

標準マクロ・サブルーチン使用手引

(Standard Macro/Subroutine User's Guide)

当手引きは、情報開発の標準マクロ・サブルーチンの
使用法を示したものである。

発効日 ： 1998年11月 2日
主管部門： 情報開発・システム開発センター

経緯

1987年3月：技術推進から委託され、'87年度 生産性委員会 - 標準部会が原案を作成し、関連部門（情報開発 全ライン管理者および 情報開発企画）の審議（レビュー）を受けて初版発行。

1998年11月：2000年問題への対応に伴い、情報開発標準についても全面的見直しを行い第二版発行。

標準マクロ・サブルーチン使用手引

1 まえがき	- 1 -
1.1 目的・概要	- 1 -
1.2 対象部門・対象業務	- 1 -
1.3 準拠および関連標準	- 1 -
1.4 部門別役割・責任	- 1 -
2 PL/I用マクロ・サブルーチン	- 2 -
2.1 標準エラー処理マクロ	- 2 -
2.2 INCBマクロ	- 5 -
2.3 INTSRT	- 6 -
2.4 BSRCH	- 8 -
2.5 BSRCH2	- 9 -
2.6 DTECNV1	- 10 -
2.7 DTECNV2	- 11 -
2.8 DTECNV3	- 12 -
2.9 DTECNV4	- 14 -
2.10 DTECNV5	- 14 -
2.11 DTECNV6	- 15 -
2.12 CHKDGT	- 16 -
2.13 SC02U22 (ENQ,DEQ)	- 17 -
2.14 SM00DYA (DYNAMIC ALLOCATION)	- 18 -
3 CSP用サブルーチン	- 20 -
3.1 CDTECNV1	- 20 -
3.2 CDTECNV2	- 20 -
3.3 CDTECNV3	- 21 -
3.4 CDTECNV4	- 21 -
3.5 CFLDCHK1	- 21 -
3.6 CBITCHK	- 23 -

1 まえがき

1.1 目的・概要

使用頻度が高く、共通に使用され、品質が保証された標準マクロ、サブルーチンの使用法を示し、開発・保守の効率向上を図る。

1.2 対象部門・対象業務

情報開発が作成あるいは保守する PL/I プログラムで、該当マクロやサブルーチンが使用可能な部分に適用する。

1.3 準拠および関連標準

なし

1.4 部門別役割・責任

(1)情報開発.システム開発センター

標準マクロ・サブルーチンの開発・保守を行う。

変更に応じて標準の改訂を行う。

(2)情報開発.各ソリューション

開発実施作業の効率化のため必要に応じて標準マクロ・サブルーチンを使用する。

2 PL/I用マクロ・サブルーチン

2.1 標準エラー処理マクロ

2.1.1 概要

標準エラー処理マクロは、PL/I プログラムの異常終了時に、OS 上の ABEND とするためのマクロである。 PL/I はプログラムが異常終了しても、OS 上の ABEND とならないため当マクロを使用して ABEND させる。

2.1.2 使用方法

()内番号は 説 明 項 目 番 号

```
SAMPLE : PROC OPTIONS(MAIN);  
(1) %INCLUDE SYSLIB2(SMONERR1);  
(2) 主処理ステートメント  
    IF ABEND させるべき、不都合な条件であれば  
    THEN  
        SMCONST2 ( ABENDの理由を記述する );  
    EOJ:                                /* END OF JOB PROCESS */  
(3) %INCLUDE SYSLIB2(SMHEAD3);  
(4) %XDESC= 'プログラムの説明を記述する';  
(5) %XPGMR= 'プログラマーの名前';  
(6) %INCLUDE SYSLIB(SMCTIME1);  
(7) SMCOUNT4 (C1,C2,C3,C4,C5,C6, . . . . . C14,END);  
    . . . . . その他任意の INFORMATION . . . . .  
(8) %INCLUDE SYSLIB2(SMCHEK5);  
(9) %XTWOWAY=3;  
(10) SMEND7 (H1,H2,H3, . . . . . ,H14,END);  
    END SAMPLE;
```

[説明]

- (1) 主処理ステートメントの前に表示通り書く (REQUIRED)
マクロの内容は、SYS1.CTRL (SMONERR1) を参照
 - (2) 主処理ステートメントにて下記に示す様なチェックを行い、結果に満足できなければ ABEND を発生させ PROGRAM を中止させるために表示する。
-

標準マクロ・サブルーチン使用手引

•SMCONST2(); の表示方法

- ❖ ABEND を起こさせた理由を 内に表示する。
- ❖ CHARACTER で MAX. 60桁以内のこと。
- ❖ Quotation でかこむ必要はない。
- ❖ () 内はカッコ、コンマ以外の EBCDIC ならなんでも良い。

[使用例] SMCONST2 (INPUT SEQ. ERROR);

