PDP-11/34 Ethernet Software

BY **Gregory L. Thomas**

September 6, 1979

Filed on: [MAXC]<ASDSoftware>PDP-Software.press

XEROX
XBS / ADVANCED SYSTEMS DEPARTMENT
701 Aviation Boulevard / El Segundo / California 90245

PDP-11/34 Ethernet Software

The software which is provided to support the Ethernet hardware consists of five distinct levels:

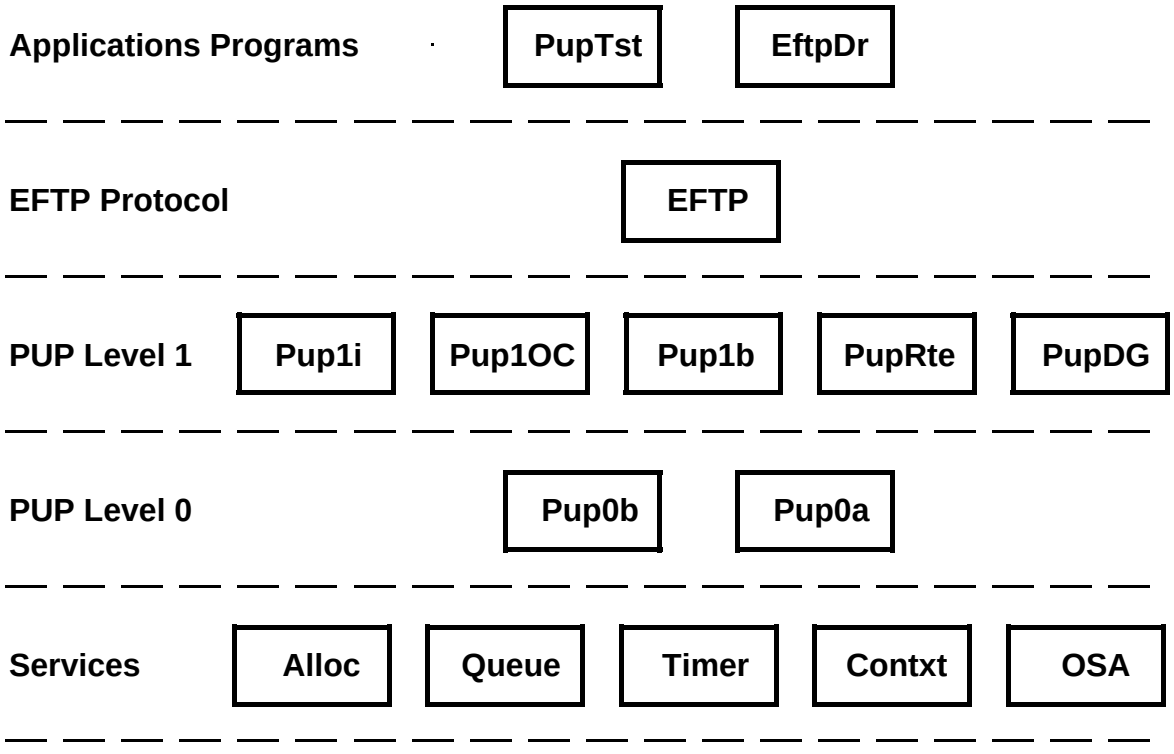
Services --- This is a collection of routines for memory allocation, queuing, timing, tasking, and moving data.

PUP Level 0 --- These routines provide the user's interface to the Ethernet hardware. They output and input PUP's to and from queues and handle interrupts.

PUP Level 1 --- The routines at this level provide for building PUP's, calculating checksums, routing via gateways, decommutating to sockets, socket management, and PUP validity checking.

EFTP Protocol --- A single-ACK-per-packet reliable protocol is provided in this package along with mechanisms for transferring data to and from packets. This protocol is compatible with EFTP on the Alto.

Applications Programs --- Two programs are provided. The first is a test routine which tests the correct operation of the hardware and software by forwarding a PUP from one host to another. The second is a file transfer routine (usable under RT-11 only) which allows for the transfer of files between PDP-11's and Altos.



The Ethernet software was written to be run in one of three modes: standalone, under RT-11, or under RSX-11M. In order to accomplish this it was necessary to provide multiple versions of those routines which access the hardware or make monitor calls. In addition, RSX-11M loadable drivers are provided with the software.

