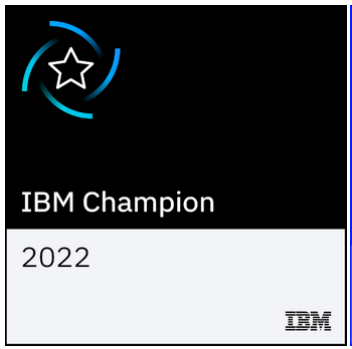
Recent additions to the modern RPG language

Simon Hutchinson

RPGPGM.COM

IBM Champion for Power 2022

IBM i Community Advocate



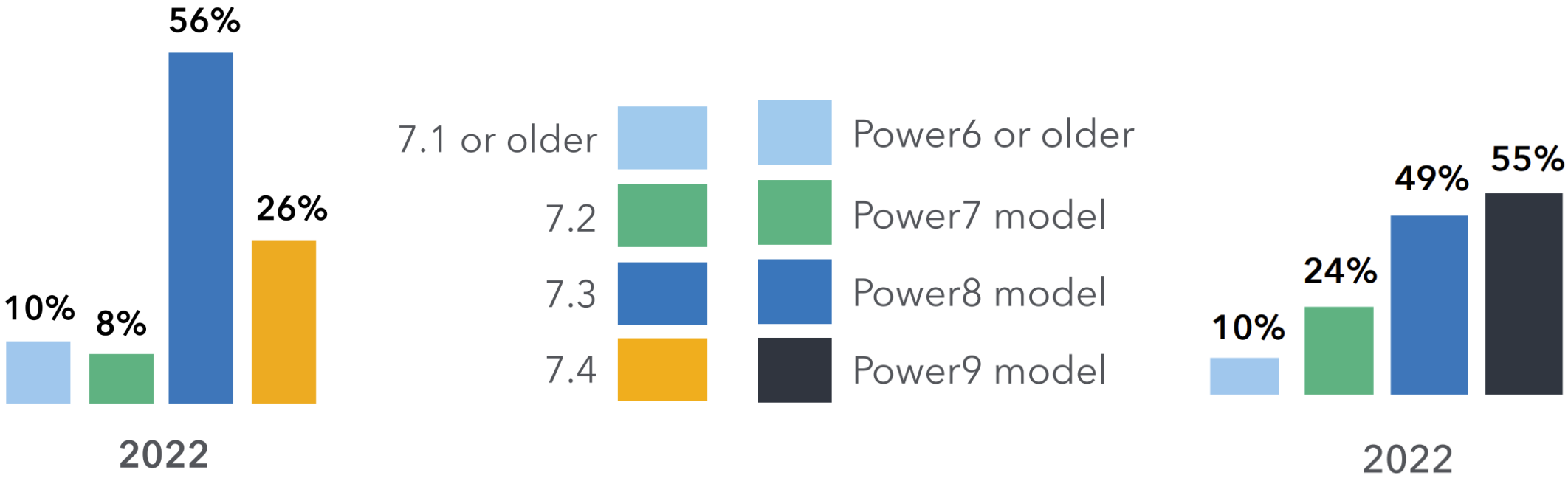
© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Disclaimer

- All examples use *IBM i* release 7.5 & 7.4 TR6
  - Released May 2022
- What is shown may not work for some older releases or TRs
- All examples are deliberately simple to illustrate the point
  - K.I.S.S (Keep It Simple Simon)

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

What are people using?



From HelpSystems' 2022 IBM i Marketplace Survey
Survey taken before availability of 7.5 and Power10 models

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

What is the latest release I could use?

Servers	IBM i 6.1_1	IBM i 7.1_1	IBM i 7.2_1	IBM i 7.3	IBM i 7.4
Power10 **E1080**				✓	✓
Power9 **S914, S922 (VIOS only), H922 (VIOS only), S924, H924, E980** **S922 Entry (single native I/O partition only)**		✓	✓	✓	✓
Power8 **S814, S822 (VIOS only), S824, E870, E870C, E880, E880C** **S812 (no IBM i 7.1, native I/O only)**		✓	✓	✓	✓
Power7/7+ **710, 720, 730, 740, 750 (VIOS only), 760 (VIOS only), 770, 780, 795**	✓	✓	✓	✓	

- IBM i 7.5: Power9 & Power10 only

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Keep your IBM i partitions up to date

- Security enhancements
- Latest fixes
- (*Latest cool features & functions to play with*)

- Latest Technology Refreshes for supported releases
  - 7.5 base
  - 7.4 TR6
  - 7.3 TR12
- CUM PTFs should be applied at least once a year

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Monitor for specific error messages (7.4)

```
**free
dcl-f TESTFILE usage(*delete) usropn ;
dcl-s Number packed(3:0) ;

dsply ('Program : P010') ;

monitor ;
  delete TESTFILER ;
on-excp 'RNX1211' ;  // Must be specific error id
  dsply ('1 Captured by ON-EXCP') ;
on-error 1211 ;
  dsply ('1 Captured by ON-ERROR') ;
on-error ;
  dsply ('1 None of the above') ;
endmon ;

monitor ;
  Number = Number / Number ;  // Divide by 0
on-excp 'MCH1211' ;
  dsply ('2 Captured by ON-EXCP') ;
on-error ;
  dsply ('2 None of the above') ;
endmon ;

monitor ;
  Number = 999 + 1 ;  // Receiver value too small
on-excp 'MCH1210' ;
  dsply ('3 Captured by ON-EXCP') ;
on-error ;
  dsply ('3 None of the above') ;
endmon ;

*inlr = *on ;
```

```
DSPLY  Program : P010
DSPLY  1 Captured by ON-EXCP
DSPLY  2 Captured by ON-EXCP
DSPLY  3 Captured by ON-EXCP
```

