



Internet-Anbindung mit eNET



http-Server



eigerScript



- Konfiguration der Server Zugangsdaten
- Konfiguration der eNET-Karte
- eNET-Treiberdokumentation





INHALTSVERZEICHNIS

EINLEITUNG	3
KONZEPT	3
HTTP-SERVER	5
eNET-NETZWERKKARTE	6
KOMMUNIKATION MIT DEM SERVER	8
REQUEST UPDATE	8
SEND FILE	10
REQUEST FILE	12
APPLIKATIONS-BEISPIEL	14

Datum	Autor	Beschreibung
01.07.2008	SLU	Erstausgabe
22.06.2009	SLU	Anpassung an eNET 3.0





EINLEITUNG

Das vorliegende Dokument beschreibt die Internetanbindung eines eiger FOX Computers mithilfe einer eNET-Netzwerkkarte. Es beschreibt die Kommunikations-Protokolle zwischen einem http-Server und der eNET-Netzwerkkarte, die Datenserver-Register und -Methoden der eNET-Netzwerkkarte sowie den eigerScript-Treiber für die Ansteuerung der eNET-Netzwerkkarte unter eigerScript.

Die Kommunikation kann mit dem http-Server update.eigergraphics.com getestet werden. Aus dieser WEB-Applikation lassen sich Dateien von und zu einem eiger FOX System austauschen.

Vorraussetzung dafür ist die korrekte Anmeldung der eNET-Netzwerkkarte am System mithilfe einer vierstelligen Kennzahl, die auf jeder eNET-Karte oberhalb der MAC-Adresse angegeben ist.

Dafür muss S-TEC electronics vorab für den Kunden einen Benutzer und ein Projekt erfassen, unter dem die eNET-Netzwerkkarte angemeldet werden kann.

KONZEPT

Das Konzept der Internetanbindung basiert auf der Verwendung des http-put Protokolls zwischen einer eNET-Netzwerkkarte und einem http-Server sowie auf der Verwendung der eigerScript Datenserver Befehle zwischen der eNET-Netzwerkkarte und einem eiger FOX Computer.

Eine Verbindung kann nur von der eNET-Netzwerkkarte aus, zum http-Server aufgebaut werden und besteht aus einem http-put Request. Der Verbindungsaufbau muss aus eigerScript mithilfe der Datenserver-Befehle auf der eNET-Karte gestartet werden. Dazu steht ein Treiber in eigerScript zur Verfügung.

Der http-put Protokollheader wird erweitert mit einem XML-Header, bestehend aus diversen Feldern für Status, Name, MAC-Adresse, Datum- und Uhrzeit, etc.

Optional kann an den XML-Header eine Datei angehängt werden. Dies geschieht im Falle eines Dateiuploads vom eigerFOX zum WEB-Server. Die Datei wird dabei immer HEX-kodiert übermittelt.

Als Antwort erhält das eigerFOX-System vom http-Server einen http-put reply header mit anschliessendem XML-Header, der eine Fehlernummer und -Beschreibung des http-Servers enthält.

Optional kann der Server an den XML-Header der Antwort eine Jobliste oder eine Datei anhängen. Dies im Falle eines Dateidownloads vom Server zum eigerFOX. Eine Datei wird dabei immer HEX-kodiert übermittelt.





industrial electronics

Die Jobliste wird im Klartext übermittelt und besteht aus einer Zeile pro Job. Jede Zeile hat zwei Argumente, die durch ein Semikolon getrennt werden. Das zweite Argument bestimmt den Job-Typ während das erste einen Parameter wie z.B. einen Dateinamen bestimmt.

