# TRANSACT/XL: Strategy for Migration to Native Mode

by Gary Peck
Hewlett-Packard
19111 Pruneridge (Mail Stop 44MH)
Cupertino, CA 95014

* * * * *

ABSTRACT

Native Mode TRANSACT is a way to take full advantage of the capabilities of the Series 900. It compiles TRANSACT source or P-code into Native Mode object code.

This presentation covers:

Steps in the TRANSACT/XL migration process

Ease of conversion from MPE/V to MPE/XL and from CM to NM

Minor exception conditions that may require code modifications

Practical experience from actual application migrations

Expected performance

* * * * *

## TRANSACT/XL COMPILER

In the past, a migration to a new computer has been a scary prospect. The question boils down to this: Is it possible to take thousands of lines of source code that were running on your existing computer, recompile it with a NEW compiler on a NEW computer, and have them run better than before with high reliability and exceptional performance? This is clearly possible with Hewlett-Packard's new TRANSACT/XL compiler. In addition, a carefully executed migration plan practically guarantees the success of the migration.

On the Series 900 machines, TRANSACT programs can run in Compatibility Mode (CM) or Native Mode (NM). Compatibility Mode is a program environment that executes classic 3000 instructions on the Series 900 machine. Native Mode uses the innate features of the new instruction set and the MPE/XL operating system to take full advantage of Precision Architecture. In CM, TRANSACT source files are processed by TRANCOMP into P-code files, which are then executed by the TRANSACT processor just as on classic 3000 machines under MPE/V. In NM, TRANSACT provides a compiler that reads either ASCII source files or TRANSACT P-code files and generates Native Mode object modules.

MIGRATION PROCESS

A successful migration will result by following the stages in the migration process: training, planning, preparation, installation, CM operation, and NM operation. Briefly, the migration site receives training on MPE/XL the migration process; the project goals, milestones, and tasks are planned; the staff completes tasks in preparation for arrival of the Series 900 equipment and software; the hardware, MPE/XL and subsystems are installed; the staff restores MPE/V applications and tests in CM; and the staff compiles and tests applications in NM. The migration process is covered in the tutorial presentation, "Steps to a Successful Migration" and in the XL migration guides.

I will focus on aspects of migration peculiar to TRANSACT. In the planning and preparation stages, migration tool utilities are usually used to flag potential incompatibilities with the NM side and to predict resource utilization. These utilities are not relevant to TRANSACT applications since TRANSACT applications are NOT made up of program files. Instead TRANSACT applications are groupings of P-code (IP) files (more like data) which are interpreted by the TRANSACT run-time processor. The only program files are TRANCOMP and TRANSACT themselves which have been converted and tested in CM for you by Hewlett-Packard. The TRANSACT/XL compiler runs in Native Mode.

There are few, if any, changes to be made when converting from a TRANSACT/V source program to a TRANSACT/XL source program. Therefore, NO automatic conversion utility has been provided. Changes are made manually with a text editor of choice. Virtually all TRANSACT features are supported by TRANSACT/XL. The few features not supported are:

Run-time resolution of data item definitions from a dictionary
Test modes
INITIALIZE built-in command
CALLs to TRANSACT/V, REPORT/V, or INFORM/V (TRANSACT/XL calls are supported)

TRANSACT language features that are specific to the MPE/V environment but not applicable in the MPE/XL environment are ignored by the TRANSACT/XL compiler (e.g., NOLOAD, SWAP). Some additional considerations for the PROC verb will be covered in a moment.

New features designed to maximize effectiveness of XL applications can be quickly incorporated. These are principally new compile options available in the "INFO=" string of the TRANSACT/XL compile commands, TRANXL, TRANXLLK, and TRANXLGO. These are the same kinds of compile commands used by other NM languages. The compile options, including OPTIMIZE and SUBPROGRAM, will be discussed later.

The existing TRANSACT language feature set, Dictionary/V, System Dictionary, and Native Language Support are supported at compile time. TurboIMAGE, MPE/V and MPE/XL file systems (KSAM and MPE files), VPlus, NLS, and both the IEEE and HP standard floating point formats are supported at run-time.

STEPS TO MIGRATE TRANSACT/V to TRANSACT/XL

1) RESTORE your TRANSACT/V source files onto the MPE/XL system.

2) TEST in Compatibility mode (CM). Look for "setup" mistakes because you are trying to duplicate your production environment on a new machine. Look for differences, if any, between the XL environment and the classic 3000.

NOW on to Native Mode...

3) Examine each program for the following special conditions.

3a) Does it use the PROC verb to call system intrinsics?

