Date Data Type

Defining Date Data Type Program Variables:

```
 *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D OrderDate       S                D

D                 DS
D DueDate                           D   INZ
D  DueYear                          4   Overlay(DueDate)
D  DueMonth                         2   Overlay(DueDate:6)
D  DueDay                           2   Overlay(DueDate:9)
```

Dates should be initialized when subfields are defined.

Internally, date variables have a length of 4 bytes.
Externally, they have an apparent length that varies according to the format.

Date Formats Supported by the DATFMT Keyword:

Name	Description	Format	Length
*ISO	International Standards Organization	yyyy-mm-dd	10
*EUR	IBM European Standard	dd.mm.yyyy	10
*JIS	Japanese Industrial Standard CE	yyyy-mm-dd	10
*USA	IBM USA Standard	mm/dd/yyyy	10
*JUL	Julian	yy/ddd	6
*MDY	Month/Day/Year *	mm/dd/yy	8
*DMY	Day/Month/Year *	dd/mm/yy	8
*YMD	Year/Month/Day *	yy/mm/dd	8

* Separator characters can be specified with the DATFMT keyword: / - , . &(blank)

Default Formats and Overrides:

- The default date format is *ISO.
- The H-spec DATFMT keyword overrides the default.
- The D-spec DATFMT keyword overrides the H-spec.
- The C-spec format code specified in Factor 1 for MOVE operations overrides the D- and H-specs.

Moving Non-Date Fields to Date Fields

```
D                 DS
D DueDate                         D   INZ
D  DueYear                        4   OverLay(DueDate)
D  DueMonth                       2   OverLay(DueDate:6)
D  DueDay                         2   OverLay(DueDate:9)
```

V3R1:

```
 * Move signed numeric data to date data
D NumMDY          S               6S 0 INZ(011597)
...
C     *MDY          MOVE      NumMDY        DueDate
```

```
 * Move packed numeric data to date data
D PackYMD         S               6P 0 INZ(970115)
...
C     *YMD          MOVE      PackYMD       DueDate
```

```
 * Move edited character data to date data
D EditMDCY        S                10   INZ('01/15/1997')
...
C     *USA          MOVE      EditMDCY      DueDate
```

```
 * Move packed Julian to date data
D NumJUL          S               5P 0 INZ(97015)
...
C     *JUL          MOVE      NumJUL        DueDate
```

```
 * Move character Julian to date data
D CharJUL         S                6   INZ('97/015')
...
C     *JUL          MOVE      CharJUL       DueDate
```

Factor 1 must indicate the date format when non-date fields are moved to date fields.

Character fields must include a separator character (can be blank)

V3R2/6:

```
 * Move packed date with century digit (0=19, 1=20) to date data
D PackCYMD        S               7P 0 INZ(0970115)
...
C     *CYMD         MOVE      PackCYMD      DueDate
```

```
 * Move character date with century digit and no separators to
 * date data
D CharCYMD        S                7   INZ('0970115')
...
C     *CYMD0        MOVE      CharCYMD      DueDate
```

Century digit is allowed for YMD formats.

Character fields with a century digit can be moved by specifying "0" (zero) as the separator.

V3R7:

```
 * Move unedited character with no separators to date data
D CharMDY         S                6   INZ('011597')
...
C     *MDY0         MOVE      CharMDY       DueDate
```

The "0" separator is not supported for other formats.

Time Data Type

Defining Time Data Type Program Variables:

```
     *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D SendTime        S                T

D                 DS
D InTime                           T   INZ
D  InHour                          2   Overlay(InTime)
D  InMin                           2   Overlay(InTime:4)
D  InSec                           2   Overlay(InTime:7)
```

Internally, time variables have a length of 3 bytes.
Externally, they have an apparent length of 8 bytes.

Date Formats Supported by the DATFMT Keyword:

Name	Description	Format	Length
*ISO	International Standards Organization	hh.mm.ss	8
*EUR	IBM European Standard	hh.mm.ss	8
*JIS	Japanese Industrial Standard CE	hh:mm:ss	8
*USA	IBM USA Standard	hh:mm AM(or PM)	8
*HMS	Hours/Minutes/Seconds *	hh:mm:ss	8

* Separator characters can be specified with the TIMFMT keyword: **: . , &** (blank)

Default Formats and Overrides:

The default time format is *ISO.

