Published: 12/09/68
(Supersedes: BQ.6.03, 07/24/67)

Identification

The Event Channel Table (ECT)
Michael J. Spier

Purpose

The function of the Interprocess Communication facility (IPC) is to "keep book" on all the event messages received by a process. A process knows that it is going to receive event messages of one or more "types" (or "nature" or "meaning") and so it gives each type of event a unique event channel name. Associated with each event channel name is an event channel which is a mail box for events of that specific kind.

The IPC sees to it, when one process (sending process) sends an event message concerning a given event channel name to some other process (receiving process), that at some point the event message gets transcribed into the receiving process' event channel where it can then be retrieved by it.

This section describes the event channel and the table in which it resides, named the event channel table (ECT).

The event channel

An event channel is a mailbox into which the IPC puts all event messages associated with one event channel name. As mentioned above, the definition of "event channel name" is rather vague. It applies to one or more event messages to which the process wishes to associate a common significance. The events associated with those event messages do not necessarily have to be observed in the same manner. For example, the process may have an event channel named "logout" into which messages will normally be put by the Shell, however the Device Manager may signal over the same event channel in case of a device-hangup, or the system-shutdown procedure in case of a system shutdown, or the accounting module in the case where the process might have run out of resource allotments. And yet, to the receiving process any of these event messages, regardless of its origin, would simply indicate that the process' logout procedure has to be invoked.

The event channel name, then, is a symbolic name associated by the process with a group of events; it is meaningful only within the receiving process. Event channel names are unique within a process, and uniquely identify their associated event channel.

An event channel is an entry in an event channel table; it is the head of a queue (threaded list) of event message entries which are also allocated within the same ECT.

Normally, when a process cannot proceed with its execution until some event has happened, it interrogates the associated event channel to see whether or not that event has already occured. If yes, it is satisfied and resumes its execution, else it calls the Traffic Controller's entry point block and blocks itself until an event message is received by it. An event channel which is explicitly interrogated in this manner is said to be an event-wait channel.

Another possibility exists, by which a process can set up an event channel and specify that whenever an event message is received over that channel a call should automatically be issued to some procedure known as the channel's associated procedure. Such an event channel is said to be an event-call channel. Briefly, event call channels allow a limited kind of "multiprogramming" to be done within a process.

An event channel entry in the ECT has the following structure:

```
dcl  1 event_channel based(p),
     2 type fixed,                    /*type=1*/
     2 thread,
       3 forward bit(18),             /*forward thread in
                                        event call list*/
       3 backward bit(18),            /*backward thread*/
     2 name fixed bin(71),            /*event channel name*/
     2 flags,                         /*various indicators*/
       3 ev_call bit(1),              /*0=event wait channel,
                                        1=event call channel*/
       3 inhibit bit(1),              /*1=channel inhibited.
                                        this flag is set by
                                        ipc$cutoff*/
       3 used bit(1),                 /*1=channel has been
                                        signalled over*/
     2 trailer bit(18),               /*pointer to event call
                                        trailer*/
     2 queue,                         /*channel's event message
                                        queue*/
       3 head bit(18),                /*head of queue*/
       3 tail bit(18),                /*tail of queue*/
     2 filler fixed;                  /*padding to make entry
                                        8 words*/
```

Following is an itemized description of the event channel structure:

type — to identify the various entries in the ECT. An event channel always has type=1.

thread — is used only if this is an event call channel, to thread it into the event call list: the list is double-threaded using forward and backward pointers.

name — this is the event channel's name by which it is retrieved in the ECT.

flags — event call this flag is set to "0"b for an event wait channel and to "1"b for an event call channel.

inhibit sometimes a user may wish to disregard arriving event messages (without, though, forgetting about them). When this flag is "1"b then the channel is considered to be "empty" for reading purposes. It has no effect on the reception of event messages.

used this flag is set to "1"b whenever an event message is appended to the channel; it is never reset to zero.

It is useful in determining (mainly for debugging purposes) whether or not an event channel has ever been signalled over.

trailer — An event call channel contains more information than does an event wait channel. When the ev_call flag is set to "1"b, this item points to an ECT entry known as event call trailer which contains the additional information.

queue — this item points to the queue of event messages which are appended to the event channel ("in the mailbox"). In order to preserve the sequence of events, messages are appended to the tail of the queue and removed off its head.

filler — An event channel entry, like all other ECT entries is allocated in the ECT's entry table, which is an array of fixed length items. Currently, the size of an ECT entry is 8 words long and all shorter entries are padded accordingly.

channel-1 and channel-2

Every event channel table contains two event channels named channel-1 and channel-2. These channels are created during ECT initialization time; the only difference between them and regular event channels is that their event channel names are not unique to a process. The event channel names of both channels are constant within a given ring and known to all processes. A process that knows its own ring n channel-1 name, knows every other process' ring n channel-1 name. Thus processes can communicate with one another over these two channels without first having to establish any basic interprocess communication.

