Release 1.0 of the UNIX Virtual Protocol Machine

P. F. Long
C. Mee, III

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

This memorandum describes the initial release of the Virtual Protocol Machine (VPM), a new UNIX† synchronous communication subsystem. The VPM is built around the KMC11, a small, high-speed microcomputer that connects to the UNIBUS of a PDP-11 or VAX-11/780. The VPM is a software construct for implementing link protocols on the KMC11 in a high-level language.

A compiler, *vpmc*, is provided to translate a high-level description of a protocol (protocol script) into the instruction set of the virtual machine. *Vpmc* supports C-like control-flow constructs, a modest subset of C-like statements and expressions, and a set of communication primitives that permit implementation of byte-oriented protocols such as BISYNC. (Primitives that support bit-oriented protocols such as HDLC have been defined and will be available in a later release of VPM.) An interpreter is provided that runs in the KMC11 and interprets the virtual machine instruction set. A UNIX driver, *vpm.c*, provides the interface between the user process's *open*, *close*, *read*, and *write* calls and the protocol script being executed by the interpreter. Besides providing the benefits of a high-level language implementation of protocols, the VPM approach permits portable protocol implementations.

The VPM software consists of five components:

1. *vpmc*: a UNIX compiler for the protocol description language.
2. VPM interpreter: the KMC11 program that controls the overall operation of the KMC11 and interprets the protocol script.
3. *vpm.c*: the UNIX driver that provides the interface to the VPM.
4. *vpmstart*: a UNIX command that copies a load module into the KMC11 and starts it.
5. *vpmtrace*: a UNIX command that prints an event trace for debugging while the protocol is running.

The procedures for installation and use of the VPM commands and the VPM driver are described; the pertinent manual entries are attached.

INTRODUCTION

The Virtual Protocol Machine (VPM) is a new UNIX synchronous communications subsystem built around the KMC11 microcomputer. The KMC11 is a small, high-speed, 8-bit microcomputer manufactured by DEC. It connects to the UNIBUS of a PDP-11 or VAX-11/780 and can become UNIBUS master, thus giving it direct memory-access capability (DMA), as well as the ability to control other UNIBUS devices. While other DEC communications devices provide direct-memory access, the KMC11 is the only one that is also fully programmable. Thus the KMC11 can provide most of the CPU power and some of the address space required to do data communications, thereby relieving the main CPU of these burdens. All is not roses, however:

---

† UNIX is a trademark of Bell Laboratories.

the KMC11 must be programmed in an unfamiliar and somewhat awkward assembly language. This, together with a requirement to provide several varieties of the BISYNC protocol and with a need to support, in the future, other link protocols such as HDLC, was the motivation for the development of the VPM.

The VPM is a software construct for implementing link protocols on the KMC11 using a high-level language. A compiler, *vpmc*, is provided to translate a high-level description of a protocol (*protocol script*) into the instruction set of the virtual machine. *Vpmc* uses a variant of Ratfor [1] as a front end to provide control-flow constructs such as *if-else*, *for*, *while*, *switch*, and *repeat-until*, as well as other benefits. *Vpmc* supports a modest subset of C-like statements and expressions, plus a set of communications primitives that permit succinct and easily-understood implementations of byte-oriented protocols such as BISYNC. These primitives allow the protocol scripts to reflect the essential structure of the protocol, while hiding details that arise from a particular hardware-software environment. (Primitives that support bit-oriented protocols such as HDLC have been defined and will be available in a later release of VPM.) An interpreter is provided that runs in the KMC11 and interprets the virtual machine instruction set. This program also controls the communications line and provides the interface to the UNIX host machine. The compiled protocol script is loaded with the interpreter into the KMC11. A UNIX driver, *vpm.c*, provides the interface between the user process's *open*, *close*, *read*, and *write* calls and the protocol script executed by the interpreter in the KMC11. (The UNIX *kmc* driver is used to implement this interface.) For a pictorial overview of VPM, see Figures 1 and 2.

