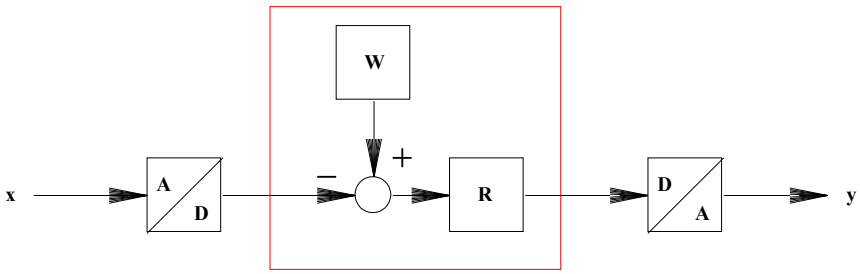
Digitalregler

1) Allgemeines

Während früher Regler mit Operationsverstärkern aufgebaut waren, findet man heute überwiegend Digitalregler. Der Vorteil ist, dass die Eigenschaften des Reglers durch den Regelalgorithmus, d.h. die Software in einem Mikrocontroller, festgelegt werden. So ergibt sich eine große Flexibilität, und es lassen sich jenseits der klassischen P-, I- und D-Regler noch andere Regler definieren.
Es können auch ohne Probleme mehrere Regelgrößen erfasst und geregelt werden.
Da die Informationen in digitaler Form vorliegen, können sie an eine Zentrale übermittelt und gespeichert werden.

2) Struktur des Reglers

Prinzip:



- Die analoge Regelgröße X wird von einem AD-Wandler (Analog-Digital-Wandler) in einen Zahlenwert gewandelt.
- Der Sollwert W ist als Zahlenwert in der Software des Mikrocontrollers (Regelalgorithmus) festgelegt.
- Die Regeldifferenz E = W – X wird als Zahlenwert berechnet.
- Aus der Regeldifferenz E wird der Zahlenwert der Stellgröße Y berechnet. Diese Berechnung erfolgt unterschiedlich je nachdem welcher Reglertyp gewünscht wird. Es muss dabei eventuell auf vorher abgespeicherte Werte von X bzw. Y zurückgegriffen werden.
- Ein DA-Wandler (Digital-Analog-Wandler) wandelt den Zahlenwert in ein analoges Signal, die Stellgröße Y.

Der beschriebene Vorgang muss zyklisch mit einer genau definierten Zykluszeit T_0 ablaufen.

Im einfachsten Fall besteht die Software für den Regler aus einer Schleife, in der der Reihe nach Unterprogramme für die beschriebenen Schritte aufgerufen werden.

Erweiterungen:

- Soll der Sollwert extern an einem Potentiometer eingestellt werden, kann er über einen zweiten AD-Wandler eingelesen werden.
  Der Sollwert kann auch über eine serielle Schnittstelle von einem PC oder einem anderen Controller vorgegeben werden.
  Auch eine manuelle Einstellung mit Tipptasten ist möglich.

- Es können mit einem Mikrocontroller mehrere Regelgrößen verarbeitet werden, wenn man die Software entsprechend erweitert und mehrere AD- und DA-Wandler benutzt.

Mit modernen Controllern lassen sich Regler aufbauen, die ein quasi-analoges Verhalten haben. Es ergeben sich aber einige Einschränkungen durch AD- und DA-Wandlung, und durch die Zykluszeit, die man kennen muss, um die Regler richtig anzuwenden.

3) AD- Wandler

a) Einfluss der Auflösung des AD-Wandlers

Die analoge Regelgröße wird in einen wertediskreten Zahlenwert umgewandelt.
Es gibt also nicht mehr unendlich viele, sondern nur eine endliche Zahl von möglichen Werten beim Ergebnis.

Hierdurch ist die **Auflösung** des Reglers begrenzt.

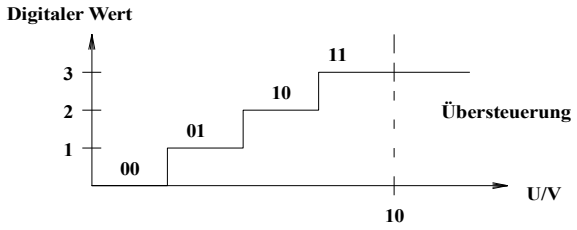
Beispiel mit einem (unrealistischen) 2-Bit- Wandler:

2 Bit $\rightarrow$ $2^2 = 4$ Zustände, d.h. es gibt die Zahlenwerte 0, 1, 2, 3

Allgemein:

n bit $\rightarrow$ 2^n Zustände:	0, 1, 2, ..., 2^n-1

Wenn die Eingangsspannung ein Normsignal mit 0...10V sein soll, ergibt sich die folgende Zuordnung:



Die Auflösung für unser Beispiel wäre miserabel, nämlich 10V/4 = 2.5V.

Allgemein:

Auflösung $\Delta X = \frac{Xmax}{2^n}$

Aufgabe DigR 1

a) *Ein ATMega8-Kontroller enthält AD-Wandler mit 10 Bit Auflösung.Es wird die interne Referenzspannung von 2.56V benutzt. Welcher Zahlenbereich wird beim Digitalisieren benutzt? Welche Auflösung ergibt sich?*

b) *Wie kann der vom AD-Wandler gelesene Wert Xdig in einen Spannungswert X (0 ... 2,56) umgerechnet werden?*

c) *Wie kann der Eingangsspannungsbereich auf 10V erweitert werden? Welche Auflösung ergibt sich dann?*

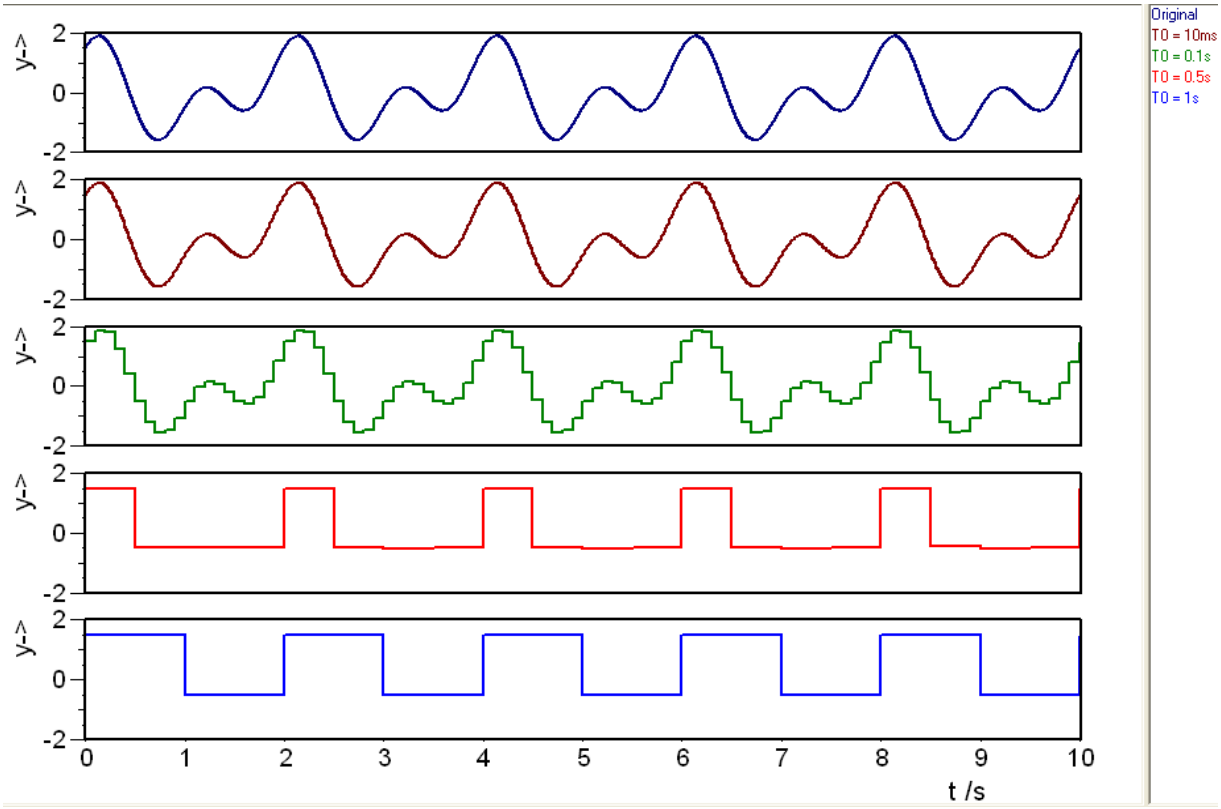
b) Einfluss der Abtastzeit des AD-Wandlers

Der AD-Wandler nimmt zu bestimmten Momenten kT_0 eine Probe des Signals (Sample and Hold). Dieser Wert wird dann in einen Digitalwert umgewandelt.

Wie kurz die Abtastzeit mindestens gewählt werden muss, sagt uns das **Theorem von Shannon:**

Um keine unzulässige Verfälschung des Signals zu bekommen, müssen wir mit mindestens dem doppelten der höchsten im Signal enthaltenen Frequenz abtasten, also mindestens 2x pro Periode des höchsten Frequenzanteils.

Beispiel:
Analoges Originalsignal mit Frequenzanteilen von 1Hz und 0.5Hz, digitalisiert mit verschiedenen Abtastzeiten.



Nach Shannon sollten wir mit mindestens 2Hz, also $T_{0min} = 0.5s$ abtasten. Das Ergebnis dieser Minimalforderung ist aber recht kläglich, während noch längere Abtastzeiten schlimme Fehler produzieren.
Die oberen beiden Kurven mit kurzen Abtastzeiten stellen gute Näherungen des Originals dar.

Bemerkung:
Im Gegensatz zur Darstellung in vielen theoretischeren Abhandlungen wird hier der abgetastete Wert als gespeichert (bis zur nächsten Abtastung konstant) dargestellt. Dies ist gerechtfertigt, denn der Wert steht ja dem Mikrocontroller während dieser Zeit zur Verfügung.