•CHECK POINTS の使用例

- ❖ Table recordのInput Count Error Check
- ❖ Control Card Syntax Error Check
- ❖ Input File Sequence Error Check
- ❖ Input File の Type Error/Date Error Check
- ❖ Array の bound over/under Error Check
- ❖ NUMERIC FIELDの SIZE ERROR CHECK
- ❖ その他作業の中止を行いたい ERROR 発生 の CHECK

(3) 表示通り書く事 (REQUIRED)

以下の CODING は PROGRAM AUDIT REPORT を作成するためのものです。これ以降 OUTPUTするものはすべてこの REPORT に出ます。

- (4) %XDESC=' '; 最大 50字以内でプログラムの説明を記述する
- (5) %XPGMR=' '; 最大 20字以内でプログラマーの名前を記述する
- (6) %INCLUDE SYSLIB(SMCTIME1); このとおり書く

当ステートメントは、プログラムのコンパイル日時を SYSPRINT に出力し、確認に使用するためのものです。

(7) PROGRAM が ABEND した時に原因を発見する為に必要な情報を出力させる。

•情報の使用例

- ❖ INPUT/OUTPUT の RECORD COUNT
- ❖ Table record の INPUT COUNT
- ❖ Update した RECORD の COUNT
- ❖ Addition した RECORD の COUNT
- ❖ Delete した RECORD の COUNT
- ❖ Correct した RECORD の COUNT
- ❖ ERROR RECORD の COUNT
- ❖ 特定の Control Card の内容 (Date Card, Selection Card 等々)
- ❖ ARRAY の BOUND
- ❖ ABEND の時に取ってもらいたい ACTION 等々

•SMCOUNT4 (C1,C2,C3,C4,C5,C6, C14,END); の表示方法

- ❖ C1, C14 は FIXED BIN (31,0) として DCL する事。
- ❖ C1, C14 の各 COUNTER は MAX 14個表示出来る。
- ❖ 最後にはかならず END を表示する事、END は 14個に含めないで考える。

標準マクロ・サブルーチン使用手引

- ❖ REPORT 上には次の様に OUTPUT される。

C1 = C1 の内容

C2 = C2 の内容

:

C14 = C14 の内容

故に C1 ~ C14 はその COUNTER の内容が明確に判る様な変数名を付ける事。

[使用例] INPUT_COUNTER, ADDED_RECORD_VOL 等々

- ❖ C1,C2,・・・C14,END 各々の前後に blank があってはならない。

[使用例] (C1, C 2 , C3, END) は不可

- ❖ SMCOUNT4 の (・・・) 内の VARIABLE NAME, END はすべて CHARACTER CONSTANT STRING とみられる。 故に ONE CARD 以上にまたがる時は、SMCOUNT4 を複数行コーディングする。

- ❖ 各 VARIABLE はコンマで区切る事。

- ❖ COUNTER が MAX を越える時は SMCOUNT4 を追加すれば良い。

SMCOUNT4 は FIXED BIN(31,0) の COUNTER のみしか扱いません。

それ以外の OUTPUT は PL/I STATEMENT により SYSPRINT 上にこの場所で出力する。

•カウンタ類の出力方法は必ずしも SMCOUNT4 を使用しなくとも、PUT EDIT でも良い。・・・70.2.1「PL/I コーディング・ガイドライン」参照。

(8) 表示通り書く事 (REQUIRED)。

(9) 表示通り書く事 (REQUIRED)。

(10) 当 CODING の最後として指定する (REQUIRED)。

- SMEND7 (,END); の指定方法

- ❖ ON ERROR に入る ABEND 発生の際に、HEX. DUMP し CHECK に利用したい変数名 (ELEMENT VARIABLE) を 内に指定する。

- ❖ 変数名は CHARACTER STRING で CHAR() として必ず DCL する事。

- ❖ 上記以外のものは CONVERSION して与える事。

- ❖ 変数名は MAX. 14 個まで表示出来る。

- ❖ 変数名の最後には必ず END を表示する。

- ❖ 変数名, END の前後に blank があってはならない。

- ❖ 表示する変数名が無い時は、SMEND7(END); と表示する。

- ❖ 変数名, END はすべて CHARACTER CONSTANT STRING とみられる。

故に ONE CARD 以上にまたがる時は、SMEND7 を複数行コーディングする。

- ❖ 各変数名はコンマで区切る事。

2.1.3 注意事項

- ON ERROR はコーディングしてはならない。その他の ON CONDITION は自由に設定して良い。
 - ❖ 基本的には、異常時には全て PROGRAM を ABEND させるが、例外的に ON CONDITION がおこった時に、NORMAL に PROGRAM を終了させたい時は ON CONDITION 内に GO TO XEOJ; を表示する。
- USER が DCL してはならない 変数名 と ラベル常数
 - ❖ 変数名・・・XONCODE, XPAGE, XABEND, XUSER, SS01SUB
 - ❖ ラベル常数・・・XEOJ, XEXIT
- (1)～(10) の MACRO の指定順序は変化させてはならない。

