MicroPower/Pascal Release Notes

Order No. AA–FQ16E–TK

digital

software

MicroPower/Pascal Release Notes

Order No. AA–FQ16E–TK

June 1987

This manual documents late techincal changes, restrictions, and corrections not covered elsewhere in the MicroPower/Pascal V2.4 documentation.

Operating System and Version:	Micro/RSX Version 3.0 RSX–11M Version 4.2 RSX–11M–PLUS Version 3.0 RT–11 Version 5.2 VAX/VMS Version 4.0
Software Version:	MicroPower/Pascal–Micro/RSX Version 2.4 MicroPower/Pascal–RSX Version 2.4 MicroPower/Pascal–RT Version 2.4 MicroPower/Pascal–VMS Version 2.4

Digital Equipment Corporation   Maynard, Massachusetts

Contents

Preface

MicroPower/Pascal software consists of tools for developing dedicated real-time applications for LSI-11, LSI-11/2, LSI-11/23, LSI-11/23-PLUS, LSI-11/73, PDP-11/03, PDP-11/23, PDP-11/23-PLUS, PDP-11/73, PDP-11/83, MicroPDP-11/53 (RAM only), MicroPDP-11 microcomputers and CMR-21, FALCON, FALCON-PLUS, KXT11-CA, and KXJ11-CA single-board computers as target systems.

Structure of This Document

This manual documents software restrictions not covered elsewhere in the MicroPower/Pascal manual set and provides document corrections and clarifications. The manual also provides an annotated list of all run-time software files on your distribution kit media and, for RT-host-system users, a directory listing of the MicroPower/Pascal-RT distribution kit contents.

This manual is divided into three chapters:

- Chapter 1 lists restrictions you must observe when using MicroPower/Pascal software, as well as clarifications of obscure error conditions.
- Chapter 2 lists corrections to and additional material for the other books in the MicroPower/Pascal manual set.
- Chapter 3 contains an annotated listing of the MicroPower/Pascal run-time software files.

Intended Audience

MicroPower/Pascal application designers and programmers should be familiar with the information in this manual before attempting to design, code, or build application software. The manual discusses host-independent problems and problems that are specific to a particular host development system (VMS, RSX, or RT).

Chapter 1
Clarifications, Restrictions, and Warnings

1.1 Compatibility of Different Versions

Any new version kit is a complete distribution-media replacement for the previous MicroPower/Pascal release. Assume that every system software component has been modified in a way that may be incompatible with other components from a previous release. Whenever a new version update is adopted for use, all the preexisting user-developed source modules that make up an application must be recompiled or reassembled and rebuilt using only MicroPower/Pascal system software components from that version update. All command files must be recreated to prevent skew errors that are potentially difficult to diagnose from arising during later maintenance of the application.

Also, components of the MicroPower/Pascal system software from the latest version update cannot be used with other components from a previous version. That is, components from different software versions cannot be "mixed and matched."

1.2 Differences Between V2.4 and V2.3

MicroPower/Pascal V2.4 is a maintenance release that provides fixes for V2.3 software problems. It also provides new functionality, which is listed in Section 1.2.1.

1.2.1 New Features for Version 2.4

- New interface routines
    CREATE_BINARY SEMAPHORE_P
    CREATE_COUNTING_SEMAPHORE_P
    CREATE_RING_BUFFER_P
    CREATE_QUEUE_SEMAPHORE_P
- FALACP.PAS—small ACP for FALCON
- Compiler and OTS
    - HEX, BIN, and OCT permitted for INPUT of INTEGERs, UNSIGNEDs, and LONG_INTEGERs (MicroPower/Pascal–VMS only)

- Terminal (TT) driver
  - New STOP request
  - Framing characteristics selection
- New exception codes
  - ES$ABO, I/O aborted
  - ES$NIP, No I/O in progress

1.2.2 Software Fixes

1.2.3 Compiler—NAME Attribute String Size

The size restriction on the NAME attribute string (6 characters) was not enforced.

1.2.4 Compiler—Default Positional Actual Parameter

The use of a default positional actual parameter after a nonpositional parameter was not flagged.

