

Introduction

OS-9 System Call Descriptions

You use system calls to communicate between the OS-9 operating system and assembly language level programs. There are three general categories of system calls:

- User-state
- I/O
- System-state

All system calls have a mnemonic name for easy reference. User and system state functions start with F\$. I/O related functions begin with I\$. The mnemonic names are defined in the relocatable library file `usr.l` or `sys.l`. You should link these files with your programs.

The OS-9 I/O system calls are simpler to use than in many other operating systems. This is because the calling program does not have to allocate and set up file control blocks, sector buffers, etc. Instead, OS-9 returns a path number word when a file/device is opened. You can use this path number in subsequent I/O requests to identify the file/device until the path is closed. OS-9 internally allocates and maintains its own data structures, you never have to deal with them.

System state system calls are privileged and can only execute while OS-9 is in system state (when it is processing another service request, executing a file manager, device driver, etc.). System state functions are included in this manual primarily for the benefit of those programmers who are writing device drivers and other system-level applications. For a full description of system state and its uses, refer to Chapter 2 of the ***OS-9 Technical Overview***.

System calls are performed by loading the MPU registers with the appropriate parameters and executing a Trap #0 instruction, immediately followed by a constant word (the request code). Function results (if any) are returned in the MPU registers after OS-9 has processed the service request. All system calls use a standard convention for reporting errors; if an error occurred, the carry bit of the condition code register is set and register d1.w contains an appropriate error code, permitting a BCS or BCC instruction immediately following the system call to branch on error/no error.

Here is an example system call for the Close service request:

```
MOVE.W Pathnum (a6),d0
TRAP #0
DC.W I$Close
BCS.S Error
```

Using the assembler's OS9 directive simplifies the call:

```
MOVE.W Pathnum (a6),d0
OS9 I$Close
BCS.S Error
```

Some system calls generate errors themselves; these are listed as **POSSIBLE ERRORS**. If the returned error code does not match any of the given possible errors, then it was probably returned by another system call made by the main call.

The **SEE ALSO** listing for each service request shows related service requests and/or chapters that may yield more information about the request.

In the system call descriptions which follow, registers not explicitly specified as input or output parameters are not altered. Strings passed as parameters are normally terminated by a null byte.

End of Introduction
