

Application Note 12

Analog- und Digitalsignale einlesen und ausgeben mit eigerScript Methoden und Beispiele

Letzte Bearbeitung: 18. April 2008

Christoph Angst

Unterägeri, 11. März 2008

© S-TEC electronics AG, CH-6300 Zug
s-tec@bluewin.ch
www.s-tec.ch
www.eigergraphics.com



Dieses Dokument können Sie von der Internetseite www.eigergraphics.com als PDF-Datei herunterladen.

Inhaltsverzeichnis

1	Einleitung	2
2	Komponenten und Schnittstellen des FOX57	3
2.1	Komponenten und Anschlüsse von FOX57	3
2.2	Schnittstellen für Analog-Inputs	4
2.3	In-/Outputs des Expansion Connectors	5
3	eigerScript-Methoden für In/Output	7
3.1	Methoden der Klasse InOut für Analog-Kanäle	7
	InOut.Read_ADC(VarInt:Kanal,VarInt:ADC-Wert)	7
3.2	Methoden der InOut-Class für Digital-Kanäle	7
	InOut.DigitalOutputDriver(VarInt:Kanal,VarInt:Funktion)	8
	Register el.DA_P76 für PWM-Output OP76	10
	InOut.PWM_Out(VarInt:OP72,VarInt:Value)	11
4	Anwendungsbeispiele	12
4.1	Temperaturmessung	12
4.2	Voltmeter	13
4.3	Potentiometer	16
4.4	Licht ein- und ausschalten	21
4.5	Digital Output OP92 – Spielereien mit dem Buzzer	22
4.6	Display-Hinterleuchtung dimmen („Standby-Modus“)	25

1 Einleitung

In dieser Application Note wird gezeigt, welche eigerScript Methoden für den Einsatz der analogen und digitalen In- und Outputs des eigerPanels 57 zur Verfügung stehen. Zu dieser Application Note gehören auch konkrete Programmbeispiele. Diese sind in der eigerDemoCD unter dem Verzeichnis Application Notes abgespeichert und können direkt via CompactFlash Card auf dem eigerPanel 57 ausgeführt werden. Beachten Sie dabei, dass die Verzeichnis-Struktur stimmt (inkl. obligatorische Startdateien FOXLOGOLE.GI und START.FOX). Eine genaue Anleitung dafür finden Sie im „eigerScript – Schnelleinstieg“ (zu finden auf der CD oder auf der Download-Seite von www.eigergraphics.com).

2 Komponenten und Schnittstellen des FOX57

2.1 Komponenten und Anschlüsse von FOX57

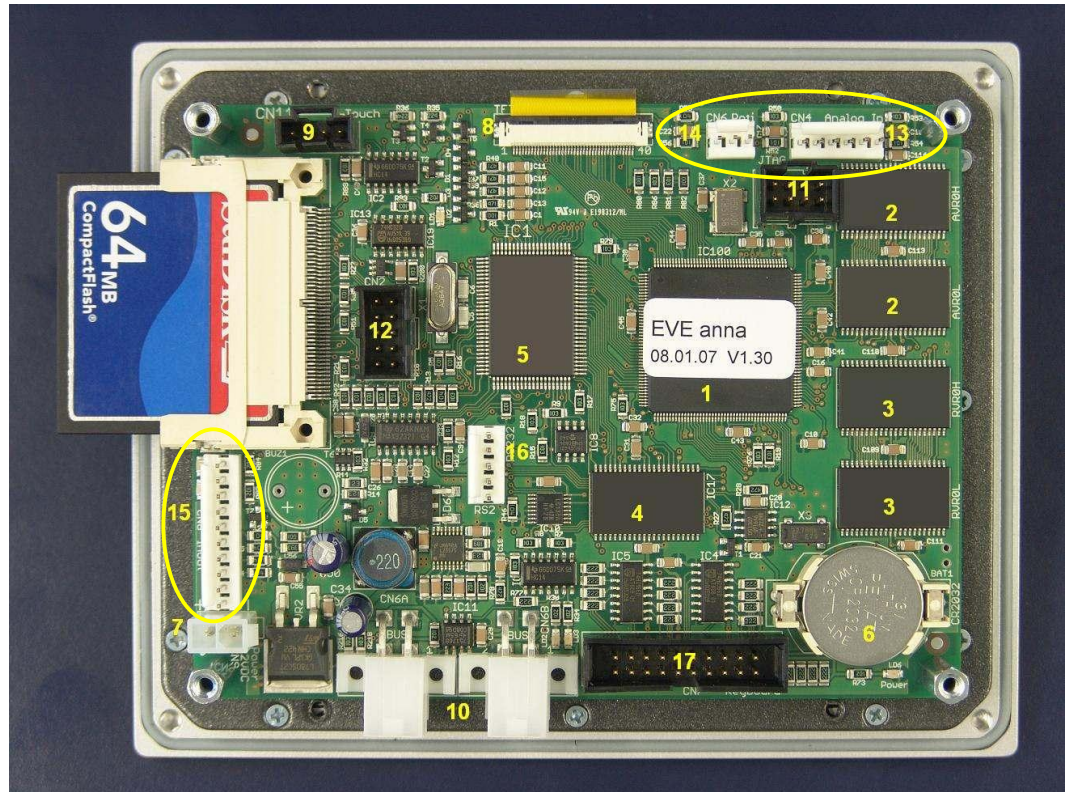


Abbildung 1: Komponenten des embedded Computers für das eigerPanel 57

- | | |
|---|--|
| 1. Graphic Controller, eigerVideo Engine (EVE anna) | 9. Anschluss Touchpanel |
| 2. Videospeicher Accessible Video Ram (AVR) | 10. BUS (Serielle Schnittstelle RS485) |
| 3. Videospeicher Refresh Video Ram (RVR) | 11. Programmier-Schnittstelle für EVE anna (JTAG) |
| 4. Arbeitsspeicher (RAM) | 12. Programmier-Schnittstelle für Microprozessor (S-PROG10) oder mit Adapterkabel F4337 als FOX-COM1 (UART1) verwendbar. Debug-Schnittstelle |
| 5. CPU (Micro-Prozessor) | 13. Analog-Eingänge |
| 6. RTC-Batterie | 14. Analogeingang Poti |
| 7. Power Supply 12V | 15. Externe I/O |
| 8. Anschluss Display VGA | 16. serielle Schnittstelle RS232, FOX-COM2 (UART2) |
| | 17. 16-KEY Keyboard Input |

2.2 Schnittstellen für Analog-Inputs

Der embedded Computer FOX 57 verfügt über 3 Analog-Eingänge mit 10 Bit Auflösung (0 ... 1023). Die Eingänge PA2 und PA3 sind im Steckanschluss CN4 vereint, während PA0 als separate Stiftleiste (CN6) angeordnet ist (vgl. Abbildung 2). Die Eingangsspannung darf 0 bis 3.3 Volt betragen, d.h. **für PA2 und PA3 darf die Eingangsspannung 3.3 Volt nicht überschreiten.**

Der Analog-Digital-Wandler (A/D-Wandler) bereitet die analogen Eingangssignale automatisch in digitale Signale auf, wobei die Auflösung 10 Bit ($2^{10} = 1024$) beträgt. Das bedeutet, dass die eingehende Spannung zwischen 0 bis 3.3 Volt in den entsprechenden digitalen Wert zwischen 0 und 1023 umgewandelt wird.

Die 10k-Widerstände von PA2 und PA3 gegen LGND (Logik-Ground) dienen als Bezugswiderstände für 10k-NTC-Sensoren (**N**egative **T**emperature **C**oefficient). Diese messen am genauesten bei 25 °C (Zimmertemperatur). Wenn eine Signalquelle an den A/D-Wandler angeschlossen werden soll, muss der relativ geringe Innenwiderstand von 10 kΩ berücksichtigt werden.

Der Anschluss PA0 ist mit einem Bezugswiderstand von 100 kΩ ausgestattet und für Potentiometer konzipiert.

Eingangsspannung [Volt]	A/D-Wert
0	0
1.0	310
2	620
3	930
3.3	1023

Berechnung des digitalen A/D-Wertes:

$$\text{A/D-Wert} = 1023 / 3.3 * \text{Eingangsspannung}$$

1Bit (TIC) entspricht 3.225 mV

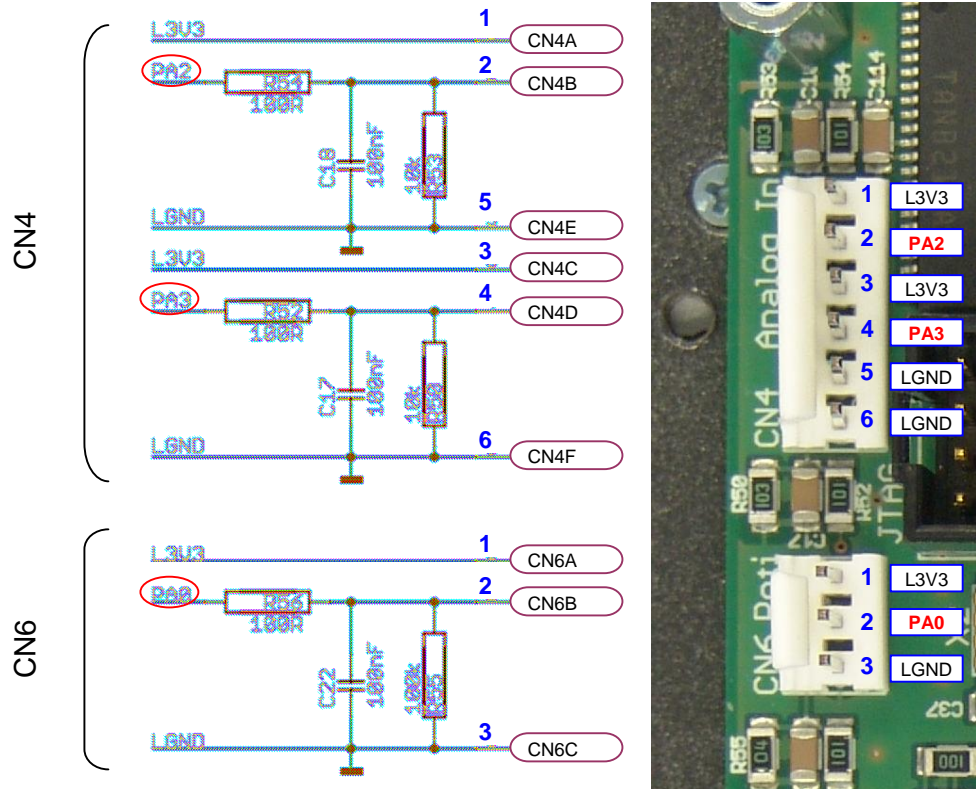


Abbildung 2: Drei Analog-Anschlüsse (2 Anschlüsse auf CN4 und 1 Anschluss auf CN6) – Schema und Foto der Pins auf dem eigerPanel 57. eigerScript nimmt mittels der Bezeichnungen PA0, PA2 und PA3 den entsprechenden Analog-Input entgegen.