New op code ON-EXCP
List error message ids to monitor for

ON-EXCP must be before ON-ERROR

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Send message to job log (7.4)

```
**free
dcl-s Text varchar(50) ;

dsply ('Program : P011') ;

// This sends an *INFO message to the job log
snd-msg 'This is the start of P011' ;

Text = 'Second message from P011' ;
snd-msg Text ;

// Send message using a predefined CPF message
snd-msg %msg('CPF9898' : 'QCPFMSG' : 'This is my replacement text') ;

*inlr = *on ;
```

```
<< From job's job log >>
DSPLY  Program : P011
This is the start of P011
Second message from P011
This is my replacement text.
```

It is possible to send *ESCAPE messages too.
See RPGPGM.COM for examples

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

%UPPER & %LOWER change case (7.3)

```
**free
dcl-s String varchar(30) ;

dsply ('Program : P001') ;

String = 'simon was here' ;
String = %upper(String) ;
dsply ('1 : ' + String) ;

String = %lower(String) ;
dsply ('2 : ' + String) ;

String = %upper(String:1:5) ;
dsply ('3 : ' + String) ;

String = %lower(String:2) ;
dsply ('4 : ' + String) ;

String = 'áâãåæçèéêëìíîïñòóôõøùúûý' ;
String = %upper(String) ;
dsply ('5 : ' + String) ;

*inlr = *on ;
```

```
DSPLY  Program : P001
DSPLY  1 : SIMON WAS HERE
DSPLY  2 : simon was here
DSPLY  3 : SIMON was here
DSPLY  4 : Simon was here
DSPLY  5 : ÁÂÃÅÆÇÈÉÊËÌÍÎÏÑÒÓÔÕØÙÚÛÝ
```

Upper case all characters from the first position to the fifth

Lower case from the second position onwards

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Variable length arrays (7.4)

```
**free
dcl-s FixedArray char(1) dim(100) ;
dcl-s VariableArray char(1) dim(*var:100) ;
dcl-s AutoArray char(1) dim(*auto:100) ;

dcl-s Number uns(5) ;
dsply ('Program : P002') ;

// Fixed number of elements
Number = %elem(FixedArray) ;
dsply ('1 : ' + %char(Number)) ;

// Variable number of elements
Number = %elem(VariableArray) ;
dsply ('2 : ' + %char(Number)) ;

// What happens if I use the 1st element
VariableArray(1) = 'X' ;
dsply ('3 : ' + %char(Number)) ;

*inlr = *on ;
```

```
DSPLY  Program : P002
DSPLY  1 : 100
DSPLY  2 : 0
An array index is out of range (C G D F).
```

Moving value into first element when array is zero length

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Variable length arrays (7.4)

```
**free
dcl-s FixedArray char(1) dim(100) ;
dcl-s VariableArray char(1) dim(*var:100) ;
dcl-s AutoArray char(1) dim(*auto:100) ;

dcl-s Number uns(5) ;
dsply ('Program : P002A') ;

// Set this array to have 5 elements
%elem(VariableArray) = 5 ;
Number = %elem(VariableArray) ;
dsply ('4 : ' + %char(Number)) ;

// Move values into those 5 elements
VariableArray(1) = '1' ;
VariableArray(2) = '2' ;
VariableArray(3) = '3' ;
VariableArray(4) = '4' ;
VariableArray(5) = '5' ;

// Now set VariableArray back to 0
%elem(VariableArray) = 0 ;
Number = %elem(VariableArray) ;
dsply ('5 : ' + %char(Number)) ;

// Keep the previous values
%elem(VariableArray:*keep) = 10 ;
dsply ('6 : ' + VariableArray(5)) ;

// Delete previous values
%elem(VariableArray) = 0 ;
%elem(VariableArray) = 10 ;
dsply ('7 : ' + VariableArray(5)) ;

*inlr = *on ;
```

```
DSPLY  Program : P002A
DSPLY  4 : 5
DSPLY  5 : 0
DSPLY  6 : 5
DSPLY  7 :
```

Must set number of elements

*KEEP retains previous values in array elements

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Auto extending length arrays (7.4)

```
**free
dcl-s FixedArray char(1) dim(100) ;
dcl-s VariableArray char(1) dim(*var:100) ;
dcl-s AutoArray char(1) dim(*auto:100) ;

dcl-s Number uns(5) ;
dsply ('Program : P002B') ;

// Show AutoArray has 0 elements
Number = %elem(AutoArray) ;
dsply ('8 : ' + %char(Number)) ;

// Move something into 1st element
AutoArray(1) = 'X' ;
Number = %elem(AutoArray) ;
dsply ('9 : ' + %char(Number)) ;

*inlr = *on ;
```

```
DSPLY  Program : P002B
DSPLY  8 : 0
DSPLY  9 : 1
```

*AUTO will auto-extend when move values to unused elements

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Auto extending length arrays (7.4)

```
**free
dcl-ds DS_Array qualified dim(*auto:100) ;
  FirstColumn char(1) ;
  SecondColumn packed(4:0) ;
end-ds ;

dcl-s Number uns(5) ;
dsply ('Program : P002C') ;

exec sql DECLARE C0 CURSOR FOR SELECT * FROM RPGPGM1.TESTTABLE FOR READ ONLY ;
exec sql OPEN C0 ;

exec sql FETCH C0 FOR 100 ROWS INTO :DS_Array ;
exec sql GET DIAGNOSTICS :Number = ROW_COUNT ;   // No. rows retrieved

exec sql CLOSE C0 ;

dsply ('10 Rows fetched : ' + %char(Number)) ;

Number = %elem(DS_Array) ;  // No. of elements in the array
dsply ('11 Array elements : ' + %char(Number)) ;

*inlr = *on ;
```

```
DSPLY  Program : P002C
DSPLY  10 Rows fetched : 4
DSPLY  11 Array elements : 4
```

