Aufgabe 3 — Verarbeitung digitaler Signale

Lernziel: Programmieren mit Zeiten

Unterlagen: Ada Language Reference Manual
Vorlesungsunterlagen

Aufgabenstellung

Entwickelt einen einfachen Dekodierer für Morsezeichen. Das Programm muß mittels zweier nebenläufiger Tasks realisiert werden.

Es steht wieder die Endabschalter-Textbox zur Verfügung, die an die Echtzeitrechner angeschlossen ist. Dort, wo in Aufgabe 1 der Endabschalter 0 abzufragen war, ist jetzt ein elektronischer Morsegeber angeschlossen. (Bitte beachten: Der Morsegeber ist parallel zu dem Endabschalter angeschlossen. Damit nicht ein Dauerträger empfangen wird, muß Schalter 0 offen sein, also nach oben geschaltet. Ferner ist die Polarität des Anschlusses zu beachten: Der rote Stecker kommt in die rote Buchse, der blaue Stecker in die blaue Buchse) Die Lösung der Aufgabe 1 kann zum Erkennen der Morsetaste nach einigen Veränderungen weiterverwendet werden.

Im Verzeichnis `/home/pdv/pdv/lehre/ees/zu_a3` geben wir in der Datei `morse.txt` die Morsezeichen vor. Wer will, kann zur Zeichendefinition und Ausgabe der Zeichen die Vorgabe aus `morse_def.a` verwenden, es können aber auch eigene Routinen entwickelt werden. Ferner wird die Bibliothek `Calendar` benötigt, also sollte in Eurem Programm irgendwo `with Calendar; use Calendar;` stehen. `morse_def.a` liegt nicht in der `eeslib/` als Bibliothek vor, sondern muß in Euer Verzeichnis kopiert und dort kompiliert werden, wenn es verwendet werden soll.

Der Dekodierer soll mittels **zweier Tasks** realisiert werden: Eine Task `lesen` soll den Schalterzustand überwachen und registrieren, wie lange die Morsetaste gedrückt bzw. offen ist. Sobald eine Pause registriert wird, die dem Abstand zwischen zwei Zeichen (Buchstaben, Satzzeichen, Verkehrszeichen) entspricht, sollen die Zeiten (entsprechen den Punkten, Strichen und Pausen) einer zweiten Task `auswerten` übergeben werden, die dann ermittelt, welches Zeichen gelesen wurde.

Damit der Dekodierer mit verschiedenen Gebegeschwindigkeiten zurechtkommt, muß er zunächst kalibriert werden. Hierzu sind eine Reihe „v“ (`...-`) zu geben. Aus dem Kalibriervorgang resultieren dann die Werte für die Länge der Punkte bzw. Striche. Nach dem Kalibrieren sollen dann die gemorsten Zeichen entschlüsselt und auf dem Monitor ausgegeben werden. Zeichen, die nicht erkannt wurden, sollen mit einem Bindestrich - dargestellt werden. Macht der Funker eine Pause, die länger als die normale Pause zwischen zwei Zeichen ist, so soll einmal ein Freizeichen (*space*) ausgegeben werden. Das Kalibrieren kann in der Task `lesen` erfolgen.

Folgende weitere Randbedingungen sollen gelten: Das Programm soll terminieren, wenn ein Dauerträger von 60 Sekunden gegeben wird oder der Funker mehr als 60 Sekunden Pause macht. Ferner soll das Programm terminieren, wenn $\overline{\text{SK}}$ (`...-.-`) oder $\overline{\text{AR}}$(`.-.-.`)

gegeben wurde. Bei diesen Zeichen handelt es sich wie bei den Zeichen $\overline{\text{KN}}$, $\overline{\text{DN}}$, $\overline{\text{BT}}$, $\overline{\text{AAA}}$, $\overline{\text{MIM}}$, $\overline{\text{IMI}}$ und $\overline{\text{ERR}}$ um Satz- bzw. Verkehrszeichen, die aus mehreren Buchstaben zusammengesetzt sind. Zur besseren Unterscheidung werden diese Zeichen mit einem Überstrich versehen, wenn die Darstellung in Buchstaben gewählt wird. Für diese Aufgabe ist dies aber nicht möglich, bitte verwendet die ganz normale Darstellung `/ . , ?`, $\overline{\text{ERR}}$ wird als ~ geschrieben.

Zur Theorie der Morsezeichen: Die Basiseinheit ist der Punkt, daraus wird das Verhältnis Punkt : Strich : Elementabstand : Zeichenabstand : Wortabstand mit 1 : 3 : 1 : 3 : 7 abgeleitet. Allerdings ist manuelles Geben mit einer klassischen Taste nicht allzu genau, daher sind Toleranzen vorzusehen, z.B. kann der Punkt durchaus auch mal 1,5 Punkte lang werden, der Strich aber nur 2,5 Punkte lang sein. Ebenso kann die Länge der Pausen zwischen den Elementen etwas schwanken.

Damit Euer Dekodierer nicht zu empfindlich auf schwankende Gebegeschwindigkeiten reagiert, sind Fristen für die Punkte bzw. Striche vorzusehen. Im einfachsten Fall und bei niedrigen Gebegeschwindigkeiten kann es ausreichen, Punkte und Striche zu unterscheiden, indem alle Werte kleiner einer Grenze als Punkte und alle Zeichen größer dieser Grenze als Striche interpretiert werden. Beachtet hierbei, daß die Auflösung der Zeit, die bei Verwendung der Standarduhr erreicht werden kann, bei 0,01 Sekunden liegt.

agbp es vy 73

(Gruß der Morse-Funker: Always good brass pounding and with very best compliments)

Hinweise zum elektronischen Morsegeber:

Groß- und Kleinschreibung werden nicht unterschieden. Es können die Ziffern 0 bis 9, die Buchstaben a-z, ä, ö, und ü sowie die Satzzeichen Komma, Punkt, Trennung, Schrägstrich und Fragezeichen auf der Tastatur eingegeben werden. Das Fragezeichen ist ohne SHIFT erreichbar, der Schrägstrich liegt auf dem + (also rechts neben dem ü). Ferner sind die Tasten F1 mit „vvv“ und die F2 mit „paris“ belegt. Tasten, die der Morsegeber nicht interpretiert, erzeugen keine Ausgaben und keine Fehler. Mit ALT-A wird $\overline{\text{AR}}$, mit ALT-S wird $\overline{\text{SK}}$ eingegeben.

Der Morsegeber puffert einige Zeichen, sollte an der Tastatur schneller eingegeben werden als gemorst wird. Mit dem Tabulator (TAB) kann die Ausgabe jederzeit abgebrochen werden.

Der Morsegeber verfügt über 2 Geschwindigkeiten, die mit der ENTER-Taste umgeschaltet werden. Bei Geschwindigkeit 2 leuchtet die CAPS-LOCK-LED der Tastatur. Geschwindigkeit 1 ist langsam, Geschwindigkeit 2 ist schnell. Euer Programm soll beide Geschwindigkeiten beherrschen, wobei ein Neustart des Programms erlaubt ist.

Die DEL-Taste erzeugt einen Dauerton, zum Abschalten noch einmal DEL drücken. Die ESC-Taste wird **nicht** benutzt.

Durch Aus- und Wiederanschalten erfolgt ein Reset des Morsegebers auf seine Standardwerte. Bitte geht pfleglich mit den Morsegebern um!