What is a PDP-8?

Charles Lasner

 For the purpose of this archive, a PDP-8 is any machine, whether

actual, planned, or simulated, past, present or future, which

executes the instruction set of the classic DEC PDP-8 made starting

in 1965, or any newer design compatible with it. Products relating

to various other models with either frills or deficiencies will also

be included in the archive, with relevant differences noted.

 Various existent models will be enumerated, but the list is by no

means exhaustive. All known large-scale commercial models will be

covered, but various individuals have produced independent models on

a smaller basis.

 Some of the information is obscure, and comes from the personal

recollection of the author, Charles Lasner, of CLA Systems. All

information is presented as accurately as possible with no

representation as to copyright. This document is purely for

information purposes only.

Pre-1965 machines.

 Prior to the PDP-8, the first relatively small-scale 12-bit

computer was produced for laboratory work, the Laboratory Instrument

Computer, or LINC. It was designed using the then-standard DEC

System Modules. Many LINC designers were members of MIT's Lincoln

Laboratories and/or DEC employees at various times in their careers.

The LINC project was also responsible for LINCtape, which is a

block-replaceable disk device implemented physically serially on a

reel of tape. While DEC took no business interest in the LINC

project, they supported it by basically maintaining a "hands-off"

policy as various DEC facilities were tapped for its development.

 DEC's answer to the LINC was the PDP-5, which is a PDP-8 with

several design compromises. The largest incompatibility is the usage

of memory location 00000 as the program counter (PC). This was seen

as an economy move because register logic was much more expensive

than simple gating, and the memory plane was already a "given" in the

design. So, the PDP-8 design was bastardized in a subset where

location 00000 was not usable for ordinary storage purposes. Indeed,

storing into 00000 is equivalent to branching to the address value

stored there (-1). (This is the method FOCAL, 1969 uses to determine

operation on the PDP-5. Store one less than the address of an "I am

a PDP-5" routine into 00000 and you wind up in that routine.)

 There are a few other compromises, such as the inability to

freely OR together group one operate condition bits to produce

combinations such as CMA RAL, but this was primarily due to existent

design specification deficiencies, rather than any specific intention

to limit performance. In spite of this, the PDP-5 can execute most

non-conflicting combinations of these bits. This will produce

multiple operations within the same memory cycle, and in a proscribed

order. (The PDP-8 improved the flexibility later such that there

were very few restricted combinations; newer-still models improved

the situation so that there are virtually no restrictions.) Memory

speeds for both the LINC and PDP-5 are 8 microseconds, consistent

with available memory for 1962.

 While the LINC technically is not a PDP-8, it partially shares a

common history, and is also an offshoot of options on other models,

so its information should be included in a 12-bit DEC computer

archive.

 The PDP-5 is capable of running a small portion of the PDP-8

programs available, most notably a variant version of the DECtape

Library System for the type 555 DECtape only, and FOCAL, 1969. The

PDP-5 interrupt structure is to store the PC into 00001 and start

execution at 00002 whereas all PDP-8's store into 00000 and start

execution at 00001. FOCAL, 1969 self-modifies to accommodate the

hardware differences when it is first started up on a PDP-5. All

combinations of operate bits not implemented on the PDP-5 are avoided

in FOCAL, 1969.

 The PDP-5 supports a subset of the negative buss available on the

PDP-8. The PDP-5 buss connection is achieved using large

military-type connectors. DEC made available a mechanical buss

converter which makes most negative buss peripherals compatible with

the PDP-5, most notably the PDP-8 expansion box type 804, which

allows various lab peripherals (type 34D oscilloscope controller and

D-A converter, etc.) and a Calcomp plotter interface. Some of the

earliest 12-bit submissions to DECUS were for such a machine. Also,

PDP-5's started at 1K, although 4K was a popular option. Extended

memory is theoretically compatible with PDP-8 standards, but the buss

does not implement 3-cycle Direct Memory Access (DMA) nor a DMA

field. Extended memory software had to use the Data Field (DF)

during the DMA transfers, thus interrupts were illegal during

transfers since an interrupt clears the DF register. The later TC01

DECtape controller, which uses 3-cycle DMA, is not compatible with

the single-cycle type 555 DECtape, although the tape format is the

same. LINCtape is not compatible with DECtape format, but uses the

same media running in the opposite direction. Blank media can be

formatted for either system. Clearly DECtape is conceptually based

on LINCtape, but with improvements added. (Some LINCtape systems

include modifications to read/write DECtapes, with some restrictions,

etc.)

 LINCtape is 12-bit oriented, and can search bi-directionally, but

can only accomplish read/write transfers in the forward direction.

DECtape is 18-bit oriented, and can transfer while moving backwards

as well. Block lengths are chosen to be multiples of 18-bits for

compatibility with other DECtape systems, so the PDP-8 standard block

size is 129 12-bit words. (Some software treats the 129th word as a

nuisance word, while at least one system, the 4K Disk Monitor System,

actually takes advantage of it.)

The classic PDP-8 appears in 1965.

 This is the beginning of the line as far as the bulk of this

archive is concerned. The PDP-8 is the emergence of the older PDP-5

design with several improvements:

1) 3-cycle DMA on the buss.

2) a break field on the buss for extended memory transfers via DMA

 without restriction.

3) Interrupts at 00000; no special PC in memory kludge.

4) Much faster memory (1.5 microsecond cycle time for all operations

 other than I/O).

5) More flexibility when combining operate instructions. The only

 restricted combination is the use of the IAC operation with any

 of the four rotate instructions: RAR, RAL, RTR, RTL.

6) Newly designed Extended Arithmetic Element (EAE) option. The

 PDP-5 EAE was more like an I/O device, similar to the PDP-9/15.

 The PDP-8 EAE uses group three of the operate instructions and

 avoids external buss cycles.

7) 4K as standard minimum size and size increment.

8) Wire-wrapped backplane for lower cost and improved reliability

 and much smaller so-called "flip-chip" cards that are 6 times the

 logic density. PDP-5 modules were much larger, slower, and

 required soldered back-plane sockets assembled by hand.

 The PDP-8 revolutionized what has become today the notion of

"mini" computer, and indirectly the micro-computer or PC, which are

somewhat confused terms for what is actually the notion of a

"personally approachable" machine. For the first time, a computer

used by a sole individual within the relative privacy of a small

office, etc. was feasible. (The LINC was virtually always installed

in a large laboratory room whereas PDP-8's are just smaller enough to

wind up in private offices. Further, less than 100 LINC's were built

by diverse groups such as the US Army, whereas several thousand

PDP-8's were made by DEC.)