Besides providing the benefits of a high-level language implementation of protocols, such as ease of programming and maintainability, the VPM approach permits portable protocol implementations. Portability can be achieved in several ways. First, because the interpreter and the compiled protocol script execute in the KMC11, they are the same regardless of the software running in the main CPU or, for that matter, regardless of the CPU itself. For example, the same interpreter and compiled protocol script can be used for UNIX/RT on a PDP-11 or for UNIX on a VAX-11. More general forms of portability are also possible. The instruction set of the virtual machine can be translated into almost any assembly language using one of the UNIX macro processors, such as *m4* [2]. This does *not* require that the assembler for the target machine have a macro expansion capability. (We may use this approach in the future to translate protocol scripts into KMC11 assembly language, thus gaining speed over the present virtual machine interpreter.) Another possibility for portability arises because Ratfor is used as a front-end; by limiting a protocol script to a statement and expression syntax acceptable to a Fortran compiler, the protocol is portable to machines that support Fortran in a suitable real-time environment. Finally, minor changes to a protocol script will yield a C implementation of the protocol. With any of these methods, the functions provided by the primitives (including the interfacing with communication devices and the execution environment) must be supplied by suitable library routines or system calls.

RELEASE 1.0

Release 1.0 of VPM is restricted to byte-oriented, half-duplex protocols such as BISYNC. A separate KMC11-B is required for each communications link. Each KMC11 running VPM must be equipped with a suitable DMC11 line unit. A DMC11-DA line unit is required for operation at speeds up to 19.2K bits/sec; a DMC11-FA or DMC11-MD is required for operation at speeds of 56K bits/sec. The modem control available on the DMC11-DA line unit permits both inward and outward dial-up communication.

---

[1] B. W. Kernighan, *RATFOR—A Preprocessor for a Rational Fortran*, Bell Laboratories.

[2] B. W. Kernighan, *The M4 Macro Processor*, Bell Laboratories.

This release of the VPM software is intended for use with UNIX Edition 1.1 or later. Operation with other versions of UNIX has not been tested. The VPM software consists of five components:

1. *vpmc*: UNIX compiler for the protocol description language.
2. VPM interpreter: the KMC11 program that controls the overall operation of the KMC11 and interprets the protocol script.
3. *vpm.c*: the UNIX driver that provides the interface to the VPM.
4. *vpmstart*: a UNIX command that copies a load module into the KMC11 and starts it.
5. *vpmtrace*: a UNIX command to print a debugging event trace.

Manual entries for *vpmc*(1C), *vpmstart*(1C), *vpmtrace*(1C), and *vpm*(4) are attached to this memorandum. A release tape containing the VPM software and manual entries is available from the authors. Installation procedures are described in the appendix to this memorandum.

Acknowledgements

The idea of using the KMC11 to interpret a protocol description was suggested by L. A. Wehr. He also offered useful suggestions and criticisms as the project implementation progressed.

APPENDIX

Hardware Installation and Switch Settings

The KMC11 microprocessor and DMC11 line unit must be installed in adjacent slots of a PDP-11 or VAX-11/780 backplane. The microprocessor and line unit are interconnected by a one-foot mylar cable. The line unit is connected to a suitable modem by a 25-foot modem cable. The device address and interrupt vector address switches on the KMC11 should be set for the selected addresses. All switches and jumpers on the DMC11 line unit should be in the normal configuration prescribed by the relevant DEC maintenance manual with one exception: the NO CRC switch (switch S2 in switch pack number 1) should be in the ON position. The purpose of this switch setting is to inhibit hardware CRC generation. Hardware CRC generation is not used with this release of the VPM software.

