

## Application Note 22

### Debug – Fehlern auf der Spur

**Christoph Angst**

Unterägeri, 24. Juni 2008

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



Dieses PDF-Dokument ist Teil der eigerDemoCD. Sie können es auch von der Downloadseite unserer Homepage [www.eigergraphics.com](http://www.eigergraphics.com) herunterladen.

---

## Inhalt

<b>1</b>	<b>Einleitung</b> .....	<b>3</b>
<b>2</b>	<b>HyperTerminal von Windows</b> .....	<b>4</b>
2.1	Anleitung zum Einrichten des Hyper Terminal von Windows XP .....	4
<b>3</b>	<b>eigerScript-Methoden der Debug-Class</b> .....	<b>7</b>
	Debug.Print_String(VarStr) .....	7
	Debug.Print_IntegerHex(str,VarInt) .....	8
	Debug.Print_LongHex(str,VarLong).....	9
	Debug.Print_SingleHex(str,VarSingle).....	10
	Debug.Print_Char(VarInt) .....	12
	Debug.Print_CRLF() .....	13
<b>4</b>	<b>Anhang</b> .....	<b>15</b>
4.1	ASCII-Zeichensatz.....	15

# 1 Einleitung

Fehler sind des Programmierers täglich Brot. Häufig sind es nur kleine Schreib- oder Formfehler, die den Programmablauf irgendwo auf Sand setzen. Deshalb ist es hilfreich, wenn man dem eigerPanel sozusagen über die Schulter schauen kann, wie es die einzelnen Programmschritte abarbeitet und wo es allenfalls hängen bleibt. Zu diesem Zweck dient einerseits das HyperTerminal, welches über die COM-Schnittstelle RS232 die Informationen vom eigerPanel entgegennimmt. Im Speziellen bieten auch die Debug-Methoden von eigerScript gute Dienste, mit denen Sie an wichtigen Schlüsselstellen ganz bestimmte Informationen abfragen oder aussagekräftige Marker setzen können. Dadurch kommen Sie den meisten Fehlern innert nützlicher Frist auf die Spur.

Die vorliegende Application Note soll

1. Sie in den HyperTerminal einführen, der auf jedem Windows-PC standardmässig installiert ist, und
2. Ihnen die eigerScript Methoden der Debug-Class erläutern.



## 2 HyperTerminal von Windows

### 2.1 Anleitung zum Einrichten des Hyper Terminal von Windows XP

Mit Hilfe des „Hyper Terminals“ von Windows XP können Sie mit Ihrem eigerPanel 57 kommunizieren. Dazu muss Ihr PC über die seriellen Schnittstellen mit dem eigerPanel 57 verbunden sein. Wenn Sie hierfür den Ausgang COM1 des eigerPanels 57 verwenden (vgl. Abbildung 1), wird der HyperTerminal auch zum Debug Terminal, auf welchem die Aktivitäten des eigerPanels laufend aufgezeichnet werden.

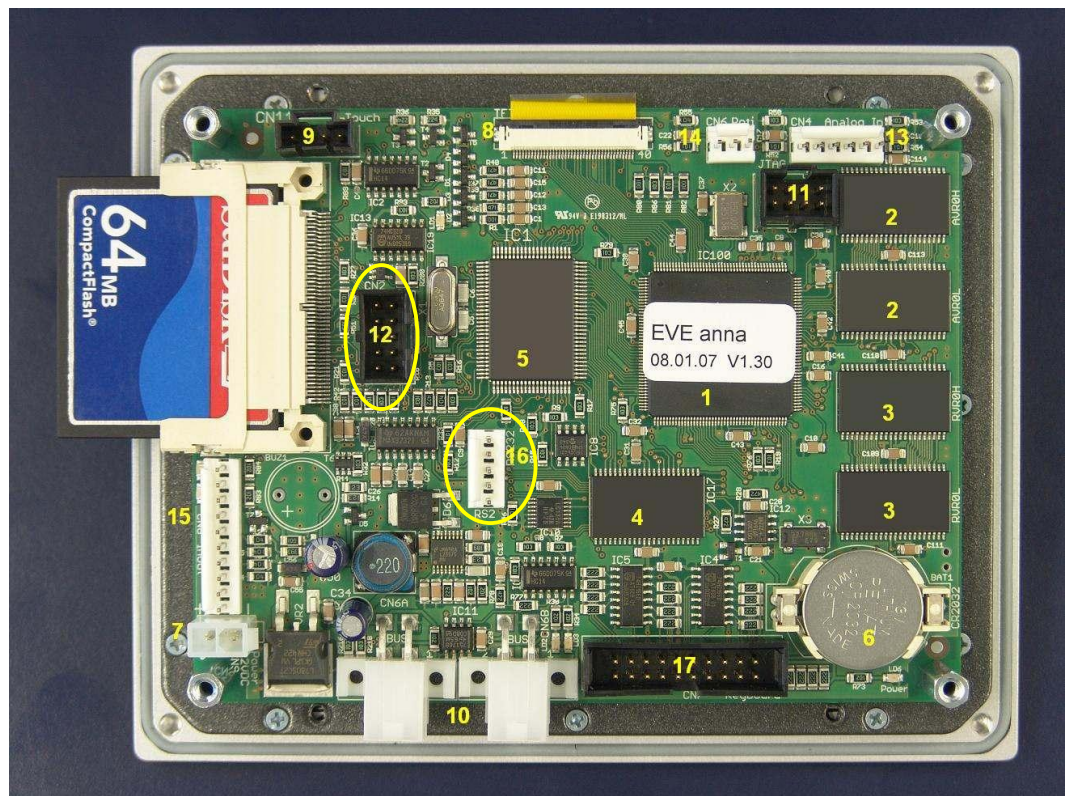


Abbildung 1: Komponenten des embedded Computers für das eigerPanel 57

- |   |   |
|---|---|
| 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 |
| 5. CPU (Micro-Prozessor)                            | 13. Analog-Eingänge   |
| 6. RTC-Batterie                                     | 14. Analogeingang Poti  |
| 7. Power Supply 12V (serielle Schnittstelle RS232)  | 15. Externe I/O   |
| 8. Anschluss Display VGA                            | 16. FOX-COM2 (UART2)  |
|   | 17. 16-KEY Keyboard Input   |

Das Einrichtungsfenster für den Hyper Terminal öffnen Sie über

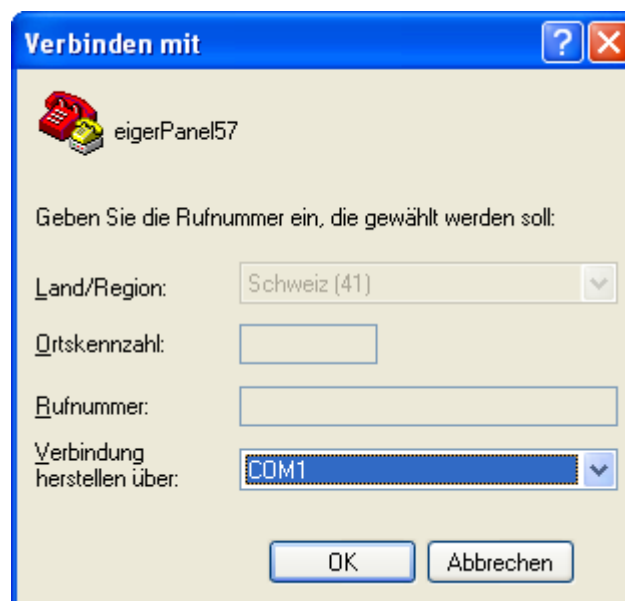
**Start > Programme > Zubehör > Kommunikation > HyperTerminal .**

Geben Sie für die Verbindung einen sinnvollen Namen ein, z.B. „eigerPanel57“ Abbildung 2.



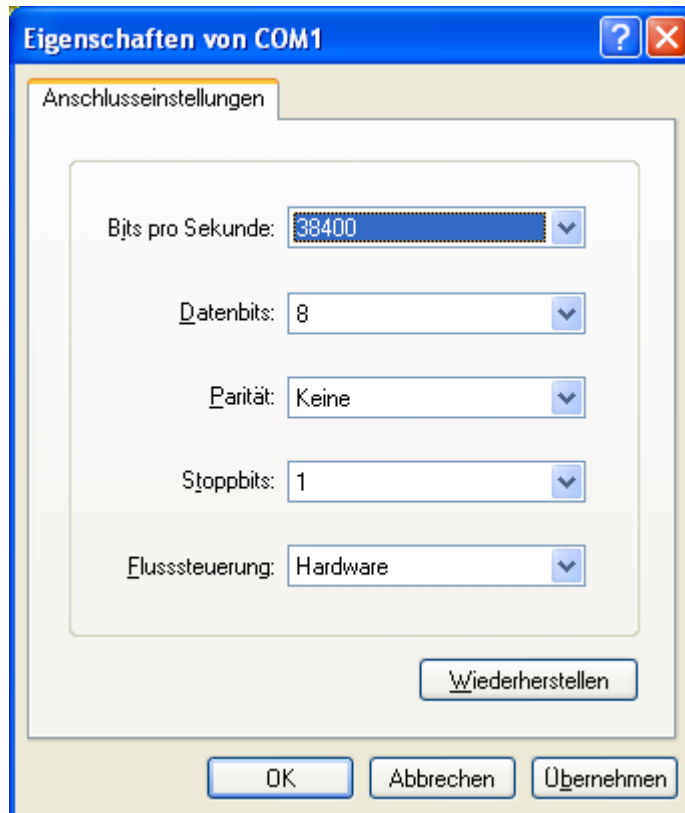
**Abbildung 2:** Eingangsfenster zur Installation einer neuen Verbindung über die serielle Schnittstelle.

