Recap SW3

Claudio Imboden, Dominik Geisseler T5

Synchronisation

- Warum synchronisieren?
  - Damit der Prozess mit der Umgebung interagieren kann
  - Korrektes Resultat zur richtigen Zeit

- Realtime Synchronisation
  - Einfach
  - Ineffizient
  - Nicht übertragbar

  (Compiler, Clock)

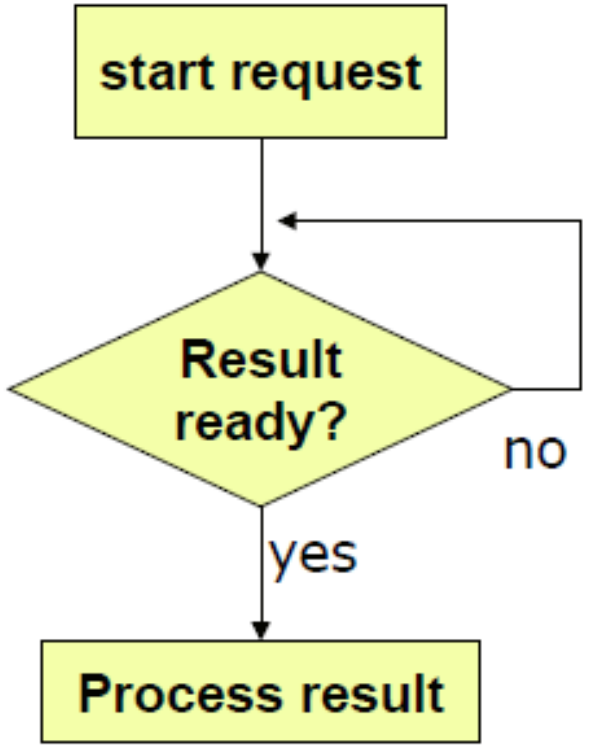
```
#pragma OPTION ADD "-Cu=i100"

void delay(void) {
  volatile unsigned char i;

  for(i=0;i<100;i++);
}
```

Synchronisation

- Polling/Gadfly
  - Wiederholt abfragen bis das Resultat bereit ist.
  - Programm ist blockiert
  - Aktives warten

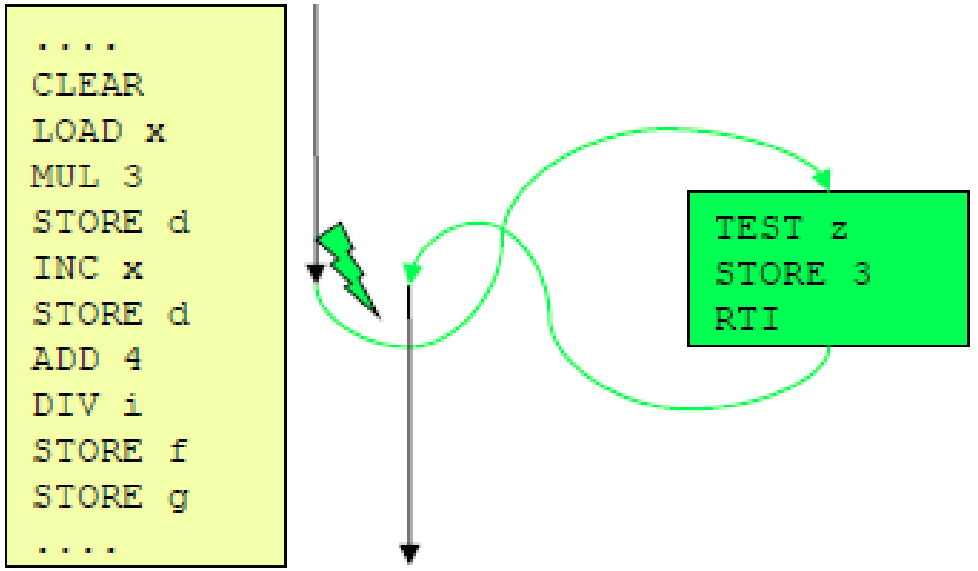


Source: Google Picture Library

Synchronisation

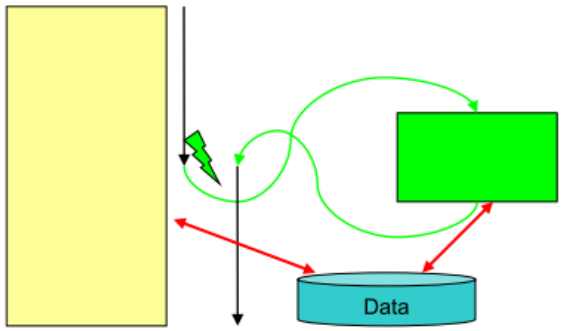
- Interrupts
  - Kein Warten
  - Bessere Performance, aber mehr Overhead
  - ISR so kurz wie möglich
  - Kompliziert