2.2 INCBマクロ

2.2.1 概要

INCBマクロは、CRD または、URD から PL/I の DCL を作るマクロである。

2.2.2 使用方法

- BASED の場合：

INCB WWWWW,[Y or N],RRRRRRR,PP,C,SSSSSS,SSSSSS,・・・・・・；

WWW

Structure が Base する Work Area Name。頭 1 桁を 'W' にして、Total 5桁であること。

Y or N:

'Y' の時、

この Work Area が STATIC で作られ、長さは指定された CRD の Total Length が計算して取られ、この WORK AREA (STATIC) の Address が、指定された CRD (BASED) のポインターにアサインされる。

'N' の時

Work Area をとらないし、CRD レイアウト (BASED) のポインターもアサインしない。

RRRRRRR

CRD の RECORD NAME (MEMBER NAME)

PP (OPTION)

各 Field Name の頭につける Prefix。(Structure Name の頭にもつく)

Input Structure	I1 ~ I9, J1 ~ J9, K1 ~ K9
Output Structure	O1 ~ O9, P1 ~ P9, S1 ~ S9

標準マクロ・サブルーチン使用手引

Work Structure	W1 ~ W9, Z1 ~ Z9, B1 ~ B9
----------------	---------------------------

C (OPTION)

Work Area (STATIC) を、FIXED 0, CHAR b で Initialize する
 'C' の時。

SSSSSS (OPTION):

CRD レイアウトの中、当 PGM で使用する Field のみを SSSSSS,SSSSSS,
 SSSSSS と指定すると、指定された Field のみ Generate され、残りの
 使用しない Field は Length が計算され正しく Dummy out される (CRD 上の
 Record Length および Field の配列は変わらない)。

SSSSSS を指定しないと、該当レイアウトの全 Field を展開して DCL が作成
 される。

INCB の行にコメントは書けない。

•BASED 以外の場合：

INCB UNDEF,N,RRRRRR,PP,C,SSSSSS,SSSSSS ;

UNDEF :

'UNDEF' と指定する。

N :

BASED 以外の場合は、常に 'N'

(以下 BASED に同じ)

2.3 INTSRT**2.3.1 概要**

Subroutine INTSRTは、ある Table 中の Fixed Length (Max = 256 Byte) の
 Records を In-Core Sort する Subroutine である。

2.3.2 CALL Statement Format

CALL INTSRT(table,control-field,n);

table

一次元の Constant Table を指定する。Table が Array なら Array Name,
 Structure なら Structure Name, Array of Structure なら Array of Structure
 Name を指定する。

control-field

Sort Control Field Name を指定する。Data Attribute は、Character でなけ
 ればならない。Table が Array なら Array Name, Structure または Array of

Structure の場合は Sort したい Field を指定する。これは 1 Field しか指定できないので、2 つ以上に分割されている場合は連続 Field になるよう Arrange し直し、DEF 等により 1 Control Field にしなければならない。

n

Table 中実際に存在する Record 数を FIXED BIN(15) の Variable Name で指定する。

2.3.3 使用例

1 . Array の場合

```
DCL  TBL(100)  CHAR(5),  
      ICNT  FIXED  BIN(15);  
      :  
CALL  INTSRT(TBL,TBL,ICNT);
```

2 . Structure または Array of Structure の場合

```
DCL   1  TBL(100),  
      2  TC  CHAR(5),  
      2  TD  CHAR(5),  
      ICNT  FIXED  BIN(15);  
CALL  INTSRT ( TBL,TC,ICNT);
```

[注] 1 の TBL は各 Element の Ascending 順に、2 の TBL は TC の順に並べかえられる。

2.3.4 INTSRT使用上の注意

MVS/XA で INTSRT を Call する際に Control Field を DEF で指定すると User ABENDする現象が起きています。(下記のケース)

```
DCL  1  A(10),  
      2  B  CHAR(2),  
      2  C  CHAR(3),  
      2  D  CHAR(5),  
      X  CHAR(5) DEF  A  POS(1);  
CALL  INTSRT(A,X,CNT);
```

このようなコーディングをしている場合は DEF を使用しないようにプログラムを変更してください。

つまり、SORT を行う対象となる ARRAY, ARRAY OF STRUCTURE および STRUCTURE 内に含まれる FIELD 名を CONTROL FIELD に指定しないと INTSRT は正しく実行されません。

このトラブルは、NON-XA の OS による PROTECTION 機能が弱かったため ABEND しなかったものが、MVS/XA になり顕在化したものです。

2.4 BSRCH

2.4.1 概要

Subroutine BSRCH は、ある Value が Table の何番目に存在するか、Binary Search Technique を用いて検索する Subroutine である。