1.2.5 Compiler—Parameter File Variable Generates Bad Code

Use of a file variable that was a VAR parameter in a READ or WRITE generated superfluous MACRO-11 code that referenced a nonlocal R/W p-sect. Therefore, the code was not reentrant.

1.2.6 Compiler—Assignments to UNSAFE Variables

For assignments to a variable that had the UNSAFE attribute, the compiler did not detect the invalid case in which the representation of the expression was larger than the variable's storage allocation. Contrary to what is stated in the *MicroPower/Pascal Language Guide,* the compiler did not issue an error diagnostic but instead generated code that could provide invalid results.

1.2.7 Compiler—Component of Non-PACKED ARRAY Disallowed As Actual VAR Parameter

The compiler got an error when a component of a non-PACKED ARRAY constant was passed as an actual VAR parameter.

1.2.8 Compiler—INITIALIZE or TERMINATE Attribute on PROCESSes

The use of either the INITIALIZE or TERMINATE attribute on PROCESSes or FUNCTIONs did not generate a diagnostic message.

1.2.9 Compiler—Pointer Type Reference Removed Attributes

If a type identifier was referenced in a pointer type definition in a procedure that is more deeply nested than the one in which the identifier was declared, any attributes of that type would not be in effect at that nested level and deeper nesting. A type defined in PREDFL.PAS lost its attributes after a pointer type referenced it anywhere in the program.

1.2.18 Compiler—Multiple Use of PREDFL.PAS Identifiers Has Side Effect

If a PREDFL.PAS identifier (for example, STATUS or NAME) was declared as a variable within an application and if certain indeterminate conditions were present, then using the variable as the actual parameter for a VAR parameter would not flag the variable as being modified. A subsequent reference to the variable might then result in a "Must assign value before using variable" error.

1.2.19 Compiler—Bad ISD for Pointer to Large Record

If a pointer variable to a large record (more than 12 component fields and subfields) was declared, its ISD record may have been incorrectly generated; you may not have been able to examine its contents through PASDBG.

1.2.20 Compiler—Wrong ISD for LONG_INTEGER Component

If the component type of an array, a conformant array, a pointer, or a file was LONG_INTEGER, its ISD was erroneously generated as UNSIGNED. This caused problems when you tried to examine one of those components, using PASDBG, because a LONG_INTEGER value uses 4 bytes but an UNSIGNED value uses only 2 bytes.

1.2.21 Compiler—Bad LONG_INTEGER Constants

The following ranges of LONG_INTEGERs were not generated correctly:

```
26048 + m * 65536 + n          m = 0..65535, n = 0..63
58816 + m * 65536 + n          m = 0..65535, n = 0..63
1707081728 + n                 n = 0..4194303
-440401920 + n                 n = 0..4194303
271977446 + 2097152 * n        n = 0..63
```

LONG_INTEGER variables were fine. The problem was just with constants from this list.

1.2.22 Compiler—Bad OF Constant Caused Infinite Loop

An error in specifying the repetition constant in the OF expression of a structured constant caused the compiler to go into a seemingly infinite loop.

1.2.23 Compiler—File Buffer Reference Could Not Be Actual Parameter

Bad code was generated when passing a file buffer reference (f^) as an actual parameter.

1.2.24 Compiler—RSX Compiler Aborted with Too Many Identifiers

If the compiler identifier limit of 997 was reached in the RSX compiler, the compiler aborted with an "Odd Address Trap" rather than terminating with a diagnostic message.

1.2.25 MIB—Failed to Build an I&D Process

In MicroPower/Pascal-RT, MIB failed to build an I&D process greater than 64K bytes when you specified the /S switch.

1.2.35 PASDBG—SET, CANCEL, and SHOW BREAKPOINT 000000

Setting two breakpoints, canceling the first, and setting another breakpoint resulted in altering the physical address of the second breakpoint to 000000. The breakpoint was still set at the correct location on the target and did function correctly.

