

Projektarbeit

Entwurf einer Regelung zur Anbindung
regenerativer Wärmequellen
an bestehende Heizungsanlagen

Malte Alpers, Matrikel-Nr.: 16965062
Christoph Niemand, Matrikel-Nr.: 16005015

Projektarbeit im Labor im Thermodynamik und Energietechnik

Thema: Entwurf einer Regelung zur Anbindung regenerativer Wärmequellen an bestehende Heizungsanlagen.

Problem: Regelungseinrichtungen von führenden Heizungsanlagenbauern sind oft wenig flexibel bzw. lassen eine Kombination von Komponenten unterschiedlicher Hersteller nur bedingt zu.

Aufgabe:

- Entwurf der hydraulischen Anbindung und Positionierung der Temperaturmessstellen
- Festlegen von Schaltkriterien zur Ansteuerung von Pumpen / Ventilen für die verschiedenen Betriebsbedingungen
- Entwurf einer Mikrokontrollereinrichtung bzw. Auswahl der geeigneten Komponenten und Programmierung der Regelung
- Die Regelung soll folgende Anforderungen erfüllen:
 - als „Add-On“ für den nachträglichen Einbau
 - kompatibel mit Bauteilen unterschiedlicher Hersteller
 - einfache Kopplung der Heizsysteme
 - unkomplizierte Bedienung und Wartung
 - überschaubare Kosten für Anschaffung und Betrieb
 - hohe Betriebssicherheit

Inhaltsverzeichnis

1	Vorbemerkungen	3
1.1	Auswahl der Komponenten.....	3
1.2	Besonderheiten regenerativer Energien	4
2	Hydraulische Einbindung und Funktion	5
2.1	Begriffe und Hinweise.....	5
2.2	Festbrennstoff-Heizkessel zur Rücklaufanhebung	7
2.3	Solkollektor zur Trinkwassererwärmung.....	10
3	Hardware	11
3.1	Beschreibung der Komponenten	11
3.1.1	Reglerboard (CC-TOOLS).....	12
3.1.2	C-Control II Unit (CONRAD ELECTRONIC).....	14
3.2	Temperatursensoren	15
3.2.1	Übersicht.....	15
3.2.2	Positionierung	17
3.2.3	Messfehler	18
3.3	Stückliste	21
3.4	Bestückung des Reglerboards.....	22
3.4.1	Messkomponenten für KTY-Sensoren.....	22
3.4.2	Messkomponenten für Pt-Sensoren	23
3.4.3	Messkomponenten für AD592-Sensoren.....	24
3.5	Anschluss weiterer Komponenten an das Reglerboard.....	25
3.6	Abgleich der Messadapter	26
4	Programmierung.....	27
4.1	Vorraussetzungen.....	28
4.2	Regelungsgrundlagen.....	28
4.3	Module der Regelung	30
4.3.1	Modul „temp“ (Temperatursensor-Auswertung).....	31
4.3.2	Modul „watch“ (Fehlerüberwachung).....	33
4.3.3	Modul „control“ (Regelungsfunktionen).....	34
4.3.4	Modul „display“ (LC-Display-Ausgabe).....	37
4.3.5	Modul „terminal“ („Wartung“ über PC)	37
4.4	Programmvarianten	38
5	Ausblick.....	43
5.1	Sonderfunktionen.....	43
6	Fazit	49
7	Anhang	50
7.1	Anlagen.....	50
7.2	Hinweise zur Programmierung in C2	50
7.3	Programm-Quelltexte.....	56
7.3.1	Modul „temp“ (Temperatursensor-Auswertung).....	56
7.3.2	Modul „watch“ (Fehlerüberwachung).....	59
7.3.3	Modul „control“ (Regelungsfunktionen).....	63
7.3.4	Modul „display“ (LC-Display-Ausgabe).....	72
7.3.5	Modul „terminal“ („Wartung“ über PC)	76
7.4	Bedienungsanleitung - Modul „terminal“	87
8	Quellen	91

1 Vorbemerkungen

Die Autoren hatten nur geringe Vorkenntnisse in Heizungs-, Elektrotechnik und Programmierung.

Die Regelung wurde nur im Labor mit simulierten Temperaturen und nicht an einer bestehenden Heizungsanlage getestet.

Es wurden die marktüblichen Funktionen der gängigen Heizungssystemanbieter anhand deren Planungsunterlagen und Handbücher analysiert.

Es soll nicht unerwähnt bleiben, dass bei der hier vorgestellten Lösung für „Elektronik-Laien“ einige kleine Hürden im Hardwarebereich zu überwinden sind. Es sollten grundlegende Erfahrungen im Umgang mit Lötkolben, elektrischen Messgeräten und elektronischen Bauelementen vorhanden sein.

1.1 Auswahl der Komponenten

Prinzipiell kommen zur Realisierung dieses Projekts viele Möglichkeiten in Betracht. So z.B. die in „Bastlerkreisen“ weit verbreiteten (weil kostengünstigen und leistungsfähigen) AVR Mikrokontroller von ATMEL. Der Nachteil ist jedoch, dass für diese relativ wenige fertige Zubehörkomponenten erhältlich sind, und somit ein erheblicher „Bastelaufwand“ und sehr gute Elektronikkenntnisse erforderlich wären. Außerdem ist der Einarbeitungs- und Programmieraufwand sehr hoch. Lösungen mit den Produkten der in der Industrie etablierten Anbieter (z.B. Siemens, Moeller) sind entweder wenig flexibel oder deutlich teurer (insbesondere die Erweiterungsmodule).

Da die Lösung auch (relativ) einfach nachzubauen sein sollte, wurde der Mikrokontroller C-Control II (CC2) von CONRAD ELECTRONIC und die darauf abgestimmten Zubehörkomponenten von CC-TOOLS gewählt. Da der Inhaber von CC-TOOLS (André Helbig) als Solarteur tätig ist, sind hier viele, für den Aufbau einer Heizungsregelung geeignete, Komponenten erhältlich (z.B. das Reglerboard). Zudem gibt es ein zentrales Supportforum zur CC2 („CC2Net.de“, betreut vom Inhaber von CC-TOOLS) mit vielen hilfreichen Beiträgen / Informationen.

Vorteil der CC2 gegenüber anderen Mikrocontrollern (z.B. ATMEL) ist das Betriebssystem, welches die Ansteuerung der Ein-/Ausgänge und das Multithreading übernimmt. Dadurch wird das Programmieren wesentlich erleichtert.

Ein weiterer Vorteil gegenüber Standard-Heizungsregelungen ist die fast unbegrenzte Erweiterungsmöglichkeit und Anpassung an veränderte Bedingungen. Es können parallel auch noch weitere, nicht heizungstechnische, Aufgaben bearbeitet werden.

Durch die direkte Programmierung im Quelltext erhält der Programmierer die volle Kontrolle über die Funktionen der Regelung.

Nachteil der CC2 ist der relativ hohe Preis des Mikrocontrollers (z.B. gegenüber ATMEL).

1.2 Besonderheiten regenerativer Energien

Unter „regenerativen Energien“ sollen hier die Verbrennung von Festbrennstoffen (Scheitholz, Pellets, Getreide usw.) und die Wärmenutzung der Sonnenenergie (Solarthermie) verstanden werden.

Festbrennstoffe

Problematisch ist hier die schlechte Regelbarkeit. Die Feuerung kann nicht beliebig Ein / Aus geschaltet werden und im Teillastbereich (regelbare Kessel) verschlechtern sich die Abgaswerte stark. Ein Festbrennstoffkessel sollte daher in Kombination mit einem Pufferspeicher (ausreichend groß; teilweise Vorgeschrieben) und unter Volllast betrieben werden. Dadurch wird außerdem der Bedienkomfort verbessert, indem nicht sofort bei einem Wärmebedarf bzw. seltener Brennstoff nachgelegt werden muss.

Ein Sonderfall sind vollautomatische Pellet- bzw. Schüttgut-Heizkessel, die automatisch anfahren und abschalten. Durch die automatische Brennstoffzufuhr als Schüttgut ist ein (quasi) Ein/Aus-Betrieb möglich.

Für einen einwandfreien Betrieb muss eine Mindestrücklauftemperatur (i.d.R. 65 °C) eingehalten werden, um den Rauchgastaupunkt nicht zu unterschreiten (Korrosion). Dazu wird eine Rücklaufanhebung eingebunden. Das Vorlaufwasser wird solange über einen Bypass zurückgeführt, bis die Mindesttemperatur erreicht ist.

Des Weiteren ist eine „thermische Ablaufsicherung“ erforderlich bzw. vorgeschrieben. Wird die Wärme des Kessels nicht abgenommen, steigt die Temperatur im Kessel, bis bei 95 °C das Sicherungsventil öffnet und über einen separaten, internen Wärmetauscher die Überschusswärme an kaltes Frischwasser abgeführt wird. Im Normalbetrieb sollte ein Ansprechen der Ablaufsicherung verhindert werden, indem der Pufferspeicher ausreichend dimensioniert wird (der Speicher sollte mindestens die Wärme einer Charge aufnehmen können). Hier ist auch der Betreiber für eine korrekte Funktion verantwortlich. So sollte kein Holz nachgelegt werden, wenn der Speicher schon (fast) bis zu seiner Maximaltemperatur (z.B. 90 °C) geladen ist.

Solarthermie

Hier ist die Verfügbarkeit problematisch (die Sonne scheint nicht immer dann, wenn Wärme benötigt wird). Daher ist auch hier ein Speicher erforderlich. In Europa sind wegen der Frostgefahr Zweikreissysteme üblich. Hier ist der Solarkreis hydraulisch vom restlichen System getrennt und mit einer Wärmeträgerflüssigkeit gefüllt.

Gewöhnlich wird die Solarthermie zur Erwärmung von Brauchwasser verwendet. Bei ausreichender Dimensionierung der Anlage kann diese aber auch zur Heizungsunterstützung verwendet werden. Dies ist besonders für die Übergangszeiträume (Frühjahr / Herbst) geeignet, wenn niedrige Vorlauftemperaturen im Heizkreis benötigt werden.

Für einen guten Wirkungsgrad sollte der Rücklauf (zum Kollektor) möglichst kalt sein. Daher ist hier auch eine ausgeprägte Temperaturschichtung im Speicher (warmes Wasser oben, kaltes Wasser unten) besonders sinnvoll. Bei Solaranlagen sind einige Sonderfunktion sinnvoll (siehe Kap. 5).

2 Hydraulische Einbindung und Funktion

Zur Beschreibung der grundsätzlichen Möglichkeiten und der Programmfunktionen werden hier exemplarisch zwei einfache und bewährte Anlagentypen vorgestellt. Die Möglichkeiten hängen von den vorhandenen Einrichtungen ab, und davon wie weit in bestehende Regelungen eingegriffen werden soll / darf (Vorschriften, Garantien beachten!).

Für weitere Varianten siehe Fachliteratur, Planungsunterlagen der Heizungssystemanbieter, [1] und [7].

Die hier vorgestellten Hydraulikschemas sind nur Prinzipschemas. Für einen korrekten Betrieb sind noch weitere Einbauten erforderlich (Sicherheitsventile, Rückschlagklappen usw.). Des Weiteren sind die hydraulischen Bedingungen zu beachten. Es sind die gesetzlichen und herstellerepezifischen Vorschriften / Normen zu berücksichtigen.

Eine optimale Energieausnutzung kann nur durch eine abgestimmte Regelung der gesamten Heizungsanlage erreicht werden. Die CC2 ist dafür leistungsfähig genug und nur durch den Programmieraufwand begrenzt.

2.1 Begriffe und Hinweise

Begriffe

- Kessel : Festbrennstoff-, Öl-, Gasheizkessel (Wärmeerzeuger)
- Speicher : Mit Wasser gefüllter Behälter zur Speicherung von Überschusswärme
- Vorlauf : Warmes Wasser zum Verbraucher
- Rücklauf : Kaltes Wasser vom Verbraucher

- Geschlossenes System
Die Kompensation der Volumenänderung des Heizungswassers durch Temperaturänderungen im System (Druckausgleich) erfolgt durch ein Ausdehnungsgefäß.
- Offenes System
Die Kompensation der Volumenänderung des Heizungswassers erfolgt durch ein Ausdehnungsgefäß, das mit der Umgebungsluft in Verbindung steht. Hier kommt es zum Sauerstoffeintrag in das System (Korrosionsgefahr).
- Schichtbeladung
Um möglichst schnell ein nutzbares Temperaturniveau im Speicher zu erhalten, kann der Solarkreis im „low-flow“-Betrieb (geringer Durchfluss) betrieben werden. Durch die längere Verweilzeit im Kollektor heizt sich das Wärmeträgermedium stärker auf. Um dieses Temperaturniveau optimal zu nutzen, ist eine ausgeprägte Temperaturschichtung erforderlich. Dazu wird der Kollektorstromlauf in die Temperaturschicht des Speichers mit einer ähnlichen Temperatur geleitet um die Schichtung nicht zu zerstören. Dies kann extern durch mehrere Einspeiseanschlüsse (meist 2) und Umschaltventil(e), gesteuert durch die Heizungsregelung (aktiv), oder selbst regulierend durch interne Einbauten (passiv) erfolgen (Schichtenspeicher).

Speicherarten

- Pufferspeicher
Heizwasserspeicher, i.d.R. einfacher Behälter ohne interne Wärmetauscher. Alle von Heizwasser durchflossenen Systeme werden direkt angeschlossen.
- Warmwasserspeicher (WW-Speicher)
Auch Brauchwasserspeicher oder Boiler genannt. Für die Trinkwassererwärmung. Meist als bivalenter Speicher ausgeführt: kann über zwei interne Wärmetauscher von zwei Wärmequellen beladen werden (z.B. solare Beladung mit Nachheizung durch Öl-/Gaskessel).
- Kombispeicher
Heizwasser-Pufferspeicher mit integrierter Trinkwassererwärmung (meist für mehrere Wärmequellen). Es sind zahlreiche unterschiedliche Bauarten am Markt erhältlich [3]. Nachteil: die Boilertemperatur kann nicht niedriger als die Speichertemperatur sein (Verkalkungsgefahr).
- Schichtspeicher
Grundsätzlich ist jeder Speicher ein Schichtspeicher, wenn die Schichtung nicht zerstört wird: z.B. Einspeisung senkrecht zur Schichtung, Prallblech am Frischwassereintritt. Bei speziellen Schichtspeichern wird durch besondere Einbauten versucht die Schichtung herzustellen und aufrechtzuerhalten (meist bei Kombispeichern). Siehe auch „Schichtbeladung“ (oben).