 Further, PDP-8's were the first machine offered with options that

"mattered." (Again, credit is due the LINC project for building in

peripherals such as LINCtape into the basic design. By NOT allowing

the tape to be an option, software was written to deal with it. The

PDP-8 was the last machine widely sold as a Central Processing Unit

(CPU) without a bundled mass-storage peripheral, and also the first

sold with an optional mass-storage peripheral as a popular option.)

The usual PDP-8 configuration was at least the minimum required to

use the software included in this archive, namely 4K (or more), a

mass-storage device (such as TC01 DECtape), and a terminal (usually a

model 33 or 35 Teletype).

 The first software to appear on this configuration was the PDP-8

version of the DECtape Library System, which actually has its roots

in the PDP-5 and LINC-8 versions. (All three versions are

incompatible.) These early software systems are basically binary

program savers, with a primitive filing system. Programs are saved,

and can then be called into memory by invoking the filename. A

primitive "index" facility is provided as the only other built-in

command. (Eventually there was an update to at least one version of

this system where the PAL III assembler could take input from a

"hidden" reserved text area instead of paper-tape, and output to a

"hidden" reserved binary area that could be loaded and saved. Source

files could also be edited and saved in a somewhat analogous manner,

but the system remained otherwise quite primitive.)

 DEC eventually released a system known as the Disk Monitor

System. It could be configured for use with DECtape or various disks

such as RF08 or DF32. It is known for being somewhat clumsy to use,

and used a structure not translatable to newer peripherals (since it

requires 129 words/block devices). (The PDP-8 DECtape format

standard is 129 words/block. The RF08 and DF32 disks are

word-oriented, and can be dealt with as 129 word blocks or any other

size up to 4096 words/block. Newer devices are addressed as

collections of blocks of 128 or even 256 words/block.)

 In 1967, an independent operating system appeared known as the

R-L monitor system or MS/8. It was designed primarily by Richard

Lary, who was also the principal architect for OS/8, RTS8, and more

recently DEC's large-scale mass-storage systems' internal protocol.

This system was refined and upgraded, finally being submitted to

DECUS by myself and others in 1970.

 In 1971, a replacement system was started using most of the

original concepts, which has gradually evolved into P?S/8, the

current version being 8Z as of 1992. The original concepts mostly

still apply: to run on the minimum system of a single DECtape or

other mass storage device, a 4K PDP-8 CPU, and a console device which

is at least a Teletype. All additional hardware options are used,

but not strictly necessary to run the minimal system.

 At the same time as P?S/8 was being developed, Richard Lary and

others started PS/8, later known as OS/8, for the same environment

plus the requirement of extended memory control and at least 8K of

memory. The programming discipline was identical in both systems:

the instruction set of the classic PDP-8 was to be used as the basis

for the operating system, no newer frills were allowed to be depended

on. (Device handlers present only on newer systems, such as LINCtape

on PDP-12, are allowed to use the additional CPU features guaranteed

to be present. Some newer devices, such as RX02, may not assume

this, since they may be connected to a PDP-8 CPU through an

appropriate buss converter (DW8E-N).) OS/8 development, as well as

P?S/8 development, persists to this day, although admittedly on a

"non-profit" basis by concerned individuals, some of whom are

contributors to the archive.

 An issue of compatibility must be raised here:

The PDP-8 instruction set is the basis for all of the "major"

software usually found on these machines, and more particularly the

combination rules of the PDP-8 operate instruction subset. Other

models, the PDP-5 and PDP-8/s, are incompatible with this level of

instruction set implementation; some newer models are actually

superior (slightly) to the PDP-8 in this area. Had the original

PDP-5 been released with the program counter kludge deleted, PDP-8

programming may have taken a slightly different direction, but due to

the relative popularity of the PDP-8, it has become the standard for

minimal instruction set requirement.

The PDP-8/s economy machine.

 DEC decided to offer a CPU-only machine for use in dedicated

applications, such as embedded systems within dedicated x-ray

spectrophotometers, etc., as offered by companies such as Picker

Instruments. These applications didn't require a fast machine, nor a

complete one, as the OEM probably would design custom hardware and

software to control his instruments as part of the overall instrument

package.

 Thus was born the PDP-8/s, a serial-logic implementation of the

PDP-8, but with certain restrictions. The original design was

designated PDP-10, and had a disk/drum memory, similar to some older

computers from the '50's. The idea was that every machine would have

32K and parity memory on a moving disk, and the overall physical size

would be much smaller than existing models, and also somewhat less

expensive. The tradeoff would be slight incompatibility and

significant loss of speed. Some PDP-8/s instructions take 78

microseconds to execute!

 Since the PDP-8/s is even more restrictive than the PDP-5

regarding operate combination, even less software runs on it.

Despite these significant limitations, the Disk Monitor System can

(slowly) run on the -8/s, using a specially slowed-down version of

the DF32. Extended memory is compatible, but not often implemented.

(The memory component actually implemented was a relatively small 4K

core plane on a quad card, the first of its kind: a physical

forerunner of the common Omnibus card!) FOCAL, 1969 self-modifies to

accommodate the sloth of the -8/s, so that paper-tape reader and

other time-out routines are properly calibrated.

 Another limitation of the PDP-8/s is the lack of a built-in

console device. To use the -8/s with a terminal requires an outboard

terminal interface such as a PT08 setup for device 03/04 usage.

(This was an accommodation to its primary application as a device

controller - CPU only in a small box.)

The LINC-8 appears.

 DEC took a more active role in the LINC project by releasing the

LINC-8. It was quickly realized that the LINC processor couldn't be

implemented as a peripheral device, and should be highly integrated

into the CPU design. The LINC-8 consists of a modified PDP-8

processor and memory interface, and an additional complete LINC

instruction-set CPU with high-speed DMA access to the memory buss.

When the LINC processor runs, the PDP-8 CPU is TOTALLY locked out.

There is a front panel DMA activity light on the LINC-8 (on the PDP-8

side, the same indicator as on the PDP-8). When the LINC processor

is running, this light glows brightly :-).

 Various portions of the LINC are simulated by a resident PDP-8

program (known as the PROGram OF OPeration, or PROGOFOP). This

program is read in from the beginning of the LINCtape (which is

actually a PDP-8 peripheral). At this point, all LINC operations are

available, including a complete (simulated) LINC console. Enough

hardware (and software) is in place at this point to allow all LINC

operations, including the use of dual switch registers, a front panel

instruction executive ("DO") key, a hardware load (boot) key,

EXAMINE, FILL, RESUME, and even FETCH and EXEC stops for LINC

programs. (A FETCH match causes a PDP-8 interrupt, thus stopping the

LINC CPU, and restarting the PROGOFOP interrupt processor, etc.)

