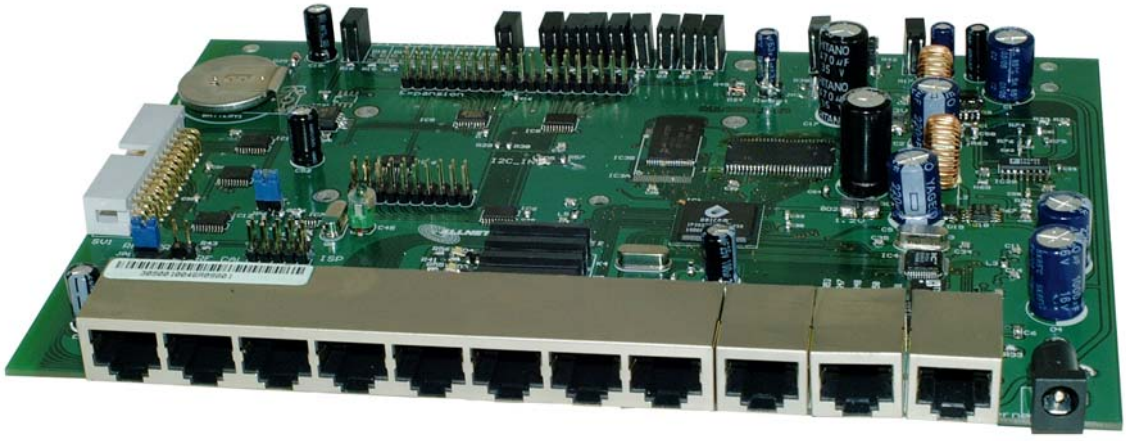
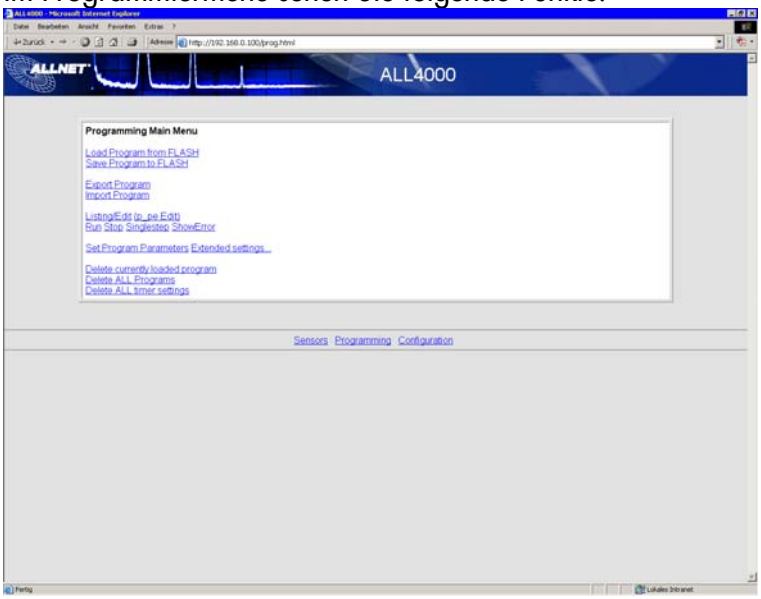


# ALL4000 Programmierhandbuch

V006

# Einstellungen und Grundsätzliches

Im Programmiermenü sehen Sie folgende Punkte:




**Load Program from FLASH/Save Program to FLASH**
Ermöglicht es, Programme aus dem FLASH-Speicher in den PRAM Ausführungsspeicher zu laden, zum Ausführen oder Bearbeiten.
Beim ALL3000RF und ALL3100 kann einer von 64 Speicherplätzen gewählt werden (0...63) - beim ALL4000 nur der Speicherplatz 0.

**Listing/Edit**
Hier kann das aktuelle Programm betrachtet und verändert werden. Es ist möglich, während das Programm ausgeführt wird, Änderungen daran vorzunehmen und diese somit sofort zu beobachten.

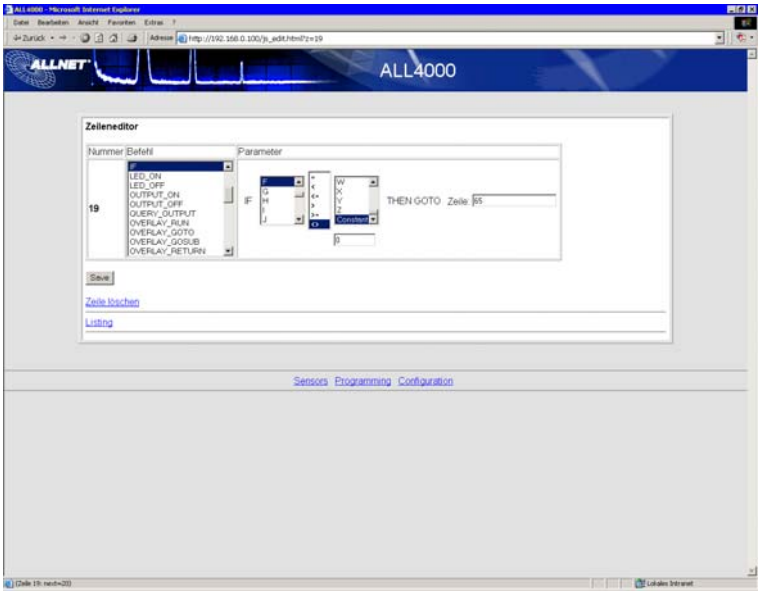
**Set Program Parameters/Extended Settings**
Wartezeit zwischen Programmschritten: Hier kann die Verarbeitungsgeschwindigkeit beeinflußt werden. Der Standardwert von 20 ms ergibt 50 Befehle pro Sekunde. Gültige Werte liegen zwischen 5 ms und 5000 ms.
Programm #0 beim Booten automatisch starten: Wenn hier eine 1 eingetragen ist, lädt der ALL4000 sofort beim Einschalten das Programm vom FLASH-Speicherplatz 0 und beginnt mit dessen Ausführung. Somit können Steuer- und Regelprogramme automatisch in Gang gesetzt werden.