Hinweise

Bei größeren Speichern werden meist externe Wärmetauscher verwendet. Diese haben eine bessere Wärmeübertragung, geringere Verkalkungsneigung und es können mehrere Speicher mit einem Wärmetauscher beladen werden.

Auch der Öl-/Gasheizkessel sollte die Möglichkeit zur Pufferung erhalten, um das Takten zu verringern (z.B. Kombispeicher).

Schichtenspeicherung ist nur bei darauf abgestimmten Komponenten, Regelung und im „low-flow“-Betrieb sinnvoll (teuer, aufwändig). Vgl. [3], [5] und [7].

Grundsätzlich sollte die erzeugte Wärme direkt für den Wärmebedarf genutzt werden und nur die Überschusswärme in den Speicher geleitet werden (u.a. Wärmeverluste des Speichers). Der Speicher sollte daher auch nicht zu groß gewählt werden. Es kann hier jedoch zu Problemen kommen (z.B. Wärmestöße in den Heizkreis).

Es sollte auf eine gute Wärmedämmung geachtet werden. Dabei sind auch die Anschlüsse und Armaturen zu beachten. Hier entstehen häufig Wärmebrücken, über die ein nicht unerheblicher Wärmeverlust entstehen kann (hauptsächlich oben). Deshalb sollte die Warmwasserentnahmeleitung von unten in den Speicher geführt sein. Das Dämmmaterial muss für den Einsatzzweck geeignet sein (witterungs-, temperaturbeständig). Vgl. [1] und [4].

Für weitere Informationen siehe [3] und [7].

2.2 Festbrennstoff-Heizkessel zur Rücklaufanhebung

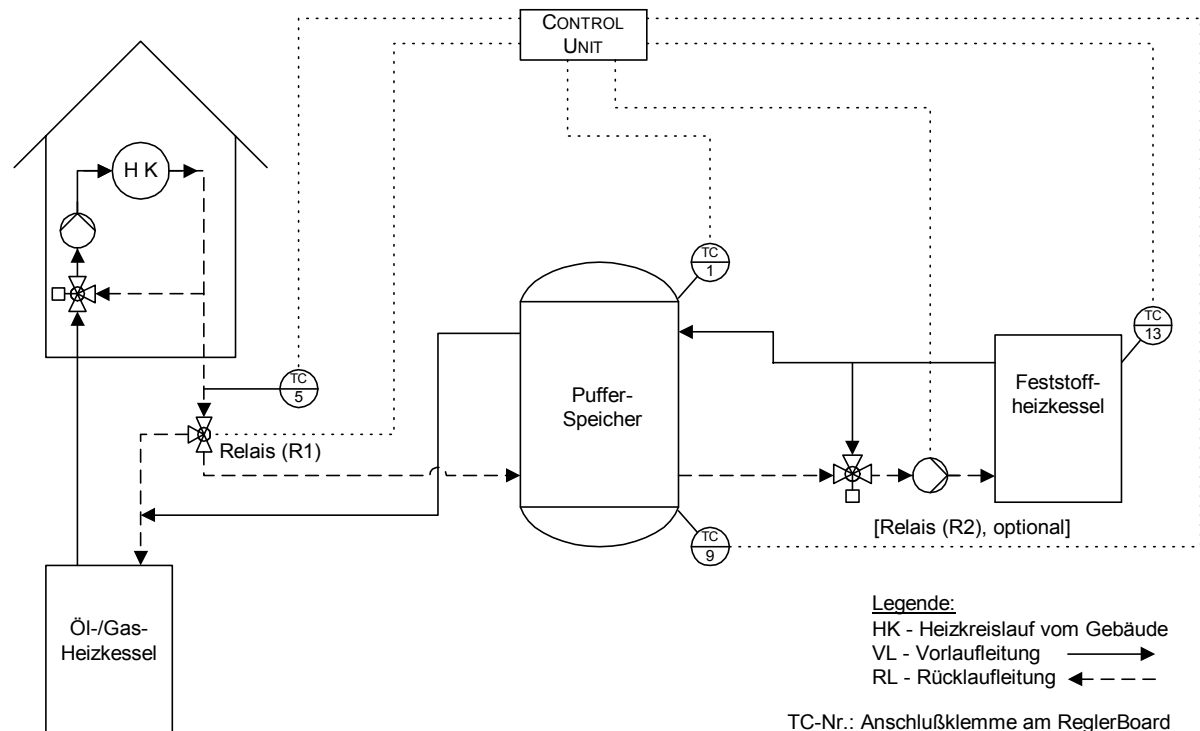


Abbildung 1 Hydraulikschema Festbrennstoffkessel zur HK-Rücklaufanhebung

Hier wird z.B. ein vorhandenes Heizungssystem durch einen Festbrennstoffkessel in Kombination mit einem Pufferspeicher erweitert.

Funktionsprinzip:

Ist die Temperatur im Feststoffkessel (TC 13) größer als die Temperatur im Pufferspeicher unten (TC 9) und die Mindesttemperatur im Kessel erreicht, wird die Speicherladepumpe (R2) eingeschaltet. Der Pufferspeicher heizt sich auf.

Ist die Speichermaximaltemperatur erreicht, die Kesselminimaltemperatur unterschritten oder die Temperatur im Kessel kleiner als die untere Speichertemperatur, wird die Ladepumpe ausgeschaltet.

Ist die obere Speichertemperatur (TC 1) größer als die Rücklauftemperatur (TC 5) aus dem Heizkreis (HK), wird das 3-Wege-Ventil (R1) umgeschaltet und der HK-Rücklauf durch den Pufferspeicher geleitet. Wärmeres Wasser aus dem oberen Speicherbereich fließt in den Öl-/Gasheizkessel (Rücklaufanhebung). Dieser muss nun nur noch, bei Bedarf, auf die gewünschte Vorlauftemperatur des HK nachheizen.

Unterschreitet die obere Speichertemperatur die Rücklauftemperatur, wird das Ventil wieder zurückgeschaltet und der HK-Rücklauf fließt direkt in den Öl-/Gaskessel.

Merkmale:

- Der Pufferspeicher wird nur zur Beladung durch den Festbrennstoffkessel verwendet.
- Geringer Installationsaufwand. Optimal zur Nachrüstung bestehender Heizungsanlagen z.B. mit Öl-/Gaskessel mit angeflanschem Warmwasserspeicher.
- Für einen sinnvollen Betrieb muss der Heizkreis mit einer Mischerregelung versehen sein.
- Der Öl-/Gaskessel wird immer durchströmt
- Regelung des Feststoffkessels optional durch Control Unit
- Nachteil: Die vom Festbrennstoffkessel erzeugte Wärme wird nicht direkt genutzt (Wärmeverluste; Speicher muss erst aufgeladen werden).
- Optimierung: Bei ausreichendem Temperaturniveau im Speicher könnte der Öl-/Gaskessel mit einem 3-Wege-Ventil hydraulisch abgetrennt werden, um Auskühlungsverluste durch den Kessel zu verhindern.
Evtl. Ausschaltverzögerung der Ladepumpe vorsehen um die Restwärme beim Ausbrand zu nutzen.
- Die Nachheizung durch den Öl-/Gaskessel könnte durch Vortäuschen einer hohen Kesseltemperatur unterdrückt werden, wenn der Festbrennstoffkessel in Betrieb ist (siehe „Nachheizunterdrückung“ im „Ausblick“).

Alternative: Speicherentladung durch externen Wärmetauscher

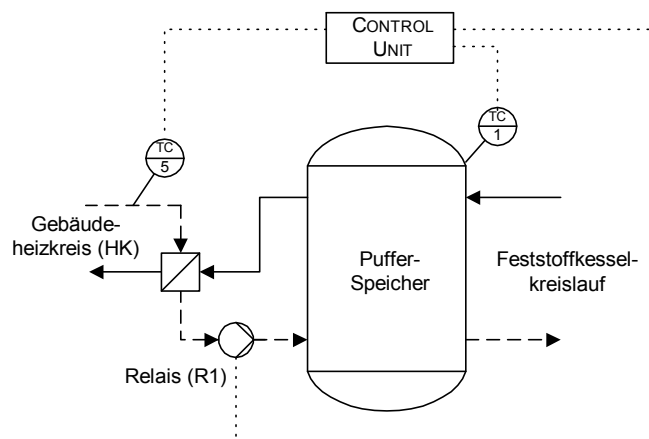


Abbildung 2 Variante mit externem Wärmetauscher

Hier wird das 3-Wege-Ventil in Abbildung 1 durch einen externen Wärmetauscher ersetzt. Dadurch ist der Feststoffkesselkreis hydraulisch vom Heizungssystem getrennt. So können z.B. offene und geschlossene Systeme miteinander kombiniert werden. Die Regelung funktioniert prinzipiell wie zuvor beschrieben.

Merkmale:

- Minimale Änderungen am bestehenden Heizungssystem nötig.
- Vollständige hydraulische Trennung des Feststoffkesselkreises vom Heizungssystem.
- Problem: Es wird von der Regelung der Entladepumpe nicht erkannt, ob vom Heizkreis (HK) Wärme abgenommen wird. Dadurch kommt es zu unnötiger Zirkulation im Entladekreis (Energievergeudung, Wärmeverluste).
Als Lösung könnte über ein Hilfsrelais (angesteuert über HK-Pumpe) eine Spannung (5 V) an einen Digitalport der Regelung angelegt werden. Der Zustand des Digitalports wird dann bei den Ein- / Ausschaltbedingungen der Entladepumpe abgefragt.

2.3 Solarkollektor zur Trinkwassererwärmung

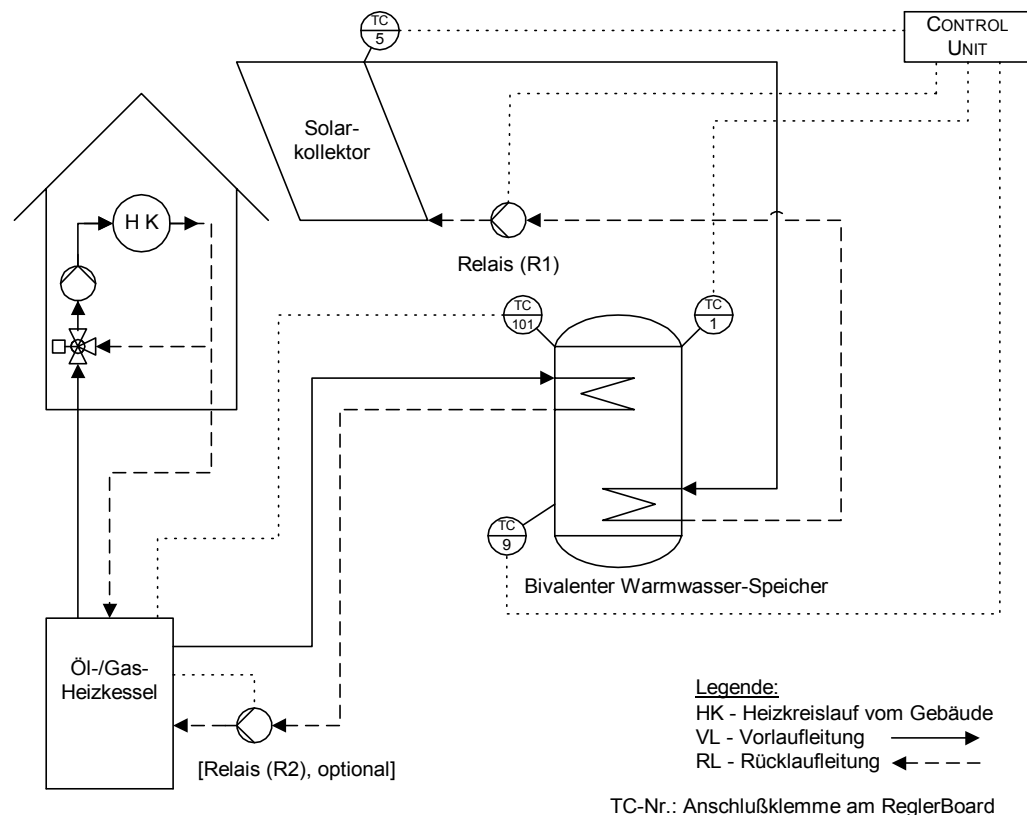


Abbildung 3 Hydraulikschema Solare Trinkwassererwärmung

Hier wird z.B. das bestehende Heizungssystem durch einen Solarkreis zur Trinkwassererwärmung erweitert. Dazu wird z.B. der vorhandene Trinkwasserspeicher durch einen bivalenten Warmwasserspeicher ersetzt, der solar und konventionell beladen werden kann.

Funktionsprinzip:

Übersteigt die Kollektortemperatur (TC 5) die untere Speichertemperatur (TC 9), wird die Solarkreispumpe (R1) eingeschaltet und der WW-Speicher aufgeheizt. Sinkt die Kollektortemperatur unter die untere Speichertemperatur, wird die Solarpumpe wieder ausgeschaltet. Reicht die Solarwärme nicht aus, um die gewünschte Warmwassertemperatur (TC 1) zu erreichen, wird die Nachladepumpe (R2) so lange eingeschaltet, bis diese erreicht ist.

Merkmale:

- Die Solarwärme wird nur zur Trinkwassererwärmung genutzt.
- Wegen der unsicheren Verfügbarkeit der Solarenergie ist i.d.R. eine Nachheizung erforderlich. Meist kann dazu der vorhandene Öl-/Gaskessel in Verbindung mit einer Nachladepumpe (Umwälzpumpe) verwendet werden. Sonst ist z.B. ein Gasdurchlauferhitzer oder ein elektrischer Heizstab erforderlich.
- Ansteuerung der Nachladepumpe optional durch Control Unit
- Alle Bauteile im Solarkreis müssen für Temperaturen $> 130\text{ °C}$ geeignet sein.
- Optimierung: Unterdrückung der Nachheizung wenn der Speicher solar beladen wird (siehe „Nachheizunterdrückung“ im „Ausblick“).