通常 INTSRT したテーブルをサーチするので、INTSRT の制限であるレコード長 (max = 256 bytes) を守る。

2.4.2 CALL Statement Format

```
CALL BSRCH(item,table,n,i);
```

Item

Search した Value を Character として DCL したものに Assign し、その Variable Name を指定する。

従って、もしその Item が PIC '99...9' 等で DCL されている時は Character で Define し、その Defined Item を指定しなければならない。比較の桁数は、その Item の Length Attribute による。

table

一次元の Constant Table を指定する。

Table が Array なら Array Name, Array of Structure なら Array of Structure 内の、Item に対応する Field Name を指定する。この Table の Attribute は CHAR でも PIC でもかまわない。

n

Table 中に実際に存在する Constant の組数を FIXED BIN(15) で DCL された Variable Name で指定する。

従って、CALL する前にその組数をこの Variable に Assign しておかなければならない。

i

Search の結果が Assign される FIXED BIN(15) の Variable を指定する。Not Found の場合は Zero となる。

Found の場合は、Start を 1 として何番目の Constant と一致したかを番号でこの Variable に Assign する。

2.4.3 使用例

•Arrayの場合

```
DCL  1  TBL,
      2  #1  CHAR(20)  INIT('A001A002A003B001B002'),
      2  #2  CHAR(20)  INIT('C001C002C003D001D002'),
      TBLX(10)  CHAR(4)  DEF  TBL;
DCL  ITEM      CHAR(4);
      :
K = 10;
CALL  BSRCH(ITEM,TBLX,K,I);
IF  I=0  THEN  GOTO  NOTFND;
      :
      :
```

•Array of Structure の場合

```
DCL  1  TBL,
      2  TBLY(100),
      3  TC  CHAR(5),
      3  TD  CHAR(5),
      TBLX(100)  CHAR(10)  DEF  TBL;
DCL  C      CHAR(5)  INIT('11AAA');
      :
ICNT=88;
CALL  BSRCH(C,TC,ICNT,I);
IF  I = 0  THEN  GOTO  ERROR;
      :
      :
```

2.5 BSRCH2

2.5.1 概要

- 今までの BSRCH の機能に加えて、
- 指定された KEY をもとにテーブルをサーチして見つからない場合に指定された KEY より小さい範囲で一番近い KEY (直前の KEY) の位置を返す。
- RANGE SEARCH, GENKEY SEARCH などに使用できる。

2.5.2 Call Statement Format

```
CALL  BSRCH2  ( SEARCH_KEY,TBL,TBL_SIZE,RET1,RET2)
```

標準マクロ・サブルーチン使用手引

•SEARCH_KEY, TBL, TBL_SIZE, RET1 は BSRCH と全く同じ

RET1

Match しない時は 0 を返す

RET2

FIXED BIN(15,0) 指定された KEY に等しいか、小さい範囲で一番近い key の位置

2.5.3 使用例

TBL

	KEY	DATA	SEARCH_KEY	RET1	RET2	
1	E		A	0	0	
2	F		F	2	2	見つかった時
3	G					
4	J		H	0	4	Jの位置を返す
5	L					
6	M					
7	N					

2.6 DTECNV1

2.6.1 概要

DTECNV1 は、Calendar Date を Julian Date に変換する Subroutine である。

Alias Name : RPD8EAE

2.6.2 CALL Statement Format

CALL DTECNV1(date1,date2,function);

date1

- Julian Date に変換される Calendar Date を表す Item を指定する。
- Item は CHAR(6) で Declare する。 Character 以外で Declare されている場合は、Character で Define し、その Defined Item を指定する。
- 形式は DDMMYY
 - DD Day
 - MM Month
 - YY Year

date2

- 変換された Julian Date を表す Item を指定する。
- Item は FIXED(5) で Declare する。

- 形式は YYDDD+
YY : Year
DDD : Day

function

- Item は CHAR(1) で Declare する。 Character 以外で Declare されている場合は、Character で Define し、その Defined Item を指定する。
- Function の値が示す意味は次のとおりである。

変換前 (Input) - - - - 変換の条件

- 0 : 単純変換を行う。(Data ERROR はどの場合も Check される。)
- 1 : 金曜日か否かを Check する。
- 2 : 次の金曜日に変換する。
- 3 : 金曜日か否かを Check し、金曜日以外の場合は次の金曜日に変換する。

変換後 (Output) - - - - 結果の状態

- 0 : 単純変換の成功を示す。
- 5 : 金曜日であったことを示す。(変換は成功)
- 6 : 次の金曜日に変換されたことを示す。
- 9 : Data Error を示す。

Input と Output の対応関係

Input	Output
0	0 or 9
1	0 , 5 or 9
2	6 or 9
3	5 , 6 or 9

[注] 変換に失敗するとDate2には値 00000 が返される。

2.6.3 使用例

```
DCL CAL_DATE CHAR(6) INIT('081598'),
DATE1 CHAR(6),
DATE2 FIXED(5),
FUNC CHAR ( 1 ) INIT('0');
```

```
DATE1 = CAL_DATE;
CALL DTECNV1(DATE1,DATE2.FUNC);
IF FUNC='9' THEN PUT DATA(DATE1);
```

