U2 Implementierung RC-Tiefpassfunktion

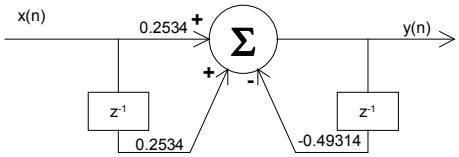
Umfeld

Im Rahmen eines Praxisbeispiels wurde die analoge Übertragungsfunktion *G(s)* eines RC-Tiefpass 1. Ordnung mit fc=1kHz mittels Bilinearer z-Transformation in die frequenzdiskrete Übertragungsfunktion *G(z)* übergeführt und als IIR-Filter realisiert werden:

$$G(s) = \frac{6283.185}{s + 6283.185} \qquad \text{(analog)} \qquad f_S = 9.6kHz$$

$$G(s) = \frac{6517.52}{s + 6517.52} \qquad \text{(analog,prewarped)}$$

$$G(z) = G(s)\Big|_{s=\frac{2}{T_S}\frac{z-1}{z+1}} = \frac{0.2534(z+1)}{z - 0.49314} = \frac{0.2534 + 0.2534z^{-1}}{1 - 0.49314z^{-1}}$$



IIR-Filter 1. Ordnung in direkter Form.

Das Filter auf dem DSP56002 EVM implementiert und untersucht werden. Dazu ist die Übertragungsfunktion als IIR-Filter in direkter Form, d.h. mit zwei Verzögerungsgliedern, zu codieren und zu testen.

Aufgaben

1. Aufsetzen eines neuen, leeren C-Projektes (z.B. Uebung2_RC-Tiefpass, keine Leerschläge erlaubt) auf der Grundlage des Prototypen-Files. Kommentieren und Anpassen des Prototypen.

2. Implementieren der Übertragungsfunktion durch Definition der Filterfunktion, die die eigentliche Übertragungsfunktion aufnimmt:

```
_fract filterRC(_fact x)
{ static _fract z1;    //  Zeitverzoegerung(en) definieren
  ....
  return ...    // Ausgangssignal y retournieren
}
```

   Hinweise:
   Benutzen Sie zur Rechnung den Datentyp fract. Er verkörpert den Festkommadatentyp des DSP56002 Signalprozessors mit dem Wertebereich [-1,1). Zu beachten ist, dass der Codec mit 9.6kHz Samplerate initialisiert wird. Dazu ist die symbolische Konstante `SAMPLE_RATE_9` aus `CodecConst.h` zu benutzen.

3. Austesten mit Musik, evtl. ausmessen des Amplitudenganges mit NF-Voltmeter.

Als Hilfe erhalten Sie eine mögliche Lösung, jedoch mit globalen Variablen realisiert: