Waitms Methode erklärt

Diese Funktion `Waitms` dient dazu, den Programmablauf für eine bestimmte Anzahl von Millisekunden zu verzögern. Sie verwendet einen 16-Bit-Timer/Zähler (`counter`), der vermutlich durch einen Timer-Interrupt inkrementiert wird. Hier ist, wie die Funktion funktioniert:

1. `static uint16_t aktTime, diff;` deklariert zwei statische 16-Bit-Variablen, die ihren Wert zwischen Aufrufen der Funktion behalten.
2. `uint16_t countertemp;` deklariert eine temporäre 16-Bit-Variable.
3. `cli();` deaktiviert globale Interrupts, damit der Zähler `counter` nicht während des Zugriffs verändert wird.
4. `aktTime = counter;` speichert den aktuellen Wert des Zählers in `aktTime`.
5. `sei();` aktiviert die globalen Interrupts wieder.
6. Die `do`-`while`-Schleife beginnt:
   - `cli();` deaktiviert Interrupts.
   - `countertemp = counter;` speichert den aktuellen Wert des Zählers in `countertemp`.
   - `sei();` aktiviert Interrupts wieder.
   - `diff = countertemp + ~aktTime + 1;` berechnet die Differenz zwischen `countertemp` und `aktTime` mit Übertragsberücksichtigung.
   - Die Schleife läuft weiter, solange `diff` kleiner als `msWait` ist.

Wenn die Schleife beendet ist, ist die gewünschte Verzögerung von `msWait` Millisekunden abgelaufen.

Der Grund für die Deaktivierung/Aktivierung der Interrupts ist, dass der 16-Bit-Zähler `counter` möglicherweise während des Zugriffs durch einen Interrupt verändert werden könnte. Dies könnte zu falschen Ergebnissen führen, wenn die beiden Bytes des 16-Bit-Wertes nicht atomar gelesen werden. Die `cli()`/`sei()`-Anweisungen stellen sicher, dass der Zähler nicht zwischen dem Lesen der beiden Bytes verändert wird.

Die Verwendung der temporären Variable `countertemp` zusammen mit der Differenzberechnung `diff = countertemp + ~aktTime + 1` berücksichtigt den Fall, dass der Zähler `counter` während der Verzögerung überläuft (von 0xFFFF auf 0x0000 zurückspringt).