Array extends to accommodate returned rows of data

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Load multiple elements in array (7.3)

```
**free
dcl-s AutoArray char(1) dim(*auto:100) ;

dcl-s Number uns(5) ;
dsply ('Program : P003') ;

// Load first 10 elements of array
AutoArray = %list('A' : 'B' : 'C' : 'D' : 'E' : 'F' : 'G' : 'H' : 'I' : 'J') ;

// How many elements in array
Number = %elem(AutoArray) ;
dsply ('1 No. elements : ' + %char(Number)) ;

// What is in the 10th element
dsply ('2 10th element : ' + AutoArray(10)) ;

*inlr = *on ;
```

```
DSPLY  Program : P003
DSPLY  1 No. elements : 10
DSPLY  2 10th element : J
```

%LIST loads multiple array elements

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Read all the elements in the array (7.3)

```
**free
dcl-s AutoArray char(1) dim(*auto:100) ;
dcl-s Element char(1) ;
dsply ('Program : P004') ;

AutoArray = %list('A' : 'B' : 'C' : 'D' : 'E' : 'F' : 'G' : 'H' : 'I' : 'J') ;

// Loop through & display all elements
for-each Element in AutoArray ;
  dsply Element ;
endfor ;

*inlr = *on ;
```

```
DSPLY  Program : P004
DSPLY  A
DSPLY  B
DSPLY  C
DSPLY  D
DSPLY  E
DSPLY  F
DSPLY  G
DSPLY  H
DSPLY  I
DSPLY  J
```

FOR_EACH loop & read ALL the elements in the array

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Minimum & maximum value in array (7.3)

```
**free
dcl-s AutoArray char(2) dim(*auto:50) ;
dcl-s Number uns(5) ;
dsply ('Program : P005') ;

AutoArray = %list('TX' : 'CA' : 'AZ' : 'UT' : 'NM') ;

// Minimum value in array
dsply ('Min element : ' + %char(%minarr(AutoArray))) ;
dsply ('Min value : ' + AutoArray(%minarr(AutoArray))) ;

// Maximum value in array
dsply ('Max element : ' + %char(%maxarr(AutoArray))) ;
dsply ('Max value : ' + AutoArray(%maxarr(AutoArray))) ;

*inlr = *on ;
```

```
DSPLY  Program : P005
DSPLY  Min element : 3
DSPLY  Min value : AZ
DSPLY  Max element : 4
DSPLY  Max value : UT
```

%MINARR return array element
No. with lowest value

%MAXARR return array element
No. with highest value

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Improved sorting for data structure arrays (7.3)

```
**free
dcl-ds DS_Array qualified dim(*auto:100) ;
  FirstColumn char(1) ;
  SecondColumn packed(4:0) ;
end-ds ;

dsply ('Program : P006') ;

// Load DS_Array
exec sql DECLARE C0 CURSOR FOR SELECT * FROM RPGPGM1.TESTTABLE FOR READ ONLY ;
exec sql OPEN C0 ;
exec sql FETCH C0 FOR 100 ROWS INTO :DS_Array ;
exec sql CLOSE C0 ;

<< Breakpoint 1
// Sort ascending on the 2nd subfield
sorta DS_Array %fields(SecondColumn : FirstColumn) ;

<< Breakpoint 2
// Sort descending on the 1st subfield
sorta(d) DS_Array %fields(FirstColumn : SecondColumn) ;

<< Breakpoint 3
*inlr = *on ;
```

SORTA ascending by SecondColumn, FirstColumn

SORTA(D) descending by FirstColumn, SecondColumn

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Improved sorting for data structure arrays (7.3)

```
<< Breakpoint 1
EVAL ds_array
DS_ARRAY.FIRSTCOLUMN(1) = 'F'
DS_ARRAY.SECONDCOLUMN(1) = 0001.
DS_ARRAY.FIRSTCOLUMN(2) = 'T'
DS_ARRAY.SECONDCOLUMN(2) = 1000.
DS_ARRAY.FIRSTCOLUMN(3) = 'F'
DS_ARRAY.SECONDCOLUMN(3) = 0002.
DS_ARRAY.FIRSTCOLUMN(4) = 'T'
DS_ARRAY.SECONDCOLUMN(4) = 0501.

<< Breakpoint 2
EVAL ds_array
DS_ARRAY.FIRSTCOLUMN(1) = 'F'
DS_ARRAY.SECONDCOLUMN(1) = 0001.
DS_ARRAY.FIRSTCOLUMN(2) = 'F'
DS_ARRAY.SECONDCOLUMN(2) = 0002.
DS_ARRAY.FIRSTCOLUMN(3) = 'T'
DS_ARRAY.SECONDCOLUMN(3) = 0501.
DS_ARRAY.FIRSTCOLUMN(4) = 'T'
DS_ARRAY.SECONDCOLUMN(4) = 1000.

<< Breakpoint 3
EVAL ds_array
DS_ARRAY.FIRSTCOLUMN(1) = 'T'
DS_ARRAY.SECONDCOLUMN(1) = 1000.
DS_ARRAY.FIRSTCOLUMN(2) = 'T'
DS_ARRAY.SECONDCOLUMN(2) = 0501.
DS_ARRAY.FIRSTCOLUMN(3) = 'F'
DS_ARRAY.SECONDCOLUMN(3) = 0002.
DS_ARRAY.FIRSTCOLUMN(4) = 'F'
DS_ARRAY.SECONDCOLUMN(4) = 0001.
```