Im nächsten Fenster muss nur die Schnittstelle Ihres PC gewählt werden, über welche Sie die Verbindung herstellen wollen, z.B. COM1 (Abbildung 3).





**Abbildung 3:** Wahl der seriellen Schnittstelle Ihres PC

Für alle weiteren Einstellungen dient das folgende Fenster (Abbildung 4). Unter anderem wählen Sie hier auch die Anzahl Bits pro Sekunde (Baudrate). Mit der Baudrate 38'400 liegen Sie in der Regel nicht schlecht. Diese muss mit der Geschwindigkeit übereinstimmen, welche Sie in Ihrem eigerScript-Programm mit dem Befehl `Serial.SetBaudrate()` festlegen (vgl. Application Note Serielle Schnittstellen RS-232 programmieren). Für die Debug-Funktion muss die Baudrate jedoch nicht speziell gesetzt werden.



**Abbildung 4:** Einstellungen für die neue Verbindung.

Damit ist die Verbindung installiert und bereit für die Kommunikation. Bei aktiver Verbindung (Einstellung in der Symbolleiste:  ) wird alles, was Sie im HyperTerminal schreiben direkt ans eigerPanel57 weitergegeben.

Wenn Sie diese Einstellungen speichern, können Sie in Zukunft den HyperTerminal für Ihre Verbindung „eigerPanel57“ über

**Start > Programme > Zubehör > Kommunikation > HyperTerminal > eigerPanel57.ht**

aufzurufen.



Sollten Sie im Fenster des HyperTerminals nicht sehen, was Sie schreiben, dann liegt es wahrscheinlich an der aktuellen Konfiguration. Sie können dies regulieren unter Datei > Eigenschaften > Einstellungen > ASCII Konfiguration. Dort können Sie wählen, ob Sie die „einggegebenen Zeichen lokal ausgeben (lokales Echo)“ wollen oder nicht.

### 3 eigerScript-Methoden der Debug-Class

#### Debug.Print\_String(VarStr)

Mit der Methode `Debug.Print_String(VarStr)` wird eine Stringvariable ausgegeben. Der String wird im HyperTerminal in der letzten Zeile einfach hinten angefügt (vgl. Abbildung 5). Dies macht es manchmal schwierig, die Stringausgabe im Log-Fenster des HyperTerminals aufzuspüren. Um das Auffinden zu erleichtern, kann eine zweite Debug-Methode mit einem speziellen String in Hochkomma vorangestellt werden, welcher einen neuen Zeilenbeginn bewirkt: `'\r\n'` (vgl. Beispiel-Code 2).

**Beispiel-Code 1:** Subroutine mit der Methode `Debug.Print_String(VarStr)`

```

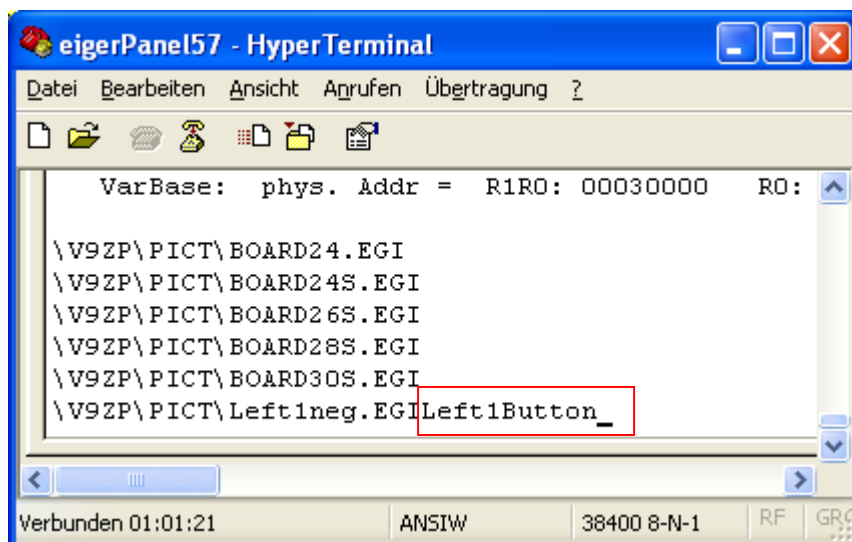
; Buttons zeichnen .....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Button.$ := 'Left1Button'

  Debug.Print_String ( Button.$ )

ENDSUB

```



**Abbildung 5:** Ausgabe der Stringvariablen `Button.$` im HyperTerminal nach Aktivierung der Subroutine `SUB Left1Button_down` (vgl. Beispiel-Code 1). Der mit `Debug.Print_String(VarStr)` ausgegebene String „Left1Button“ wird in der letzten Ausgabezeile einfach hinten angefügt.