1.2.36 PASDBG—STEP n Did Not Work Properly

The PASDBG command STEP n (where n was the number of statements to be stepped over) did not work properly. One line was stepped over no matter what the value of n.

1.2.37 PASDBG—VMS Access Violation Following SET HOST

Running PASDBG from a system to which one has SET HOST caused a VMS access violation error when single STEPping or GOing to a breakpoint. The access violation error was followed by a register dump and a return to the VMS prompt.

1.2.38 PASDBG—STEPping in High Places

When in MACRO mode, attempting to single STEP through instructions did not work if the address of the instruction was equal to or greater than 1000000. The application proceeded as if a GO had been issued.

1.2.39 PASDBG—Redefined Terminal Characteristics

PASDBG redefined some terminal characteristics, such as escape, but did not reset them upon exit. This could cause problems when the user tried to use an editor after using PASDBG.

1.2.40 PASDBG—LOAD/EXIT Appeared to Hang System

PASDBG could apparently hang the system during loads when the /EXIT switch was used with the LOAD command. This was caused by an unnecessarily long timeout value for the initial six packets (approximately 20 minutes) being sent to the target. In V2.4, the timeout value is set to 2 minutes for the first six packets and 4 seconds for the remaining packets.

1.2.41 OTS—EXP Function Problem in non-FPP Libraries

The EXP function died with a "Trap to 10" error if the mantissa of the result was zero. This error occurred if a non-FPP library (for example, EIS or NHD) was used.

1.2.42 OTS—STATE_CODE_MODIFIER_TYPE Declaration Wrong in PREDFL.PAS

The declaration of STATE_CODE_MODIFIER_TYPE in PREDFL.PAS did not account for the SM.ABI bit. In V2.4, the Boolean ABORT_TO_INACTIVE has been added to this type declaration to account for this bit. Also, RES3 and RES4 have been renamed to FPA_PENDING and BLOCKED_ON_COMPLEX, respectively.

1.2.43 OTS—FSPAS.PAS Program Name Changed

FSPAS.PAS's program name now matches its file name. Also the start and completion messages have been changed accordingly. The new first line of the program is:

```
[System(MicroPower), Data_space(3000), Stack_size(400)] PROGRAM FSPAS;
```

1.2.51 XE Driver—May Send Garbage

If you did not provide the reply semaphore parameter on those XE driver interface routines (XESUB.PAS) that take a reply semaphore as an optional parameter, you may have gotten unexpected errors.

Each interface routine used a local variable to format a request for the XE driver. Since the variables were local, they were allocated off the stack. Since MicroPower/Pascal did not initialize those variables on entry to the routines, their content was unpredictable. If the reply semaphore field of the request contained garbarge as a result of a null parameter, there was likely to be a severe error when the driver attempted to reply to the request.

1.2.52 YF Driver—Port-C Bits Not Handled Properly

The port-C bits of the FALCON–PLUS parallel port were not handled properly in the interface routines in YFDRVP.PAS.

V2.4 has two interface routines, $YFRD and $YFWRT, to handle the data properly.

1.2.53 Kernel—RESET_RING_BUFFER Call Resulted in an ES$IPM Exception

A RESET_RING_BUFFER call from Pascal or a RBUF$ call from MACRO would result in an ES$IPM exception.

1.3 Bootstrap Problem

Warning: A halt routine in the bootstrap is entered when a nonexistent memory trap occurs during the booting or loading of an application. The target will crash in ODT, usually as a result of incorrect memory configuration data contained in the configuration file for the target being loaded.

1.4 Pascal Compiler Problems

This section contains notes about all Pascal language constructs except those I/O procedures that depend on MicroPower/Pascal file system support, which may not be included in a given application system.

1.4.1 VAR Formal Parameters with UNSAFE Attribute

Warning: For procedure/function calls and process invocations involving a VAR formal parameter with the UNSAFE attribute, the compiler does not detect the invalid case in which the storage allocation for the actual parameter is smaller than that for the corresponding UNSAFE VAR formal parameter (see the *MicroPower/Pascal Language Guide*). Contrary to what is stated there, the compiler does not issue an error diagnostic but instead generates code that may produce invalid results.