Aufgabe DigR 2
Welche Abtastfrequenz ist zum Abtasten eines Audiosignals (20Hz ...20kHz) nötig?

c) Der AD-Wandler eines Mega8-Controllers in BASCOM

Zunächst muss der AD-Wandler initialisiert werden:

```
'AD-Wandler mit Uref = +5V
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
```

Dann können die Werte gelesen und umgerechnet werden. Beim Mega8 gibt es 6 Kanäle für den AD-Wandler, so dass man problemlos z.B. Kanal 0 für den Istwert und Kanal 1 für den Sollwert nehmen kann.

Ein Unterprogramm zum Auslesen des Istwertes sähe dann so aus:

```
Get_x:
      Wtmp = Getadc(0)
      X = Wtmp / 10.23                  'in Wert 0...100% umrechnen
Return
```

Zunächst wird der 10bit-Wert des AD-Wandlers, Kanal 0, in eine temporäre Word-Variable (16 Bit, es gibt keine 10 bit – Variablen) gelesen. Dieser Wert dann in den Wert X der Regelgröße umgerechnet, welcher in einer Variable vom Typ Single (Fliesskomma) steht.

Selbstverständlich muss im Initialisierungsteil des Programmes Speicherplatz für alle benutzten Variablen reserviert werden.

3) DA- Wandler

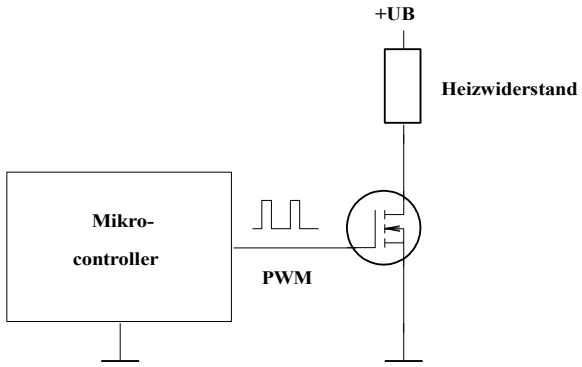
a) Allgemeines

Für den DA-Wandler am Ausgang gelten ähnliche Überlegungen wie für den AD-Wandler am Eingang.
Die Stellgröße kann nur 2^n unterschiedliche Werte annehmen, wenn eine Auflösung von n Bit benutzt wird.
In der Praxis ist oft eine 8 Bit-Wandlung (256 Stufen) ausreichend.

b) Die DA-Wandlung mit einem AVR-Controller

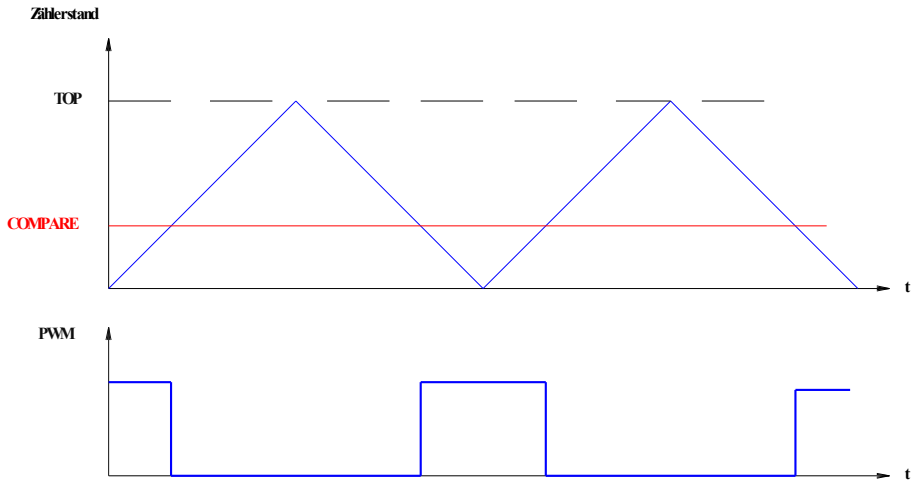
Leider enthalten die AVR-Controller zwar einen AD-Wandler, aber keinen DA-Wandler.

Für viele Anwendungen ist es aber gar nicht nötig, einen separaten DA-Wandler mit reinem Analog Ausgang zu nehmen. Die meisten Mikrocontroller bieten die Möglichkeit, ein PWM-Signal auszugeben. Wenn die Frequenz des PWM hoch genug gewählt wird, lassen sich viele Verbraucher über einen MOSFET direkt damit ansteuern. Wegen der Trägheit der üblichen Regelstrecken merkt man nicht, dass eigentlich ein digitales Signal benutzt wird.
Der Wirkungsgrad ist hoch, da nur ein- und ausgeschaltet wird.



Wenn man doch ein analoges Signal für die Stellgröße braucht, kann man ein PWM-Signal mit einem Tiefpass glätten oder ein DA-Wandler-IC benutzen.

Besonders elegant kann das PWM-Signal erzeugt werden, indem man das Output Compare Register des Timers benutzt. Hierbei zählt der Hardware-Timer des Controllers laufend hoch und runter. Der Zählerstand wird mit dem Inhalt des Compare-Registers verglichen und der PWM-Ausgang dementsprechend gesetzt oder rückgesetzt:



Durch Verändern des Compare-Wertes kann die Impulsdauer länger oder kürzer gemacht werden.

Der ganze Vorgang läuft automatisch im Hintergrund ab, sobald der Timer konfiguriert und der Compare-Wert gesetzt ist.