**Beispiel-Code 2:** Subroutine wie in Beispiel-Code 1. Die Methode `Debug.Print_String(VarStr)` wird hier zweimal angewendet. Die Zeichenfolge `\r\n` (Return and New Line) wird vom Windows-HyperTerminal als Befehl zum Zeilenumbruch verstanden (vgl. Abbildung 6).

```

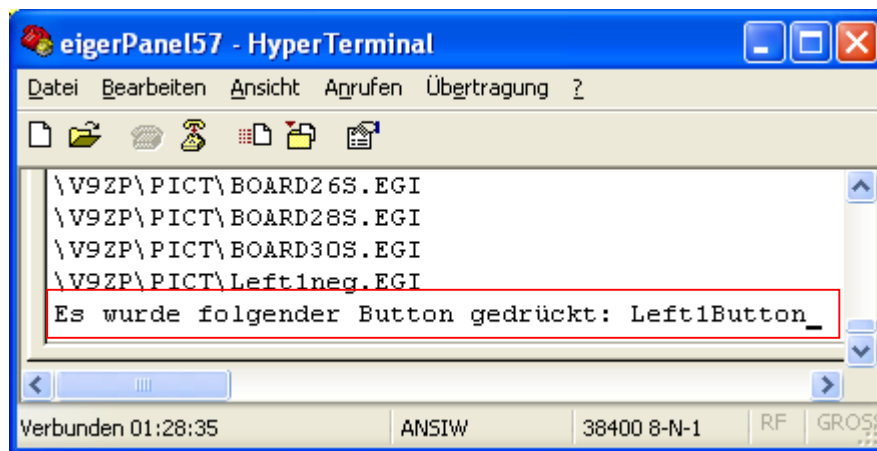
; Buttons zeichnen .....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Button.$ := 'Left1Button'

Debug.Print_String ('\r\nEs wurde folgender Button gedrückt: ')
Debug.Print_String ( Button.$ )

ENDSUB

```



**Abbildung 6:** Ausgabe der Strings im HyperTerminal nach Aktivierung der Subroutine `SUB Left1Button_down` (vgl. Beispiel-Code 2). Dank `\r\n` erhalten die beiden Strings eine neue Zeile.

### Debug.Print\_IntegerHex(str,VarInt)

Mit der Methode `Debug.Print_IntegerHex(str,VarInt)` wird auf dem HyperTerminal der Wert einer Integer<sup>1</sup>-Variablen angezeigt. Gleichzeitig kann der Variablen-Anzeige noch ein erklärender String vorangestellt werden (z.B. „Zähler = “, vgl. Beispiel-Code 3).

Angezeigt wird der 16 Bit Hex-Wert des Variablen-Inhalts (vgl. Abbildung 7). Um den Wert der Integer-Variablen als Dezimalzahl anzuzeigen, muss die Integer-Variablen erst in einen String umgewandelt werden und dann als String angezeigt werden (analog zu Beispiel-Code 6).

<sup>1</sup> Datentyp „Integer“: Ganzzahl, Wertebereich -32'768 bis +32'767.

**Beispiel-Code 3:** Subroutine mit der Methode `Debug.Print_IntegerHex(str,VarInt)`.

```

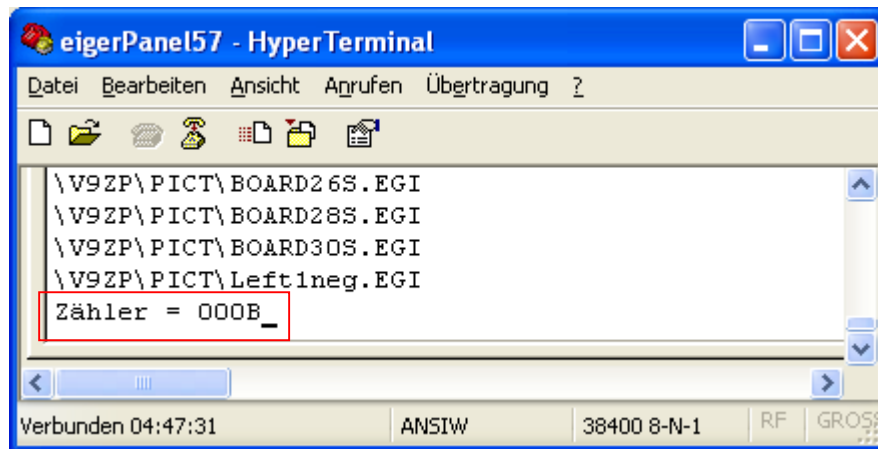
; Buttons zeichnen .....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Counter.I := 11

  Debug.Print_CRLF ( )
  Debug.Print_IntegerHex ( 'Zähler = ' , Counter.I )

ENDSUB