1.4.6 No Warning-Only Diagnostics

Restriction: All errors and other conditions detected by the compiler are effectively fatal. Thus, syntax diagnostics and other compiler messages that are normally warning-only or informational, such as the "not standard Pascal" syntax messages (if requested), prevent the compiler from generating valid code for the compilation unit. If any warning messages are displayed (PASCAL–W–...) and if the compiler does continue code generation, do not attempt to use the generated code.

1.4.7 Use of Set Subset Operator, < =, May Result in Incorrect Code

Restriction: The compiler generates incorrect code for certain set expressions involving dynamic sets and the subset operator <=. In particular, if the dynamic set is tested to be a subset of a nondynamic set, incorrect code results. A dynamic set is specified with a set constructor with at least one element being a variable or expression.

1.4.8 Expressions Involving Empty Set Generate Incorrect Code

Restriction: The compiler incorrectly handles set expressions involving the empty set constant ([ ])—for example, "if seta * [] <> ...." This problem occurs because the compiler interprets the empty set [] as 32 bytes long, even though it is in an expression with a 1-byte set. To avoid this problem, assign the empty set [ ] to a set identifier (variable) of the correct length and replace [ ] in the expression with this set identifier.

1.4.9 Generated Code for Set Intersection Not ROMable

Restriction: The generated code for set intersection (seta * setb) will violate the read-only attribute of the constant-storage (.pcon.) program section if setb is a set constant. You can avoid this problem if you use the technique described in the previous section.

1.4.10 Set Constructors Not Allowed in Structured Constants

Restriction: A set constructor—for example, "[elt1,elt2]"—may not be used as a value in a structured constant. When a variable of a structured data type includes a set, its components may receive values only by explicit assignment.

1.4.11 Nonlocal GOTO Addresses Are Incorrect

Warning: If a program with a nonlocal GOTO—from a nested block to an outer block—is compiled with the /D option, the statement numbers of the label and the GOTO statement will be inaccurate.

1.4.12 BYTE Attribute Does Not Always Work

Restriction: The compiler may generate incorrect code when referencing variables in a packed record that use the BYTE attribute. If a pointer to the record is passed as a value parameter to a procedure, incorrect results may occur. Correct code is produced if the parameter is passed as a VAR parameter.

1.4.20 EXTERNAL Default Variables

Restriction: An invalid OBJ file is produced when compiling a program containing a procedure that uses an EXTERNAL variable as a default parameter. Subsequently, when RELOC processes the file, the error "Bad RLD symbol" is produced.

1.4.21 Using CHAR As FOR Loop Index

Warning: If a program containing a FOR loop with a variable of type CHAR as the index is compiled with the /D (/DE) option, incorrect code will be produced by the compiler.

1.4.22 Undetected Overflows

Warning: Overflow is not detected for add, subtract, multiply, and divide for integers, long integers, and unsigned integers.

1.4.23 LONG_INTEGER VAR Parameters

Restriction: If a program contains a LONG_INTEGER VAR parameter, X, that is used in an expression of the form $X := X + or - ...;$ or in an expression of the form $X := ... + or - X;$ incorrect code will be produced by the compiler. A workaround is to assign X to a temporary variable, use the temporary variable in place of X, and finally assign the temporary variable back to X.

1.4.24 Variable or Type Declaration Size Restrictions

Restriction: A variable or type declaration may not exceed the size of 65536 bytes. Furthermore, a PACKED variable or type cannot exceed the size of 65536 bits (or 8192 bytes).

1.4.25 Constant Multiply Expressions Whose Result Exceeds 32767

Restriction: If you do an assignment to an UNSIGNED from an expression involving integer arithmetic and the resulting value exceeds 1/2 of 65536 (that is, it sets the sign bit), you receive a compile time error indicating integer overflow or divide by 0. For example:

```
VALUE := (252*256) + 251;       { 64763.=176373(8) }
```

A workaround is to not perform the multiplication; explicitly put the result in the code (VALUE := 64763).

