに落してください。

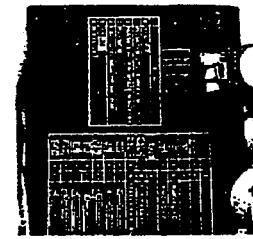
ご 注 意

この次で入れる電池は3個とも新しい電池にしてください。1～2個だけ新しいのにすると，古い電池から液漏れしてHP-16C本体を腐食させてしまうことがあります。
電池の方向（＋と－側）を間違えないでください。間違えると不揮発性メモリーの内容が消えてしまうことがあります。

5. 新しい電池を3個電池室へ入れてください。HP-16Cの裏側の図にあるように平らな面（＋マークがあります）がゴム足の方に向くようにしてください。

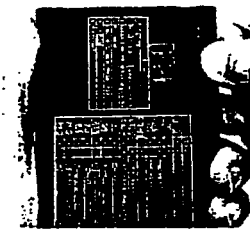


6. 電池蓋をつまんでHP-16C裏側の蓋のみぞに入れます。



7. 電池蓋が裏側と同じ高さになるようにしてから，電池蓋を完全にしめてください。

8. [ON]を押してスイッチを入れてください。もし不揮発性メモリーの内容が消えてしまったらPr Errorの表示になります。そのときはキーのどれかを押すとこの表示が消えます。それから必要なプログラムなどを入れてください。



動作の確認（自己診断）

HP-16Cのスイッチを入れても表示しないとか，正しく働かないときには次のようにして点検してください。

キーを押しても表示が変わらないとき

1. [ON]を押したまま[D]を押し[ON]を先に放します。こうするとXレジスタの内容が変わるので，後で[BSP]を押してください。

2. それでもまだキーを押しても変化しないときには，まず電池を外してもう一度入れ直してください。このときに電池の向きに気をつけてください。

3. それでもまだキーを押しても働かないときには電池を外してから，HP-16Cの電池側端子をゼムピンのようなものでほんのちょっとの間ショートしてみてください。(電池側端子の両側のプラスチック フィルムを外側に曲げないとショートできないかも知れません。) ショートするのはほんの短時間だけです。このようにすると不揮発性メモリーの内容が消えてしまうので，電池を入れ直してから [ON] を2−3回押す必要があると思います。

4. それでもまだ働かなかったら，新しい電池と交換してみてください。それでも働かなければそのHP-16Cは修理が必要です。

キーを押すと表示が変わるとき

1. スイッチを切ってから，[ON] を押したまま [×] を押します。

2. [ON] を先に放してから [×] も放してください。これはHP-16C内の電気回路の動作試験（自己診断）の開始です。回路が正常なら約15秒後（それまではrunningの文字を点滅します）に −8.8.8.8.8.8.8.8.8.8. の表示と状態表示の文字を全部表示します（ただし低電圧表示の＊は表示しません)*。このときにError 9の表示だったり，何も表示しなかったり，何か変な表示のときには修理が必要です。†

＊この試験をやるとHP-16Cでいつもは使わないものまで表示します。

† [ON]/[×] や [ON]/[÷] 試験のあとでError 9を表示しても，何とか使いたいときは21ページの方法で不揮発性メモリーをリセットしてください。

注 [ON] キーを押したまま [+] キーや [÷] キーを押しても同じように試験できます。この試験は主として製造工程や修理途中のときの方法です。

上記2の操作では正しく表示するけれども，計算途中で Error 表示をするようでしたら，Error 番号に応じてこの本の該当部分を読み直してください。それでもまだ分からないときはお買い上げの販売店や YHP に，手紙または電話でお問い合わせくださるとよいでしょう。

保証について

保証の内容

HP-16C は材質上や製造工程上の不具合に対してお買求めの日から1年間の保証をいたします。他の方に贈物としたり，転売された場合は自動的にその人に権利が移りますが，保証期間はあくまでも初めにお買求めの日から1年間だけです。保証期間中の修理・改造・部品交換などの費用とお客様への返送料は当社負担ですが，お客様から当社の修理担当への送料はお客様負担でお願いいたします。