Before

After SORTA

After SORTA(D)

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Improved IFs, Range & List (7.3)

```
**free
dcl-s Color char(10) inz('ORANGE') ;
dsply ('Program : P007') ;

// Use list to compare single value to a list of values
if ('RED' in %list('GREEN' : 'BLUE' : 'YELLOW' : 'PINK' : 'CYAN' : 'RED')) ;
  dsply ('1 Found') ;
else ;
  dsply ('1 Not found') ;
endif ;

if (Color in %list('GREEN' : 'BLUE' : 'YELLOW' : 'PINK' : 'CYAN' : 'RED')) ;
  dsply ('2 Found') ;
else ;
  dsply ('2 Not found') ;
endif ;

// Range test
if %date() in %range(d'2022-08-01' : d'2022-10-03') ;
  dsply ('3 Yes')
else ;
  dsply ('3 No') ;
endif ;

*inlr = *on ;
```

```
DSPLY  Program : P007
DSPLY  1 Found
DSPLY  2 Not found
DSPLY  3 Yes
```

%LIST used to compare to a list of values

%RANGE used to compare to a range, e.g. dates

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Defining a DS based on an embedded DS (7.3)

```
**free
dcl-ds Family1 qualified ;
  Number uns(10) ;
  HeadName char(40) ;
  DateOfBirth date ;
  dcl-ds Member dim(10) ; // Embedded DS automatically qualified
    Name char(40) ;
    DateOfBirth date ;
  end-ds ;
end-ds ;

dcl-ds Person likeds(Family1.Member) ; // Based on embedded DS above

dcl-ds Family2 qualified ;
  Number uns(10) ;
  HeadName char(40) ;
  DateOfBirth date ;
  dcl-ds Member likeds(Person) dim(10) ; // Can define using LIKEDS
end-ds ;

dsply ('Program : P008') ;

Family1.Number = 34 ;
Family1.HeadName = 'WESLEY HIGGINS' ;
Family1.DateOfBirth = d'1949-09-24' ;
Family1.Member(1).Name = 'JENNY HIGGINS' ;
Family1.Member(1).DateOfBirth = d'1952-12-16' ;

Family2.Number = 49 ;
Family2.HeadName = 'SIMEON DI CARLO' ;
Family2.DateOfBirth = d'1972-03-21' ;
Family2.Member(1).Name = 'MARIA DI CARLO' ;
Family2.Member(1).DateOfBirth = d'1971-09-14' ;

*inlr = *on ;
```

Embedded Data Structure

Used in LIEKDS to define a new Data Structure

Use new Data Structure to define an embedded Data Strucutre

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Defining a DS based on an embedded DS (7.3)

```
EVAL family1
FAMILY1.NUMBER = 34
FAMILY1.HEADNAME = 'WESLEY HIGGINS                          '
FAMILY1.DATEOFBIRTH = '1949-09-24'
FAMILY1.MEMBER.NAME(1) = 'JENNY HIGGINS                                '
FAMILY1.MEMBER.DATEOFBIRTH(1) = '1952-12-16'
FAMILY1.MEMBER.NAME(2) = '                                             '
FAMILY1.MEMBER.DATEOFBIRTH(2) = '          '
FAMILY1.MEMBER.NAME(3) = '                                             '

EVAL family2
FAMILY2.NUMBER = 49
FAMILY2.HEADNAME = 'SIMEON DI CARLO                         '
FAMILY2.DATEOFBIRTH = '1972-03-21'
FAMILY2.MEMBER.NAME(1) = 'MARIA DI CARLO                               '
FAMILY2.MEMBER.DATEOFBIRTH(1) = '1971-09-14'
FAMILY2.MEMBER.NAME(2) = '                                             '
FAMILY2.MEMBER.DATEOFBIRTH(2) = '          '
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Overlay subfields in a Data Structure (7.3)

```
**free
dcl-ds *n ;
  Name1 char(10) ;
  Name2 like(Name1) ;
  Name3 like(Name1) ;
  Name4 like(Name1) ;
  Name5 like(Name1) ;
  NameArray char(10) dim(5) samepos(Name1) ;
end-ds ;

dsply ('Program : P009') ;

NameArray = %list('DAVID' : 'MARY' : 'ANGELO' : 'DANIEL' : 'MARK') ;

dsply ('Name1 = ' + Name1) ;

*inlr = *on ;
```

```
DSPLY  Program : P009
DSPLY  Name1 = DAVID
```

SAMEPOS overlays the array starting at the subfield Name1

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Ctl-Opt to stop blank to number error (7.3)

```
**free
ctl-opt expropts(*alwblanknum) ;

dcl-s Character char(10) ;
dcl-s Number packed(9:2) ;

dsply ('Program : P012') ;

Character = ' ' ;

// Convert to decimal
Number = %dec(Character : 9 : 2) ;

dsply ('Number = ' + %char(Number)) ;

*inlr = *on ;
```

*ALWBLANKNUM Allows blank to be converted to number.
Does not mean any other character

```
<< Error message without *ALWBLANKNUM >>
A character representation of a numeric value is in error (C G D F).