These channels are very useful for "broadcast" type events where the signalling process knows about the receiving processes, without otherwise interacting with them. A typical user of these channels is the locker module, which needs to communicate with a process whose ID it found in a lock-word but about which it otherwise knows nothing.

The event call trailer

An event call trailer contains all the additional information required by an event call channel in order to be able to issue a call to its associated procedure.

An event call trailer has the following structure:

```
dcl  1 event_call_trailer based(p),
     2 type fixed,                      /*type=2*/
     2 thread,
       3 forward bit(18),               /*unused*/
       3 backward bit(18),              /*backpointer to event
                                          channel*/
     2 data_ptr pointer,                /*pointer to associated
                                          data*/
     2 procedure_ptr bit(18),           /*pointer to associated
                                          proc entry*/
     2 priority fixed,                  /*relative location in
                                          ev-call list*/
     2 filler(2) fixed;                 /*padding*/
```

type — an event call trailer always has type=2.

thread — forward is unused

backward backpointer to the event channel entry

data ptr every event call channel may be associated with some data. This association is established during channel creation time and remains static. This pointer may point to some data area, or to some arguments or possibly to an entry in some table, etc. It is passed as argument to the associated procedure to distinguish between more than one channel associated with a single procedure.

procedure ptr this item points towards an associated procedure entry in the ECT. More than one event channel may be associated with a single procedure which may, in turn, make free use of IPC. This may result in an error condition by which the procedure is invoked recursively by the IPC (because more than one event is pending in an associated event call channel). To insure against such recursion, every associated procedure has its separate entry in the ECT which is interrogated whenever an event call is to be made. In this way, recursive calls are detected and inhibited.

priority this number specifies the relative location of this channel's entry on the event-call list; it is assigned by the user in order to define relative interrogation priorities among his channels, as the event call list is scanned sequentially starting with the low-value priority numbers.

filler padding to make this entry 8 words long.

The associated procedure entry

The associated procedure entry has the following structure,

```
dcl  1 associated_procedure based(p),
       2 type fixed,                    /*type=3*/
       2 thread,                        /*associated procedure
                                          entry list*/
         3 forward bit(18),             /*forward thread*/
         3 backward bit(18),            /*backward thread*/
       2 pointer pointer,               /*pointer to procedure's
                                          entry*/
       2 inhibit fixed,                 /*1=procedure is
                                          inhibited*/
       2 count fixed,                   /*number of associated
                                          channels*/
       2 filler(2);                     /*padding*/
```

type — an associated procedure entry always has type=3.

thread — all associated procedure entries are threaded into the associated procedure list, which is double threaded.

pointer — this is a pointer to the procedure's entry point.

inhibit — whenever the procedure is invoked by IPC, this flag is set to 1. IPC does not issue an event call to an inhibited procedure.

count — this is a count of the event call channels which are currently associated with this procedure. When the count drops to zero then this entry is deleted.

filler — padding to make the entry 8 words long.

The event message

The event message contains all the information which is associated with an event. Whenever an event is signalled, an associated event message is appended to the event queue of the appropriate event channel.

Events are signalled by two modules, hcs_$wakeup and pxss$dst_wakeup; the first is used for signalling events which originated in non-hardcore procedures (user-events), the latter is invoked by the DIM or GIM to signal the completion of an I/O. This latter kind of event, named device signal is characterized by a relative lack of relevant control information. The following description points out the differences between user events and device signals.