• [ON] / [+] 試験は [ON] / [−] 試験と大体同じですが試験終了の表示が出ません。どれかのキーを押すと約15秒後に上のような表示になります。[ON] / [+] 試験はキーと表示を組合わせた試験です。[ON] を離すと表示の一部だけが見えます。試験を始めるには各段ごとに左から右に，第1段目が終ったら2段目というように全部のキーを順番に押してください。各キーを押すごとに表示が少しずつ変わります。全部のキーを順番通り正しく押すと 16 の表示になるはずです。（[ENTER] キーは3段目のときと4段目のときの両方に押してください。）計算機が正しく働かないときや，キー順序を間違えると Error 9 を表示します。なおこのようにキーを押す順番を間違えたときに Error 表示が出ますが，この場合には修理の必要はありません。試験を途中で打切るにはわざとキーの順序を間違えてください（当然のことですが Error 9 の表示になります）。Error 9 や 16 の表示はどれかのキーを押せば消えます。

保証を適用できない場合

通常使用外での故障（床面や水中などへの落下，誤使用や誤接続などを含む）の場合や当社以外で修理や改造したものは保証の対象外です。また保証期間内でも保証書がないとやはり保証の対象にはなりません。HP-16Cの電池は保証の対象外です。

修　　理

当社の修理担当の住所は下記の通りで直接こちらへお送りください。送料はお客様，返送料はYHP負担です。

〒229　神奈川県相模原市矢部1-27-15
横河・ヒューレット・パッカード㈱
コンピュータ サービス部
相模原サービス二課
電話　0427-59-1311（大代表）

外国へ出張の場合の修理先は製品に同封の多国語版のサービスカードをご参照ください。この場合でも保証書は国際的に通用いたします（多少の例外はあるかも知れません）のでお持ち下さい。この場合でも修理先への送料，通関手数料，関税等はお客様の負担になります。

修理料金

前記のように1年間の保証期間がありますが，保証期間切れや期間内でも保証書の添付がないときの修理は有料で機種ごとの定額制になっています。普通の使用方法以外による故障修理は別途料金になります。この場合は修理前に見積料金を連絡してご了解を得てから修理します。

修理完了後の保証期間

有料修理したものは3か月間の保証（使用部品と修理技術について）をいたします。修理保証書をご覧ください。

修理依頼品の発送について

もし修理が必要になりましたら次の方法でご返送くださるようお願いいたします。

- サービスカードに故障状況を含めて全部記入してください。
- 保証期間内の修理の場合は保証書または修理保証書を付けてください。

輸送途中の破損を防ぐために HP-16C をケースに入れ，サービスカード（または故障状況とお客様の住所，お名前，連絡電話番号を書いた紙）と保証書（保証期間内の場合だけ）を適当な箱に入れて当社の修理担当に郵便またはトラック便でお送りください。輸送途中の破損や紛失は保証の対象になりませんので書留小包にしたり，輸送保険を付けることをお勧めします。

現品が保証期間内でも期間満了後でも当社の修理担当までの送料はお客様のご負担でお願いします。

修理が完了しましたら保証期間内のものは送料 YHP 負担でお送りいたします。保証適用外のものは代金引換郵便にてお送りいたします(大部分の外国も同様です)。お問い合わせは前記の修理担当までお願いいたします。

温度範囲

- 使用時　0～55℃
- 保管時　−40～65℃

受信障害について

HP-16C は弱い電波を出すのでラジオやテレビ受信障害の原因となることがあります。HP-16C はアメリカ FCC 文書 20780 の規則 15 条J項のクラス B コンピュータの規定に合格していますが，これは一般住宅用受信機を対象にしています。したがって特殊なラジオやテレビに対しては保証しておりません。もし HP-16C がラジオやテレビ受信の障害原因と思われましたら HP-16C のスイッチを入り切りして確めてください。もし障害の原因なら次の方法を組み合わせて防止してください。

- 受信アンテナやフィーダの方向を変えてみる。
- HP-16C の使用方向を変えてみる。
- 受信機と HP-16C の距離を離してみる。

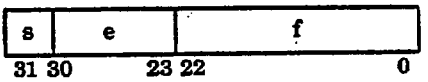
付　　録 D

表現形式変換プログラム

コンピュータによって数値の表現形式（フォーマット）が異なります。そこである数値を別の表現形式に変換することが必要になります。ここではIEEE（アメリカ電子電気技術者協会）規格案の実数2進数表現*とHP-16Cの実数表現との変換プログラムを記載します。

