1 Zeitgeber/Zähler 1
1.1 Zähler 2
1.2 Taktwahl, Vorteiler 3
1.3 Zeitstempel (Capture) 3
1.4 Frequenzerzeugung 4
1.5 Ausgangsschaltung 5
2 Typische Anwendungen 6
2.1 Pulsweitenmodulation (PWM) 6
2.1.1 Anwendung Servomotor 7
2.2 Echtzeituhr 7
2.3 Watchdog 7

1 Zeitgeber/Zähler

Nach den GPIOs sind Zähler/Zeitgeber (Timer/Counter) die wichtigste und universellste Peripherieeinheit eines µC. Daher findet sich in jedem heutigen µC mindestens ein derartiges Element, meist aber eine Vielzahl.

Das liegt daran, dass mit einem Timer viele verschiedene technische Aufgaben gelöst werden können. Die Anforderungen sind dabei von der jeweiligen Aufgabe abhängig. Deswegen finden sich aus Kostengründen oft Elemente mit unterschiedlichen Merkmalen, d.h. nicht jeder Zähler/Zeitgeber kann gleich gut für jede Aufgabe verwendet werden.

Der Praktikums-µC hat gleich 12 Module, die Timer als zentrales Element beinhalten. Das sind ein Systick-Timer, TIM1, TIM2, TIM3, TIM6, TIM7, TIM15, TIM16, LPTIM, die RTC und zwei Watchdoggs (IWDG und WWDG)

Im Folgenden wird ein idealisiertes und vereinfachtes Element behandelt, das alle üblichen Funktionen erfüllen kann. Der Aufbau orientiert sich an einem Universaltimer. Die besprochenen Elemente können bei unterschiedlichen µC anders heißen, sind aber in ihrer Funktion ebenso vorhanden.

Sehr häufig bieten moderne µC noch mehr Erweiterungen an einem Timer an, um spezielle Aufgaben besser lösen zu können. Solche Erweiterungen werden hier nicht vorgestellt.

Gelbe Kästchen stellen Ergänzungen zu bereits vorher beschriebenen Funktionen dar. Die Kästchen selbst stellen Funktionseinheiten dar, die vom Programm beeinflusst werden können (indem Werte in passende Peripherieregister geschrieben werden) oder die Ergebnisse an das Programm liefern können (indem passende Peripherieregister gelesen werden). Wichtig ist dabei, dass diese Elemente unabhängig von Rechenkern sind, d.h. sie arbeiten parallel zum Programm und erfüllen nach der Einstellung ihre Aufgabe ohne Verbrauch von Rechenzeit. Unter bestimmten Umständen kann zur Energieeinsparung auch der Rechenkern selbst schlafen gelegt werden, wobei einzelne Zähler/Zeitgeber noch weiterhin funktionieren und ggf. den Kern wieder aufwecken.

1.1 Zähler

Das Kernelement ist der Zähler, der eine bestimmte Bitbreite *n* hat (z.B. 16 oder 32 Bit). In Abbildung 1 ist das das Register *CNT*.