**Delete Current Program**
Löscht alle Zeilen des gerade geladenen Programms im PRAM. Somit können Sie beginnen, ein neues Programm zu erstellen, ohne mühselig alle Zeilen einzeln löschen zu müssen.

**Delete all Programs**
VORSICHT: Diese Funktion löscht ohne weitere Nachfrage alle im FLASH gespeicherten Programme, sowie das im PRAM zur Ausführung gespeicherte Programm.

## Der Editor

Grundsätzlich wird beim Aufruf des Editors ein komplettes Text-Listing des gerade im PRAM geladenen Programmes angezeigt. Klicken auf die Zeilennummer öffnet diese Zeile zum Bearbeiten oder Löschen.

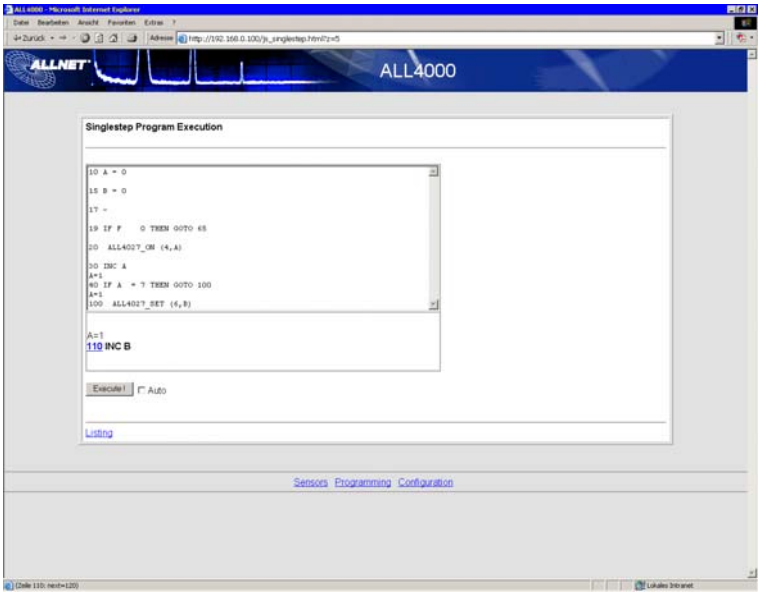



Links wird die Zeilennummer angezeigt. Aus der direkt daneben befindlichen Auswahlbox „Befehl“ können Sie nun den gewünschten Befehl anklicken. Daraufhin werden im Feld „Parameter“ mögliche Optionen angezeigt.
Wenn die gewünschten Änderungen ausgeführt sind, drücken Sie bitte den Knopf „Save“, damit diese zum ALL4000 übertragen werden.

Alle Programmänderungen beziehen sich immer auf das im RAM gespeicherte Programm, nicht auf die Kopie im FLASH. Denken Sie daher bitte daran, öfters mal Ihr Programm mit der Save-To-FLASH-Funktion abzuspeichern.

## Der Debugger

Manchmal funktioniert ein selbst geschriebenes Programm nicht auf Anhieb so, wie man es sich vorgestellt hat. Hier ist dann der Debugger nützlich. Er ermöglicht es, die Programmzeilen Schritt für Schritt anzuarbeiten, und zeigt gleichzeitig die Inhalte aller Variablen vor und nach der Verarbeitung an. Auch eine Liste der letzten 10 Befehle ist vorhanden, so daß man sofort erkennen kann, was genau in dem Programm vorgeht.




Es ist auch möglich, dem Programm einfach bei der Ausführung zuzusehen, indem die Checkbox „Auto“ aktiviert wird. Dann läuft das Programm mit 1 Schritt pro Sekunde ab, und man kann in Ruhe beobachten, was passiert.

# Befehlsübersicht

**0 – NOP (-)**
Leere Anweisung
Dieser Befehl führt keinerlei Aktion aus. Wird standardmäßig vom ALL4000 vergeben, wenn eine neue Programmzeile angelegt wird. Kann zur besseren Lesbarkeit (Strukturierung des Quellcodes) eingesetzt werden.

**1 – GOTO (Zeilennummer)**
Sprunganweisung
Ein unbedingter Sprung zu der angegebenen Programmzeile wird ausgeführt. Falls die Programmzeile nicht vorhanden ist, wird ein Fehler (Fehler # 100: Zeile nicht gefunden) ausgelöst. Es kann entweder eine Konstante als Zeilennummer angegeben werden, oder auch eine Variable.

**2 – INC (Variable)**
Variable um 1 erhöhen
Der Inhalt der angegebenen Variablen wird um 1 erhöht (inkrementiert). Falls hierbei der maximal darstellbare Wertebereich einer Variablen (16 bit) überschritten wird (also der Wert von 32767 überschritten würde), wird mit dem kleinsten darstellbaren Wert (-32768) weitergearbeitet.
Die Anweisung „INC A" bewirkt dasselbe wie die Anweisung „A = A + 1"

**3 – DEC (Variable)**
Variable um 1 verringern
Der Inhalt der angegebenen Variablen wird um 1 heruntergezählt (dekrementiert). Falls hierbei der maximal darstellbare Wertebereich einer Variablen (16 bit) überschritten wird (also der Wert von -32768 unterschritten würde), wird mit dem größten darstellbaren Wert (32767) weitergearbeitet.
Die Anweisung „DEC A" bewirkt dasselbe wie die Anweisung „A = A - 1"

**4 – SET LED (Variable oder Konstante)**
8-bit Bitmuster mit den LED´s auf der Frontseite des Geräts anzeigen
Die untersten 8 bits der angegebenen Variablen bzw. Konstanten werden zur Ansteuerung der benutzerdefinierbaren LED´s benutzt.

| LED | Bit | dezimaler Wert |
|---|---|---|
| LED0 | bit0 | 1 |
| LED1 | bit1 | 2 |
| LED2 | bit2 | 4 |
| LED3 | bit3 | 8 |
| LED4 | bit4 | 16 |
| LED5 | bit5 | 32 |
| LED6 | bit6 | 64 |
| LED7 | bit7 | 128 |

Der Wert der OR-Verknüpfung der Dezimalwerte der einzuschaltenden LED´s kann als Parameter übergeben werden.
Sollen beispielsweise die LED´s 0 und 7 aufleuchten, so lautet der Befehl „SET LED 129".

