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 ($1 \le i \le k$),

(1) task T_i^s appears only once in the circular structure.

(2) software flag SW_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_flag_i is asserted (to acknowledge the triggering request from device i) only when SW_flag_i is not currently asserted.

(3) software flag SW_flag_i is cleared only in the service task T_i^s. In addition, the clearance of SW_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_flag_i is asserted.

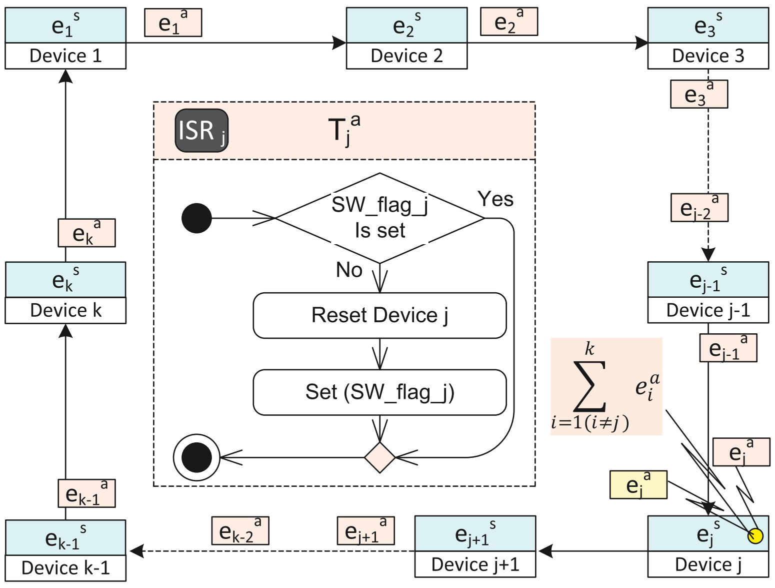

Figure 12.18
Worst-case response time: ISR_j executes as soon as flag SW_flag_j has been cleared at the beginning of T_j^s.

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 γ_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 preempts the execution of ISR_j. In such a worst case, the total execution time of the ISRs, including the one acknowledging γ_j, is given by $\sum_{i=1}^{k} e_i^{\mathrm{a}}$. After this, the new request γ_j is successfully acknowledged.
- T_j^{s} completes its service to the "old" request in e_j^{s}. Now, γ_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 $(1 \le i \le k, i \ne j)$ that can be successfully acknowledged (after servicing the "old" request). Thus, for each i $(1 \le i \le k, i \ne j)$, the execution time is $e_i^{\mathrm{s}} + e_i^{\mathrm{a}}$;
- As the control comes to T_j^{s} again, it starts to service γ_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 γ_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

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

As we have explained before, owing to the introduction of interrupts, the worst-case outstanding period for a device j $(1 \le j \le k)$ is $\sum_{j=1}^{k} e_j^{\mathrm{a}}$—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 12.6 Processing stages for external events

Event from Device *i*	Raised	Detection			Acknowledgment			Service		
	At	By	At	Amount	By	At	Amount	By	At	Amount
Round robin	Any time	Software	T_i^{d}	$e_i^{\mathrm{d}} > 0$	Software	T_i^{a}	$e_i^{\mathrm{a}} \ge 0$	Software	T_i^{s}	$e_i^{\mathrm{s}} > 0$
Round robin with interrupts	Any time	Hardware (in no time)		$e_i^{\mathrm{d}} = 0$	Software	ISR_i	$e_i^{\mathrm{a}} \ge 0$	Software	T_i^{s}	$e_i^{\mathrm{s}} > 0$