2.3 In-/Outputs des Expansion Connectors

Der Expansion Connector (CN8) besteht aus mehrheitlich digitalen Ein- und Ausgängen (vgl. Abbildung 3). Der digitale Ausgang 92 (OP92) ist parallel auch noch mit dem Buzzer verbunden, d.h. gleichzeitig mit der Signalausgabe über OP92 ertönt auch ein Alarmton. Da es sich dabei um ein digitales Signal handelt, kann dessen Lautstärke nicht beeinflusst werden.

Das eigerPanel 57 können wir auch als **Voltmeter** einsetzen. Der hierfür vorgesehene Analog-Eingang VOLT_IN ist ebenfalls auf der Stiftleiste CN8 angesiedelt (vgl. Abbildung 3, Abbildung 4). Das Voltmeter ist für maximal 50 Volt konfiguriert (50 V und mehr ergeben als maximalen Wert 1024). Höhere Eingangsspannungen sollten vermieden werden (vgl. Tabelle 1, S.7).

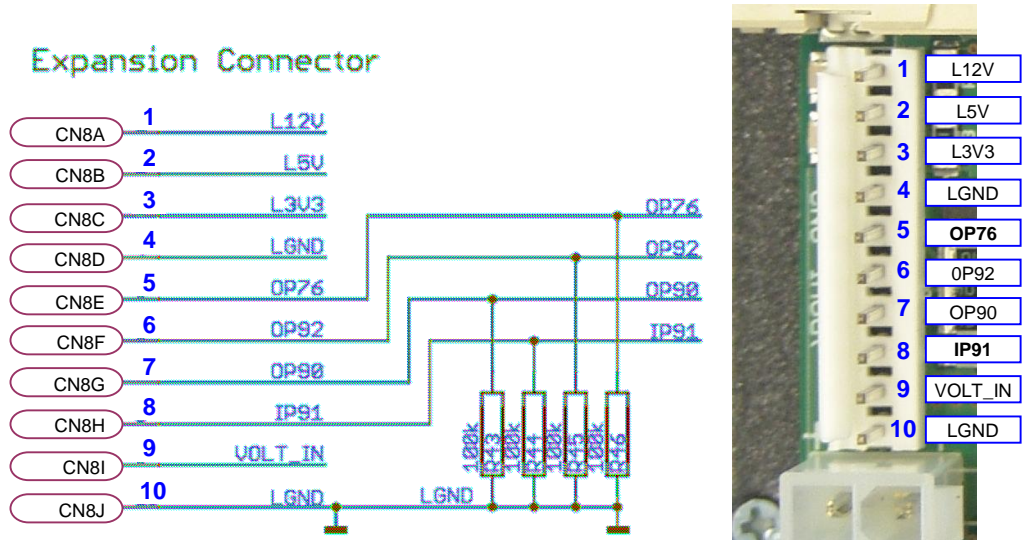


Abbildung 3: Schema und Steckleiste des Expansion Connectors mit den Outputs OP76, OP92, OP90 und den Inputs IP91 und VOLT_IN (entspricht PA1). Das Detailschema des Voltage Inputs VOLT_IN ist in Abbildung 4 dargestellt.

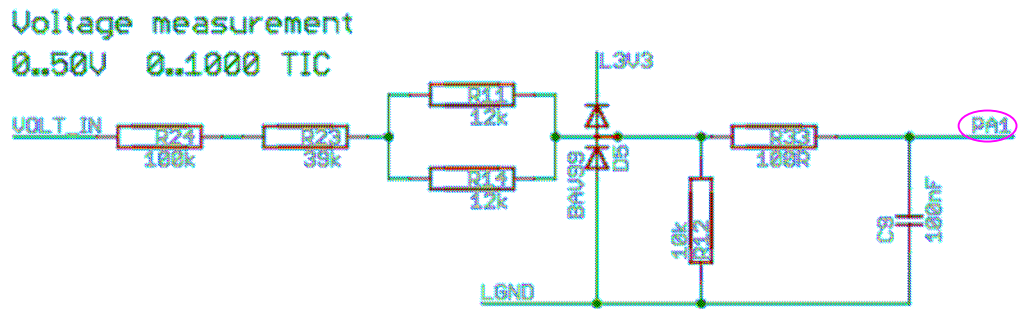


Abbildung 4: Schema des Voltage Inputs (VOLT_IN entspricht PA1). Der Voltage-Input ist im Expansion Connector angesiedelt (vgl. Abbildung 3).

3 eigerScript-Methoden für In/Output

Die Methoden für das Management der Ein- und Ausgänge sind in eigerScript in der Klasse InOut zusammengefasst. Die nachfolgende Beschreibung dieser Klasse und der zugehörigen Methoden sind zum Teil dem Befehlskatalog „Befehlsreferenz_eigerScript_eVM.pdf“ entnommen.

Die Methoden der Klasse InOut bedienen die Peripherie des Mikrocomputersystems.

3.1 Methoden der Klasse InOut für Analog-Kanäle

Tabelle 1: Analog-Kanäle und ihre Eingänge im eigerPanel 57

Kanal	Eingang	Maximale Eingangsspannung
0	PA0	3.3 V
1	PA1 (Voltage Input VOLT_IN)	50 V
2	PA2	3.3 V
3	PA3	3.3 V

InOut.Read_ADC(VarInt:Kanal,VarInt:ADC-Wert)

Mit dem Befehl `InOut.Read_ADC(Kanal,ZielVariable)` wird ein Analog-Kanal eingelesen (vgl. Tabelle 1), beispielsweise von einem NTC-Sensor (Kanäle 2,3) oder einem Potentiometer (Kanal 0). Der Eingangswert (Eingangsspannung 0 bis 3.3 V) wird durch den A/D-Wandler (ADC = Analog to Digital Converter) in einen Wert zw. 0 und 1023 (10-Bit Auflösung) umgewandelt, welcher in der Zielvariablen gespeichert wird (vgl. Kap.2.2, S.4).

Beispiel-Code 1: Analogwandler Kanal 0 in Register `eI.Int_2` einlesen.

```
InOut.Read_ADC(0,eI.Int_2)
```

Ein ausführliches Anwendungsbeispiel finden Sie auf Seite 7 (Beispiel-Code 8).

3.2 Methoden der InOut-Class für Digital-Kanäle

Die InOut-Class enthält auch eine Reihe von Befehlen und Funktionen, mit welchen die digitalen Ausgänge (vgl. Tabelle 2) angesteuert werden können. Nebst einfacher Ein-/Aus-Ansteuerung können auch Pulsweiten, -pausen sowie Anzahl Wiederholungen bestimmt werden.

Der weitere Programmablauf wird durch die Ausgabe des Signals nicht aufgehalten. Das eigerPanel reagiert währenddessen weiterhin auf Events und arbeitet nachfolgende Befehle ab.

Tabelle 2: Digital-Output-Kanäle des Expansion Connectors CN8 (eigerPanel 57)

Kanal	Beschreibung
OP90	Digital Output, 3.3 V
OP92	Digital Output, 3.3 V, parallel geschaltet mit dem Buzzer
OP72	PWM für die Bildschirm-Hinterleuchtung, Duty-Cycle [0..100%] = [0..1200]
OP76	PWM, 3.3 V, Frequenz 1kHz, Duty-Cycle [0..100%] = [0..3000]

`InOut.DigitalOutputDriver (VarInt:Kanal, VarInt:Funktion)`

Mit dem Befehl `InOut.DigitalOutputDriver (Kanal, Funktion)` wird über den angesprochenen Output-Kanal eine Spannung erzeugt (Funktion `Output_On`) bzw. abgebrochen (`Output_Off`). Die Spannung beträgt 3.3 V für OP90 und OP92 (vgl. Tabelle 2).

Für die Digital-Output-Kanäle OP90/92 stehen folgende Funktionen zur Verfügung:

Funktionen für OP90 und OP92:

- **Output_On:** startet die Spannungsausgabe

Beispiel-Code 2: Start der Spannungsausgabe über den Kanal OP92.

```
InOut.DigitalOutputDriver (OP92, Output_On)
```

- **Output_Off:** beendet die Spannungsausgabe

Beispiel-Code 3: Ende der Spannungsausgabe über den Kanal OP92.

```
InOut.DigitalOutputDriver (OP92, Output_Off)
```

- **Output_Toggle:** Wenn der Output High ist, wird auf Low gewechselt und umgekehrt. Aus ON wird OFF und umgekehrt.