The H-spec TIMFMT keyword overrides the default.

The D-spec TIMFMT keyword overrides the H-spec.

The C-spec format code specified in Factor 1 for MOVE operations overrides the D- and H-specs.

Moving Non-Time Fields to Time Fields

```
D                 DS
D InTime                          T   INZ
D  InHour                         2   Overlay(InTime)
D  InMin                          2   Overlay(InTime:4)
D  InSec                          2   Overlay(InTime:7)
```

```
 * Move signed numeric data to time data
D NumHMS          S               6S 0 INZ(123000)

C     *HMS        MOVE      NumHMS        InTime
```

Factor 1 must indicate the time format when non-time fields are moved to time fields

```
 * Move packed numeric data to time data
D PackHMS         S               6P 0 INZ(123000)

C     *HMS        MOVE      PackHMS       InTime
```

```
 * Move edited character data to time data
D EditHMS         S               8    INZ('12 30 00')

C     *HMS&       MOVE      EditHMS       InTime
```

Character fields must include a separator character (can be blank)

```
 * Move USA style time to time data
D CharUSA         S               8    INZ('12:30 PM')

C     *USA        MOVE      CharUSA       InTime
```

Timestamp Data Type

Defining Timestamp Data Type Program Variables:

```
 *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D StartTime       S                 Z

D                 DS
D EntDate                           Z   INZ
D  EntYear                        4     Overlay(EntDate)
D  EntMonth                       2     Overlay(EntDate:6)
D  EntDay                         2     Overlay(EntDate:9)
D  EntHour                        2     Overlay(EntDate:12)
D  EntMin                         2     Overlay(EntDate:15)
D  EntSec                         2     Overlay(EntDate:18)
D  EntMicSec                      6     Overlay(EntDate:21)
```

Internally, timestamp variables have a length of 10 bytes.
Externally, they have an apparent length of 26 bytes.

There is Only One Timestamp Format (*ISO):

yyyy-mm-dd-hh.mm.ss.nnnnnn

Moving Non-Timestamp Fields to TImestamp Fields:

V3R1:

```
  * Move edited character data to timestamp data
D EditTS          S               30    INZ('1997-01-15-12.30.00.000000    ')
...
C                   MOVEL     EditTS         EntDate
```

```
  * Move numeric data to timestamp data
D NumTS           S               20S 0 INZ(19970115123000000000)
...
C                   MOVE      NumTS          EntDate
```

V3R7:

```
  * Move unedited character data to timestamp data
D CharTS          S               20    INZ('19970115123000000000')
...
C     *ISO0         MOVE      CharTS         EntDate
```

*ISO is optional in Factor 1. Use "0" to specify that the source field does not contain separators.

Testing Non-Date/Time/Timestamp Data

```
CL0N01Factor1++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C      *MDY          TEST(D)                FieldName                 80
```

Optional: Factor 1 determines which format to test the data against. Required for non-timestamp data.

Must be blank.

The D,T, and Z extenders must be used for numeric or character data.

Required: Identifies the field to be tested.

Required: Error indicator turns on if Factor is invalid.

```
D BadDate          S                6S 0 INZ(022997)
...
C      *MDY          TEST(D)                BadDate                   80
```

```
D BadTime          S                8    INZ('25:62:00')
...
C      *HMS          TEST(T)                BadTime                   80
```

Invalid values may go undetected without a TEST operation.

```
D GoodTime         S                8    INZ('05:25 PM')
...
C      *USA          TEST(T)                GoodTime                  80
C      *HMS          TEST(T)                GoodTime                  80
```

```
D BadTimeSt        S               26S 0 INZ(19960232256200123456)
...
C                    TEST(Z)                BadTimeSt                 80
```

