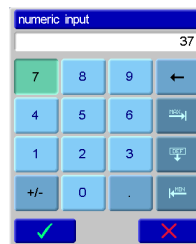


Application Note 14

Controls als Popup



Christoph Angst

© S-TEC electronics AG, CH-6300 Zug

s-tec@bluewin.ch

www.s-tec.ch

www.eigergraphics.com

Erstellt: 31. März 2009

Update: 1. April 2009

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Einleitung	3
Numeric Keypad (Zahlenblock)	4
Numeric NUMKeypad in mein Projekt integrieren.....	5
Verweis auf CTRL_NUM1.EVS in meiner Projekt-View	5
Parameter für Numeric NUMKeypad definieren	5
Callback-Routinen für NUMKeypad formulieren.....	7
NUMKeypad zeichnen	8
Spezialitäten aus dem Includefile CTRL_NUM1.EVS	9
Display.Direct()	9

Einleitung

In dieser Application-Note lernen Sie diverse bereits vorgefertigte Controls kennen, die Sie in Ihre Anwendung als Popup-Eingabefenster einbauen und nach Belieben abändern können. Die hier vorgestellten Controls stammen aus der Demo-Applikation TG12, die Sie als Bestandteil des eigerPanel 57 DemoKits erhalten haben bzw. von der Download-Seite unserer Homepage www.eigergraphics.com herunterladen können.



In dieser Application Note wird manchenorts auf das DemoProjekt **TG12** verwiesen. Das **TG12** gehört zum Zubehör des eigerPanel 57 DemoKits. Sie können zudem die aktuellste Version des **TG12** jederzeit von unserer Download-Seite auf www.eigergraphics.com frei herunterladen, auf Ihrem eigerPanel ansehen und vieles entdecken.

Dieses Dokument erklärt zur Zeit (April 2009) nur die Verwendung des NUM1 Numeric Keypads. Nach und nach finden Sie hier auch die anderen Keypads, die Sie auch im Demo-Application TG12 (Pfad: *System > Controls*). Das Prinzip ist immer dasselbe. Wenn Sie den Einsatz des NUM1 Numeric Keypads einmal durchstudiert haben, sind Sie fähig, auch die anderen Keypads in Ihr Projekt zu integrieren.

Wenn Sie Fragen haben, schreiben Sie uns oder rufen Sie uns an.

Christoph Angst, S-TEC electronics AG
Unterägeri, den 1. April 2009

Numeric Keypad (Zahlenblock)

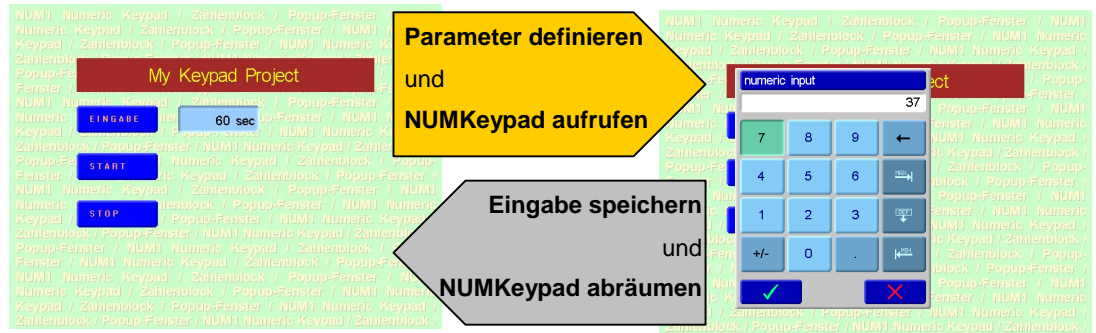
Abbildung 1:
Numeric
NUMKeypad
(Zahlenblock) als
Popup-Fenster.
Screenshot aus dem
DemoProgramm
TG12 (Pfad: *System*
> *Controls* > *NUM1*
Numeric
NUMKeypad)



Zu dieser Application Note gehört das Beispiel-Projekt „NUM1“. Sie finden dieses Projekt in der CD, die Sie mit dem eigerPanel 57 DemoKit erhalten haben. Es liegt auch auf der Downloadseite unserer Homepage www.eigergraphics.com bei der entsprechenden Application Note zum Gratis-Download bereit.

Numeric NUMKeypad in mein Projekt integrieren

Abbildung 2: Das NUMKeypad wird als Popup-Fenster über die View. Dem Keypad werden gewisse Parameter mitgegeben. Beim Abräumen das NUMKeypad den Eingabewert in einer Variable zurück.