 Various LINC-specific operating systems (such as LAP6) run from

this (partially simulated) environment. Most of these require only

4K of memory; a few support additional memory. (Some only support 8K

maximum.) There are some purely PDP-8-side LINC-8-only systems that

leave the LINC processor disabled such as the LINC-8 Library System,

CPS-4K, and CPS. (CPS requires 8K of memory).

 More recently, modifications have appeared to the LINC-8 as

specified and designed by Cooley Laboratories, the source of the CPS

systems. These hardware changes were augmented by Charles Lasner of

CLA Systems and David Soergel of Syracuse University. The current

version of the modification package allows P?S/8 to run in 4K, and

OS/8 to run in 8K. (It is possible to run alternate versions of

P?S/8 and OS/8 in 8K and 12K respectively on unmodified machines.)

The changes do not affect existing software ignorant of the changes.

 An additional feature of the modifications is the ability to

access (using special software written by Charles Lasner and others)

standard DECtapes and also Data General NOVA-based LINCtapes. NOVA

LINCtapes are created on drives and controllers for the NOVA which

are manufactured by Computer Operations Corp. of Beltsville, Md.

P?S/8 includes software to treat DECtapes on a LINC-8 as a block

device. Using the system's handlers, DECtapes and LINCtapes are

interchangable at the system handler call level. (Note that DECtape

conversion on LINCtape drives is not as reliable as using LINCtape

media, so this is not really recommended for general-purpose use;

P?S/8 design allows for this, but is only recommended for

block-oriented copying programs such as P?S/8 BLKCPY. This is a

utility to copy or compare blocks from different devices. Error

handling is very interactive with the user so that marginal tapes may

be reliably copied, etc.)

 Using Computer Operations tape drives and a custom interface,

Cooley Labs personnel developed a positive-buss version of the LINC-8

tape interface that allows CPS and other software to run on PDP-8's.

Only the PDP-8 <=> LINC register interface of the LINC-8 is

implemented, and only insofar as it pertains to tape operations.

PDP-8/i and PDP-8/l.

 The appearance of TTL (M-series) logic allowed DEC to scale down

the size of logic somewhat. The "FLIP-CHIP" boards of the PDP-8 and

LINC-8 are single-sided, and accommodate either several gates or at

most three flip-flops (less if additional gating is required on the

flip-flop). By making the boards double-sided, the new limit was

whatever required a total pinout of 36 connections per card.

Unfortunately, the limiting factor is often the connector, not the

"real-estate" of the board. Many small M-series cards are mostly

empty space, because the 36 pins are all used by as little as three

chips.

 In any case, logic density is somewhat higher in the TTL

implementation, so the PDP-8/i was able to be packaged in a slightly

smaller cabinet. Since large peripherals are often included in a

complete system, net improvement may boil down to how much empty

space is present in a standard cabinet or two. A minor mechanical

issue is that PDP-8 planes are often mounted high in the cabinet

leaving the bottom portion potentially underutilized. PDP-8/i is

implemented primarily on a vertical sliding metal backplane mounted

in the cabinet base. (There was also a relatively obscure standalone

unit with large support legs and no rack cabinet.)

 Since most peripherals at this time were negative buss (PDP-8

compatible), the buss of the PDP-8/i was implemented as negative as

well. Several years later, an ECO was issued for the 8/i to

implement positive buss drivers if desired. This was included as

part of a DEC refurbishing plan for 8/i's, etc.

 The PDP-8/i has a nominal cycle time identical to the PDP-8, but

in practice, it is often somewhat slower because of vendor problems

with the memory stacks. (The memory interface is asynchronous

allowing for this form of "detuning" if necessary.) Some individual

systems were known to run as slowly as 1.8 microseconds/cycle.

 The PDP-8/i has no restriction on operate combination, and even

some "interesting" quirks when incompatible options are chosen, such

as RAL and RAL simultaneously. (The net result is to zero both the

highest and lowest bits; this property is used to identify the 8/i in

programs such as FOCAL, 1969.) When EAE is implemented (a trivial

option to install), all PDP-8 EAE instructions are implemented,

including one additional operation undefined in the PDP-8 (load the

step counter, which is only needed in situations such as a

background/foreground system, where both contexts require EAE

independently). (It is possible to "fake" this operation by setting

up a table-driven Normalize (NMI) instruction that produces a

predictable value for the step counter.)

 PDP-8/i can accommodate 4K or 8K in the basic chassis, and up to

32K using external expansion memory modules known as MM-8/i. There

are also other vendors of 8/i-compatible memory such as Memorex,

which produced a 24K add-on box to achieve 32K more economically than

DEC's method. DEC eventually produced an add-on memory box using

PDP-8/e memory to accomplish pretty much the same thing.

 As an economy measure, DEC produced the PDP-8/l. This is

essentially the PDP-8/i design stripped down somewhat to produce a

lower-cost entry level system. PDP-8/i has a nominal cycle time of

1.6 microseconds, and most machines generally do so, because this

represents the best typical speed that the 8/i-type memory could

\*actually\* do. (It is rare to find a reliable PDP-8/i system running

as fast as 1.55 microseconds/cycle.)

 The PDP-8/l consists of a 4K memory and CPU, and no extended

memory. It has a curious unique feature called Memory Protect, which

write-locks the last page of memory, presumably to protect the

paper-tape-oriented RIM and BIN loaders sometimes stored there. The

buss is positive only, and was often used with newer devices such as

DF32D-P or TC08P.

 PDP-8/l is often upgraded to 8K by using a standard 8/l expander

box which provides all extended memory as well as an additional 4K of

8/i-style memory. There is also an obscure option which provides 8K

additional memory and memory controller, and also an extra set of

front panel switchs for the Instruction Field (IF) and Data Field

(DF). (The single IF and DF switches are built into the 8/l front

panel for use with a total of 8K memory, but with 12K memory, two

switches each are required, so the extra switches are on this

external box. This is somewhat unwieldy when performing manual

operations!) The standard expander box also provides some optional

plug-in options such as a clock, plotter interface, and D-A

converter, whereas the obscure box provides memory and control ONLY.

It is even legitimate to use both boxes. The memory box to provide a

total of 12K, and the (mostly empty) expander box to provide support

for a plotter or D-A converter, etc.

 PDP-8/l also prevents execution of EAE instructions, since this

option is not available. KERMIT-12 uses this property to distinguish

an 8/i from an 8/l. The instruction 7601 will clear the AC on an

8/i, but will NOT on the 8/l, because it is an EAE instruction to

clear the AC, and thus is inhibited. A curious aside is how FOCAL,

1969 attempts to distinguish the 8/i from the 8/l: the console

Teletype is used as a "clock" to determine if the machine is fast

enough. Unfortunately, if the machine is upgraded to a terminal

faster than 110 baud, then this test will always determine the

machine to be an 8/l regardless of which it is! Furthermore, since

many 8/i's are slower than nominal (virtually all are slower than

1.50, and many are slower than the 1.60 nominal of the 8/l, which

actually often is the case, since 8/l often had better grade memory),

this test is almost worthless. Fortunately, there is no serious

problem in FOCAL, 1969 other than the misidentification. (As long as

the serious issues of PDP-5 or PDP-8/s detection are served the

program will function; the other issues are merely frills. The newer

models such as PDP-8/e are reported as PDP-8.)