No Factor 1 format is necessary for Timestamp test

```
H DATFMT(*USA) TIMFMT(*USA)

D NewTime          S                T    INZ(T'05:25 PM')
D NewTimeSt        S                Z    INZ(Z'1997-01-28-17.25.00.000000')

D                  DS
D NewDate                           D    DATFMT(*ISO) INZ(D'01/28/1997')
D  NewYr                            4    Overlay(NewDate)
D  NewMo                            2    Overlay(NewDate:6)
D  NewDay                           2    Overlay(NewDate:9)

D BirthDate1       S                6S 0 INZ(123139)
D BirthDate2       S                6S 0 INZ(010140)


 * BirthDate1 = 12/31/39.  NewDate will be 12/31/2039.
C      *MDY          MOVE      BirthDate1     NewDate
 * BirthDate2 = 01/01/40.  NewDate will be 01/01/1940.
C      *MDY          MOVE      BirthDate2     NewDate

C                    MOVE      35             NewDay
C                    TEST                     NewDate                   80
```

Date and Time fields must be initialized to the H-spec format.

Subfields allow bad values to be introduced into date, time, or timestamp data.

Assumptions for two-digit years:
<=39 = 20xx
>=40 = 19xx

Date Addition

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D-HiLoEq -
C     InvDate        ADDDUR     30:*days      DueDate                  80
```

Optional:
Date, time, timestamp, literal, constant, subfield, array, or array element.

Required:
(Duration: Duration Code)
Numeric field, array element, or constant (with 2 decimal positions). See duration code table.

Required:
Date, time, timestamp, array, or array element.

Optional:
Error indicator turns on if Factor 1, Factor 2, or Result are invalid.

Duration Codes	
Full Name	**Short Name**
*YEARS	*Y
*MONTHS	*M
*DAYS	*D
*HOURS	*H
*MINUTES	*MN
*SECONDS	*S
*MSECONDS	*MS

```
 * Set the DueDate for 30 days after the Invoice Date:
C     InvDate        ADDDUR     30:*days      DueDate                  80

 * Extend the policy expiration date by 6 months:
C                    ADDDUR     6:*m          ExpDate

 * Extend the date of full depreciation by 3 years:
C     RepairDate     ADDDUR     3:*years      DeprDate

 * Convert TimeStamp to Greenwich Mean Time by adding 6 hours:
C                    ADDDUR     6:*hours      TimeStamp

 * All of these will yield the same result: February 28, 1997
C     D'1997-01-28'  ADDDUR     1:*months     ExpDate
C     D'1997-01-29'  ADDDUR     1:*months     ExpDate
C     D'1997-01-30'  ADDDUR     1:*months     ExpDate
C     D'1997-01-31'  ADDDUR     1:*months     ExpDate
```

Date Subtraction

Calculating a New Date, Time, or Timestamp:

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C     TimeStamp     SUBDUR     8:*hours     GMT                    80
```

Optional:
Date, time, timestamp, literal, constant, subfield, array, or array element.

Required:
(Duration: Duration Code)
Numeric field, array element, or constant (with 0 decimal positions).

Required:
Date, time, timestamp, array, or array element.

Optional:
Error indicator turns on if Factor Factor 2, or Rest are invalid.

```
 * Convert TimeStamp to Greenwich Mean Time by subtracting 8 hours:
C     TimeStamp     SUBDUR     8:*hours     GMT                    80

 * What was the date one week ago?
C     CurrDate      SUBDUR     7:*days      OldDate                80

 * All of these will yield the same result: February 28, 1997
C     D'1997-03-28' SUBDUR     1:*months    ExpDate
C     D'1997-03-29' SUBDUR     1:*months    ExpDate
C     D'1997-03-30' SUBDUR     1:*months    ExpDate
C     D'1997-03-31' SUBDUR     1:*months    ExpDate
```

Date Subtraction

Calculating a Duration:

```
CL0N01Factor1++++++Opcode&ExtFactor2++++++Result++++++++Len++D+HiLoEq
C     CurrDate      SUBDUR    DueDate       Diff:*days              80
```

Required:
Date, time, timestamp, literal, constant, subfield, array, or array element.

Required:
(Duration:Duration Code)
Numeric field, array element, or constant (with 0 decimal positions).

Optional:
Error indicator turns on if Factor 1, Factor 2, or Result are invalid.

