WSL Guide

1. Migrated Code

The final migrated code for each module is supplied in WSL.

The WSL code is represented internally as a parse tree structure. The textual layout and formatting of the WSL is therefore completely configurable just by modifying the code generator and associated translation tables. For example: keywords can be changed, the number of statements on a line can be changed, each closing keyword (FI, OD etc.) can be put on a separate line and the formatting of procedure calls and function calls modified. Likewise, the marked-up WSL is configured by a style sheet so the colours of each WSL element type can be changed.

2. WSL Syntax Description

A WSL program consists of a list of statements separated by semicolons: the semicolon is a statement separator.

2.1. Compound Statement

A compound statement is usually bracketed by keywords such as IF ... FI and WHILE ... DO ... OD

2.2. Comments

A comment is represented as a comment statement and is coloured grey.
Note, that a comment that starts with "*" was originally a comment line in the assembler, other comments were attached to an assembler statement.

2.3. Local Variable

A local variable is written: VAR < v1 := e1, v2 := e2, ... >: ... scope of the local variable ... ENDVAR
where e1 is the initial value of variable v1 and VAR ... ENDVAR delimits the scope of the variables.

2.4. IF Statement

An IF statement is written:
IF condition1 THEN statements1
…
ELSIF condition2 THEN statements2
...
ELSE statementsN FI

2.5. WHILE Loop

A WHILE loop is written: WHILE condition DO ...body... OD

2.6. DO Loop

An Floop (unbounded or infinite loop) is written: DO ...body... OD
This loop can only be terminated by a statement of the form EXIT(n) which will terminate the enclosing n nested DO ... OD loops.

2.7. Procedure

A procedure is defined as:
PROC foo() ==
... body of procedure... END
and is called as: foo()

In the migrated code the extracted procedures do not have parameters.

An external procedure call is written:
!P foo(e1, e2, ... VAR v1, v2, ...)
where e1, e2,... are value parameters and v1, v2, ... are variable parameters.
The variable parameters pass a value to the procedure and return a result. The special VAR parameter "os" represents the environment. A procedure with an "os" parameter can affect the environment in some way: for example, printing a message or reading to or writing from a file.

TPF procedures are shown as the name followed by the original parameter list from the assembler as a string . For example: DBCLS "REF=QR011W,ABORT"

The procedure: !P fill(L_G_BD_WORK0_1[35]
  VAR L_G_BD_WORK0_1[36..290]);
will fill the field in the VAR parameter with copies of the byte given in the first parameter.

2.8. Assignments

Assignments are written: var := expression

An expression or lvalue of the form a[foo].bar means that "foo" is a pointer to a structure and bar is a record in the structure.

The notation foo[n1..n2] extracts the sequence of bytes from n1 to n2 inclusive from the variable foo. For example:
a[QU42WORK].Q42REG3[1..4] := r0;
sets the first four bytes of record Q42REG3 in the structure pointed at by QU42WORK to the value of r0 (general register 0).

!XF address_of(foo) returns the address of variable foo.

A hex string is denoted 0xXXXXXX, where XXXXXX are the hex digits. So this assignment:
L_G_BD_WORK4_1[1..3] := 0x156C4E
copies the three hex bytes X'156C4E' into the first three bytes of the variable L_G_BD_WORK4_1.

2.9. Other

a[expr, len] refers to the "len" byte string at memory location "expr"
so for example a[r1 + 12, 5] is the five byte string starting at the address r1 + 12 (offset 12 bytes from the address in r1).

a[CE1CR5, 2] = "P4" tests if the two byte string at address CE1CR5 contains the value "P4".

a[expr].field.subfield refers to a subfield in a record where the address of the record is in "expr"

A sub-segment of a field or variable is defined as:
field[start..end] or a[expr].field[start..end] where "start" and "end" are byte numbers, numbered from 1. So:

EBW003[1..3] refers to the first three bytes of variable EBW003
CE1FA5[1..2] := 0x0000 sets the first two bytes of CE1FA5 to binary zeros.