Verweis auf CTRL_NUM1.EVS in meiner Projekt-View

Das NUMKeypad-Popup liegt als Datei „CTRL_NUM1.EVS“ bereit. Sie wird mit dem Schlüsselwort **INCLUDEFILE** ins Projekt integriert (Beispiel-Code 1).

Die NUMKeypad-Includedatei enthält kein Hauptprogramm und stellt deshalb auch keine eigenständige View dar. Sie besteht im wesentlichen aus Variablen- und Konstanten-Definitionen sowie Subroutinen, die für den Aufbau und die Funktionalität des NUMKeypads verantwortlich sind. Diese Variablen, Konstanten und Subroutinen können in der Projektview verwendet bzw. aufgerufen werden, wie wenn sie direkt im View-Script geschrieben worden wären.

Beispiel-Code 1: Integrations-Zeile für das Numeric-NUMKeypad Control, im Deklarationen-Teil der Projektview bzw. in der EPR-Projektdatei.

```
INCLUDEFILE      'CTRL_NUM1.EVS'          ; Numeric-NUMKeypad
Control
```

Parameter für Numeric NUMKeypad definieren

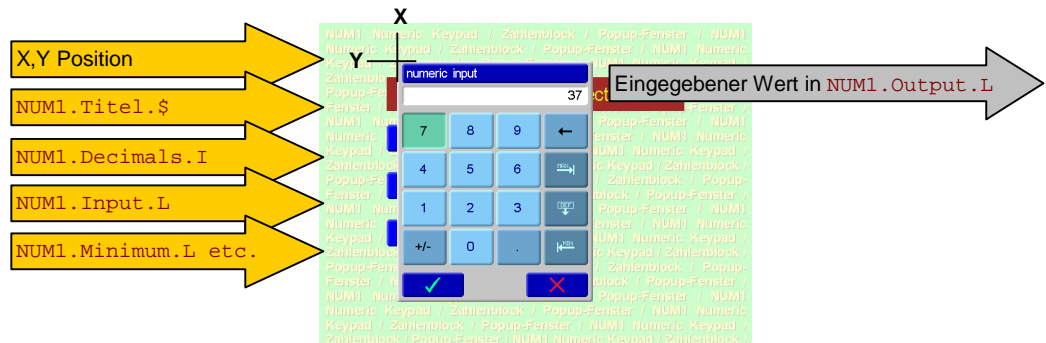
Parameter, die dem NUMKeypad-Aufruf mitgegeben werden (vgl. Abbildung 3 und Beispiel-Code 2, erster Teil):

- X- und Y-Position des NUMKeypads auf dem Display
→ Register `eI.Pos_X1` und `eI.Pos_X2`
- Titel für das NUMKeypad (z.B. „numeric input“)
→ String-Variable `NUM1.Titel.$`



- Anzahl Dezimalstellen für das Eingabefeld
→ Integer-Variable `NUM1.Decimals.I`
- Startwert für das Eingabefeld (z.B. 37)
→ Long-Variable `NUM1.Input.L`
- Minimum-, Maximum- und Default-Wert
→ Long-Variablen `NUM1.Minimum.L`, `NUM1.Maximum.L` und `NUM1.Default.L`

Abbildung 3:

Parameter, die dem NUMKeypad „mitgegeben“ werden müssen, und Wert, den das NUMKeypad in einer Long-Variablen zurückgibt.



Subroutinen in der View, welche vom NUMKeypad aus aufgerufen werden (vgl. Beispiel-Code 2, zweiter Teil):

- Funktion, die bei Betätigung der OK-Taste  ablaufen soll → die Adresse (Name) der Subroutine wird in die Variable `NUM1.CALLBACK_OK.L` geladen.
- Funktion, die bei Betätigung der CANCEL-Taste  ablaufen soll → die Adresse der Subroutine wird in die Variable `NUM1.CALLBACK_CANCEL.L` geladen.
- Funktion, die bei Betätigung irgendeiner Nummern-Taste ablaufen soll → Adresse der Subroutine wird in die Variable `NUM1.CALLBACK_NUM.L` geladen.
- Funktion, die bei Betätigung irgendeiner Taste des NUMKeypads ablaufen soll → die Adresse der Subroutine wird in die Variable `NUM1.CALLBACK_ANYKEY.L` geladen. Drücke ich also beispielsweise auf die Taste 7, so wird die Zahl 7 ins Eingabefeld aufgenommen und gleichzeitig die in `NUM1.CALLBACK_ANYKEY.L` angegebene Subroutine abgearbeitet, z.B. mit dem Buzzer ein Beep-Signal ausgegeben.