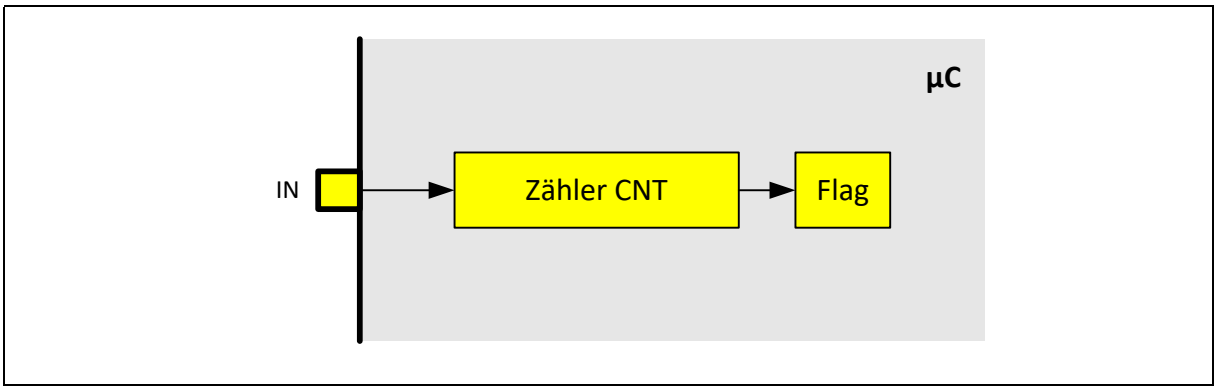

Abbildung 1: Kernelement: Zähler

Bei jeder steigenden Flanke an einem Pin (hier *IN*) wird der Zähler um eins hochgezählt. Der µC kann den Zähler sowohl mit einem neuen Wert beschreiben (z.B. auf null setzen) als auch den aktuellen Wert auslesen. Wenn der Maximalwert 2^n-1 erreicht wird, dann gibt es mehrere Möglichkeiten (Abbildung 2).

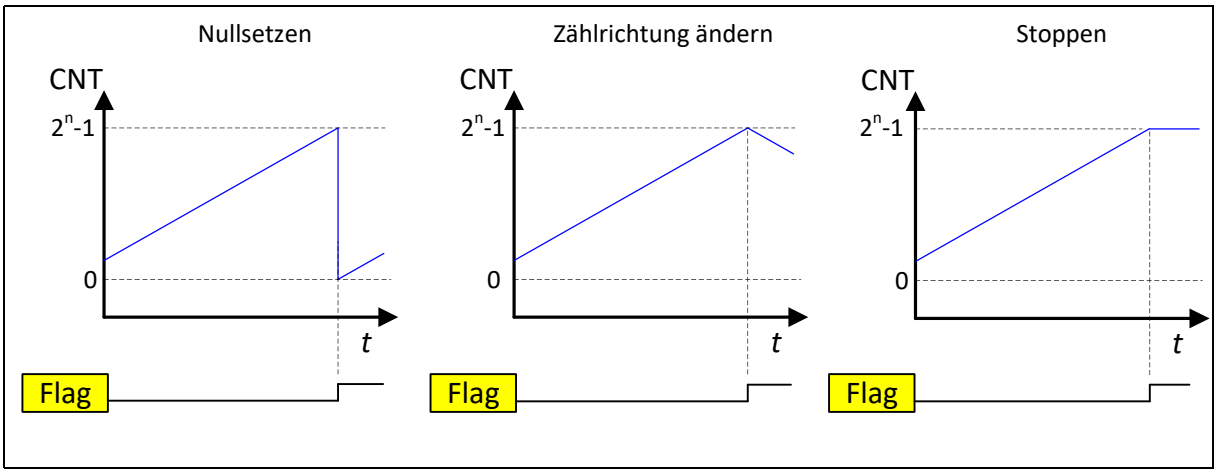

Abbildung 2: Zähler nach Erreichen des Endwerts

Am weitesten verbreitet (d.h. in allen übliche µC verfügbare Option) ist das automatische Nullsetzen. Der Zähler beginnt wieder mit der Zahl 0. Dabei wird gleichzeitig mit dem Überlauf ein Flag gesetzt. Das ist nötig, weil der µC ansonsten möglicherweise gar nicht merkt, dass seit der letzten Abfrage des Zählerstandes ein Überlauf eingetreten ist. Der µC kann das Überlaufbit abfragen oder er kann automatisch einen Interrupt auslösen lassen.
Die zweite, inzwischen ebenfalls sehr häufig anzutreffende Aktion ist die Umkehr der Zählrichtung. Nach dem Erreichen des Maximalwerts beginnt der Zähler abwärts zu zählen. Hat er dann die 0 erreicht, beginnt er wieder aufwärts zu zählen. Je nach µC kann an beiden Wendepunkten eine Flag gesetzt werden oder nur am oberen Wendepunkt. Eine dritte Möglichkeit (erheblich seltener anzutreffen) ist ein Stopp des Zählers.
Der Praktikums-µC bietet die Möglichkeiten Nullsetzen, Rückwärtszählen und Stopp.

Eine typische Anwendung ist die Zählung von Impulsen, die zu schnell hintereinander kommen oder zu kurz sind, um per Programm durch Abfrage an einem GPIO erkannt zu werden. Zudem kann sich der µC in dieser Zeit anderen Aufgaben widmen.

