- **Device Data Area Table**
  - Used by the I/O Interface to locate the driver's data area for the driver's OPEN, CLOSE, READ, WRITE, and CNTRL routines.

4.2.1 Driver Address Table

When a task makes an I/O Interface call, the executive must locate the driver associated with the specified device (major number) and operation (i.e. READ). It does so via a **Driver Address Table** provided by the user. The physical address of the table and the number of devices are specified to the executive via configuration parameters.

The **Driver Address Table** for a system with N devices can be described by the following declarations:

```
struct drvaddr drvatab[N];

struct drvaddr
{
int (*init_driver)();
int (*open_driver)();
int (*close_driver)();
int (*read_driver)();
int (*write_driver)();
int (*cntrl_driver)();
int resvd1;
int resvd2;
}
```

As shown, the **Driver Address Table** is an array of N structures, one for each device. Each structure contains eight entries. The first six entries contain pointers to functions (routines) within the driver associated with the device. The last two entries are reserved for future use.

4.2.2 Device Data Area Table

Many, if not most, devices need a data area where the device driver can store information specific to the device. Although a statically allocated area can be used, it is usually more convenient to dynamically allocate this area when the device is initialized. The I/O Interface contains services to support such dynamic allocation.

The **Device Data Area Table** is supplied and maintained by the I/O Interface. The table contains one long word entry for each device in the system. The entry is used to maintain the address of the data area for the device.

The device driver's INIT routine is responsible for allocating the device's data area and returning its address to the I/O Interface. This memory can come from any source - static data, a region, or a partition. On exit, the INIT routine must return the address of the data area to the I/O Interface. The I/O Interface saves this address in the **Device Data Area Table**. Whenever a device driver routine (other than INIT) is called, the I/O Interface passes the data area address to the driver.

4.3 Device Initialisation

During system initialisation, the executive automatically calls the driver's INIT routine for each device. They are called sequentially, beginning with device 0 and ending with the last device in the system.

Since drivers can only be called by tasks, the executive calls the driver's INIT routine on behalf of a system initialisation task, defined by configuration parameters. The *mode* of the system initialisation task (also a configuration parameter) is used as the *mode* while the executive calls the INIT routines of the drivers. If the driver's INIT routine makes a RTEID call which blocks, control is passed to an idle task provided by the executive until an interrupt unblocks the driver.

Although the driver's INIT routine is always called at system startup, it may also be called by a task, either to re-initialise a driver or when a new device driver is dynamically loaded.

4.4 Parameter Passing

All directives except *de_init* require a user provided parameter block. The format and content of the parameter block depends on and is determined entirely by the particular driver and device it controls. Its function is to pass input parameters to the driver.

In a system with an MMU, the address of the parameter block is a logical address. The I/O Interface will convert it to a physical address before passing it to the driver. Within the parameter block, addresses may be either logical or physical, as defined by the driver. The I/O Interface does not examine or translate any fields within the parameter block.

4.5 I/O Interface in C Language

The I/O Interface may be called in the C language as follows:

Function	Parameters
de_init	(dev)
de_open	(dev, argp, &rval)
de_close	(dev, argp, &rval)
de_read	(dev, argp, &rval)
de_write	(dev, argp, &rval)
de_cntrl	(dev, argp, &rval)

dev is a 32-bit device number formatted as follows:

bits 31-16 = major device number
bits 15-0 = minor device number

argp is a pointer to a parameter block which contains device and operation specific parameters. The format and contents of the block is determined by the driver.

rval is an output parameter in which READ, WRITE and CNTRL routines may return information about the call.

4.6 I/O Interface in Assembly Language

The I/O Interface may be called by loading parameters into specific CPU registers and executing a TRAP instruction. The following assembly language interface is used:

INPUT

D0.W = function number as follows:
1 = INIT
2 = OPEN
3 = CLOSE
4 = READ
5 = WRITE
6 = CNTRL
7 = RESVD1
8 = RESVD2

D1.L = Device number (major and minor)
A0.L = Pointer to parameter block (except INIT)

OUTPUT

D0.L = Error code - 0 indicates successful return
D1.L = Return value from OPEN, CLOSE, READ, WRITE and CNTRL
A1.L = Address of device data area (INIT only)

4.7 Driver Interface in Assembly Language

The I/O Interface calls the user provided driver using the following assembly language convention:

INPUT

D0.L = tid
D1.L = Device number (major and minor)
A0.L = Physical address of parameter block (except INIT)
A1.L = Physical address of device data area (except INIT)

OUTPUT

D0.L = Error code - 0 indicates successful return
D1.L = Return value from OPEN, CLOSE, READ, WRITE and CNTRL
A1.L = Address of device data area (INIT only)

4.8 Error Handling

There are a number of errors which can occur during a driver call. In general, there are two types:

1. Errors detected by the I/O Interface.
2. Errors detected and returned by the driver.

All I/O Interface generated errors are detected prior to calling the driver. In these cases, the I/O supervisor loads register D0 with an error code and returns to the caller without ever passing control to the driver. To distinguish between I/O Interface errors and driver errors, error codes below 10000H (16-bit values) are reserved for use by the I/O Interface. Below is a list of the errors which are detected by the I/O Interface:

Illegal Function Code
Illegal Major Device Number
Illegal to call driver from ISR
Illegal parameter block address (MMU version only)

Drivers should always return error codes which are greater than 10000H (non-zero in the upper 16-bits).

Error codes returned from the driver's INIT routine are ignored by the executive. If a driver's INIT encounters a fatal error during system startup, the *k_fatal* directive may be used.

4.9 I/O Interface Routines in C Language

The I/O Interface routines as called in the C language are described in the following pages.

4.9.1 INIT

NAME

de_init – "Initialise a Device Driver"

SYNOPSIS

```
uint de_init ( dev )

uint dev;    /* 32-bit device number */
```

DESCRIPTION

The INIT routine will be called during system initialization. The function of INIT is to setup the hardware as necessary and to initialise the driver dependent variables. If the driver needs to allocate a data area for its use, it would do so in the INIT routine. The address of this data area is saved in the Device Data Area Table by the I/O Interface.

RETURN VALUE

If the call succeeds, then 0 is returned.

If the call was not successful, an error code is returned.

ERROR CONDITIONS

To be defined.

NOTES

Not callable from ISR.