Installing the VPM Software on a UNIX System

In order to read the release tape, change to the directory into which the *vpm* software is to be read (say, *vpmdir*), then execute:

```
cpio −iBdv </dev/rmt0
```

The executable programs, shell procedures, manual entries, and examples of protocol scripts will be read into the current directory and the following six subdirectories will be created and loaded: *util*, *plsrc*, *ratsrc*, *bisyncb*, *drvsrc*, and *demo*. *Util* will contain some processors that may be needed: *awk*, *cpp*, *kas*, *kasb*, *kunb*, *kun*, and *m4*. (These processors are provided in case the versions on your system are not compatible with the release tape.) *Plsrc* will contain the source required to make *pl*, the main pass of *vpmc*. *Ratsrc* will contain the source required to make *vratfor*, a modified version of Ratfor used as a preprocessor for *pl*. *Bisyncb* will contain the VPM interpreter source for the the KMC11-B. *Drvsrc* will contain the source required to make the UNIX driver, *vpm.c*, and the command *vpmtrace*. *Demo* will contain demonstration programs and programs for checking the operations of the KMC11 and the VPM software.

Installation of the VPM Driver and Commands

To add the VPM driver to a UNIX Edition 1.1 system, do the following:

1. Add the following line to the file */etc/master*:

   ```
   vpm   0    36   6    vpm   0    0    15   1    5
   ```

2. Add the following two lines to the file */usr/src/uts/cf/cfigpa* (or its equivalent) for each VPM line to be added:

   ```
   vpm    0       0       0
   kmc11  vector  address priority
   ```

   If the KMC11s that are to be used have already been configured, the lines immediately above relating to KMC11s should *not* be added. See *config*(1M), *master*(5), and *Setting up UNIX* for more information.

3. To make a UNIX system that includes the VPM driver, copy *vpmmkdrv*, found in *vpmdir*, to */usr/src/uts/cp* or its equivalent. Check the *defines* at the beginning of *vpmmkdrv* to verify that the directories used are appropriate for your system. Then execute:

   ```
   vpmmkdrv sysname dfile
   ```

   where *sysname* is the name to be given to the system and *dfile* is the file modified in step 2 above. *Dfile* must be a simple file name (not a full path name).

4. To install the VPM commands, check the *defines* at the beginning of the shell procedure *vpmmkcmds* to verify that the directories used are appropriate for your system. Then execute:

   vpmmkcmds

5. Use *mknod*(1M) to create a node for each VPM line and each KMC11:

   /etc/mknod /dev/vpm? c *major minor*

   where *major* and *minor* are both octal; *major* is determined by *vpm*'s position in the *cdevsw* table and *minor* defines the KMC11 and VPM as follows: the two most significant bits denote the KMC11 number (0-3) and the three least significant bits denote the VPM number. For example, if KMC11s 2 and 3 are to be used for VPM, then the minor device numbers should be 0200 and 0301, respectively.

Compiling Protocol Scripts

The manual entry for *vpmc*(1C) describes the protocol description language. See also the examples of protocol scripts included on the release tape: *demo.r*, *demo.c*, *hasp.r*, and *mod40.r*.

When checking a protocol script for syntax errors, the **−c** option may be used.

Syntax errors detected by *ratfor* are noted as follows:

*****F ratfor:syntax error, line *n*, file *filen*

The line number *n* is in file *filen*.

Syntax errors detected by *pl* are noted as follows:

***** pl: syntax error, input line *n*.

To examine this line, a temporary file must be created as follows:

vpmc −m −r filen >temp

The temporary file can then be inspected using *ed*. The line number *n* refers to this file.

When all syntax errors have been eliminated, a KMC11 load module can be created by omitting the **−c** or **−r** options on the *vpmc* command.

Testing Protocols

When a load module suitable for testing has been made using *vpmc*, *vpmstart* may be used to load the file into the KMC11 and to start the interpreter. To view and record the trace records simultaneously execute:

vpmtrace | tee eventfile

A high-speed CRT terminal is best if you wish to get an impression of what is happening in real time. When a user program opens the VPM device, interpretation of the protocol script begins. Script interpretation ends if the VPM device is closed. Various error conditions can also terminate the script; they are described in *vpm*(4).