3 Hardware

3.1 Beschreibung der Komponenten

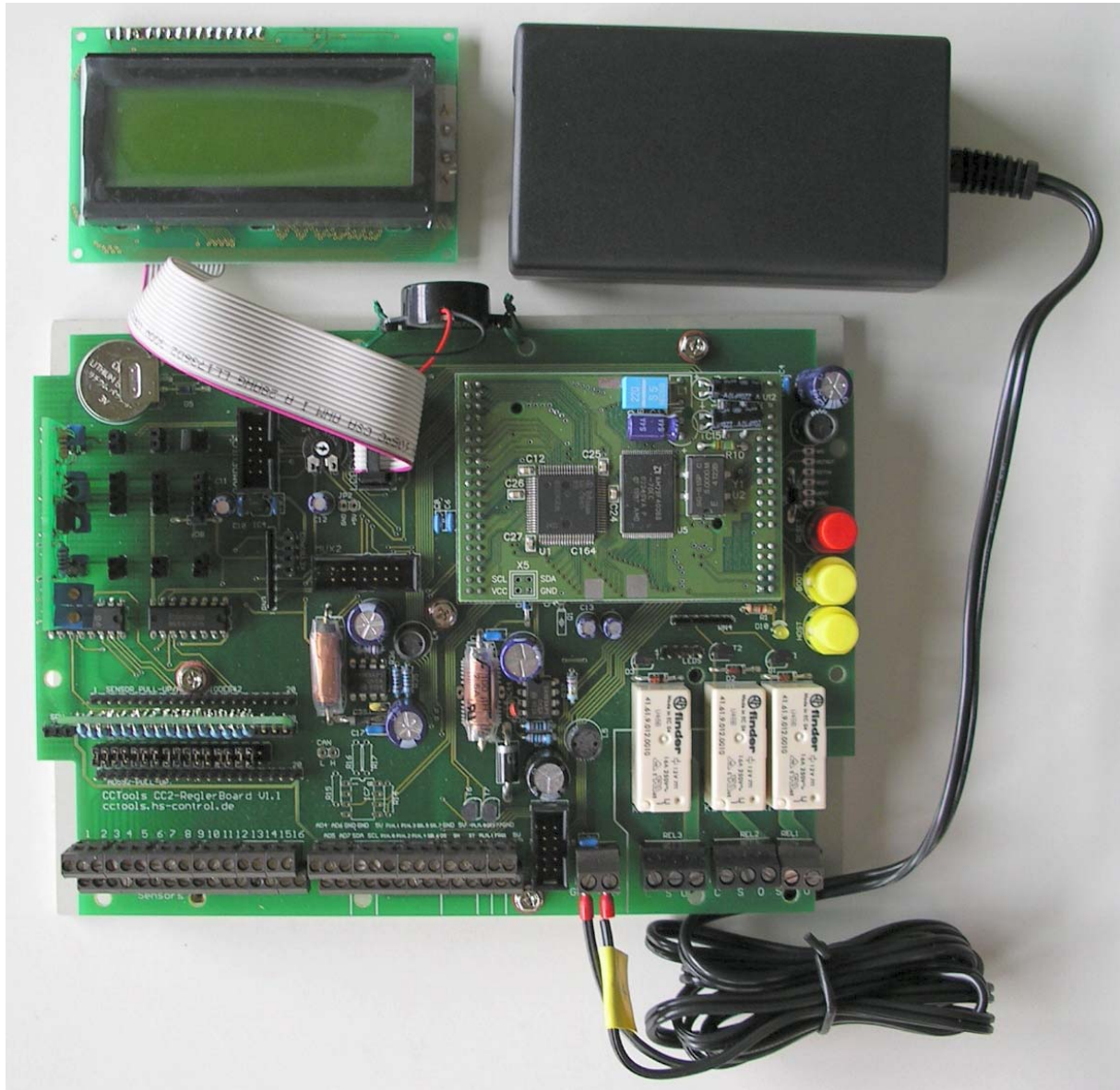


Abbildung 4 Reglerboard mit C-Control II Unit, Messadapter, LC-Display, „Beeper“ und Netzteil

Das Reglerboard von CC-TOOLS auf Basis der C-Control II Unit von CONRAD ELECTRONIC wurde speziell für die Heizungsregelung entwickelt. So können über aufsteckbare Zusatzplatinen bis zu 16 Temperatursensoren (über Multiplexer) ausgewertet werden. Es sind drei Relais zum Schalten von Pumpen, Ventilen usw. und Anschlüsse für LC-Display und Matrix-Tastatur vorhanden. Alle Anschlüsse sind als Schraubklemmen bzw. Buchsen für Pfostenstecker für einfachen Anschluss ausgeführt. Das Platinenlayout ist für den Einbau in das Bopla Regler-Gehäuse RCP2000 abgestimmt.

Durch zahlreiche Zubehörkomponenten kann das System erweitert werden:

- Relaisplatinen für weitere Relais (auch elektronische Lastrelais)
- Porterweiterungen für weitere Ein-/Ausgänge
- Speichererweiterungen
- ...



Bemerkung: Alle Platinen sind in Industriequalität gefertigt.

Abbildung 5 Bopla Gehäuse RCP2000

3.1.1 Reglerboard (CC-TOOLS)

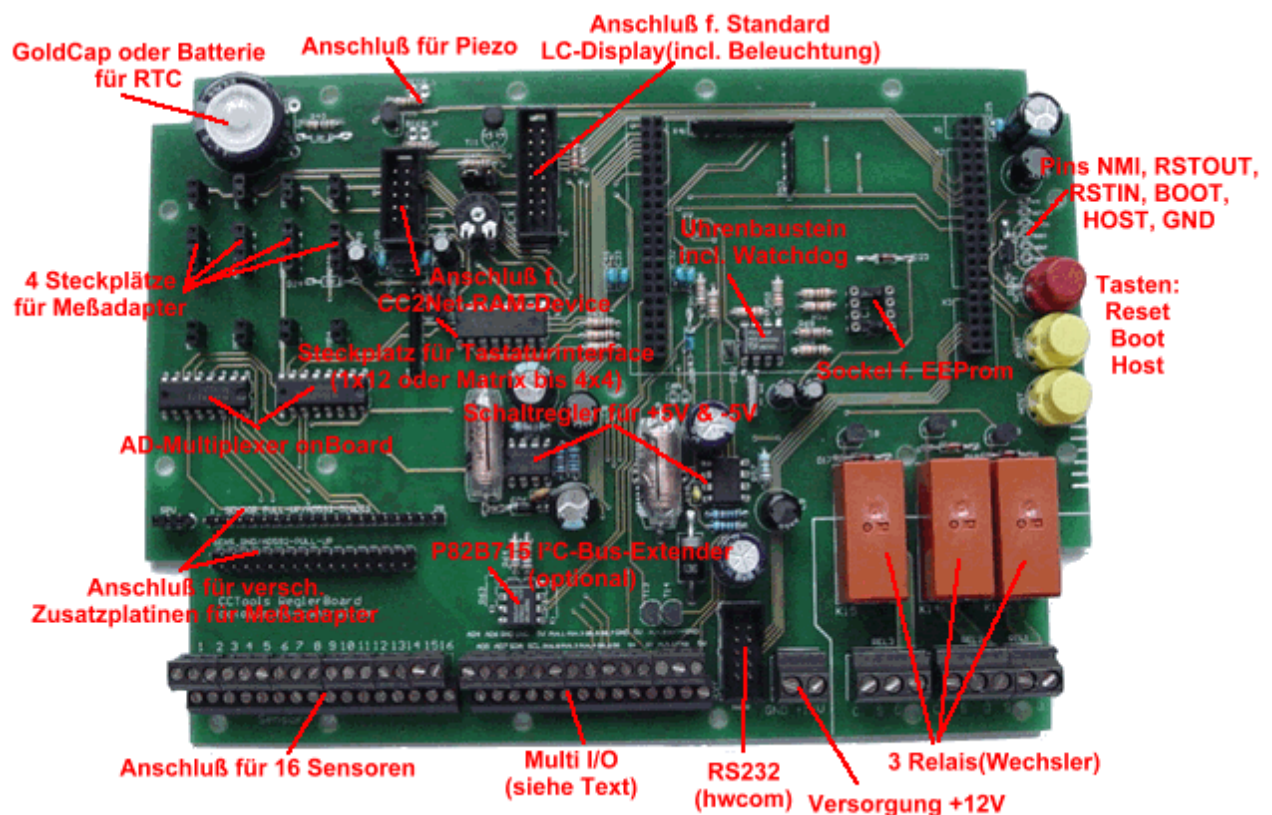


Abbildung 6 Reglerboard (Quelle: CC-TOOLS)

Merkmale (Quelle: CC-TOOLS):

- 3 Relais mit Wechselkontakten (max. 8A, 250V)
angesteuert über ein Schieberegister.
- Anschluss für Standard LC-Displays im 4Bit-Mode incl. Beleuchtung (16pol.)
- Löt pads für Piezo-Schallwandler. Entweder an 5V oder an 12V, verstärkt
über eine Transistorstufe, für eine größere Lautstärke.
- Schaltregler für +5V (max. 750mA) und -5V (für Messadapter)
- 4 Steckplätze für Messadapter mit +5V und -5V Versorgung
- AD-Multiplexer. So werden nur 4 Messadapter und AD-Ports für
16 Sensoren benötigt. Angesteuert wird der Multiplexer über ein
Schieberegister.
- I²C-Bus Echtzeituhr, Batterie- oder GoldCap-gepuffert
mit Watchdog-Funktion (Timeout über Softwareeinstellbar)
- Sockel für ein seriellen I²C-Bus EEPROM
- P82B715 I²C-Bus Extender (optional), um den I²C-Bus als Feldbus
zu verwenden
- Sockel für Tastaturadapter (f. 1x12 oder bis zu 4x4 Matrixtastatur)
- Löt pads für CAN-Bus
- 3 Taster: Reset, Boot, Host
- Löt pads für: NMI, RSTIN, RSTOUT, BOOT, HOST

- Alle restlichen Anschlüsse herausgeführt:(mittlere Klemmleiste)
 - I²C-Bus (SDA,SCL)
 - 4 freie AD-Ports (Bei der Verwendung einer Tastatur ist AD7 belegt)
 - Ports P1H.0 bis P1H.4
 - restliche Ports des Schieberegisters SR.5, 6, 7
 - Anschluss für weitere Schieberegistererweiterungen
(DS,SHCLK(=P1H.6),STCLK(=P1H.7))
 - PLM.0 & 1 mit vorgeschaltetem Transistor (Open-Kollektor) (max.500mA)
 - DCF77(FRQ.0) & FRQ.1 (jeweils mit 10k-Pullup)
 - +5V Ausgang max. 750mA
 - 10pol. Wannenstecker für serielle Schnittstelle (hwcom) passend zum Schnittstellen-
adapter, der jeder C-Control II Unit beiliegt.

Technische Daten (Quelle: CC-TOOLS):

Betriebsspannung: 10 bis 15V (Gleichspannung)
Versorgung der Messadapter: +/-5V max. 100mA
Max. Belastung des 5V-Schaltregler: 750mA
Max. Belastung des SR-Ports (0/5V): je 20mA
Max. Belastung des PWM-Ports (Open-Collector): 500mA
Max. Belastung des Relais (Wechsler): 8A/250V
Max. Speicherplatz des ext. RAM (gepuffert): 240Byte

3.1.2 C-Control II Unit (CONRAD ELECTRONIC)

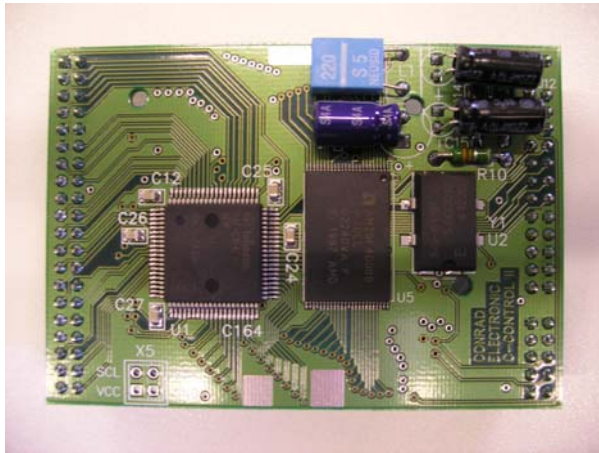


Abbildung 7 C-Control II Unit NV

Kern des Systems ist ein 16bit-Mikrocontroller des deutschen HiTech-Unternehmens Infineon-Technologies. Dieser Mikrocontroller übernimmt die wichtigen Steuerungsaufgaben der Boardelektronik. Der Benutzer programmiert das System und bestimmt, was wann und wie gesteuert werden soll. So kann man z.B. Sensormesswerte erfassen und festlegen welche Schaltsignale ausgegeben werden sollen, z.B. für Relais und Stellmotoren.

Über verschiedene digitale Schnittstellen kann die C-Control II mit anderen Komponenten Ihrer Anwendung kommunizieren, z.B. mit Sensoren, Aktoren und anderen

Steuerungssystemen. In Verbindung mit einer DCF77-Funkuhraktivantenne weiß C-Control II, was die Stunde geschlagen hat und kann präzise Schaltuhrfunktionen übernehmen. Durch das Hardwaredesign mit FLASH-Speicher kann das gesamte Betriebssystem jederzeit aktualisiert und erweitert werden. Erhebliche Vorteile bieten die speziell entwickelte Programmiersprache C2. Durch Multithreading lassen sich parallel ablaufende Ereignisse einfach koordinieren. Bei herkömmlicher Programmierung würde ein ganzes Programm blockiert werden, wenn der Steuercomputer z.B. auf einen Datenempfang über die serielle Schnittstelle wartet. Durch Multithreading kann das Warten im Hintergrund erfolgen, während der Steuercomputer andere Aktionen ausführt. C-Control II kann auch 32bit Integer- und 64bit Fließkommazahlen verarbeiten.

Technische Daten:

16 Bit-Mikrocontroller C164CI
extern 5MHz, intern 20 MHz Takt
512 kB FLASH-ROM
64 kB SRAM
16 freie Digitalports, davon einige mit Spezialfunktionen belegbar (2WB-Interface, Counter, Frequenzzähler, Drucker, 2. serielle Schnittstelle)
8 A/D-Konverterports mit 10 Bit-Auflösung
serielle Schnittstelle mit RTS/CTS-Handshake
hochwertiger RS232-Pegelwandler-IC mit besonderem Überspannungsschutz
3 PLM-Ausgabeports mit variabler Ausgabefrequenz für Digital-Analog-Wandlung und Ansteuerung von Servomotoren
CAN-Bus-Anschluss inklusive Bustreiber-IC
I²C-Bus
weiter Versorgungsspannungsbereich 8...24 V durch integriertes Schaltregelnetzteil
stabilisierte 5 V Spannung zur Versorgung externer ICs abgreifbar
Betriebssystem und Programmierung: ca. 64 kB Betriebssystem im FLASH-ROM
Virtuelle Maschine mit Multithreading (bis zu 254 parallele Threads),
Unterprogrammaufrufen,
Fließkomma-Arithmetik, Stringoperationen, komfortablen I/O-Routinen
max. 128kB Anwenderprogramm
max. 128kB Konstantenspeicher z.B. für Tabellen und Texte
192kB frei für Assembler Routinen (Assembler nicht im Lieferumfang enthalten)

3.2 Temperatursensoren

3.2.1 Übersicht

In der Heizungstechnik werden zur Temperaturmessung hauptsächlich analoge Widerstandssensoren verwendet (PTC / NTC). Oft kommen Pt1000 (auch Pt500) oder KTY (81-210) Sensoren zum Einsatz. Mit dem Reglerboard können auch AD592 Sensoren ausgewertet werden (temperaturabhängige Konstantstromquelle).