1.4.26 Compiler Fails to Generate Lazy I/O Call for File of Array Reference

Restriction: The compiler does not generate a lazy I/O call for the file buffer pointer (F^) in a file of array reference like "item := f^[index]", where f is a file of array elements and the index of the array is given as a variable or expression. (Files of records without arrays work correctly, as do files of arrays where the indices are given as constants.)

A workaround is to precede a statement such as "item := f^[index]" with an assignment like "temp := f^[1]". In that way, you force the lazy I/O call so that subsequent references to f^ are valid.

The network example in the *MicroPower/Pascal I/O Services Manual* violates this restriction by including references to "f^[j]". See the workaround in Section 2.1.1.

1.6.3 CMR–21 HALT Instruction Will Not Always Work Correctly

Warning: If a HALT instruction is executed while the page control register (PCR) setting is not equal to 0 (user-accessible memory in the range 100000 to 140000), control will not be passed to ODT. This characteristic is inherent in the CMR–21 architecture, whether or not it is documented as such.

1.6.4 FALCON Configuration File Baud Rate Selection When Using FALODT

Clarification: If you specify BREAK=SFWODT to the FALCON configuration macro, you must also specify a baud rate by removing a semicolon in the FALCON or FALCON-PLUS configuration file. The text to be edited is supplied in the prototype configuration files CFDFAL.MAC and CFDFPL.MAC, as follows:

```
; If the BREAK=SFWODT option of the FALCON macro is used, select
; the proper baud rate for the console line from the following
; table.  Remove the ";" from the beginning of the line.  The
; baud rate selected must match the setting of the console
; terminal to ensure correct operation of the Software ODT
; module.
;
; D.SLU1 == 2       ; 300 baud
; D.SLU1 == 12      ; 600 baud
; D.SLU1 == 22      ; 1200 baud
; D.SLU1 == 32      ; 2400 baud
; D.SLU1 == 42      ; 4800 baud
; D.SLU1 == 52      ; 9600 baud
; D.SLU1 == 62      ; 19200 baud
; D.SLU1 == 72      ; 38400 baud
```

1.6.5 Use of Descriptor (DESC) to Create Shared Regions

Restriction: DIGITAL recommends that users NOT create unnamed shared regions. The use of DESC in CREATE_SHARED_REGION calls may not be supported in future versions of MicroPower/Pascal.

1.6.6 MACRO–11 Parsing Problem Affects MEMORY Configuration Macro Usage

Restriction: MACRO–11's algorithm for parsing keyword parameters dictates that you exercise caution in specifying expressions as size arguments to the MEMORY configuration macro. When in doubt, completely enclose size expressions in angle brackets. For example:

```
MEMORY  base=0, size=<124.*32.>+1, type=RAM
```

will be interpreted as:

```
MEMORY  base=+1, size=<124.*32.>, type=RAM
```

unless you use angle brackets as follows:

```
MEMORY  base=0, size=<<124.*32.>+1>, type=RAM
```

Note

This caution applies to any macro with keyword parameters, not just the MEMORY macro.

1.8 Device Driver Problems

1.8.1 Driver Mapping

Clarification: PAR 0 of driver mapped processes is reserved by DIGITAL for future device driver interfaces. Therefore, designers of user-written device drivers should not use PAR 0 in driver mapping for any application that might migrate to a future version of the MicroPower/Pascal software.

1.8.2 DY Driver Does Not Support 22-Bit Addressing

Clarification: The RX02 driver does not support 22-bit addressing. That limitation is imposed by the RXV21 controller, which has an addressing limit of only 18 bits.

1.8.3 XL Driver

The notes in this section, retained from the V2.0 release notes, are included for compatibility purposes. The XL driver, contained in object form on current kits, is being phased out and may not appear in future releases.

1.8.3.1 XL Does Not Support Read/Write to Ring-Buffered Lines

Clarification and Warning: The XL serial-line device driver does not support read or write (block mode) requests to a line that has been connected to a ring buffer; in addition, the device driver does not detect and reject such a request as an error. If a read or a write request is sent to the XL driver for a line already connected to a ring buffer, the request will be queued but never processed. Thus, a process waiting for a response to such a request will be blocked indefinitely.