In BASCOM für den Timer1 sieht die Initialisierung z.B. so aus:

```
Config Timer1 = Pwm , Prescale = 1 , Pwm = 8 , Compare A Pwm = Clear Down
```

Im Beispiel wird ein nichtinvertierendes 8-Bit-PWM benutzt (es hat also 256 unterschiedliche Werte, d.h. der Compare-Wert darf von 0 bis 255 geändert werden, entsprechend ändert sich dann die Impulsdauer.

Der Compare-Wert wird einfach gesetzt:

```
Pwm1a = 30
```

ergibt z.B. eine Impulsdauer von 30/256 = 11.7%.

4) Zykluszeit

Ein Zyklus besteht aus 3 Teilen:

- Der AD-Wandler nimmt zu bestimmten Momenten kT_0 eine Probe des Signals (Sample and Hold). Dieser Wert wird dann in einen Digitalwert umgewandelt.
- Aus diesem Digitalwert errechnet der Regelalgorithmus den digitalen Wert für die Stellgröße. Die Berechnung dauert eine gewisse Zeit T_R.
- Der DA-Wandler gibt den Wert der Stellgröße aus, zum Zeitpunkt $kT_0 + T_R$

Wenn die Rechenzeit kurz ist kann sie vernachlässigt werden.

Damit der Regler auch auf schnelle Änderungen der Regelgröße reagieren kann, muss die Zykluszeit so klein wie möglich gewählt werden (Shannon).
Andererseits benötigt der Regelalgorithmus eine gewisse Zeit für die Berechnungen. Diese muss ihm auf jeden Fall zur Verfügung gestellt werden, damit das Programm nicht „hängen bleibt“.

Da viele Regelstrecken träge sind, genügt in der Praxis oft eine „gemächliche“ Zykluszeit im Bereich 1...100ms.

Am besten benutzt man einen Timer um zyklisch per Interrupt das Unterprogramm für den Regler aufzurufen. Dann besteht das Hauptprogramm aus einer leeren Schleife, oder es kann sonst was tun.

```
...
'Timer2 gibt Zykluszeit vor.
'2ms-Interrupt mit Timer2 Output Compare
'CLK = 16MHz -> TCLK=0.0625us
'Prescaler = 128 - > Tclk2 = 128*0.0625us = 8us
'OC2 = 2000us/8us = 250
Config Timer2 = Timer , Prescale = 128 , Clear Timer = 1 , Compare = Disconnect
Ocr2 = 250
Enable Oc2
Enable Interrupts
On Oc2 Control
....

'*****************************************************************************
' HAUPTPROGRAMM (leer )
Do

Loop
'*****************************************************************************
'-----------------------------------------------------------------------------
Control:
'Regelalgorithmus
   Gosub Get_w                          'Sollwert lesen
   Gosub Get_x                          'Istwert lesen
   E = W - X                            'Regeldifferenz
   Gosub P_control                      'Regelalgorithmus
   Gosub Output_y                       'Stellgröße ausgeben
Return
```

Es werden hier Unterprogramme benutzt um das Regelprogramm übersichtlich zu halten. Bei einer Änderung, zum Beispiel auf einen PI-, PD- oder PID-Regler, braucht nur das betreffende Unterprogramm geändert zu werden.

5) Regelalgorithmus für einen P-Regler

Für einen analogen P-Regler gilt: $y = K_{PR} \cdot e$

Wenn man daran erinnern will, dass y und e zeitlich veränderliche Funktionen sind, kann man schreiben:

$$y(t) = K_{PR} \cdot e(t)$$

Beim digitalen Regler gibt es keine zeitkontinuierlichen Funktionen mehr, die Werte existieren nur zu bestimmten Vielfachen kT_0 der Abtastzeit.

Dies kann man dadurch zum Ausdruck bringen, dass man t = kT0 in die obige Formel einsetzt:

$$y(kT_0) = K_{PR} \cdot e(kT_0)$$

Da die Abtastzeit T_0 eine Konstante ist, wird oft eine einfachere Schreibweise benutzt, bei der die Werte von e bzw. y einfach durch nummeriert werden.

So ist e(k) der k-te Abtastwert der Regeldifferenz.

Also: $y(k) = K_{PR} \cdot e(k)$

Auch die Schreibweise $y_k = K_{PR} \cdot e_k$ ist üblich.

Der Regelalgorithmus besteht einfach darin, die Regeldifferenz zu rechnen, sie mit KPR zu multiplizieren und als Stellgröße auszugeben.