Gelegentlich werden auch Digitalsensoren für die Raumtemperaturerfassung verwendet (z.B. DS1631Z).

Es sind die Eigenschaften bezüglich des Einsatzzwecks zu beachten (z.B. Temperaturbeständigkeit).

Vorteile analoger Sensoren:

- hohe Genauigkeit und Temperaturbeständigkeit (Pt-Sensoren)
- Geringe Abmessungen und einfacher Anschluss (2-adriges Kabel, Widerstandssensoren sind verpolungssicher)

Nachteile analoger Sensoren:

- Aufbereitung der Messsignale durch Operationsverstärker-Schaltung erforderlich
- Störanfälligkeit
- Teuer (Pt-Sensoren)

Bemerkung: Temperaturdifferenzen werden im Folgenden in Kelvin K angegeben (eine Differenz von 1 K ist identisch einer Differenz von 1 °C).

Es werden hier folgende analoge Sensoren betrachtet, die mit den erhältlichen Messadaptern für das Reglerboard ausgewertet werden können:

Pt-Sensoren

Die Funktion des Pt-Temperatursensors basiert auf der temperaturabhängigen Änderung des elektrischen Platinwiderstands. So besitzt beispielsweise ein Pt100 bei 0 °C einen Widerstand von 100 Ω. Die Eigenschaften sind genormt nach IEC 751 bzw. DIN EN 60751.

Pt-Sensoren mit hohen Nennwiderständen (z.B. Pt1000) haben eine höhere Empfindlichkeit als solche mit niedrigem Nennwiderstand, da die Kennliniensteigung direkt proportional zu R_0 , dem Nominalwiderstand bei 0 °C ist.

Technische Daten des Pt1000 (IEC 751 / DIN EN 60751):

Messwiderstand	: 1000 Ω bei 0 °C
Messbereich	: -70 °C bis +600 °C (Toleranzklasse B, Dünnschicht)
Toleranz	: ± 0.3 °C bei 0 °C (Toleranzklasse B)

Bemerkung:

Die Sensoren sind in unterschiedlichen Toleranzklassen erhältlich. Pt1000 im Heizungsbau i.d.R. Toleranzklasse B ($\pm 0,3 + 0,005 \cdot |t|$, t in °C).

Wichtige Vorteile:

- hohe Präzision
- geringer Alterungsdrift (Langzeitstabilität)
- lange Lebensdauer
- nahezu lineare Kennlinie
- einfache Austauschbarkeit der Sensoren durch genormte Kennlinie und geringe (Widerstands-) Toleranzen
- hohe Temperaturwechselfestigkeit
- weiter Temperatureinsatzbereich

KTY-Sensoren

KTY-Sensoren sind Silizium-Temperaturensensoren. Sie zeichnen sich durch eng tolerierte Widerstände (bis $\pm 1\%$) mit guten, reproduzierbaren Temperaturbeiwerten aus. Ihre geringe Masse bewirkt eine sehr kurze Ansprechzeit ($\sim 2\text{ s}$). KTY besitzen eine Widerstands-Temperatur-Charakteristik mit nicht linearem Verlauf. Es muss daher i.d.R. eine Linearisierung vorgenommen werden. Im Vergleich zu anderen Sensoren, wie z.B. Pt-Sensoren sind sie kostengünstig in der Anschaffung.

Bemerkung:

KTY-Sensoren werden von PHILIPS und INFINEON hergestellt. Meist wird der KTY81-210 von PHILIPS verwendet.

Achtung: Oft werden Datenblätter des KTY10-6 von INFINEON angegeben (wird nicht mehr hergestellt). Die Kennlinie ist aber nicht exakt gleich mit der des KTY81-210!

Technische Daten des KTY81-210:

Messwiderstand	: 2000 Ω bei +25 °C
Messbereich	: -55 °C bis +150 °C
Toleranz	: $\pm 1,3\text{ K}$ bei 25 °C

AD592 (ANALOG DEVICES):

Der AD592 ist eine temperaturabhängige Konstantstromquelle. Dieser besitzt eine lineare Kennlinie und lässt sich leicht auswerten. Anders als bei Widerstandssensoren wird das Messergebnis nicht von dem Leitungswiderstand beeinflusst (Konstantstrom).

Technische Daten des AD592AN:

Messstrom	: 298,2 μA bei +25 °C (analog der Kelvinskala)
Steigung	: 1 μA pro K
Messbereich	: -25 °C bis +105 °C
Toleranz	: $\pm 1,5\text{ °C}$ bei 25 °C

Bemerkung:

Bei Testmessungen mit zwei AD592-Sensoren traten folgende Effekte auf. Die Sensoren wurden mit Schrumpfschlauch elektrisch isoliert und in siedendes Wasser getaucht. Dabei wurde gewartet bis sich ein stabiler Messwert eingestellt hatte. Dann wurde der Sensor kurz aus dem Wasser gezogen (Abkühlung bis auf ca. 50...60 °C) und wieder eingetaucht. Dieser Vorgang wurde wiederholt, bis sich der Endwert nicht mehr erhöhte.

Es dauerte jeweils mehrere Minuten bis sich ein stabiler Messwert eingestellt hatte. Dabei stieg die Temperatur bei einem Sensor über 3 Messungen von 96,9 auf 98,9 °C.

Die AD592 Sensoren scheinen daher für Temperaturmessungen in Heizsystemen nur bedingt geeignet zu sein.

Ausführungen

Die verschiedenen Sensoren sind in unterschiedlichen Ausführungen bzw. Bauformen erhältlich. Dabei sind Grundsätzlich Bauteile (ohne Kabel und Isolierung) und „fertige“ Sensoren (in einer Metallhülse mit eingepresstem Kabel) zu unterscheiden (AD592 ist nur als Bauteil erhältlich). Bauteile sind billiger aber auch schwieriger zu handhaben, da hier z.B. noch das Kabel angelötet und der Sensor ggf. elektrisch isoliert werden muss. Für den Außenbereich kommen i.d.R. nur „fertige“ Sensoren in Frage.

Kabelarten

- Kunststoff- / Ölflexkabel (PVC): -5...+80 °C
- Silikonkabel: -50...+180 (230) °C
- Teflonkabel (PTFE): -50...+230 (300) °C

Für den Kollektorfühler sollten Silikonkabel mit Teflon-Ummantelung der Adern verwendet werden (dauerhaltbar bis 230 °C). Vgl. [1].

3.2.2 Positionierung

Eine korrekte Temperaturmessung hängt entscheidend von der richtigen Positionierung und Anbringung ab.

Grundsätzlich können die Sensoren in Tauchhülsen (möglichst spielfreier Sitz) oder als Anlegefühler (Rohrschelle, Schlauchbinder; Korrosion und Hitzebeständigkeit beachten!) angebracht werden. Messungen in Tauchhülsen sind am genauesten und haben die kürzeste Reaktionszeit. Bei Speichern werden die Sensoren i.d.R. in Tauchhülsen und bei Rohren als Anlegefühler angebracht. Dabei sollte man Wärmeleitpaste verwenden und ggf. eine Zugentlastung vorsehen (vgl. auch [4]).

Sensoren sollten keiner Dauerfeuchte ausgesetzt werden, da eine gute Isolierung nicht immer gegeben ist (Gießharz ist nicht dauerhaft feuchtigkeitsfest). Bei Tauchhülsen können außerdem Schäden durch Frost auftreten.

Damit eine möglichst genaue Temperaturmessung mit Anlegefühlern an Rohrleitungen erreicht werden kann, sollten die Temperatursensoren wie in Abbildung 7 dargestellt angebracht werden. Wichtig ist die richtige Isolierung der Leitungen, hier sollte eine Dicke von 15 mm nicht unterschritten werden. Um die Genauigkeit der Messung weiter zu verbessern, sollte Wärmeleitpaste verwendet werden.

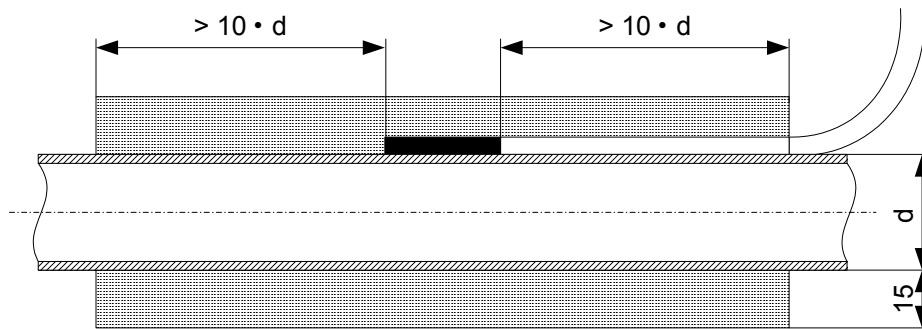


Abbildung 8 Anbringung von Anlegefühlern an Rohrleitungen

Untersuchungen der verschiedenen Einflüsse bei der Messung mit Anlegefühlern haben folgende Ergebnisse erbracht [1]:

Die Untersuchungen wurden unter folgenden Bedingungen durchgeführt: $T_{\text{Medium}} = 70\text{ °C}$, $T_{\text{Umgebung}} = 20\text{ °C}$.

Ergebnisse (Auswahl):

- | | |
|--|----------------------|
| 1. Sensor auf Rohr mit Farbanstrich, ohne Isolierung: | $T = 61,2\text{ °C}$ |
| 2. Sensor auf gesäubertem Rohr, mit Isolierung (15mm): | $T = 69,0\text{ °C}$ |
| 3. Sensor mit Wärmeleitpaste, mit Isolierung (15mm): | $T = 69,9\text{ °C}$ |

Der Kollektorfühler wird in der Vorlaufleitung des Kollektors an der wärmsten Stelle angebracht. Wegen der hohen Stillstandstemperaturen von bis zu über 200 °C sind nur Pt1000 geeignet.

Bei Pufferspeichern ohne Wärmetauscher (WT) werden die Sensoren möglichst weit unten bzw. oben angebracht.

Bei WW-Speichern mit Rippenrohr-WT knapp oberhalb des WT. Bei Glattrohr-WT im unteren Drittel bis Mitte des WT. Beim oberen WT (Nachheizung) im unteren Bereich des Bereitschaftsvolumens.

3.2.3 Messfehler

Bemerkung: Hardware bedingte Messfehler durch die Messadapter, das Reglerboard und die CC2 sollen nicht berücksichtigt werden. Es werden nur systematische (konstante) Messfehler betrachtet.

Widerstand der Anschlussleitung(en)

Dieser geht bei Widerstandssensoren (z.B. Pt und KTY) voll als Messfehler in die Messung ein (es wird eine, dem Leitungswiderstand entsprechend, zu hohe Temperatur gemessen; bei PTC).

Der Leitungswiderstand wird wie folgt berechnet:

$$R_{\text{Ader}} = \rho \cdot l / A$$

$$R_{\text{Kabel}} = 2 \cdot R_{\text{Ader}}$$

- ρ - spezifischer Widerstand [$\Omega \cdot \text{mm}^2/\text{m}$]; Kupfer bei 20 °C = 0,01786 $\Omega \cdot \text{mm}^2/\text{m}$
 l - einfache Leitungslänge (eine Ader) [m]
 A - Leitungsquerschnitt [mm^2]

z.B.: 50 m Kabel mit einem Querschnitt von 0,5 mm^2 = 3,6 Ω

Beim Pt1000 bedeutet dies einen Messfehler von ca. 1 K. Zum Vergleich: bei einem Pt100 wäre der Messfehler fast 10 K (!). Aufgrund der höheren Widerstandswerte des KTY (81-210) ergibt sich ein Messfehler von ca. 0,2 K.

Wird ein Leitungsquerschnitt von 1,0 mm^2 verwendet (1,8 Ω), liegt der Messfehler beim Pt1000 entsprechend bei ca. 0,5 K.

Die Messung mit AD592 Sensoren wird vom Leitungswiderstand nicht beeinflusst, da dieser eine temperaturabhängige Konstantstromquelle ist.

Der Messfehler durch den Leitungswiderstand kann im Programm durch einen Temperatur-Offset korrigiert werden („individueller Offset“).

Empfohlene Anschluss- bzw. Verlängerungskabel (Sensor-Grundwiderstand beachten, s.o.):

- bis 20 m: 2 x 0,5...0,75 mm^2
- bis 50 m: 2 x 0,75...1,0 mm^2
- bis 80 (100) m: 2 x 1,0...1,5 mm^2 (evtl. Glättung durch RC-Glied erf.)

Kennlinie

Jeder Sensortyp besitzt eine charakteristische Kennlinie die i.d.R. nicht linear ist.

Toleranzen

Individuelle Abweichungen (Toleranzen) jedes einzelnen Sensors von den vom Hersteller angegebenen Werten:

- Offset
Konstante Abweichung über den gesamten Messbereich (Parallelverschiebung der Kennlinie). z.B.: Es wird generell eine um 2 K zu hohe Temperatur gemessen.
- Steigungsfehler
Die Steigung ist steiler oder flacher als vom Hersteller angegeben. z.B.: Die Widerstandszunahme bei steigender Temperatur ist größer als angegeben.
- Linearitätsfehler
Abweichung von der vom Hersteller angegebenen (Nicht-) Linearität.

Fehlerkorrektur

Da der Abgleich der Messadapter immer für alle 4 Sensoren an diesem erfolgt, können die Fehlerkorrekturen nur im Programm erfolgen. Mit vertretbarem Aufwand können nur die Nichtlinearität (der charakteristischen Kennlinie) und der individuelle Offset korrigiert werden (was aber ausreichend sein sollte).