Beispiel-Code 4: Wechseln der Einstellung (ON ↔ OFF).

```
InOut.DigitalOutputDriver (OP92, Output_Toggle)
```

- **Output_Pulse:** gibt eine bestimmte Anzahl Spannungssignale aus. In den Registern `eI.P90_Pulse_Count` und `eI.P92_Pulse_Count` wird die Anzahl der ausgehenden Impulse festgelegt. Die *Pulsdauer* kann im Register `eI.P90_Time_ON` bzw. `eI.P92_Time_ON` angegeben werden, in Millisekunden. Für die *Pausendauer* zwischen den Impulsen ist das Register `eI.P90_Time_OFF` bzw. `eI.P92_Time_OFF` vorgesehen; die Einheit ist ebenfalls Millisekunden.

Beispiel-Code 5: Drei kurze Pulse, auf OP92 auch als Alarm zu hören (vgl. Abbildung 5).

```
eI.P92_Pulse_Count := 3 ; Anzahl Pulse
eI.P92_Time_ON := 50 ; Pulsdauer in Millisekunden
eI.P92_Time_OFF := 100 ; Pausendauer in Millisekunden
InOut.DigitalOutputDriver (OP92, Output_Pulse)
```

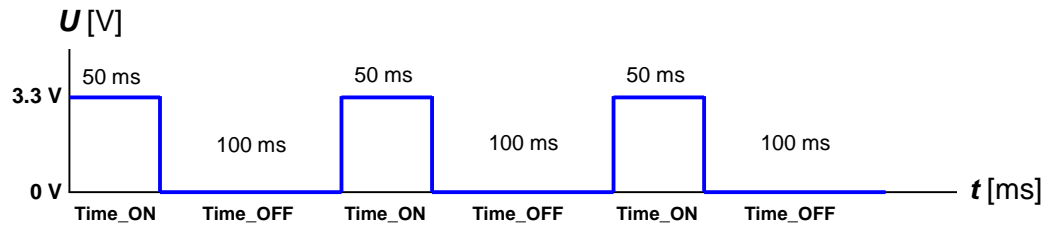


Abbildung 5: Signalverlauf aufgrund von Beispiel-Code 5.

- **Output_PulseTouchDown:** reagiert auf Berührung des Touchpanels mit der Ausgabe eines Signals. Die Pulsdauer kann durch entsprechende Zuweisung im Register `eI.P90_Time_ON` bzw. `eI.P92_Time_ON` bestimmt werden.

Beispiel-Code 6: Kurzer Puls auf OP92 bei Berühren des Displays (auch als Alarm zu hören).

```
eI.P92_Time_ON := 50 ; Pulsdauer in Millisekunden
InOut.DigitalOutputDriver(OP92,Output_PulseTouchDown)
```

Register:

Mit den Digital-Outputs stehen verschiedene Register in Verbindung. Die meisten sind oben bereits in ihrem Kontext erwähnt. In der folgenden Tabelle 3 sind die betreffenden Register aufgelistet und nachfolgenden Blockschema dargestellt (Abbildung 6).

Tabelle 3: Register zu den Digital-Outputs OP90 und OP92.

Register*	Beschreibung
<code>eI.P90_Output</code>	Enthält den aktuellen Zustand des Outputs: 0 = OFF, 1 = ON.
<code>eI.P90_Pulse_Count</code>	Enthält die Anzahl der zu sendenden Impulse.
<code>eI.P90_Time_ON / _OFF</code>	Enthält die Anzahl Millisekunden für die Pulsdauer (ON) bzw. für die Pausendauer (OFF) eines Impulses.

* P90 steht auch für P92, z.B. `eI.P92_Output`.

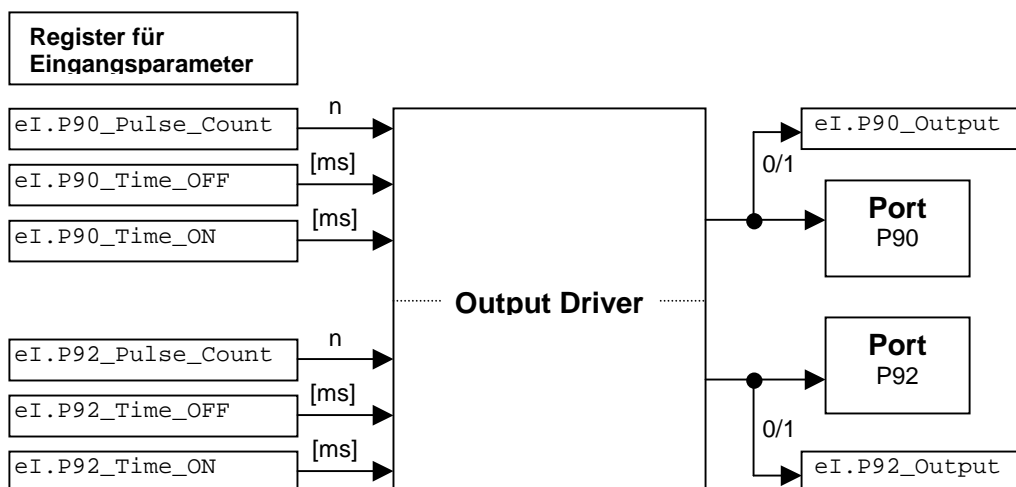


Abbildung 6: Blockschema für die digitalen Ausgänge P90 und P92

Buzzer:

Das eigerPanel 57 ist mit einem Buzzer (Alarm) ausgerüstet. Dieser ist mit dem Output OP92 parallel geschaltet. Damit wird gleichzeitig mit einer Befehlsausgabe der Alarm betätigt. Der Alarm ertönt, solange der Kanal OP92 auf „On“ gesetzt ist. Bei Anwendung der Funktion `Output_Pulse` ertönt der Alarm entsprechend der Anzahl Impulse, die im Register `eI.P92_Pulse_Count` gespeichert ist.

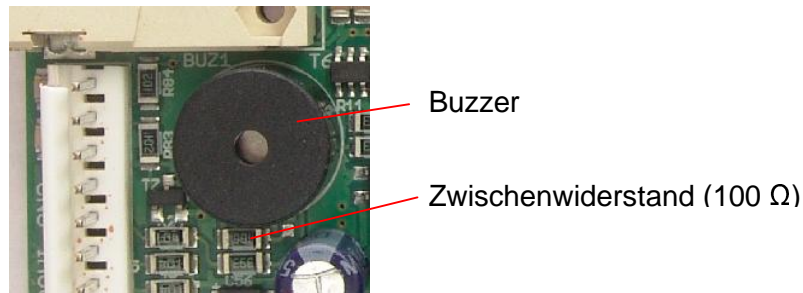


Abbildung 7: Buzzer und Zwischenwiderstand auf dem eigerPanel 57



Buzzer „ausschalten“: Falls unbedingt ein zweiter Output ohne Buzzer benötigt wird, kann der Buzzer oder der zugehörige Zwischenwiderstand (100 Ω) ausgelötet werden (vgl. Abbildung 7).

Register `eI.DA_P76` für PWM-Output OP76

Im Register `eI.DA_P76` wird mit einem Integerwert die Pulsdauer für den PWM Output OP76 (vgl. Abbildung 3) festgelegt. Der Registereintrag kann Werte zwischen 0 und 3000 annehmen, wobei die Pulsdauer $t = 3000$ (vgl. Abbildung 8) der vollen Periodenlänge (T) entspricht. Die Frequenz des Signals beträgt 1kHz und kann zur Zeit (Stand 22.10.07) noch nicht geändert werden.

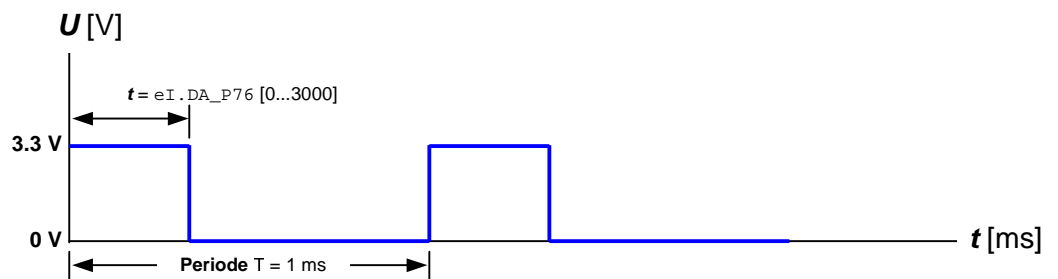


Abbildung 8: Puls- und Periodendauer beim Signal des PWM-Outputs OP76 (Frequenz = 1kHz).

InOut.PWM_Out(VarInt:OP72,VarInt:Value)

Mit dem Befehl `InOut.PWM_Out(VarInt:OP72,VarInt:Value)` wird die Bildschirm-Hinterleuchtung gesteuert. Die dafür verwendete Integervariable kann einen Wert zwischen 0 und 1200 annehmen. Bei 0 ist der Bildschirm dunkel und bei 1200 mit vollem Licht hinterleuchtet.

Mit diesem Befehl kann beispielsweise ein Dimmer oder ein Sleepmodus implementiert werden.

Beispiel-Code 7: Bildschirm-Hinterleuchtung mit halber Lichtstärke.

```
INTEGER Hinterleuchtung = 600  
InOut.PWM_Out(OP72,Hinterleuchtung.I)
```

4 Anwendungsbeispiele

4.1 Temperaturmessung

eigerScript-Methode:

InOut.Read_ADC(Kanal,Zielvariable)

Wenn Sie dieses Dokument von der CD oder vom Internet heruntergeladen haben, finden Sie im selben Verzeichnis auch das eigerScript-Projekt TEMP mit dem Programmcode TEMP_001.EVS. Laden Sie das Projekt TEMP mit einer CompactFlash Card auf Ihr eigerPanel57. Mit Hilfe der NTC-Sensoren (Teil der DemoKit-Ausrüstung) können Sie nun die Temperatur messen. Sie finden dieses Programm auch auf der Beispielsammlung TG12 als TG12_223.EVS (*System > Analog Inputs*).