January 1981

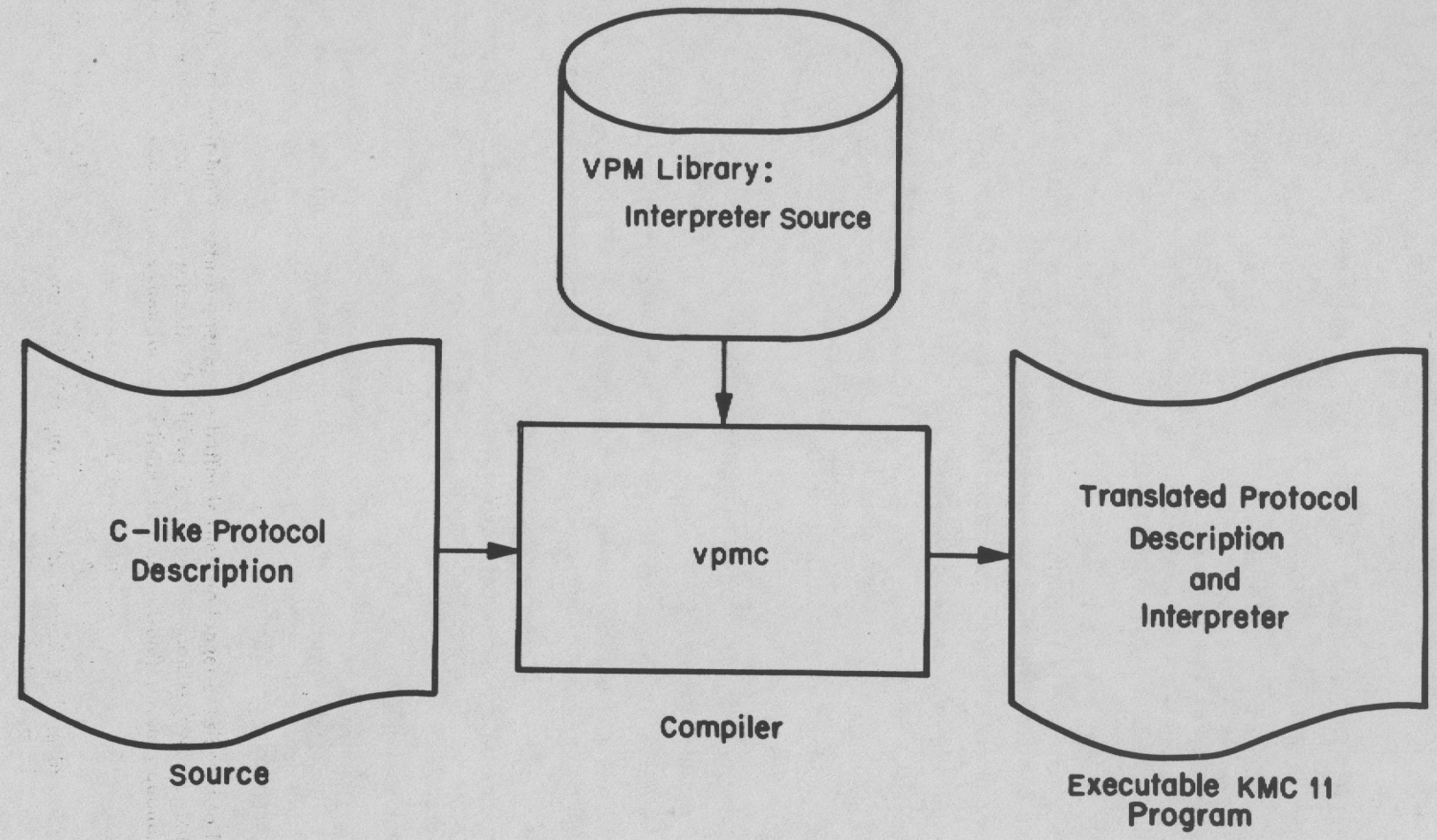
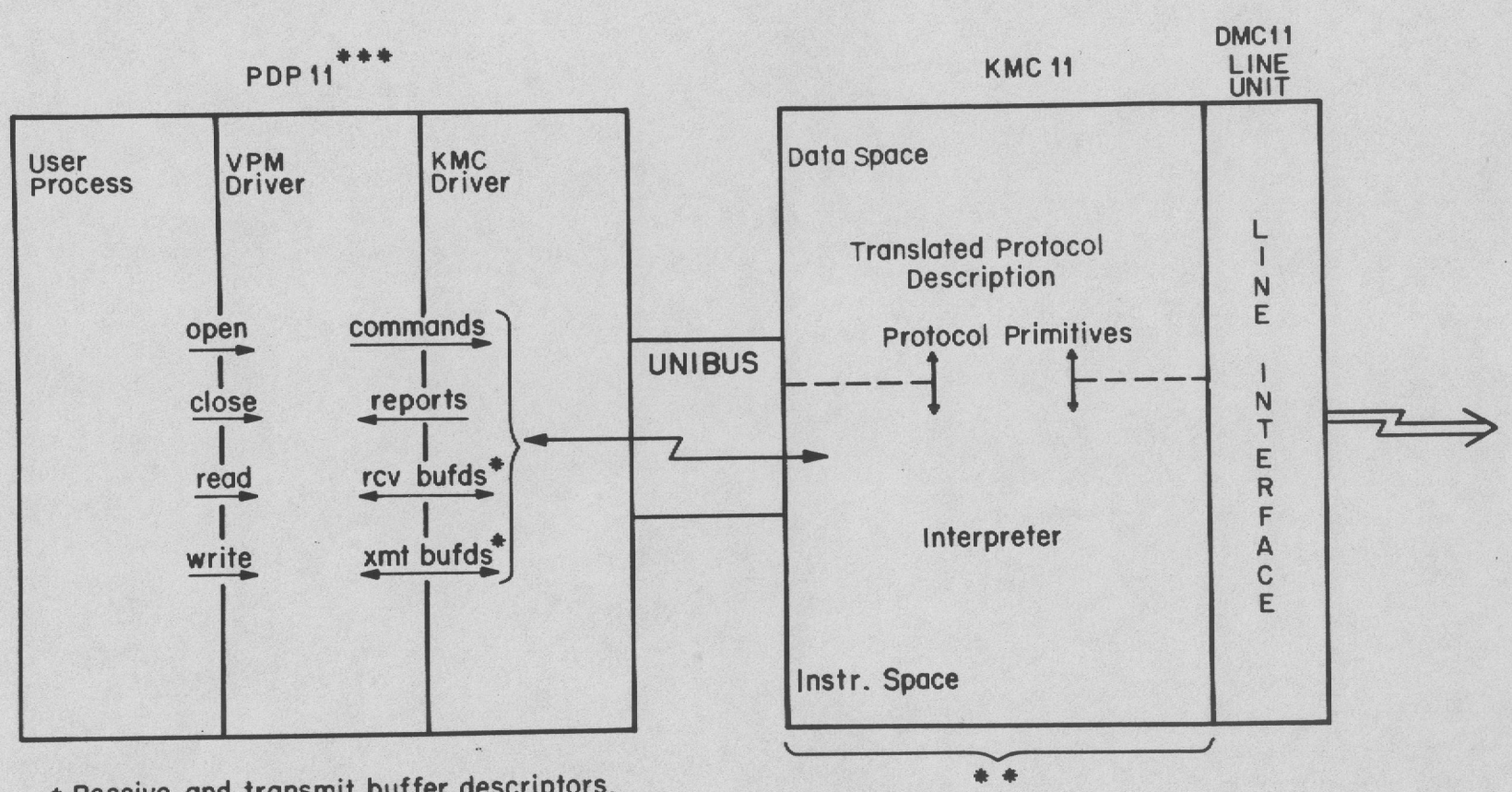


Figure 1
Protocol Compilation Process



* Receive and transmit buffer descriptors.

** Executable KMC11 program produced by *vpmc* & downloaded by *vpmstart*.

*** Release 2 will also run on the VAX.

Figure 2
VPM Components and Interfaces