Die Long-Variablen (`NUM1.CALLBACK_OK.L` etc.) für die Subroutinen-Adressen sind im Includefile `CTRL_NUM1.EVS` bereits vordefiniert und dort vorerst mit dem Wert -1 (d.h. ohne Funktion) belegt. Mit dem Befehl `Load.CallBackAddress()` wird in der Long-Variable die physikalische Speicheradresse der zugewiesenen Subroutine gespeichert.

Am einfachsten ist es, wenn Sie die Namen dieser Long-Variablen nicht verändern. Hingegen liegt es in Ihrer Freiheit, Namen und Inhalt der Subroutinen selbst zu bestimmen, welche mit dem OK-, Cancel-, Num- oder irgendeinem („Anykey“) Button aufgerufen werden sollen.

Beispiel-Code 2: Subroutine mit den Parametern, die dem NUMKeypad mitgegeben werden.

```

SUB Load_NUMKeypad                ; Parameter laden und NUMKeypad
aufrufen

    Load.Pos_X1Y1 ( NUM1_X1 , NUM1_Y1 )
    NUM1.Titel.$      := 'Zeit für Countdown eingeben [sec]'
    NUM1.Decimals.I   := 0
    NUM1.Input.L      := MyNumber.L
    NUM1.Minimum.L    := 1
    NUM1.Maximum.L    := 3600
    NUM1.Default.L    := 60

    Load.CallBackAddress ( NUM1.CALLBACK_OK.L, Countdown_OK )
    Load.CallBackAddress ( NUM1.CALLBACK_CANCEL.L, NumKey_CANCEL )
    Load.CallBackAddress ( NUM1.CALLBACK_NUM.L, NumKey_NUM )
    Load.CallBackAddress ( NUM1.CALLBACK_ANYKEY.L, Numkey_ANYKEY )

    CallSubroutine ( NUM1.DRAW )      ; NUMKeypad aufrufen (Sub in
CTRL_NUM1.EVS)

ENDSUB

```

Callback-Routinen für NUMKeypad formulieren

Die Subroutinen, die vom NUMKeypad aus aufgerufen werden, müssen nun noch ausformuliert werden. Aufgrund von Beispiel-Code 2 tritt nach Betätigung der NUMKeypad-Taste „OK“ die Subroutine in Aktion, die wir mit `SUB Countdown_OK` benannt haben. Gleichzeitig wird das Popup-Fenster wieder abgeräumt. Der auf dem Keypad eingegebene Wert steht nun in der Long-Variable `NUM1.Output.L` zur Verfügung. Im Beispiel-Code 3 wird dieser Wert von der `SUB Countdown_OK` verarbeitet und an die Subroutine der `SUB ShowCountdownTime` zur Bildschirm-Anzeige weitergegeben.

Beispiel-Code 3: Subroutinen, die vom NUMKeypad aus aufgerufen werden sollen.

```

; Callback-Funktionen für NUMKeypad .....
SUB Countdown_OK
  MyTime.L := NUM1.Output.L      ; Eingabe aus NUMKeypad
  übernehmen
  Str.Clear ( MyTime.$ )        ; Inhalt des Anzeige-Strings
  löschen
  Str.Cvt_Long ( MyTime.$ , MyTime.L , 0 )      ; vorbereiten für
  Countdown-Anzeige
  CallSubroutine ( ShowCountdownTime )
ENDSUB

SUB NumKey_CANCEL
ENDSUB

SUB NumKey_NUM
ENDSUB

SUB Numkey_ANYKEY
  eI.P92_Pulse_Count := 1
  eI.P92_Time_ON := 10          ; ms
  eI.P92_Time_OFF := 50        ; ms, Pause wenn Pulse_Count > 1
  InOut.DigitalOutputDriver( OP92, Output_Pulse )
ENDSUB

```

NUMKeypad zeichnen

Die Subroutine `SUB NUM1.DRAW`, welche das NUMKeypad zeichnet, liegt in der Includedatei `CTRL_NUM1.EVS`. Sie wird mit einem gewöhnlichen `CallSubroutine`-Befehl aufgerufen (vgl. Beispiel-Code 2, letzter Befehl).

Die Subroutine `SUB Load_NUMKeypad` (Beispiel-Code 2) und damit den Zeichenbefehl für das NUMKeypad rufen Sie in Ihrer View beispielsweise durch betätigen eines Buttons auf (vgl. Beispiel-Code 4).