Die Temperaturmessung wird in der View TEMP_001.EVS mit Hilfe eines Timers gesteuert, der alle 500 Millisekunden eine Temperaturmessung auslöst, d.h. die Subroutine **SUB GetTemp_CH2** aufruft (vgl. Beispiel-Code 8 und Abbildung 9).

Beispiel-Code 8: Übernahme und Verarbeitung der analogen Temperatur-Eingangswerte eines NTC-Sensors vom Input PA2 (Kanal 2).

```

SUB GetTemp_CH2

; Temperatur in Grad * 10 z. B. 230 für 23.0 Grad

InOut.Read_ADC(2,AnalogInput_CH2.I) ; Analogwert einlesen
Input.I := AnalogInput_CH2.I

InOut.Read_ADC(2,AnalogInput_CH2.I) ; Analogwert einlesen
Input.I := Input.I + AnalogInput_CH2.I

InOut.Read_ADC(2,AnalogInput_CH2.I) ; Analogwert einlesen
Input.I := Input.I + AnalogInput_CH2.I

InOut.Read_ADC(2,AnalogInput_CH2.I) ; Analogwert einlesen
Input.I := Input.I + AnalogInput_CH2.I
Input.I := Input.I / 4 ; Mittelwert bilden

Load.Data_from_ARRAY_Integer(TempTab_ADC10bit_10k10k,Input.I,AnalogI
nput_CH2.I)

Str.Clear(DDT_TimeString.$)
Str.Cvt_IntegerDeci(DDT_TimeString.$,Input.I,5,1)
Str.Concat(DDT_TimeString.$,'%C')

Fill.LabelParameter(AnalogValueStyle_64B_ON )
Label.Text(DDT_TimeString.$)

ENDSUB

```

Erklärung zu Beispiel-Code 8: An Kanal 2 ist ein NTC-Sensor angeschlossen. Je höher die (Raum-)Temperatur, um so besser leitet der NTC-Sensor den Strom, d.h. je höher die zu messende Temperatur, um so kleiner der Sensor-Widerstand. Die Eingangsspannung im Kanal 2 ist dadurch bei tiefen Temperaturen näher bei 0 V und steigt bei zunehmender Wärme gegen 3.3

V. Mit Hilfe des A/D-Wandlers ergibt sich daraus ein Digitalwert zwischen 0 und 1023. Dieser Wert wird in unserem Beispiel mit der Methode `InOut.Read_ADC()` in die Integer-Variablen `AnalogInput_CH2.I` eingelesen.

Um die Ausgabe des Temperaturwertes auf das Display etwas ruhiger zu gestalten, wird die Temperatur für einen Ausgabezyklus 4 mal gemessen. Der Mittelwert wird schlussendlich als in der Integer-Variablen `Input.I` gespeichert. Nun folgt die Umwandlung des Digitalwertes in den Temperaturwert (°C). Dazu dient die Umrechnungstabelle `TempTab_ADC10bit_10k10k`. Sie enthält 1024 Zeilen mit jeweils dem Digitalwert (= Zeilennummer) und dem entsprechenden Temperaturwert in °C mal 10. Enthält die Variable `Input.I` den Wert 512, so lesen wir mit Hilfe der Methode

```
Load.Data_from_ARRAY_Integer(TempTab_ADC10bit_10k10k, Input.I, AnalogInput_CH2.I)
```

den Wert 250 aus der Tabelle, was 25.0 °C entspricht. Das richtige Zahlenformat mit einer Dezimalstelle erhalten wir durch Verwendung der Methode `Str.Cvt_IntegerDeci(DDT_TimeString.$, Input.I, 5, 1)`, welche z.B. die Zahl 250 in den maximal fünf Zeichen langen String '25.0' umwandelt und diesen der String-Variablen `DDT_TimeString.$` zuführt. Nun können wir den String mit den Methoden `Fill.LabelParameter()` und `Label.Text()` aufs Display ausgeben.

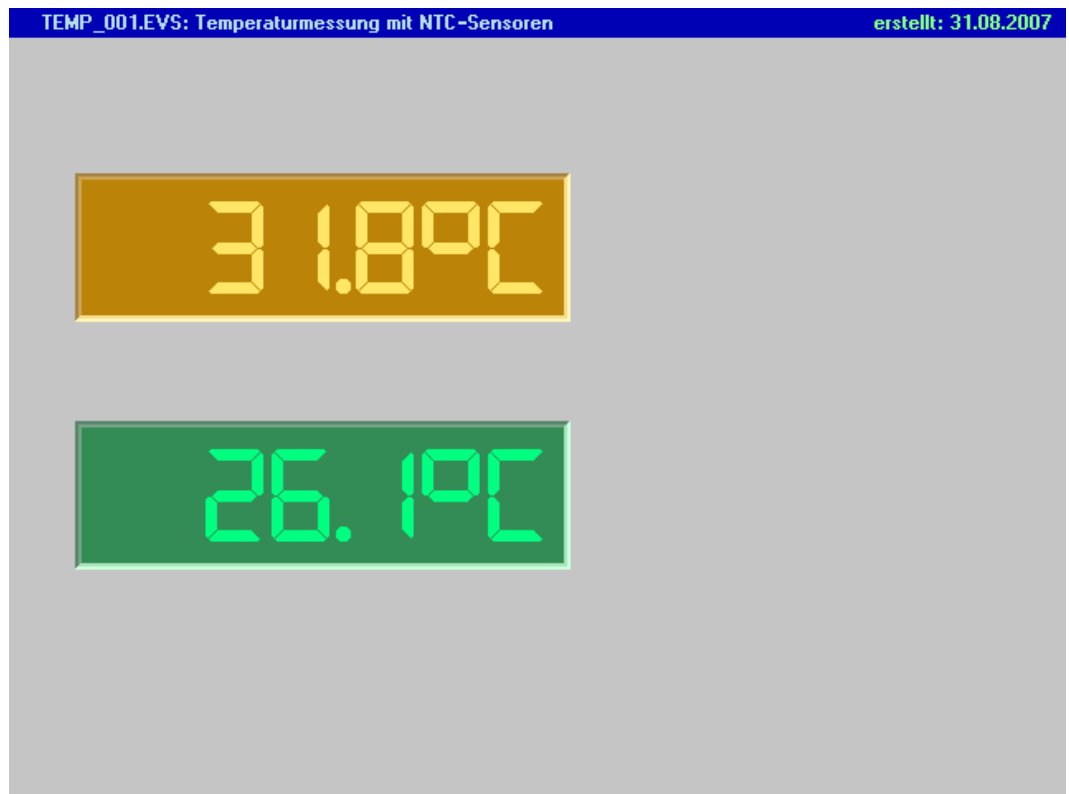


Abbildung 9: Screenshot von der Temperaturanzeige auf dem eigerPanel 57 - Display, generiert durch den Programmcode in der Source-Datei TEMP_001.EVS.

4.2 Voltmeter

Das eigerPanel 57 kann über den Analog-Eingang PA1 oder VOLT_IN (vgl. Abbildung 3 und Abbildung 4, S.6) zusammen mit dem entsprechenden Ground (CN8J) als Spannungsmesser eingesetzt werden.

eigerScript holt den auf Kanal 1 gemessenen Spannungswert mit der gleichen Methode ab, wie bei der Temperaturmessung:

```
InOut.Read_ADC(Kanal, Zielvariable)
```

Der A/D-Wandler (ADC) überträgt die gemessene Spannung in einen Digitalwert zwischen 0 und 1023, d.h. 0 für 0 Volt; 1023 für 50 Volt (Maximum).



Vorsicht: Achten Sie unbedingt darauf, dass Sie für Ihr Voltmeter wirklich nur PA1 (VOLT_IN) benützen! Nur PA1 ist für Spannungen höher als 3.3 Volt ausgelegt, d.h. bis maximal 50 Volt. Eine Verwechslung der Anschlüsse kann Ihr eigerPanel 57 unwiderruflich zerstören. Der embedded computer FOX 57 verarbeitet auf PA1 Spannungen von 0 bis 50 Volt. **Höhere Eingangsspannungen sollten vermieden werden.**

Ein Beispiel für ein Voltmeter-Programm ist das eigerProjekt VOLT mit der Programmdatei VOLT_001.EVS. Sie finden es auf der eigerDemoCD oder auf der Download-Seite www.eigergraphics.com im Verzeichnis „Application Notes“. Das Voltmeter wird im Programm VOLT_001.EVS mit Hilfe eines Timers gesteuert, der alle 500 Millisekunden eine Spannungsmessung auslöst, d.h. die Subroutine **SUB Get_and_Draw_Measure_Value** aufruft (vgl. Beispiel-Code 9 und Abbildung 9). Diese Subroutine verarbeitet die gemessenen Werte für die Anzeige auf dem Display des eigerPanel 57 (vgl. Abbildung 10, S.15)