Hinweis: da der ALL4000 die LEDs auch zur Statusanzeige der Sensoren benutzt, werden hier vorgenommene Anzeigen sofort wieder durch den Sensorstatus überschrieben. Um dies zu verhindern, können Sie im Menü „Programmierung->Extended Settings“ diese Anzeige über den Punkt „Aktivitätsanzeige über LEDs“=0 deaktivieren.

**5 – GOSUB (Zeilennummer)**
Subroutine an angegebener Zeilennummer ausführen
Ähnlich wie bei GOTO, wird hier zu der angegebenen Zeilennummer gesprungen. Sobald der Interpreter dann eine RETURN-Anweisung findet, springt er zurück zu der Stelle, an der das zugehörige GOSUB stand, und fährt hier mit der nächsten Anweisung fort.
Mit Hilfe der GOSUB...RETURN Befehle ist es möglich, häufiger benötigte Programmteile von verschiedenen Stellen im Programm aus aufzurufen. (Unterprogramm)
Das ineinander-verschachteln von GOSUB´s (Unterprogramme rufen Unterprogramme auf) ist bis zu 10 Ebenen tief möglich.
Es kann entweder eine Konstante als Zeilennummer angegeben werden, oder auch eine Variable.

Beispiel:

```
10 A = 0
20 B = QUERY_OUTPUT(0)
30 IF B = 0 THEN GOTO 20
40 GOSUB 1000
50 GOTO 20
990 -
1000 INC A
1010 SET_LED: A
1020 RETURN
```

Dieses Programm hat ein Unterprogramm ab Zeile 1000, das die Variable „A“ bei jedem Aufruf um 1 erhöht, und dann diesen Zahlenwert als Bitmuster auf den eingebauten LED´s anzeigt.
Die Hauptschleife in Zeile 20 und 30 prüft, ob die Steckdose Nummer 0 eingeschaltet ist. Ist sie es nicht, wird in der Schleife gewartet. Ist sie eingeschaltet, so wird das Unterprogramm bei jedem Durchlauf des Programms aufgerufen.

**6 – RETURN**
Rücksprung zum aufrufenden GOSUB
Es wird zu derjenigen GOSUB-Anweisung zurückgekehrt, die dieses Unterprogramm aufgerufen hatte.

**7 – CLR**
Löschen aller Variableninhalte
Mit diesem Befehl werden die Inhalte aller Variablen „A“ ... „Z“ auf den Wert 0 gesetzt. Diese Funktion wird NICHT automatisch beim Programmstart ausgeführt; Variableninhalte bleiben bis zum Ausschalten des ALL4000 erhalten.

**8 – COMPUTE (Variable) = (Variable oder K.) [ (Verknüpfung) (2. Variable oder K.) ]**
Variablen Werte zuweisen oder Berechnungen ausführen
Diese Anweisung hat 2 Möglichkeiten:
1. Einfache Wertzuweisung
Beispiele:

A = 50 (Der Variablen A wird der Wert 50 zugewiesen)
X = Y (Die Variable X soll den Inhalt von der Variablen Y bekommen)

2. Berechnung
Beispiele:

A = 1 + 1 (Der Variablen A wird der Wert 2 zugewiesen (1 + 1 ergibt 2))
B = A + 100 (Die Variable B soll den Inhalt von A zuzuglich 100 erhalten)
E = F * G (E wird das Produkt von F und G zugewiesen)
A = A / 2 (A wird halbiert)
B = B OR 1 (Bit 0 in B wird gesetzt)
B = B AND 255 (alle bits bis auf die niedrigsten 8 in B werden gelöscht)
C = D XOR E (C erhält das Ergebnis der X-OR Verknüpfung von D und E

**9 - IF (Variable oder K.) (Vergleichsoperator) (Variable oder K.) THEN GOTO (Zeilennummer)**
Sprung zur Zeilennummer, wenn das Ergebnis des Vergleichs wahr ist
Mit dieser Anweisung können Programmverzweigungen in Abhängigkeit von Bedingungen realisiert werden.
Es kann entweder eine Konstante als Zeilennummer angegeben werden, oder auch eine Variable.
Beispiele:

IF X > 100 THEN GOTO 1000 (Wenn X größer als 100 ist, zu Zeile 1000 springen)
IF A = B THEN GOTO 2000 (Wenn A und B gleich sind, zu Zeile 2000 springen)

**10 - LED_ON (Variable oder Konstante)**
gezieltes Einschalten einer LED
Die Leuchtdiode mit der angegebenen Nummer wird eingeschaltet. Alle anderen LED´s bleiben von dieser Operation unbeeinflußt.
Beispiele:

LED_ON 0 (Led #0 einschalten)
LED_ON A (Led mit Nummer in Variable A einschalten)

Gültige Werte sind 0...15 - wird ein anderer Wert angegeben, so wird der Befehl ignoriert.

**11 - LED_OFF (Variable oder Konstante)**
gezieltes Ausschalten einer LED
Die Leuchtdiode mit der angegebenen Nummer wird ausgeschaltet. Alle anderen LED´s bleiben von dieser Operation unbeeinflußt.
Beispiele:

LED_OFF 7 (Led #7 abschalten)
LED_OFF X (Led mit Nummer in Variable X ausschalten)

Gültige Werte sind 0...15 - wird ein anderer Wert angegeben, so wird der Befehl ignoriert.