Da im Programm möglichst nicht mit Fließkommazahlen gerechnet werden sollte (siehe Kap. 7.2) wird nicht mit einer Polynomfunktion linearisiert. Stattdessen wird der Messbereich unterteilt (gesplittet) und Teillinearisationen vorgenommen (siehe Kap. 4.3.1). Die Genauigkeit wird über die Anzahl bzw. Größe der Teilbereiche (Temperaturbereiche) beeinflusst. Aufgrund der starken Nichtlinearität des KTY sind hier bei gleichem Messbereich mehr Unterteilungen als beim Pt1000 erforderlich. Beim AD592 ist keine Linearisierung erforderlich, da dieser über seinen Messbereich sehr linear ist.

Beispiel:

Ohne Teillinearisation hätte ein Pt1000 bei einem Messbereich von $-30 \dots 130 \text{ }^{\circ}\text{C}$ in der Mitte ($50 \text{ }^{\circ}\text{C}$) eine Abweichung von ca. 1 K (Kelvin) von der charakteristischen Kennlinie. Mit Teillinearisation zwischen $50 \dots 100 \text{ }^{\circ}\text{C}$ liegt die Abweichung bei $75 \text{ }^{\circ}\text{C}$ bei ca. 0,1 K.

Zum Vergleich: der KTY (81-210) hätte im Bereich $-30 \dots 130 \text{ }^{\circ}\text{C}$ bei $50 \text{ }^{\circ}\text{C}$ eine Abweichung von ca. 12 K (!). Bei Teillinearisation zwischen $60 \dots 80 \text{ }^{\circ}\text{C}$ bei $70 \text{ }^{\circ}\text{C}$ ca. 0,2 K.

Eine Möglichkeit für die Ermittlung des individuellen Offsets wäre die Messung der Temperatur in Eiswasser ($0 \text{ }^{\circ}\text{C}$). Nachteilig ist, dass der Sensor dazu elektrisch isoliert sein muss, was nicht immer der Fall ist. Außerdem kann der Sensor noch einen erheblichen Steigungsfehler aufweisen. Da es bei der Heizungsregelung vor allem darauf ankommt, dass die Sensoren in einem bestimmten Temperaturbereich (Regelungsbereich) den gleichen Temperaturwert erzeugen, sollte ein relativer Offsetabgleich der Sensoren untereinander durchgeführt werden. Dazu werden alle Sensoren auf eine Temperatur gebracht, die etwa in der Mitte des Regelungs-Temperaturbereichs (nicht des Gesamtmessbereichs) liegt. Die Sensoren können dazu z.B. gemeinsam auf ein Heizungsrohr mit der entsprechenden Temperatur gespannt werden (Wärmedämmung vorsehen) oder in den Düsenaustritt eines Föns (Querströmung der Umgebungsluft verhindern). Es sollte eine ausreichende Zeit gewartet werden, bis die Sensoren einen stabilen Wert erzeugen. Aus den angezeigten Werten wird dann der Mittelwert gebildet, auf den die Sensoren dann im Programm abgeglichen werden („individueller Offset“). Es ist ggf. der durch den Leitungswiderstand bedingte Offset zu addieren.

Bemerkung:

Der Korrekturaufwand sollte im Verhältnis zur Messgenauigkeit (Toleranz) des verwendeten Sensors und den Anforderungen der Anwendung stehen. Bei Messungen mit hohem Anspruch an die Messgenauigkeit sollten generell Pt1000 Sensoren verwendet werden. Aufgrund der hohen Qualität bzw. Genauigkeit ist bei diesen nur eine Linearisierung erforderlich (evtl. Offset-Korrektur der Leitungswiderstände). Bei kleinem Messbereich bzw. geringen Anforderungen können diese auch entfallen.

3.3 Stückliste

Bezeichnung	Preis je Bausatz ² [€]	Preis je Baustein ³ [€]
CC2-ReglerBoard Bundle ¹	206,50	240,50
Schaltnetzteil 12V / 2,5 A für ReglerBoard	-/-	13,80
LC-Display 4x20 + Flachbandkabel	-/-	31,50
Piezo-Schallwandler („Beeper“)	-/-	1,50
AD592-Meßadapter ⁴	7,00	12,00
AD592-Diodenarray ⁴	7,00	11,00
AD-Pull-Up-Widerstände für AD592 ⁴	4,50	8,00
KTY10-Meßadapter ⁴	7,00	12,00
AD-Pull-Up-Widerstände für KTY10 ⁴	4,20	7,20
Pt1000-Meßadapter ⁴	7,00	13,00
Konstant-Strom-Multiplexer für Pt1000 ⁴	9,80	14,50

¹ Das CC2-ReglerBoard-Bundle besteht aus dem CC2-ReglerBoard mit Option GES (Echtzeituhr gepuffert über Goldcap) sowie der C-Control II Unit und Bopla-Gehäuse. Es sollte die Variante mit Goldcap (GES) statt der Batterie bevorzugt werden (wird automatisch geladen). Nachteil: hält für nur ca. 10 Tage. Das sollte aber ausreichend sein.

² Bausatz: Die mitgelieferten Bauteile müssen noch mit der Platine verlötet werden.

³ Baustein: Das Modul wird als fertiger Baustein geliefert

⁴ Messadapter für jeweils 4 Sensoren (max. 4 Adapter für 16 Sensoren). Von den Zusatzplatinen wird nur jeweils 1 Stk. für 16 Sensoren benötigt. Es besteht die Möglichkeit, zwei verschiedene Sensortypen gleichzeitig zu verwenden (z.B. Pt + KTY, 1 Sensortyp je Messadapter). Dazu werden teilbestückte Zusatzplatinen benötigt (bei Bestellung angeben).

Beispiel-Stückliste für eine Regelungseinheit mit 4 Pt1000-Sensoren

Stück	Bezeichnung	Preis je Baustein [€]	Preis je Anzahl [€]
1	CC2-ReglerBoard Bundle	240,50	240,50
1	Schaltnetzteil 12V / 2,5 A + Kabel	13,80	13,80
1	LC-Display 4x20 + Flachbandkabel	31,50	31,50
1	Piezo-Schallwandler („Beeper“)	1,50	1,50
1	Pt1000-Meßadapter	13,00	13,00
1	Pt1000-Konstant-Strom-Multiplexer	14,50	14,50
4	Pt1000-Sensor (3 m Silikonkabel)	20,00	80,00
		Summe	394,80

3.4 Bestückung des Reglerboards

Bevor eine Temperaturmessung durchgeführt werden kann, muss das Board mit den für den eingesetzten Sensortyp entsprechenden Messkomponenten bestückt werden.

3.4.1 Messkomponenten für KTY-Sensoren

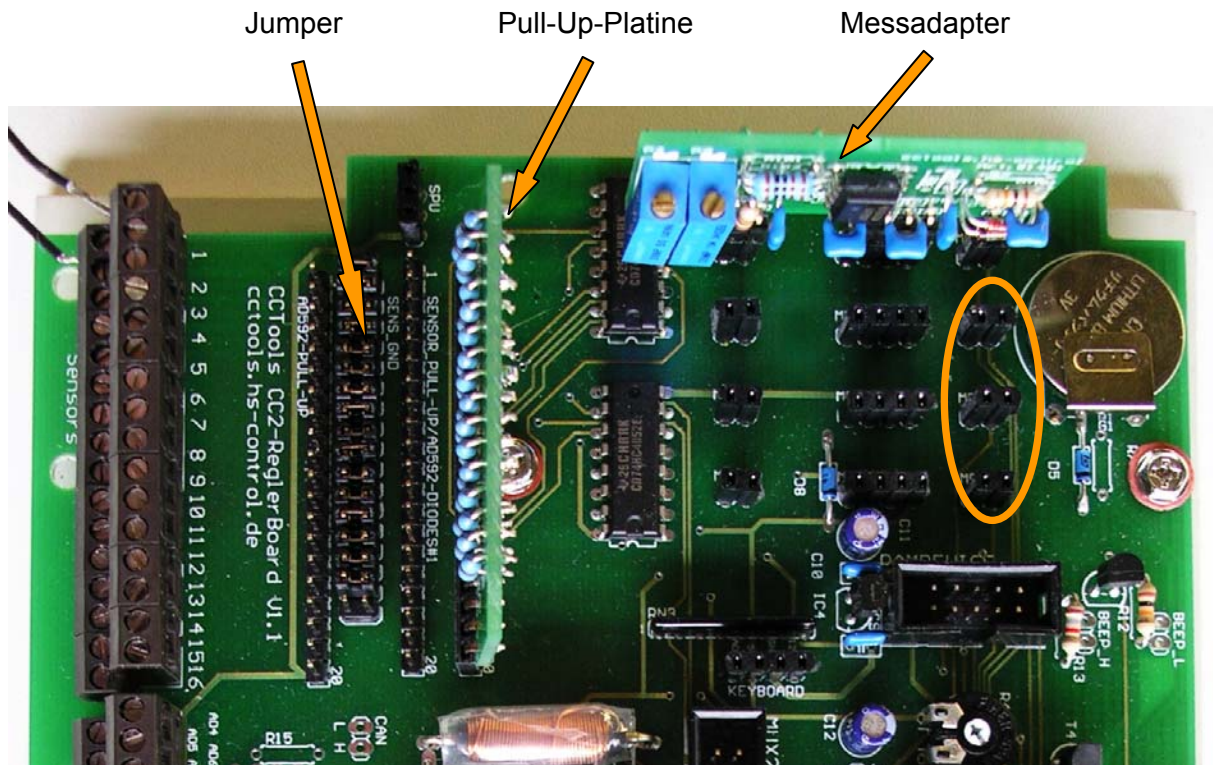


Abbildung 9 Reglerboard mit KTY-Messkomponenten

Zur Temperaturmessung mit KTY-Sensoren werden folgende Komponenten benötigt:

- KTY-Messadapter (Art.-Nr. 1401)
- AD-Pull-Up-Platine für KTY (Art.-Nr. 1417-xK)

Bemerkung:

Die Stiftsockel (SENS_GND) müssen mit 16 Jumpers bestückt werden, der Sockel ganz rechts (17) bleibt unbestückt.

Die Ports der nicht bestückten Messadaptersteckplätze MP2B, MP3B, MP4B zeigen einen undefinierten Wert. Die 2-poligen Buchsen dieser Steckplätze sollten daher z.B. mit einem 2-beinigen Stiftsockel mit aufgesteckten Jumpers überbrückt werden (Ports auf Masse).

Der AD-Wert ist dann „Null“. Als Temperatur wird -99,9 °C (Kurzschluss) angezeigt.

3.4.2 Messkomponenten für Pt-Sensoren

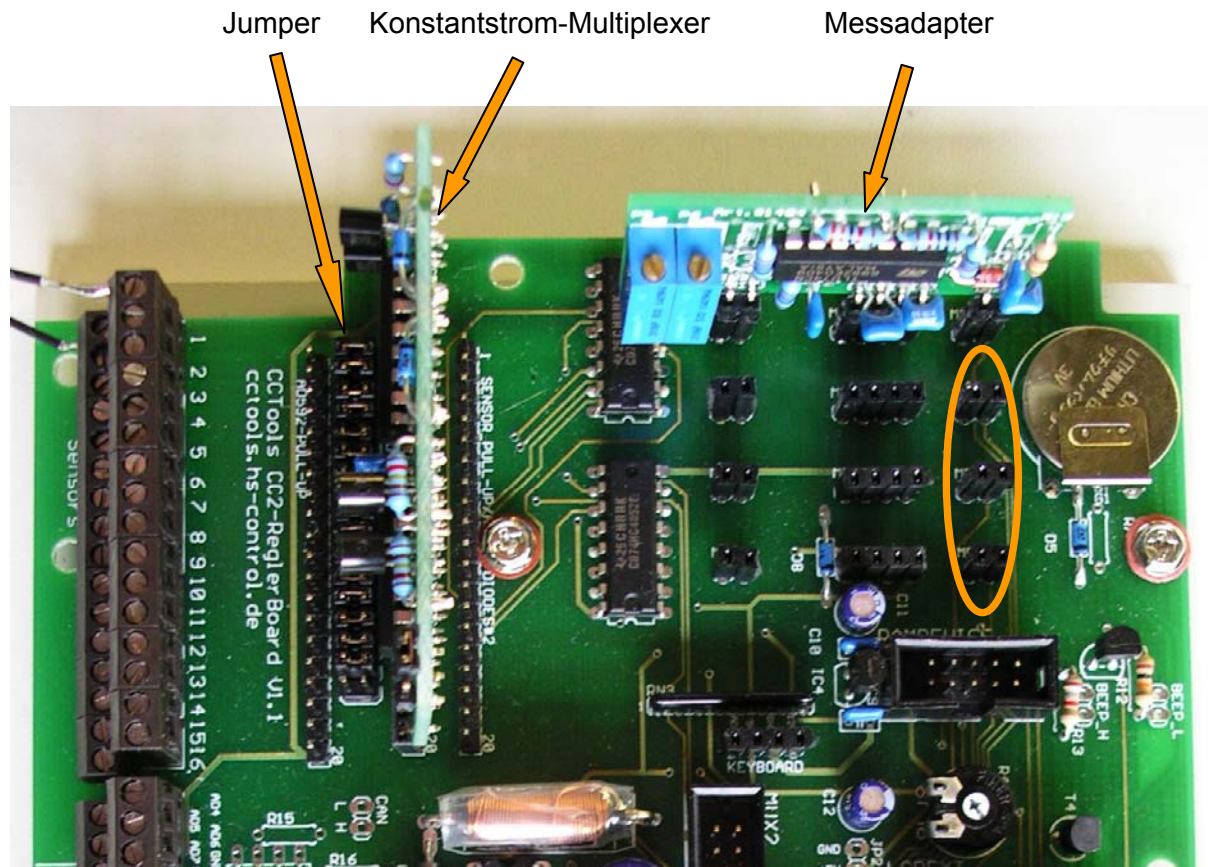


Abbildung 10 Reglerboard mit PT-Messkomponenten

Zur Temperaturmessung mit Pt-Sensoren sind folgende Komponenten nötig:

- PT-Messadapter (Art.-Nr. 1404)
- Konstant-Strom-Multiplexer für PT (Art.-Nr. 1418)

Bemerkung:

Die Stiftsockel (SENS_GND) müssen mit 16 Jumpern bestückt werden, der Sockel ganz rechts (17) bleibt unbestückt. Mit einem Messadapter lassen sich bis zu vier Sensoren betreiben.