6) Einfacher P-Regler in BASCOM

```
'P-Regler
'Eingang Istwert: ADC0
'Eingang Sollwert: ADC1
'Ausgang: PWM mit Timer1 an OC1A = PB1

'Intern werden alle Werte als normierte Single 0...100 (%) gehandhabt

'Abtastzeit 2ms

'-----------------------------------------------------------------------------
$regfile = "m8def.dat"
$crystal = 16000000


'Timer2 gibt Zykluszeit vor.
'2ms-Interrupt mit Timer2 Output Compare
'CLK = 16MHz -> TCLK=0.0625us
'Prescaler = 128 - > Tclk2 = 128*0.0625us = 8us
'OC2 = 2000us/8us = 250
Config Timer2 = Timer , Prescale = 128 , Clear Timer = 1 , Compare = Disconnect
Ocr2 = 250
Enable Oc2
Enable Interrupts
On Oc2 Control

'Timer1 als PWM mit
'  Prescale=1 -> fCTR = FXTAL = 16MHz , -> fPWMZähler = 16MHz/256 = 62.5kHz
'  ergibt 31.25kHz PWM-Frequenz da 1x rauf + 1x runter zählen
'  PWM = 8  -> 8bit PWM    (möglich: 8/9/10 bit)
Config Timer1 = Pwm , Prescale = 1 , Pwm = 8 , Compare A Pwm = Clear Down
Pwm1a = 0

'AD-Wandler mit Uref = +5V
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
```

```
'Variablen für Regelung
Dim W As Single                         'Sollwert
Dim Kpr As Single                       'Proportionalbeiwert
Dim E As Single                         'Regeldifferenz
Dim X As Single                         'Regelgrösse
Dim Y As Single                         'Stellgrösse


'temporäre Variablen (Single, Byte, Word):
Dim Stmp As Single
Dim Btmp As Byte
Dim Wtmp As Word


'--------------------------------------------------------------------------

'KPR -Wert:
Kpr = 10

'**************************************************************************
' HAUPTPROGRAMM (leer )

Do

Loop
'**************************************************************************
'--------------------------------------------------------------------------
Control:
'Regelalgorithmus

   Gosub Get_w                          'Sollwert lesen
   Gosub Get_x                          'Istwert lesen
   E = W - X                            'Regeldifferenz
   Gosub P_control                      'Regelalgorithmus
   Gosub Output_y                       'Stellgrösse ausgeben

Return

'--------------------------------------------------------------------------
Get_x:
      Wtmp = Getadc(0)
      X = Wtmp / 10.23                  'in Wert 0...100% umrechnen
Return

'--------------------------------------------------------------------------
Get_w:
      Wtmp = Getadc(1)
      W = Wtmp / 10.23
Return

'--------------------------------------------------------------------------
P_control:
      Y = Kpr * E
Return

'--------------------------------------------------------------------------
Output_y:
      'Begrenzung
           If Y > 100 Then Y = 100
           If Y < 0 Then Y = 0

      'Ausgabe als PWM-Wert
           Btmp = Y * 2.55              'über Byte-Zwischenvariable
           Pwm1a = Btmp
Return
'--------------------------------------------------------------------------
```

Wichtig ist die Begrenzung im Unterprogramm Output_y, da der PWM-Wert nur von 0 bis 255 gehen darf.

Bemerkung:
Das Programm für einen P-Regler könnte wesentlich einfacher gestaltet werden (ohne Timer-Interrupt), indem man in einer Schleife des Hauptprogramms der Reihe nach die nötigen Unterprogramme aufruft:

```
' HAUPTPROGRAMM

Do
   Gosub Get_w                          'Sollwert lesen
   Gosub Get_x                          'Istwert lesen
   E = W - X                            'Regeldifferenz
   Gosub P_control                      'Regelalgorithmus
   Gosub Output_y                       'Stellgrösse ausgeben
Loop
```

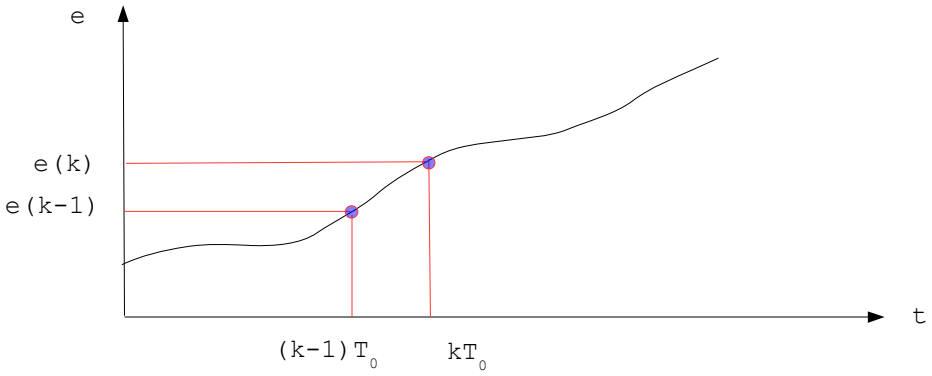
Das Problem ist aber dann, dass man keine Kontrolle über das Timing hat, bzw. diese nur umständlich zu erreichen wäre.
Für einen P-Regler ist dies völlig unwichtig.
Aber wir wollen die vorgestellte Programm-Strunktur später für einen PID-Regler benutzen, und bei diesem ist eine feste, bekannte Zykluszeit T_0 sehr wichtig.