PDP-12.

 DEC re-entered the LINC market with the LINC-8/i or PDP-12 as it

became known. (The modules themselves often say LINC-8/i.) The same

basic scheme is used as how a PDP-8 became a LINC-8, except for one

difference: the CPU was designed to be a dual processor with a mode

bit, and all operations are in hardware. No simulator is used. (For

compatibility, there is a "trap simulator" mode to emulate the

limitations of the LINC-8, so software dependent on this feature can

be attempted.) There is only one integrated control panel, and all

of the features are real, not simulated as is the LINC-8's LINC

control panel.

 The PDP-12 can run all of the PDP-8 software, as long as the

system device can be a LINCtape instead of DECtape. (Optional disk

peripherals can be plugged in just as on a PDP-8/i or PDP-8/l, or

even some PDP-8/e peripherals on the DW8E, with some restrictions.)

There are also several PDP-12-specific operating systems, most

notably DIAL-MS.

 The PDP-12 supports the FPP-12 floating point processor option.

This option can be used on any model, but there is a special

connection to the PDP-12 backplane to allow "lockout" mode where the

PDP-8 only runs during interrupts (shades of the LINC-8!). This

speeds up the floating point operations relative to using the FPP-12

on the other models.

 The PDP-12 console represents the 12-bit programmer's zenith.

FETCH and EXEC stop are a standard feature. All registers are

displayed, and there is even a variable speed slow-run option. All

instructions can be executed from the built-in dual switch register.

 The buss is positive, and in later models includes a built-in DMA

multiplexer and hardware priority interrupt structure and hardware

pushdown stack (KF-12B backplane). All registers, including

operating mode, are stored during an interrupt by this hardware, (if

enabled) to speed real-time event handling.

 PDP-12 has virtually all required lab peripherals built-in

(digital inputs, sense switches, relays, D-A converters, A-D

converters for analog inputs and potentiometer knobs, programmable

real-time clock with internal clock, external clock, and 60 Hz

inputs). The D-A is connected to a 12" built-in oscilloscope display

with character and graphics output to the screen or an external scope

option.

 Memory expansion is either by any PDP-8/i method, or by a special

box designed by DEC later to use PDP-8/e memory boards. When this

option was introduced, a companion box was introduced for the

PDP-8/l. This was known as "the plot to blow up the PDP-8/l" and

included a box with two sets of two switches to allow the 8/l front

panel and box to provide the complete set of six IF and DF switches

for 32K manual access.

 PDP-12 cycle time is nominally 1.6 microseconds, but if faster

memory can be provided, there is a hardware hack: a certain module

can be plugged in backwards to make the machine run 20% faster at

your own peril. (It won't work unless the memory can keep up!)

 There is an option known as TC12-F which allows DECtape

conversion along similar lines to the LINC-8 modifications mentioned

above.

 The PDP-12 is housed in a special large 30" wide cabinet

physically compatible with DEC's standard 19" cabinets for expansion.

The standard configuration includes a programmer's table to be placed

just beneath the elaborate console, etc.

PDP-8/e, f, m.

 In late 1970, DEC announced the PDP-8/e. Ergonomically, this

machine is a follow-on to the minimalism of the PDP-8/l. A rotary

switch allows one of several choices of display, but there is only a

single display. The memory address is independently displayed in a

15-bit display, but there is no distinguishing the purpose of the

memory access, i.e., the program counter or effective address of

data, etc. An eventual option was offered to demultiplex the buss

into a complete display of all registers simultaneously, which was

called the MARS (Memory Address Register and Status) panel. Since it

was prohibitively expensive, few were actually produced.

 PDP-8/e implements a more sensible physical arrangement: much

larger boards are used which are nominally four wide and two high

relative to the original boards, but are actually more than eight

times as dense because of lack of handle and edge "overhead." There

are now four times as many pins to deal with a common buss, known as

Omnibus. All cards are on the Omnibus, and adjacent cards can have

"private" interconnects on the module tops as necessary. (Indeed

some cards have eight edges and you are warned not to plug the card

in upside down!) There is a standard two-card interconnect plug for

the purpose of linking two cards together on one of the four possible

top edges. A complete machine is now the size of a breadbox. There

is even a table-top version to avoid rack mounting if desired.

 By using newer TTL logic, and better buss components, the Omnibus

has better basic speed than the positive buss. Complete peripherals

are often implemented on a single card. The largest known peripheral

is four cards using several board-top interconnects.

 The CPU is somewhat faster, with one-cycle instruction speeds of

1.2 microseconds, with a 1.4 microsecond second cycle. DMA priority

is done on the buss to allow up to 13 DMA devices. Buss converters are

available to allow compatibility with all former devices on the

positive and negative busses. Buss logic slows the buss down only on

these "external" operations to avoid slowing down "internal"

operations.

 The CPU adds a few new frill instructions, including a new

byte-swap (BSW) instruction which defines the combination of bits

formerly used for the quirky RTL or RTR combination without the

rotate bit specified. (Rotate neither direction twice.) Interrupt

handling instructions are added to improve interrupt latency, and

there is an instruction to cause a power-clear on the buss to all

peripherals (CAF).

 The MQ register is standard, whether EAE is present or not. This

assists in simulating EAE if not installed. EAE is a large superset

of the PDP-8/i-type EAE of the 8/i or PDP-12, and includes

double-precision instructions, etc.

 Many new peripherals are available which are downsized versions

of older peripherals, as well as some newer ones unique to the

Omnibus such as the VT8E, etc.

 The basic PDP-8/e comes with an incandescent lamp front panel and

room for 20 slots in one Omnibus back plane. The power supply and

box can accommodate an additional 20-slot back plane as an option,

but interconnection will take up one slot in each for an effective

space of 38 slots. Using expansion cables, an additional box can be

added to add on another 20 or 40 more slots (minus the interconnect

overhead).

 There is an economy version of the 8/e known alternately as 8/f

or 8/m. There really is no difference between these two other than

marketing and the name on the plastic silk-screen front panel. The

8/f uses a compatible front panel with LED's instead of lamps, and

has a smaller power supply and box. Only one Omnibus back plane can

be accommodated in the basic box, but the 8/e expansion chassis can

be used if required.

 The PDP-8/m is identical to the PDP-8/f except that the front

panel is an option. If no front panel is installed, an "operator's

panel" is used instead. This is a minimal panel that can only start

the machine up with no switches or display available other than a run

indicator.