```



**Abbildung 7:** Ausgabe im HyperTerminal aufgrund der Methode `Debug.Print_IntegerHex('Zähler = ',Counter.I)`. Der Wert der IntegerVariablen „Counter.I“ wird als Hexadezimalwert ausgegeben (B entspricht der Dezimalzahl 11). Dank `Debug.Print_CRLF()` in eine neue Zeile geschrieben (vgl. Beispiel-Code 8).

### `Debug.Print_LongHex(str,VarLong)`

Mit der Methode `Debug.Print_LongHex(str,VarLong)` wird auf dem HyperTerminal der Wert einer Long<sup>2</sup>-Variablen angezeigt. Gleichzeitig kann der Variablen-Anzeige noch ein erklärender String vorangestellt werden (z.B. „Mittelwert = “, vgl. Beispiel-Code 4).

Angezeigt wird der 32 Bit Hex-Wert des Variablen-Inhalts (vgl. Abbildung 8). Um den Wert der Long-Variablen als Dezimalzahl anzuzeigen, muss die LongVariable erst in einen String umgewandelt werden und dann als String angezeigt werden (analog zu Beispiel-Code 6).

<sup>2</sup> Datentyp „Long“: Ganzzahl, Wertebereich -2'147'483'648 bis +2'147'483'647.

**Beispiel-Code 4:** Subroutine mit der Methode `Debug.Print_LongHex(str,VarLong)`, welche den Hex-Wert von `Mean.L` auf das HyperTerminal ausgibt (vgl. Abbildung 8).

```

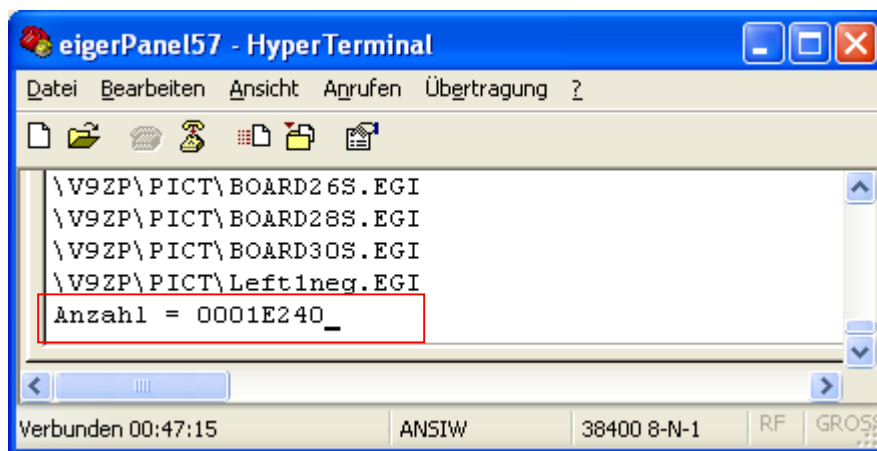
; Buttons zeichnen .....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Mean.L := 123456

  Debug.Print_CRLF ( )
  Debug.Print_LongHex ( 'Mittelwert = ' , Mean.L )

ENDSUB

```



**Abbildung 8:** Ausgabe im HyperTerminal aufgrund der Methode `Debug.Print_LongHex('Mittelwert = ',Mean.L)`. Der Wert der Long-Variablen „Mean.L“ wird als 32 Hex-Wert ausgegeben und dank `Debug.Print_CRLF()` in eine neue Zeile geschrieben (vgl. Beispiel-Code 8). 1E240 entspricht der Dezimalzahl 123456.

### Debug.Print\_SingleHex(str,VarSingle)

Mit der Methode `Debug.Print_SingleHex(str,VarSingle)` wird auf dem HyperTerminal der Wert einer Single<sup>3</sup>-Variablen angezeigt. Gleichzeitig kann der Variablen-Anzeige noch ein erklärender String vorangestellt werden (z.B. „Mittelwert =“, vgl. Beispiel-Code 5).

Angezeigt wird der 32 Bit Hex-Wert des Variablen-Inhalts (vgl. Abbildung 9). Um den Wert der Single-Variablen als Dezimalzahl mit Kommastellen anzuzeigen (vgl. Abbildung 10), muss die Single-Variable erst in einen String umgewandelt werden und dann als String angezeigt werden (vgl. Beispiel-Code 6 und Abbildung 10).