1.2 Taktwahl, Vorteiler

In vielen Fällen wird eine präzise Messung der Zeit oder eine regelmäßige Benachrichtigung über eine abgelaufene Zeitscheibe einer Task benötigt. Dazu wird wie in Abbildung 3 gezeigt lediglich die Taktquelle vom externen Anschluss *IN* mit einem Schalter *S* auf den ohnehin vorhandenen Systemtakt *Takt* umgeschaltet. Üblicherweise kann der Zähler auf diese Weise auch ganz angehalten werden. Hier wird das durch die Wahl einer Null mittels *S* dargestellt. Eine konstante Null liefert keine Ereignisse, also wird auch nicht gezählt.

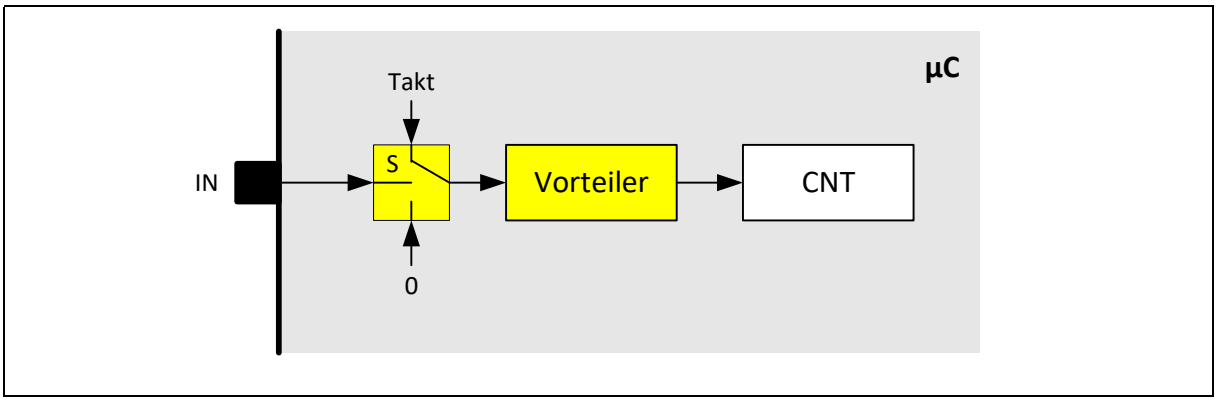

Abbildung 3: Taktquellenwahl und Vorteiler

Da der Systemtakt sehr schnell ist, können mit „kurzen“ Zählern nur kurze Zeiten gemessen werden. Angenommen, der Systemtakt beträgt wie beim Praktikums-µC 80 MHz und man möchte jede Sekunde einen Interrupt durch Überlauf. Dann muss der Zähler bis 80000000 zählen und das erfordert 27 Bit. Die Universaltimer außer TIM2 und TIM3 haben aber nur 16 Bit, man könnte also mit ihnen das Problem nicht direkt lösen.
Für solche Fälle kann einem Zähler ein Vorteiler vorangeschaltet werden. Das ist nichts anderes als ein weiterer Zähler, der fest auf die Betriebsart „Nullsetzen bei Überlauf“ eingestellt ist und für dessen Endwert *m* der Hersteller entweder einige Werte zur Auswahl vorgibt (typisch sind einige Zweierpotenzen, z.B. 2, 8, 64, 256, 1024) oder dessen Endwert auch frei einstellbar ist.
Wählt man *m=64*, dann läuft der Vorteiler alle 64 Ereignisse über. Nur jedes 64ste Ereignis am Eingang führt also zu einem Zählvorgang in *CNT*.
Diese Reihenschaltung der beiden Zähler sowie die Auswahl der Taktquelle ist in Abbildung 3 dargestellt.
Der Praktikums-µC hat 16Bit-Zähler als Vorteiler, deren Endwert frei programmierbar ist.