The PDP-11 Ethernet software is composed of and was generated from the following Alto packages:

EFTP Protocol
PUP Levels 1 & 0
Context
Queue
Timer

All of these packages are documented in a manual entitled "Alto Software Packages" which is stored in two sections at:

[MAXC]<ALTODOCS>PACKAGES1.PRESS
[MAXC]<ALTODOCS>PACKAGES2.PRESS

In addition, the modules "OSA", which contains some of the routines and functions of the Alto operating system, and "Alloc" are documented in the "Alto Operating System Reference Manual" at:

[MAXC]<ALTODOCS>OS.PRESS

All Alto BCPL code was syntactically modified to conform to the requirements and limitations of the PDP-11 DOS compiler, but is otherwise unchanged. The BCPL compiler generated MACRO-11 source code with the name *.PAL. All assembly language code was rewritten in MACRO-11 and given the name *.MAC, *.RT, or *.RSX. Because the code is virtually unchanged from the Alto implementation, the Alto documentation is totally definitive and trustworthy with two exceptions:

(1) Because of a PDP-11 BCPL limitation, defaulted arguments must be set to zero (rather than omitted).
(2) Timer increments were changed from .01 seconds to .1 seconds because of the resolution available from a 60-cycle clock.

The module OSA.*, which simulates the Alto operating system, contains the entry point for the program and sets up memory areas for the allocation of stacks and dynamic memory pools. It also provides procedures to increment the variable BINCLK every 100 milliseconds. These two functions must be modified as a function of the operating system and the hardware clocks available to the system.

The module PUPTST.* contains a test routine which will forward a pup from one host to another using EFTP protocol. The destination host is predefined as being 344#. Two Altos using EFTP.RUN can be used to perform this test.

Source File Information

The source code is available in three forms: (1) as a dump file on a file server, (2) on an RT-11 floppy, and (3) on an RSX-11M floppy.

The dump file is located at [MAXC]<ASDSoftware>PDP-Ethernet.dm and consists of the following files:

DESCRIPTION	BCPL	MACRO11	COMMENTS
Applications	DLIB .HDR		
Programs	PUPTST.BPL	PUPTST.PAL	
	EFTPDR.BPL	EFTPDR.PAL	
		EFTPSB.MAC	RT-11 Subroutine for EFTPDR
EFTP Protocol	EFTP .HDR		
	EFTP .BPL	EFTP .PAL	
PUP Level 1	LEVEL1.HDR		
	PUP1I .BPL	PUP1I .PAL	
	PUP1OC.BPL	PUP1OC.PAL	
	PUP1B .BPL	PUP1B .PAL	
	PUPRTE.BPL	PUPRTE.PAL	
	PUPDG .BPL	PUPDG .PAL	
PUP Level 0	LEVEL0.HDR		
	PUP0B .BPL	PUP0B .PAL	
		PUP0A .MAC	Standalone or RT-11
		PUP0A .RSX	RSX-11M
Services	PUPLIB.HDR		
	ALLOC .BPL	ALLOC .PAL	
		QUEUE .MAC	Standalone or RT-11
		QUEUE .RSX	RSX-11M
		TIMER .MAC	
		CONTXT.MAC	
		OSA .MAC	Standalone
		OSA .RT	RT-11
		OSA .RSX	RSX-11M
I/O Drivers		ENDRV .RSX	RSX-11M
		ENTAB .RSX	RSX-11M

The RT-11 floppy consists of the following files:

DESCRIPTION	BCPL, etc.	MACRO11	COMMENTS
EFTP Run Module	EFTP .SAV		Executable Load Module
	EFTP .COM		LINK Command File
Applications Programs	DLIB .HDR		
	PUPTST.BPL	PUPTST.PAL	
	EFTPDR.BPL	EFTPDR.PAL	
		EFTPSB.MAC	RT-11 Subroutine for EFTPDR
EFTP Protocol	EFTP .HDR		
	EFTP .BPL	EFTP .PAL	
PUP Level 1	LEVEL1.HDR		
	PUP1I .BPL	PUP1I .PAL	
	PUP1OC.BPL	PUP1OC.PAL	
	PUP1B .BPL	PUP1B .PAL	
	PUPRTE.BPL	PUPRTE.PAL	
	PUPDG .BPL	PUPDG .PAL	
PUP Level 0	LEVEL0.HDR		
	PUP0B .BPL	PUP0B .PAL	
		PUP0A .MAC	
Services	PUPLIB.HDR		
	ALLOC .BPL	ALLOC .PAL	
		QUEUE .MAC	
		TIMER .MAC	
		CONTXT.MAC	
		OSA .RT	

The RSX-11M floppy consists of the following files:

DESCRIPTION	BCPL, etc.	MACRO11	COMMENTS
EFTP Run Module	EFTP .TSK		Executable Load Module
	EFTP .COM		LINK Command File
Applications Programs	DLIB .HDR		
	PUPTST.BPL	PUPTST.PAL	
EFTP Protocol	EFTP .HDR		
	EFTP .BPL	EFTP .PAL	
PUP Level 1	LEVEL1.HDR		
	PUP1I .BPL	PUP1I .PAL	
	PUP1OC.BPL	PUP1OC.PAL	
	PUP1B .BPL	PUP1B .PAL	
	PUPRTE.BPL	PUPRTE.PAL	
	PUPDG .BPL	PUPDG .PAL	
PUP Level 0	LEVEL0.HDR		
	PUP0B .BPL	PUP0B .PAL	
		PUP0A .RSX	
Services	PUPLIB.HDR		
	ALLOC .BPL	ALLOC .PAL	
		QUEUE .RSX	
		TIMER .MAC	
		CONTXT.MAC	
		OSA .RSX	
I/O Drivers	EN .TSK		Loadable Driver
	EN .COM		LINK Command File
		ENDRV .RSX	
		ENTAB .RSX	

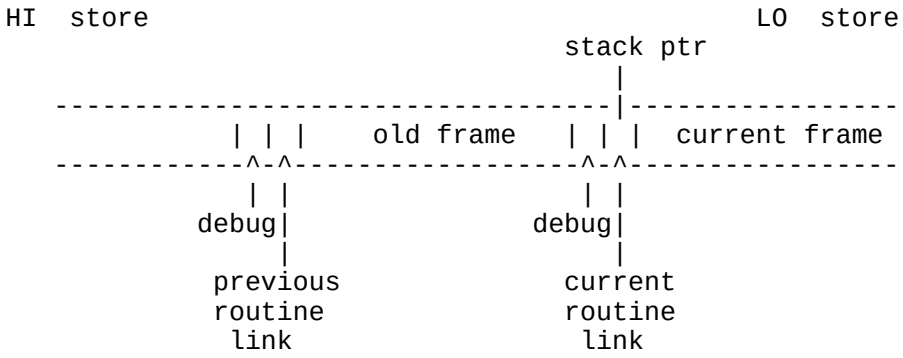
BCPL System Information

1. Register Allocation

The BCPL stack register is general register zero, the system stack register (the SP) and the program counter(the PC) are necesarily registers six and seven. On function entry registers one to four are used to pass the first four arguments, on function return register one holds any result. The only use of the system stack by the BCPL system is on function entry to hold temporarily the return link.

2. BCPL Stack Arrangement

As noted the runtime stack grows down store, and is allocated as shown.



The 'savespace-size' holding the static links of function entry is of size two, one of which is used for the code address linking, and hence also the previous frame size, and the other for debugging information or for use with the Intcode Interpreter.

Vectors are arranged to run up store, according to the BCPL definition. However the "vector" of arguments to a routine does not follow the definition - it grows down store!

3. Global Vector Linking

The Global Vector is known at link time as the Named Csect 'GLOBAL', linking of BCPL programs with this Csect is automatic. At the machine code level the conventional mechanism of accessing this Csect is by assigning a variable G to the address of global zero and offsetting from this address. Thus:

```
.CSECT GLOBAL           ;enter Csect Global
       G=.              ;G = address of global zero
       .=G+101.+101.    ;at global one hundred and one
       FUNC             ;insert the value FUNC
```

The variable G must only be assigned to once per assembler segment.

4. Function calling Sequence

```
        .
        .
JSR     PC,@G+N         ;calling through global N/2
       M                ;frame size M+2 bytes
        .
        .
```

5. Function Entry Sequence

```
SUB     @0(SP),R0       ;standard entry code
MOV     (SP)+,-(R0)     ;end of entry sequence
MOV     R0,R5           ;copy known args to the stack
MOV     R1,-(R5)        ;first arg on
MOV     R2,-(R5)        ;second on, etc up to four args
        .
        .               ;code of the routine
```

6. Function Exit Sequence

```
        .               ;code of the routine
        .
MOV     (R0)+,R5        ;result, if any, must be in R1
ADD     (R5)+,R0
JMP     (R5)            ;return completed
```

7. BCPL Addresses

At all times it must be remembered that BCPL manipulates addresses as integers. These integers are the addresses of consecutive sixteen bit fields in store and hence must be word addresses. To convert a BCPL address to a machine address one must thus convert to a byte address, which is most easily performed by a single left shift.

8. BCPL Strings

BCPL strings are vectors, considered as a sequence of bytes, the less significant half of each word preceeding the more significant, and these pairs being treated in their order of appearance in the vector. The value of the first byte of the string is the number of bytes in the string, excluding itself.