IEEEの表現

IEEE規格案の単精度実数の2進数指数部あり表現 (single-precision, floatingpoint binary format) は次の通りです。



全体で32ビットで，　符号のsが　1ビット
指数部のeが　8ビット
小数部分のfが　23ビット　です。

x（Xレジスタ内の内容）の値vを次のように解釈します。

(a) e=255でf≠0のときはvは数値ではない。

(b) e=255でf=0のときは$v=(-1)^s\infty$。

(c) 0<e<255のときは$v=(-1)^s2^{(e-127)}(1.f)$。

(d) e=0でf≠0のときは$v=(-1)^s2^{(-126)}(0.f)$。

(f) e=0でf=0のときは$v=(-1)^s0$。

HP-16Cの実数状態では次のように変換します。

---

* このIEEEの規格案はIEEEコンピュータ部会の実数表現委員会の754小委に提案されたものです。これはIEEE発行のComputer誌の1981年3月号51～62ページに掲載されました。

IEEE の数	X レジスタ	キャリー (フラグ4)	範囲外 (フラグ5)
0		0	0
−0		1	0
±∞	$\pm 9.999999999 \times 10^{99}$	1	1
その他の数値	前記(c)または(d)の通り	0	0
数値ではない	$1\ (-1)^s(0.f)2^{23}$	1	0

IEEE 表現から HP-16C 表現へ変換するプログラム

次のプログラムで IEEE 規格案の単精度実数の 2 進数表現の数値を HP-16C の実数 10 進数表現に変換できます。

キー操作	表 示	キー操作	表 示
[g] [LBL] B	001-43,22, b	[x⇄y]	018- 34
[HEX]	002- 23	8	019- 8
[f] SET COMPL [2's]	003- 42 2	[f] [MASKL]	020- 42 7
2	004- 2	[g] [x=y]	021- 43 49
0	005- 0	[GTO] 4	022- 22 4
[f] [WSIZE]	006- 42 44	[R↓]	023- 33
[f] [SL]	007- 42 A	[g] [x=0]	024- 43 40
[ENTER]	008- 36	[GTO] 3	025- 22 3
[ENTER]	009- 36	[x⇄y]	026- 34
[g] [x=0]	010- 43 40	1	027- 1
[GTO] 2	011- 22 2	8	028- 8
1	012- 1	[f] [SB]	029- 42 4
8	013- 8	[g] [LBL] 1	030-43,22, 1
[f] [MASKR]	014- 42 8	[g] [F?] 4	031-43, 6, 4
[f] [AND]	015- 42 20	[CHS]	032- 49
[f] [XOR]	016- 42 10	[x⇄y]	033- 34
[g] [LSTx]	017- 43 36	8	034- 8

キー操作	表　示	キー操作	表　示
f RLn	035-　42 E	g CLx	051-　43 35
9	036-　9	g x≠y	052-　43 0
7	037-　7	GTO 5	053-　22 5
-	038-　30	1	054-　1
g CF 4	039-43, 5, 4	4	055-　4
g LBL 2	040-43,22, 2	5	056-　5
f FLOAT .	041-42,45,48	ENTER	057-　36
g RTN	042-　43 21	g LBL 5	058-43,22, 5
g LBL 3	043-43,22, 3	x⇆y	059-　34
1	044-　1	g F? 4	060-43, 6, 4
8	045-　8	CHS	061-　49
f SB	046-　42 4	g ASR	062-　43 b
x⇆y	047-　34	x⇆y	063-　34
GTO 1	048-　22 1	g SF 4	064-43, 4, 4
g LBL 4	049-43,22, 4	GTO 2	065-　22 2
R↓	050-　33		

例

キー操作	表　示	(STATUS: 2-32-0000)
HEX 80000000	80000000 h	-0。
GSB B	0.000000 00	Cを表示。
HEX 7F800000		+∞。
GSB B	9.999999 99	CとGを表示。
HEX 00800000		$2^{-126}\times(1.00\ldots00)$。
GSB B	1.175494-38	
HEX 3F800001		$2^{0}\times(1.00\ldots01)=1+2^{-23}$。
GSB B	1.000000 00	
f CLEAR PREFIX	1000000119	