**12 - OUTPUT_ON (Variable oder Konstante)**
gezieltes Einschalten eines Funkempfängers
Der Funkempfänge mit der angegebenen Nummer wird eingeschaltet. Alle anderen Empfänger bleiben von dieser Operation unbeeinflußt. Wenn der Empfänger vorher ausgeschaltet war, wird die Sendung des „EIN"-Signals in der nächsten Sendeperiode bevorzugt vorgenommen.
Beispiel:

OUTPUT_ON 15 (Empfänger #15 einschalten)

Gültige Werte sind 0...15 - wird ein anderer Wert angegeben, so wird der Befehl ignoriert.

**13 - OUTPUT_OFF (Variable oder Konstante)**
gezieltes Ausschalten eines Funkempfängers
Der Funkempfänge mit der angegebenen Nummer wird ausgeschaltet. Alle anderen Empfänger bleiben von dieser Operation unbeeinflußt. Wenn der Empfänger vorher eingeschaltet war, wird die Sendung des „AUS"-Signals in der nächsten Sendeperiode bevorzugt vorgenommen.
Beispiel:

OUTPUT_OFF 7 (Empfänger #7 abschalten)

Gültige Werte sind 0...15 - wird ein anderer Wert angegeben, so wird der Befehl ignoriert.

**14 - (Variable) = QUERY_OUTPUT (Variable oder Konstante)**
Abfragen des Schaltzustandes eines Funkempfängers
Mit diesem Befehl kann festgestellt werden, ob ein bestimmter Empfänger ein- oder ausgeschaltet ist. Dies ist KEIN Rückkanal, sondern lediglich eine Abfrage des internen Schaltzustands in der Sende-Queue des ALL4000.
Beispiele:

A = QUERY_OUTPUT 5 (A ist 0, wenn Empfänger 7 abgeschaltet ist, sonst 1)
X = QUERY_OUTPUT Y (X erhält den Status (0/1) des Empfängers in Y)

Programmbeispiel:

```
10 B = QUERY_OUTPUT(0)
20 SET_LED: B
30 GOTO 10
```

Dieser 3-Zeiler zeigt den Status der Funksteckdose #0 mit der LED #0 am Gerät an.

**15 - OVERLAY_RUN (Programmnummer oder Variable)**
obsolet, im ALL4000 nicht mehr implementiert.

**16 - OVERLAY_GOTO (Programmnummer oder Variable) (Zeilennummer oder Variable)**
obsolet, im ALL4000 nicht mehr implementiert.

**17 - OVERLAY_GOSUB (Programmnummer oder Variable) (Zeilennummer oder Variable)**
obsolet, im ALL4000 nicht mehr implementiert.

**18 - OVERLAY_RETURN**
obsolet, im ALL4000 nicht mehr implementiert.

**19 - (Variable) = QUERY_RTC_SECOND**
Abfrage des Sekundenzählers der eingebauten Echtzeituhr
Mit Hilfe dieses Befehls ist der Zugriff auf die „Sekunden"-Anzeige der RTC möglich.

**20 - (Variable) = QUERY_RTC_MINUTE**
**21 - (Variable) = QUERY_RTC_HOUR**
**22 - (Variable) = QUERY_RTC_DAY**
**23 - (Variable) = QUERY_RTC_MONTH**
**26 - (Variable) = QUERY_RTC_WEEKDAY**
Wie 18., nur für die restlichen Uhrenfelder.

**24. (Variable) = QUERY_SENSOR (Variable oder Konstante)**
Abfrage eines Sensorwertes
Die direkt an einem der ALL4000 Ports angeschlossenen Sensoren können mit dem Befehl (Variable) = QUERY_SENSOR(0...7) abgefragt werden. Der Rückwert ist in Hundertsteln (1/100) angegeben. Eine Temperatur von 23.5 C gibt also den Wert 2350 zurück.

**25 - DELAY (Variable oder Konstante)**
Verzögerung der Programmausführung
Üblicherweise läuft das interpretierte Programm mit einer festen Geschwindigkeit von 50 Befehlen pro Sekunde ab. Wenn gewünscht ist, an bestimmten Stellen etwas abzuwarten, kann mit diesem Befehl eine „Pause" in der Länge von der angegebenen Anzahl Programmschritte eingelegt werden.
Beispiel:

```
DELAY 50 (1 Sekunde warten)
```

**27 – SET LED2 (Variable oder Konstante)**
8-bit Bitmuster mit den LED´s auf der Frontseite des Geräts anzeigen
Selbe Funktion wie SET_LED, jedoch für die rechten 8 LEDs.

**28 – PING_PREPARE1 (Variable oder Konstante, Variable oder Konstante)**
Die ersten 2 Ziffern einer über PING anzusprechenden IP-Adresse werden mit diesem Befehl spezifizert.

**29 – PING_PREPARE2 (Variable oder Konstante, Variable oder Konstante)**
Die zweiten 2 Ziffern einer über PING anzusprechenden IP-Adresse werden mit diesem Befehl spezifizert.

**30 – n = PING (Variable oder Konstante)**
anPINGen einer IP-Adresse. Der Parameter übergibt die Timeout-Zeit in Programmzyklen, bevor der Ping-Request als „tot" angesehen wird.
der Rückwert n enthält die Anzahl der tatsächlich vergangenen Programmzyklen bis die Antwort von der Ziel-IP eingetroffen ist. Wenn ein Timeout auftrat (= keine Antwort kam), dann ist dieser Rückwert -1.

Beispiel: Anpingen von 192.168.20.1, Timeout 1 Sekunde:
```
10 PING_PREPARE1 (192,168
20 PING_PREPARE2 20,1)
30 A = PING(50)
```

**31 – NOBREAK (noch nicht implementiert)**
Ab dieser Anweisung werden alle Befehle vom Interpreter ohne Wartezyklen ausgeführt.

**32 – LCD_OUT (noch nicht implementiert)**
Ausgabe von Zeichen an ein angeschlossenes Character-LCD-Display.

**33 – DOEVENTS (noch nicht implementiert)**
Sofort nach dieser Anweisung wird ein Wartezyklus für den internen I2C-Bus ausgeführt, so daß die I/O-Einstellungen wirksam werden.

