EE458 - Embedded Systems
Lecture 15 – Events and Signals

• Outline
  - Events
  - Signals

• References
  - RTC: Chapter 8
  - CUG: Chapters 15, 16
Events and Signals
Introduction to Events

- An event register is a special register in a task's TCB. The event register consists of a group of binary event flags that are used to track the occurrence of certain events.

- Not all RTOSes support events.

- Event registers are either 8, 16, or 32 bits wide. Each bit is treated as a binary flag and can be either set or cleared.
Events and Signals
Introduction to Events

An Event Register
Events and Signals
Event Register Features

- The application programmer must associate an event with a particular event flag.
- Event flags can be set by any task or ISR. Only the task associated with the TCB that contains the event register can wait on an event (or set of events).
- Events are not queued. If an event is sent multiple times before being received, only the first send has any effect.
Events and Signals

Event Register Features

- A task can specify the set of events it wants to receive. An event set is constructed by either ANDing or ORing bits from the event register.

- If an AND is used then all event bits must be set before the waiting task unblocks.

- If an OR is used the task will unblock when any event bit is set.
Events and Signals
Event Applications

- Event registers are typically used for **unidirectional** activity synchronization. No data is associated with an event. Other methods must be used for passing data.

- An event register does not have a built-in mechanism for identifying the source of an event. This can be achieved by associating bit groups from the event register with particular tasks as shown on the next slide.
Events and Signals
Event Applications

Identifying The Source of an Event
Events and Signals
RTEMS Events

- There are only two RTEMS event directives: send and receive. (The event register is automatically created as part of the TCB.) RTEMS event registers are 32-bit registers.
- The prototype for the event send directive is:

```c
rtems_status_code rtems_event_send (rtems_id id, rtems_event_set event_in);
```
Events and Signals

RTEMS Events

- The `id` argument is the id of the task that owns the event register.
- The `event_in` argument specifies the event flags that should be set. The set of valid events is RTEMS_EVENT_0 through RTEMS_EVENT_31. Several events can be set simultaneously by `ORing` these constants.
Events and Signals

RTEMS Events

• The receive directive looks like:

```c
rtems_status_code rtems_event_receive (  
rtems_event_set event_in,  
rtems_option option_set,  
rtems_interval ticks,  
rtems_event_set *event_out
);
```

• This directive causes the task to wait until the event given in `event_in` occurs.
Events and Signals
RTEMS Events

- The `option_set` should be created using the `RTEMS_WAIT` (default), `RTEMS_NO_WAIT`, `RTEMS_EVENT_ALL` (AND, default) and `RTEMS_EVENT_ANY` (OR) constants.

- The ALL option will cause the task to wait until all events have occurred. The ANY option will cause the task to wait only until any single event has occurred. You can also use the `RTEMS_DEFAULT_OPTIONS` constant for the default set of options.
Events and Signals
RTEMS Events

- When the RTEMS_WAIT option is used, the ticks argument specifies a maximum wait time. Set ticks to RTEMS_NO_TIMEOUT to cause the task to wait forever.

- The event_out parameter is returned to the calling task with the value that corresponds to the events in event_in that were satisfied. The events will be cleared in the event register.
Events and Signals
RTEMS Events

- Set `event_in` to RTEMS_PENDING_EVENT to poll the event register. The event register is returned in the `event_out` argument and the event set is left unaltered.

- Set `event_in` to RTEMS_ALL_EVENTS and use the RTEMS_NO_WAIT and RTEMS_EVENT_ANY options to clear the event register. The old value of the register is returned in the `event_out` argument.
Events and Signals
Introduction to Signals

- A signal is a software interrupt that is sent from one task or ISR to another task. The receiving task is diverted from its normal execution path and runs a signal handler.

- Not all RTOSes support signals. (RTEMS does)

- Signals are asynchronous, that is the receiving task may be interrupted by a signal at any time.
Events and Signals
Introduction to Signals

- The number of signals is RTOS dependent.
- To process a signal a task must install a signal handler (also known as an asynchronous signal routine or ASR). Some RTOSes may supply a default ASR.
- Some RTOSes allow a different signal handler to be installed for each signal. Some allow only a single ASR to be installed that must handle all signals.
Events and Signals
Introduction to Signals

Signals
# Events and Signals

## Typical Signal Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch</td>
<td>Install a signal handler</td>
</tr>
<tr>
<td>Release</td>
<td>Removes an installed handler</td>
</tr>
<tr>
<td>Send</td>
<td>Send a signal to another task</td>
</tr>
<tr>
<td>Ignore</td>
<td>Prevent delivery of a signal</td>
</tr>
<tr>
<td>Block</td>
<td>Block signal delivery</td>
</tr>
<tr>
<td>Unblock</td>
<td>Allow signal delivery</td>
</tr>
</tbody>
</table>
RTEMS allows a task to install only a single ASR to handle all signals. (The ASR can take different actions depending on the signal.)

There are only two RTEMS signal related directives: \texttt{rtems\_signal\_catch()} and \texttt{rtems\_signal\_send()}.

A signal set is constructed by a bitwise OR of the desired signals. Valid signals are RTEMS\_SIGNAL\_0 to RTEMS\_SIGNAL\_31.
Events and Signals
RTEMS Signals

• The catch routine has the following prototype:

```c
rtems_status_code rtems_signal_catch(
    rtems_asr_entry asr_handler,
    rtems_mode       mode
);
```

• The mode argument specifies the task mode under which the ASR is run. You can choose to disable preemption, timeslicing, etc.
Events and Signals

RTEMS Signals

• The calling syntax for the user supplied ASR is:

```c
rtems_asr_user_routine(
    rtems_signal_set_set signals
);
```

• The ASR receives the set of **pending** signals as an argument. The signals are cleared when the ASR is run.
Events and Signals

RTEMS Signals

- A task may invalidate its ASR and discard all pending signals by using a NULL value as the argument in a call to the catch directive. New signal sets sent to the task are then discarded.

- Any directive that can be called from a task can be called from an ASR.

- Recall that signal processing can be enabled and disabled via rtems_task_mode() directive calls.
Events and Signals
RTEMS Signals

- The calling syntax for the send directive is:
  
  ```c
  rtems_status_code rtems_signal_send(
      rtems_id         id,
      rtems_signal_set signal_set
  );
  ```

- This may be called from any task or ISR. The `id` is the ID of the task being signaled. The `signal_set` contains the signal set to be sent to the task.
Events and Signals

RTEMS Signals

- If a signal set is sent to a task whose ASR is valid but disabled, the signal set is left pending.
- Sending a signal set has no effect on a task's state. If the task is blocked, the task will remain blocked. The signals will be processed when the task becomes the running task.