is at the end of task $T_j^s$, while in Figure 12.17(c) the clearance of the software flag is the first thing to do in $T_j^s$.

### 12.3.3 Worst-Case Event Response Time

Let us use Figure 12.18 to analyze the worst-time event response time. Our analysis is based on the following assumptions. For each device $i (1 \leq i \leq k)$,

1. task $T_i^s$ appears only once in the circular structure.
2. software flag $SW_{\text{flag}_i}$ is asserted only in $ISR_i$. In addition, the design pattern as shown in the center of Figure 12.18 is used for $ISR_i$: $SW_{\text{flag}_i}$ is asserted (to acknowledge the triggering request from device $i$) only when $SW_{\text{flag}_i}$ is not currently asserted.
3. software flag $SW_{\text{flag}_i}$ is cleared only in the service task $T_i^s$. In addition, the clearance of $SW_{\text{flag}_i}$ is placed at the beginning of $T_i^s$ (i.e., the case as illustrated in Figure 12.17(c)).
4. the execution time of a decision point (condition evaluation and branching) is negligible. In particular, $e_i^a$ (the execution time of $ISR_i$) is 0 when $SW_{\text{flag}_i}$ is asserted.

---

**Figure 12.18**
Worst-case response time: $ISR_j$ executes as soon as flag $SW_{\text{flag}_j}$ has been cleared at the beginning of $T_j^s$. 
One worst-case scenario is illustrated in Figure 12.18:

- \( T_j^s \) starts to execute to service the current service request from device \( j \). The first thing to do in \( T_j^s \) is to clear \( SW\_flag\_j \).
- As soon as \( SW\_flag\_j \) is cleared, a new service request from device \( j \), denoted by \( \gamma_j \), triggers the execution of \( ISR_j \). In the worst case, device \( j \) has the lowest interrupt priority, and each of the other devices has raised a request that preempt the execution of \( ISR_j \). In such a worst case, the total execution time of the ISRs, including the one acknowledging \( \gamma_j \), is given by \( \sum_{i=1}^{k} e_i^a \). After this, the new request \( \gamma_j \) is successfully acknowledged.
- \( T_j^s \) completes its service to the “old” request in \( e_j^s \). Now, \( \gamma_j \) is the only request from device \( j \) to be serviced.
- Along the execution path, in the worst case, there is at most one request from each device \( i \) \((1 \leq i \leq k, i \neq j)\) that can be successfully acknowledged (after servicing the “old” request). Thus, for each \( i \) \((1 \leq i \leq k, i \neq j)\), the execution time is \( e_i^s + e_i^a \);
- As the control comes to \( T_j^s \) again, it starts to service \( \gamma_j \).
- As soon as \( SW\_flag\_j \) is cleared, another new service request from device \( j \) triggers the execution of \( ISR_j \). The amount of execution time is \( e_j^a \), and this new request is successfully acknowledged.
- \( T_j^s \) completes its service to \( \gamma_j \) in \( e_j^s \).

Thus, according to the above worst-case analysis, the worst-case event response time for a device \( j \) is given by

\[
e_j^s + \sum_{i=1}^{k} (2 \times e_i^a + e_i^s).
\]

As we have explained before, owing to the introduction of interrupts, the worst-case outstanding period for a device \( j \) \((1 \leq j \leq k)\) is \( \sum_{j=1}^{k} e_j^a \)—the execution time of all ISRs. As compared with the round-robin architecture, this is significantly less and leads to much better hardware concurrency.

Table 12.6 summarizes the four processing stages for external events.

<table>
<thead>
<tr>
<th>Event from Device ( i )</th>
<th>Raised</th>
<th>Detection</th>
<th>Acknowledgment</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>At</td>
<td>By</td>
<td>At Amount</td>
<td>By</td>
</tr>
<tr>
<td>Round robin</td>
<td>Any time</td>
<td>Software</td>
<td>( T_i^d ) ( e_i^d &gt; 0 )</td>
<td>Software</td>
</tr>
<tr>
<td>Round robin with interrupts</td>
<td>Any time</td>
<td>Hardware (in no time)</td>
<td>( e_i^d = 0 )</td>
<td>Software</td>
</tr>
</tbody>
</table>