2.7 DTECNV2

2.7.1 概要

DTECNV2 は、Julian Date を Calendar Date に変更する Subroutine である。

Alias Name : RPD8EAF

2.7.2 CALL Statement Format

```
CALL DTECNV2(date1,date2,function);
```

date1

- Calendar Date に変換される Julian Date を表す Item を指定する。
- Item は FIXED(5) で Declare する。
- 形式は YYDDD+

date2

- 変換された Calendar Date を表す Item を指定する。
- Item は CHAR(6) で Declare する。 Character 以外で Declare されている場合は、Character で Define し、その Defined Item を指定する。
- 形式は DDMMYY

function

DTECNV1 と同様

[注] 変換に失敗すると Date2 に値 '000000' が返される。

2.7.3 使用例

```
DCL JUL_DATE FIXED(5) INIT(98320),  
      DATE1 FIXED(5),  
      DATE2 CHAR(6),  
      FUNC CHAR(1) INIT('0'),  
      DATE1X CHAR(3) DEF DATE1;  
      :  
      :  
DATE1 = JUL_DATE;  
CALL DTECNV2(DATE1,DATE2,FUNC);  
IF FUNC='9' THEN CALL HEXDUMP(DATE1X);
```

2.8 DTECNV3

2.8.1 概要

DTECNV3 は、Program で指定した Julian Date に User の指定する日数を加算または減算した後の Julian Date を求める際に使用する Subroutine である。

当 Subroutine を使用することによって、Date Card, Parameter Card など

Preparation に人手の介入を避け得る Case がある。

Alias Name : RPD8EAH

2.8.2 CALL Statement Format

`CALL DTECNV3(Juldate,Increment,Function);`

Juldate

- Julian Date を表す Item を指定する。
- Item は、FIXED(5) で Declare する。
- 形式は YYDDD

Increment

- Julian Date に加算 (または減算) すべき日数を表す Item を指定する。
- Item は FIXED(5) で Declare する。
- 形式は DDDDD ±

例 実際の Coding は次のようになる。

- 200日前の Julian Date を求める場合 INCREMENT=-200;
- 200日後の Julian Date を求める場合 INCREMENT=200;

function

- Item は CHAR(1) で Declare する。
- Function の値が示す意味は次のとおりである。

変換前 (Input) - - - - 変換の条件

0: 変換を行う。

変換後 (Return) - - - - 結果の状態

0: 変換が正しく行われたことを示す。

9: Errorを示す。

Input と Output の対応関係

Input=0 Return=0 or 9

[注] 変更がうまくいくと juldate には increment で指定した日数の合計が返される。

2.8.3 使用例

```
DCL JUL-DATE FIXED(5) INIT(98201),  
      JULDATE FIXED(5),  
      INCREMENT FIXED(5),  
      FUNC CHAR(1) INIT('0'),  
      JULDATEX CHAR(3) DEF JULDATE;
```



```
INCREMENT=-200;  
JULDATE=JUL-DATE;  
CALL DTECNV3(JULDATE,INCREMENT,FUNC);  
IF FUNC='9' THEN CALL HEXDUMP(JULDATEX);
```

この結果 FUNC='0' のとき JULDATE=98001 となる。

2.9 DTECNV4

2.9.1 概要

DTECNV4 は 2 つの日付の期間 (日数) を算出する。

2.9.2 CALL Statement Format

```
CALL DTECNV4(Jul-1,Jul-2,Def,Func);
```

Jul-1

End Date, Julian, FIXED(5)

Jul-2

Start Date, Julian, FIXED(5)

Def

結果の日数, 符号付, FIXED(5)

Func

(DTECNV3 の場合と同じ), CHAR(1)

2.9.3 使用例

```
DCL JULD1 FIXED(5) INIT(98201),  
      JULD2 FIXED(5) INIT(98025),  
      DAYS FIXED(5),  
      FUNC CHAR(1) INIT('0');  
CALL DTECNV4(JULD1,JULD2,DAYS,FUNC);  
IF FUNC='9' THEN ~
```

2.10 DTECNV5

2.10.1 概要

DTECNV5 は西暦を和暦に変換する Subroutine である。

2.10.2 CALL Statement Format

CALL DTECNV5(yy,date1,date2,元号,元号init);

yy (INPUT VALUE)

- 和暦に変換される西暦の上2桁を表す ITEM を指定する。
- ITEM は CHAR(2) で DECLARE する。
- この ITEM を BLANK にした場合、
date1 で指定された西暦の下2桁が 50 以下の時には '20' が
51以上の時には '19' が指定されたものとする。

date1 (INPUT VALUE)

- 和暦に変換される DATE を表す ITEM を指定する。
- ITEM は CHAR(6) で DECLARE する。
- 形式

yy	:	西暦の下2桁
mm	:	月
dd	:	日

date2 (OUTPUT VALUE)

- 和暦に変換された DATE を表す ITEM を指定する。
- ITEM は CHAR(6) で DECLARE する。
- 形式は yymmdd
- 変換に失敗すると、'000000' が戻される。

