Published: 08/08/68

Identification

I/O System Status Reporting: Basic Format and Implementation.
J. F. Ossanna.

Purpose

This section describes the format of the status returned by the I/O System (IOS) and the methods of status reporting.

Status Reporting

The status string described herein is the one returned as the last parameter of every IOS outer call.

Ordinarily the IOS is used with the workspace synchronization mode having the default value synchronous (see BF.1.04). In this case, the IOS performs no delayed status reporting to the caller. That is, upon initial return from a call to the IOS, the status string reflects only what is known about the transaction at that time. Usually, this is sufficient. Typically, _read_ transactions are physically complete, if no error has occurred, and _write_ transactions are logically complete (see BF.1.04). A user that needs to be assured of physical completion of _write_ calls can set the write synchronization mode to synchronous, causing the IOS not to return until such completion.

Delayed status reporting is provided when the workspace synchronization mode has been set to asynchronous (see BF.1.04). During each _read_/_write_ call the IOS computes a pointer to the user's status string for use in future status updating. Later, upon receiving control, the IOS stores directly in the user's status strings updated versions of the status for all outstanding transactions. The _nelemt_ (number of elements transmitted) parameter of these calls is similarly updated. It is the user's responsibility to retain the storage for _status_ (and _nelemt_) until no further updating will occur.

Delayed Fatal Error Indication

It is always possible that a fatal error may occur in attempting to physically complete a transaction which was successfully logically completed. Certain Device Interface Modules (DIMs), upon detection of such a fatal error, abort all outstanding transactions following the one aborted by the error, and enter an error state in which the next call is rejected. The status of the rejected call will indicate that a fatal error occurred in a previous transaction; and the act of rejection terminates the error state, permitting future calls. If the next call following the rejected call is an _upstate_ call, the _status_ of the _upstate_ call will contain a status string indicating the status of the earlier call that suffered the error (the transaction index will also be that of the earlier call).

If an _upstate_ call is received while in the error state, the status corresponding to the earlier call is returned and the error state is terminated.

Status String Format

The status string is passed as a 72-bit bit string, and must be alligned on a word boundary. It is divided into parts as shown in Table 1. The status string is intended to be examined using a mismatched declaration. The first 36 bits are actually an integer, and should be referenced, for example, as follows.

```
dcl code fixed based(p);
if addr(status) -> code then ...;
```

If no error has occured, _code_ is zero. The _code_ is divided into major and minor codes according to the relation

```
p -> code = a*1000000 + b;
```

where _a_ indicates major codes and _b_ minor codes. A list of IOS-wide codes is given in Table 2.

Many of the status indicators are of interest primarily when asynchronous workspace is used. Some users will be interested in the end-of-data indicator. Other bits are of interest in special cases.

The unique transaction index is returned for all calls except the _upstate_ call.

The Upstate Call

The _upstate_ call mentioned earlier has the following form.

```
call upstate(ioname, status);
```

In addition to the above described function of delayed reporting of fatal errors, the call provides a method of giving control to the IOS to permit internal updating and possibly additional work to be physically initiated. When the workspace synchronization mode is asynchronous, the call permits the IOS to update the user's status strings.

Table 1.

I/O system status string format.

Bit	Purpose
1-36	36 bits of fatal and/or advisory status (an integer).
37-54	18 status indicators (see below).
37	successful logical initiation (see Section BF.1.04).
38	successful logical completion (see Section BF.1.04).
39	successful physical initiation (see Section BF.1.04).
40	successful physical completion (see Section BF.1.04).
41	transaction terminated (no more status change).
42	unassigned.
43	unassigned.
44	unassigned.
45	unassigned.
46	end-of-data indicator.
47	unassigned.
48	unassigned.
49	unassigned.
50	sync control; sync events active.
51	device absent from channel.
52	ioname detached.
53	abort was due to quit condition.
54	transaction aborted.
55-72	unique transaction index.

Table 2.

Codes returned in status bits 1-36 as an integer.
Code = A*1000000 + B.

A (Major codes)

0	advisory status (important but not necessarily fatal).
1	error codes returned by the I/O Switch.
2	errors reflected by DIMs.
3-5	errors detected by DIMs.
6-8	undefined.
9	fatal error in previous transaction.

B (Minor codes)

A,	B	reason
0,	-	to be defined.
1,	1	invalid argument count.
	2	ioname not found.
	3	ioname already attached and active.
	4	typename not found.
	5	ioname not active.
	6	missing entry in outer module.
2,	10*rcode+1	reflected GIM errors.
	10*fscode+2	reflected file system or SMM errors.
	3	reflected IPC errors.

3 and up, to be defined.