1.3 Zeitstempel (Capture)

Auch wenn das Programm den aktuellen Zählerstand in CNT jederzeit abfragen kann, ist es manchmal besser, diese Abfrage durch Hardware erledigen zu lassen. Das gilt speziell für zeitkritische Anwendungen. Man könnte beispielsweise mit Hilfe eines vom Systemtakt gespeisten Timers eine Entfernungsmessung mittels Licht realisieren wollen.
Man würde dazu einen Lichtpuls auslösen und gleichzeitig den Zähler starten. Trifft der reflektierte Puls wieder ein dann liest man den aktuellen Zählerstand aus und kann so die Entfernung bestimmen. Bei einer Taktfrequenz von 200 MHz entspricht aber jeder Taktimpuls einer Strecke von 1,5 m (d.h. 75 cm mehr oder weniger Entfernung zum Reflektionsort). Man muss also sofort beim Eintreffen des Ereignisses den aktuellen Zählerstand auslesen, wenn man auf eine halbwegs brauchbare Auflösung Wert legt.

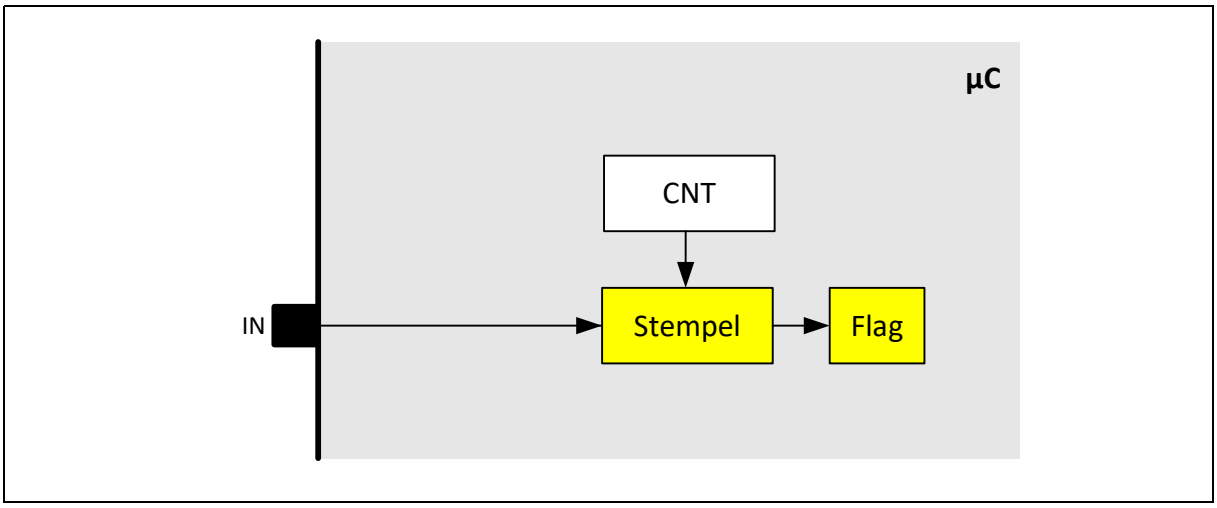

Abbildung 4: Zeitstempel (Capture)

Für derartige Anwendungen bieten Zähler sog. Capture-Register an, die beim Eintreten eines Ereignisses eine Kopie des aktuellen Zählerstands speichern. Sie wirken also wie ein Zeitstempel. In Abbildung 4 würde der empfangene Impuls am Eingang *IN* dazu führen, dass der gerade aktuelle Zählerstand im Capture-Register *Stempel* gespeichert wird. Zugleich wird ein *Flag* gesetzt, so dass das Programm auch darüber informiert wird (Abfrage oder ISR), dass ein Zeitstempel genommen wurde. Der Zähler kann weiterzählen und das Programm kann in Ruhe den geretteten Zählerstand aus dem Register *Stempel* lesen – diese Aktion ist dann nicht mehr zeitkritisch.
Viele µC haben mehrere Capture-Register, die alle an denselben Zähler angeschlossen sind. Das ist vor allem dann keine Einschränkung, wenn es um die zeitliche Zuordnung von Ereignissen ohne einen individuellen Startpunkt für die Zeitmessung geht. Der (einzige) Zähler liefert dann die globale Zeitinformation entsprechend einer ständig laufenden Uhr.

1.4 Frequenzerzeugung

Mit den bisher vorgestellten Elementen ist es nicht möglich, ein bestimmtes Zeitintervall zwischen zwei Überläufen einzustellen, da sowohl der Vorteiler als auch der Zähler eine vorgegebene Länge (in Bitstellen) haben und damit der Teiler feststeht. Man möchte aber in der Praxis eine nahezu beliebige Zeit zwischen zwei Überläufen einstellen können.