1.8.3.2 XL Driver Does Not Reply on Connect/Disconnect to Ring Buffer Requests

Clarification: Unlike other driver implementations, the XL driver does not reply to the user on connect/disconnect ring buffer requests.

1.8.4 TK50 Hardware Streaming Requirements

Warning: The TK50 tape drive subsystem will operate in stream mode only if its command buffer is kept supplied with commands in a timely fashion. Typically, that can be achieved only by issuing send/receive-level I/O request packets to the MU driver from a high-priority process and by use of promptly maintained triple buffers for the data going to or from the device. (Obviously, the requesting process cannot afford to be locked out of run state for any significant period of time.)

1.9 Supervisor-Mode Shared Library Problems

1.9.1 Incorrect Exception Reporting

Warning: PASDBG may not report correct exception information if an exception occurs within a supervisor-mode shared library module.

If "HELP keyword" is entered, the help text will be the same as for "HELP MPP".

To install the help file as part of DCL help, follow the same basic procedure, but in step 2, edit the root file LB:[1,2]DCL.HLP rather than MCR.HLP.

1.10.5 Running MPBUILD Command Files Simultaneously from Same Directory

Warning: MPBUILD–RSX command files generate temporary files with the MPPOUT.MPP name, which are deleted after each step. Therefore, running two MPBUILD-generated command files simultaneously out of the same directory produces unpredictable results.

1.11 RT–11 Host System Considerations

The following general information will be useful if you have RT–11 software.

- MicroPower/Pascal–RT software is supported only with the RT–11 XM (Extended Memory) monitor supplied on the distribution kit. If you decide to build (SYSGEN) an unsupported monitor, avoid using CSR address 176500 and vector address 300, which are reserved in the host system for the TD driver. A conflict occurs if you attempt to build a system with multiterminal support, using the default SYSGEN values for the second terminal. Thus, configure your system hardware and software to avoid the conflict.
- The MPP.SAV compiler is supported only as a background virtual job. Therefore, remember to run the compiler by using the R command (R MPP. <RET>), not the RU or RUN command.

1.11.1 System Crashes During Execution of Large Programs or Command Files

Restriction: A problem in the RT–11XM Version 5.2 monitor can cause the system to crash following the .EXIT of either a large program or a program executed from within a large command file. The internal indicator that determines whether KMON has been swapped out is not properly reset, resulting in execution of random instructions or data during the .EXIT operation. The program that was executing has run to completion, so you need not repeat the operation after you have rebooted the system. This problem can occur when executing the MicroPower/Pascal–RT compiler.

1.11.2 Erroneous "KMON–F–Not Enough Memory" Error

Warning: The RT–11XM Version 5.2 monitor can erroneously report a "KMON–F–Not enough memory" error when the RUN command is used to start a virtual job. The job executes successfully with the R command. If this problem occurs, perform the following steps:

1. Unload any unused device drivers.
2. Unload any idle foreground or system jobs.
3. Try to rerun the program.

Chapter 2
Document Corrections and Additions

2.1 MicroPower/Pascal I/O Services Manual

2.1.1 Network Example Violates Current Lazy I/O Restrictions

Restriction: The network example in the *MicroPower/Pascal I/O Services Manual* violates the lazy I/O restrictions stated in Sections 1.4.18 and 1.4.26 of these Release Notes. The example contains the file-of-array reference "f^[j]" and uses that reference as a procedure parameter ("write(output,f^[j])".)