<sup>3</sup> Datentyp „Single“: Wertebereich  $1.175 \times 10^{-38}$  bis  $3.403 \times 10^{38}$ , Genauigkeit 7 bis 8 Dezimalstellen.

**Beispiel-Code 5: Subroutine mit der Methode `Debug.Print_SingleHex(str,VarSingle)`, welche den Hex-Wert von `Mean.S` auf das HyperTerminal ausgibt (vgl. Abbildung 10).**

```

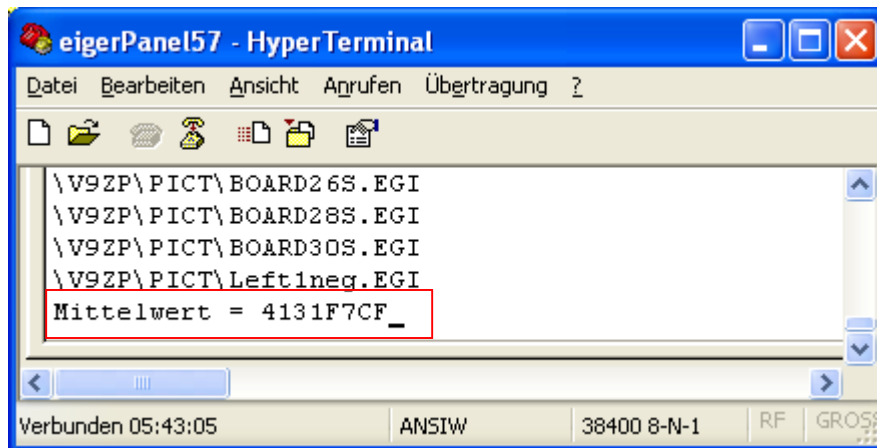
; Buttons zeichnen .....
SUB Left1Button_down           ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Mean.S := 11.123

  Debug.Print_CRLF ( )
  Debug.Print_SingleHex ( 'Mittelwert = ' , Mean.S )

ENDSUB

```



**Abbildung 9: Ausgabe im HyperTerminal aufgrund der Methode `Debug.Print_SingleHex('Mittelwert = ',Mean.S)`. Der Wert der Single-Variablen „Mean.S“ wird als 32 Hexadezimalwert ausgegeben und dank `Debug.Print_CRLF()` in eine neue Zeile geschrieben (vgl. Beispiel-Code 8).**

**Beispiel-Code 6: Subroutine mit Umwandlung der Single-Variablen in eine Stringvariable vor der Debug-Ausgabe des Variablen-Werts auf das HyperTerminal (vgl. Abbildung 10).**

```

; Buttons zeichnen .....
SUB Left1Button_down           ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Mean.S := 11.123

  Str.Cvt_Single ( HilfsString.$ , Mean.S , 4 , 3 )
  Debug.Print_String ( '\\r\\nMittelwert = ' )
  Debug.Print_String ( HilfsString.$ )

ENDSUB

```

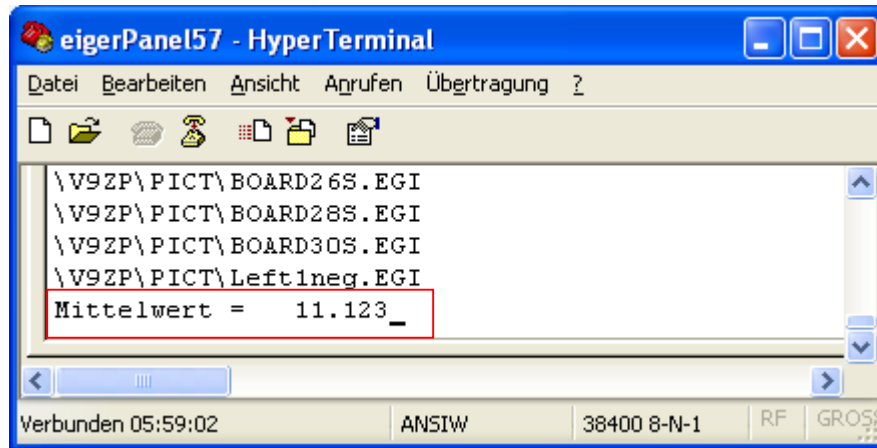


Abbildung 10: Ausgabe im HyperTerminal nach Konvertierung der Single-Variablen in eine String-Variable (vgl. Beispiel-Code 6).

### Debug.Print\_Char (VarInt)

Mit der Methode `Debug.Print_Char(VarInt)` wird auf dem HyperTerminal das ASCII-Zeichen, welches dem Wert der angegebenen Integer-Variable entspricht. Die ASCII-Zeichen und deren Werte sind aus einer ASCII-Tabelle ersichtlich. Der Wert in der Integer-Variable kann ein Hex- oder ein Dezimalwert sein<sup>4</sup>. Aufgrund des Beispiel-Code 7 wird der Inhalt der Integer-Variable auf dem HyperTerminal als Dollar-Zeichen angezeigt (Abbildung 11).