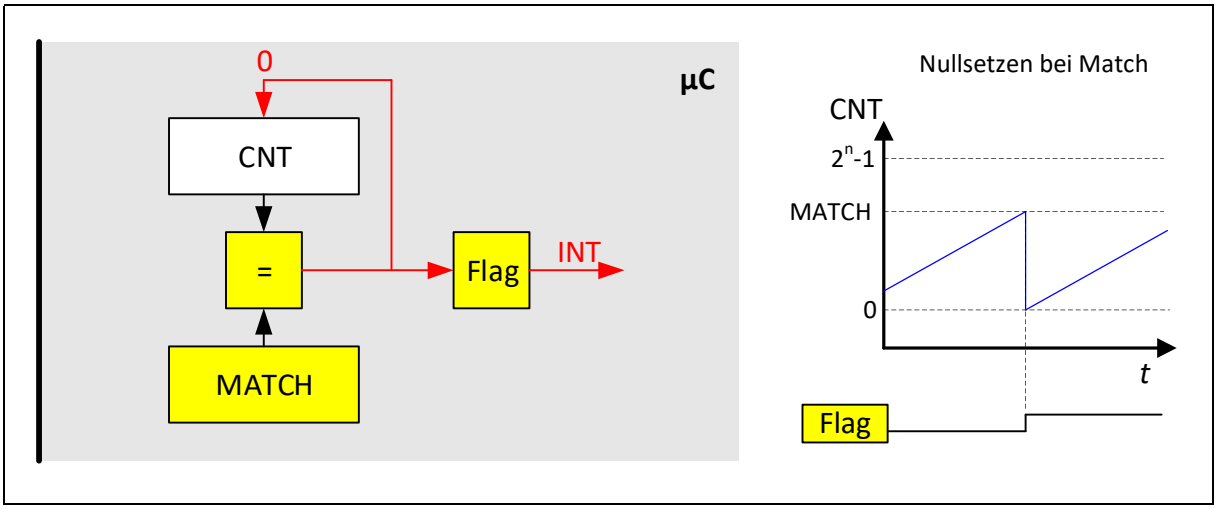

Abbildung 5: Vergleichsregister und Zeitverlauf

Die technische Lösung für diese Aufgabe ist der ständige Vergleich des aktuellen Zählerstands in CNT mit einem vom Anwender programmierbaren Wert. Wird der Wert erreicht, dann wird

ein Flag gesetzt und zusätzlich kann der Zähler auf null zurückgesetzt werden. Abbildung 5 zeigt links die Struktur der Hardware und rechts den Zeitverlauf des Zählerinhalts, wenn bei Erreichen des Vergleichswerts auf Null gesetzt wird.
Der Vergleichswert wird in sog. Match-Register geschrieben. Die meisten µC bieten mehrere Match-Register pro Zähler an. Für die hier beschriebene Aufgabenstellung genügt aber ein einziges Match-Register. Je nach µC kann auch eine andere Aktion beim Erreichen des Endwerts gewählt werden (analog Abbildung 2).
Bei dem Praktikums-µC heißt das Match-Register für diese Aufgabe „Auto-reload register“

1.5 Ausgangsschaltung

Bisher konnten zwar Ereignisse erzeugt werden, aber die Aktionen beschränkten sich auf Veränderungen am Zähler selbst oder auf das Setzen eines Flags. Man kann natürlich mittels einer ISR auf ein Flag reagieren, aber das kostet erstens Zeit und zweitens kann es ja sein, dass das Programm gerade nicht unterbrochen werden darf, aber trotzdem an einem Anschluss eine sofortige Reaktion nötig ist.

