Published: 07/27/67

Identification

The Device Signal Table Manager
J. H. Saltzer, M. J. Spier

Purpose

The Device Signal Table Manager is a hardcore ring procedure which serves two apparently (but not) unrelated functions:

1. Interface to interprocess communication for the hardcore ring. The device signal table contains, for each device attached to the system, storage space which is used for communicating signals arriving from the device to the device manager process.

2. Repository of device-index receiving-process id assignments. Entries are provided to establish such an assignment and to check whether a given assignment exists. (The assignment of device indexes to devices is a different problem, discussed in section BT.1.02.)

The Device Signal Table

The device signal table contains one entry for each device attached to the system and which may be under the control of some process. Such devices include, for example, typewriters, magnetic tape channels, magnetic tape drives, and the system clocks. Each such device has a device index, which is the index of the device in the device signal table. The device signal table contains four items of information for each device:

1. Identification number of the process to which this device is currently assigned. If this number is zero, the device is, by convention, unassigned. This item is used to check the validity of requests by a process to use the device, and to determine what process to wake up when the signals come from the device.

2. An event_id generated by set_dev_signal (see below) in response to a signal arriving from the device. Due to memory shortage (the Device Signal Table is wired down) this table item can accommodate only one event_id at a time. It is read and reset to zero by the receiving process' Wait Coordinator. Set_dev_signal stores the event_id only if this table item is reset to zero, any event_ids generated in response to subsequent device signals are ignored for as long as this item's value is non zero.

3. Event_count. This count is incremented by one whenever an event is signalled by the device. It is read and reset by the wait-coordinator together with the event_id.

Note: Items 2 and 3 are the device signal channel referred to in MSPM section BQ.6.03. This is a wired down event-count-mode channel dedicated to a specific I/O device. It is associated with and coupled to some receiving process' event channel for as long as the process has the authorization to use the device. The Wait Coordinator reads all the device signal channels associated with the receiving process and transcribes their contents into the corresponding event channels.

4. Route information. This is a bit string in which may be stored information needed to figure out the hardware route to the device. The format and interpretation of this information varies from device to device.

<u>The device signal table manager</u>

The device signal table manager contains two types of entries, <u>protection</u> entries and <u>signalling</u> entries. The protection entries are used by hardcore modules as part of their validation procedures.

Three protection entries are provided:

```
     call dstm$set_auth(dev_inx,prcs_id,ev_chn)
```

```
declare dev_inx fixed bin(17), prcs_id bit(36), ev_chn bit(70);
```

places the process identifier "prcs_id" in the device signal table as the process responsible for the device whose index is "dev_inx", then calls the Interprocess Group Event Channel Manager's entry ipgecm$link_dev_chn(ev_chn, dev_inx) (see MSPM section BQ.6.05) to couple the device signal channel (dev_inx) to the event channel identified by "ev_chn".

The entry set_auth is used, for example, by the GIOC Interface Module in response to a call from the Registry File Maintainer to assign a device to a process.

The second protection entry is used to disassociate an entry in the device signal table from a process.

```
     call dstm$reset_auth(dev_inx)
```

places a zero-value process_id into the entry whose index is "dev_inx", then performs a call to the Interprocess Group Event Channel Manager's entry ipgecm$unlink_dev_chn (dev_inx) which looks up the process' device-signal-channel-list in the Event Channel Table, finds the event channel associated with "dev_inx" and disassociates it.

The third protection entry is used to check the validity of a request to use the device

```
     f=dstm$check_auth(dev_inx,prcs_id)
```

The returned value of "f" is "1"b if "prcs_id" is currently assigned the device "dev_inx" as indicated by the device signal table entry for "dev_inx". Otherwise, the returned value of "f" is "0"b. This entry would be called, for example, by the GIOC Interface Module when it receives a request from some process to do I/O on a typewriter. Unless check_auth returns "1"b, the GIM would reject the request.

Note: The Basic File System calls the DSTM even though it is not a user of the Interprocess Communication Facility. It therefore does not specify an event channel name when it calls set_auth, thus circumventing the automatic coupling of the device signal channel to an event channel.

Signalling entries

Two entries are provided for the purpose of interprocess event signalling.

```
     call dstm$set_dev_signal(dev_inx)
```

increments the event count for "dev_inx" and generates and stores an event id in the "ev_id" location if that location is reset to zero. Set_dev_signal then calls "wakeup" in the process exchange for the process whose (non-zero value) id it finds in "dev_inx". This entry would be used, for example, by an interrupt handler which has decoded an interrupt only far enough to determine what device it came from. The interrupt handler would call set_dev_signal to increment the device's event count and to notify the process interested.

The second signalling entry is

```
     call dstm$read_dev_signal(dev_inx,ev_id,ev_count)
```

```
declare (dev_inx,ev_count) fixed bin(17), ev_id bit(70);
```

which checks to make sure that the calling process is the one to which the device signal channel -- pointed to by "dev_inx" -- is assigned. It then copies the device signal channel's contents into "ev_id" and "ev_count", resets the device signal channel and returns to its caller.

Entry read_dev_signal is the only entry of the device signal table procedure which is accessible to administrative ring procedures. It is intended to be called by the Wait Coordinator whenever a wakeup occurs and the process waking up is armed for signals arriving from hardcore devices.

The Wait Coordinator calls a ring 1 procedure named ecm$get_dev_signal and hands it as argument the device-signal-channel-list (see figure 1 in MSPM section BQ.6.06). Get_dev_signal calls read_dev_signal for each element (device-signal-channel) of the list, and on return from the device signal table manager enters the received event signal into that event channel.

Physical Route Information

The device signal table contains, for convenience, an extra item for each device, the route information. This bit string has a device dependent interpretation, but in general it is a handle which will lead ultimately to the hardware address of the device. For example, the GIOC Interface Module stores as route information the segment number of a block of data describing the hardware channel to the device. Two entries are provided, to set and to read the route information.

```
     call dstm$set_route(dev_inx,route)

declare dev_inx fixed bin(17), route bit(18);
```

will place the bit string "route" in the route information for device "dev".

```
     call dstm$get_route(dev_inx,route)
```

will obtain the route information for device "dev" and place it in the bit string "route".

The device signal table declaration

Following is the EPL declaration of the device signal table.

```
declare 1 dst(n) ctl(dst_ptr),
        2 prcs_id bit(36)          /* Process id */,
        2 ev_id bit(70)            /* Event id */,
        2 ev_count fixed bin(17)   /* Event count */,
        2 route bit(18)            /* Route information */;
```