**Beispiel-Code 7:** Subroutine mit der Methode `Debug.Print_Char(VarInt)`, welche entsprechend dem Dezimal- oder Hex-Wert von `Currency.I` ein ASCII-Zeichen auf das HyperTerminal ausgibt. In diesem Fall ist es das Dollar-Zeichen \$ (vgl. Abbildung 11).

```

; Buttons zeichnen.....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Currency.I := 36 ; oder 0x24 (als Hex-Wert)

Debug.Print_CRLF ( )
Debug.Print_Char ( Currency.I )

```

<sup>4</sup> Das Dollar-Zeichen (\$) beispielsweise hat gemäss ASCII-Tabelle den Wert 36 (Dezimal-Wert) bzw. 0x24 (Hex-Wert in eigerScript-Schreibweise). Mehr zum Thema ASCII-Tabelle und ASCII-Zeichen finden Sie im Anhang (S. 15).

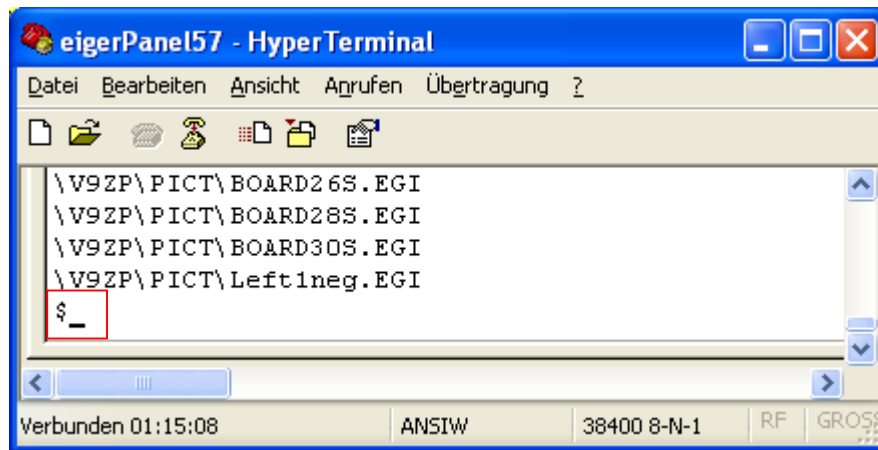


Abbildung 11: Ausgabe im HyperTerminal aufgrund der Methode `Debug.Print_Char` (`Currency.I`). Das Dollar-Zeichen entspricht gemäss ASCII-Tabelle dem Dezimalwert 36 bzw. dem Hex-Wert 0x24 (eigerScript-Schreibweise). Dank `Debug.Print_CRLF()` wird das Zeichen in eine neue Zeile geschrieben (vgl. Beispiel-Code 8).

### Debug.Print\_CRLF()

Mit der Methode `Debug.Print_CRLF()` wird auf dem HyperTerminal ein Zeilenumbruch erwirkt. Die Methode kann beispielsweise einer anderen Debug-Methode vorgelagert werden, um für die Debug-Ausgabe im HyperTerminal-Fenster eine separate Zeile zu erhalten (Beispiel-Code 8).

**Beispiel-Code 8:** Subroutine wie im Beispiel-Code 1 mit der vorgelagerten Methode `Debug.Print_CRLF()`.

```

; Buttons zeichnen .....
SUB Left1Button_down          ; Tastatur 71.33.24 Button 1
  Load.Pos_X1Y1 ( 172 , 136 )
  File.Read_EGI ( 'C:\\V9ZP\\PICT\\Left1neg.EGI' )

  Button.$ := 'Left1Button'

  Debug.Print_CRLF ( )
  Debug.Print_String ( Button.$ )

ENDSUB