7) Regelalgorithmus für ein D-Glied

Für einen analogen D-Regler gilt: $y = T_D \cdot \frac{de}{dt}$

Angenähert bedeutet dies $y = T_D \cdot \frac{\Delta e}{\Delta t}$

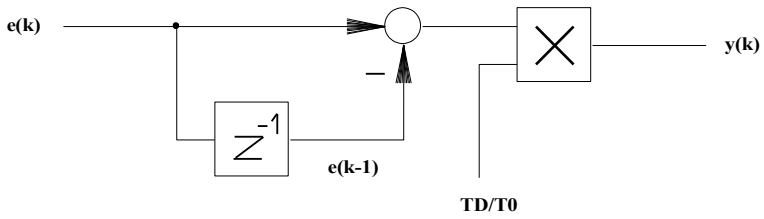
Beim digitalen Regler wird aus der unendlich kleinen Zeit dt die kleinstmöglich verfügbare Zeit, nämlich die Abtastzeit T_0.



$$y(k) = T_D \cdot \frac{e(k) - e(k-1)}{T_0}$$

$$y(k) = \frac{T_D}{T_0} \cdot [e(k) - e(k-1)]$$

Im Signalflussplan kann der Algorithmus so dargestellt werden:



Dabei bedeutet der z^{-1} – Block eine Verzögerung um einen Takt.
Dies entspricht der Verwendung eines abgespeicherten Wertes e(k-1)

Aufgabe DigR 3
Realisiere ein D-Glied mit T_D = 5s in BORIS nach obigem Signalflussplan.
Wähle die Abtastzeit zu 0.1s, ebenso die Simulationsschrittweite (dies macht Sinn, denn der Regler berechnet ja nur jedes T_0 einen neuen Wert) .

Überprüfe die Funktion indem du die Anstiegsantwort aufnimmst, z.B. mit einem Signal mit einer Anstiegsgeschwindigkeit von 1V/s.

Vergleiche mit dem gerechneten Wert der Ausgangsspannung.

8) Regelalgorithmus für einen I-Regler

Für einen analogen I-Regler gilt: $$y = \frac{1}{T_I} \cdot \int e\, dt$$

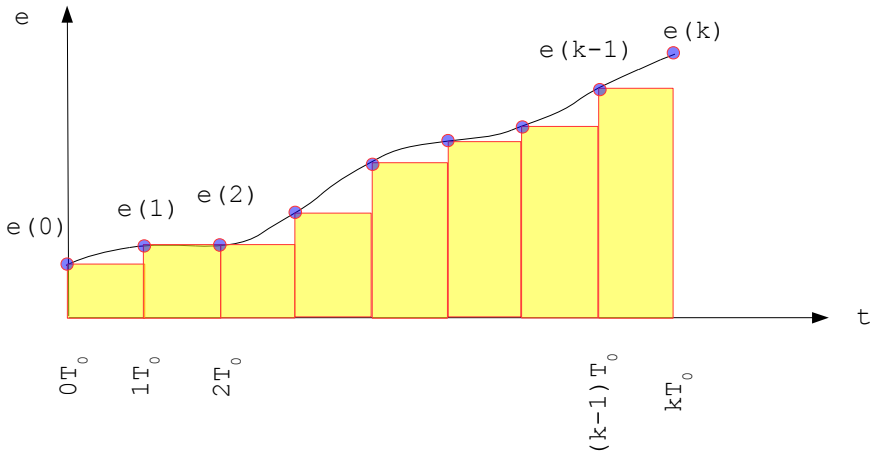
Genauer betrachtet integrieren wir vom Einschaltmoment t=0 bis zum aktuellen Zeitpunkt t:

$$y = \frac{1}{T_I} \cdot \int_0^t e\, d\tau$$

Die Integrationsvariable wurde hier in τ umbenannt damit nicht zweimal t in unterschiedlicher Bedeutung auftaucht.

Das Integral ist anschaulich die Fläche unter der e-Kurve, vom Zeitpunkt t = 0 bis zum Zeitpunkt t.

Diese Fläche kann beim Digitalregler durch eine Summe von Rechteckflächen angenähert werden:



$$y(k) = \frac{1}{T_I} \cdot [e(0)\,T_0 + e(1)\,T_0 + ... + e(k-2)\,T_0 + e(k-1)\,T_0]$$

$$y(k) = \frac{T_0}{T_I} \cdot [e(0) + e(1) + ... + e(k-2) + e(k-1)]$$

$$y(k) = \frac{T_0}{T_I} \cdot \sum_{i=0}^{k-1} e(i)$$

Bei unserer Herleitung wurden Rechtecke benutzt mit einer Höhe e(0), e(1), ...e(k-1), das Ergebnis ist im Beispiel eine etwas zu kleine Fläche. Genausogut hätte man Rechtecke mit einer Höhe von e(1), e(2), ...e(k-1), e(k) benutzen können, in diesem Fall wäre das Ergebnis ein wenig zu groß gewesen. Es gibt auch die Möglichkeit, mit Trapezen zu arbeiten, was eine bessere Approximation ergibt.
Wenn die Abtastzeit klein ist ergibt sich ein sehr kleiner Unterschied zwischen diesen verschiedenen Berechnungsmethoden.
Die Formeln sehen je nach benutzter Methode ein wenig anders aus.

Wenn man sich die Formel für y(k) anschaut sieht man dass über alle eingelesenen Werte der Regeldifferenz summiert wird.

Dies könnte man tun indem man alle Werte abspeichert und jedes Mal neu die Summe berechnet, was speichermäßig und rechenzeitmäßig schlecht möglich ist.

Effizienter ist es auf den vorher berechneten Wert y(k-1) der Stellgröße zurückzugreifen. Man spricht hier von einem **rekursiven Algorithmus**.