Required:
Date, time, timestamp, literal, constant, subfield, array, or array element.

```
  * How many days overdue is the unpaid invoice?
C     CurrDate      SUBDUR    DueDate       Diff:*days              80

  * How many days has the item been on back order?:
C     CurrDate      SUBDUR    OrdDate       DaysBack:*d             80

  * How long was the telephone call in terms of minutes?:
C     EndTime       SUBDUR    StartTime     Len:*mn                 80
```

Extracting Date Elements

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C                   EXTRCT    CurrDate:*m   M                 2 0  80
```

Must be blank.

Required:
(Source Field: Duration Code)
Date, time, or timestamp field, subfield, array, array element, or table.

Required:
Numeric, character, subfield, array or table element.

Optional:
Error indicator turns on if Factor is invalid.

```
 * Extract the month number from the current date:
C                   EXTRCT    CurrDate:*m   M                 2 0  80
```

```
 * Extract the calendar month and day from the Julian date:
C                   EXTRCT    JulDate:*m    MM                2
C                   EXTRCT    JulDate:*d    DD                2
```

```
 * Override the '1940-2039' assumption on two-digit years:
C     *MDY          Move      BTHDAT        BirthDate
C                   Extrct    BirthDate:*y  Year              4 0
C                   If        Year>2019
C                   SubDur    100:*years    BirthDate
C                   EndIf
```

Moving Date Fields to Non-Date Fields

```
D DueDate         S               D
D CurrTime        S               T
D TimeStamp       S               Z
```

```
  * Move DueDate back to numeric database field in MDY format:
C     *MDY          MOVE      DueDate       NUMMDY
```

```
  * Move DueDate back to character field in Julian format:
C     *JUL          MOVE      DueDate       JULDAT
```

```
  * Move date portion of TimeStamp to DueDate, then to character
  * database field in *USA format:
C                   MOVE      TimeStamp     DueDate
C     *USA          MOVE      DueDate       OLDDUE
```

```
  * Move current time to character field in 12-hour clock format:
C     *USA          MOVE      CurrTime      CURTIM
```

```
  * Move DueDate to new field in the job's date format:
C     *JOBRUN       MOVE      DueDate       NEWDAT
```

Free-Form Expressions

The EVALuate Operation:

```
     *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+.
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C                    EVAL      Result=Expression
```

Operation Precedence in Expressions:

1. ()
2. Built-in functions, user-defined functions
3. unary+, unary -, NOT
3. ** (exponentiation)
4. *, /
5. binary +, binary -
6. =, <>, >, >=, <, <=
7. AND
8. OR

```
  * Calculate NetPay:
C                    EVAL(H)   NetPay=RegHrs*RegRate+OTHrs*OTRate-
C                              FedTax-FICA-StTax-
C                              ThisTax-ThatTax-OtherTax-MoreTax
```

```
  * Calculate the Hypotenuse of a Triangle (A²+B²=C²):
C                    EVAL      SideC2 = SideA**2 + SideB**2
C                    SQRT      SideC2         SideC
```

```
  * Calculate the monthly payment on a loan:
D Payment          S                9P 2
D Principal        S               11P 2
D Int              S               30P15
D Term             S                4P 0

  * Principal loan amount is $100,000:
C                    Eval      Principal=100000
  * Annual interest is 9.5%:
C                    Eval      Int=9.5
  * Term is 30 years at 12 periods per year:
C                    Eval      Term=30*12

  * Convert annual interest rate to periodic (monthly) rate:
C                    Eval      Int=Int/12
  * Move interest rate decimal point two positions to the left:
C                    Eval      Int=Int/100

  * Calculate the periodic (monthly) payment:
C                    Eval(H)   Payment=
C                              Principal*(Int/(1-((Int+1)**(-Term))))
```

$$\text{Payment} = \text{Principal}\left(\frac{\text{Int}}{1-(\text{Int}+1)^{-\text{Term}}}\right)$$

Built-In Functions (BIFs)

V3R1 BIFs:

%ELEM – Get Number of Elements.

Syntax: %ELEM(table-name)
%ELEM(array-name)
%ELEM(multiple-occurrence-ds-name)

Example:

```
 * Define the Product Array:
D PRODARRAY       S              9    DIM(12) CTDATA PERRCD(6)
 * Initialize PRODELEM with the number or PREDARRAY elements:
D PRODELEM        S               2P 0 INZ(%ELEM(PRODARRAY))
```

%SIZE – Get Size in Bytes.

Syntax: %SIZE(variable-name)
%SIZE('literal')
%SIZE(array-or-table-name:*ALL) *ALL is optional
%SIZE(multiple-occurrence-ds-name:*ALL) *ALL is optional

Example:

```
 * Initialize PRODSIZE with the size of the entire PRODARRAY
D PRODSIZE        S               3P 0 INZ(%SIZE(PRODARRAY:*ALL))
```

%SUBST – Get Substring.

Syntax: %SUBST(variable:start:length) Length defaults to the string's remaining length.
%SUBST('literal':start:length)
expressions allowed

Example:

```
 * Put the 1st 80 characters of TEXT into String:
C                   EVAL      String=%SUBST(TEXT:1:80)
 * Put the 1st 40 characters of TEXT into the 2nd 40 bytes of String:
C                   EVAL      %SUBST(String:41:80)=%SUBST(TEXT:1:40)
```

%TRIM – Trim Blanks at Edges.

Syntax: %TRIM(variable-name)

Example:

```
 * Append first and Last name fields together with one blank in bet
C                   EVAL      PayTo=%TRIM(FNAME)+' '+%TRIM(LNAME)
```

%TRIML – Trim Leading Blanks.

%TRIMR – Trim Trailing Blanks.

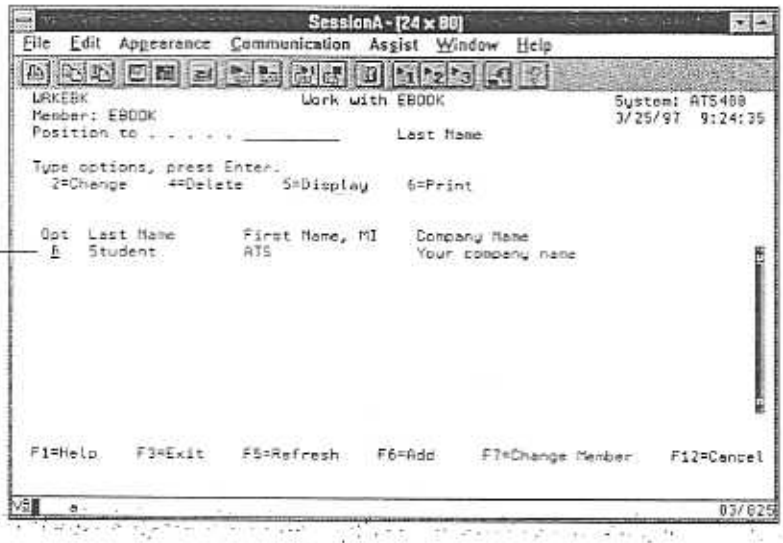
%ADDR – Get Address of Variable.

Syntax: %ADDR(variable-name)
%ADDR(variable-name(index))
%ADDR(variable-name(expression))

%PADDR – Get Procedure Address.

Syntax: %PADDR(procedure-name)

The Current PRTEBKR Program



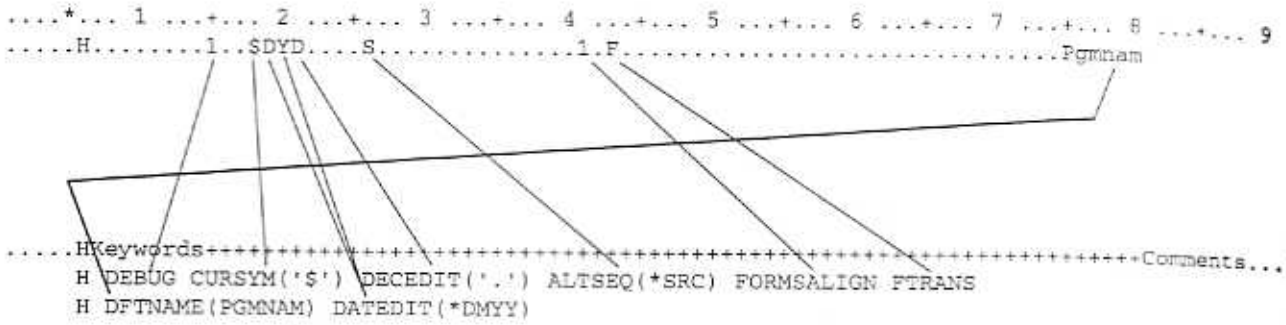
WRKEBK