HP-16C 表現から IEEE 表現へ変換するプログラム

次のプログラムで HP-16C の実数 10 進数表現の数値を IEEE 規格案の単精度実数の 2 進数表現の数値に変換できます。フラグ 5 (範囲外, G) をセットしたら，変換結果が ±∞ です。(このプログラムと前のプログラムで使っているラベルは重複してないので，両方のプログラムを入れておけます。)

キー操作	表 示	キー操作	表 示
g LBL A	001-43,22, A	0	025- 0
f SET COMPL 2's	002- 42 2	f WSIZE	026- 42 44
HEX	003- 23	8	027- 8
g CF 4	004-43, 5, 4	0	028- 0
g CF 5	005-43, 5, 5	+	029- 40
g x≠y	006- 43 49	1	030- 1
g RTN	007- 43 21	8	031- 8
9	008- 9	f MASKL	032- 42 7
D	009- d	f AND	033- 42 20
+	010- 40	g F? 4	034-43, 6, 4
x⇄y	011- 34	g ISZ	035- 43 24
g CF 0	012-43, 5, 0	f SL	036- 42 A
g x<0	013- 43 2	RCL I	037- 45 32
g SF 0	014-43, 4, 0	F	038- F
g ABS	015- 43 8	F	039- F
x⇄y	016- 34	g x>y	040- 43 3
g x<0	017- 43 2	GTO 7	041- 22 7
GTO 9	018- 22 9	x⇄y	042- 34
1	019- 1	R↓	043- 33
+	020- 40	R↓	044- 33
g LBL 6	021-43,22, 6	g CLx	045- 43 35
STO I	022- 44 32	g R↑	046- 43 33

キー操作	表　示	キー操作	表　示
R↓	023-　33	g R↑	047-　43 33
2	024-　2	g SF 5	048-43, 4, 5
g LBL 7	049-43,22, 7	g LBL 9	062-43,22, 9
R↓	050-　33	g ABS	063-　43 8
f OR	051-　42 40	3	064-　3
g F? 0	052-43, 6, 0	0	065-　0
GSB 8	053-　21 8	g x≤y	066-　43 1
9	054-　9	x≷y	067-　34
f RRn	055-　42 F	R↓	068-　33
g CF 4	056-43, 5, 4	0	069-　0
g RTN	057-　43 21	x≷y	070-　34
g LBL 8	058-43,22, 8	f SB	071-　42 4
8	059-　8	+	072-　10
f SB	060-　42 4	0	073-　0
g RTN	061-　43 21	GTO 6	074-　22 6

例

キー操作	表　示	(STATUS: 2-32-0000)
f FLOAT .		
8 f EEX 72	8　72	
GSB A	7F800000 h	G を表示。オーバーフローして +∞。
f FLOAT .		
1.404 f EEX	1.404　00	
45 CHS	1.404　-45	
GSB A	1 h	
f FLOAT .		
3.141592654	3.141592654	π。
GSB A	40490Fdb h	

機能の要点と索引

HP-16C の各機能の要点は次の通りで，詳しくはそれぞれのページを見てください。機能は次のグループ別にまとめてあります。機能名が略語になっているものの大部分には正式名称も付けておきました。なお X, Y, Z, T レジスタを単に X, Y, Z, T と略記し，それぞれの内容を x, y, z, t と略記しました。

[ON]

HP-16C の電源スイッチ（16 ページ）。[−]と一緒に使うと不揮発性メモリーのリセット（21 ページ），[·]と一緒に使うと数値の桁区切りの変更(65 ページ)，[D]，[×]，[+]，[÷]と一緒に使うと HP-16C の動作確認（111 ページ）。

クリア

[BSP]

back space，1 字取り消し。計算状態では右端の数字を消し，数値入力を区切った後では表示全体を 0 にする（18 ページ）。プログラム入力状態では表示している命令を削除する（77 ページ）。

[CLx]

clear x，x をクリア。X の内容をクリアして 0 にする（18 ページ）。

CLEAR [PRGM] clear program memory, プログラム用メモリーをクリア。計算状態ではメモリー内容を消さずに 000 ラインに合わせるだけ。プログラム入力状態ではプログラム用メモリー全体を削除 (77 ページ)。