An event message entry has the following structure:

```
dcl  1 event_message based(p),
       2 type fixed,                    /*type=4*/
       2 thread,                        /*the event queue thread*/
         3 forward bit(18),             /*forward thread*/
         3 backward bit(18),            /*backpointer to event
                                          channel*/
       2 channel fixed bin(71),         /*event channel name*/
       2 message fixed bin(71),         /*event message*/
       2 sender bit(36),                /*sending process' ID*/
       2 origin,                        /*origin of event*/
         3 device_signal bit(18),       /*0=user event,
                                          1=device signal*/
         3 ring bit(18);                /*ring of signalling
                                          procedure*/
```

note: the last two items are right adjusted (packed fixed bin(18)).

type — an event message entry always has type=4.

thread — the double thread of the event message queue.

channel — The event channel name to which this message is directed. Device signal messages contain a truncated event channel name (only the left-hand 20 bits).

message — a 72-bit message associated with this event. The sending process may put into this item whatever information it deems suitable. For device signal messages, this item contains a right-adjusted device index.

sender — The ID of the sending process. The device signal messages do have this item, however it is of no practical value (except perhaps for debugging).

origin — this item describes the origin of the event message. device signal when this item is non-zero (contains a right-adjusted 1), the message is a device signal. When this item is zero the message is a user-event. ring this item contains the right-adjusted ring number of the procedure that signalled the user-event.

Figure 1 shows the layout of the above-described ECT entries.

Figure 2 shows the relationship between the event-call channel entries, event trailers and the associated-procedure entry.

The ECT Lists

There are a number of threaded lists running through the ECT. They are as follows:

event call list — all event call channels are threaded into a list which is sequentially scanned for interrogation; event call channels are entered into this list at creation time; their relative position in the list is specified by their priority number.

associated procedure list — When an event call channel is declared it may be associated with a procedure which already has an entry in the ECT. Therefore the associated procedure list is maintained to allow easy scanning of all current associated procedure entries.

empty list whenever an ECT entry is deleted, if it is the very last item in the ECT then the ECT is shrunk by one entry. However if the deleted entry is imbedded within the ECT then it is put on the empty list.

The ECT

The ECT is declared to be an internal static array in procedure <ipc>. There is an ECT per active non-hardcore ring. The IPC maintains in segment <process_info> an array of 63 pointers to the ECT's of (potential) rings 1-63. This array is readable in all rings and allows a procedure in ring n to access the ECTs of rings n+1->63.

The ECT consists of three major parts,

1. The ECT header which contains a number of control variables as well as other useful information.

2. The entry table in which entries are allocated. This table is of variable size; it grows or shrinks as a result of entry allocation.

3. The ITT message transcription area which is the area following the entry table. It is always separated from the entry table by a one-entry buffer zone.

The ECT header contains the following items:

size current size (number of entries) of the entry table.

channel count current number of event channel entries.

cell count current number of non-channel entries.

empty count current number of entries on the empty list.

For debugging purposes, the following zero check is made by the IPC primitives:

size - channel_count - cell_count - empty_count = 0

free entry pointer relative pointer to the above-mentioned one-entry buffer between the entry table and the ITT transcription zone.

empty list relative pointer to the head of the empty list. When empty_count is zero, this item must also be zero.

event call list head of the event call list.

associated procedure list head of the associated procedure list

wait call priority this item determines whether event wait or event call channels should be interrogated first. A non-zero value signifies event-call priority. This item can be set to either value by means of special calls to the IPC.

mask calls this item, when non-zero, causes the IPC to refrain from interrogating its event call channels (event calls "masked"). Special calls to the IPC are provided for the setting of this flag.

itt queue this relative pointer points to the base of the itt transcription area.

next free itt this relative pointer points to the first unused entry in the ITT transcription area; procedure hcs_$block copies an event message into the transcription area indirectly through this pointer.

channel 1 pointer relative pointer to this ECT's channel-1 entry

channel 2 pointer relative pointer to this ECT's channel-2 entry

channel 1 name this ring's channel-1 name

channel 2 name this ring's channel-2 name

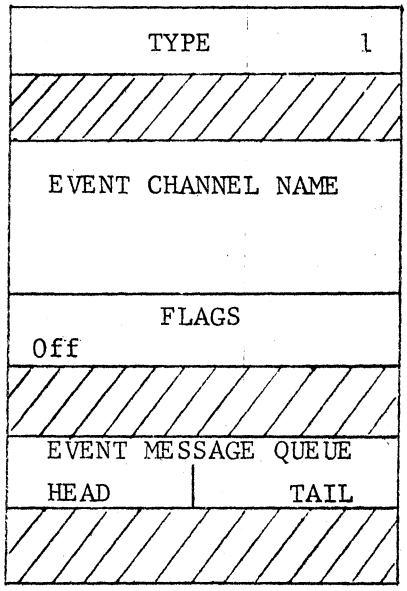
Special calls are provided to return channel-1 or channel-2 names to the user.