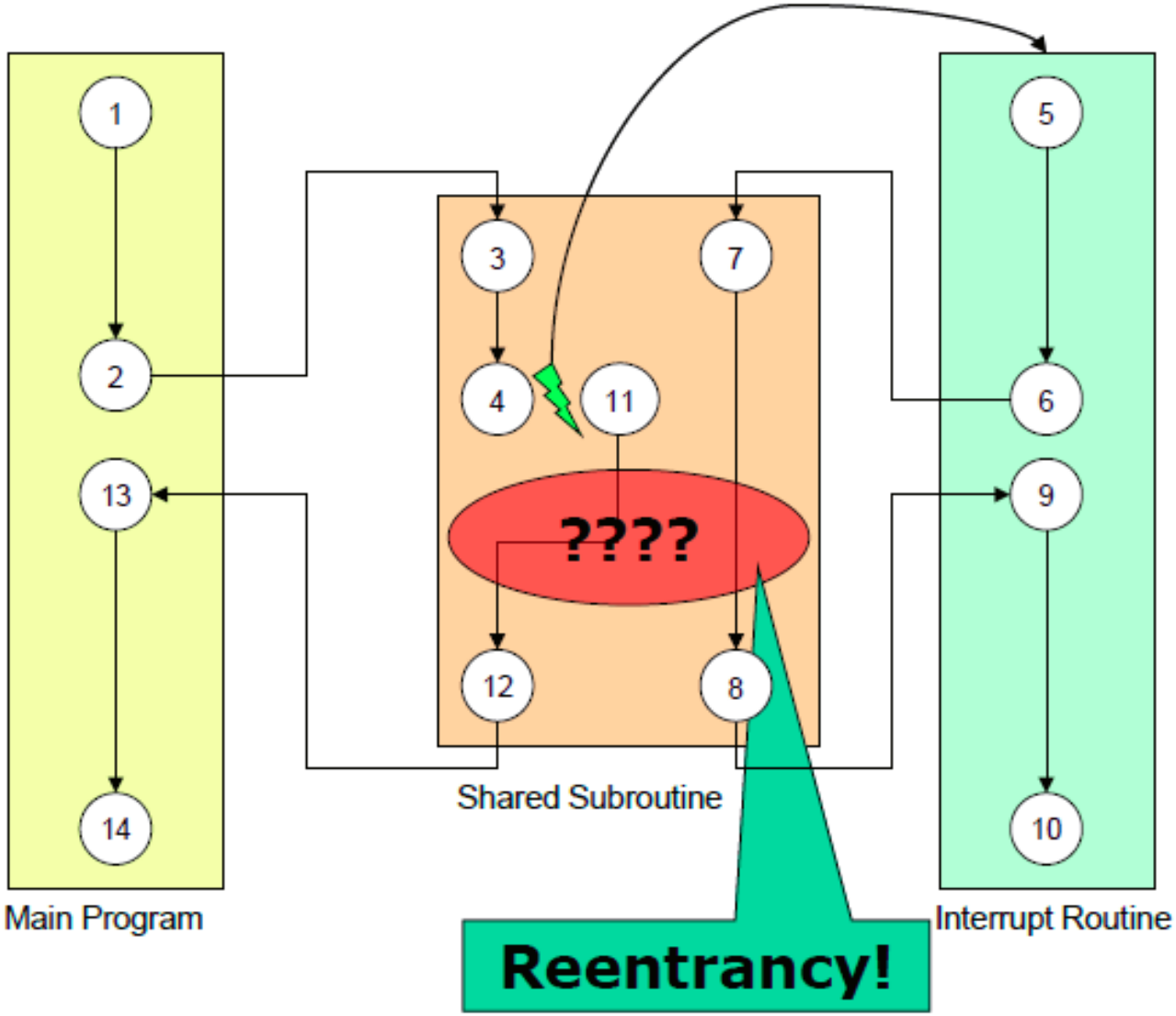
  (Debugging, Priorität)