Die Ports der nicht bestückten Messadaptersteckplätze MP2B, MP3B, MP4B zeigen einen undefinierten Wert. Die 2-poligen Buchsen dieser Steckplätze sollten daher z.B. mit einem 2-beinigen Stiftsockel mit aufgesteckten Jumpern überbrückt werden (Ports auf Masse). Der AD-Wert ist dann „Null“. Als Temperatur wird -99,9 °C (Kurzschluss) angezeigt.

Am Konstant-Strom-Multiplexer können 2 Spannungen gewählt werden (über Jumper):

- 5V : für Widerstände bis ca. 2,1 k Ω (bis ca. 280 °C bei Pt1000)
- 12V : für Widerstände bis ca. 3,5 k Ω (bis ca. 700 °C bei Pt1000)

3.4.3 Messkomponenten für AD592-Sensoren

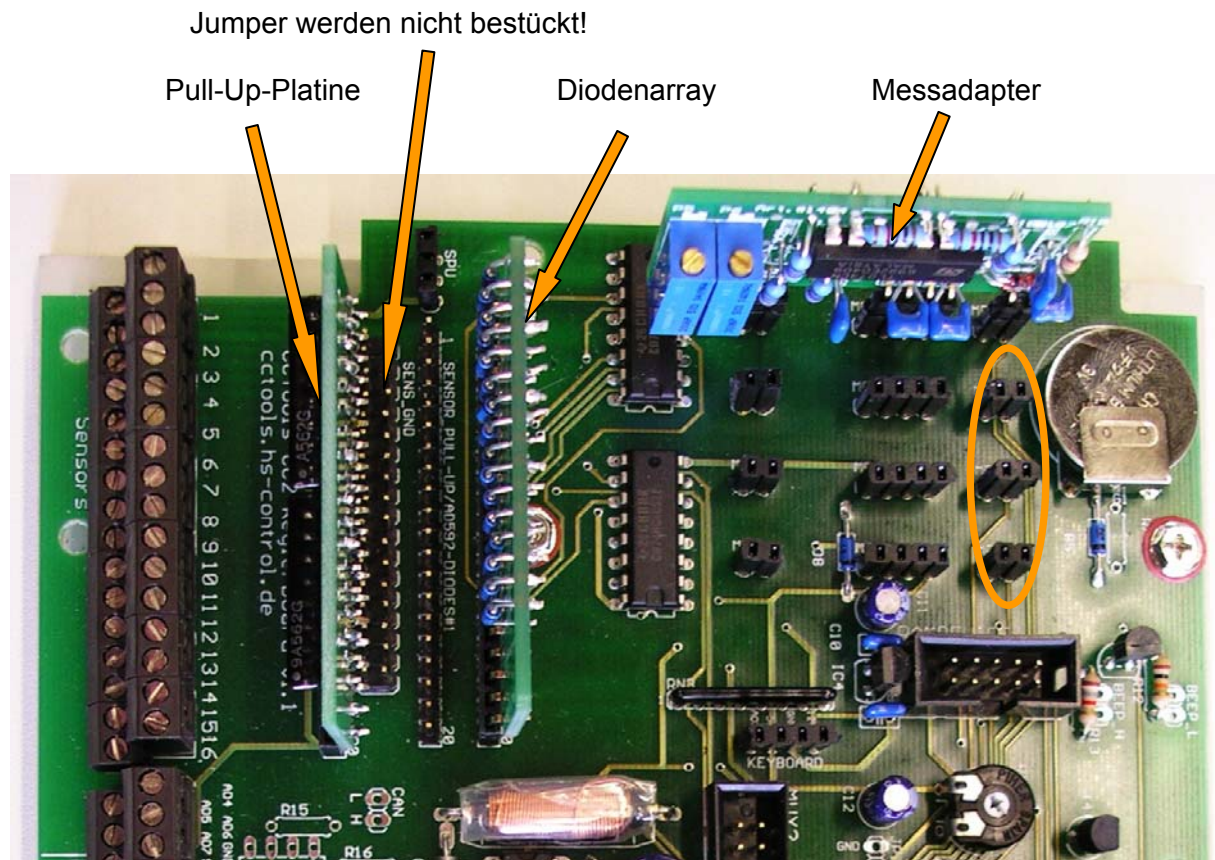


Abbildung 11 Reglerboard mit AD592-Messkomponenten

Zur Temperaturmessung mit AD592-Sensoren werden folgende Komponenten benötigt:

- AD592-Messadapter (Art.-Nr. 1404-x)
- AD592-Diodenarray (Art.-Nr. 1419)
- AD-Pull-Up-Platine für AD592 (Art.-Nr. 1417-xA)

Bemerkung:

Die Stiftsockel (SENS_GND) werden nicht bestückt. Mit einem Messadapter lassen sich bis zu vier Sensoren betreiben.

Die Ports der nicht bestückten Messadaptersteckplätze MP2B, MP3B, MP4B zeigen einen undefinierten Wert. Die 2-poligen Buchsen dieser Steckplätze sollten daher z.B. mit einem 2-beinigen Stiftsockel mit aufgesteckten Jumpern überbrückt werden (Ports auf Masse). Der AD-Wert ist dann „Null“. Als Temperatur wird 999,9 °C (Bruch) angezeigt.

3.5 Anschluss weiterer Komponenten an das Reglerboard

Weitere Komponenten wie LC-Display, Sensoren, Pumpen, Ventile usw. werden wie in der Dokumentation des Reglerboards beschrieben angeschlossen.

Hinweis zur Dokumentation des Reglerboards

In der vorliegenden Dokumentation waren 2 Fehler:

1. Die Reihenfolge der Relais-LED-Anschlüsse ist vertauscht: ganz rechts ist GND, links daneben 1, dann 2 und 3.
2. Beim Anschlussbeispiel des Heizungsmischer-Motors muss die Leitung für Mischer ZU am Relais 2 angeschlossen sein, bzw. an dem Relais mit dem 230V-Leiter an Common (wenn der Mischer ZU fahren soll wenn beide Relais EIN sind).

Bemerkungen:

- Kabel mit Pfostensteckern: rote Markierung am Kabel an Pin 1.
- LCD: das Flachbandkabel muss 1 zu 1 an die Löt pads des Displays angelötet werden (kann auch verlötet bestellt werden).

Freie Ports

Die nicht belegten Ports des Reglerboards können frei verwendet werden. So können die Digitalports z.B. zur optischen Ausgabe von Zuständen (z.B. Blinken einer LED bei einem Fehler) oder als Eingang zur Erkennung von Signalen verwendet werden.

LED's werden in Reihe mit einem geeigneten Vorwiderstand an Masse (GND) und einen Digitalport (P1H.0 bis P1H.4) angeschlossen. Der Widerstand muss so gewählt werden, dass der max. Strom von 5 mA pro Port nicht überschritten wird (5V Versorgungsspannung). Hier passen i.d.R. 680 Ω .

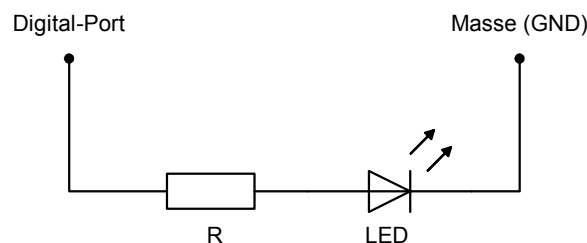


Abbildung 12 Anschluss von LED's

3.6 Abgleich der Messadapter

Vor der ersten Temperaturmessung müssen die Messadapter auf den gewünschten Messbereich abgeglichen werden um den Eingangsbereich bzw. die Auflösung des Analog/Digital-Wandlers voll auszunutzen. Der Messbereich sollte nicht größer als die zu erwartenden Temperaturen gewählt werden (,+’ bzw. ,-' ca. 10...20 °C).

Dazu wird das Programmtool „rbkalib.c2p“ verwendet (C:\Programme\C-Control II\Samples\ReglerBoard'). Das Programm muss in die CC2 geladen werden (siehe Kap. 4). Die Datenausgabe erfolgt über ein Terminal-Programm (z.B. HyperTerminal; 57.600 Baud).

Der Onboard-AD-Multiplexer wird abgeschaltet und die Messadapter auf die ersten 4 Sensorklemmen geschaltet (Klemme 1...4 = Messadapter A...D). Nun kann der Abgleich vorgenommen werden. Dazu wird ein Widerstand angelegt, der der kleinsten zu erfassenden Temperatur entspricht (je nach Sensortyp), z.B. mit Hilfe eines Trimpotentiometers (für Pt- und KTY81-210-Sensoren wurde eine Excel-Tabelle erstellt, Ordner: Datenblätter). Zuerst wird der Offset auf ,0' abgeglichen, so dass die Anzeige etwas über 0 schwankt (zwischen 0 und 5). Als nächstes wird ein Widerstand angelegt, der der maximalen Temperatur entspricht. Dann wird die Steigung verstellt, bis ein Wert zwischen 10225 und 10230 angezeigt wird. Dieses Verfahren muss für jeden Messadapter durchgeführt werden.

Bemerkung: Bei der Widerstandsvorgabe an den Sensorklemmen sind die Messtoleranzen des Messgeräts zu beachten! Bei Verwendung üblicher Multimeter können sich z.B. beim Pt1000 Abweichungen von ca. $\pm 1...3$ K (Kelvin) ergeben! Es kann daher sinnvoll sein, passende Präzisionswiderstände (0,1 % Toleranz) zu beschaffen.

Bemerkung zum KTY-Messadapter:

Der Abgleichen des KTY-Messadapters ist etwas aufwendig, da sich Offset und Steigung gegenseitig beeinflussen. Das hat zur Folge, dass der Vorgang so oft wiederholt werden muss, bis der gewünschte Messbereich erreicht worden ist.

Bemerkung zum AD592-Messadapter:

Mit Hilfe eines geeigneten Trimpotentiometers (z.B. 100 k Ω) und eines Strommessgerätes, lässt sich der Stromdurchfluss in μ A entsprechend der unteren und oberen Temperaturgrenze in K (Kelvin) simulieren.

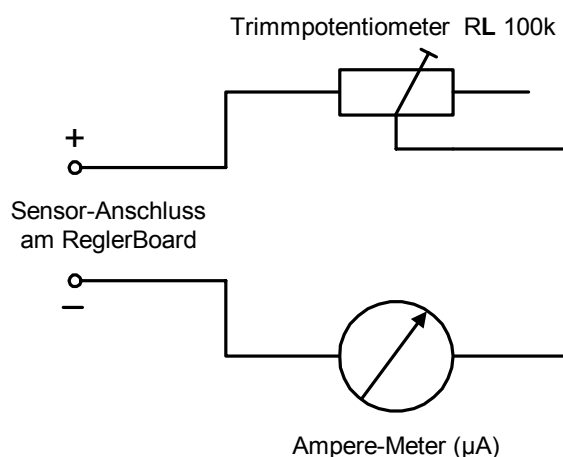


Abbildung 13 Abgleich AD592-Messadapter

4 Programmierung

Die Programmierung der C-Control II erfolgt direkt im Quelltext in der eigens für die CC2 entwickelten Sprache „C2“. C2 ist an die höhere Programmiersprache C angelehnt. Durch den vereinfachten Syntax ist C2 jedoch wesentlich leichter zu erlernen.

Nach der offiziellen Beschreibung von CONRAD ELECTRONIC ist die Programmiersprache C2 nur bedingt für den Einstieg in die Programmierung von Mikrocomputern geeignet. Für die Anpassung der hier vorgestellten Programme sind jedoch keine besonderen Programmierkenntnisse erforderlich (siehe auch Kap. 4.4). Sollen die Programme erweitert oder eigene Programme geschrieben werden, ist eine intensive Einarbeitung aber unumgänglich.

Da das Handbuch zur CC2 „nur“ eine technische Referenz und keine Anleitung ist, fällt der Einstieg etwas schwer.

Zunächst ist es erforderlich sich mit den Grundlagen der C2-Sprache und der Bedienung der Entwicklungsumgebung (IDE) vertraut zu machen. Die Bedienung der IDE ist in der Online-Hilfe beschrieben. Es wird dringend empfohlen die CC2Net-FAQs, „Tips und Ergänzungen“ und mindestens die folgenden Kapitel im CC2-Handbuch aufmerksam zu lesen: 4.4.3 / 5.1 – 5.5.5 / 5.7.1 – 5.9.5 / 5.10.1 und 5.10.5 / (6).

Für weitere Informationen zur Programmierung in C2 siehe Anhang (Kap. 7.2) und [2]. Allgemeine Programmbeispiele befinden sich im Programm-Ordner der IDE: ‚C:\Programme\C-Control II\Samples\‘.

Hinweise zum Handbuch der C-Control II

Das Handbuch bezieht sich auf die nicht mehr erhältliche CC2-Version im vergossenen Kunststoffgehäuse mit internem Display. Der hier verwendeten „CC2-Unit NV“ fehlt lediglich das Gehäuse mit internem Display.

Achtung:

Die Beschreibungen der Module (Kapitel 7 im Handbuch) beziehen sich auf die Original-Module. Bei den vom CC2Net modifizierten Modulen sind einige Befehle geändert / vereinfacht worden. Daher sollten die Hinweise zu den neuen Modulen beachtet werden (Hilfe in der IDE und im Quelltext der Module).

Anders als im Kapitel 3.2.3 des Handbuchs beschrieben, beträgt der Messbereich an den Analogeingängen der CC2 ‚0...4,092 V‘ (4,096 V Spannungsreferenz). Durch die 10bit (1024) Auflösung des A/D-Wandlers wird der Messbereich in 1023 Teile geteilt (4 mV pro Bit bzw. Digit).

4.1 Vorraussetzungen

Die Software zur C-Control II wird von CONRAD schon seit längerer Zeit nicht mehr weiterentwickelt. Die Weiterentwicklung des Betriebssystems und der Funktionsmodule wurde von „CC2Net.de“ fortgeführt. Einige Module wurden grundlegend geändert, bei anderen wurden die Zugriffe bzw. Funktionen vereinfacht und Fehler beseitigt. Es sollten daher keine Dateien von der Original-CD verwendet werden!

Es muss das Betriebssystem (OS) OSOPT V3.0 oder neuer und der Systemtreiber ‚sys0002.hex‘ (in Segment 3) in die CC2-Unit geladen werden. Dazu wird das Download-Tool von CC2Net benötigt. Bei Bezug der CC2-Unit über CC-TOOLS ist das neueste Betriebssystem bereits vorinstalliert (‚sys0002.hex‘?).