(Ein rekursiver Algorithmus greift auf schon vorher berechnete Werte zurück).

Am einfachsten erhält man ihn durch Differenzbildung zwischen y(k) und y(k-1):

$$y(k)=\frac{T_0}{T_I}\cdot[e(0)+e(1)+.....+e(k-2)+e(k-1)]$$
$$y(k-1)=\frac{T_0}{T_I}\cdot[e(0)+e(1)+.....+e(k-3)+e(k-2)]$$

Bei der Differenzbildung fallen fast alle Terme weg:

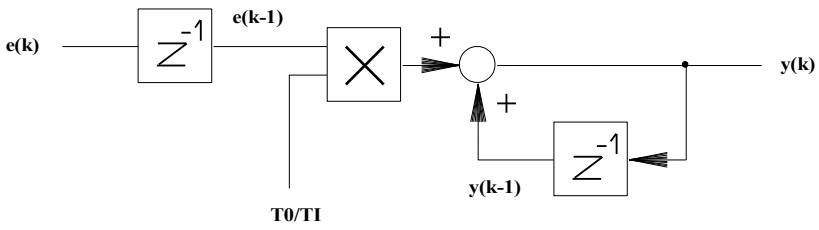
$$y(k)-y(k-1)=\frac{T_0}{T_I}\cdot e(k-1)$$

und umgestellt nach y(k) erhält man

$$y(k)=y(k-1)+\frac{T_0}{T_I}\cdot e(k-1)$$

Der neue Wert der Stellgröße wird also aus ihrem vorherigen, gespeicherten Wert y(k-1) und dem vorherigen, gespeicherten Wert e(k-1) der Regeldifferenz gebildet.

Im Signalflussplan sieht das so aus:



Aufgabe DigR 4
Realisiere ein I-Glied mit T_I = 5s in BORIS nach obigem Signalflussplan.
Wähle die Abtastzeit zu 0.1s, ebenso die Simulationsschrittweite.

Überprüfe die Funktion indem du die Sprungantwort aufnimmst.

Vergleiche mit dem zu erwartenden Verlauf der Ausgangsspannung.

9) Regelalgorithmus für einen PID-Regler

Für einen analogen PID-Regler gilt: $y = K_{PR} e + \frac{1}{T_I} \cdot \int e\,dt + T_D \frac{de}{dt}$

Durch **Einsetzen der nichtrekursiven** Algorithmen für den digitalen P-, I- und D-Anteil

$$y_P(k) = K_{PR} \cdot e(k)$$

$$y_I(k) = \frac{T_0}{T_I} \cdot \sum_{i=0}^{k-1} e(i)$$

$$y_D(k) = \frac{T_D}{T_0} \cdot [e(k) - e(k-1)]$$

in $y_{PID} = y_P + y_I + y_D$ erhält man

$$y_{PID}(k) = K_{PR} \cdot e(k) + \frac{T_0}{T_I} \cdot \sum_{i=0}^{k-1} e(i) + \frac{T_D}{T_0} \cdot [e(k) - e(k-1)]$$

Bemerkung:
Zur Herleitung werden die nichtrekursiven Algorithmen benutzt. Der rekursive I-Algorithmus enthält y(k-1), dies ist aber das abgespeicherte Ausgangssignal nur des I-Anteils, und nicht des ganzen PID-Regelers. Wir wollen gleich einen rekursiven PID-Algorithmus herleiten, bei dem y(k-1) das gespeicherte Ausgangssignal des ganzen Reglers ist.

Der rekursive Algorithmus wird wieder hergeleitet durch die Bildung der Differenz von y(k) und y(k-1):

$$y_{PID}(k) = K_{PR} \cdot e(k) + \frac{T_0}{T_I} \cdot \sum_{i=0}^{k-1} e(i) + \frac{T_D}{T_0} \cdot [e(k) - e(k-1)]$$

$$y_{PID}(k-1) = K_{PR} \cdot e(k-1) + \frac{T_0}{T_I} \cdot \sum_{i=0}^{k-2} e(i) + \frac{T_D}{T_0} \cdot [e(k-1) - e(k-2)]$$

Schreibt man die Summen aus, sieht man dass sich bei der Differenzbildung wieder die meisten Terme aufheben:

$$y_{PID}(k) = K_{PR} \cdot e(k) + \frac{T_0}{T_I} \cdot [e(0) + e(1) + ... + e(k-2) + e(k-1)] + \frac{T_D}{T_0} \cdot [e(k) - e(k-1)]$$

$$y_{PID}(k-1) = K_{PR} \cdot e(k-1) + \frac{T_0}{T_I} \cdot [e(0) + e(1) + ... e(k-2)] + \frac{T_D}{T_0} \cdot [e(k-1) - e(k-2)]$$

Differenzbildung:

$$y_{PID}(k) - y_{PID}(k-1) = K_{PR} \cdot [e(k) - e(k-1)] + \frac{T_0}{T_I} \cdot [e(k-1)] + \frac{T_D}{T_0} \cdot [e(k) - 2e(k-1) + e(k-2)]$$

$$y_{PID}(k)=y_{PID}(k-1)+[K_{PR}+\frac{T_D}{T_0}]e(k)+[\frac{T_0}{T_I}-K_{PR}-2\frac{T_D}{T_0}]e(k-1)+\frac{T_D}{T_0}e(k-2)$$