**34 – RELAIS_ON (Variable oder Konstante)**
Das angegebene interne Relais (0...3) wird eingeschaltet.

**35 – RELAIS_OFF (Variable oder Konstante)**
Das angegebene interne Relais (0...3) wird ausgeschaltet.

**36 – n = QUERY_RELAIS_ON (Variable oder Konstante)**
Der Status des jeweiligen Relais wird zurückgegeben (0 oder 1).

**37 – TTL_ON (Variable oder Konstante)**
Der angegebene interne TTL_Pin (0...63) wird auf logisch „1“ gesetzt.
0...7: Ports 0-7 auf Chip 0 (Front-LEDs linke Seite)
8...15: Ports 0-7 auf Chip 1 (Front-LEDs rechte Seite)
16...23: Ports 0-7 auf Chip 2 (nicht vorhanden)
24...31: Ports 0-7 auf Chip 3
32...39: Ports 0-7 auf Chip 4
40...47: Ports 0-7 auf Chip 5
48...55: Ports 0-7 auf Chip 6
56...63: Ports 0-7 auf Chip 7

**38 – TTL_OFF (Variable oder Konstante)**
Der angegebene interne TTL_Pin (0...63) wird auf logisch „0“ gesetzt.
Zuordnung der Nummern zu den I/O Chips siehe TTL_ON.

**39 – n = QUERY_TTL (Variable oder Konstante)**
Der angegebene interne TTL_Pin (0...33) wird abgefragt.
Zuordnung der Nummern zu den I/O Chips siehe TTL_ON.
Rückwert ist 0 oder 1.

**40 – TTL_SET (Variable oder Konstante, Variable oder Konstante)**
8-Bittiges Setzen der I/O-Pins des angegebenen Chips.
1. Parameter: Chip-Nummer (0...7)
2. Parameter: Ausgangs-Bitmap (0...255)

**41 – n = QUERY_TTL8 (Variable oder Konstante)**
Die I/O-Leitungen des angegebenen Chips (0...7) werden abgefragt.
Rückwert ist 0.255.

**42 – ALL4027_ON (Variable oder Konstante, Variable oder Konstante)**
1. Parameter: Portnummer 0...7
2. Parameter: Pinnummer 0...7
Der angegebene externe TTL_Pin (0...7) wird auf logisch „0“ gesetzt.

**43 – ALL4027_OFF (Variable oder Konstante, Variable oder Konstante)**
1. Parameter: Portnummer 0...7
2. Parameter: Pinnummer 0...7
Der angegebene externe TTL_Pin (0...7) wird auf logisch „1“ gesetzt.

**44 – ALL4027_SET (Variable oder Konstante, Variable oder Konstante)**
1. Parameter: Portnummer 0...7
2. Parameter: Bitmaske 0...255
Die 8 angegebenen externen TTL_Pins werden der Bitmaske entsprechend gesetzt.

## Fernsteuerung über das Web-Interface

Es ist möglich, alle Schaltvorgänge im ALL4000 über simple HTTP-Requests auszulösen.

Die URL der generischen Schaltbefehle lautet:
http://192.168.0.100/uo?d=<porttyp>&n1=<portnummer>&n2=<wert>
wobei allen 3 Parametern als Defaultwert 0 zugrundegelegt wird.

Die Porttypen des ALL4000 sind wie folgt definiert:
0 - Default (2 - Funksteckdosen)
1 - Interne Relais (nach Relaisnummer 0..3) 0 oder 1
2 - Funksteckdosen (nach Dosennummer 0...15) 0 oder 1
3 - I2C Einzel-Ausgänge (Portweise numeriert von 0...7) 0 oder 1
4 - I2C 8-fach-Ausgänge (Portweise numeriert von 0...7) 0...255
5 - interne TTL-Ausgänge (Chipweise numeriert von 0..7) 0...255

Soll beispielsweise Relais 5 am ALL4027 an Port 3 eingeschaltet werden, so fordern Sie diese URL an:
http://192.168.20.1/uo?d=4&n1=3&n2=223
d=4: Porttyp „I2C 8-fach Ausgang“
n1=3: AL4027 an Port 3 ist gemeint
n2=32: 2^5 ist 32, also das 5. Bit gesetzt. 255-32 ist 223 (Relais sind Active-Low)

Möchten Sie Funksteckdose #10 ausschalten, so fordern Sie diese URL an:
http://192.168.20.1/uo?d=2&n1=10&n2=0
d=2: Porttyp „Funksteckdose“
n1=10: Dose 10 ist gemeint
n2=0: Ausschalten

Als Rückgabe-Webseite erhalten Sie immer eine einfache Textseite vom ALL4000, in der Ihnen angezeigt wird, welche Parameter wie interpretiert worden sind. Wollen Sie nachprüfen, ob der Schaltvorgang auch tatsächlich erfolgt ist, so können Sie dies jederzeit mit der XML-Seite tun.

# Die Abfrage der Sensoren über HTTP (XML und YML)

Zwei Formen der Formatierung stehen zur Verfügung:

### 1. XML (Standard)

Die Anwendung kann vom ALL4000 Webserver die folgende Page anfordern:
http://192.168.0.100/xml