Es werden die aktuellen vom CC2Net modifizierten Module benötigt. Es sollte daher die neueste Entwicklungsumgebung (IDE SP2.13 oder neuer) von CC2Net installiert werden.

Wichtig:

Das Modul „pcf8583.c2“ (Ansteuerung des Echtzeit-Uhrenbausteins auf dem Reglerboard) der aktuellen IDE (SP2-13) enthält einen Fehler, der den Programmablauf der hier vorgestellten Programme stoppt. Der Fehler im Modul wurde von den Autoren dieser Arbeit korrigiert (inoffiziell). Die vorhandene Version muss durch die korrigierte Version ersetzt werden (siehe Ordner „Weitere Module“)!

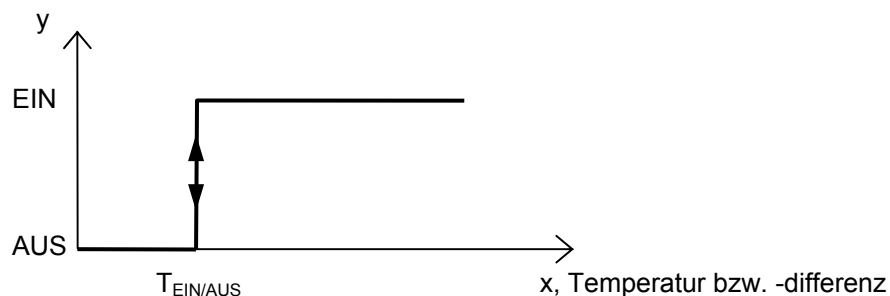
4.2 Regelungsgrundlagen

In der Heizungstechnik werden meist Zweipunktregler bzw. -verhalten eingesetzt (das Verhalten wird im Programm erzeugt). Ein Zweipunktregler ist ein unstetiger Regler, d.h. es gibt nur 2 Schaltstellungen (EIN / AUS).

z.B.:

WENN	$T > T_{\text{EIN/AUS}}$	DANN „Pumpe“ EIN
WENN	$T < T_{\text{EIN/AUS}}$	DANN „Pumpe“ AUS

Zweipunktregler:



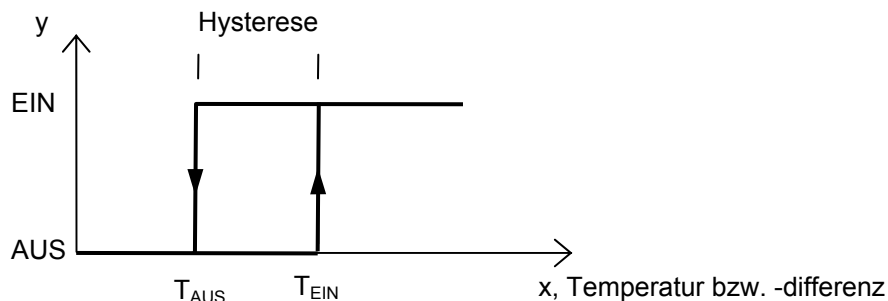
Da bei diesem Verhalten nur ein Schalterpunkt existiert, würde das System innerhalb seiner Trägheit immer um den Schalterpunkt schwanken und die Pumpe würde in sehr kurzen Abständen EIN / AUS geschaltet (ein weiteres Problem wären die nicht ganz stabilen Messsignale der Sensoren). Deshalb wird eine Hysterese benötigt.

z.B.:

WENN $T \geq T_{\text{EIN}}$ DANN „Pumpe“ EIN
 WENN $T \leq T_{\text{AUS}}$ DANN „Pumpe“ AUS

$$T_{\text{EIN}} = T_{\text{AUS}} + \text{Hysterese}$$

Zweipunktregler mit Hysterese:



Thermostatregelung

Es soll z.B. eine Solltemperatur (ungefähr) bzw. eine Mindesttemperatur gehalten werden. Es wird bei einer festgelegten Temperatur eingeschaltet und bei einer um eine Hysterese höheren Temperatur ausgeschaltet.

Für eine Überschusswärmenutzung wäre das Schaltverhalten umgekehrt.

z.B. Nachheizung des WW-Speichers:

EIN bei $T_{\text{Speicher}} \leq 40^\circ\text{C}$
 AUS bei $T_{\text{Speicher}} \geq 45^\circ\text{C}$

Differenztemperaturregelung

Es soll bei einer nutzbaren Temperaturdifferenz geschaltet werden. Es müssen die Wärmeverluste in den Rohrleitungen und Einbauten und ggf. das notwendige Temperaturgefälle des Wärmetauschers berücksichtigt werden, damit die Pumpe nur dann läuft, wenn auch Wärme übertragen werden kann.

z.B. Solarpumpe:

EIN bei $(T_{\text{Kollektor}} - T_{\text{Speicher}}) \geq \Delta T_{\text{EIN}}$
 AUS bei $(T_{\text{Kollektor}} - T_{\text{Speicher}}) \leq \Delta T_{\text{AUS}}$

Ausschaltemperaturdifferenz: $\Delta T_{\text{AUS}} = \text{Verluste}$
Einschaltemperaturdifferenz: $\Delta T_{\text{EIN}} = \Delta T_{\text{AUS}} + \text{Hysterese}$

Bemerkung: Temperaturdifferenzen werden in Kelvin K angegeben (eine Differenz von 1 K ist identisch einer Differenz von 1 °C).

Dynamische Differenz:

Es kann auch eine dynamische Differenz verwendet werden um die höheren Wärmeverluste bei steigender Temperatur zu berücksichtigen. Dabei wird die Differenz pro 64 K ab 0 °C (vereinfacht) definiert.

Die Berechnung in „C2-Form“ (vgl. Kap. 7.2) wäre dann z.B.:

$$\Delta T_{\text{AUS,dyn}} = (T \cdot \Delta T_{\text{statisch}}) / 640 \quad (\text{Temperaturen in } 1/10 \text{ °C bzw. K, z.B. } 8 \text{ K} = 80)$$

Achtung: Die Berechnung oder Parameterübergabe (wenn als ‚function‘ definiert) muss im Relais-Thread innerhalb der „loop“-Schleife erfolgen (wie die Sensorzuweisungen), damit der Wert bei jedem Durchlauf neu ermittelt wird! Klammern nicht entfernen!

4.3 Module der Regelung

Da nur die System-Module für die CC2 vorhanden waren, mussten die Module zur Regelung selbst programmiert werden. Die Programmierten Module werden im Folgenden beschrieben.

Es soll hier noch mal angemerkt werden, dass die Autoren keine Programmierexperten sind. Die erstellten Programme bzw. Module ließen sich noch optimieren (z.B. Speicherverbrauch, Geschwindigkeit, usw.) und einige Funktionen „eleganter“ lösen. Es wurde jedoch auch darauf geachtet, dass der Programmcode auch für Einsteiger möglichst leicht nachzuvollziehen ist.

Die Regelung wurde nur im Labor mit simulierten Temperaturen und nicht an einer bestehenden Heizungsanlage getestet. Die Regelungsfunktionen wurden zwar ausgiebig getestet (manuell), dennoch können Fehler nicht ausgeschlossen werden. Generell sollten die Regelungsfunktionen vor der Verwendung getestet werden. Beim Testen der Funktionen müssen ggf. Zeitverzögerungen im Programm beachtet werden (z.B. Mindesteinschaltzeit eines Relais).

Hinweise

Die Module bzw. das Programm müssen im Quelltext vor der ersten Verwendung an die konkrete Aufgabe angepasst und dann in die CC2 geladen werden (siehe auch Kap. 4.4).

Da Messfehler durch die Messadapter entstehen können (z.B. ungenauer Abgleich) sollten miteinander zu vergleichende Sensoren (Temperaturen) möglichst an einem Messadapter

angeschlossen sein. Dadurch betrifft ein Messfehler dann alle Sensoren an diesem gleichermaßen.

Alle Temperaturen werden i.d.R. in 1/10 °C bzw. 1/10 K angegeben (z.B. 36,8 °C = 368)

Es werden im gesamten Programm keine Fließkommazahlen ‚float‘ verwendet (keine Nachkommastellen!).

Achtung: Werden die Sensor- / Relaiszuordnungen im Modul „control“ geändert, müssen auch die Zuweisungen in den Modulen „watch“ und „terminal“ entsprechend geändert werden (Siehe Hinweise im Quelltext).

Für Details siehe Beschreibung in den Modul-Quelltexten.

4.3.1 Modul „temp“ (Temperatursensor-Auswertung)

Dieses Modul übernimmt das Auslesen der 16 Analog-Ports für die Temperaturmessung über Messadapter und Multiplexer. Die vom A/D-Wandler übermittelten Digitalwerte werden dann über Auswertformeln in Temperaturen umgerechnet. Das Ansteuerungsmodul „rbports“ für das Reglerboard nimmt dabei eine Interpolation vor und übergibt Werte von 0 bis 10230 für den Messbereich der Analog-Ports von 0 bis 4,092 V.

Im oberen Teil des „temp“-Moduls kann ggf. der individuelle Offset für jeden der 16 Sensoren eingetragen werden.

Danach folgen die Auswertfunktionen, die ggf. angepasst werden müssen! Hier gibt es 2 Formen, ohne und mit Linearisierung (vgl. Kap. 3.2.3):

Auswertfunktion ohne Linearisierung:

$$T = \frac{\text{aktuellerADWert} \cdot \text{Temperaturmessbereich}}{1023} \pm \text{Offset}$$

- Temperaturmessbereich in Kelvin [K], entsprechend der eingestellten Temperaturgrenzen am Messadapter (Kap. 3.6), z.B. -20 bis +115°C = 135 K
- Offset in 1/10 K, z.B. untere Temperaturgrenze: -20 °C, Offset = -200
- Rückgabewert T (Temperatur) in 1/10 °C-Schritten, z.B. 638 = 63,8 °C

Auswertfunktion für Teillinearisation, z.B. für KTY, Pt1000:

Es wird innerhalb der gewählten Temperaturteilmbereiche interpoliert.

$$T = \frac{(\text{aktuellerADWert} - \text{ADTeilbereichWert}) \cdot \text{Temperaturteilmbereich}}{\text{AD - Wertebereich}} \pm \text{TeilOffset}$$

- Teilmessbereich in Kelvin [K], z.B. +60 bis +80 °C = 20 K
- AD-Wertebereich: (oberer ADWert - unterer ADWert) /10, (ohne Nachkommastellen!)
- Teil-Offset: die untere Temperatur des Teilbereichs in 1/10 K, z.B. Teilbereich = 100...130 °C, Teil-Offset = 1000
- Aktueller-AD-Wert: der sich aus der laufenden Temperaturmessung ergebene digitalisierte Messwert.
- AD-Teilbereich-Wert: der AD-Wert der unteren Temperatur des Teilbereichs.
- Rückgabewert T (Temperatur) in 1/10 °C-Schritten, z.B. 638 = 63,8 °C

Die Auswertformeln werden von der oberen Temperaturgrenze (AD-Wert = 10230) nach unten abgearbeitet. Siehe Beispiele im Quelltext des Moduls.

Vorgehensweise:

1. Aufteilen des festgelegten Temperaturmessbereichs (Hardware-Abgleich) in Teilbereiche, abhängig vom Messbereich und Kennlinie, sowie der gewünschten Genauigkeit. Je mehr Teilbereiche, desto genauer wird die Messung.
2. Die AD-Werte der Teilbereichsgrenzen werden durch simulierte Temperaturen mit ‚rbkalib‘ ermittelt (siehe Hardware-Abgleich).
3. Eintragen der Werte in die Auswertformeln.
4. Kontrolle der Ausgegebenen Temperaturen mit simulierten Temperaturen (Trimmpotentiometer) in jedem (!) Temperaturteilbereich!

Für den KTY81-210 könnte folgende Aufteilung sinnvoll sein (Messbereich: -40 bis +130 °C):

-40 , -20 , 0 , +30 , +60 , +80 , +100 , +130

Der „main“-Thread am Ende des Moduls übermittelt die ausgelesenen Digitalwerte and die angegebene Auswertfunktion und übergibt den berechneten Temperaturwert in 1/10 °C Schritten an die Arrayvariable ‚T[i]‘ (z.B. 64,8 °C = 648). Platzhalter ‚i‘ ist die Nr. des Sensors entsprechend der Klemmenbezeichnung am Reglerboard (1...16). Für die Zuordnung der Sensorklemmen zu den Messadaptern siehe Dokumentation des Reglerboards.

Die Temperaturwerte der Sensoren lassen sich dann aus den anderen Modulen mit ‚temp.T[i]‘ abfragen.

Wichtig

Um Tipp- bzw. Rechenfehler in den Formeln zu bemerken sollte mit simulierten Temperaturen am Sensoreingang die Temperatúrausgabe (bei Linearisierung für jeden Teilbereich!) kontrolliert werden!

Unterer AD-Wert (‚0‘) und oberer AD-Wert (‚10230‘) werden als Fühler-Kurzschluss (-999) bzw. Fühler-Bruch (9999) interpretiert und zurückgegeben (beim AD592 umgekehrt: ‚0‘ = 9999 / ‚10230‘ = -999). D.h. bei Überschreiten des eingestellten Messbereichs am Messadapter kann nicht zwischen einer Messgrenzenüberschreitung und einem Fühlerdefekt unterschieden werden.

4.3.2 Modul „watch“ (Fehlerüberwachung)

Dieses Modul übernimmt die Fehlerüberwachung um Fehlfunktionen zu vermeiden. So kann es z.B. bei einem Fühlerdefekt zu schwerwiegenden Fehlfunktionen kommen. Liegt z.B. ein Fühlerbruch am Kollektorfühler vor (Wert = 9999) wäre die Kollektortemperatur immer größer als die Speichertemperatur und damit die Ladepumpe ständig EIN. Der Speicher würde dann z.B. nachts über den Kollektor entladen. Hat der Speicher eine Nachheizung, würde diese versuchen den Wärmeverlust ausgleichen!

Es werden Fehlervariablen $F(i)$ definiert. Der Status dieser Variablen ist zu Beginn ,0' (kein Fehler) und wird im jeweiligen Fehlerfall auf ,1' gesetzt. Der Status muss dann in den relevanten Threads mit `watch.F(i)` abgefragt werden (z.B. Relais darf nur einschalten, wenn der Status $F(i) = ,0'$ ist)!