Mit den Abkürzungen

$$q_0=K_{PR}+\frac{T_D}{T_0}$$
$$q_1=-K_{PR}+\frac{T_0}{T_I}-2\frac{T_D}{T_0}$$
$$q_2=\frac{T_D}{T_0}$$

ergibt sich eine übersichtliche Darstellung:

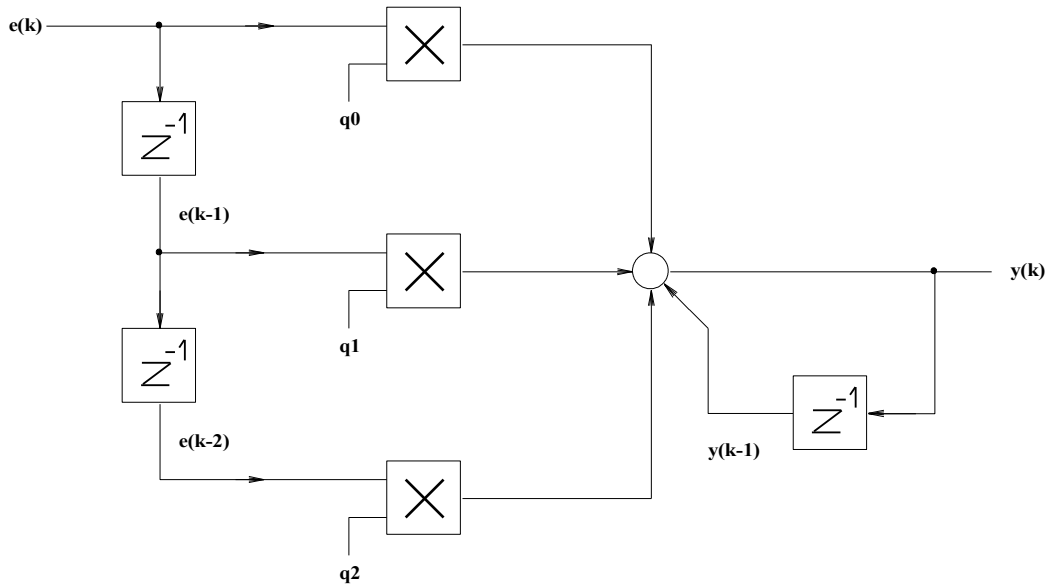
$$y_{PID}(k)=y(k-1)+q_0e(k)+q_1e(k-1)+q_2e(k-2)$$

Die Koeffizienten q_0, q_1, q_2 können auch mit T_N und T_V geschrieben werden:

$$q_0=K_{PR}(1+\frac{T_V}{T_0})$$
$$q_1=-K_{PR}(1-\frac{T_0}{T_N}+2\frac{T_V}{T_0})$$
$$q_2=K_{PR}\frac{T_V}{T_0}$$

Sie brauchen nur einmal zu Beginn des Programms oder bei einer Parameteränderung berechnet zu werden.

Darstellung im Signalflussplan:



PID-Regler in BASCOM:

Das Grundgerüst für den P-Regler kann beibeghalten werden, mit zwei Änderungen:

- Stallt des Algorithmus P_Control wird der Algorithmus PID_Control aufgerufen
- Für alle neu hinzugekommenen Variablen muss im Initialisierungsteil des Programms Speicherplatz reserviert werden.

```
Pid_control:
       'Y = Y1 + Q0*E + Q1*E1 + Q2*E2
       'die Berechnung ist etwas kompliziert da immer nur 1 Operation pro Zeile

           Tmpreg2 = Q2 * E2
           Tmpreg1 = Q1 * E1
           Tmpreg1 = Tmpreg1 + Tmpreg2  'Tmpreg1=   Q1 * E1 +  Q2 * E2
           Tmpreg2 = Q0 * E
           Tmpreg1 = Tmpreg1 + Tmpreg2  'Tmpreg1= Q0 * E + Q1 * E1 +  Q2 * E2
           Tmpreg1 = Tmpreg1 + Y1

           Y = Tmpreg1

           'Werte speichern – Achtung: die Reihenfolge E1, E2 ist wichtig!
           E2 = E1
           E1 = E
           Y1 = Y

Return
```

Hier werden die Koeffizienten q_0, q_1, q_2 benutzt. Diese müssen vorher natürlich mit den richtigen Werten gesetzt werden.

Am besten schreibt man ein kleines Unterprogramm welches diese Koeffizienten aus K_{PR}, T_N und T_V berechnet.

9) Digitale PI- und PD-Regler

Aufgabe DigR 5
Leite den Algorithmus für einen digitalen PD-Regler her, ausgehend von den Algorithmen für P- und D-Regler.

Aufgabe DigR 6
Leite den Algorithmus für einen digitalen PI-Regler her, ausgehend von den Algorithmen für P- und I-Regler. Benutze für den I-Anteil zunächst den nichtrekursiven Algorithmus.

Leite dann den rekursiven PI-Algorithmus durch Differenzbildung y(k) – y(k-1) her.