<< With *ALWBLANKNUM >>
DSPLY  Program : P012
DSPLY  Number = .00
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Ctl-Opt to handle decimal separators (7.3)

```
**free
ctl-opt expropts(*usedecedit) ;

dcl-s Character char(10) ;
dcl-s Number packed(9:2) ;

dsply ('Program : P013') ;

Character = '1,234.56' ;

// Convert to decimal
Number = %dec(Character : 9 : 2) ;

dsply ('Number = ' + %char(Number)) ;

*inlr = *on ;
```

*USEDECEDIT Removes decimal separator characters when convert to decimal

```
<< Error message without *USEDECEDIT >>
A character representation of a numeric value is in error (C G D F).

<< With *USEDECEDIT >>
DSPLY  Program : P013
DSPLY  Number = 1234.56
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Ctl-Opt – Can use both (7.3)

```
**free
ctl-opt expropts(*alwblanknum : *usedecedit) ;
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Debug constants (7.3)

```
**free
ctl-opt debug(*constants) ;

dcl-c Constant1 const('This is a constant') ;
dcl-c Constant2 'Another constant' ;

<< Breakpoint
*inlr = *on ;
```

DEBUG(*CONSTANTS) allows you do debug constants like any variable

```
ev constant1
CONSTANT1 = 'This is a constant'

ev constant2
CONSTANT2 = 'Another constant'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Special options for RPG debugging

- _QRNU_RETVAL : Change value returned from procedure
- _QRNU_NULL_name : See or change null indicator for variable
- _QRNU_TABI_name : See or change index of a RPG table
- _QRUN_DSI_name : See of change index of a MODS (multiple occurrence data structure)

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_RETVAL : Procedure return value (7.3)

```
**free
ctl-opt debug(*retval) dftactgrp(*no) ;

dcl-pr TestProcedure char(1) ;
end-pr ;

dcl-s MainCharacter char(1) ;

MainCharacter = TestProcedure() ;

dsply ('MainCharacter = ' + MainCharacter) ;

*inlr = *on ;
//----------------------------------------
dcl-proc TestProcedure ;
  dcl-pi *n char(1) ;
  end-pi ;

  return *on ;
// Breakpoint
end-proc ;
```

DEBUG(*RETVAL) allows to use _QRNU_RETVAL

Breakpoint must be within procedure

```
ev _qrnu_retval
_QRNU_RETVAL = '1'
ev _qrnu_retval = '0'
_QRNU_RETVAL = '0' = '0'

DSPLY  MainCharacter = 0
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_RETVAL : With data structure (7.3)

```
**free
ctl-opt debug(*retval) dftactgrp(*no) ;

dcl-ds MainDS qualified ;
  First char(1) ;
  Second char(2) ;
end-ds ;

dcl-pr TestProcedure likeds(MainDS) ;
end-pr ;

MainDS = TestProcedure() ;

dsply ('MainDS = ' + MainDS.First + ' ' + MainDS.Second) ;

*inlr = *on ;
//-------------------------------------------------------
dcl-proc TestProcedure ;
  dcl-pi *n likeds(MainDS) ;
  end-pi ;

  dcl-ds ProcedureDS likeds(MainDS) ;

  ProcedureDS.First = '1' ;
  ProcedureDS.Second = '2' ;

  return ProcedureDS ;

end-proc ;
```

```
ev _qrnu_retval.first
_QRNU_RETVAL.FIRST = '1'
ev _qrnu_retval.second
_QRNU_RETVAL.SECOND = '2 '
ev _qrnu_retval.second = '3'
_QRNU_RETVAL.SECOND = '3' = '3 '

DSPLY  MainDS = 1 3
```

Refer to the data structure subfields

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_RETVAL : With array (7.3)

```
**free
ctl-opt debug(*retval) dftactgrp(*no) ;

dcl-s Array char(1) dim(5) ;

dcl-pr TestProcedure char(1) dim(5) ;
end-pr ;

Array = TestProcedure() ;

dsply ('Array(1) = ' + Array(1)) ;

*inlr = *on ;
//-------------------------------------
dcl-proc TestProcedure ;
  dcl-pi *n char(1) dim(5) ;
  end-pi ;

  dcl-s ProcArray char(1) dim(5) ;

  ProcArray = %list('A' : 'B' : 'C' : 'D' : 'E') ;

  return ProcArray ;

end-proc ;
```

```
ev _qrnu_retval(1)
_QRNU_RETVAL(1) = 'A'
EVAL _qrnu_retval(1) = '1'
_QRNU_RETVAL(1) = '1' = '1'

DSPLY  Array(1) = 1
```

Refer to the array element

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_NULL_name : Null indicator (7.3)

```
**free
ctl-opt alwnull(*usrctl) ;

dcl-s Var1 char(1) nullind ;

dsply ('Program : P016') ;

%nullind(Var1) = *on ;

// Breakpoint 1 : ev _qrnu_null_var1
%nullind(Var1) = *off ;

// Breakpoint 2 : ev _qrnu_null_var1
*inlr = *on ;
```

```
// Breakpoint 1
ev _qrnu_null_var1
_QRNU_NULL_VAR1 = '1'

// Breakpoint 2
ev _qrnu_null_var1
_QRNU_NULL_VAR1 = '0'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_TABI_name : RPG table (7.3)

```
**free
dcl-s Table1 char(1) dim(5) ctdata ;

*inlr = *on ;

**
1
2
3
4
5
```

```
ev table1
TABLE1 = '1'

ev table1(1..5)
TABLE1(1) = '1'
TABLE1(2) = '2'
TABLE1(3) = '3'
TABLE1(4) = '4'
TABLE1(5) = '5'

ev table1 = %index(3)
TABLE1 = %INDEX(3) = 3
ev table1
TABLE1 = '3'

ev _qrnu_tabi_table1
_QRNU_TABI_TABLE1 = 3

ev _qrnu_tabi_table1 = 5
_QRNU_TABI_TABLE1 = 5 = 5
ev table1
TABLE1 = '5'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

_QRNU_DSI_name : MODS index (7.3)