Zurückgeliefert wird in etwa folgendes:

```
<xml><data>
<devicename>ALL4000</devicename>
<n0>0</n0><t0> 62.31</t0><min0> 45.63</min0><max0> 100.02</max0><l0>-55</l0><h0>150</h0><s0>65</s0>
<n1>1</n1><t1> 17.33</t1><min1> 10.54</min1><max1> 100.02</max1><l1>-55</l1><h1>150</h1><s1>3</s1>
<n2>2</n2><t2>-20480.00</t2><min2>-0.10</min2><max2> 30.04</max2><l2>-55</l2><h2>150</h2><s2>0</s2>
<n3>3</n3><t3>-20480.00</t3><min3> 20480.00</min3><max3>-20480.00</max3><l3>-55</l3><h3>150</h3><s3>0</s3>
<n4>4</n4><t4>-20480.00</t4><min4> 0.00</min4><max4> 100.02</max4><l4>-55</l4><h4>150</h4><s4>0</s4>
<n5>5</n5><t5>-20480.00</t5><min5> 8.20</min5><max5> 24.50</max5><l5>-55</l5><h5>150</h5><s5>0</s5>
<n6>6</n6><t6> 1.65</t6><min6>-39.64</min6><max6> 42.38</max6><l6>-55</l6><h6>150</h6><s6>133</s6>
<n7>7</n7><t7>-20480.00</t7><min7>-0.40</min7><max7> 124.23</max7><l7>-55</l7><h7>150</h7><s7>0</s7>
<fn0>1</fn0><ft0>0</ft0><fs0>0</fs0>
<fn1>2</fn1><ft1>0</ft1><fs1>0</fs1>
<fn2>3</fn2><ft2>0</ft2><fs2>0</fs2>
<fn3>4</fn3><ft3>0</ft3><fs3>0</fs3>
<fn4>5</fn4><ft4>0</ft4><fs4>0</fs4>
<fn5>6</fn5><ft5>0</ft5><fs5>0</fs5>
<fn6>7</fn6><ft6>0</ft6><fs6>0</fs6>
<fn7>8</fn7><ft7>0</ft7><fs7>0</fs7>
<fn8>9</fn8><ft8>0</ft8><fs8>0</fs8>
<fn9>10</fn9><ft9>0</ft9><fs9>0</fs9>
<fn10>11</fn10><ft10>0</ft10><fs10>0</fs10>
<fn11>12</fn11><ft11>0</ft11><fs11>0</fs11>
<fn12>13</fn12><ft12>0</ft12><fs12>0</fs12>
<fn13>14</fn13><ft13>0</ft13><fs13>0</fs13>
<fn14>15</fn14><ft14>0</ft14><fs14>0</fs14>
<fn15>16</fn15><ft15>0</ft15><fs15>0</fs15>
<rn0>0</rn0><rt0>0</rt0>
<rn1>1</rn1><rt1>0</rt1>
<rn2>2</rn2><rt2>0</rt2>
<rn3>3</rn3><rt3>0</rt3>
<it0>188</it0><it1>191</it1><it2>255</it2><it3>255</it3><it4>255</it4><it5>255</it5><it6>255</it6><it7>255</it7>
<date>08.04.2006</date><time>08:09:47</time><ad>1</ad><i>10</i><f>0</f>
<sys>12</sys><mem>33128</mem><fw>2.51</fw><dev>ALL4000</dev>
</data></xml>
```

**Beschreibung der einzelnen Felder:**

Devicename: Bezeichnung des Geräts
n<#>: Bezeichnung des Sensors
t<#>: Messwert des Sensors (-2048.00 wenn kein Sensor erkannt wurde)
min<#>: Kleinster gemessener Wert an diesem Sensor
max<#>: Höchster gemessener Wert an diesem Sensor
l<#>: Unterer Grenzwert für diesen Sensor
h<#>: Oberer Grenzwert für diesem Sensor
s<#>: Sensortyp (0: Nicht erkannt)
fn<#>: Bezeichnung der Funksteckdose
ft<#>: Sollzustand der Funksteckdose (0 oder 1)
fs<#>: Dosen-Anzeigebild der Funksteckdose
rn<#>: Bezeichnung des Relais
rt<#>: Zustand des Relais (0 oder 1)
it<#>: Zustand der TTL-EINgänge des jeweiligen Chips
<date>: Datum von RTC
<time>: Zeit von RTC
<ad>: Zoom 0/1 (obsolet) - immer 1 beim ALL4000
<i>: Zeitintervall für einen horizontalen Pixel in der Webanzeige in Sekunden
<f>: Temperatur-Skala für Webanzeige (0: Grad C, 1 - Grad F, 2 - Kelvin)
<sys>: System Counter, wird jede Sekunde erhöht. Als Uptime nutzbar
<mem>: Freier Speicher auf dem Heap des ALL4000
<fw>: Firmware-Version
<dev>: Gerätetyp, liefert immer „ALL4000“ zurück

**2. YML (obsolet, wird aber vom Javascript auf der Anzeigeseite verwendet)**
proprietäres kompaktes Datenformat. Die URL lautet:
http://192.168.0.100/s
Zurückgeliefert wird folgender Datenstring:

```
A5684B1837C-2048000D-2048000E-2048000F-2048000G165H-2048000IU2.51V33132W08:22:48X0Y201Z!YMLOK@ALL4000
```

**Felder:**
Wert ab Delimiter „A“: Empfänger 0
...
Wert ab Delimiter „P“: Empfänger 1
Wert ab Delimiter „U“: Firmware-Version
Wert ab Delimiter „V“: Freier Speicher
Wert ab Delimiter „W“: Uhrzeut von RTC
Wert ab Delimiter „X“: Programmiercounter
Wert ab Delimiter „Y“: (unbenutzt)
Wert ab Delimiter „!“: „YMLOK“ (muss bei gültigem Wert immer vorkommen)
Wert ab Delimiter „@“ bis Ende: Gerätename

## Programmbeispiele

Hier noch eine Sammlung kleiner Beispielprogramme, um die Möglichkeiten der Programmierung zu veranschaulichen.