Beispiel-Code 9: Auszug aus dem Voltmeter-Programm VOLT_001.EVS. Mit der Subroutine **SUB Get_and_Draw_Measure_Value** werden die gemessenen Spannungswerte als Digitalwerte übernommen, umgerechnet und auf dem Display angezeigt.

```

SUB Get_and_Draw_Measure_Value
Display.Prepare( )

InOut.Read_ADC( 1,ADC_VOLT_IN.I )           ; Der Messwert von Kanal 1
                                              ; (PA1 entspricht VOLT_IN)
                                              ; wird als Digitalwert in
                                              ; die Integervariable
                                              ; ADC_VOLT_IN.I eingelesen.

Voltage_Value.I := ADC_VOLT_IN.I           ; 0 bei 0 V, 1023 bei 50 V
Voltage_Value.I := Voltage_Value.I / 2     ; dies ergibt den Integerwert
                                              ; 511 bei 50 V
                                              ; (Nachkommawert nicht
                                              ; berücksichtigt)

Voltage_Value.I := Voltage_Value.I * 50    ; Anpassung von 1024-er-
                                              ; System auf 1000-er-System
Voltage_Value.I := Voltage_Value.I / 51    ; ergibt den Integerwert
                                              ; 500 bei 50 V

Str.Clear( Voltage_Value.$ )              ; Wenn Stringvariable nicht
                                              ; leer ist, wird mit Str.Cvt
                                              ; nachfolgender Wert einfach
                                              ; hinten angefügt

Str.Cvt_IntegerDeci( Voltage_Value.$,Voltage_Value.I,4,1 ) ; Spannungswert in Stringvariable
                                              ; ausgeben mit einer
                                              ; Nachkommastelle (ergibt
                                              ; '50,0' bei 50 V)

Load.Pos_X1Y1( 170,50 )
Fill.LabelParameter( Measure_Field_Style ) ; Ausgabe des analogen
                                              ; Wertes auf im "oberen"
                                              ; Feld

Label.Text( Voltage_Value.$ )

Digital_Value.I := ADC_VOLT_IN.I

```

```

Str.Clear( Digital_Value.$ )
Str.Cvt_Integer(Digital_Value.$,Digital_Value.I,4) ;Übergabe des
                                                    digitalen Wertes (Digits)
                                                    an Digital_Value.$

Load.Pos_X1Y1( 170,200 )
Fill.LabelParameter( Measure_Field_Style ) ; Layout-Parameter für
                                                    die Ausgabe
Label.Text( Digital_Value.$ ) ; Ausgabe des digitalen
                                                    Wertes im "unteren" Feld

ENDSUB

```

Erklärung zu Beispiel-Code 9: Die zu messende Spannung wird an den Eingang PA1 (VOLT_IN auf CN8) und den entsprechenden Ground (CN8J) angeschlossen. Dieser Eingang ist für die Programmiersprache eigerScript der Kanal 1. Mit Hilfe des A/D-Wandlers ergibt sich aus der gemessenen Spannung ein Digitalwert zwischen 0 und 1023. Dieser Wert wird in unserem Beispiel mit der Methode `InOut.Read_ADC()` in die Integer-Variablen `ADC_VOLT_IN.I` eingelesen. Der Inhalt der Integer-Variablen `ADC_VOLT_IN.I` wird an eine weitere Integervariable kopiert, welche der schrittweisen Umrechnung in die entsprechende Anzahl Volt dient. Für die Ausgabe auf dem Display (vgl. Abbildung 10) muss das Integer-Ergebnis mit `Str.Cvt_IntegerDeci()` in eine Stringvariable `Voltage_Value.$` eingelesen werden.

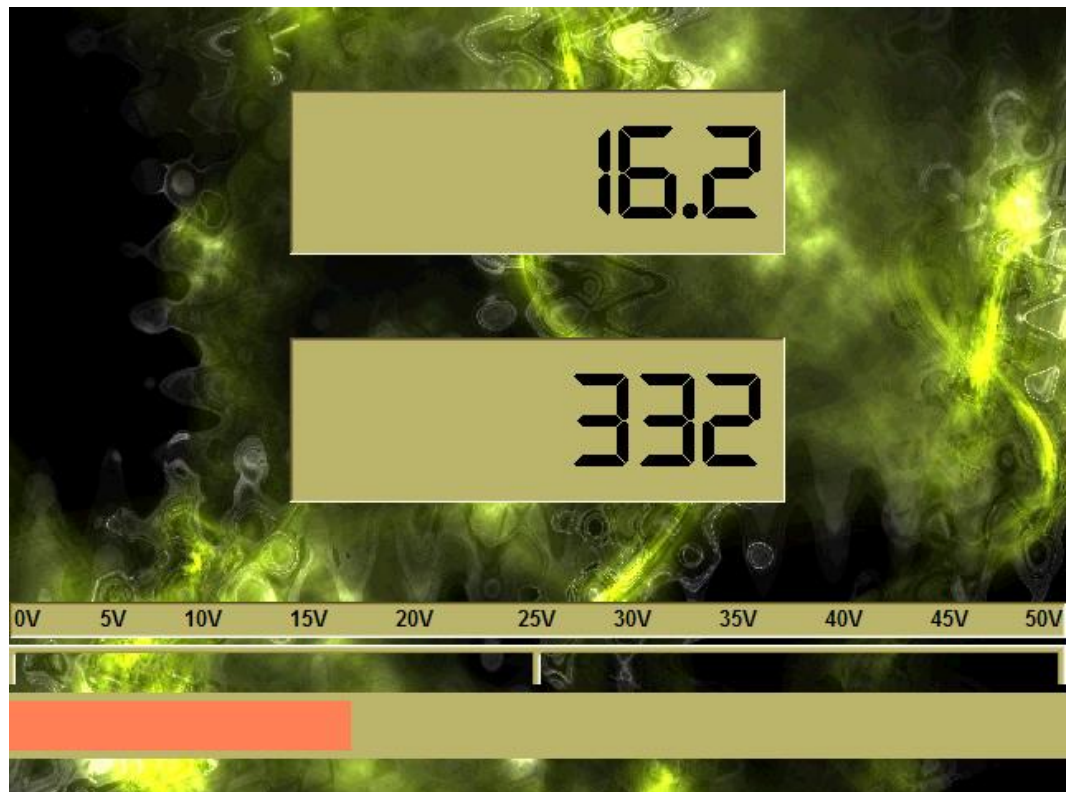


Abbildung 10: Screenshot von der Spannungsanzeige auf dem eigerPanel 57 - Display, generiert durch den Programmcode in der Source-Datei VOLT_001.EVS. In diesem Beispiel geben die obere Zahl und der Rollbalken die gemessene Spannung an (z.B. 16.2 V). Die 16.2 Volt entsprechen dem hier ebenfalls angezeigten Digitalwert 332 (Maximum: 1023).

4.3 Potentiometer

Am eigerPanel 57 können gleichzeitig 3 Potentiometer angeschlossen werden. Die betreffenden Analog-Eingänge PA0, PA2 und PA3 sind in Kapitel 2.2 (S.4) und Abbildung 2 (S.5) erklärt und abgebildet. Das Auf- und Zudrehen eines Potentiometers können wir unter anderem auch auf dem Display z.B. als Bewegung als Geschwindigkeitsänderung (Bildabfolge) oder Farbwechsel visualisieren.

Im folgenden Programmbeispiel KNUT lassen wir den kleinen Eisbären Knut über die Bildfläche laufen (vgl. Abbildung 11). Sein Tempo steuern wir mit dem Potentiometer.

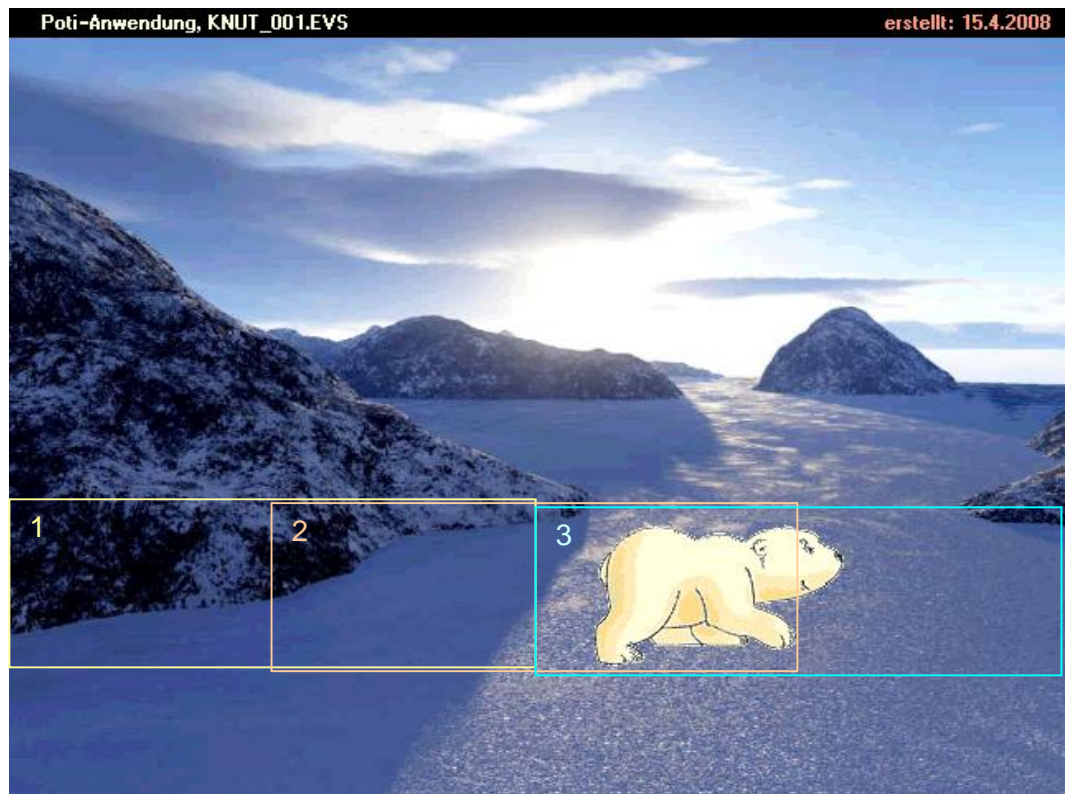


Abbildung 11: Screenshot der Poti-Anwendung KNUT. Der kleine Eisbär Knut erkundet die arktische Landschaft. Wie schnell er das tut, bestimmen wir mit dem Potentiometer. Knuts Bewegung ist eine Abfolge von 6 Bildern. Das Hintergrundbild bleibt statisch, d.h. es wird nicht zusammen mit Knut neu dargestellt. Damit Knut keine Bildspur hinterlässt, wird jedes Knut-Bild zusammen mit einem der drei Hintergrundausschnitte dargestellt, welcher das vorangegangene Knut-Bild überdeckt. Ein ständiger Neuaufbau des ganzen Bildes wäre zu langsam für einen schnellen Bewegungsablauf. Der Bildausschnitt wird in Abhängigkeit der aktuellen Position des Eisbären gewählt.

