Indentification

Interprocess Event Signalling Primitives

R. J. Feiertag

Purpose

The following discussion describes some primitive functions inherent to event signalling.

Primitives

```
call signal_event(user_process_id, channel_branch_directory,
                   channel_branch_name, code);

call signal_event_ptr(user_process_id, channel_branch_ptr, code);
```

This supervisor primitive causes the process specified by user_process_id (a character string identifying a process, e.g., Feiertag.Multics.34) ← !
to be signalled on the channel associated with the branch specified by channel_branch_directory and channel_branch_name or channel_branch_ptr. As in all succeeding primitives any errors are reflected in code, a zero code means no error has occurred.

```
call block(channel_list, channel_index, static_data, code);
```

This user ring call causes the process to give up the processor if no event signals are pending. When an event signal arrives over a channel contained in channel_list this call returns. While in block, a procedure associated with a synchronous call channel may be invoked if an event signal is pending on that channel. Channels may be specified in one of three manners: as a number (special channels only), as a path name, of a branch associated with a channel, or as a pointer of a branch associated with a channel (another possibility is the unique id of a branch associated with a channel). Various means will be provided for expressing these various types of identification in a channel_list. channel_index indicates which channel in channel_list caused the return. static_data is the pointer associated with the channel when it was created.

```
    call create_event_channel(channel_branch_directory, channel_branch_
                name, type, static_data, procedure, sync, priority,
                code);


    call create_event_channel_ptr(channel_branch_ptr, type, static_data,
                procedure, sync, priority, code);
```

This user ring primitive creates an event channel by making an entry in the event channel table (ECT). A subsequent call to signal_event specifying this branch will cause an event to be signalled on this channel. If the specified branch does not exist it will be created with default access only to the creating process. type indicates a call or wait channel. static_data is a pointer to some data that will be passed to the calling or called procedure when the event is processed. procedure is the event call procedure. sync is the call channel synchronization, synchronous or asynchronous. priority is the priority of this channel. Note that procedure, sync, and priority apply only to call channels.

call delete_event_channel(channel_branch_directory, channel_branch_name, code);

call delete_event_channel_ptr(channel_branch_ptr, code);

The channel associated with the specified branch is deleted. The branch itself is not deleted.

call create_special_channel(channel_number, type, static_data, procedure, sync, priority, code);

A special channel is created and its number is returned in channel_number. Other arguments are the same as in create_event_channel.

call delete_special_channel(channel_number);

This special channel known by channel_number is deleted.

call signal_special_event (process_id, channel_number, code);

This supervisor primitive, callable only within the supervisor, causes the process specified by process_id to be signalled with an event over the special channel specified by channel_number.