### Temperaturregelung Brutkasten

Das Programm kann die Heizung über Funksteckdose 0 ein- und ausschalten. Ein Sensor, der direkt am ALL4000 angeschlossen ist, kontrolliert die Temperatur. Bei Unterschreiten einer Temperaturgrenze von 35,5 Grad C wird die Heizung eingeschaltet.

```
10 T = QUERY_SENSOR(0)
15 IF T < -32000 THEN GOTO 100
20 IF T < 3550 THEN GOTO 120
30 -
100 OUTPUT_OFF: 0
110 GOTO 10
120 OUTPUT_ON: 0
130 GOTO 10
```

Erklärung:

Zeile 10: Sensor0 wird abgefragt. Der Meßwert wird der Variablen T zugewiesen

Zeile 15: Falls kein Sensor angeschlossen ist, wird ein wert von -32767 zurückgegeben. Um in diesem Fall zu verhindern, daß die Heizung ständig eingeschaltet wird und der Brutkasten zu heiß wird, wird bei solchen ungültigen Meßwerten die Heizung abgestellt. (Zeile 100)

Zeile 20: Wenn die Grenztemperatur von 35.5 Grad C (3550 Zentigrad) unterschritten ist, soll die Heizung eingeschaltet werden. (Teile 120

Zeile 30: Leere Zeile, der besseren Lesbarkeit halber

Zeile 100: Funksteckdose #0 wird abgeschaltet

Zeile 110: Sprung zum Anfang, so daß erneut gemessen und geprüft werden kann

Zeile 120: Funksteckdose #0 wird eingeschaltet

Zeile 130: Sprung zum Anfang

### Knight Rider

Hier wandert einfach ein Lichtpunkt von links nach rechts und wieder zurück.

```
10 A = 1
20 SET_LED: A
30 A = A * 2
40 IF A <= 128 THEN GOTO 20
45 -
50 A = A / 2
60 SET_LED: A
70 IF A > 1 THEN GOTO 50
80 GOTO 30
```

Erklärung:

Zeile 10: Unsere Laufvariable A wird auf den Anfangswert 1 (= LED 0) gesetzt

Zeile 20: Der Wert A wird binär mit den Geräte-LED´s angezeigt

Zeile 30: A wird verdoppelt: 2, 4, 8, 16... usw.

Zeile 40: Wenn A kleiner oder gleich 128 ist, Springe zu Zeile 20 und wiederhole das Spiel

Zeile 50: A wird halbiert: 128, 64, 32, 16... usw.

Zeile 60: Der Wert wird wieder visualisiert

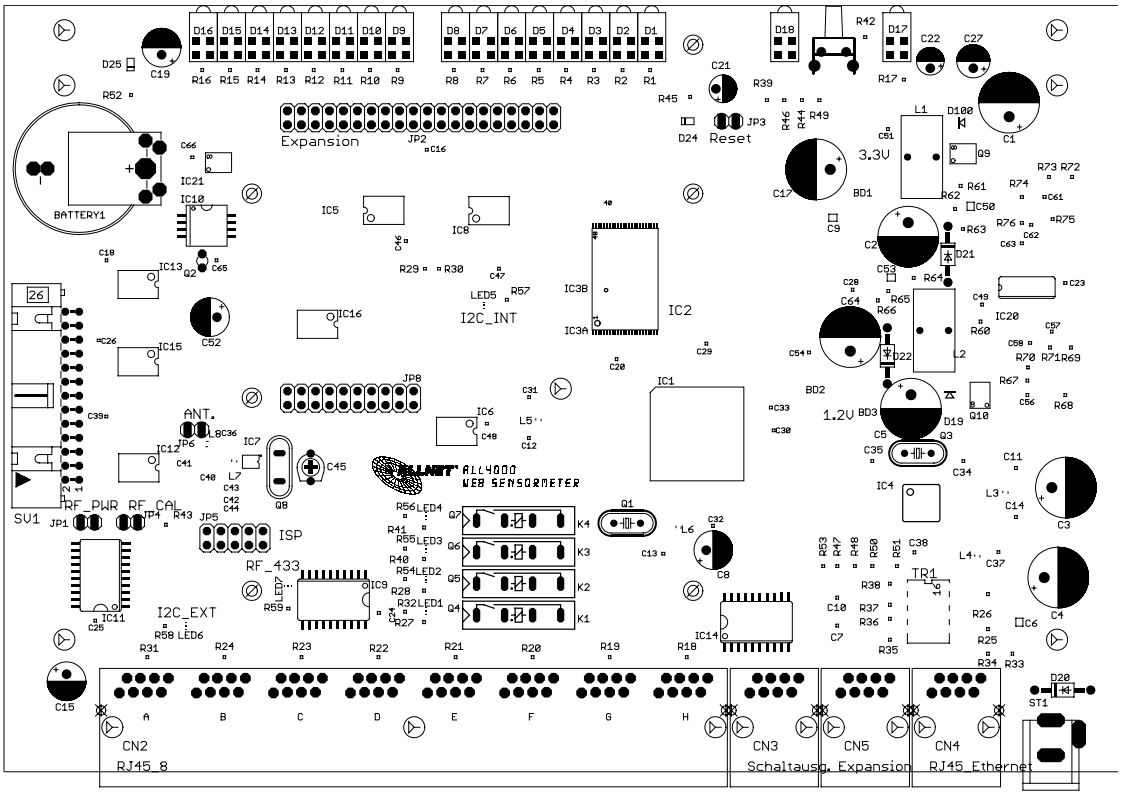
Zeile 70: Solange A größer als 1 ist, soll dies wiederholt werden: Gehe zu Zeile 50

Zeile 80: Ausgangszustand erreicht, Springe zurück zum Anfang !