Der animierte Knut besteht aus einer Sequenz von 6 Bildern (Knut1.EGI bis Knut6.EGI, abgelegt im Ordner KNUTPICT). Diese Bilder rufen wir in einer Endlos-Schleife nacheinander auf. Ein Timer bestimmt das Intervall (vgl. Beispiel-Code 10, S.17, Ziffer **A**). Jedes Mal, wenn der Timer abgelaufen ist, lesen wir mit dem Befehl `InOut.Read_ADC(0, AnalogInputPoti.I)` vom Analog-Eingang PA0 das Signal des Potentiometers (**B**) ein und starten den Timer von neuem (**C**). Entsprechend der aktuellen Einstellung des Potentiometers erhält die Integer-Variable `AnalogInputPoti.I` einen Wert zwischen 0

und 1023 zugewiesen. Diesen Wert übernehmen wir unverändert als Anzahl Millisekunden für das neue Timer-Intervall zwischen den Knut-Bildern (A).

Knut soll sich vorwärts bewegen, d.h. jedes neue Bild soll gegenüber dem vorangegangenen um eine bestimmte „Schrittlänge“ nach rechts versetzt werden. Wir erhöhen also bei jedem Ablauf des Timers auch die X-Koordinate von Knut jeweils um z.B. 10 Pixel (D). Wenn die X-Koordinate 640 Pixel erreicht hat ist Knut rechts aus dem Bild gelaufen. Das ist der Zeitpunkt, wo die X-Koordinate wieder auf 0 gesetzt wird (E), so dass der Eisbär auf der linken Seite wieder erscheint.

Knut kann ohne weiteres aus dem Bild laufen, weil das Bild sozusagen über den rechten Bildrahmen hinausgeht. Die Displayfläche umfasst 640x480 Pixel, während die beiden Videospeicher AVR und RVR¹ des eigerPanels 57 eine Fläche von 1024x512 unterstützen. Dadurch steht auf der rechten Seite des Bildes ein unsichtbarer Streifen von 384 Pixeln Breite zur freien Verfügung. Auf dieser „Ablagefläche“ deponieren wir gleich bei Programmstart drei Ausschnitte des Hintergrundbildes, welche die „Laufbahn“ von Knut abdecken (F) (vgl. Abbildung 11).

Damit auf dem Display nur immer ein einziges Bild von Knut sichtbar ist, müssen wir das vorangegangene Knut-Bild restlos abdecken. Zu diesem Zweck kopieren wir je nach Knuts Position den Ausschnitt 1, 2 oder 3 aus dem unsichtbaren Randbereich an die Ursprungsstelle des Hintergrundbildes zurück (G) und überdecken damit das aktuelle Knut-Bild vollständig. Dann „legen“ wir das Folgebild von Knut auf diesen Ausschnitt (H). Damit dieser Bildwechsel schnell und ohne sichtbare Zwischenschritte abläuft, bereiten wir diesen Vorgang im „unsichtbaren“ AVR vor und kopieren dann den vorbereiteten Bildausschnitt mit dem neuen Knut vom AVR ins RVR (I). Auf diese Weise läuft Knut durch die arktische Landschaft.

Beispiel-Code 10: Ganzes Programm-Script von KNUT

```

;-----
; Titel      : Poti-Anwendung Knut   View 1
; File       : KNUT_001.EVS
;-----
; Compiler   : eigerScript
;
; System     : eigergraphics.com; FOXS embedded computer
;
; Beschreibung: Eisbär Knut (Bild = 160 x 100 Pixel) läuft durch
;              die arktische Landschaft. Die Knuts Tempo
;              ist mit dem Poti (Eingang PA0) regulierbar.
;
; Version    : Initialversion: 15.4.2008
;
; Autor      : Christoph Angst

```

¹ **AVR und RVR:** der FOX embedded Computer des eigerPanels 57 verfügt über zwei Videospeicher, das RVR (Refresh Video Ram) und das AVR (Accessible Video Ram). Der Inhalt des RVR wird auf dem Display angezeigt, während das AVR nicht direkt mit dem Display verbunden ist; sein Inhalt bleibt unsichtbar. Ohne spezielle Anweisung bedient der Graphic-Controller jeweils beide Videospeicher parallel mit der gleichen Bildinformation. Mit dem Befehl `Display.Prepare()` gelangt die Bild-Information nur in das „unsichtbare“ AVR und mit `Display.Direct()` wird nur das RVR zusammen mit dem Display bedient. Der Inhalt des AVR erscheint erst auf dem Display, wenn man die dort gespeicherte Information mit dem Befehl `Display.Show()` – bzw. einen vorgängig mit `Load.Geometry_XYWH()` bestimmten Ausschnitt davon mit `Display.ShowWindow()` – ins RVR kopiert.

```

;
;-----
; (c) 2005-2007      S-TEC electronics AG, CH-6300 Zug; 041 754 50 10
;-----

EIGERPROJECT 'KNUT' ; Projektbezeichnung: 1.Teil des EVI-Dateinamens
EIGERVIEW 1 ; Viewnummer: 2.Teil des EVI-Dateinamens: KNUT_001.EVI

IMPORT          'DEF_eVM_OpCodes_0$70.h'           ; Token
IMPORT          'DEF_eVM_Registers_0$70.h'         ; Register
FUNCLIB        'DEF_eVM_Functions_0$70.lib'        ; Funktionsbibliothek

INCLUDEFILE     'DEF_eiger_Colors_0$70.INC' ; Farbdefinitionen
INCLUDEFILE     'DEF_eiger_Types_0$70.INC' ; eiger Definitionen
INCLUDEFILE     'DEF_eiger_StackMachine_0$70.INC' ; Stack Machine

; D E K L A R A T I O N E N _____

STRING [60] Titel.$ = 'Poti-Anwendung, KNUT_001.EVS'
STRING [60] Titel_Right.$ = 'erstellt: 15.4.2008' ; Titelleiste
                    rechts

STRING [40] BildDatei.$

INTEGER HA_X.I
INTEGER Zaehler.I = 0x30
INTEGER AnalogInputPoti.I
INTEGER KnutX.I = 100 ; X-Position von Knut
INTEGER KnutY.I = 300 ; Y-Position von Knut
CONST Schrittlaenge = 10 ; Schrittlänge für Knut (jedes neue
                        Bild rückt so viele Pixel nach rechts)

; S U B R O T I N E N _____

; Titelleiste schreiben .....

SUB Draw_Titel

    Fill.LabelParameter ( Titel_Style )
    Label.Text ( Titel.$ ) ; Text für linke Titelseite

    Fill.LabelParameter ( Titel_Right_Style )
    Label.Text ( Titel_Right.$ ) ; Text für rechte Titelseite
ENDSUB

SUB MoveLittleKnut
; Bildabfolge (Knut1.EGI - Knut6.EGI)-----

IF Zaehler.I == 0x36 THEN ; 0x36 = 6 für Knut6
    Zaehler.I := 0x31 ; 0x31 = 1 für Knut1
ELSE
    Zaehler.I := Zaehler.I + 1; nach Knut1.EGI kommt
    Knut2.EGI etc.
ENDIF

; Zusammensetzen des neuen Bildnamens inkl. Pfad -----
BildDatei.$ := 'C:\\KNUT\\PICT\\Knut' ; = Knut
Str.AddChar ( BildDatei.$ , Zaehler.I ); = z.B. Knut1
Str.Concat ( BildDatei.$ , '.EGI' ) ; = z.B. Knut1.EGI