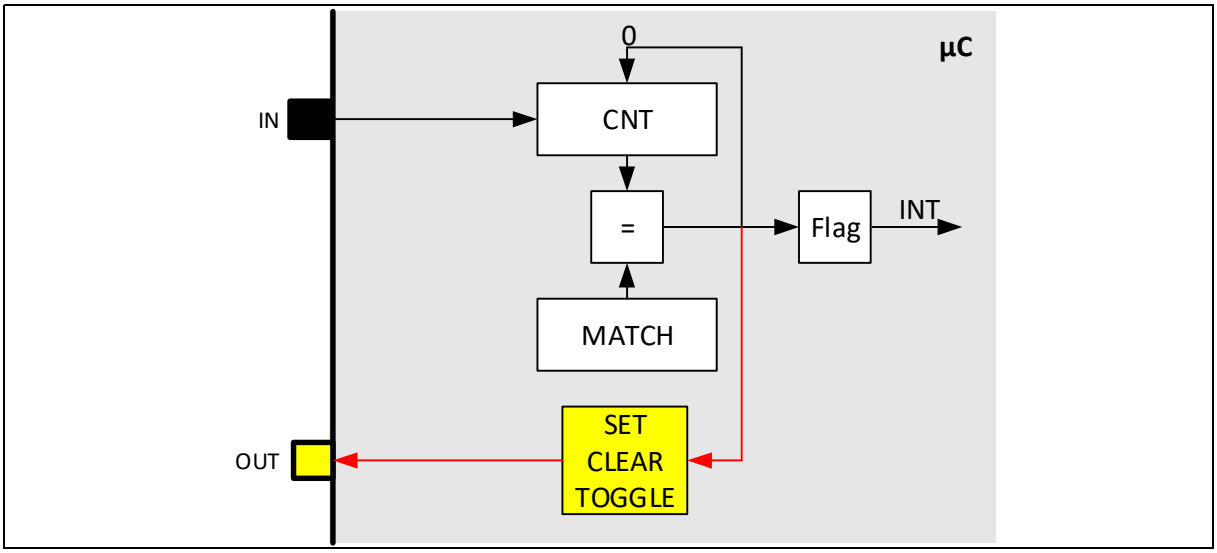

Abbildung 6: Aktion an einem Ausgang

Aus diesem Grund können praktisch alle heutigen µC auch unmittelbar einen oder mehrere Ausgänge automatisch vom Zähler beeinflussen lassen. In Abbildung 6sind drei typische Aktionen angegeben, die beim Erreichen eines Vergleichswerts an einem Pin ausgelöst werden können:

1. *SET*
   Der Ausgang wird auf H gesetzt und bleibt auf diesem Wert stehen

2. *CLEAR*
   Der Ausgang wird auf L gesetzt und bleibt auf diesem Wert stehen

3. *TOGGLE*
   Der Ausgang wechselt seinen Wert (L -> H, H -> L)

Diese Aktionen sind optional, denn es gibt auch die Möglichkeit, gar nichts zu tun (ebenso wie es auch bei den Aktionen für den Zähler die Möglichkeit gab, nichts zu tun).

Bei dem Praktikums-µC heißen die Match-Register, die Ausgangssignale erzeugen können, „Compare register“.

2 Typische Anwendungen

2.1 Pulsweitenmodulation (PWM)

Eine sehr häufige Anwendung ist die Erzeugung eines *pulsweitenmodulierten* Signals.

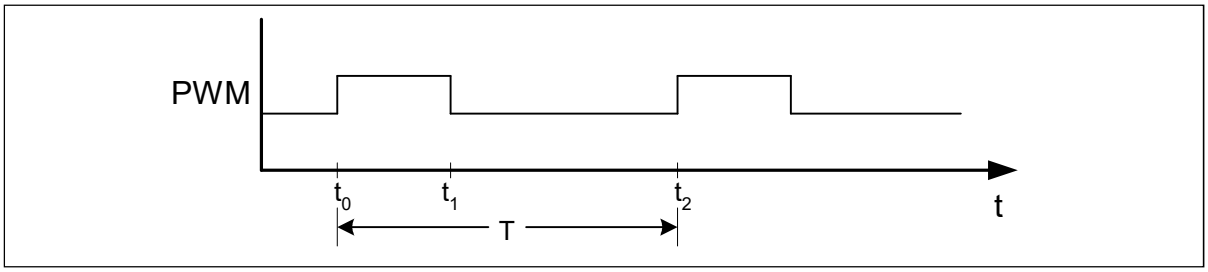

Abbildung 7: PWM-Signal

Ein PWM-Signal ist ein periodisches Rechtecksignal, bei dem der H-Anteil (der Puls) von variabler Länge ist. In Abbildung 7 ist die Frequenz des Signals durch *T* definiert und wird nicht verändert. Die Pulsweite ist durch t_1-t_0 definiert, sie kann sich in jeder Periode ändern. Woher die Änderung kommt und was sie bedeutet, hängt von der Anwendung ab. So könnte beispielsweise ein analoges Audiosignal in ein PWM-Signal umgesetzt werden, indem die Amplitude die Pulsweite bestimmt. In der Audiotechnik können mit diesem Prinzip hocheffiziente Leistungsverstärker (Class-D Verstärker) gebaut werden.