# Hardware-Kurzbeschreibung

Übersicht über die Schnittstellen des ALL4000

### Ansicht der Hauptplatine des ALL4000




# Steckerbelegungen

### Stromversorgungsstecker ST1

Standard-Buchse für Hohlstecker mit 1.6 mm Stift.
Innen: +, Außen: -
Versorgungsspannung 5...24 VDC, ca. 190...350 mA

### Netzwerkanschluß CN4

Standard-RJ45-Buchse für die Verbindung des ALL4000 mit dem Switch oder Hub.
Pin1: TX+
Pin2: TX-
Pin3: RX+
Pin4: 75R nach Masse
Pin5: 75R nach Masse
Pin6: RX-
Pin7: 75R nach Masse
Pin8: 75R nach Masse

**Hinterer Expansionsstecker CN5**
Standard-RJ45-Buchse, derzeit noch ohne Verwendung.
(möglicherweise später SPI-Port für 3.3V DOG LCD)
Pin1: Masse
Pin2: +3,3 V
Pin3: (G17 in)
Pin4: (G16 out)
Pin5: (G13 out)
Pin6: (G20 out)
Pin7: (G21 out)
Pin8: (G24 out)

**Relaisausgang CN3**
Standard-RJ45-Buchse führt die Relaiskontakte nach außen.
Reedrelais, max. Belastbarkeit 75V/100 mA !
Pin1 + 2: Relais 0
Pin3 + 4: Relais 1
Pin5 + 6: Relais 2
Pin7 + 8: Relais 3

**Sensoranschlüsse CN2 A...H**
8 Standard-RJ45-Buchsen für den Anschluß der I2C-Sensoren und Funkmodule des ALL3000/ALL4000 Programms
Pin1: Unregulierte Eingangsspannung von ST1-Mitte
Pin2: n.c.
Pin3: I2C Clock
Pin4: I2C Data
Pin5: Masse
Pin6: +3,3 V
Pin7: n.c.
Pin8: RF Clock

**Seitlicher Expansionsstecker SV1**
gewinkelter Wannenstecker, 26-polig, zur freien Verwendung
Pin1: Masse
Pin2: +3,3 V
Pin3: Chip3 Bit 0
Pin4: Chip5 Bit 7
Pin5: Chip3 Bit 1
Pin6: Chip5 Bit 6
Pin7: Chip3 Bit 2
Pin8: Chip5 Bit 5
Pin9: Chip3 Bit 3
Pin10: Chip5 Bit 4
Pin11: Chip3 Bit 4
Pin12: Chip5 Bit 3
Pin13: Chip3 Bit 5
Pin14: Chip5 Bit 2
Pin15: Chip3 Bit 6
Pin16: Chip5 Bit 1
Pin17: Chip3 Bit 7
Pin18: Chip5 Bit 0
Pin19: Chip4 Bit 0
Pin20: Chip4 Bit 7
Pin21: Chip4 Bit 1
Pin22: Chip4 Bit 6
Pin23: Chip4 Bit 2
Pin24: Chip4 Bit 5
Pin25: Chip4 Bit 3
Pin26: Chip4 Bit 4

**Innerer Expansionsstecker JP9**
Pfostenstecker, 20-polig, zur freien Verwendung
Pin1: Chip7 Bit 0
Pin2: Chip6 Bit 0
Pin3: Chip7 Bit 1
Pin4: Chip6 Bit 1
Pin5: Chip7 Bit 2
Pin6: Chip6 Bit 2
Pin7: Chip7 Bit 3
Pin8: Chip6 Bit 3
Pin9: Chip7 Bit 4
Pin10: Chip6 Bit 4
Pin11: Chip7 Bit 5
Pin12: Chip6 Bit 5
Pin13: Chip7 Bit 6
Pin14: Chip6 Bit 6
Pin15: Chip7 Bit 7
Pin16: Chip6 Bit 7
Pin17: Masse
Pin18: Masse
Pin19: +3,3 V
Pin20: unregulierte Eingangsspannung von ST1

**Vorderer Expansionsstecker JP2**
Pfostenstecker, 40-polig, 16 Pins LED´s, 16 direkte CPU-Verbindungspins
Pin1: CPU RD0
Pin2: Chip0 Bit 0
Pin3: CPU RD1
Pin4: Chip0 Bit 1
Pin5: CPU RD2
Pin6: Chip0 Bit 2
Pin7: CPU RD3
Pin8: Chip0 Bit 3
Pin9: CPU RD4
Pin10: Chip0 Bit 4
Pin11: CPU RD5
Pin12: Chip0 Bit 5
Pin13: CPU RD6
Pin14: Chip0 Bit 6
Pin15: CPU RD7
Pin16: Chip0 Bit 7
Pin17: unregulierte Eingangsspannung von ST1
Pin18: unregulierte Eingangsspannung von ST1
Pin19: +3,3 V
Pin20: +3,3 V

Pin21: CPU RD8
Pin22: Chip1 Bit 0
Pin23: CPU RD9
Pin24: Chip1 Bit 1
Pin25: CPU RD10
Pin26: Chip1 Bit 2
Pin27: CPU RD11
Pin28: Chip1 Bit 3
Pin29: CPU RD12
Pin30: Chip1 Bit 4
Pin31: CPU RD13
Pin32: Chip1 Bit 5
Pin33: CPU RD14
Pin34: Chip1 Bit 6
Pin35: CPU RD15
Pin36: Chip1 Bit 7
Pin37: Int. I2C Data
Pin38: Int. I2C Clock
Pin39: Masse
Pin40: Masse

**Stecker für Programmiergerät JP5**
Pfostenstecker, 10-polig (nicht für Anwender)
Pin1: CPU RH5
Pin2: CPU /TSS
Pin3: Masse
Pin4: CPU TSCK
Pin5: n.c.
Pin6: CPU RH4
Pin7: n.c.
Pin8: CPU TSI
Pin9: +3,3 V
Pin10: CPU TSO

## Jumper

**Stromversorgung für Funksender JP1 (RF_PWR)**
Über diesen Jumper wird der 433-MHz-Funksender mit Strom versorgt.
Jumper ist standardmäßig gesetzt, Abziehen macht den Sender stromlos.

**Funksender-Kalibrierung JP4 (RF_CAL)**
Setzen dieses Jumpers läßt den Funksender kontinuierlich abeiten, um die Frequenz abgleichen zu können.

**Antennenwahl JP6 (ANT.)**
Über diesen Jumper wird der Ausgang des Funksenders mit der Onboard-Antenne verbunden.
Jumper ist standardmäßig gesetzt, Abziehen deaktiviert die eingebaute Antenne - in diesem Fall muss eine geeignete Antenne am rechten Pin dieses Jumpers angeschlossen werden.

**Reset JP3**
Kurzschließen dieses Jumpers löst einen Reset und Neustart des ALL4000 aus.