元号 (OUTPUT VALUE)

- 変換された和暦の元号を表す ITEM (漢字) を指定する。
- ITEM は CHAR(10) で Declare する。

元号init (OUTPUT VALUE)

- 変換された和暦の元号の INITIAL を表す ITEM を指定する。
- ITEM は CHAR(1) で DECLARE する。
- ITEM の値が示す意味は次の通りである。

'M'	:	明治
'T'	:	大正
'S'	:	昭和
'H'	:	平成

2.11DTECNV6

2.11.1概要

DTECNV6 は和暦を西暦に変換する Subroutine である。

2.11.2CALL Statement Format

CALL DTECNV6(元号init,date1,yy,date2);

元号init (INPUT VALUE)

- 西暦に変換される元号の INITIAL を表す ITEM を指定する。
- ITEM は CHAR(1) で DECLARE する。
- ITEM の値が示す意味は次の通りである。

'M' : 明治

'T' : 大正

'S' : 昭和

'H' : 平成

blank・・・date1 で指定された和暦の年が 51 以上の場合、昭和とみなし、
50 以下の場合には平成とみなす。

date1 (INPUT VALUE)

- 西暦に変換される DATE を表す ITEM を指定する。
- ITEM は CHAR(6) で DECLARE する。
- 形式

yy : 和暦の年

mm : 月

dd : 日

yy (OUTPUT VALUE)

- 西暦に変換された年の上2桁を表す ITEM を指定する。
- ITEM は CHAR(2) で DECLARE する。

date2 (OUTPUT VALUE)

- 西暦に変換された DATE を表す ITEM を指定する。
- ITEM は CHAR(6) で DECLARE する。
- 形式は

yy : 西暦の下2桁

mm : 月

dd : 日

- 変換に失敗すると、'000000' が戻される。

2.12CHKDGT

2.12.1概要

CHKDGT は、ある Item の Check Digit を計算して、その値を返す Subroutine である。

2.12.2CALL Statement Format

CALL CHKDGT(item,check-digit);

item

- Check Digit の計算の対象となる Item を指定する。
- Item は Character で Declare する。 Character 以外で Declare されている場合は、Character で Define し、その Defined Item を指定しなければならない。

check-digit

- 計算の結果求められた Check Digit を受け取る Item を指定する。
- Item は CHAR(1) で Declare する。 Character 以外で Declare されている場合は、Character で Define し、その Defined Item と指定しなければならない。

2.12.3 Check Digitの算出方式

Modules 10 方式

- (1)Unit Position と、そこから左へ向かって一つおきの位置にある数をそれぞれ 2 倍する。
- (2)それぞれの積と 2倍されなかった数とを合計する。ただし、積が 10 以上の場合は 1 と 10 の位に展開し、それぞれの位の数を加算する。
- (3)Total を、それに等しいかまたはおおきくかつ 10 の倍数であるような数の中で最小のものから差し引く。
- (4)その差が Check Digit となる。

$$\begin{array}{r}
 \begin{array}{cccccc}
 1 & 0 & 9 & 9 & 5 & \\
 \times & & \times & & \times & \\
 (1) & 2 & & 2 & & 2 \\
 (2) & 2 & + & 0 & + & 1 + 8 + 9 + 1 + 0 & = 21 \\
 (3) & & & & & 30 \\
 & & & & & - 21 \\
 (4) & & & & & \boxed{9}
 \end{array} \\
 \boxed{1 \quad 0 \quad 9 \quad 9 \quad 5 \quad 9}
 \end{array}$$

2.12.4 使用例

```

DCL CODE CHAR(5) INIT('09420'),
ITEM CHAR(5),
DGT CHAR(1);
ITEM=CODE;
CALL CHKDGT(ITEM,DGT);

```

2.13 SC02U22 (ENQ,DEQ)

2.13.1概要

同時使用する可能性のあるファイルを更新する時は、当サブルーチンを使用して、ファイルが同時に更新され破壊されるのを防ぐ。

2.13.2使用方法

•CODING例

```
DCL DDNAME CHAR(8);  
DDNAME='XXXXXXXX';          DD名をセットする  
CALL ENQ(DDNAME);  
READ FILE(XXXXXXXX) INTO (XXX) KEY (XXXX);  
XXX=XX;  
REWRITE FILE(XXXXXXXX) FROM (XXX) KEY (XXXX);  
CALL DEQ(DDNAME);
```

•LINK EDITの方法

SC02U22をINCLUDEする

```
//LKED.SYSIN DD *  
INCLUDE SUBLIB(SC02U22)  
//SUBLIB DD DSN=SA00.DS.NCAL,DISP=SHR
```

[注] DISP=SHR の時は RESERVE(ENQ), RELEASE(DEQ) となる。

2.14SM00DYA (DYNAMIC ALLOCATION)

