To: Distribution

From: Noel I. Morris

Date: December 11, 1974

Subject: Reporting status from peripheral devices

Overview

A new table-driven mechanism is being provided to analyze the status returned by the IOM for peripheral devices and to report on error conditions found in the status. This mechanism is designed primarily for those devices which are driven through the I/O Interfacer, but it is not limited to such devices. A status interpretation table for each different type of peripheral device is required. A primitive is provided which is given the name of the peripheral device, the IOM status, and a pointer to the status interpretation table. This primitive will report all error conditions found in the status via calls to com_err_. The primitive will also return 18 bits of information to its caller to aid in determining what action should be taken on the basis of the status.

Analyzing termination status

When termination status indicating an error is returned by the IOM, the following call can be made to analyze the status:

```
dcl analyze_device_stat_ entry (char (*), ptr, bit (72)
        aligned, bit (18) aligned);

call analyze_device_stat_ (devname, tablep, iom_status,
        flags);
```

devname — is the name of the device for which status has been received. It is used as the first argument passed to com_err_. (Input)

tablep — is a pointer to the status interpretation table for the device. See section below for a description of the status interpretation table format. (Input)

---

iom_status is the 72 bits of status returned by the IOM. (Input)

flags are 18 bits of information designed to aid the caller in deciding how to act on the status. Currently, two bits are defined: substr (flags, 1, 1) is used to indicate that the I/O operation for which status was returned should be re-executed unconditionally. substr (flags, 2, 1) will cause substr (flags, 1, 1) to be set if the initiation bit in the IOM status is on. That is, it will force an indication that the I/O operation for which status was returned should be re-executed only if it failed during initiation of the operation. (Output)

Note that analyze_device_stat_ should not be called with IOM status that does not indicate an error condition. An error bit is provided for such purposes by the I/O Interfacer. Other users should determine that one or more of the following fields in the IOM status bits are nonzero before calling analyze_device_stat_: power, major status, IOM central status, IOM channel status.

Analyzing system fault status

Certain serious IOM errors do not cause termination status to be stored. Instead, they cause an IOM system fault which returns IOM system fault status to the user. This condition is indicated by an IOM interrupt level of 1. In this case, a different call should be made to analyze the status information:

```
declare analyze_system_fault_ entry (char (*), bit (72)
        aligned);

call analyze_system_fault_ (devname, iom_status);
```

(Arguments are as above.)

Status Interpretation Table

The status interpretation tables for each device are coded in mexp. Each table consists of a header, and an array of substatuses for each major status which the device can generate. The format of the table header is as follows:

```
declare 1 stable (15) based (tablep) aligned,
        2 offset bit (18) unaligned,
        2 count fixed bin (17) unaligned;
```

stable — is an array which contains an element for each possible major status.

offset — is a relative pointer to the structure describing this major status and its possible substatuses. If offset is zero, the corresponding major status is not expected for this device.

count — is a count of the different substatuses expected for this major status.

Each major status and its associated substatuses are described by the following structure:

```
declare 1 sinfo based (sp) aligned,
          2 maj char (24),
          2 sub (count),
            3 control char (6),
            3 flags bit (18),
            3 desc char (32);
```

sinfo — is the structure.

maj — is a description of the major status.

sub — is an array of substatuses expected for this major status.

control — is 6 characters of information describing the substatus bit configuration. Only three characters are permitted in the control string: "0", "1", and "X". In order for a given substatus to match a table entry, each "0" or "1" in the control string must be matched by a corresponding "0" or "1" bit in the substatus. Bits corresponding to "X" characters are not compared.

flags — are 18 bits of information which can be used to determine what action should be taken for this status. See above for a description of the defined flag bits.

desc — is a description of the substatus.