Es existieren folgende JOB-Typen:

Typ	Job	Beispiels-Jobzeile	Beschreibung
1	REQUEST FILE	C:/MYPR/MYFILE.TXT;1	Der Server fordert damit den eigerScript-Treiber auf, die angegebene Datei vom Server zum FOX runterzuladen. Der eigerScript-Treiber speichert die Datei auf der CompactFlash.
2	SEND FILE	C:/MYPR/MYFILE.TXT;2	Der Server fordert damit den eigerScript-Treiber auf die angegebene Datei von der CompactFlash an den Server hochzuladen.
6	DELETE FILE	C:/MYPR/MYFILE.TXT;6	Der Server fordert damit den eigerScript-Treiber auf angegebene Datei von der CompactFlash zu löschen.
4	RESTART	;4	Führt einen Reset des eigerFOX-Systems aus.

- Eventuelle Dateiodner im Pfad der angegebenen Datei müssen unbedingt auf der CompactFlash bereits existieren!
- Während der Übermittlung von Dateien ist das eigerFOX-System aus der Sicht der Kundenapplikation blockiert.





industrial electronics

HTTP-SERVER

Bevor mit einem http-Server kommuniziert werden kann, müssen auf der eNET-Netzwerkarte die Zugangsdaten konfiguriert werden. Diese können auf der eNET-Karte dauerhaft in ein EEPROM gespeichert werden. Nach einem Reset übernimmt die eNET-Karte dann diese Werte. Die Konfiguration erfolgt unter eigerScript mithilfe der Datenserverbefehle.

Folgende Zugangsdaten müssen konfiguriert werden:

<i>IP-Adresse des Servers</i>	Diese muss als 32bit-Integer übermittelt werden
<i>Port-Nummer des Servers</i>	In aller Regel gleich 80
<i>Hostname des Servers[31]</i>	Die URL, wie z.B. ,update.eigergraphics.com'
<i>Server-Pfad[127]</i>	Der Dateipfad auf dem Server, z.Bsp: ,/interfaces/receive.php'

Das folgende eigerScript-Code Beispiel zeigt wie die Zugangsdaten mithilfe des eNET-Treibers ,EIGER/DRIVER_ENET.EVS' konfiguriert und dauerhaft auf der eNET-Karte gespeichert werden können. Bitte beachten Sie, dass die Länge für Hostname und Server-Pfad die angegebenen Grössen nicht überschreiten.

```
; Includes -----
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'           ; eNET-Treiber

; Diese Routine konfiguriert die Zugangsdaten auf der eNET-Karte für
; den 'update.eigergraphics.com' Server

SUB      INIT_SERVER

; Server IP Adresse setzen
ENET.TMP.$ := '77.59.241.84'                   ; IP-Adr als String-Variable
CallSubroutine( NET.CVT_ADR_TO_LONG )
ENET.SERVERIP.L := ENET.TMP1.L                ; IP-Adr. als Long-Variable

; Port Nummer setzen
ENET.PORT.I := 80                             ; Default http Portnummer

; Hostname setzen
ENET.HOST.$ := 'update.eigergraphics.com'     ; Max. 31 Zeichen!

; Server-Pfad setzen
ENET.PATH.$ := '/interfaces/receive.php'     ; Max. 127 Zeichen!

; Konfiguration speichern
CallSubroutine( ENET.INIT_SERVER )

ENDSUB
```

Code-Beispiel 1: Initialisierung des http-Servers





industrial electronics

eNET-NETZWERKKARTE

Die eNET-Netzwerkkarte übernimmt die ganze Kommunikation mit einem http-Server und muss vorgängig mit den gewünschten Netzwerk-Einstellungen konfiguriert werden. Diese können auf der eNET-Karte dauerhaft in ein EEPROM gespeichert werden. Nach einem Reset übernimmt die eNET-Karte dann diese Werte. Für die Netzwerkeinstellungen müssen folgende Werte konfiguriert werden:

<i>DHCP Modus</i>	1 für DHCP aktiviert, 0 für DHCP deaktiviert.
<i>IP-Adresse</i>	Die gewünschte lokale IP-Adresse (falls DHCP deaktiviert).
<i>Subnet</i>	Die gewünschte lokale Subnet Maske (falls DHCP deaktiviert).
<i>Gateway</i>	IP-Adresse des Gateways (falls DHCP deaktiviert).

