SIEMENS

Microcontrollers

ApNote AP1636

☐ additional file
APXXXX01.EXE available

PEC - „Lowering the own Interrupt Priority“

Very often it is needed to lower the interrupt priority while running a interrupt-service-routine (ISR),like after the last PEC-transfer.

Harald Lehmann / Siemens Cupertino

1 Abstract 3

2 Lower the “own” priority by changing the ILVL-field in the PSW-Register 4

2.1 The interrupted-level was “low” 4

2.2 The interrupted-process has a higher priority then the targeted priority for the “End-of-PEC-Interrupt” service routine 4

3 A different approach 6

AP1636 ApNote - Revision History		
Actual Revision : Rel.01	Previous Revison: none	
Page of actual Rel.	Page of prev. Rel.	Subjects changes since last release)

1 Abstract

Very often it is needed to lower the interrupt priority while running a interrupt-service-routine (ISR),like after the last PEC-transfer.

Example: For storing a time-critical signal, a PEC-transfer will be used. Because the processing of the stored data is not time-critical it will be done in the ISR triggered by the last PEC transfer.

The PEC will be initialized with a counter value of “1”. So, after a single PEC-transfer an additional Interrupt is generated. This interrupt is on the same high interrupt level as the PEC (14 or 15). If this PEC-triggered ISR stays at this level nearly every other interrupt is disabled and has to wait, even while the program is now running the non-critical part.

Now it is on the programmers responsibility to free the system for higher priority-levels. There are different possibilities to do that.

2 Lower the “own” priority by changing the ILVL-field in the PSW-Register

On the first view this looks like the best and easiest solution, although it will have severe disadvantages.

2.1 The interrupted-level was “low”

The CPU level is 0, no other interrupt is served. Now, an Interrupt with a single PEC-transfer with a followed “End-of-PEC-Interrupt” is running. Within the ISR the priority is lowered for other high-level-requests. The system will work correctly.

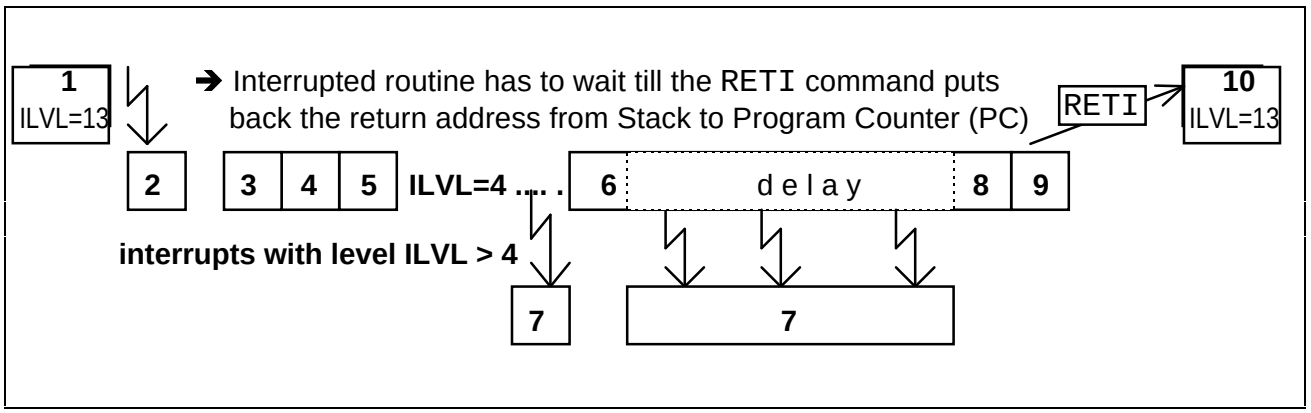
2.2 The interrupted-process has a higher priority then the targeted priority for the “End-of-PEC-Interrupt” service routine

The targeted priority for the “End-of-PEC-Interrupt” service routine is defined as the priority which is changed in the ISR.

Example: An Event with the high priority “ILVL=13” can accept a latency within 10µs. Only PEC-transfers are usually able to interrupt level 13. Because a PEC-transfer will only “steal” one cycle, it will not be a problem due to the latency requirement. Every PEC-transfer (PEC-counter <255) will end up with the “End-of-PEC-Interrupt” and will take more or less time in processing. Assuming the serial interface received 20 character, which need further processing in the “End-of-PEC-Interrupt”-service-routine. And assuming the time-window for this sequence is 50ms.

The following sequence displays this case:

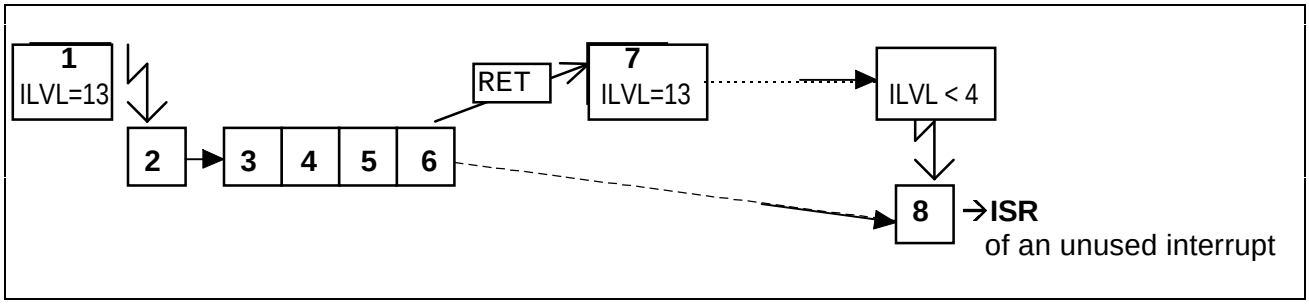
1. CPU running with high-priority-event at level ILVL=13
2. The last character of a block(20) is received by the serial interface
3. PEC-transfer is followed by the “End-of-PEC-Interrupt”
4. Reinitialize the PEC for the serial interface
5. Lower the “own” priority-level to ILVL=4
6. Start processing of the 20 characters, will take max. 50ms.
7. different interrupts with level ILVL>4 will occur
8. Finish the processing of the 20 characters.
9. ISR will be terminated with `RETI`
10. Reestablishing of interrupted routine at level ILVL=13



With this setup a latency-time smaller than 10µs is nearly impossible to guarantee!

3 A different approach:

1. CPU running with high-priority-event at level ILVL=13
2. The last character of a block(20) is received by the serial interface
3. PEC-transfer is followed by the “End-of-PEC-Interrupt”
4. Reinitialize the PEC for the serial interface
5. Setting of an unused interrupt-request-flags(at level ILVL=4) which is pointing to the routine to processing the 20 characters
6. Terminate “End-of-PEC-Interrupt” with `RETI`
7. **Reestablishing of interrupted routine at level ILVL=13**
8. Routine to process the 20 characters will wait till its priority (ILVL=4) is high enough to interrupt the current CPU priority.



The disadvantage of this solution is that 2 interrupt-vectors for one event are used, and note that the code-size will increase slightly.

Alternatively could the interrupt priority during the ISR in the ILVL-Register be lowered (changed), then setting the “own” Interrupt-Request-Flag and processing the `RETI` instruction. A check of the current interrupted CPU level at the beginning of the ISR has to be implemented, too.

This should only be used if there are NO other interrupt vectors available and NO event which need a PEC response would occur during this procedure.

Because of the above mentioned reasons, “lowering the *own* priority” should be avoided.