CLEAR [REG] clear register, レジスタをクリア。データ記憶レジスタ全部を 0 にする (72 ページ)。

CLEAR [PREFIX] 途中まで押した前操作キーを取り消す (17 ページ)。実数演算状態ではその有効数字 10 桁を一時的に表示する (62 ページ)。

数値入力

[0]~[9], [A]~[F] digit key, 数字キー。そのときの基数に応じたキーだけ使える (29 ページ)。

[.] decimal point, 小数点。実数のときだけ使える (61 ページ)。

[ENTER] 入れる。x を Y にコピーし, 数値入力を区切り次にスタック上昇しないようにする。複数の数値入力の区切りに使用する (23 ページ)。

[CHS] change sign, 符号を変える。そのときの状態に応じた x の補数, または x の正負符号を逆にする (31 と 61 ページ)。

[EEX] enter exponent, 指数部の入力。実数演算状態のときだけ使い, [EEX] の後にキーインする数値を 10 の指数部と見なす (61 ページ)。

スタック操作

[x⇄y] x exchange y, x と y の交換 (24 ページ)。

[R↓], [R↑] roll down, roll up, 下方に移動, 上方に移動。スタックの内容が一つずつ下か上に回転移動する (24 ページ)。

数値と表示の指定

[HEX], [DEC], [OCT], [BIN]	number base mode, 基数指定。整数演算状態にして，表示の基数を変える (29 ページ)。
SHOW { [HEX], [DEC], [OCT], [BIN] }	指定した基数で x を一時的に表示する (30 ページ)。
SET COMPL { [1's], [2's], [UNSGN] }	set complement mode, 補数状態の変更。1 の補数，2 の補数または符号なしの状態にする (31 ページ)。
[WSIZE]	word size, ワード長。x の絶対値 (0～64) を使ってワード長 (1～64) を変更するとスタック下降する (33 ページ)。
[WINDOW] {0～7}	x の指定した 8 桁部分を表示する (34 ページ)。
[<], [>]	scroll left, scroll right, 左に移動，右に移動。表示部の数値を左または右に 1 桁分移動して見えなかった部分を見えるようにする (35 ページ)。
[SF], [CF] {0～5}	set flag, clear flag, フラグのセット，フラグのクリア。指定したフラグをセットまたはクリアする (38 ページ)。
[STATUS]	HP-16C の状態確認。そのときの補数，ワード長，フラグの状態を一時的に表示する (38 ページ)。
[FLOAT] {0～9, [·]}	実数演算状態にして，小数点以下の表示桁数を指定した通り (0～9) にするか，指数部あり表示にする ([·] のとき)。整数演算状態から実数演算状態にするときには整数のときの x と y を使って実数の 10 進数が $y \cdot 2^x$ に等しくなるように変換して X レジスタに入れ，残りのスタックとラスト x はクリアする (59 ページ)。

数学演算

[+], [−], [×], [÷]	四則演算用で，計算後スタック下降する。[÷] は整

数演算のときには小数部分を表示しない（43ページ）。

[RMD] remainder, 剰余。|y| を |x| で割った余り（符号はyと同じ）を計算し，スタック下降する（45ページ）。

[√x], [1/x] square root, reciprocal, 平方根と逆数。xだけを使うのでスタック移動はない。[√x] は整数演算のときは小数部分を表示しない。[1/x] は実数演算状態だけしか使えない（46と64ページ）。

[DBL×], [DBL÷], [DBLR] double multiplication, division, remainder, 2倍長の乗算，除算，剰余。[DBL×] は2倍長の積を求めてXとYに入り，2倍長の被除数をYとZ（除数はX）に入れてから [DBL÷] または [DBLR] を求めると答がXに入る（55ページ）。

[ABS] absolute value, 絶対値。xの絶対値を求め，スタック移動はない（46ページ）。

ビット操作

[SL], [SR] shift left, shift right, 左にシフト，右にシフト。X中のビットを1ビットだけ左または右にシフト。端から出たビットはキャリーに入り，新しくXに入るビットは必ず0（49ページ）。

[ASR] arithmetic shift right, 符号はそのままで右にシフト。X中の全ビットを右に1ビット分シフトし，左端の符号ビット（1または2の補数状態のとき）はそのまま残る（49ページ）。