Optional können folgende Felder nach eigenem Bedarf benutzt werden. Sie werden bei jedem Verbindungsaufbau dem http-Server in einem XML-Header übermittelt. Es sind alles Textfelder mit einer begrenzten Anzahl Zeichen.

<i>System-ID[31]</i>	Kurzbezeichnung für das System, z.Bsp. ‚KA100-33‘
<i>System-Name[255]</i>	Systembeschreibung, z.Bsp: ‚Kaffeemaschine 10‘
<i>Version[31]</i>	Versionsstand des Systems z.Bsp: ‚V1.03‘

Das nachstehende Feld dient zur Identifikation des eigerFOX-Computers, der die eNET-Karte steuert. Die wird benötigt, falls das System über mehrere eNET-Karten verfügt. Das Portal ‚update.eigergraphics.com‘ unterstützt im Moment nur den Wert ‚eMASTER‘.

<i>Panel[63]</i>	eigerFOX-Computer Identifikation = ‚eMASTER‘
------------------	--

Zusätzlich existieren folgende vier Felder, die auch bei jedem Verbindungsaufbau dem http-Server im XML-Header übermittelt werden aber nicht dauerhaft auf der eNET-Karte gespeichert werden können.

<i>Datum[9]</i>	Aktuelles Systemdatum im Format: JJJJMMTT (20081130)
<i>Zeit[11]</i>	Aktuelle Systemzeit im Format: HH:MM:SS (16:53:17)
<i>Status[255]</i>	Optionale Statusbeschreibung
<i>Error[255]</i>	Optionale Fehlerbeschreibung

Das folgende Code-Beispiel zeigt wie all diese Konfigurationsdaten mithilfe des Treibers ‚EIGER/DRIVER_ENET.EVS‘ dauerhaft auf der eNET-Karte gespeichert werden können.





```
; Includes -----
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'           ; eNET-Treiber

; Diese Routine konfiguriert die eNET-Netzwerkarte -----
SUB      INIT

; Deaktiviere DHCP Modus
ENET.DHCP.I := 0

; IP-Adresse setzen
ENET.TMP.$ := '192.168.0.1'           ; IP-Adr. als String
CallSubroutine( NET.CVT_ADR_TO_LONG )
ENET.LOCALIP.L := ENET.TMP1.L        ; IP-Adr. als Long-Variable

; Subnetmaske setzen
ENET.TMP.$ := '255.255.255.0'        ; Subnet als String
CallSubroutine( NET.CVT_ADR_TO_LONG )
ENET.SUBNET.L := ENET.TMP1.L        ; Subnet als Long-Variable

; Schreibe Gateway
ENET.TMP.$ := '192.200.1.1'          ; Gateway als String
CallSubroutine( NET.CVT_ADR_TO_LONG )
ENET.GATEWAY.L := ENET.TMP1.L       ; Gateway als Long-Variable

; SystemID setzen
ENET.SYSTEMID.$ := 'Meine System-ID'

; System Name setzen
ENET.SYSTEMNAME.$ := 'Mein Systemname'

; Version setzen
ENET.VERSION.$ := 'Mein System-Version'

; Schreibe Panel
ENET.PANEL.$ := 'Mein System-Panel'

; Konfiguration speichern
CallSubroutine( ENET.INIT )

ENDSUB
```