The workaround version of the example is:

```
[SYSTEM(MICROPOWER)] PROGRAM TM(INPUT,OUTPUT);
CONST
    big_array_size = 1024;
TYPE
    big_array = ARRAY [1..big_array_size] OF LONG_INTEGER;
VAR
    f : FILE OF big_array;
    j : INTEGER;
    l : LONG_INTEGER;
BEGIN
  OPEN (f,'SY$NET:"TASK=TM"',HISTORY:=NEW);
  RESET(f);
  l := f^[1];   { Reference f^ with constant subscript to force lazy
                  I/O call, so that later f^ refs are valid }
  FOR j := 1 TO big_array_size DO
    BEGIN
      l := f^[j];  { Assign to var to avoid passing f^ as parameter }
      WRITE(OUTPUT,L);
      IF l <> j
        THEN WRITE(' <----Error, should have been ',j:1);
      WRITELN;
    END;
  CLOSE(f);
END.
```

For mapping type GENERAL (the default type), the hardware mapping registers cannot be accessed directly, but their contents can be ascertained as follows:

```
VAR
    q    : QUEUE_SEMAPHORE_DESC;
    info : INFO_BLOCK;
    addr : UNSIGNED;
    parv : UNSIGNED;
    item : your_favorite_type;
BEGIN
{SEND some data to ourselves, the data being the data item we're
interested in.  Then, using the RET_INFO block available in
RECEIVE, obtain the hardware register contents for the virtual
address of the REF_DATA }
    IF CREATE_QUEUE_SEMAPHORE (DESC := q) THEN ;
    SEND (DESC := q, REF_DATA := item, REF_LENGTH := 1);
    RECEIVE (DESC := q, RET_INFO := info);
    DESTROY (DESC := q);
    addr := info.address.address;       {Get virtual address}
    parv := info.address.par_value;     {Get virtual page base}
        ...
END;
```

The second method may seem cumbersome or unorthodox but it has two advantages over the first (direct mapping register) method:

1. General mapping is less restrictive in terms of the virtual address space available to the static process family. The *MicroPower/Pascal Run-Time Services Manual* details the mapping types, uses, and restrictions.
2. The I&D-space separation feature of J11-based processors does not affect the operation of the second method. The INFO_BLOCK returned contains the proper values regardless of the processor type and is therefore a more general mechanism. (Processor independence may not be a consideration for applications developed for a well-defined hardware environment.)

Each method returns two variables (ADDR and PARV) that contain enough information to locate the physical address of the data item (ITEM). To translate those values into a 22-bit physical (Q-bus) address, you can do the following:

```
VAR
    bus_address : LONG_INTEGER;         {Enough for all 22 bits}
    offset : UNSIGNED;
BEGIN
    offset := UAND(addr,%O'17777');     {How far from page base}
    bus_address := parv;                {Conv page base to 32 bits}
    bus_address := (bus_address * 64) + offset; {Construct full addr}
        ...
END;
```

The variable BUS_ADDRESS now contains the full 22-bit Q-bus address, such as might be used in a device DMA transaction.

Chapter 3
Run-Time Software Files

This chapter lists the names of files of interest to users of MicroPower/Pascal when building an application. These files may be used in the compile, assembly, and build stages of application building. A brief annotation indicates the purpose of each file.

File Name	Description
ACPPFX.MAC	Ancillary control process prefix file
ADINC.PAS	Include file for A/D driver
ADPFX.MAC	A/D driver prefix file
ADSUB.PAS	A/D driver routines
ARBVFY.COM/CMD	Verification command file for arbiter side with KXT11–CA or KXJ11–CA
ARBVFY.PAS	Verification program for arbiter side with KXT11–CA or KXJ11–CA
CARS2.PAS	Example program
CARS3.PAS	Example program for installation verification
CFDCMR.MAC	Prototype configuration file for CMR–21
CFDFAL.MAC	Prototype configuration file for FALCON
CFDFPL.MAC	Prototype configuration file for FALCON–PLUS
CFDKJJ.MAC	Prototype configuration file for mapped KXJ11–CA
CFDKJU.MAC	Prototype configuration file for unmapped KXJ11–CA
CFDKTC.MAC	Prototype configuration file for KXT11–CA
CFDMAP.MAC	Prototype configuration file for mapped target