```

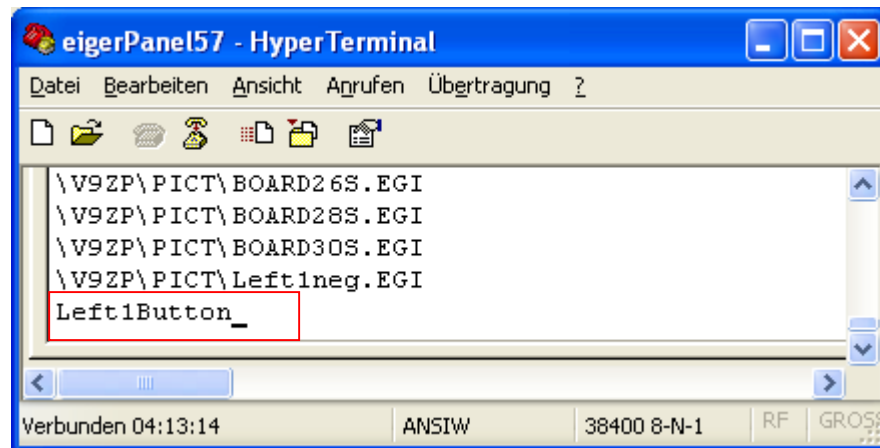


Abbildung 12: Die Methode `Debug.Print_String(Button.$)` bringt den Inhalt der Stringvariablen „Button.\$“ ins Fenster des HyperTerminals. Dank der vorgelagerten Methode `Debug.Print_CRLF()` wird der String in eine Zeile geschrieben (vgl. Beispiel-Code 8).

## 4 Anhang

### 4.1 ASCII-Zeichensatz

Der ASCII-Zeichensatz besteht aus 256 Zeichen. Das ist die maximale Anzahl Zeichen, die mit 8 Bit, d.h. einem Byte unterschieden werden können:  $2^8 = 256$ . Entsprechend ihrer Reihenfolge in der ASCII-Tabelle ist den Zeichen ein ASCII-Wert zwischen 0 bis 255 zugeordnet.

In der IT-Welt ist neben dem Dezimalsystem (10er-Einteilung, d.h. 0-9) auch das Hexadezimalsystem (16er-Einteilung, d.h. 0-F) sehr verbreitet. Je nach Schreibweise kann der ASCII-Wert eines Zeichens unterschiedlich geschrieben werden.

**Tabelle 1:** ASCII-Werte in unterschiedlichen Schreibweisen

Zeichen	Dezimal	Hexadezimal	eigerScript
NUL	000	00	0x00
0	048	30	0x30
Z	090	5A	0x5A
z	122	7A	0x7A
ß	223	DF	0xDF

Der ASCII-Zeichensatz besteht aus den Sonderzeichen (00-1F), dem Standard-Datensatz (00-1F) und dem erweiterten Zeichensatz (80-FF). Während der Standard-Datensatz fix normiert ist, kann die Zusammenstellung des erweiterten Zeichensatzes variieren. Der ASCII-Zeichensatz, der in eigerScript genutzt wird, ist in Tabelle 2 abgebildet.

Die Steuerzeichen in Tabelle 3 sind applikationsspezifisch und werden je nach Empfängergerät unterschiedlich ausgewertet.

**Tabelle 2:** ASCII-Codetabelle ohne Steuerzeichen (0x00 bis 0x1F). Die Zeilen- und Spalten-Nummerierung entspricht dem Hexadezimalsystem (K hat z.B. den ASCII-Wert 75, in hexadezimaler Schreibweise 4C und in der Schreibweise von eigerScript ). ScreenShot des eigerScript-Programms FONT (mitgeliefert als Teil des DemoProgramms TG12, oder als Download unter [www.eigergraphics.com](http://www.eigergraphics.com)).

**Tabelle 3:** Sonder- und Steuerzeichen des Standard-Datensatzes mit ihren ASCII-Werten

Dez	Hex	Abk.	Beschreibung	Dez	Hex	Abk.	Beschreibung
000	00	NUL	Null-Wert	016	10	DEL	Delete
001	01	SOH	Start Of Header	017	11	DC1	Device Control 1 i (1 ≤ i ≤ 4)
002	02	STX	Start Of Text	018	12	DC2	Device Control 2
003	03	ETX	End Of Text	019	13	DC3	Device Control 3
004	04	EOT	End Of Transmission	020	14	DC4	Device Control 4
005	05	ENQ	Enquiry	021	15	NAK	Negative Acknowledge
006	06	ACK	Acknowledge	022	16	SYN	Synchronous Idle (SYNC)
007	07	BEL	Bell, Signalzeichen	023	17	ETB	End Of Transmission Block
008	08	BS	Backspace, Rückschritt	024	18	CAN	Cancel
009	09	HT	Horizontal Tab, Tabulator in Zeilenrichtung	025	19	EM	End Of Medium
010	0A	LF	Line Feed, Zeilenwechsel	026	1A	SUB	Substitute
011	0B	VT	Vertical Tab, Tabulator in Spaltenrichtung	027	1B	ESC	Escape
012	0C	FF	Form Feed, Blatt wird vom Drucker ausgegeben (Seitenvorschub)	028	1C	FS	File Separator
013	0D	CR	Carriage Return, Wagenrücklauf oder Zeilenumbruch	029	1D	GS	Group Separator
014	0E	SO	Shift Out	030	1E	RS	Record Separator
015	0F	SI	Shift In	031	1F	US	Unit Separator