[RL], [RR] rotate left, rotate right, 左に回転，右に回転。X中のビットを1ビットだけ左または右に回転。片方から出たビットは他端に入ると同時にキャリーにも入る（51ページ）。

[RLC], [RRC] rotate left through carry, rotate right through carry, キャリーを通って左に回転，キャリーを通

って右に回転。[RL]、[RR]と同じ様に回転するが，片方から出たビットがキャリーに入り，それまでキャリーにあったビットが他端に入る（51ページ）。

[RLn]、[RRn]、[RLCn] multiple rotation，複数ビット回転。Yのビットがxビット分だけ（左または右，キャリーを通るか通らないか）回転し，スタック下降して新しいビットがXに入る（51ページ）。

[LJ] left-justify，左揃え。X内にあったビットをワード長に応じて左揃えしてYに入り，左揃えしたときのビットのシフト数がXに入る（49ページ）。

[MASKL]、[MASKR] mask left，mask right，左マスク，右マスク。xの絶対値のビット数で左または右揃えしたマスクを作ってXに入る（54ページ）。

[SB]、[CB] set bit，clear bit，ビットをセット，ビットをクリア。Y中のビットの内，xの絶対値番目のビットをセット（1にする）またはクリア（0にする）して，スタック下降する。ビットの番号は0〜ワード長−1で，0が右端（53ページ）。

[#B] number of bits，ビットの合計。X中のビットを合計してXに入る。スタック移動はない（54ページ）。

[NOT]、[OR]、[AND]、[XOR] logical（Boolean）operator，論理演算用。[OR]，[AND]，[XOR]はXとY内の2進数を使って論理演算をしてその答をXに入れる。（スタック下降する。）[NOT]はX内だけを使う（47ページ）。

メモリーと記憶

[STO]{0〜F、[I]、[(i)]} store，レジスタに入れる。xのコピーを指定したレジスタに入れる（70と73ページ）。

[RCL]{0〜F、[I]、[(i)]} recall，呼び出す。指定したレジスタの内容のコピーをXに入れる（70と73ページ）。

[x≷I], [x≷(i)]	インデクス・レジスタの部分参照。
[LSTx]	last x register, ラストxレジスタ。最後の演算直前にXに入っていた値をラストxレジスタから呼び出してXに入れる (24ページ)。
[MEM]	memory status, メモリー状態。次の二つを一時的に表示。(1)プログラム用メモリー配分を変えずに追加できる命令数。(2)データ記憶用として使えるレジスタ数 (69ページ)。
CLEAR [REG], CLEAR [PRGM]	clear storage registers, clear program memory, レジスタをクリア, プログラム用メモリーをクリア。123ページのクリアを参照。

インデクス・レジスタ

[I]	index register(R_I), インデクス・レジスタ。[GTO] や [GSB] でプログラムの流れを間接的に制御したり, [DSZ] や [ISZ] でループを制御するのにも使える記憶レジスタ (72ページ)。
[(i)]	indirect operations, 間接指定。どの記憶レジスタも間接的に指定でき, R_F よりも後の記憶レジスタはこの方法以外では使えない。インデクス・レジスタにレジスタ番号を入れておく必要がある(72ページ)。
[x≷I]	x exchange R_I, xとR_Iの交換。インデクス・レジスタの内容とxを交換する (73ページ)。
[x≷(i)]	インデクス・レジスタの内容で間接的に指定したレジスタの内容とxとを交換する (73ページ)。
[DSZ], [ISZ]	R_Iを使ってループを制御する。条件判断を参照 (129ページ)。

プログラム用

[P/R]	program/run mode, プログラム入力状態と計算

状態の切り替え。プログラム命令の入力用の状態と，プログラム走行や手操作用の計算状態の切り替え（76ページ）。

[LBL] {0～F} label，ラベル。プログラムの走行開始点の指定用（77ページ）。

[RTN] return，戻る。プログラム走行を止めて，メモリー中の000ラインに戻る。サブルーチン実行中のときは元の [GSB] 命令の次に戻るだけ（78ページ）。