2.14.1概要

SM00DYA はデータセットのダイナミック・アロケーションを行うサブルーチンである。このサブルーチンを使用すると、どのデータセットを使用しているかという情報が JCL で判断できないため、使用は必要性のある部分にのみ限定し、基本的には使用しない。使用にあたっては、事前にシステム開発センター SA00 担当者の承認を得る。

機能

- 既存 DATA SET の ALLOCATION
- NEW DATA SET の ALOOCATION
- ALLOCATED DATA SET の DE-ALLOCATION

2.14.2使用方法

標準マクロ・サブルーチン使用手引

(1) PARM の SET UP このレイアウトは SYS1.DCL(SM00DYN) にある。

(M):Mandatory (O):Option

ELEMENT	既存D/SのALLOC	NEW D/SのALLOC	UNALLOCATION
CAOXD	b	b	U(M)
CDISP	SHR,MOD,OLD(M)	NEW(M)	NO-CHECK
NDDSA	ANY(M)	ANY(M)	ANY(M)
NDSNB	ANY(M)	ANY(M)	ANY(M)
CSPAC	N/A	TRK,CYL,BLK(M)	N/A
QSPPR	N/A	ANY(M)	N/A
QSPSC	N/A	ANY(O)	N/A
NVOLS	N/A	ANY(M)	N/A
NUNIT	N/A	ANY(M)	N/A
CPROT	N/A	Y or b(0)	N/A
QBLKS	N/A	ANY(M)	N/A
QLREL	N/A	ANY(M)	N/A
QRECF	N/A	F,FB,FBA,VB,VBA,U(M)	N/A

(2) CALL方法

```

DCL 1 PARM STATIC UNALIGNED,
%INCLUDE SM00DYN;

DCL PTRX PTR;

PTRX=ADDR(PARM);

CALL SM00DYA(PTRX);

```

(3) Other Information

ATTR	MNEMONIC	UNIT	KEY	#	LEN	PARM
DDNAME	DALDDNAM	0001		0001	n	ANY
DSN	DALDSNAM	0002		0001	n	ANY
DISP-1st	DALSTATS	0004		0001	0001	04(NEW)
DISP-2nd	DALNDISP	0005		0001	0001	02(CATLG)
DISP-3rd	DALCDISP	0006		0001	0001	04(DELETE)
SPACE-TRK	DALTRK	0007		0000	-	-
SPACE-CYL	DALCYL	0008		0000	-	-
SPACE-BLK	DALBLKLN	0009		0001	0003	ANY(BLOCK Len)
SPACE-PRIME	DALPRIME	000A		0001	0003	ANY
SPACE-SEC	DALSECND	000B		0001	0003	ANY
VOL SER	DALVLSER	0010		0001	0006	xxxxxx
UNIT	DALUNIT	0015		0001	0005	ANY
PROTECT=YES	DALPROT	0061		0000	-	-
DCB-BLKSIZE	DALBLKSZ	0030		0001	0002	xx
DCB-LRECL	DALLRECL	0042		0001	0002	xx
DCB-RECFM	DALRECFM	0048		0001	0001	xx X'90'-FB

3 CSP用サブルーチン

IMS 下の CSP370/RS のみサポートする。

3.1 CDTECNV1

3.1.1 概要

CDTECNV1 は、Calendar Date を Julian Date に変換する Subroutine である。

3.1.2 CALL Statement Format

```
CALL CDTECNV1 cdate,jdate,return_cd (NONCSP,NOMAPS;
```

cdate (INPUT)	CHAR(6)
jdate (OUTPUT)	PACK(5)
return_cd (INPUT/OUTPUT)	CHAR(1)

3.1.3 使用例

•主作業域での定義

名前	LVL	TYPE	SCOPE	LENGTH	BYTES	DESCRIPTION
CDATE	77	CHA	LOCAL	6	6	CALENDAR DATE
JDATE	77	PACK	LOCAL	5	3	JULIAN DATE
RETC	77	CHA	LOCAL	1	1	RETURN CODE

•プロセスでの定義

```
MOVE '010491' TO CDATE;  
MOVE '0' TO RETCD;  
CALL CDTECNV1 CDATE,JDATE,RETC (NONCSP,NOMAPS;  
IF RETCD = '0';  
:  
ELSE;  
:  
END;
```

3.2 CDTECNV2

3.2.1 概要

CDTECNV2 は、Julian Date を Calendar Date に変更する Subroutine である。

3.2.2 CALL Statement Format

```
CALL CDTECNV2 jdate,cdate,return_cd (NONCSP,NOMAPS;
```

jdate (INPUT)	PACK(5)
cdate (OUTPUT)	CHAR(6)
return_cd (INPUT/OUTPUT)	CHAR(1)

3.3 CDTECNV3

3.3.1 概要

CDTECNV3 は、Program で指定した Julian Date に User の指定する日数を加算または減算した後の Julian Date を求める際に使用する Subroutine である。

当 Subroutine を使用することによって、Date Card, Parameter Card など Preparation に人手の介入を避け得る Case がある。