Reentrancy

- Keine globalen Variablen
- Keine gemeinsame Ressourcen
- Deaktivieren von Interrupts



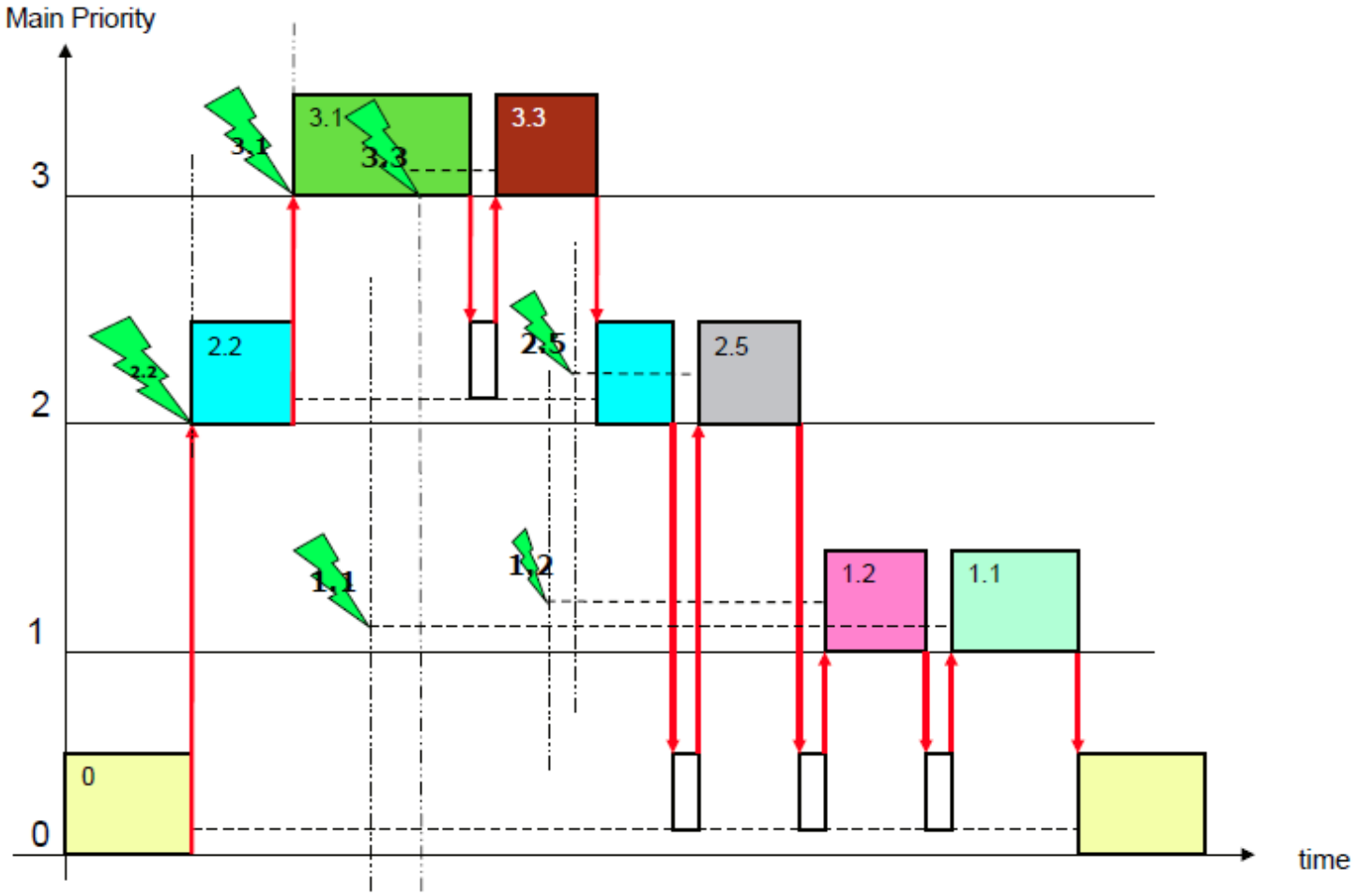


Interrupt priorities

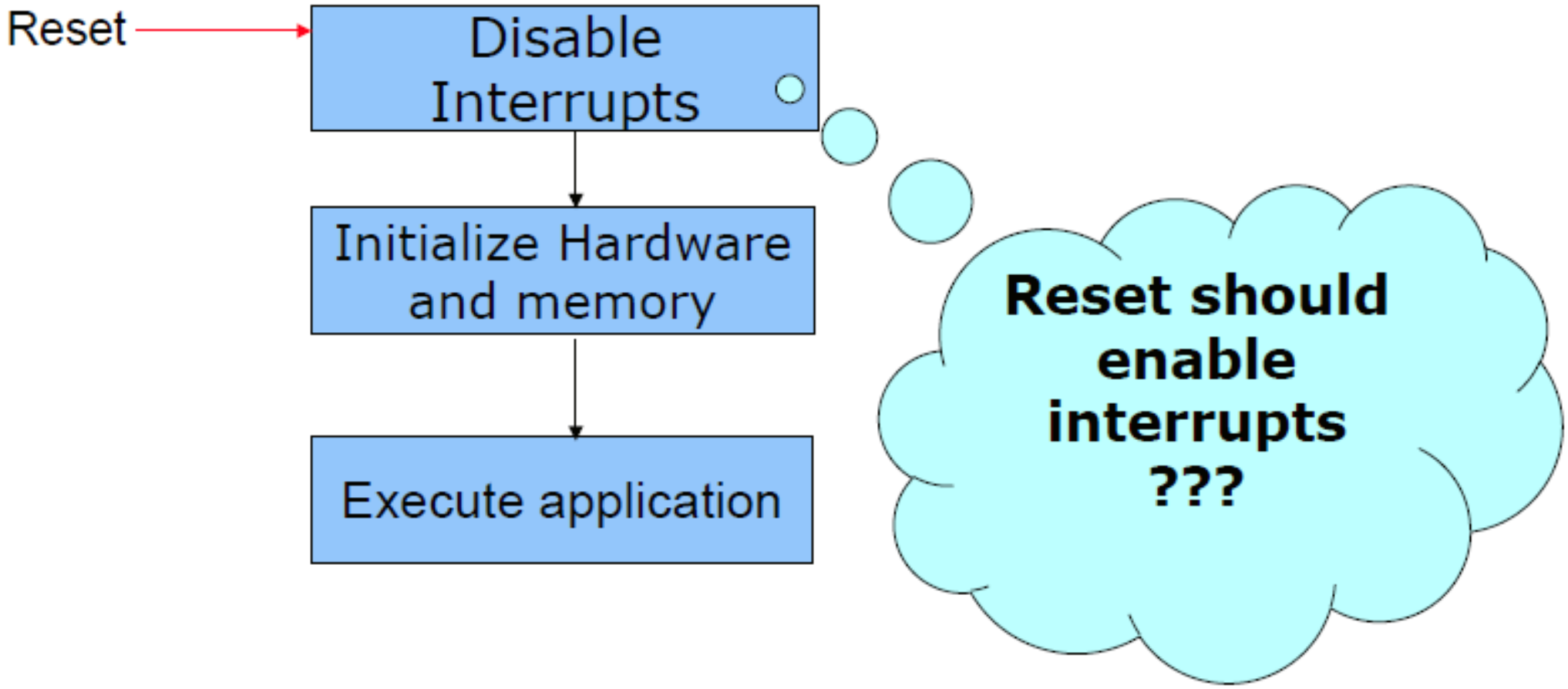
- Interrupts können auch unterbrochen werden!
- Vorsicht!!! Nicht alle Architekturen verwenden die selben Prioritätsregeln.

Interrupt priorities

- Prioritäten und Subprioritäten
- Nach jeder ISR wird der Stack zuerst wieder abgebaut



Interrupts


Reset
Disable Interrupts
Initialize Hardware and memory
Execute application
Reset should enable interrupts ???