The entry table starts off with length zero. A call to allocate an entry first tries the empty list. If there are entries on it, it detaches one and returns a pointer to it to its caller. If the empty list is empty, then the entry-table is grown by one entry and the pointer to it returned to the caller. A call to delete an ECT entry first compares the pointer to it with free_entry_pointer; if the entry to be deleted is the last one in the entry table then the table is shrunk by one entry. Otherwise it is threaded onto the empty list. The entry table is thus guaranteed to remain packed at the base of the area in which the ECT resides. This increases the probability that the ECT could remain within a single page of core.

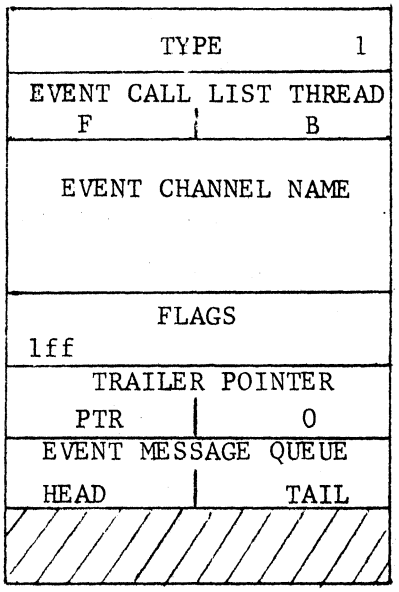
The area following the end of the entry table is reserved for ITT message transcription. When a process returns from the Traffic Controller's entry point block, it is given a list of zero or more event messages which are allocated in the Interprocess Transmission Table (ITT). The process copies these messages into the ITT transcription area of the ECTs to which these messages are directed.

Figure 3 illustrates the layout of the ECT.