The functions !XF bit_and(e1, e2), !XF bit_or(e1, e2) etc. are bit manipulation functions that carry out bitwise AND/OR etc. operations on the given values. So !XF bit_and(e1, e2) is equivalent to the C expression (e1 & e2), !XF bit_or(e1, e2) is equivalent to (e1 | e2) and so on.

Procedures such as !P bit_or_1(expn, N VAR variable) are equivalent to:
variable[1..N] := !XF bit_or(expn, variable[1..N]) where "N" gives the length of the operation in bytes.

A string in {curly brackets} is an annotation which describes the preceding element in the program.

A test of a data item with a single bit equate is presented as an IF with the equate plus the comment on the equate (which is assumed to be a description of the bit being tested). In the form: IF (#SH0QCT Partition is cut over jvb11/96)?

Similarly, where a data item with a single bit equate is set (or unset) this is presented as TURN ON (or OFF) with the equate plus comments in parentheses like: TURN ON (#UI2XFM -EXIT THROUGH UIO.);

3. WSL Marked-Up Pseudocode – Module QPD7

This is the Marked-Up WSL Pseudocode for the Module QPD7. The WSL uses a CSS file so that colours can be customised as required. For example: Comments are Grey, TPF Statements are Blue on Grey.
In addition, there are arrows in the leftmost column that link to the Assembler.

```
BEGIN
    <ENTRY POINT>
    <NAME=A_00000000>
   EB0EB := r9;
   * Overriding Parameters-  XREF(SHORT)
   * PCONTROL(GEN,MCALL,ON)
→  GLOBZ_REGR "REGR=RG1"
   {TPF: Get core address of globals};
   *                                                        JMB1196
   SET UP LNIATA OF TERMINAL
→  EBROUT := EBW003[1..3];
   RETRIEVE AAA
→  ENTRC "WGR1"
   {TPF Session Management - AAA Retrieval}
   {Enter program WGR1 with expected return};
   WAS AAA RETRIEVAL A SUCCESS?     JMG
   NO  - BRANCH TO SEND ERROR       JMG
→  IF !XC bit_test?(EBW040, 0x80)
     THEN *
          AAA RETRIEVAL ERROR              JMG
→         SERRC "E,120E50"
          {TPF: Issue system error and exit}
     ELSE BASE OF AAA
→         r5 := CE1CR1;
→         !P bit_and_1(0xFE, 1 VAR EBRS01);
          WA0AA := r5;
→         !P bit_and_1(0xEF, 1 VAR WA0ET3{0E9  END TRANSACTION CONTROL BYTE 3});
→         !P bit_and_1(0xFD, 1 VAR WA0ET2{0E8  END TRANSACTION CONTROL BYTE 2});
          CLEAR NEW SEG IND           CM 10/81
→         !P bit_and_1(0xFE, 1 VAR WA0ET1{0E7  END TRANSACTION CONTROL BYTE 1});
          MULTI-HOST BYTE             MKM59001
→         r3 := !XF address_of(WA0UID{037   MULTI-HOST AIRLINE USER ID});
          LL 59001
→         MHINF "ECB,REG=RGB";
          ERROR RETURN?               MKM59001
          YES - BRANCH                MKM59001
→         IF !XF address_of(WA0UID{037   MULTI-HOST AIRLINE USER ID}) = 0
            THEN QPD70300()
            ELSE
               ===== 4 Comments hidden (click to expand) =====
→                GLOBZ_REG "REGS=R14"
                 {TPF: Get core address of globals};
               ===== 43 Comments hidden (click to expand) =====
→                @GLOBYS := r14;
                 r15 := @SH0HC;
                 HAS CITY BEEN CONVERTED      JMB1196
                 SH0HS := r15;
                 NO, USE OLD PACKAGE          JMB1196
→                IF (#SH0QCT Partition is cut over        jvb11/96)?
                   THEN LET QUAL KNOW WHERE COMING FRMMB1296
→                       EBT000[1..4] := "QPD7";
                        YES USE NEW PACKAGE          JMB1196
→                       ENTNC "QUAL"
                        {CONTINUOUS QUEUE (QLI)}
                        {Enter program QUAL with no return expected}
                   ELSE *                                                JMB1196
                        JMB1196
→                       ENTRC "IGR1"
                        {Ignore PNR}
                        {Enter program IGR1 with expected return};
→                       ENTRC "WGB1"
                        {TPF Session Management - AAA Init/Lock}