 Though never marketed this way, the 8/f-8/m LED front panel is

compatible with the 8/e box, but not vice-versa. The smaller power

supply of the 8/f box doesn't provide lamp power. There is a

power-fail restart option for these machines which must be strapped

for the analog consideration of which box it's plugged into.

(Earlier revisions of this card only function in 8/e boxes, since

there is no 8/f-8/m-specific circuit alternative.)

 A minor configuration consideration when adding on an expansion

box is that the so-called POWER-OK signal cannot be connected to

unlike boxes due to incompatibility. This means that one box cannot

realize that there is a shut-down condition initiated by the other

box. This only affects "canned systems" such as an unattended

application which uses RK8E/RK05's, etc. In actuality, the worst

thing that can happen is that the latest location referenced in a

running program will get corrupted, or that the RK05 may fail to

unload the disk at the appropriate time. Operator usage of the

machine would always include manual halt and shutdown of the disk

packs, so this isn't a consideration in this case. (It isn't likely

that an 8/f and 8/e expander chassis and RK05 would ever get used in

an unattended application anyway.)

 The PDP-8/e sold for far less money than its predecessors, and

was the first reasonable system for less than $10,000 including

useful peripherals. As such, it represents the first offering of a

practical "personal computer."

PDP-8/a

 This is the collective name for a group of semi-compatible

products. The original offering was a 10-slot box which supported

slow memory. A new Omnibus processor was introduced that is slower

than the PDP-8/e CPU board set. The new card includes its own

termination (at the wrong physical end of the buss!), and is 1.5

microseconds every cycle. EAE is not available, nor physical

expansion beyond the single box. All cards are (potentially) 50%

wider than the standard Omnibus cards. There are six buss feet as

well, but the sixth foot has no connections implemented, and the

fifth is only partially utilized. The CPU doesn't use the fifth

slot. The original memory options are a slowed-down semiconductor

RAM card and a ROM card with scratchpad RAM for a few locations of

memory. These memories require a memory stall option be supported by

the CPU. (The 8/a CPU board is known as the KK8A.) Eventually, the

PDP-8/e CPU (known as the KK8F when describing the board set) was

upgraded to support the same option.

 The PDP-8/a was originally slated for OEM applications continuing

the same philosophy started with the PDP-8/s. The difference here is

that this much machine is barely capable of being "useful" to a more

general audience.

 Eventually larger chassis boxes appeared. The first was a

12-slot box with a switching power supply and battery backup, and a

20-slot box with an additional power supply capable of supporting

core memory. (An obscure 12-slot box exists in limited production,

which supports core memory, but lacks the battery.) A front panel

with octal readout is optional, and requires a specific option board

capable of many other features be installed. (In the PDP-8/e, the

front panel itself plugs into the buss.)

 There are many memory options for PDP-8/a systems. An early 8K

core memory stack, MM-8AA, was failure prone, and was rated only for

use with the slow non-EAE KK8A CPU. Eventually, the MM-8AB was

introduced, which is 16K core memory. A 32K MOS dynamic memory card

was introduced which performs on-board refresh, thus occasionally

stalling the CPU. (This was the first PDP-8 with an "irregular"

cycle time.)

 Eventually, the notion of PDP-8/e and PDP-8/a was blurred as

well, because 8/a configurations were brought out containing 8/e

processors and terminator cards and even EAE. In essence, it is

totally reasonable to plug nearly anything from the 8/e into the 8/a

SOMEHOW. Somehow may include using an 8/e box as an expander for an

8/a, or it can be considered vice-versa.

 So, the new system concept was basically Omnibus configuration

according to the Omnibus Configuration Guide, an official DEC

publication. There are guidelines for using either the 8/e or 8/a

processor, multiple boxes of either kind, and even recommended

mounting positions in the cabinets to proscribe cable folding

procedures, etc.

 Software can only detect what type of CPU is present, and whether

EAE (if possible) is installed. There are several operate

instruction quirks in both CPU's, and there is a slight difference:

 The PDP-8/e CPU interprets the instruction: RTR RTL as "load what

page of memory I am currently running on +16 into the AC" whereas the

PDP-8/a CPU loads the updated current absolute address of the program

into the AC. By positioning the 7016 instruction at relative 15 of a

handler, it is possible to use the loading address information in a

handler for use on either system. (This has actually been done in a

practical handler!)

 Eventually, various memory options appeared for the hex wide

boxes. DEC supports a memory scheme for use with four of the 32K

dynamic memories (or equivalent in up to eight 16K core stacks) for a

maximum of 128K. DEC's system has a few advantages:

1) A 64K-maximum mode that is unconditionally compatible with all

 hardware.

2) A DMA arbiter to allow existing peripherals to address the full

 128K. (There are parallels to this problem and method of

 solution in the PDP-11 world.)

3) Close compatibility with existing software philosophy while

 introducing the extensions. (No additional instructions beyond

 initialization are required, just that the range of possible

 instructions is wider.

 A competitive system was introduced by CESI Inc. The CESI method

also supports DEC's memory scheme electrically, but is software

incompatible when extending the memory space. (All methods are

compatible up to 32K.)

 The CESI method requires additional instructions per usage beyond

the initialization to access the additional memory, and there is no

individualized arbiter for DMA extension. This prevents full

flexibility when used with existing peripherals. (CESI's reasoning

is that THEIR peripherals have an additional INCOMPATIBLE mode beyond

normal DEC standard, in which instance each peripheral does full DMA

addressing, thus their memory controllers need not bother. Problems

occur when you use mixed systems since the master DMA arbiter must

either be ALWAYS used, or NEVER used.)

 A useful feature of the CESI scheme is that it allows full access

to 128K with complete hardware compatibility. DEC's method requires

memory access be limited to a maximum of 64K to accomplish this.

Furthermore, the CESI card has a totally incompatible mode where only

CESI memory cards can be used. (Their memory cards can be strapped

for DEC compatibility or the CESI-exclusive mode as required.) In

this mode, the maximum memory size is quadrupled. Thus up to 256K

memory is addressable with full hardware compatibility, and up to

512K if the conflicting hardware is avoided. (Device code 30-37 is

in conflict with the full implementation mode of all of these

schemes. Avoiding the conflict thus cuts in half the total memory

space available in the configuration. The CESI 128K maximum mode has

no such conflict, while the DEC 128K mode drops to 64K maximum, and

the CESI 512K mode drops to 256K. Most peripherals can be strapped

to values other than 30-37, so this isn't too much of a problem.)

Both modes of CESI operation are software compatible with each other

up to 128K.