Code-Beispiel 2: Initialisierung der eNET-Karte

Das folgende Code-Beispiel zeigt wie die XML-Felder für Status- und Fehlerbeschreibung gesetzt werden können. Die XML-Felder für Datum- und Uhrzeit müssen nicht selbst gesetzt werden. Die Trieberroutinen für die Kommunikation zum Server setzen diese bei jedem Verbindungsaufbau selbst.

```
; Includes -----
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'           ; eNET-Treiber
```





industrial electronics

```
; Diese Routine setzt die Felder für den Systemzustand -----  
SUB   SET_STATUS_AND_ERROR  
      ; System-Status setzen  
      ENET.STATUS.$ := 'Mein aktueller Systemstatus ist ...'  
  
      ; System-Fehlermeldung setzen  
      ENET.ERROR.$ := 'Mein aktueller Fehlerstatus ist ...'  
  
      ; Werte auf Karte schreiben  
      CallSubroutine( ENET.SET_STATUS_AND_ERROR )  
ENDSUB
```

Code-Beispiel 3: Setzen des Systemzustandes

KOMMUNIKATION MIT DEM SERVER

Das Konzept der eNET-Karte sieht vor, dass sämtliche Verbindungen mit dem Server immer nur von der eNET-Karte aus aufgebaut werden. Dazu dienen drei verschiedene Telegrammtypen:

- | | |
|-------------------|--|
| 1. REQUEST UPDATE | Übermittlung des Systemzustands und Empfang einer Jobliste vom Server. |
| 2. SEND FILE | Upload einer Datei an den Server |
| 3. REQUEST FILE | Download einer Datei vom Server |

REQUEST UPDATE

Das Request-Update Telegramm wird durch die eNET-Karte an den http-Server verschickt falls aus eigerScript die eNET-Treiberoutine *ENET.SEND_UPDATE* aufgerufen wird. Als Antwort erhält die eNET-Karte vom Server eine optionale Jobliste. Das Request-Update Telegramm beginnt mit einem http-Header der folgendermassen aussieht:

␣ = Carriage Return + Line Feed (\r\n).

```
PUT PATH HTTP/1.1␣  
Host: HOSTNAME␣  
Content-Type: text/html␣  
Content-Length: CONTENT_LENGTH␣  
Expect: 100-continue␣  
Connection: Keep-Alive␣
```





industrial electronics

↵

Die Felder **HOSTNAME** und **PATH** entsprechen den eNET-Treibervariablen ENET.HOSTNAME.\$ und ENET.PATH.\$ (siehe Code-Beispiel 1). Das Feld **CONTENT_LENGTH** besteht aus der Länge des nachfolgenden XML-Headers. Der http-Server antwortet mit:

```
HTTP/1.1 1 Continue↵
```

↵

Darauf sendet die eNET-Karte folgenden XML-Header (mit Tabulator \t eingerückt), dessen Länge in Bytes im obigen http-Header vermerkt wird.

```
<?xml version="1.0" encoding="ISO-8859-1"?>↵
<Data>↵
  <Message>↵
    <Date>DATE</Date>↵
    <Time>TIME</Time>↵
  </Message>↵
  <System>↵
    <MAC>MAC-ADRESSE</MAC>↵
    <Id>SYSTEMID</Id>↵
    <Name>SYSTEMNAME</Name>↵
    <Panel>PANEL</Panel>↵
    <ERROR>ERROR</ERROR>↵
    <Status>STATUS</Status>↵
  </System>↵
  <Information>↵
    <Direction>Request</Direction>↵
    <Type>Update</Type>↵
    <Detail>↵
      <Version>VERSION</Version>↵
    </Detail>↵
  </Information>↵
</Data>↵
```

Die folgenden Felder werden durch die eNET-Karte selbstständig ausgefüllt:

DATE	Datum im Format YYYYMMDD.
TIME	Zeit im Format HH:MM:SS
MAC-ADRESSE	Die MAC-Adresse der eNET-Karte im Format m5:m4:m3:m2:m1:m0.