[R/S] run/stop，走行兼停止。プログラム用メモリー中のそのときのラインから走行を始めるか，そのラインで走行を止める（80ページ）。

[PSE] pause，一時休止。プログラムを約1秒間止めてxを表示する（80ページ）。

[GTO] ラベル 指定したラベルにプログラムの実行を移す（ジャンプ）。プログラム可能（92ページ）。

[GTO] [·] *nnn* ライン番号がnnnのところに合わせる。プログラム不能（87ページ）。

[GSB] ラベル go to subroutine，サブルーチン実行。プログラム走行中では指定したラベルのサブルーチンへジャンプし，[RTN] に出合うと元の [GSB] 命令の次に戻ってプログラムを続行する。キー操作で [GSB] を使うと，指定したラベルからプログラム走行を開始する（92ページ）。

[SST] single step，1ステップ。プログラム入力状態ではプログラム用メモリー中を1ライン分，またはそれ以上（キーを押し続けている間）進む。計算状態ではそのときのプログラム・ラインを表示して実行し，次のラインに進む（87ページ）。

[BST] back step，後退。プログラム用メモリーを1ライン戻って，ライン番号とプログラム命令を表示する。プログラム入力状態でキーを押し続けると複数のラインを戻ることが出来る（88ページ）。

条件判断

[F?] {0~5}, [B?] {0~ワード長-1}	flag set?, bit set?, フラグのセットは？, ビットのセットは？ 指定したフラグまたはビットをセットしてあるかどうかの判定。セットしてあればプログラムはそのまま進行し，クリアしてあればその次の命令のラインだけを飛び越して先に進む (94 ページ)。
[x≦y], [x≦0], [x>y], [x>0], [x≠y], [x≠0], [x=y], [x=0]	conditional test, 条件判断。それぞれ x と 0 または y と比較する。この比較の結果がイエスならプログラムはそのまま進行し，ノーならその次の命令だけを飛び越して先に進む (93 ページ)。
[DSZ], [ISZ]	decrement and skip if zero, increment and skip if zero, カウンタを 1 減らして 0 だったら次を飛び越す，カウンタを 1 増やして 0 だったら次を飛び越す。インデクス・レジスタをループのカウンタとして使い，1 だけ増減してその結果が 0 だったらその次の命令だけを飛び越して先に進む (95 ページ)。

事項索引

ア行

カ行

サ行

タ行

ナ行

ハ行

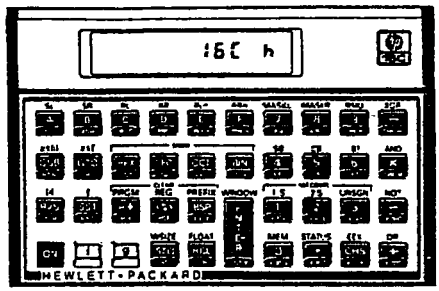
マ行

ヤ行

ラ行

アルファベット

HP-16C のキー配置と
不揮発性メモリー



自動メモリースタック

T
Z
Y
X　これを表示

ラスト x

プログラム用メモリー

7行単位で利用可能になる。

000–
001–
002–
003–
004–
005–
006–
007–
⋮
203–

プログラム用とデータ記憶用に合計203バイト（行）の記憶領域が使える。最初は記憶領域全体が記憶レジスタで，必要に応じてプログラム用メモリーに配分する。

7行目ごとのプログラム命令（001, 008, 015, …, 197）をキーインするたびに，データ記憶用の7バイトがプログラム用メモリーに切り替わる。7バイト分に必要なレジスタの個数はそのときのワード長によって変わる。

R_I

インデクス・レジスタ
（プログラム用にはならない）

記憶レジスタ

最大個数はワード長によって変わる。直接操作できるのは R_0～R_F だけ。

R_0
R_1
⋮
R_9
R_A
⋮
R_F
R_{10}
⋮
$R_{.F}$
R_{33}
R_{34}

横河・ヒューレット・パッカード株式会社

東部支社：〒168　東京都杉並区高井戸東三丁目29番21号
Tel. 03-331-6111（大代表）
西部支社：〒532　大阪市淀川区西中島5-4-20（中央ビル）
Tel. 06-304-6021（代表）
本社・工場：〒192　東京都八王子市高倉町9番1号
Tel. 0426-42-1231（大代表）

00016-90005 Rev.B - 3/86
JAPAN

Printed in Cana

Reorder No. or
Manual Part No.
00016-90005-E0386