                        {Enter program WGB1 with expected return};
→                       r5 := CE1CR1;
```

```
→                       EBCM01 := 0x40;
→                       !P bit_or_1(0x01, 1 VAR EBCM03);
                        IS THERE A QRAK FILE ADDR?   EB02/88
                        WA0AA := r5;
                        YES - CONTINUE PROCESSING    EB02/88
→                       IF WA0GMA{0DC   GENL OR SPVY MSG OR 06-QRAK FILE ADRS} = 0
                          THEN IN PROGRAM IN ERROR          EB02/88
→                               EXITC 0
                                {TPF: Terminate transaction}
                          ELSE *
                               LOAD ADDRESS OF QRAK
→                              EBCFA5 := WA0GMA{0DC   GENL OR SPVY MSG OR 06-QRAK FILE ADRS};
                               AND ID                    BLM 11/85
→                              CE1FA5[1..2] := 0x0000;
→                              EBCRC5 := WA0GMD{0DB   GENL OR SPVY MSG OR 06-QRAK RCD CODE CHECK};
                               RELEASE LEVEL 5             JLU 1/91
→                              CRUSA "S0=5"
                               {TPF: Release any core block on level 5};
                               FIND THE RECORD ON LEVEL 5
→                              FINWC "D5"
                               {TPF: Issue I/O request to read file on level 5 and wait for it to be loaded};
                               IF FINWC_ERROR = 1
                                 THEN QPD70300()
                                 ELSE LOAD THE BASE OF LEVEL 5
→                                     r6 := CE1CR5;
                                      IS THIS A-06-QRAK ?
                                      NO  -SEND AN ERROR
→                                     IF a[CE1CR5, 2] = "P4"
                                        THEN INDICATE THIS IS AN A-06-QRAK
                                             WA0AA := r5;
→                                            TURN ON (#WA0QRK F/A IN WA0GMA IS OF 06-QRAK, NOT 06-OMSG);
                                             BID
→                                            CE1FA5[1..2] := "P4";
                                           ===== 4 Comments hidden (click to expand) =====
                                             P40P4 := r6;
→                                            P40ALT{011    QMR INDICATORS} := #P40SUB;
                                             DO NOT CLEAR LNIATA            A8876
→                                            P40EAC[1..24] := 0;
                                             CLEAR QRAK RECORD
→                                            P40QIF{08C    QUEUE PLACEMENT WORK AREA} := 0;
                                             CLEAR QRAK RECORD            SRA2783
→                                            P40Q2C[1..16] := 0;
→                                            P40EAC[1..3] := "QR+";
                                             INDICATE QLP MODE
→                                            TURN ON (#WA0QLP CONTINUOUS      PNR Q DISPLAY   (QLP));
                                             RT6/13
→                                            ENTNC "QPD1"
                                             {Dead, SERRC E coded}
                                             {Enter program QPD1 with no return expected}
                                        ELSE QPD70300()
                                     FI
                               FI
                        FI
                  FI
            FI
      FI

      PROC QPD70300() ==
        *
        MKM59001
→       SERRC "R,03EEEE"
        {TPF: Issue system error};
        EB091686
→       CRUSA "S0=5"
        {TPF: Release any core block on level 5};
        EB091686
→       r7 := 1;
        EB091686
→       ENTNC "QPD4"
        {Dead, SERRC E coded}
        {Enter program QPD4 with no return expected}
      END
    END
```