industrial electronics

Die anderen Felder entsprechen den eNET-Treibervariablen ENET.SYSTEMID.\$, ENET.SYSTEMNAME.\$, ENET.HOSTNAME.\$ und ENET.PANEL.\$ (siehe Code-Beispiele 2) sowie ENET.STATUS.\$ und ENET.ERROR.\$ (siehe Code-Beispiele 3). Als Antwort erhält die eNET-Karte einen http-Header mit anschliessendem XML-Header mit Fehlerbeschreibung.

```
<?xml version="1.0" encoding="ISO-8859-1"?>␣  
<HttpResponse>␣  
<Error>891</Error>␣  
<ErrorDescription>No files to Update</ErrorDescription>␣  
<HttpResponse>␣
```

In diesem Falle hat der Server keine pendenten Jobs an das eiger System zu übermitteln und der Vorgang ‚REQUEST UPDATE‘ ist abgeschlossen.

Hat der http -Server aber pendente Jobs zu übermitteln, wie Anfragen zum Dateidownload- oder -upload, so fügt er diese dem XML-Header im Klartext an und gibt eine Fehlernummer 0 zurück:

```
<?xml version="1.0" encoding="ISO-8859-1"?>␣  
<HttpResponse>␣  
<Error>0</Error>␣  
<ErrorDescription>Ok</ErrorDescription>␣  
<HttpResponse>␣  
C: /MYPR/MYFILE_1.TXT;1␣  
C: /MYPR/MYFILE_2.TXT;2␣  
C: /MYPR/MYFILE_3.TXT;6␣
```

In diesem Falle muss der http-Header der Antwort noch das Tag

```
Content-Download-Length: xxx␣
```

enthalten, das die Länge der Job-Liste (ohne XML-Header) enthalten muss. Die eNET-Treiber-routine *ENET.DO_JOBS* geht nun diese Jobliste durch und führt einen Job nach dem anderen aus. Dazu verwendet sie wiederum die Treiber-routinen *ENET.SEND_FILE* und *ENET.REQUEST_FILE*.

SEND FILE

Das Send-File Telegramm wird verwendet, um eine Datei an den Server hochzuladen. Die eNET-Treiber-routine ‚ENET.SEND_FILE‘ übernimmt diesen Vorgang. Vorgängig muss allerdings der Dateinamen (inkl. Pfad) der eNET-Karte übermittelt werden. Das folgende Code-Beispiel verdeutlicht dies.





```
; Includes -----  
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'           ; eNET-Treiber  
  
; Diese Routine übermittelt eine Datei an den Server -----  
SUB      SEND_FILE  
    ; Dateiname setzen  
    DataServer.Tx_String( ENET_NODE, ENET_XML_FILENAME, 'Meine Datei' )  
  
    ; Datei an den Server übermitteln  
    CallSubroutine( ENET.SEND_FILE )  
ENDSUB
```

Code-Beispiel 4: Senden einer Datei an den Server

Das Telegramm beginnt mit einem http-Header der folgendermassen aussieht:

↵ = Carriage Return + Line Feed (\r\n).

```
PUT PATH HTTP/1.1↵  
Host: HOSTNAME↵  
Content-Type: text/html↵  
Content-Length: CONTENT_LENGTH↵  
Expect: 100-continue↵  
Connection: Keep-Alive↵  
↵
```

Die Felder **HOSTNAME** und **PATH** entsprechen den eNET-Treibervariablen ENET.HOSTNAME.\$ und ENET.PATH.\$ (siehe Code-Beispiel 1). Das Feld **CONTENT_LENGTH** besteht aus der Länge des nachfolgenden XML-Headers plus der Länge der zu übermittelnden Datei mal zwei (HEX-Codierung!). Der http-Server antwortet mit:

```
HTTP/1.1 1 Continue↵  
↵
```

