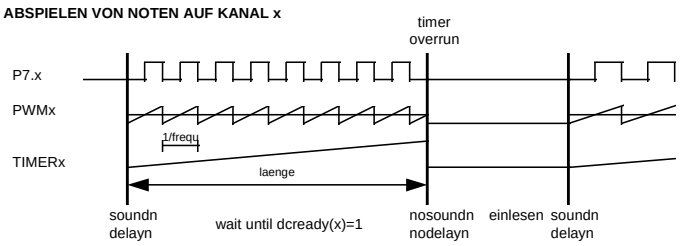
Implementierung

Die Frequenzen der einzelnen Noten werden mit Hilfe der 4-Kanal-PWM - Einheit des C167CR erzeugt. Hierbei wird während des Abspielens keine CPU-Zeit verbraucht, die PWM-Kanäle werden zu Beginn des Tones lediglich richtig initialisiert, gestartet und am Ende des Tones angehalten. Durch Änderung des Puls-Verhältnisses läßt sich auch die Klangfarbe variieren.

Die Tondauer wird durch die ersten 4 Timer-Einheiten (T0, T1, T2, T3) des C167CR festgelegt. Jeder Timer läuft für sich als sog. delay channel, und in der Interrupt Service Routine des Timers wird das globale Bit `dcx` (Delay Channel Bit) gesetzt. Die Methode `play` frägt die 4 Delay Channel Bits im Polling-Betrieb ab, und sobald ein Delay Channel „abgelaufen" ist - die Tonlänge wurde erreicht -, wird die nächste Note mit der Methode `next` eingelesen.

Die nächste Abbildung zeigt das Ausgangssignal, das Programmverhalten sowie Timer- und PWM-Einheiten:



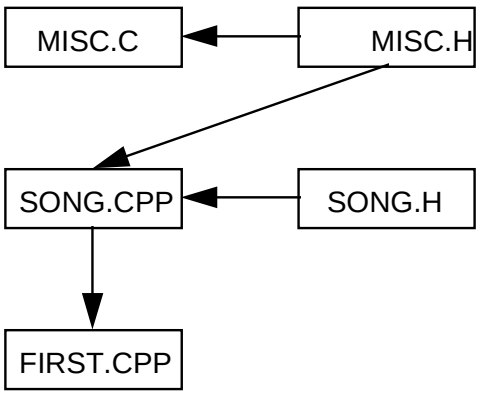
Die Funktion `soundn` setzt die Werte des PWM-Kanales x auf die Frequenz der aktuellen Note. Mit `nosoundn` wird der PWM-Kanal x deaktiviert.

Die Funktion `delayn` initialisiert den Timer x auf die in ms angegebene Notenlänge laenge. `Nodelayn` deaktiviert den Timer x.

Mit der Methode `einlesen` der Klasse `song` wird nach dem Abspielen einer Note die nächste geladen und die Werte `note`, `laenge`, `oktave`, `tempo` aktualisiert.

Implementierungsschichten

Die Applikation ist in mehrere Ebenen gegliedert: `FIRST.CPP` ist das Hauptprogramm, in dem nur die DIP-Schalterstellung abgefragt und dann das entsprechende Lied gespielt wird. In `SONG.CPP` sind die Klassen `song` und `msong` implementiert. `MISC.C` enthält alle erforderlichen Routinen zur Ansteuerung der Peripherie (dies ist in C++ direkt nicht möglich) und allgemeine Funktionen.



- Allgemeine ANSI-C Funktionen zur Ansteuerung der OnChip-Peripherie
- Klassendefinition und Implementierung von `song` und `msong`
- Hauptprogramm

Dateiliste

Allgemeine Funktionen für MC

`fehler.c`	Interruptvektoradressen für Laufzeitfehler sinnvoll beschreiben
`serio.c`	Vom Compilerhersteller mitgelieferte Datei (für `printf` erforderlich)
`serio.h`	Vom Compilerhersteller mitgelieferte Datei (für `printf` erforderlich)
`_doprint.c`	Vom Compilerhersteller mitgelieferte Datei (für `printf` erforderlich)
`cstart.asm`	Vom Compilerhersteller mitgelieferte Datei. Programm, welches vor `main()` aufgerufen wird und den MC initialisiert

MC-Programme

`first.cpp`	Hauptprogramm
`song.cpp`	Methoden - Implementierung der Klassen `song` und `msong`
`song.h`	Klassendefinition von `song` und `msong`
`misc.c`	Ansi-C Funktionen zur Ansteuerung der Peripherie
`misc.h`	Extern-Schnittstelle der allgemeinen Funktionen

Steuerdateien

`makefile`	Steuerdatei für mk166 (Generierung des ausführbaren MC-Programmes)
`cmd_link`	Kommandodatei für den Linker
`cmd_loc.e_e`	Kommandodatei für den Locater

PC-Programmdateien

`term.cpp`	Source-Code für PC-Applikation (Visualisierung der Spieldauer)
`term.exe`	Ausführbare PC-Applikation

Dokumentation

`oop.doc`	Diese Datei (im Winword 6.0 - Format)
`timer.ppt`	Zeitdiagramm
`dateien.ppt`	Implementierungsschichten

HOTLINE *„Ford Hotline, was kann ich für Sie tun?"*
KUNDE *„Ihre Autos sind Mist!"*
HOTLINE *„Was ist passiert?"*
KUNDE *„Es ist kaputt, das ist passiert!"*
HOTLINE *„Was genau haben Sie getan?"*
KUNDE *„Ich wollte schneller fahren. Also habe ich das Gaspedal bis zum Boden durchgedrückt. Eine Weile ging alles gut aber jetzt ist der Wagen kaputt und läßt sich nicht mehr starten!"*
HOTLINE *„Es liegt in Ihrer Verantwortung wenn Sie unser Produkt mißbrauchen. Was erwarten Sie nun von uns?"*
KUNDE *„Ich verlange, daß Sie mir das neueste Modell schicken, das nicht mehr kaputt geht!"*