Technisch kann ein PWM-Signal mit zwei Vergleichsregistern an einem Zähler erzeugt werden (Abbildung 8). Der Vergleichswert in *MATCH0* bestimmt die Frequenz wie in Kap. 1.4 beschrieben. Als Zusatzeffekt wird dabei aber ein Ausgangssignal, hier *OUT1*, auf H gesetzt (siehe Kap. 1.5, Aktion SET).

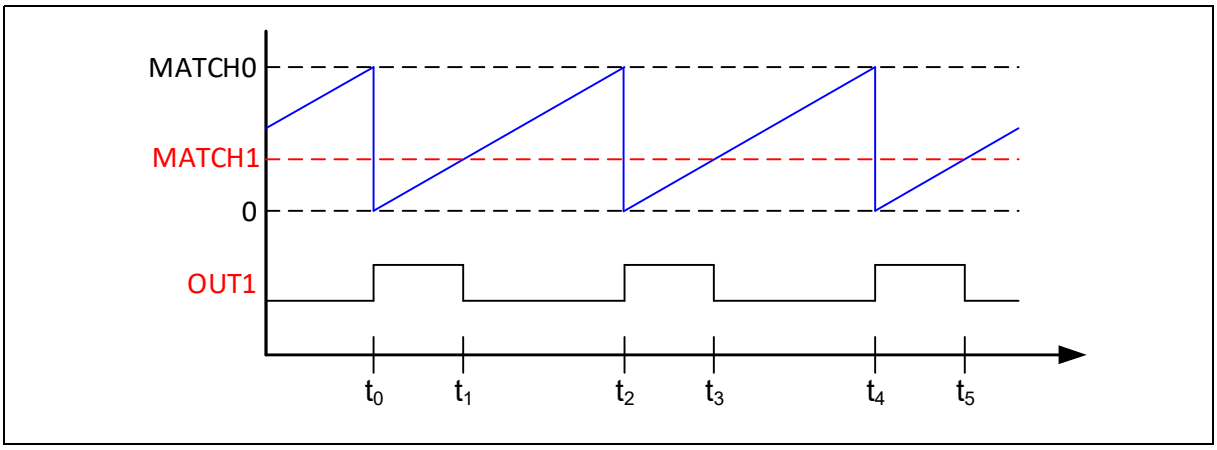

Abbildung 8: PWM-Erzeugung mit zwei Vergleichsregistern an einem Zähler

Ein zweiter Vergleichswert, hier in *MATCH1*, löst keine Aktion am Zähler aus, setzt aber den Ausgang *OUT1* wieder auf L zurück.
Möchte man die Pulsweite des PWM-Signals ändern, braucht man nur den Wert in *MATCH1* entsprechend neu zu setzen. Die übrigen Einstellungen des Zählers bleiben gleich.
Jetzt wird auch klar, warum in einem Timermodul mehrere Vergleichsregister vorhanden sind: Mit einem davon wird die Frequenz aller von diesem Modul erzeugten PWM-Signale eingestellt und die übrigen Vergleichsregister setzen jedes für sich ein zugeordnetes Ausgangssignal wieder auf L zurück. So kann man mit *n* Vergleichsregistern *n-1* PWM-Signale unterschiedlicher Pulsweite aber mit gleicher Frequenz erzeugen.