```

```

; Alte Position von Knut mit Hintergrundausschnitt ersetzen ----
Display.Prepare ( ) ; Alles unsichtbar im AVR vorbereiten
IF KnutX.I < 170 THEN ; Knut ist im ersten Drittel
  Load.Geometry_XYWH ( 700 , 0 , 320 , 100 ) ; Ausschnitt des
  kopierten Drittels
  HA_X.I := 0
G ( ELSIF KnutX.I < 340 THEN ; Knut ist
  im zweiten Drittel
  Load.Geometry_XYWH ( 700, 100 , 320 , 100 ) ; Ausschnitt des
  kopierten Drittels
  HA_X.I := 160
  ELSE ; Knut ist im dritten Drittel
  Load.Geometry_XYWH ( 700 , 400 , 320 , 100 ) ; Ausschnitt des
  kopierten Drittels
  HA_X.I := 320
ENDIF
Load.Pos_X2Y2 ( HA_X.I , KnutY.I ) ; EinfügeOrt für Ausschnitt

Display.CopyWindow ( ) ; Ausschnitt einfügen
Load.Pos_X1Y1 ( KnutX.I , KnutY.I ) ; Position für neues
Knutbild
H ( File.Read_EGI ( BildDatei.$ ) ; neuer Knut wird auf
  Hintergrundbild gesetzt
  Load.Geometry_XYWH ( HA_X.I , KnutY.I , 320 , 100 ) ; AVR-
  Ausschnitt bestimmen

I ( Display.ShowWindow ( ) ; von AVR zu RVR

B ( InOut.Read_ADC ( 0 , AnalogInputPoti.I ) ; Werte zw. 0 und
  1023
  IF AnalogInputPoti.I < 5 THEN ; verhindert Poti-Wert =
  0
    AnalogInputPoti.I := 5
  ENDIF
  C ( CallSubroutine ( Timer )

; Vorwärtsbewegung -----
E ( IF KnutX.I == 640 THEN
  KnutX.I := 0 ; Knut startet auf der linken
  Seite X = 0
  ELSE
  D ( KnutX.I := KnutX.I + Schrittlänge ;bestimmt die
  Vorwärtsbewegung
  ENDIF
ENDSUB

SUB Timer
A ( Timer.InstallLocal ( 0 , MoveLittleKnut )
  Timer.Load ( 0 , AnalogInputPoti.I )
  Timer.StartSingle ( 0 )
ENDSUB

F ( SUB CopyWindow ; Ausschnitte von Hintergrundbild in unsichtbare
  Displayzone kopieren (X2 = 800)

  Load.Geometry_XYWH ( 0 , KnutY.I , 320 , 100 ) ;
  Hintergrundbild Ausschnitt 1.Drittel
  Load.Pos_X2Y2 ( 700 , 0 ) ; Kopierort für
  Ausschnitt 1. Drittel
  Display.CopyWindow ( )

  Load.Geometry_XYWH ( 160 , KnutY.I , 320 , 100 ) ;
  Hintergrundbild Ausschnitt 2.Drittel
  Load.Pos_X2Y2 ( 700 , 100 ) ; Kopierort
  für Ausschnitt 2. Drittel
  Display.CopyWindow ( )

```

```

Load.Geometry_XYWH ( 320, KnutY.I , 320 , 100 )      ;
Hintergrundbild Ausschnitt 3.Drittel
Load.Pos_X2Y2 ( 700 , 400 )                        ; Kopierort
für Ausschnitt 3. Drittel
Display.CopyWindow ( )

ENDSUB

; H A U P T P R O G R A M M _____
BEGINVIEW
EVE.Init ( )                                     ; EVE ANNA initialisieren
Load.Pos_X1Y1 ( 0 , 0 )                           ; linke obere Ecke
File.Read_EGI ( 'C:\\KNUT\\PICT\\ARKTIS.EGI' )

CallSubroutine ( Draw_Titel )
CallSubroutine ( CopyWindow )
CallSubroutine ( MoveLittleKnut )

Timer.TableEnable

LOOP
ENDLOOP
ENDVIEW

; S T Y L E S _____

SUB Styles

Titel_Style:
INLINEWORDS (0)           ; entspricht eI.Pos_Xl
INLINEWORDS (0)           ; entspricht eI.Pos_Yl
INLINEWORDS (640)         ; entspricht eI.Width
INLINEWORDS (18)          ; entspricht eI.Height
INLINEWORDS (20)          ; entspricht eI.SpaceLeft
INLINEWORDS (8)           ; entspricht eI.SpaceRight
INLINEWORDS (0)           ; entspricht eI.HorizontalAdjust
INLINEWORDS (0)           ; entspricht eI.VericalAdjust
INLINEWORDS (black)       ; entspricht eI.FillColor
INLINEWORDS (black)       ; entspricht eI.BackColor
INLINEWORDS (black)       ; entspricht eI.LineColor
INLINEWORDS (lightyellow) ; entspricht eI.TextColor
INLINEWORDS (Pos_left)    ; entspricht eI.Position
INLINEWORDS (Orientation_0deg) ; entspricht eI.Orientation
INLINEWORDS (normal)      ; entspricht eI.Appearance
INLINEWORDS (no_border)   ; entspricht eI.BorderStyle
INLINEWORDS (Font_System_9bd) ; entspricht eI.FontNumber
INLINEWORDS (silver)      ; entspricht eI.BackgroundColor

Titel_Right_Style:
INLINEWORDS (0)           ; entspricht eI.Pos_Xl
INLINEWORDS (0)           ; entspricht eI.Pos_Yl
INLINEWORDS (640)         ; entspricht eI.Width
INLINEWORDS (18)          ; entspricht eI.Height
INLINEWORDS (20)          ; entspricht eI.SpaceLeft
INLINEWORDS (8)           ; entspricht eI.SpaceRight
INLINEWORDS (0)           ; entspricht eI.HorizontalAdjust
INLINEWORDS (0)           ; entspricht eI.VericalAdjust
INLINEWORDS (transparent) ; entspricht eI.FillColor
INLINEWORDS (black)       ; entspricht eI.BackColor
INLINEWORDS (black)       ; entspricht eI.LineColor
INLINEWORDS (lightsalmon) ; entspricht eI.TextColor
INLINEWORDS (Pos_right)   ; entspricht eI.Position
INLINEWORDS (Orientation_0deg) ; entspricht eI.Orientation
INLINEWORDS (normal)      ; entspricht eI.Appearance
INLINEWORDS (no_border)   ; entspricht eI.BorderStyle
INLINEWORDS (Font_System_9bd) ; entspricht eI.FontNumber

```

```

    INLINEWORDS (silver) ; entspricht eI.BackgroundColor
ENDSUB

```

4.4 Licht ein- und ausschalten

eigerScript-Methode:

InOut.DigitalOutputDriver (Kanal, Funktion)

Eine einfache Anwendung dieses Befehls ist beispielsweise ein Lichtschalter. Mit der Funktion `Output_On` schalten wir das Licht ein, und mit der Funktion `Output_Off` schalten wir es aus. Das eigentliche Ein- und Ausschalten des Lichts wird oft durch einen Transistor übernommen, der vom eigerPanel 57 mit einer Schaltspannung von 3.3 V angesteuert wird (vgl. Abbildung 12).

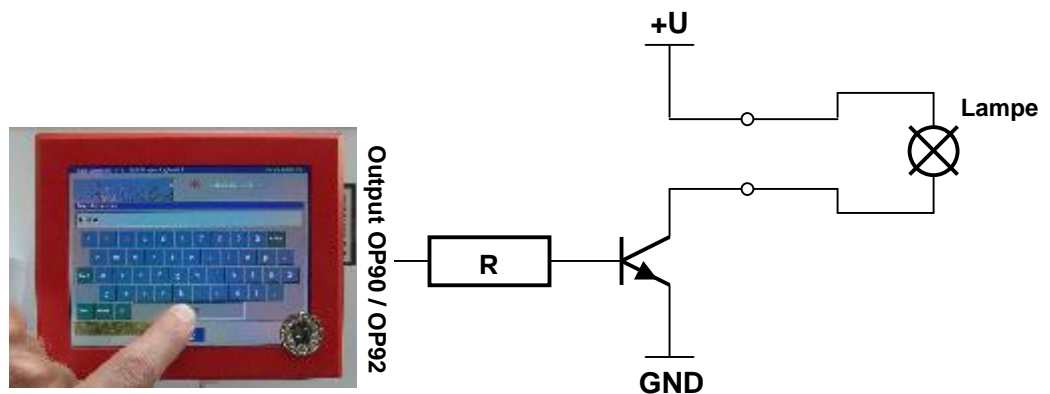


Abbildung 12: Beispiel-Schema für eine Lichtschaltung mit dem eigerPanel 57.

Ein Beispiel für ein solches Schaltprojekt ist das eigerProjekt LEDC. Sie finden dieses Projekt mit dem Programmcode in der eigerDemoCD unter dem Verzeichnis „Analog- u Digitalsignale einlesen u ausgeben“ oder auf der Download-Seite von www.eigergraphics.com unter Application Notes. Die entscheidenden Programmzeilen sind auch im folgenden Beispiel-Code 11 wiedergegeben.

Beispiel-Code 11: Subroutine für Licht-Button ruft die Subroutine auf welche das Licht ein- und ausschaltet.

```

SUB Draw_LightButton_down
  Fill.LabelParameter(LightButton_down_Style)
  Label.Text(Button_ON.S)
  CallSubroutine(Licht_ON_OFF)
ENDSUB

SUB Licht_ON_OFF
  Load.Pos_X1Y1(240,180)

  IF eI.P90_Output == 0 THEN
    ; "Licht einschalten"
    InOut.DigitalOutputDriver ( OP90,Output_On )
  ELSE ; d.h. wenn eI.P90_Output == 1
    ; "Licht ausschalten"

```

```

    InOut.DigitalOutputDriver ( OP90,Output_Off )
ENDIF
ENDSUB

```

Erklärung zu Beispiel-Code 11: Auf dem Display ist ein Lichtschalter abgebildet, welchem ein HotSpot hinterlegt ist (der entsprechende Code ist hier nicht gezeigt). Drückt man auf diesen Lichtschalter, dann wird aufgrund des „Down-Ereignisses“ die Subroutine `SUB Draw_LightButton_down` aufgerufen. Während mit dem `LightButton_down_Style` der Lichtschalter als „eingedrückt“ gezeichnet wird, ruft diese Subroutine auch die `SUB Licht_ON_OFF` auf. Diese Subroutine regelt dann die Ausgabe der Schaltspannung über den Output OP90.

Die Wirkung von Beispiel-Code 11 ist noch billiger zu haben. Der Wechsel zwischen `Output_On` und `Output_Off` können wir mit einer einzigen Funktion ersetzen: `Output_Toggle`. Die entsprechenden Programmzeilen sind im Beispiel-Code 12 wiedergegeben.