```
C*****************************************************************
C* Process user selected options.                                *
C*****************************************************************
C     *IN70          IFEQ      *ON
C                    READC     EBKSFL                               51
C     *IN51          DOWEQ     *OFF
C     OPT            CASEQ     '2'           SRCHGN
C     OPT            CASEQ     '4'           SRDLTN
C     OPT            CASEQ     '5'           SRDSPN
C     OPT            CASEQ     '6'           SRPRTN
C                    ENDCS
                       :
                       :
                       :
C*****************************************************************
C* SRPRTN - Option 6 (Print Details).                            *
C*****************************************************************
C     SRPRTN         BEGSR
C                    CALL      'PRTEBKR'     ENBR
C     ENBR           PLIST
C                    PARM                    ENENBA
C                    ENDSR
```

PRTEBKR (CLP)

```
/*****************************************************************/
/* REPLACE THIS WITH THE ILE RPG VERSION OF PRTEBKR.             */
/*****************************************************************/
            PGM        PARM(&ENENBA)
            DCL        VAR(&ENENBA) TYPE(*CHAR) LEN(7)
/*****************************************************************/
/* Tell user via message that this component is not yet implemented. */
            SNDMSG     MSG('The PRTEBKR component has not been +
                         implemented yet.') TOUSR(*REQUESTER)
            ENDPGM
```

Control (H) Specifications



Continuation Examples:

```
H DFTNAME(
H PGMNAM
H )
```

```
H DFTNAME
H (PGMNAM)
```

Source Entry Utility Formats:

RPG III (RPG/400)

Format H:

```
.....H.....1..CDYI....S.....1.F.....Pgm-id
```

RPG IV (ILE RPG)

Format H:

```
.....HKeywords+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Comments...
```

Free-Form Expressions

The EVALuate Operation:

```
     *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+.
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++
C                   EVAL       Result=Expression
```

Operation Precedence in Expressions:

1. ()
2. Built-in functions, user-defined functions
3. unary+, unary -, NOT
3. ** (exponentiation)
4. *, /
5. binary +, binary -
6. =, <>, >, >=, <, <=
7. AND
8. OR

```
     * Calculate NetPay:
C                   EVAL(H)    NetPay=RegHrs*RegRate+OTHrs*OTRate-
C                              FedTax-FICA-StTax-
C                              ThisTax-ThatTax-OtherTax-MoreTax
```

```
     * Calculate the Hypotenuse of a Triangle (A²+B²=C²):
C                   EVAL       SideC2 = SideA**2 + SideB**2
C                   SQRT       SideC2         SideC
```

```
     * Calculate the monthly payment on a loan:
D Payment         S                9P 2
D Principal       S               11P 2
D Int             S               30P15
D Term            S                4P 0

     * Principal loan amount is $100,000:
C                   Eval       Principal=100000
     * Annual interest is 9.5%:
C                   Eval       Int=9.5
     * Term is 30 years at 12 periods per year:
C                   Eval       Term=30*12

     * Convert annual interest rate to periodic (monthly) rate:
C                   Eval       Int=Int/12
     * Move interest rate decimal point two positions to the left:
C                   Eval       Int=Int/100

     * Calculate the periodic (monthly) payment:
C                   Eval(H)    Payment=
C                              Principal*(Int/(1-((Int+1)**(-Term))))
```

$$\text{Payment} = \text{Principal}\left(\frac{\text{Int}}{1-(\text{Int}+1)^{-\text{Term}}}\right)$$

Built-In Functions (BIFs)

V3R1 BIFs:

%ELEM – Get Number of Elements.

Syntax: %ELEM(table-name)
%ELEM(array-name)
%ELEM(multiple-occurrence-ds-name)

Example:

```
 * Define the Product Array:
D PRODARRAY       S              9    DIM(12) CTDATA PERRCD(6)
 * Initialize PRODELEM with the number or PREDARRAY elements:
D PRODELEM        S              2P 0 INZ(%ELEM(PRODARRAY))
```

%SIZE – Get Size in Bytes.

Syntax: %SIZE(variable-name)
%SIZE('literal')
%SIZE(array-or-table-name:*ALL) *ALL is optional
%SIZE(multiple-occurrence-ds-name:*ALL) *ALL is optional

Example:

```
 * Initialize PRODSIZE with the size of the entire PRODARRAY
D PRODSIZE        S              3P 0 INZ(%SIZE(PRODARRAY:*ALL))
```

%SUBST – Get Substring.

Syntax: %SUBST(variable:start:length) Length defaults to the string's remaining length.
%SUBST('literal':start:length)
expressions allowed

Example:

```
 * Put the 1st 80 characters of TEXT into String:
C                   EVAL      String=%SUBST(TEXT:1:80)
 * Put the 1st 40 characters of TEXT into the 2nd 40 bytes of String:
C                   EVAL      %SUBST(String:41:80)=%SUBST(TEXT:1:40)
```

%TRIM – Trim Blanks at Edges.

Syntax: %TRIM(variable-name)

Example:

```
 * Append first and Last name fields together with one blank in bet
C                   EVAL      PayTo=%TRIM(FNAME)+' '+%TRIM(LNAME)
```

%TRIML – Trim Leading Blanks.

%TRIMR – Trim Trailing Blanks.

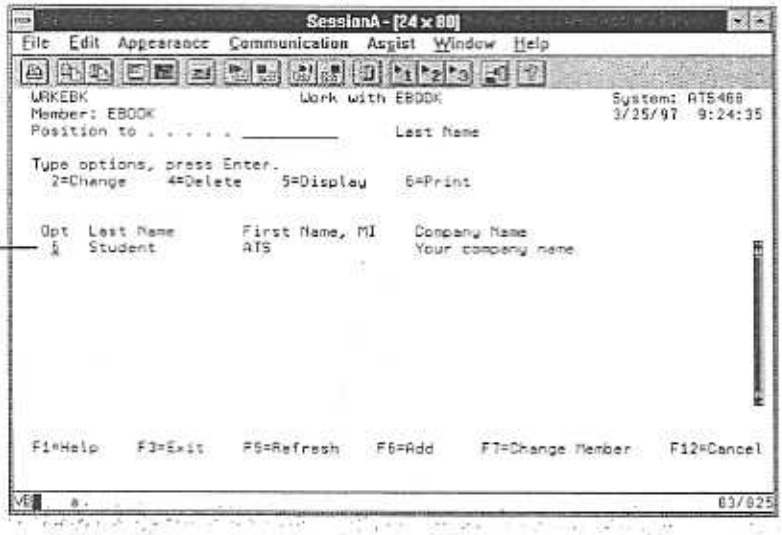
%ADDR – Get Address of Variable.

Syntax: %ADDR(variable-name)
%ADDR(variable-name(index))
%ADDR(variable-name(expression))

%PADDR – Get Procedure Address.

Syntax: %PADDR(procedure-name)

The Current PRTEBKR Program



WRKEBK

```
C*****************************************************************
C* Process user selected options.                                *
C*****************************************************************
C     *IN70          IFEQ      *ON
C                    READC     EBKSFL                               51
C     *IN51          DOWEQ     *OFF
C     OPT            CASEQ     '2'           SRCHGN
C     OPT            CASEQ     '4'           SRDLTN
C     OPT            CASEQ     '5'           SRDSPN
C     OPT            CASEQ     '6'           SRPRTN
C                    ENDCS
                       :
                       :
                       :
C*****************************************************************
C* SRPRTN - Option 6 (Print Details).                            *
C*****************************************************************
C     SRPRTN         BEGSR
C                    CALL      'PRTEBKR'     ENBR
C     ENBR           PLIST
C                    PARM                    ENENBA
C                    ENDSR
```

PRTEBKR (CLP)

```
/*****************************************************************/
/* REPLACE THIS WITH THE ILE RPG VERSION OF PRTEBKR.             */
/*****************************************************************/
             PGM        PARM(&ENENBA)
             DCL        VAR(&ENENBA) TYPE(*CHAR) LEN(7)
/*****************************************************************/
/* Tell user via message that this component is not yet implemented. */
             SNDMSG     MSG('The PRTEBKR component has not been +
                          implemented yet.') TOUSR(*REQUESTER)
             ENDPGM
```