Make sure each intrinsic is defined using the DEFINE(INTRINSIC) statement or use the new compiler option PROCINTRINSIC. These measures are not required for TurboIMAGE and VPlus intrinsics.

3b) Does it use the PROC verb to call option-variable system intrinsics with 32-bit parameters? (i.e., FCHECK, FDELETE, FGETINFO, FOPEN, MYCOMMAND, STACKDUMP, WHO)

Explicitly pass the 32-bit parameter. For example, in the following code, pass the "filesize" parameter replacing the two commas currently used to denote a null filesize with the filesize parameter and a single comma.

```
system exam1;
define(item) file-name x(20):
             foption   i(4):
             aoption   i(4):
             filenum   i(4):
             filesize  i(9):   <<32 bit integer>>
             bitmap    i(4);
define(intrinsic) fopen;
list file-name:
     foption:
     aoption:
     filenum:
     filesize,init:
     bitmap;
move (file-name) = "OLDFILE";
let (foption) = 5;               <<old ascii file>>
let (aoption) = 0;               <<read access>>
let (bitmap) = 7176;             <<1110000001000 passing 1st three>>
                                 << and filesize; this bitmap is  >>
                                 << valid in CM, but is ignored in NM >>
proc fopen(%(file-name),#(foption),#(aoption),
        <<,,,,,,,,,,, >>         <<old place-holding commas       >>
          ,,,,,,#(filesize),,,, <<each comma denotes a parameter >>
                                 <<note that there is one less comma>>
           &(filenum),#(bitmap));
```

3c) Does it use the PROC verb to call subroutines written in other languages?

The alternatives are to write a STUB for the routine or rewrite it in a Native Mode language. There may be differences between MPE/V based compilers and MPE/XL based compilers. Please refer to the individual language migration guides. For example, the MPE/XL based COBOL compiler converts hyphens to underscores. The MPE/V based COBOL compiler leaves hyphens as is.

3d) Does it use the CALL verb to call INFORM/V or REPORT/V?

Since this feature is not supported, you can choose from these workarounds (examples follow):

Continue to run the program in compatibility mode
Rewrite the INFORM/V report in Transact and compile it with Transact/XL
Use process handling to invoke Inform/V
Convert the report to BRW/XL and use the PROC verb to call the BRW/XL intrinsics

The following is a TRANSACT program that executes a BRW report. Prior to running the program, a BRW report was designed and compiled into a BRW/XL execution file named BRWEXECR.

```
system brw1,vpls=myff(mainmenu(selection));
define(item) brw-comarea      x(106):
             brw-status       i(4)=brw-comarea:
             brw-error        i(4)=brw-comarea(3):
             brw-com-length   i(4)=brw-comarea(5):
             brw-exec-file    x(36)=brw-comarea(7):
             brw-defaults     i(4),init=0:
             selection        i+(1);
list brw-comarea:brw-defaults:selection;
let (brw-com-length) = 50;
get(form) mainmenu;
if (selection) = 1 then
   do
   proc brwinitrequest ((brw-comarea));
   move (brw-exec-file) = "BRWEXECR  ";
   proc brwstartrequest ((brw-comarea),(brw-defaults));
   display brw-status:brw-error;
   proc brwstoprequest ((brw-comarea));
   doend;
end;
```

3e) Does it use the CALL verb to call a TRANSACT/V program?

Compile the called TRANSACT/V program with the TRANSACT/XL compiler using the SUBPROGRAM option. Place the called program in an RL or XL to be resolved during linking or loading.

3f) Does it access files that contain real numbers?

The code generated by the TRANSACT/XL compiler supports both IEEE and HP floating point formats. Under NM MPE/XL, internal storage of real numbers is in the IEEE format. Translation between IEEE and HP formats from and to files and databases is done after the read and before the write on I/O. If no format is specified for a file or database, IEEE numbers are assumed. The compiler option HP3000_16 is available for defining a floating point format for all the files. If the floating point format for individual files is different from that specified by the compiler option, you can express the requirements in the FILE or BASE specification of the SYSTEM statement. This is done by putting HP3000_16/32 in the "file-option-list" or in the "basetype" (follows "optlock"). Because internal representation of real numbers is different between MPE/V and MPE/XL, individual values may change slightly during conversion.

The following program illustrates converting real numbers from the MPE/V format to the MPE/XL standard format. Note that the HP3000_16 option is applied to the input file and the HP3000_32 option is applied to the output file. This causes item-name "R4" to be read as an MPE/V format real number and to be written as an MPE/XL standard format real number.

```
system convrt,file=in(read(hp3000_16))
              ,file=out(write(hp3000_32));
define(item) x2 x(2):
              i4 i(4):
              i9 i(9):
              r4 r(4);
list x2:i4:i9:r4;
find(serial) in,perform=100-convert;
exit;
100-convert:
    put out;
    return;
```

3g) Does it resolve variable definitions at run-time?
(e.g., DEFINE(ITEM) itemname *;)

Define all variables at compile time.