Beispiel-Code 12: Ein und Ausschalten der Schaltspannung mit der Funktion `Output_Toggle`.

```

SUB Licht_ON_OFF
; "Licht ein- bzw. ausschalten"
    InOut.DigitalOutputDriver ( OP90,Output_Toggle )
ENDSUB

```

4.5 Digital Output OP92 – Spielereien mit dem Buzzer

Mit dem Digital Output ist der Buzzer verbunden (vgl. Abbildung 7). Dieser ertönt, sobald bzw. solange wir den Digital-Ausgang mit Hilfe des Befehls

`InOut.DigitalOutputDriver(OP92,Funktion)`

ansteuern.

Im folgenden Programmbeispiel BUZZ erzeugen wir anhand verschiedener Funktionen unterschiedliche Alarm-Sequenzen, beispielsweise den SOS-Ruf mit Morsezeichen. Gleichzeitig zum Tonzeichen gibt OP92 eine Spannung von 3.3 Volt aus.

Sie finden dieses Projekt mit dem Programmcode in der `eigerDemoCD` unter dem Verzeichnis „Analog- u Digitalsignale einlesen u ausgeben“ oder auf der Download-Seite von www.eigergraphics.com unter Application Notes.

Das View-Layout von BUZZ (Abbildung 8) enthält vier Buttons. Diesen ist je eine der ersten vier Funktionen hinterlegt, die auf S. 8 beschrieben sind.

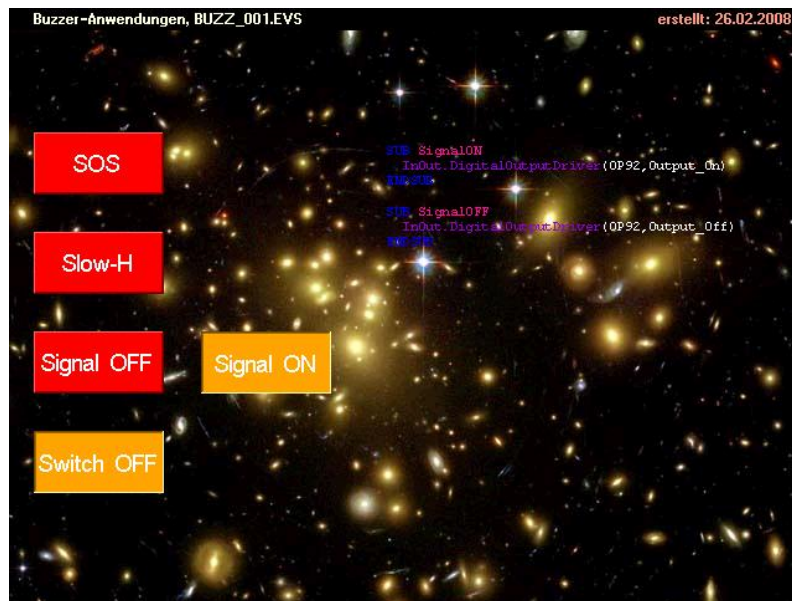


Abbildung 13: View-Layout der Buzzer-Anwendung

Beispielcode aus dem Programm BUZZ

Beispiel-Code 13: Code für die Ausgabe des SOS-Notrufs `... --- ...` (ElementGridP, ElementGridS, Punktlänge und Strichlänge sind im BUZZ-Script als Konstanten deklariert). Der SOS-Notruf wird durch den Down-Event des Buttons „SOS“ ausgelöst.

```

SUB Draw_Button_SOS_down
  IF eI.P92_Output == 0 THEN
    Load.Geometry_XYWH(Row1_X,Line1_Y,Button_W,Button_H)
    Fill.LabelParameter(Button_down_Style)
    Label.Text('SOS')
    CallSubroutine(Draw_Label_SOS)
    CallSubroutine(Load_CodeSOS)
    CallSubroutine(SOS_Signal)
    Serial.Rx_MonitorSlot_InstallLocal(COM2,8,Draw_Button_SOS_up,0x00,0x3F)
  ENDIF
ENDSUB

SUB SOS_Signal
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(ElementGridP) ; Zeit für 1.Pkt inkl. Pause
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(ElementGridP)
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(BuchstabenabstandP)
  CallSubroutine(Strich)
  Delay.Wait_n_ms(ElementGridS)
  CallSubroutine(Strich)
  Delay.Wait_n_ms(ElementGridS)
  CallSubroutine(Strich)
  Delay.Wait_n_ms(BuchstabenabstandS)
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(ElementGridP)
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(ElementGridP)
  CallSubroutine(Punkt)
  Delay.Wait_n_ms(ElementGridP)
ENDSUB

```

```

SUB Punkt
  eI.P92_Pulse_Count := 1
  eI.P92_Time_ON := Punktlänge
  eI.P92_Time_OFF := 0
  InOut.DigitalOutputDriver(OP92,Output_Pulse)
ENDSUB

SUB Strich
  eI.P92_Pulse_Count := 1
  eI.P92_Time_ON := Strichlänge
  eI.P92_Time_OFF := 0
  InOut.DigitalOutputDriver(OP92,Output_Pulse)
ENDSUB

```

Beispiel-Code 14: Code für die Ausgabe des Buchstabens „H“ als Morsezeichen (Button „SlowH“)

```

SUB Draw_Button_SlowH_down
  IF eI.P92_Output == 0 THEN
    Load.Geometry_XYWH(Row1_X,Line2_Y,Button_W,Button_H)
    Fill.LabelParameter(Button_down_Style)
    Label.Text('SlowH')
    CallSubroutine(Draw_Label_SlowH)
    CallSubroutine(Load_CodeSlowH)
    CallSubroutine(SlowH_Signal)
  ENDIF
ENDSUB

SUB SlowH_Signal
  eI.P92_Pulse_Count := 4
  eI.P92_Time_ON := Punktlänge
  eI.P92_Time_OFF := 1000
  InOut.DigitalOutputDriver(OP92,Output_Pulse)
  Delay.Wait_n_ms(4000)
ENDSUB

```

Beispiel-Code 15: Die Subroutine „SignalOFF“ wird durch den Down-Event des Buttons „Signal OFF“ aktiviert.

```

SUB SignalOFF
  InOut.DigitalOutputDriver(OP92,Output_Off)
  CallSubroutine(Erase_CodeOnOff)
  CallSubroutine(AlleButtons)
ENDSUB

```

Beispiel-Code 16:

```

SUB SignalToggle
  InOut.DigitalOutputDriver(OP92,Output_Toggle)
  IF eI.P92_Output == 1 THEN
    CallSubroutine(Load_CodeToggle) ; Code auf dem Display anzeigen
  ELSE
    CallSubroutine(Erase_CodeToggle) ; Code vom Display löschen
  ENDIF

  CallSubroutine(AlleButtons) ; Buttons entsprechend Signalausgabe
  darstellen
ENDSUB

```

4.6 Display-Hinterleuchtung dimmen („Standby-Modus“)

Das eigerPanel 57 ist mit einem Energiebedarf von maximal 4,3 Watt äusserst sparsam. Die LED-Hinterleuchtung hat bei voller Leuchtstärke eine Leistung von rund 1.2 Watt und ist somit im eigerPanel 57 das Teil mit dem höchsten Energieverbrauch. Die Leistung des embedded Computer FOX 57 allein liegt bei nur 1 Watt. Wenn das eigerPanel 57 gerade nicht bedient werden muss, kann man also mit reduzierter oder ausgeschalteter LED-Hinterleuchtung den Energieverbrauch deutlich senken. Zudem wird dadurch die Lebensdauer der LED entsprechend ausgedehnt.

Die Methode für die Steuerung der LED-Hinterleuchtung beim eigerPanel 57 ist auf Seite 11 beschrieben:

```
InOut.PWM_Out(OP72,Wert zw. 0 .. 1200)
```

Ein konkretes Anwendungsbeispiel für diese Methode ist das eigerProjekt DIMM. Bei Berührung des Displays fährt die LED-Hinterleuchtung langsam hoch, und bei Betätigung der „Mond-Taste“ löscht das Licht langsam aus (Abbildung 14). Sie finden dieses Projekt mit dem Programmcode in der eigerDemoCD unter dem Verzeichnis „Analog- u Digitalsignale einlesen u ausgeben“ oder auf der Download-Seite von www.eigergraphics.com unter Application Notes. Die entscheidenden Programmzeilen sind auch im folgenden Beispiel-Code 17 wiedergegeben.



Abbildung 14: ScreenShot der Beispielanwendung DIMM. Bei Berührung der Bildschirmfläche leuchtet das Display auf, während es zum Standby-Modus abdunkelt, wenn man auf das Mond-Piktogramm drückt.

Beispiel-Code 17: Im Beispielprogramm DIMM ruft ein ContinuousTimer kontinuierlich die Subroutine Dimmen auf, bis die Leuchtstärke entweder maximal (= 1200) oder minimal (= 0) ist. Die Leuchtstärke nimmt in 5er-Schritten zu oder ab. Auf diese Weise wird das Display langsam heller bzw. dunkler.

```
SUB Dimmen
  IF DimmRichtung.I == 1 THEN                ; -> heller
    Leuchtstaerke.I := Leuchtstaerke.I + 5
    IF Leuchtstaerke.I >= 1200 THEN
      Leuchtstaerke.I := 1200
      Timer.Stop ( 0 )
    ENDIF
  ELSIF DimmRichtung.I == -1 THEN           ; -> dunkler
    Leuchtstaerke.I := Leuchtstaerke.I - 5
    IF Leuchtstaerke.I <= 0 THEN
      Leuchtstaerke.I := 0
      Timer.Stop ( 0 )
    ENDIF
  ENDIF

  InOut.PWM_Out ( OP72 , Leuchtstaerke.I )

ENDSUB
```