3.3.2 CALL Statement Format

```
CALL CDTECNV3 jdate,cdate,return_cd (NONCSP,NOMAPS;
```

jdate (INPUT/OUTPUT)	PACK(5)
cdate (INPUT)	PACK(5)
return_cd (INPUT/OUTPUT)	CHAR(1)

3.4 CDTECNV4

3.4.1 概要

CDTECNV4 は 2 つの日付の期間 (日数) を算出する。

3.4.2 CALL Statement Format

```
CALL CDTECNV4 jdate1,jdate2,duration,return_cd (NONCSP,NOMAPS;
```

jdate1 (INPUT)	PACK(5)
jdate2 (INPUT)	PACK(5)
duration (OUTPUT)	PACK(5)
return_cd (INPUT/OUTPUT)	CHAR(1)

3.5 CFLDCHK1

3.5.1 概要

CFLDCK1は、ある Field に含まれる Character が許された範囲内のものであるかどうかを、Byte 単位に Check する Subroutine である。

3.5.2 CALL Statement Format

```
CALL CFLDCK1 Parm1,Parm2,return_cd,  
length1,length2,length3 (NONCSP,NOMAPS;
```

Parm1 (INPUT)

1. Check したい Field を、Character として DCL したもので指定する。
2. 従って、もしその Item が PIC 99・・・9 等で DCL されている時は Character で Define し、その Defined Item を指定しなければならない。

Parm2 (INPUT)

1. Parm1 で指定した Field を、どの Model Character で Check するかを Byte 単位で指定する。
2. Model Character は Quotation (' ') で囲むか、または Model Character を他に DCL する場合、Character として DCL し、その Area Name を指定する。なお Field を Separation する場合は、Period (.) を使用する。
3. Byte 単位で Field を Check するため、Model Character の個数は Parm1 で指定した Field に含み得る Byte 数と同じでなければならない。

return_cd (OUTPUT)

1. Parm1 で指定した Field 内の Character が、Parm2 で指定した条件に合致したかどうかの Return Code が返されるてくる。
2. Character で DCL した Area を指定する。
Parm2 で Field Separation を行わない場合は、1 Byte の Area を指定し、Field Separation を行った場合には、Separation の数と同じ Byte 数の Area が必要である。
例えば、Separation の数が 3 であれば、Return Code のための Area は 3 Byte 必要となる。

length1 (INPUT)

BIN(9) parm1 の長さ

length2 (INPUT)

BIN(9) parm2 の長さ

length3 (INPUT)

BIN(9) return-cd の長さ

3.5.3 Model Character Table

標準マクロ・サブルーチン使用手引

Model	Character	Condition
	A	A~Z,Blank
	B	Blank(X'40')
	C	Not Blank(Not X'40')
	I	0~9,A~I
	K	0~9,A~Z,*
	L	A~Z
	M	0~9,A~Z,Blank
	N	0~9,A~Z,Not Blank
	P	Packed Decimal,Valid Sign
	R	0~9,J~R,-0(マルチパンチ・マイナス0)
	T	A~I,J~R,-0(マルチパンチ・マイナス0),+0(マルチパンチ・プラス0)
	V	0~9,*
	W	0~9,* ,Blank
	X	Any Character (No Check)
	Y	0~9,Blank (該当FieldにBlankがあれば、Subroutineはこれを0に置き換える。)
	Z	0~9,Leading Blanks (該当Fieldの先行BlankをSubroutineは0に置き換える。)
	8	0~9,Blank
	9	0~9

3.5.4 Return Code

All 2

ParmX の指定に誤りがある。

例えば、Parm 1 で指定した Model に含み得る Byte 数と、Parm 2 で指定した Model Character の個数が同じでない。あるいは、Parm 2 の Field Separation の数と、Parm 3 で指定した Return Code の Byte 数が同じでない。

0

Parm1 で指定した Field 内の Character が、Parm 2 で指定した条件に合致した。(該当 Field は有効である。)

1

Parm 1 で指定した Field 内の Character が、Parm 2 で指定した条件に合致しなかった。(該当 Field は無効である。)

9

Model Character Table に無い Model Character を指定した。

3.6 CBITCHK

3.6.1 概要

1 バイトのキャラクター (CHAR(1) または HEX(2)) の n ビット目が '1' または '0' であるかを判断する。

3.6.2 CALL Statement Format

CALL CBITCHK checked-char,bit-position,retcd (NONCSP,NOMAPS;

変数	説明	属性	POSSIBLE VALUE
checked-char	1バイトのチェックされるキャラクター	CHAR(1),HEX(2)	ANY
bit-position	何ビット目をチェックするか	BIN(4)	1-8
retcd	リターン・コード	CHAR(1)	0:OFF 1:ON 9:ERROR

3.6.3 使用例

'01001100' というビット列からなる1バイトのキャラクターの2番目のビットをチェックする。

	CALL前	CALL後
hecked-char	01001100	01001100
bit-position	2	2
retcd	9	1