- MODS = Multiple Occurrence Data Structure
  - Use data structure arrays instead

```
     D MODS             DS                  occurs(5)

       %occur(MODS) = 3 ;

       *inlr = *on ;
```

```
ev _qrnu_dsi_mods
_QRNU_DSI_MODS = 3

ev _qrnu_dsi_mods = 2
_QRNU_DSI_MODS = 2 = 2
ev _qrnu_dsi_mods
_QRNU_DSI_MODS = 2
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Splitting a string into an array (7.3)

```
**free
dcl-s Array varchar(10) dim(*auto:10) ;
dcl-s String varchar(50) inz('Sun Mon Tue Wed Thu Fri Sat') ;
dcl-s Number uns(5) ;

dsply ('Program : P019') ;

%elem(Array) = 0 ;
Array = %SPLIT('Sun Mon Tue Wed Thu Fri Sat') ;                    //(1)
Number = %elem(Array) ;

Array = %split('Sun,Mon,Tue,Wed,Thu,Fri,Sat' : ',') ;              //(2)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ',') ;        //(3)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ', ') ;       //(4)
Array = %split('Sun,  Mon,  Tue,  Wed,  Thu,  Fri,  Sat' : ', ') ; //(5)

Array = %split(String) ;                                           //(6)
```

Auto extending array

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Splitting a string into an array

```
**free
dcl-s Array varchar(10) dim(*auto:10) ;
dcl-s String varchar(50) inz('Sun Mon Tue Wed Thu Fri Sat') ;
dcl-s Number uns(5) ;

dsply ('Program : P019') ;

%elem(Array) = 0 ;
Array = %SPLIT('Sun Mon Tue Wed Thu Fri Sat') ;                    //(1)
Number = %elem(Array) ;

Array = %split('Sun,Mon,Tue,Wed,Thu,Fri,Sat' : ',') ;              //(2)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ',') ;        //(3)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ', ') ;       //(4)
Array = %split('Sun,  Mon,  Tue,  Wed,  Thu,  Fri,  Sat' : ', ') ; //(5)

Array = %split(String) ;                                           //(6)
```

```
(1) Single space = default
EVAL number
NUMBER = 7
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = 'Mon       '
ARRAY(3) = 'Tue       '
ARRAY(4) = 'Wed       '
ARRAY(5) = 'Thu       '
ARRAY(6) = 'Fri       '
ARRAY(7) = 'Sat       '

(2) Comma
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = 'Mon       '
ARRAY(3) = 'Tue       '
ARRAY(4) = 'Wed       '
ARRAY(5) = 'Thu       '
ARRAY(6) = 'Fri       '
ARRAY(7) = 'Sat       '

(3) Comma + space & comma in delimiter only
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = ' Mon      '
ARRAY(3) = ' Tue      '
ARRAY(4) = ' Wed      '
ARRAY(5) = ' Thu      '
ARRAY(6) = ' Fri      '
ARRAY(7) = ' Sat      '
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Splitting a string into an array

```
**free
dcl-s Array varchar(10) dim(*auto:10) ;
dcl-s String varchar(50) inz('Sun Mon Tue Wed Thu Fri Sat') ;
dcl-s Number uns(5) ;

dsply ('Program : P019') ;

%elem(Array) = 0 ;
Array = %SPLIT('Sun Mon Tue Wed Thu Fri Sat') ;                    //(1)
Number = %elem(Array) ;

Array = %split('Sun,Mon,Tue,Wed,Thu,Fri,Sat' : ',') ;              //(2)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ',') ;        //(3)
Array = %split('Sun, Mon, Tue, Wed, Thu, Fri, Sat' : ', ') ;       //(4)
Array = %split('Sun,  Mon,  Tue,  Wed,  Thu,  Fri,  Sat' : ', ') ; //(5)

Array = %split(String) ;                                           //(6)
```

```
(4) Comma + space & in delimiter definition
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = 'Mon       '
ARRAY(3) = 'Tue       '
ARRAY(4) = 'Wed       '
ARRAY(5) = 'Thu       '
ARRAY(6) = 'Fri       '
ARRAY(7) = 'Sat       '

(5) Comma + 2 spaces
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = 'Mon       '
ARRAY(3) = 'Tue       '
ARRAY(4) = 'Wed       '
ARRAY(5) = 'Thu       '
ARRAY(6) = 'Fri       '
ARRAY(7) = 'Sat       '

(6) Using variable
EVAL array(1..7)
ARRAY(1) = 'Sun       '
ARRAY(2) = 'Mon       '
ARRAY(3) = 'Tue       '
ARRAY(4) = 'Wed       '
ARRAY(5) = 'Thu       '
ARRAY(6) = 'Fri       '
ARRAY(7) = 'Sat       '
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Splitting a string into an array (7.3)

```
%elem(Array) = 0 ;
Array = %SPLIT('Apple  Orange  Strawberry  Cherry  Kiwi fruit') ;   //(7)
Number = %elem(Array) ;

%elem(Array) = 0 ;
String = 'Apple  Orange  Strawberry  Cherry  Kiwi fruit' ;
String = %scanrpl('  ' : '*' : String) ;
Array = %split(String : '*') ;                                     //(8)
Number = %elem(Array) ;

*inlr = *on ;
```

```
(7) Fruits seperated by 2 spaces
EVAL number
NUMBER = 6
EVAL array(1..6)
ARRAY(1) = 'Apple     '
ARRAY(2) = 'Orange    '
ARRAY(3) = 'Strawberry'
ARRAY(4) = 'Cherry    '
ARRAY(5) = 'Kiwi      '
ARRAY(6) = 'fruit     '