Darauf sendet die eNET-Karte folgenden XML-Header. (mit Tabulator \t eingerückt). Im Anschluss an den XML-Header folgt direkt die zu sendende Datei im Hex-Format.

```
<?xml version="1.0" encoding="ISO-8859-1"?>↵  
<Data>↵  
    <Message>↵  
        <Date>DATE</Date>↵
```





```
<Time>TIME</Time>↵
</Message>↵
<System>↵
  <MAC>MAC-ADRESSE</MAC>↵
  <Id>SYSTEMID</Id>↵
  <Name>SYSTEMNAME</Name>↵
  <Panel>PANEL</Panel>↵
  <ERROR>ERROR</ERROR>↵
  <Status>STATUS</Status>↵
</System>↵
<Information>↵
  <Direction>Send</Direction>↵
  <Type>HEXFILE</Type>↵
  <Detail>↵
    <Filename> C:/MYPR/MYFILE.TXT</Filename >↵
  </Detail>↵
</Information>↵
</Data>↵
9635892A4BC045F12E320A32324C43434F232*D121214323123BAC...
```

Die Datenfelder entsprechen denen des REQUEST_UPDATE Telegramms. Der http-Server antwortet nun seinerseits mit einem Standard http-Header gefolgt von einem XML-Header. In diesem wird eine Fehlernummer und -beschreibung übermittelt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>↵
<HttpResponse>↵
<Error>0</Error>↵
<ErrorDescription>Ok</ErrorDescription>↵
<HttpResponse>↵
```

Damit hat der http-Server den Erhalt der Datei bestätigt.

REQUEST FILE

Das Request-File Telegramm wird verwendet, um eine Datei vom Server runterzuladen. Die eNET-Treiberroutine ‚ENET.REQUEST_FILE‘ übernimmt diesen Vorgang. Vorgängig muss allerdings der Dateinamen (inkl. Pfad) der eNET-Karte übermittelt werden. Das folgende Code-Beispiel verdeutlicht dies.





industrial electronics

```
; Includes -----  
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'           ; eNET-Treiber  
  
; Diese Routine lädt eine Datei vom Server -----  
SUB      REQUEST _FILE  
        ; Dateiname setzen  
        DataServer.Tx_String( ENET_NODE, ENET_XML_FILENAME, 'Meine Datei' )  
  
        ; Datei an den Server übermitteln  
        CallSubroutine( ENET.REQUEST_FILE )  
  
ENDSUB
```

Code-Beispiel 5: Download einer Datei vom Server

Das Telegramm beginnt mit einem http-Header der folgendermassen aussieht:

↵ = Carriage Return + Line Feed (\r\n).

```
PUT PATH HTTP/1.1↵  
Host: HOSTNAME↵  
Content-Type: text/html↵  
Content-Length: CONTENT_LENGTH↵  
Content-Download-Length: DOWNLOAD_LENGTH ↵  
Expect: 100-continue↵  
Connection: Keep-Alive↵  
↵
```

Die Felder **HOSTNAME** und **PATH** entsprechen den eNET-Treibervariablen **ENET.HOSTNAME.\$** und **ENET.PATH.\$** (siehe Code-Beispiel 1). Das Feld **CONTENT_LENGTH** besteht aus der Länge des nachfolgenden XML-Headers. Das Feld **DOWNLOAD_LENGTH** enthält die Dateigrösse der angeforderten Datei mal zwei (Hex-Codierung!). Der http-Server antwortet mit:

```
HTTP/1.1 1 Continue↵  
↵
```

Darauf sendet die eNET-Karte folgenden XML-Header (mit Tabulator \t eingerückt), dessen Länge in Bytes im obigen http-Header vermerkt werden muss.