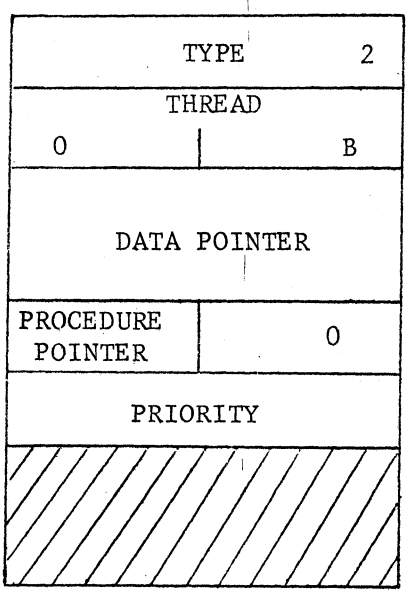
THE ECT ENTRIES



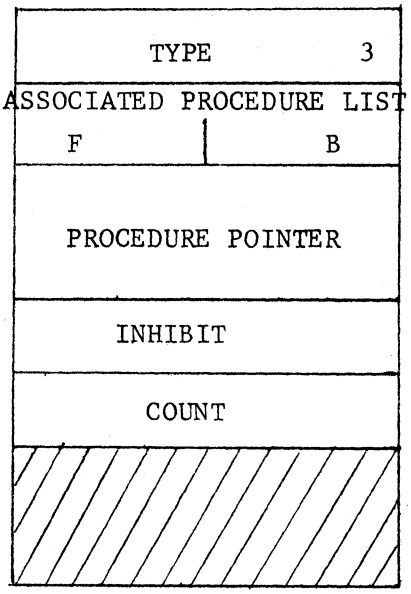
EVENT WAIT CHANNEL



EVENT CALL CHANNEL

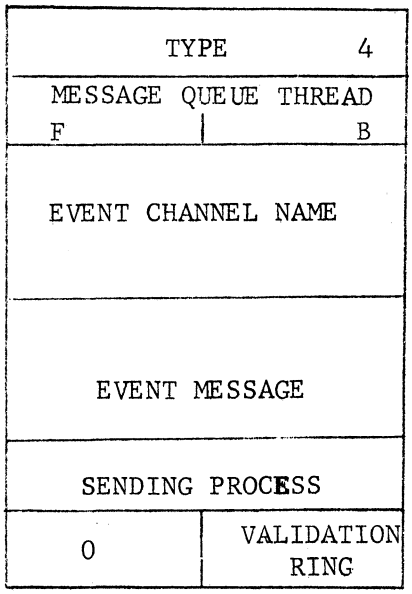


EVENT CALL TRAILER

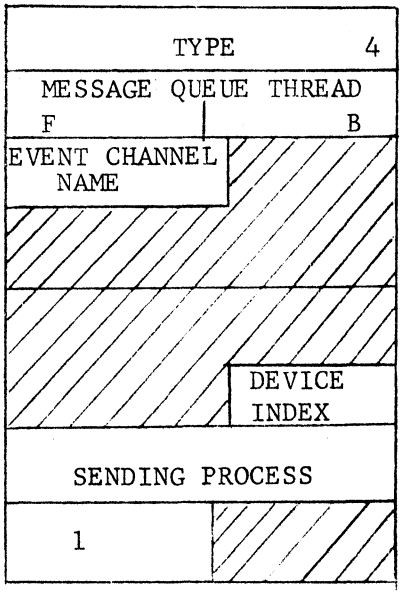