2.1.1 Anwendung Servomotor

Eine sehr beliebte Anwendung ist die Ansteuerung von Servomotoren, kurz Servo. Das sind Motoreinheiten, die eine kleine Ansteuerelektronik beinhalten, mit der der Motor eine Welle in eine bestimmte Stellung (Winkel) bringt. Der Winkel wird dabei durch ein PWM-Signal mit 50Hz Frequenz eingestellt. Der eine Endanschlag (z.B. 0 Grad) ist dann beispielsweise durch eine Pulslänge von 1ms gegeben, der andere (z.B. 180 Grad) durch eine Pulslänge von 2ms. Der Servo benötigt damit zur Ansteuerung nur eine einzige Signalleitung. Wird das Signal unterbrochen (d.h. kein oder ein ungültiges Signal am Servo vorhanden), dann bleibt der Motor stromlos. Wird die Welle (bei gegebener Ansteuerung) durch äußere Einwirkung verdreht, dann kehrt sie nach Beseitigung der Einwirkung wieder in die gewünschte Stellung zurück.

2.2 Echtzeituhr

Viele Geräte benötigen eine aktuelle Zeit (inkl. Datum) für ihre Funktion, da sie bestimmte Aktionen zu genau definierten Zeitpunkten auslösen sollen (z.B. Wecker, Alarmanlagen, Beleuchtungsanlagen). Wenn das Gerät nicht extern mit einer aktuellen Zeit versorgt werden kann, dann muss es möglichst lange eine möglichst genaue Zeitmessung durchführen. Da diese Zeitmessung auch nicht bei Stromausfall unterbrochen werden kann, ist es wichtig, dass zumindest die Zeitmessung mit minimalen Energieverbrauch in Betrieb bleiben kann, auch wenn alle anderen Einheiten des µC stillgelegt sind. Derartige Einheiten werden als Echtzeituhren (*RTC, Real Time Clock*) bezeichnet. Für sehr hohe Ansprüche an die Genauigkeit und minimalen Energieverbrauch gibt es fertige Schaltkreise. Sind die Anforderungen nicht ganz so hoch, weil beispielsweise nur Tage und Wochen, nicht aber Jahre im Energiesparbetrieb überbrückt werden müssen, dann kann ein Zeitgeber als ständig mit geringer Frequenz laufende Uhr verwendet werden.
Viele µC haben einen separaten Oszillator, der dann mit einem Quarz niedriger Frequenz betrieben wird (sog. Uhrenquarz, f=32768Hz) und im Betrieb wenige µA benötigt. An diesen Oszillator ist dann ebenfalls ein Zähler angeschlossen, der noch arbeiten kann, wenn der Rest des µC bereits schläft. Ein Überlauf oder ein voreingestellter Match weckt den µC dann wieder auf, z.B. jede Sekunde einmal.

2.3 Watchdog

Aufgrund von externen Störungen (z.B. Spannungsschwankungen) oder unerkannten Programmierfehlern ist es möglich, dass ein Gerät in einen undefinierten und nicht mehr arbeitsfähigen Zustand gerät (Programmabsturz). Da fast alle Geräte mit µC-Systemen ohne ständige Aufsicht laufen müssen, ist die Erkennung und Beendigung eines derartigen Zustands sehr wichtig. Auch dafür eignet sich ein als Zeitgeber programmierter Timer/Counter. Dieser Zeitgeber hat die spezielle Eigenschaft, dass er bei einem Überlauf den ganzen µC zurücksetzt (*Reset*) und damit einen sicheren Neustart aus jeder beliebigen Situation ermöglicht. Ein Zeitgeber mit dieser Eigenschaft heißt *Watchdog*. Nach der erstmaligen Einstellung muss der Watchdog regelmäßig rechtzeitig vor einem Überlauf zurückgesetzt werden. Bleiben diese Rücksetzaktionen aufgrund eines Programmabsturzes aus, dann wird wegen des folgenden Überlaufs der *Reset* ausgelöst. Damit der Watchdog nicht selbst während der undefinierten Aktionen bei einem Programmabsturz zufällig außer Gefecht gesetzt werden kann, sollte es mindestens sehr schwer, wenn nicht unmöglich sein, einen einmal gestarteten Watchdog wieder stillzulegen.
Der Praktikums-µC verfügt über zwei derartigen Watchdogs (IWDG und WWDG). Der IWDG wird von einem eigenen internen Oszillator gespeist und ist damit unabhängig vom Systemtakt. Wird der Watchdog einmal eingeschaltet (in der Regel beim Programmstart), ist es nicht möglich, ihn durch das Programm wieder auszuschalten. Sonst wäre es ja möglich, dass ein abgestürztes Programm den Watchdog zufällig wieder ausschaltet.