Status Interpretation Table Generation Macros

Three macros are provided for the purposes of generating a status interpretation table for a device. The first of these is designed to generate the table header. The other two generate the major status and substatus information structures.

status_table devname,(control)

This macro sets up the table header and defines a segdef so that the table can be externally accessed symbolically. It also causes the symbols "backup" and "initiate" to be defined as 400000 octal and 200000 octal respectively. These are used in setting the flags.

devname — is the name of the device for which a status interpretation table is to be generated. The segment is given a name of "devname_status_table". The table header can be found at the segdef "devname_status_table".

control — is a string of 15 numbers seperated by commas. A zero indicates that the corresponding major status is not expected for this device. A nonzero number indicates that the corresponding major status is expected and information describing that major status will appear in the status interpretation table.

status_entry maj,(description)

This macro sets up the major status description.

maj — is the major status being described. Note that this number is decimal.

description — is a character string describing this major status.

substat_entry maj,control,flags,(description)

This macro sets up infomation for substatus.

maj — is the major status for which this substatus applies. Note that this number is decimal.

control — is 6 characters controlling the examination of the substatus as described above.

flags	are the 18 bits of action flags described above. This number is octal. It may also be described symbolically.
description	is a character string describing this substatus.

Appendix

The mexp macro definitions are given below for reference:

```
" Macro definitions for peripheral status table generation.
"          created 12/1/74 by Noel I. Morris


&macro     status_table

           name        &1_status_table

           segdef      &1_status_table


           bool        backup,400000        retry previous operation
           bool        initiate,200000      set backup flag if initiate

&1_status_table:

&(2        ife         &i,0
           zero        0,0
ifend
           ine         &i,0
           zero        m&x,lm&x
ifend
&)

&(2        ine         &i,0
           set         lm&x,0
ifend
&)

&end


&macro     status_entry

m&1:       aci         "&2",24

&end


&macro     substat_entry
           set         lm&1,lm&1+1
           aci         "&2",6
           vfd         o18/&3
           aci         "&4",32

&end
```

Example

The following is an example of a status interpretation table for the card reader:

```
" CRZ_STATUS_TABLE - Status Tables for the Card Reader.
"         coded 12/1/74 by Noel I. Morris

"  ************************************************************
"  *                                                          *
"  *                                                          *
"  * Copyright (c) 1972 by Massachusetts Institute of         *
"  * Technology and Honeywell Information Systems, Inc.       *
"  *                                                          *
"  *                                                          *
"  ************************************************************

        &include  status_table


        status_table        crz,(1,1,1,0,1,0,0,0,0,1,1,0,1,0,0)


        status_entry        1,(Device Busy)

        status_entry        2,(Device Attention)

        substat_entry       2,000000,initiate,(Device off line)
        substat_entry       2,XXX0X1,initiate,
                            (Hopper empty or Stacker full)
        substat_entry       2,XXX01X,initiate,(Manual Halt)
        substat_entry       2,0X10XX,initiate,(Feed Alert)
        substat_entry       2,1X10XX,initiate,(Sneak Feed)
        substat_entry       2,X1X0XX,backup,(Card jam)
        substat_entry       2,1X00XX,backup,(Read Alert)

        status_entry        3,(Device Data Alert)

        substat_entry       3,000001,backup,
                            (Transfer timing error)
        substat_entry       3,000X10,backup,(Validity check)
        substat_entry       3,0001X0,backup,(Dual read alert)
        substat_entry       3,001000,backup,(Feed without read)

        status_entry        5,(Command Reject)
```

```
substat_entry       5,0000X1,backup,(Invalid op code)
substat_entry       5,00001X,backup,
                    (Invalid device code)
substat_entry       5,000100,backup,(IDCW parity error)


status_entry        10,(MPC Attention)

substat_entry       10,000001,initiate,(IAI error)
substat_entry       10,000010,initiate,(DAI error)
substat_entry       10,000100,initiate,
                    (DA Transfer error)
substat_entry       10,001000,initiate,(Invalid Punch)


status_entry        11,(MPC Data Alert)

substat_entry       11,000001,backup,
                    (Transmission parity error)
substat_entry       11,000101,backup,(DAI error)


status_entry        13,(MPC Command Reject)

substat_entry       13,000001,backup,
                    (Illegal procedure)
substat_entry       13,000010,backup,
                    (Illegal logical channel)
substat_entry       13,001000,backup,(Device reserved)


end
```