3h) Does it rely upon the INITIALIZE command to execute the next program?

Change user procedures to exit the program and RUN a second program at the MPE/XL command level.

4) Determine which compile options are needed. Supply these in the "INFO=" parameter on the TRANSACT/XL compile commands.

Like compatibility mode TRANCOMP, the TRANSACT/XL compiler allows you to control certain compilation features by supplying compiler options via the INFO= parameter. These options can be included on any of the commands that are used to invoke the TRANSACT/XL compiler: TRANXL, TRANXLLK, TRANXLGO, and RUN TRAN.PUB.SYS. The new compiler options are:

DYNAMIC_CALLS generates dynamic calls for all CALL statements in the program. This allows a program to be executed even if some of the programs that it calls are not available at load-time.

HP3000_16 causes the program to use the HP floating point format for all files and databases. If the NOHP3000_16 option is specified, then all files are expected to use the IEEE floating point format.

PROCALIGNED_16, PROCALIGNED_32, PROCALIGNED_64 cause the compiler to assume that all 16/32/64-bit aligned parameters are correctly aligned on 16/32/64-bit boundaries. Using this option improves run-time efficiency, since the compiler only generates a run-time check to ensure that these parameters are correctly aligned.

PROCINTRINSIC option is identical in effect to declaring intrinsics with a DEFINE(INTRINSIC) statement, but is less efficient.

SUBPROGRAM is used when compiling a program to be called by another TRANSACT/XL compiled program. No outer block is generated. The TRANSACT/XL compiler creates a single RSOM file regardless of how many SYSTEM statements are in a source file. When a source file contains more than one system, the default is to compile the first SYSTEM encountered with option NOSUBPROGRAM and the remaining with the option SUBPROGRAM as they are assumed to be subprograms called by the first system. Using the SUBPROGRAM compiler option causes all the systems in the file to be compiled with the SUBPROGRAM option.

OPTIMIZE directs the compiler to generate level 1 optimized code. Using this option causes the compile to be slower, but produces object modules that are more efficient at run-time.

5) Compile the programs under TRANSACT/XL.

6) Examine the compile listings for errors.

7) Test applications as extensively as possible. If discrepancies or defects are identified, please verify these under the TRANSACT/V processor. Please do as much as you can to isolate new defects in your applications from those in TRANSACT/XL. Report defects to the Response Centers.

Our initial migration sites have been converting thousands of lines of code with few unexpected errors in the applications and almost no defects in TRANSACT/XL. Most snags that I have observed tend to fall into three categories: setup (wrong capabilities; UDC not set); XL learning curve (LINKEDIT; XL user libraries); and test suites not fully debugged on the classic 3000.

Because of the additional testing prescribed by the migration process, a new error is just as likely to be an undiscovered defect in the application as it is a defect in TRANSACT/XL. I emphasize the importance of using a fully tested test suite so that you are fairly certain of testing the application and not the test data. Some defects may be caused in the MPE realm and come to the surface in a TRANSACT program. Be aware of differences in subsystems as well. VPlus, TurboIMAGE, and network products all have their own migration guidelines. For example, in VPlus, performing character mode I/O while VPlus "owns" the terminal will cause VPlus or the driver to hang (Vturnoff and Vturnon intrinsics alleviate this).

Preliminary performance data shows that a TRANSACT program compiled in NM is performing within the same tolerances as other NM languages. A TRANSACT/XL performance test was just completed by a major HP customer. The application was an interactive materials inventory and maintenance system comprising 500,000 lines of TRANSACT code, 260 VPlus screens, and 6 TurboIMAGE databases. A menu was used to select functions performed by individual TRANSACT programs. The menu was compiled into a Native Mode main program and the TRANSACT subprograms were compiled into a XL library. The Series 950's maximum throughput at the saturation point was as much as 2.8 times the Series 70 running FASTRAN compiled TRANSACT code. The customer also found that the maximum number of recommended users supported by the S/950 was 2 to 3 times the practical limit of users on their S/70.

TRANSACT migration to the Series 900 should be smooth and straightforward. Customer enthusiasm is running high because of the exciting performance, product reliability, ease of migration, and expert support channels.

BIBLIOGRAPHY

TRANSACT/XL Migration Guide
TRANSACT/3000 Reference Manual, Rev. 10/87, Appendix H (future update)