 A rework of the FPP-12 was introduced by DEC and sold as the

FPP-8/A. When the cards are bundled with a 12-slot backplane and

BC-80C cable, the package is known as the FPP-8/E, and is meant to be

added on as an expander box for any one-box 8/e-8/f-8/m system. This

version supports the original lockout mode, and also includes the

double-precision option as standard.

 My own personal system includes an 8/e processor in an 8/e box

with 40 slots cabled to a 20-slot 8/a box containing 32K contained in

two 16K core planes (MM-8AB). Both boxes have front panels which are

simultaneously functional. The buss terminator is in the first slot

of the 20-slot 8/a box while the processor (with EAE) is at the other

physical end, namely the front slots of the 40-slot 8/e box. I have

quad and hex peripherals, both DEC and CESI throughout the two boxes.

Any attempt to identify the system type beyond the 8/e-type CPU is

quite impossible :-).

PDP-8 micros appear.

 With the introduction of Intersil's IM6100, several PDP-8 systems

appeared in micro form. The CPU is contained on a single chip which

is PDP-8/e compatible. EAE is not available, and speed is about 2/5

of the normal 8/e. Intersil itself sold a packaged machine called

the Intercept. This design is capable of up to 32K using 4K CMOS

static RAMs. The only mass storage peripheral is the DSD-210 RX01

superset. (DSD-210 is RX01-compatible and can also format.) A

compatible version was sold by Pacific CyberMetrix as the PCM-12 and

PCM-12A (a slightly faster model). Eventually DEC placed a slower

6100 with 16K into a VT52 box to create the VT-78. Most had RX01's

but in theory could have RX02's (up to two pairs of drives). This

machine comes complete with serial and parallel interfaces for

various printers, etc. All peripherals are compatible with PDP-8/e

counterparts except for speed. The serial interfaces add a new

instruction to set the baud rate by software. Rotary switches on the

box are used to set the default baud rate values.

 The IM6100 has a few quirks in the operate instruction set which

allows for identification of the CPU type. This is the only model

where RAR RAL and RTR RTL are true NOP conditions, yet otherwise

maintaining 8/e compatibility. Further there is a total

INCOMPATIBILITY with the 8/e:

The normal addressing scheme of the -8 is that bit[4] determines

where the effective address is located: on the current memory page if

set, and on page zero of memory if clear. This allows all memory

locations to access page zero locations. In addition, if the page

zero location is in the range 0010-0017, and the instruction is

indirect, then the contents of the particular location 0010-0017

addressed is first auto-incremented before being used as the

effective address pointer. This is useful for accessing tables, etc.

On the 6100, the above is USUALLY true, but NOT if the instruction

itself is located on page zero. In this case, the PDP-8 would ignore

the page bit. The IM6100 uses the page bit differently in this

unique case. Thus the instruction 1410 is always TAD I 10, and

auto-indexes, but on page zero 1610 is equivalent only in PDP-8's.

The IM6100 creates a "feature" in that 1610 uses location 0010 as an

ordinary pointer and won't auto-increment it!

 This quirk was found by the 8/e diagnostic known as "Random TAD

Exerciser" which creates TAD instructions at random placed in random

addresses, including on page zero. The instruction is executed and

also emulated. Results are compared yielding an error message should

the above case occur on a 6100-based system.

 There is no practical software implication of this "problem,"

since usage of this case beyond the diagnostics and CPU-type checking

programs is virtually unknown. All assemblers produce the

non-problematic cases so a programmer has to go out of the way to

produce octal coding of the problem situation.

 There exists an interesting extension to the PDP-8 architecture

first implemented on the 6100. Beyond the normal instructions and

interrupt structure of the PDP-8 there is a "higher" level of

interrupt known as the Control Panel (CP) interrupt. This interrupt

is serviced even if the machine is not running! Like ordinary

interrupts, the program counter is saved, and an interrupt routine is

entered, but these are not located in any part of the 32K memory

space. Instead, there is an "alternate" space with its own program

counter called Control Panel memory. Location 0000 of CP memory is

where the program counter is saved, and 7777 of CP memory is where

the interrupt routine starts. CP memory is itself non-interruptible,

and is used to service the source of CP, not regular interrupts.

Interrupt service exit is via the normal ION; JMP I 0 sequence except

that the former run state is restored (running or halted) and control

returns to the "normal" or main memory. It is possible to create

minimal interfaces to the machine to cause instructions to

deliberately trap, so that they may emulate more complex operations,

etc. The ability to access main memory as well as CP memory can

easily be implemented as well.

 The original purpose of CP memory was to help implement a front

panel in software. The Intercept uses a variable clock to interrupt

the machine at some rate (typically 30 Hz) for the purpose of

scanning a simulated front panel. Interfaces exist for operation

switches such as ADDRESS LOAD, EXAMINE, etc., to simulate the "real"

PDP-8/e panel. The PCM-12 is quite similar. DEC's VT-78 uses the

ability to CP-interrupt on HLT to allow a register dump on the

screen, but DEC didn't implement any ability to restart the main

memory program.

 When any of these machines is first turned on, a CP interrupt

condition is forced, which can be used to initiate either a front

panel operation or a floppy bootstrap (automatically in the case of

the VT-78).

Second generation micro-PDP-8.

 Some of the problems of the 6100 were addressed and improved in

the 6120. This is a faster implementation of a design similar to the

6100. Several improvements were added:

1) 32K memory control is built-in.

2) CP memory is now 32K.

3) Instructions were added directly to force the machine to run when

 exiting to main memory. This overcomes the HLT problem in the

 VT-78 with ease.

4) Speed roughly equivalent to 8/a-8/e is achievable. Some

 instructions are actually marginally faster than their 8/e

 execution times; most are only slightly slower.

5) EAE is still not implemented, but enough hooks were added to

 possibly add it on externally.

6) Dual push-down stacks and manipulation instructions were added.

7) Power-on status can be read to help software initialize the

 machine in ways appropriate to true power-on only.

8) Instructions were added to cause panel traps directly.

9) Instructions were added to help sense the cause of the CP

 interrupts (power-on, HLT, panel trap request, or device

 interrupt request).

 In addition, an "official" interface chip was added, to ease

design effort when implementing practical systems.

 This interface chip is the source of the most serious blow to

PDP-8 software, and we haven't completely recovered from it yet. The

problem is that this interface design is NEARLY but NOT EXACTLY

compatible with the PDP-8 console interface design. It is different

in ways that are not completely overcome in the present software. It

is hoped that future software will restore the original functionality

of PDP-8 software in a manor totally compatible with all members of

the PDP-8 family. P?S/8 has already been upgraded to solve most of

these problems.

 The first machine to appear based on the 6120 was the DECmate I.