ASSOCIATED PROCEDURE ENTRY

FIGURE 1a



USER EVENT MESSAGE



DEVICE SIGNAL MESSAGE

FIGURE 1b

THE EVENT CALL CHANNEL

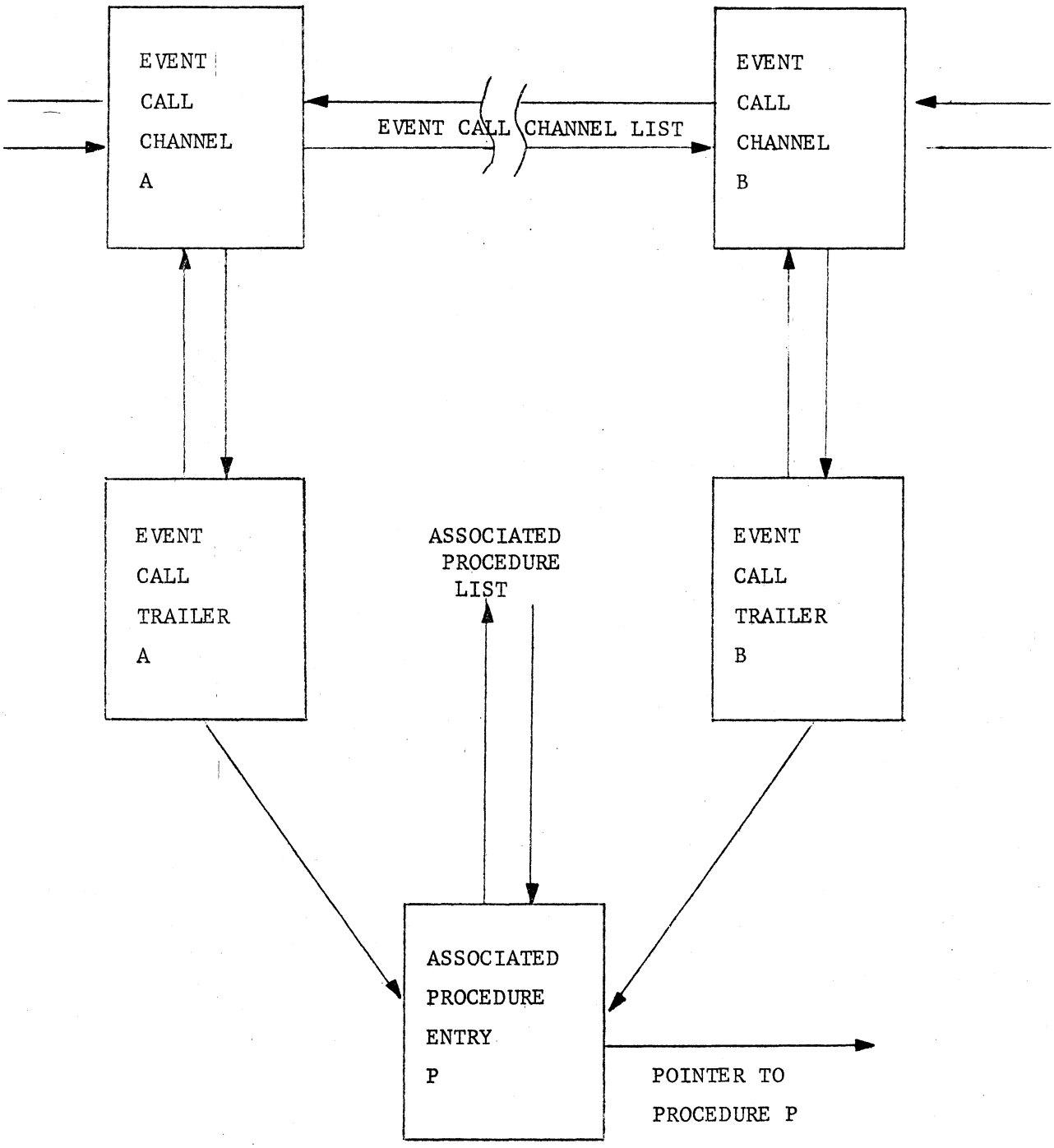


FIGURE 2

THE EVENT CHANNEL TABLE

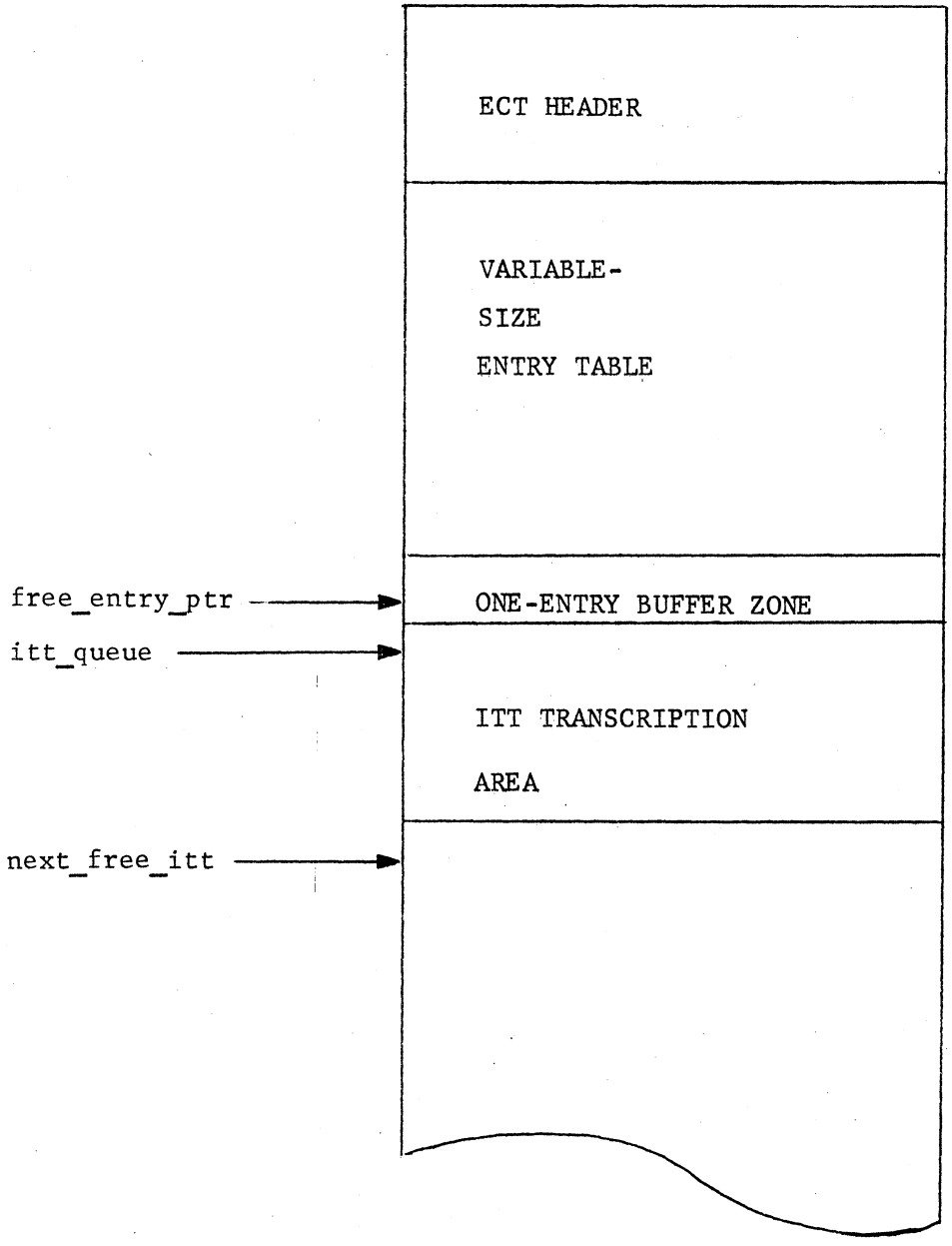


FIGURE 3