```
<?xml version="1.0" encoding="ISO-8859-1"?>↵  
<Data>↵  
    <Message>↵  
        <Date>DATE</Date>↵
```





```
<Time>TIME</Time>↵
</Message>↵
<System>↵
  <MAC>MAC-ADRESSE</MAC>↵
  <Id>SYSTEMID</Id>↵
  <Name>SYSTEMNAME</Name>↵
  <Panel>PANEL</Panel>↵
  <ERROR>ERROR</ERROR>↵
  <Status>STATUS</Status>↵
</System>↵
<Information>↵
  <Direction>Request</Direction>↵
  <Type>File</Type>↵
  <Detail>↵
    <Filename> C:/MYPR/MYFILE.TXT</Filename >↵
  </Detail>↵
</Information>↵
</Data>↵
```

Die Datenfelder entsprechen denen des REQUEST_UPDATE Telegramms. Der http-Server antwortet nun seinerseits mit einem Standard http-Header gefolgt von einem XML-Header. In diesem wird eine Fehlernummer und -beschreibung übermittelt. Anschliessend an den XML-Header folgt die angeforderte Datei in HEX-Codierung

```
<?xml version="1.0" encoding="ISO-8859-1"?>↵
<HttpResponse>↵
<Error>0</Error>↵
<ErrorDescription>Ok</ErrorDescription>↵
<HttpResponse>↵
9635892A4BC045F12E320A32324C43434F232D121214323123BAC..
```

APPLIKATIONS-BEISPIEL

Das folgende eigerScript Beispiel zeigt, wie mithilfe eines Timers periodisch mit dem Server Verbindung aufgenommen wird. Anschliessend wird geprüft ob die eNET-Karte vom Server eine Jobliste erhalten hat. Diese wird dann ausgeführt. Das Ausführen der Jobliste wird die eigerScript-Applikation temporär blockieren. Das Senden einer Update-Message an den Server erfordert aber keine Rechenzeit.





industrial electronics

```
; Includes -----
INCLUDEFILE 'EIGER/DRIVER_ENET.EVS'

; Timer-Routine -----
SUB PERIODIC_UPDATE

; Wir prüfen, ob Jobs vorhanden sind und führen sie aus
CallSubroutine( ENET.DO_JOBS )

; Wir setzen den Statustext
DataServer.Tx_String( ENET_NODE, ENET_XML_STATUS, 'Hallo' )

; Wir setzen den Fehlertext
DataServer.Tx_String( ENET_NODE, ENET_XML_ERROR, 'Keine Fehler' )

; Wir senden eine neue Update-Meldung an den Server
CallSubroutine( ENET.SEND_UPDATE )

ENDSUB

; Programm-Start -----
BEGINVIEW

; Wir laden die Variablen für Systemkennung und Netzwerkeinstellungen
ENET.SYSTEMID.$ = 'SYS 1' ; SystemID
ENET.SYSTEMNAME.$ = 'My System 1' ; Systemname
ENET.PANEL.$ = 'eMASTER' ; Panel
ENET.VERSION.$ = 'V1.0' ; System-Soft/Hardwarestand

ENET.DHCP.I = 0 ; DHCP deaktiviert
ENET.LOCALIP.L = 0x12345678 ; Lokale IP-Adresse
ENET.SUBNET.L = 0xFFFFFFFF00 ; Lokale Subnetmaske
ENET.GATEWAY.L = 0x12345678 ; IP-Adresse des Gateways

; Wir initialisieren die eNET-Karte
CallSubroutine( ENET.INIT )

; Wir laden die Variablen für die Servereinstellungen
ENET.SERVERIP.L = 0x12345678 ; IP-Adresse
ENET.PORT.I = 80 ; Portnummer
ENET.HOST.$ = 'update.eigergraphics.com' ; Hostname
ENET.PATH.$ = '/interfaces/receive.php' ; Serverpfad

; Wir initialisieren die eNET-Karte mit den Servereinstellungen
CallSubroutine( ENET.INIT_SERVER )

; Wir installieren einen Update-Timer
Timer.InstallLocal( 1, PERIODIC_UPDATE )
Timer.Load( 1, 1000 ) ; alle 1000 msec
Timer.TableEnable()
Timer.StartContinuous( 1 )

LOOP
ENDLOOP

ENDVIEW
```

Code-Beispiel 6: Applikations-Beispiel