This machine supports one or two pairs of (possibly mechanically

modified) RX02 and optional RL02's. The machine simulates device

03/04 to be a VT-100, but is slightly incompatible with PDP-8's 03/04

interface to a real VT-100. Because the keyboard is a simulated

"local" device, it is not possible to lose portions of escape

sequences, which is an improvement over the use of "real" VT-100's.

DEC attempted to make OS/8 compatible with the DECmate I and released

OS/78 V4, but the software performs only a subset of original OS/8

functionality. Execution (with the limitations caused by the

imposition of the DECmate I) is identical on the 8/e or better. The

8/e is the minimum system that can run this system due to the

presence of many instructions violating earlier machines'

limitations.

 The next machine is the DECmate II. While just as incompatible

with the PDP-8 as the DECmate I, the device 04 emulation is smoother

and faster. Keyboard emulation is totally "compatibly incompatible"

with the earlier DECmate I, except that the keyboard is the standard

VT-220 LK-201 keyboard. The running DECmate II system acquires part

of the CP program from the boot device in dedicated tracks known as

"slushware." Slushware resides on either an RX50 5.25" disk or on a

hard disk in a dedicated volume (which is called "FIRMWARE"). Disk

configurations start from an RX50 pair and optionally include:

1) an additional RX50 pair.

2) one or two external RX01 or RX02 pairs on an RX78 option board.

 Programming the RX01/02 is NOT quite compatible with RX01/02 of

 previous systems. (P?S/8 runs on all of these systems in spite

 of this :-).)

3) an RD51 disk controller with nominally a 5, 10, or 20 Megabyte

 hard disk. Other larger sizes work "unofficially."

The hard disk precludes the RX01/02 options, and there is no place

internally to mount both the extra RX50 and hard disk, but the

software could support this configuration.

 Additional options include a graphics board and an external color

monitor. Provided software can cause the whole system to emulate a

VT-240 or VT-241.

 Additional processor options include a Z80 with 64K, and a board

with the Z80 and an 8086 and 256K or 512K. (The 512K board is

rumored to support 1.0 Meg.)

 Standard features include a USART comm port, serial printer port,

100 Hz real-time clock, etc.

 Internally, the device 03/04 interface is trapped and emulated

using an external keyboard interfaced to CP memory via CP interrupts

only, and an internal DMA video chip that uses CP RAM as a video

buffer. Various character generator RAM and ROM options exist to

implement DEC's VT-220 character sets. Several interfaces exist in

support of this emulation including a bi-directional serial UART

interface to the keyboard, and a dedicated video interface, as well

as a device used only to trap the device 03/04 Input/Output Transfer

instructions (IOT's) themselves.

 The problem with all DECmates relative to the console handling

has its roots in a chip actually designed for the 6100 (and

indirectly in the RX8E). All designs before the RX8E always had a

classic form of instruction set regarding event sensing. An event

was either a level or edge used to trigger a flip-flop to hold the

event. The event flip-flop's output was gated onto the interrupt

request buss through an AND gate. The enabling side of the AND gate

is the output of another flip-flop which contains the interrupt

enable for the event itself. An instruction (traditionally a 6xx1

IOT where xx is the device code) was provided to skip on the event

set, and another instruction (typically 6xx2) was provided to clear

the event. Sometimes a side effect of the clear operation was to

clear the AC. Often an additional instruction (typically 6xx5) was

provided to load the interrupt-enable flip-flop as well. (This part

of the interface design isn't relevant to the problem because

non-interrupt software is affected. Further, earlier interfaces to

device 03/04 had no interrupt enable; it was permanently enabled.

The PDP-8/e implements interrupt enable for device 03/04 using 6035

as load interrupt enable for both sides of the device pair per the

value of AC[11]. On some early LPT: interfaces, 6xx5 enables, and

6xx7 disables interrupt enable.)

 6xx4 could be designated as some form of data read operation of

information associated with the event flag. The earliest buss

specification defines this as an OR transfer. Since the early buss

actually micro-programs IOTs by the bits set, 6xx6 would first

execute 6xx2 followed by 6xx4 in the same I/O cycle. Thus, if 6xx2

was defined to clear the AC, then 6xx6 becomes an AC load instead of

the 6xx4 OR transfer. This is the original definition of the

keyboard interface and was faithfully implemented in all PDP-8

designs from the PDP-5 through the VT-78. (No additional frill

instructions exist before the PDP-8/e implementation.) On machines

where a Teletype was possible, an additional effect of 6032 or 6036

was to cause the reader to release a single frame of paper-tape if

present. This leads to an eventual character available

indistinguishable from a user-typed event. (To separate the

functions, the PDP-8/e added 6030 to clear the flag and NOT advance

the reader; little software exists to actually take advantage of this

feature, as paper-tape was gradually phased out.)

 A key issue here is the separation of functions: the skip IOT

always skips if the event is true. The clear IOT is the only way to

clear the event. Much software was written to exploit PRECISELY this

collection of features. (I think hardware engineers actually believe

that the only console software ever written is the trivial examples

given in Introduction to Programming :-).)

 When the RX8E came along, it was clear that a lot of IOTs were

needed to handle the device, which has three event flags: DONE,

TRANSFER, and ERROR. Using traditional implementation design would

require three device codes. An alternative could have been to

implement a status register to read all conditions, but this was

rejected. Instead, an economy of instructions was introduced. Each

operation was tested with one instruction, a skip on event flag,

clear event flag in one operation. This requires some additional

software, but saves the hardware designer's MANAGER some grief :-(.

 Event flag initialization now requires some form of NOP after the

skip on, clear IOT since it might skip, even though the purpose is to

definitely clear, and the skip part is a nuisance condition. Since

the RX8E is used in custom-designed disk handlers only, compatibility

with previous designs isn't really necessary, so the design was

approved. (Looking back on the history of the RX "fiasco" it is

clear that this is the least of the RX family's problems :-).)

 Eventually the 6100 was introduced, and soon thereafter a chip

appeared called Program Interface Element (PIE). The PIE chip

contains four event flags, and has so many instructions for data that

two complete adjacent (even/odd) device codes are needed. Four of

these 16 instructions are skip on, clear event flag IOTs just like

the RX8E. Again, the purpose of the chip is for custom interfacing,

so compatibility isn't an issue. It should be noted that neither

Intersil, PCM, nor DEC ever used the PIE chip in their 6100 designs,

because it was important back then to maintain software

compatibility, so all of the major interfaces were implemented with

small-scale logic.

 The PIE chip design was part of the engineer's model for

producing the next-generation interface chip, the 6121. This chip

has some notable features: each chip is actually five devices in one,

and the chip is externally initialized by software to recognize what

devices it actually responds to. However there is a major software

problem: each of these interfaces is IDENTICAL, and slightly

incompatible with the traditional interface :-(. Without additional

logic, the 6121 implements the following instruction set:

6xx0 Set the flag.