(8) Replace 2 spaces with *
EVAL string
STRING = 'Apple*Orange*Strawberry*Cherry*Kiwi fruit          '
EVAL number
NUMBER = 5
EVAL array(1..5)
ARRAY(1) = 'Apple     '
ARRAY(2) = 'Orange    '
ARRAY(3) = 'Strawberry'
ARRAY(4) = 'Cherry    '
ARRAY(5) = 'Kiwi fruit'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Timestamp enhancements (7.3)

```
**free
dcl-s TS timestamp ;
dcl-s TS0 timestamp(0) ;
dcl-s TS1 timestamp(1) ;
dcl-s TS6 timestamp(6) ;
dcl-s TS12 timestamp(12) ;

TS = %timestamp() ;
TS0 = %timestamp() ;
TS1 = %timestamp() ;
TS6 = %timestamp() ;

TS = %timestamp(*sys) ;
TS = %timestamp(*sys:0) ;
TS = %timestamp(*sys:1) ;

TS12 = %timestamp() ;
TS12 = %timestamp(*unique) ;

*inlr = *on ;
```

Timestamp variables of different lengths

Moving timestamps of different lengths to variable

New longer timestamp variable

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Timestamp enhancements (7.3)

```
**free
dcl-s TS timestamp ;
dcl-s TS0 timestamp(0) ;
dcl-s TS1 timestamp(1) ;
dcl-s TS6 timestamp(6) ;
dcl-s TS12 timestamp(12) ;

TS = %timestamp() ;
TS0 = %timestamp() ;
TS1 = %timestamp() ;
TS6 = %timestamp() ;

TS = %timestamp(*sys) ;
TS = %timestamp(*sys:0) ;
TS = %timestamp(*sys:1) ;

TS12 = %timestamp() ;
TS12 = %timestamp(*unique) ;

*inlr = *on ;
```

Debug these lines of code
(variables defined with different number of microseconds)

```
TS = '2022-08-11-13.19.38.752586'
TS0 = '2022-08-11-13.19.38'
TS1 = '2022-08-11-13.19.38.7'
TS6 = '2022-08-11-13.19.38.752629'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Timestamp enhancements (7.3)

```
**free
dcl-s TS timestamp ;
dcl-s TS0 timestamp(0) ;
dcl-s TS1 timestamp(1) ;
dcl-s TS6 timestamp(6) ;
dcl-s TS12 timestamp(12) ;

TS = %timestamp() ;
TS0 = %timestamp() ;
TS1 = %timestamp() ;
TS6 = %timestamp() ;

TS = %timestamp(*sys) ;
TS = %timestamp(*sys:0) ;
TS = %timestamp(*sys:1) ;

TS12 = %timestamp() ;
TS12 = %timestamp(*unique) ;

*inlr = *on ;
```

Debug these lines of code
(different number of microseconds being moved to “standard” timestamp)

```
TS = '2022-08-11-13.20.55.815261'
TS = '2022-08-11-13.21.12.000000
TS = '2022-08-11-13.21.33.500000'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Timestamp enhancements (7.3)

```
**free
dcl-s TS timestamp ;
dcl-s TS0 timestamp(0) ;
dcl-s TS1 timestamp(1) ;
dcl-s TS6 timestamp(6) ;
dcl-s TS12 timestamp(12) ;

TS = %timestamp() ;
TS0 = %timestamp() ;
TS1 = %timestamp() ;
TS6 = %timestamp() ;

TS = %timestamp(*sys) ;
TS = %timestamp(*sys:0) ;
TS = %timestamp(*sys:1) ;

TS12 = %timestamp() ;
TS12 = %timestamp(*unique) ;

*inlr = *on ;
```

*UNIQUE Longer timestamp
12 “microseconds”
1 - 6 = Microseconds
7 – 12 = Unique number

```
TS12 = '2022-08-11-13.21.54.190708000000'
TS12 = '2022-08-11-13.22.09.711624000244'
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Infinite loop (7.4)

```
**free
dcl-s Number uns(3) ;

dsply ('Program : P020') ;

dow *on ;


  Number += 1 ;
  if (Number = 100) ;
    leave ;
  endif ;
enddo ;

dsply ('Number = ' + %char(Number)) ;

*inlr = *on ;
```

```
DSPLY  Program : P020
DSPLY  Number = 100
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

On exit & close *all (7.3)

```
**free
ctl-opt main(Main) ;

dcl-f TESTFILE usropn ;
dcl-f TESTFILE1 extfile('TESTFILE') extdesc('TESTFILE')
                  rename(TESTFILER:TESTFILE1R) usropn ;

dcl-s Number packed(3:0) ;

dcl-proc Main ;
  dsply ('Program : P022') ;

  open TESTFILE ;
  open TESTFILE1 ;

  Number = Number / Number ;  // Will error

  close TESTFILE ;
  close TESTFILE1 ;
  dsply ('P022: End of program') ;
end-proc ;
```

```
DSPLY  Program : P022
Attempt made to divide by zero for fixed point operation.
Application error.  MCH1211 unmonitored by P022 at statement 0000001600,
  instruction X'0000'.
```

```
                                        Display Open Files

Job . . :   QPAD144047      User . . :   RPGPGM
Number of open data paths . . . . . . . . . :

                           Member/
File        Library        Device       Scope          Ac
QDUI132     QSYS           QPAD144047   *ACTGRPDFN     *D
QDUODSPF    QPDA           QPAD144047   *ACTGRPDFN     *D
QDDSPEXT    QSYS           QPAD144047   *ACTGRPDFN     *D
TESTFILE    RPGPGM1        TESTFILE     *ACTGRPDFN     *D
TESTFILE    RPGPGM1        TESTFILE     *ACTGRPDFN     *D
QDDSPOF     QSYS           QPAD144047   *ACTGRPDFN     *D
```