Beispiel-Code 4: Subroutinen des Buttons für den NUMKeypad-Aufruf. Das NUMKeypad wird beim Up-Ereignis (erste Subroutine) aufgerufen. Gleichzeitig wird auch der Button-Up-Zustand des Buttons wiederhergestellt. Natürlich gehört auch die Installation des HotSpots dazu (Beispiel-Code 5).

```

; Button Spalte 1, Zeile 1 (Call NUMKeypad
  Popup).....
..

SUB ButtonCallNUMKeypad_LE      ; Leave-Ereignis
  Load.Geometry_XYWH ( ButtonSpalte1_X, ButtonZeile1_Y, Button_W,
  Button_H )
  Fill.LabelParameter ( Button_Up_Style )
  Label.Text ( 'Eingabe' )

```

```

ENDSUB

SUB ButtonCallNUMKeypad_UP      ; Up-Ereignis
  CallSubroutine ( ButtonCallNUMKeypad_LE )
  CallSubroutine ( Load_NUMKeypad )
ENDSUB

SUB ButtonCallNUMKeypad_DN      ; Down-Ereignis
  Load.Geometry_XYWH ( ButtonSpaltel_X, ButtonZeile1_Y, Button_W,
Button_H )
  Fill.LabelParameter ( Button_Down_Style )
  Label.Text ( 'Eingabe' )
ENDSUB

```

Beispiel-Code 5: Installationszeilen im Hauptprogramm für den HotSpot zum „ButtonCallNUMKeypad“.

```

  Load.Geometry_XYWH ( ButtonSpaltel_X, ButtonZeile1_Y, Button_W,
Button_H )
  eI.HotSpotTag := 1
  HotSpot.Install ( NIL, ButtonCallNUMKeypad_LE,
ButtonCallNUMKeypad_DN, ButtonCallNUMKeypad_UP )

```

Spezialitäten aus dem Includefile CTRL_NUM1.EVS

Display.Direct()

Das NUMKeypad erscheint als Popup-Fenster über dem Layout Ihrer View, ohne das Bild Ihrer View dadurch zu „zerstören“. Dies funktioniert, weil der FOX embedded Computer das Bild Ihrer View in zwei physisch getrennte Videospeichern geladen hat:

- „Hintergrundspeicher“ **AVR** (Accessible Video RAM). In diesem Speicher können Darstellungen aufgebaut werden, unabhängig von der Bildschirm-Anzeige.
- „Vordergrundspeicher“ **RVR** (Refresh Video RAM). Alles, was in diesen Videospeicher geladen wird, erscheint gleichzeitig auf dem Bildschirm.

Im Normalfall beziehen sich alle Anzeigebefehle auf beide Videospeicher, d.h. die Videodaten der auf dem Display sichtbaren View sind in beiden Speichern vorhanden.

Will ich meine View oder gewisse Graphiken nur in den Hintergrundspeicher speichern, so bediene ich mich des Befehls `Display.Prepare()`. Alle

nachfolgenden Zeichenbefehle beziehen sich dann nur auf den Hintergrundspeicher. Mit dem Befehl `Display.Direct()` lade ich in der Folge meine Graphiken exklusiv in den Vordergrundspeicher und zeige diese damit auch direkt auf dem Bildschirm an (vgl. Beispiel-Code 6).

Auf diese Weise wird das ganze Popup „direkt“ in den Vordergrundspeicher geladen und erscheint so auf dem Bildschirm.

Das „Abräumen“ des Popup ist dann einfach. Mit dem Befehl `Display.Show()` kopiere ich einfach meine unberührt gebliebene View in den Vordergrundspeicher bzw. auf den Bildschirm.

Als Spezialität gibt es noch den Befehl `Display.ShowWindow()`. Damit bringe ich nur einen Ausschnitt aus dem Hintergrundspeicher „nach vorne“. Der Ausschnitt durch dessen Position, Breite und Höhe bestimmt, welche in den Registern `eI.Pos_X1`, `eI.Pos_Y1`, `eI.Width` und `eI.Height` vordefiniert sind.

Beispiel-Code 6: Mit `Display.Direct()` ist gewährleistet, dass der Zahlenstring im Eingabefeld direkt auf dem Bildschirm erscheint. Der Inhalt des Hintergrundspeichers AVR bleibt dabei unbeeinflusst.

```
SUB NUM1.Draw_InputLine
  Display.Direct ( )                ; nur ins RVR ausgeben
  Transfer.HotSpotGeometry ( )
  Fill.LabelParameter ( NUM1.DisplayStyle ) ; Eigenschaften
  Label.Text ( NUM1.InputLine.$ )
ENDSUB
```