6xx1 Skip on, clear the flag.

6xx2 Clear the AC only.

6xx3 Undefined.

6xx4 Read or write. Input transfers may OR or load.

6xx5 Load interrupt enable for the flag per AC[11].

6xx6 Read or write as in 6xx4. OR ability may differ from 6xx4.

6xx7 Undefined.

Using this set of instructions, it is only possible to implement the

trivial examples of device 03 usage in Introduction to Programming,

not the ACTUAL coding used in \*REAL\* code in the various operating

systems :-( :-(.

 There are several levels of incompatibility. Device 03 and

device 04 must now SEPARATELY interrupt enable, whereas previously

6035 served both interfaces. 6045 was defined on the 8/e as skip on

interrupt from either device to help minimize polling overhead in

interrupt handlers.

 6030 was defined in the 8/e to clear the input flag. In the 6121

it SETS the flag! (6040 is actually compatible, because that should

set the output flag :-).)

 The main problem is that 6031 skips on, AND CLEARS the input

flag. This destroys the functionality of all existing software which

depends on noticing the input flag being set, and in the event of ^C

being the character struck, leaving the flag set for handling

elsewhere, often by programming swapped in. The swapped in

programming attempts to find out by executing 6031 again, and

expecting another skip, which fails to occur here.

 The DECmate II implementation of device 03 is such that all odd

IOTs are actual, and all even IOTs trap and are emulated in CP memory

by the slushware routines. When the PDP-8 program should be informed

of an available character, the slushware routines execute 6030 to set

the flag. The PDP-8 program notices the 6031 skipping, and is

expected to execute 6036, which traps to CP memory, which emulates

loading the AC with the latest character. Should 6031 skip, and 6036

not be executed, 6031 will NEVER skip again, even if more characters

are typed in on the \*real\* keyboard, which is actually device 11, and

causes CP interrupts. 6034 is traps and ORs into the AC a replay of

the last character transferred with 6036.

 This method is useful for transmitting long escape sequences for

the special function keys, because it is impossible to lose a

character in real-time, but it imposes the constraint on the PDP-8

program to NEVER fail to couple 6036 to 6031. (There are several

known bugs in OS/278, especially ODT, traceable to this problem.)

 Several techniques used in the past are also incompatible with

this implementation: the effective use of 6034 to OR in the latest

character (used especially to check for ^C being pressed) is now

impossible, since 6034 merely presents a repeat of the previous

character 6036 has already transferred, not the latest character

available without disturbing its flag. (The original idea was that

6032 cleared the flag and AC, 6034 ORed the data into the now clear

AC to effect the load. 6034 alone merely ORed in the new data

without disturbing the flag.)

 The OS/278 attempt to repair the damage caused by this

incompatibility is to merely do 6036 when 6031 skips, and not attempt

to convey the information about ^C to the rest of the system upon its

detection. This leads to an emasculation of the original OS/8

functionality.

 The P?S/8 solution to the problem is to define a soft "^C-hit"

bit. Detecting programs will execute 6031 and 6036, and then set the

bit accordingly. The keyboard monitor first looks at the soft bit

for ^C detection leading to abort action. Failing this, 6031 is

tested. If it skips, and 6036 reads ^C, then the abort action is

taken. This allows PDP-8-only programs (such as DECtape formatters,

etc.) to be compatible without modifications. A further advantage is

that other devices (graphics terminals, etc.) can simulate ^C abort

exit by setting the bit as well.

DECmate III models.

 The DECmate III is a down-sized DECmate II for most purposes.

Only a single RX50 pair is allowed, and no hard disk. The graphics

board and console video interface can share a single color or mono

monitor (unlike the DECmate II which always requires the mono

monitor).

 The printer interface baud rate cannot be changed from the fixed

rate of 4800, but is otherwise compatible. The communications option

is compatibly upgraded to include support for an optional 300/1200

baud DEC "Scholar" compatible internal modem. Certain programming

constraints must be applied when accessing the comm chip that makes

certain existing OS/278 programs incompatible with the DM III, but

this is fixable. All video and real-time interfaces are compatible.

 The graphics option is mostly compatible with the DM II graphics

option, and the APU option is essentially compatible with the Z80

APU option of the DM II. It is rumored that DEC created (but never

marketed) an XPU (80x86 version) version for the DM III as well.

 The DECmate III+ was the last model introduced. It comes with a

built-in RD51 interface to a 20 Megabyte ST-225 hard disk, but only

one floppy. The diskette drive is actually an RX33 drive, but no

attempt is made to use the high-density or double-sided capabilities.

(Since the drive runs at 360 RPM, transfers are 20% faster.) Overall

dimensions and case are identical to the DECmate III model.

 The built-in modem was dropped; all other features and options

remain the same. It is possible to produce a non-standard machine

with two floppies, and examination of the CPU board reveals the

likelihood of the modem interface being added merely by installing a

few components in areas reserved for them. The RD51 interface is

implemented on a separate board suggesting it could have been an

option. It is entirely possible that DECmate III systems could have

been produced using the DECmate III+ board to phase out the older

DECmate III boards.

Non-DEC PDP-8's.

 Besides the Intersil and Harris micro-PDP-8's, there have been

some other attempts at compatible machines over the years.

 CESI introduced a single-board machine called SBC-8 which is

based on the 6120, but uses compatible interfaces and an external

console terminal. An SCSI buss allows many disk options.

 There have been many small groups producing 6100-based machines

with various peripherals on them. Intersil itself included an

Omnibus subset (no DMA) application note with the machine. Some of

these are based on various development projects known as the 6100

Sampler Kit and the Intercept Jr. These were comparable to hobbyist

micros of the same era (1974-1976), and other manufacturer's

development systems such as for Motorola's 6800, etc.

 Fabritek once introduced a machine called the MP-12 which was

PDP-8 compatible. Little is known about it.

 Digital Computer Controls introduced several models called

DCC-112 and DCC-112H. At least one of them was positive buss

compatible, and a newer model was to be Omnibus compatible.

Eventually, DCC was sued by Data General for producing a machine

called DCC-116 which was too close to being a clone of a NOVA, and

the company was absorbed into Data General who dropped support of the

12-bit line.

 In the late '60s, Hungary demonstrated a machine that was clearly

a PDP-8/l. It is not known if it was made in Hungary or actually

shipped there from DEC in the USA. Photographs show a Hungarian

silkscreened front panel calling the machine "TPA." The rest of the

machine appears to be DEC standard. The only accompanying

information is the fact that the machine wasn't IBM-compatible :-).

A dual system was even alleged, but this was merely two identical

boxes stacked on each other. It is doubtful that any "parallel

processing" existed between them :-).

last update: 30-Sep-1992

[end of file]