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

On exit & close *all (7.3)

```
**free
ctl-opt main(Main) dftactgrp(*no) ;

dcl-f TESTFILE usropn ;
dcl-f TESTFILE1 extfile('TESTFILE') extdesc('TESTFILE')
                  rename(TESTFILER:TESTFILE1R) usropn ;

dcl-s Number packed(3:0) ;

dcl-proc Main ;
  dsply ('Program : P022A') ;

  open TESTFILE ;
  open TESTFILE1 ;

  Number = Number / Number ;  // Will error

on-exit ;
  close *all ;
  dsply ('P022A: End of program') ;
end-proc ;
```

```
DSPLY  Program : P022A
DSPLY  P022A: End of program
```

*ON-EXIT executed when procedure ends

CLOSE *ALL closes all open files

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Where to find more information? (RPG only)

- 7.5
  - https://www.ibm.com/docs/en/i/7.5?topic=irpg-whats-new
- 7.4
  - https://www.ibm.com/docs/en/i/7.4?topic=ssw_ibm_i_74/rzasc/rpgrnewrel.htm
- 7.3
  - https://www.ibm.com/docs/en/i/7.3?topic=ssw_ibm_i_73/rzasc/rpgrnewrel.htm
- 7.2
  - https://www.ibm.com/docs/en/i/7.2?topic=irpg-whats-new

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

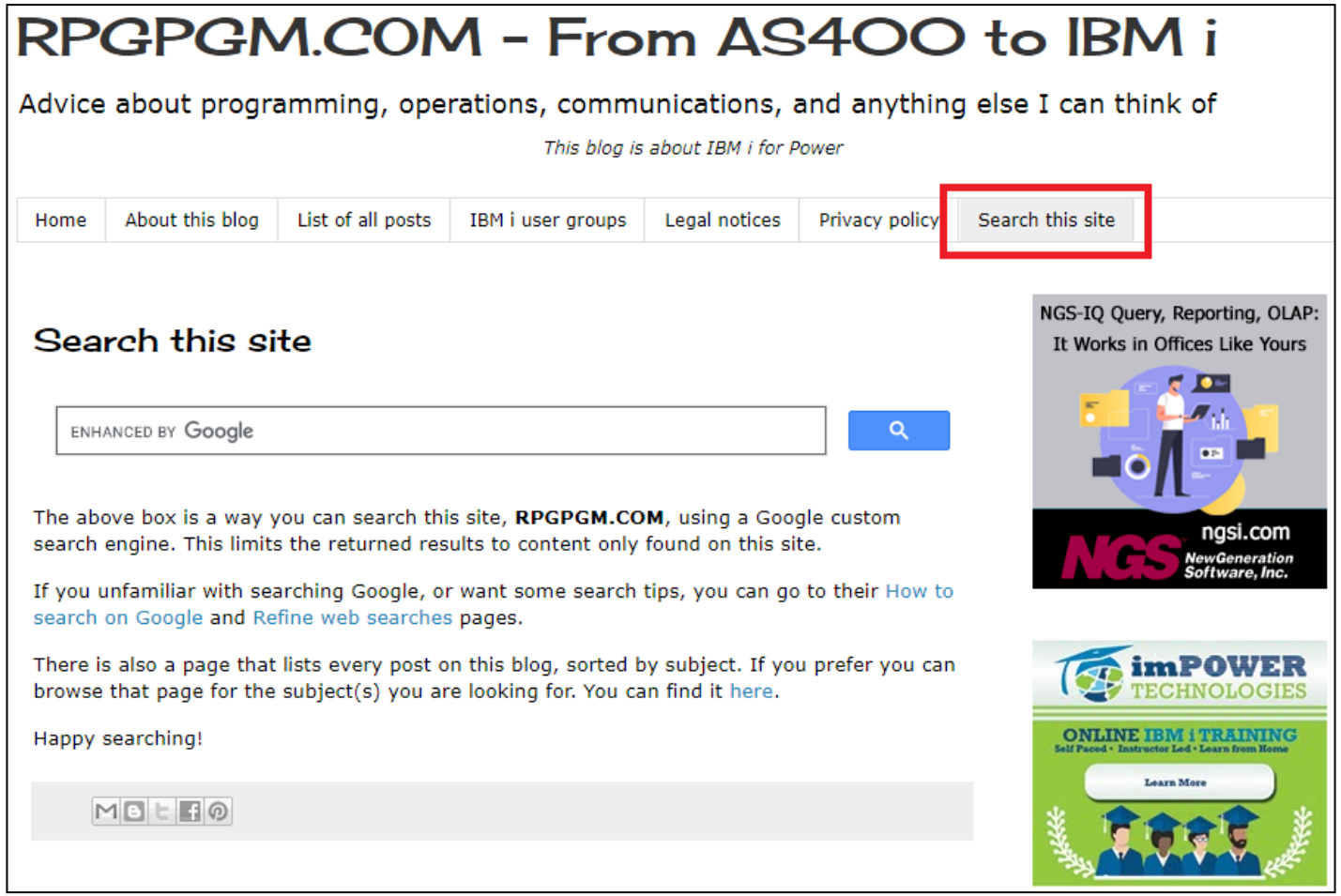
You need time to play

- Take time to play with this stuff

- Free *IBM i* partition
  - PUB400.COM
    - https://pub400.com
    - Version = **7.5**
- Free 5250 client
  - IBM ACS (Access Client Solutions)
    - https://ibm.biz/IBMi_ACS
    - Need free IBM id
    - Lot more than just 5250 emulation
    - Latest = **1.1.9.0**
    - Requires Windows 7 or higher, Java 8 or higher

© Copyright 2022 Simon Hutchinson, RPGPGM.COM

Looking for something = Search



© Copyright 2022 Simon Hutchinson, RPGPGM.COM