Interruptverwendung

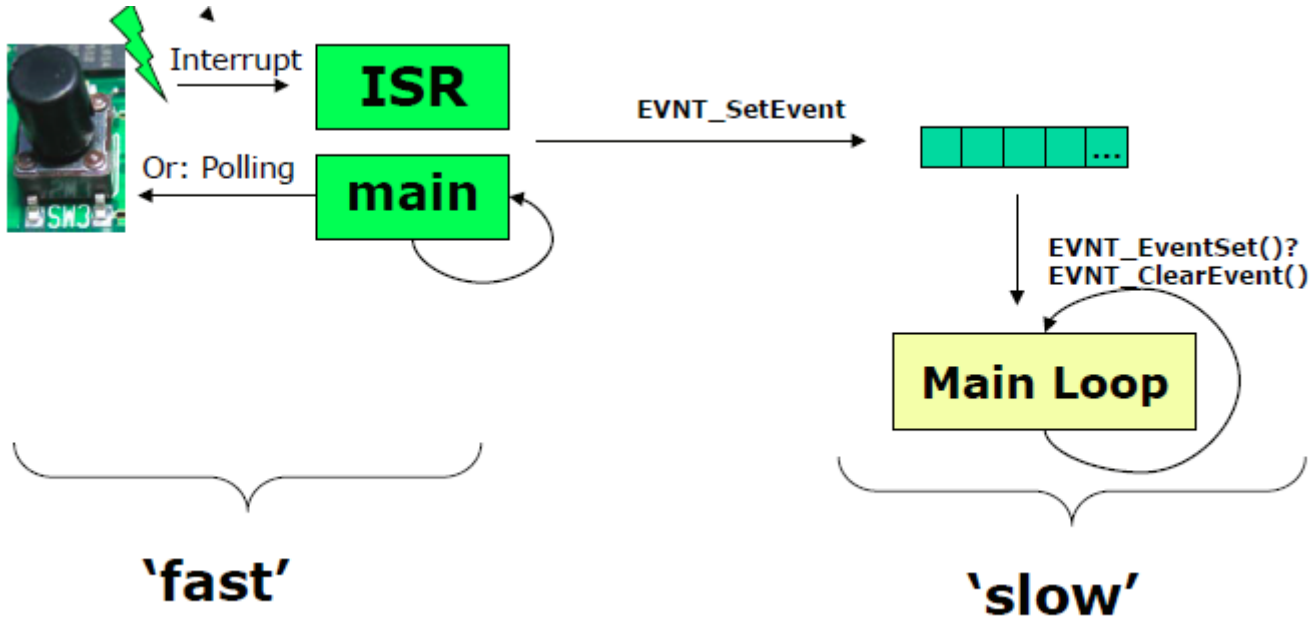
- Nur für Sachen die nicht warten können
- Verringern der Komplexität

Events

- Synchron vs. Asynchron
  - Timer vs. Knopfdruck
- Brauchen Infrastruktur
  - Setzen, löschen und Überprüfen
- Allgemein
  - Ausführung Zeitintensiver Operationen
  - Sequentielle Aktionen (Knopfdruck -> Start Countdown)
  - Nesting: Ein Event löst einen neuen Event aus
  - Auftrennung von Quelle des Events und Ausführung

Events

- Nach Möglichkeit Interruptroutinen nur zum Setzen von Eventflags benutzen



Fragen

1. Warum sollte die Laufvariable in einem einfachen Delay-Loop mit volatile markiert werden?

2. Welche Vorteile bietet die Gadfly-Synchronisation gegenüber Interrupts?

3. Was ist eine atomare Anweisung?

4. Was ist das Problem bei «shared data» im Zusammenhang mit Interrupts?

5. Warum sollten die Anweisungen EnterCritical() und ExitCritical() nicht verschachtelt verwendet werden.

Fragen

1. Warum sollte die Laufvariable in einem einfachen Delay-Loop mit volatile markiert werden?

Beispiel:

```
void delay(void) {
  volatile unsigned char i;
  for(i=0;i<100;i++);
}
```

Der Compiler versucht evtl. lehre Schlaufen wie z.b for(i=0;i<100;i++); aus dem Code zu löschen um Prozessorzyklen zu sparen. Mit dem Schlüsselwort volatile markierte Variablen könnten sich zwischen den Abfragen durch Zugriff eines fremden Prozesses ändern. Der Compiler optimiert den Code nicht.

2. Welche Vorteile bietet die Gadfly-Synchronisation gegenüber Interrupts?

- Weniger Overhead
- Einfacher
- Bei häufigen Interrupts/Events schneller

3. Was ist eine atomare Anweisung?

Eine atomare Anweisung kann nicht durch einen Interrupt unterbrochen werden.

4. Was ist das Problem bei «shared data» im Zusammenhang mit Interrupts?

Die gemeinsamen Daten werden evtl. zwischen dem lesen und schreiben von einem anderen Prozess manipuliert. Somit können Inkonsistente Zustände auftreten.

5. Warum sollten die Anweisungen EnterCritical() und ExitCritical() nicht verschachtelt verwendet werden?

Bei der Anweisung EnterCritical(); wird das aktuelle CCR-Register abgespeichert und anschließend das Interruptbit gelöscht. Wird die Funktion nochmals aufgerufen wird nun das bereits geänderte CCR-Register abgespeichert.