File Name	Description
FSPAS.PAS	File system example program
GETPUT.PAS	Pascal driver include file
GETSET.PAS	Pascal driver utility routines
GSINC.PAS	Include file for using GETSET.PAS
INTDIR.PAS	Initialize directory procedure
IOPKTS.PAS	Driver request/reply packet definitions
IOPVFY.COM/CMD	Verification command file for KXT11–CA or KXJ11–CA side
IOPVFY.PAS	Verification program for KXT11–CA
KKINC.PAS	Include file for KK 2-port RAM driver
KKPFX.MAC	KK 2-port RAM driver prefix file for KXT11–CA or KXJ11-CA
KKRWD.PAS	KK driver interface routines
KWINC.PAS	Include file for KW real-time clock
KWPFX.MAC	KW real-time clock driver prefix file
KWSUB.PAS	KW clock driver routines
KXINC.PAS	Include file for KX driver
KXJSHR.PAS	Shared memory routine for KXJ11–CA
KXPFX.MAC	KX driver prefix file
KXRWD.PAS	KX driver interface routines
KXTLO.PAS	KXT11–CA or KXJ11–CA application loader
LIBEIS.OLB/OBJ	EIS version of Pascal OTS
LIBFIS.OLB/OBJ	FIS version of Pascal OTS
LIBFPP.OLB/OBJ	FPP version of Pascal OTS
LIBNHD.OLB/OBJ	NHD version of Pascal OTS
LIBSUP.OLB/OBJ	Library dispatcher for supervisor mode
LIBUSR.OLB/OBJ	Library dispatcher for user mode
LOGNAM.PAS	Logical-name primitive definition file
MISC.PAS	Include file for miscellaneous definitions
MKBOOT.EXE/MKB.TSK	MIM file conversion program for Ethernet down-line loading (VMS and RSX)
MPBUILD.COM/CMD	Automatic command file generator (VMS and RSX)
MPBLD.COM	Automatic command file generator (RT)

File Name	Description
XAPFX.MAC	DRV11-J driver prefix file
XDDRV.PAS	Extended disk driver (source)
XEINC.PAS	I-EEE 488 Instrument Bus include file
XEPFX.MAC	I-EEE 488 Instrument Bus prefix file
XESUB.PAS	I-EEE 488 Instrument Bus routines
XLDRVK.OBJ	XL driver object file for KXT11-CA
XLDRVM.OBJ	XL driver object file for mapped target
XLDRVU.OBJ	XL driver object file for unmapped target
XLPFX.MAC	Serial-line driver prefix file
XLPFXD.MAC	Serial-line driver (w/debug) prefix file
XLPFXF.MAC	FALCON serial-line driver (w/debug) prefix file
XLPFXK.MAC	Serial-line driver for KXT11-CA
XPPFX.MAC	DPV11 driver prefix file
XSPFX.MAC	KXT11-CA or KXJ11-CA synchronous driver prefix file
YADRV.PAS	DRV11 driver
YAPFX.PAS	DRV11 driver prefix file
YBPFX.MAC	DRV11-B driver prefix file
YFDRVI.PAS	FALCON include file for YFDRVP.PAS
YFDRVP.PAS	FALCON parallel-port subroutines
YFPFX.MAC	FALCON parallel-port prefix file
YK.PAS	YK driver interface routines
YKINC.PAS	YK driver include file
YKPFX.MAC	YK driver prefix file for KXT11-CA or KXJ11-CA

HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Alaska, Hawaii, or New Hampshire	603–884–6660	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Rest of U.S.A. and Puerto Rico*	800–258–1710	
* Prepaid orders from Puerto Rico must be placed with DIGITAL's local subsidiary (809–754–7575)		
Canada	800–267–6219 (for software documentation) 613–592–5111 (for hardware documentation)	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: Direct Order desk
Internal orders (for software documentation)	—	Software Distribution Center (SDC) Digital Equipment Corporation Westminster, MA 01473
Internal orders (for hardware documentation)	617–234–4323	Publishing & Circulation Serv. (P&CS) NR03–1/W3 Digital Equipment Corporation Northboro, MA 01532

MicroPower/Pascal
Release Notes
AA–FQ16E–TK

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ____________

Name ____________ Date ____________

Organization ____________

Street ____________

City ____________ State ____________ Zip Code or Country ____________