Es wurden folgende Fehlervariablen definiert:

F0 - Handbetrieb

Dieser Status wird vom Modul „terminal“ gesetzt und blockiert bei „Handbetrieb“ ($F0 = 1$) die Relais-Threads, damit der Automatikbetrieb der Relais unterbrochen wird.

F1 - Fühlerdefekt

Bei einem Defekt bei einem der angegebenen Temperaturfühler (Kurzschluss / Bruch; siehe auch Modul „temp“) werden die angegebenen Relais aktiv ausgeschaltet und der Status $F1$ auf ,1' gesetzt um die Relais-Threads gegen Wiedereinschalten zu sichern. Nach Behebung aller Fühlerdefekte wird der Status $F1$ wieder zurückgesetzt.

Bemerkungen:

Im Programm werden standardmäßig alle Relais des Reglerboards bei einem Fühlerdefekt gesperrt.

Bei Überschreiten des eingestellten Messbereichs am Messadapter kann nicht zwischen einer Messgrenzenüberschreitung und einem Fühlerdefekt unterschieden werden (siehe Modul „temp“).

F2 – Übertemperatur im Speicher 1

Übersteigt die Temperatur im Speicher 95 °C (Speichergrenztemperatur, abhängig von der Herstellerangabe einstellen) wird das angegebene Relais, das die Speicherladepumpe steuert, aktiv ausgeschaltet und der Status $F2$ auf ,1' gesetzt, um den Relais-Thread der Ladepumpe zu sperren. Damit der Fehler bemerkt wird, wird der Status nicht zurückgesetzt. D.h. die Ladepumpe bleibt auch bei Abkühlen des Speichers gesperrt, bis ein RESET durchgeführt wird!

F10 – Solarüberwachung „dt zu hoch“

Dieser Fehlerstatus wird gesetzt, wenn bei eingeschalteter Solarpumpe die Temperaturdifferenz zwischen Kollektor und Speicher unten für eine festgelegte Zeit (z.B. 20 Min.) einen bestimmten Wert (z.B. 60 K) erreicht oder überschritten hat. Dies kann z.B. auf eine defekte Solarpumpe hindeuten (die verfügbare Wärme am Kollektor wird nicht abgenommen). Dieser Status (Fehlermeldung im Display) wird nur angezeigt und führt zu keiner Blockierung. Er wird aber nicht zurückgesetzt (RESET nötig)!

F11 – Solarüberwachung „Nachtumwälzung“

Dieser Fehlerstatus wird gesetzt, wenn die Kollektortemperatur nachts im festgelegten Zeitraum einen festgelegten Wert (z.B. 40 °C) übersteigt. Dies deutet darauf hin, dass durch eine Schwerkraftumwälzung der Speicher entladen wird (defekte Schwerkraftbremse). Es sind dabei jedoch die Umgebungstemperaturen zu beachten.

Ausgaben im Fehlerfall

Im Fehlerfall wird durch den „error“-Thread ein Signalton über einen angeschlossenen „Beeper“ (Piezoschallwandler) ausgegeben (intermittierender Ton). Gleichzeitig wird ein Ausgangssignal über einen Digitalport ausgegeben (z.B. Blinken einer angeschlossenen LED). Zusätzlich wird der entsprechende Fehler im Display angezeigt (siehe Modul „display“).

Bemerkungen

Es könnten auch mehrere Fühlerüberwachungen (und Status) für verschiedene Systemkreise definiert werden, um nicht alle Relais bei einem Defekt zu sperren.

Bei der Überwachung auf Fühlerdefekt kann auch eine Zeitdauer-Bedingung verknüpft werden um kurzzeitige Störungen zu kompensieren (z.B.: Ein Defekt muss min. 30 sek lang vorliegen, bevor ein Fehler gemeldet wird).

Ein Defekt könnte auch gemeldet werden, wenn z.B. eine ungewöhnliche Temperatur auftritt, oder eine max. Temperaturdifferenz überschritten wird. Auch bei einer ungewöhnlichen Anstiegsgeschwindigkeit könnte ein Fehler gemeldet werden.

Achtung: Bei einem Defekt eines Außentemperaturfühlers zur witterungsgeführten Vorlaufregelung kann es zu Frostschäden kommen. Hier sollte deshalb bei eindeutig zu hoher Temperatur der Wert auf 0 °C gesetzt werden.

4.3.3 Modul „control“ (Regelungsfunktionen)

Dieses Modul übernimmt die Regelungsfunktionen (i.d.R. Schalten der Relais). Die „Relais-Threads“ arbeiten nach dem gleichen Prinzip und unterscheiden sich nur durch die Ein- / Ausschaltbedingungen. Die Threads können gleichermaßen für Pumpen und Ventile verwendet werden.

Zunächst werden im Relais-Thread die Parameter definiert (z.B. Temperaturgrenzen).

Bemerkung:

- Maximaltemperatur: maximale „Solltemperatur“
- Grenztemperatur: maximal zulässige Temperatur (Schutz von Komponenten)

Dann folgt die „loop“-Schleife, die ständig durchlaufen wird. Am Anfang der Schleife erfolgen die Sensorzuweisungen, um bei jedem Durchlauf die aktuellen Temperaturen abzufragen. Danach erfolgen die Abfragen der Aus- und Einschaltbedingungen (getrennt). Anschließend wird noch geprüft, ob das Relais länger als 23 Stunden AUS war und ggf. das Relais für eine definierte Zeit eingeschaltet, um ein Festsetzen der Pumpe / Ventil zu verhindern („Pumpenkick“-Funktion).

Bemerkung: Es sind Mindest-Ein- / Ausschaltzeiten festgelegt (Standard: 1 Minute) um bei einer Fehlfunktion zu häufiges Schalten zu verhindern (oder für Sonderfunktionen). Der Durchlauf des Threads wird für die festgelegte Dauer gestoppt (siehe Quelltext). Des Weiteren kann auch eine Einschalt- und Ausschaltverzögerung (Nachlauf) festgelegt werden.

Im „main“-Thread des Moduls werden die gepufferte Echtzeituhr, die Fehlerüberwachung und die Relais-Threads gestartet. Anschließend erfolgt ggf. die Zeitschaltuhr-Funktion. Die Echtzeituhr wird hier standardmäßig nur als Gangreserve bei Stromausfall verwendet und stündlich von der Systemuhr der CC2 synchronisiert. Die Echtzeituhr bietet aber optional auch eine „watchdog“-Funktion, die bei einem Absturz des Programms einen RESET durchführt (siehe dazu die Hilfe zum Modul „pcf8583“).

Folgende Relais-Threads wurden programmiert:

„Rücklaufanhebung“

Zur Ansteuerung des Ventils / Pumpe zur Pufferspeicherentladung (Rücklaufanhebung des Heizkreises HK).

Das Ventil wird umgeschaltet (Relais EIN), wenn die Differenz zwischen oberer Speicher- und HK-Rücklauftemperatur die Einschalttemperaturdifferenz (ΔT_{EIN}) erreicht oder überschreitet. Die obere Speichertemperatur muss um eine Hysterese über der Speicherminimaltemperatur liegen. Zusätzlich werden noch die Fehlerstatus für Handbetrieb (F0) und Fühlerdefekt (F1) auf ‚0‘ überprüft.

Das Ventil wird wieder zurückgeschaltet (Relais AUS), wenn die Differenz zwischen Speicher und Rücklauf die Ausschalttemperaturdifferenz erreicht oder unterschreitet. Außerdem wird das Ventil zurückgeschaltet, wenn die Minimaltemperatur (oberer Speicherfühler) erreicht oder unterschritten wird.

„Feststoffkessel“

Zur Ansteuerung der Pufferspeicher-Ladepumpe (durch Festbrennstoff-Heizkessel).

Die Pumpe wird eingeschaltet, wenn die Kesseltemperatur die Kesselfreigabetemperatur erreicht oder überschreitet und gleichzeitig die Differenz zwischen Kessel- und unterer Speichertemperatur die Einschalttemperaturdifferenz (ΔT_{EIN}) erreicht oder überschreitet. Die obere Speichertemperatur muss um eine Hysterese unter der Speichermaximaltemperatur liegen.

Die Pumpe wird ausgeschaltet, wenn die Differenz zwischen Kessel- und unterer Speichertemperatur die Ausschalttemperaturdifferenz (ΔT_{AUS}) erreicht oder unterschreitet, oder die Kesselsperretemperatur erreicht oder unterschritten ist. Außerdem wird die Pumpe ausgeschaltet, wenn die Speichermaximaltemperatur erreicht oder überschritten wird.

„Solarpumpe“

Zur Ansteuerung der Solarkreispumpe.

Die Pumpe wird eingeschaltet, wenn die Differenz zwischen Kollektor- und unterer Speichertemperatur die Einschalttemperaturdifferenz (ΔT_{EIN}) erreicht oder überschreitet. Die obere Speichertemperatur muss um eine Hysterese unter der Speichermaximaltemperatur und die Kollektortemperatur um eine Hysterese unter der Kollektorgrenztemperatur liegen. Zusätzlich werden noch die Fehlerstatus für Handbetrieb (F0) und Fühlerdefekt (F1) auf ‚0‘ überprüft.

Die Pumpe wird ausgeschaltet, wenn die Differenz zwischen Kollektor und Speicher unten die Ausschalttemperatur (ΔT_{AUS}) erreicht oder unterschreitet. Außerdem wird die Pumpe ausgeschaltet, wenn die Speichermaximal- oder die Kollektorgrenztemperatur erreicht oder überschritten wird.

Hier ist zum zusätzlichen Schutz vor Fehlfunktion eine Abfrage des Zeitschaltuhr Status verknüpft. Diese verhindert, dass die Pumpe nachts eingeschaltet werden kann (wenn die Zeitschaltuhr-Funktion im „main“-Thread aktiviert ist).

„Thermostat“

Zur Ansteuerung der Nachladepumpe (Nachheizung des WW-Speichers, Thermostatregelung).

Über den Staus der Zeitschaltuhr ergeben sich zwei Zeitfenster:

- „im Zeitfenster“ (z.B. 6:00 – 22:00 Uhr)
- „außerhalb des Zeitfensters“ (entsprechend: 22:00 – 6:00 Uhr)

Für diese Zeitfenster werden (getrennt) Ein- und Ausschalttemperaturen festgelegt.

Die Pumpe wird eingeschaltet, wenn die obere Speichertemperatur die Einschalttemperatur (T_{EIN}) des jeweiligen Zeitfensters erreicht oder unterschreitet. Zusätzlich werden noch die Fehlerstatus für Handbetrieb (F0) und Fühlerdefekt (F1) auf ‚0‘ überprüft.

Die Pumpe wird ausgeschaltet, wenn die obere Speichertemperatur die Ausschalttemperatur des jeweiligen Zeitfensters erreicht oder überschreitet.

Wenn die Zeitschaltuhr-Funktion im „main“-Thread nicht aktiviert ist, werden die Temperaturen von „im Zeitfenster“ verwendet.

Bemerkungen

Zu jeder Ausschaltbedingung muss eine entsprechende Einschaltbedingung bestehen, die das Einschalten verhindert (siehe Quelltext)!

Bei allen Bedingungen muss auf „Relais EIN“ bzw. „Relais AUS“ geprüft werden (Klammersetzung beachten, siehe Quelltext)! Dadurch wird das Durchlaufen der Ein- / Ausschalt routine verhindert, wenn das Relais bereits AUS bzw. EIN ist.

Bei mehreren Relais-Threads sollten die „Pumpenkick“-Funktionen die Relais zeitverzögert einschalten (siehe Quelltext der „Pumpenkick“-Funktion).

Die Temperatur im Warmwasserspeicher sollte 60 °C nicht übersteigen (erhöhte Verkalungsgefahr). Die Nachheizung sollte nur bis auf ca. 45 °C aufheizen um die Versorgungsqualität zu erhalten aber nicht unnötig Energie zu verbrauchen (auch Wärmeverluste).

Die Solarpumpe sollte bei einer Kollektortemperatur (Grenztemperatur) von ca. 130 °C (Dampfbildung, abhängig vom Wärmeträgermedium und Systemdruck) gegen Einschalten gesperrt bzw. ausgeschaltet werden, um die Anlagenkomponenten zu schützen.

Bei der Festlegung der Ein- / Ausschalttemperaturdifferenzen sollte folgendes bedacht werden. Zwar kann durch niedrige Differenzen z.B. der solare Ertrag gesteigert werden, andererseits steigen dadurch die Pumpenlaufzeiten. Untersuchungen am ITW haben ergeben, dass es in der Gesamtenergiebilanz i.d.R. günstiger ist die Ausschalttemperaturdifferenz eher höher und dabei die Hysterese klein zu wählen [8]. Außerdem kann es bei zu klein gewählter Ausschalttemperaturdifferenz (kleiner der Wärmeverluste) zur ungewünschten Entladung des Speichers kommen. Hier sind auch die Ungenauigkeiten der Temperaturmessung zu beachten. Bei Wärmetauschern ist auch das notwendige Temperaturgefälle zur Wärmeübertragung zu berücksichtigen, da die Umwälzpumpe sonst läuft, obwohl keine Wärme übertragen werden kann.

4.3.4 Modul „display“ (LC-Display-Ausgabe)

Dieses Modul übernimmt die Ausgabe auf das LC-Display (programmiert für ein 4x20 Display).

Nach einem RESET wird zunächst ein Startbildschirm mit der Betriebssystemversion, der Heizungsregelungsvariante und -version (nacheinander) angezeigt („Variante“ und „Version“ müssen ggf. angepasst werden).

In der ersten Zeile stehen immer Datum und Uhrzeit. In den restlichen 3 Zeilen werden ausgewählte Temperaturen für eine festgelegte Zeit angezeigt, bevor der „Seitenwechsel“ erfolgt und weitere (ausgewählte) Temperaturen angezeigt werden.

Tritt ein Fehler ein (siehe Modul „watch“), wird / werden nur die Fehlermeldung(en) angezeigt, bis der / die Fehler behoben sind bzw. ein RESET durchgeführt wurde.

4.3.5 Modul „terminal“ („Wartung“ über PC)

Dieses Modul ermöglicht die „Wartung“ der Regelung über einen PC und bietet folgende Funktionen:

- Datum und Uhrzeit stellen
- Ausgabe (Temperaturen, Relaiszustände und Fehlerstatus)
- Sensorkontrolle
- Datenaufzeichnung (über PC)
- Handbetrieb (zum manuellen Schalten der Relais)

Eine Bedienungsanleitung zu diesem Modul befindet sich in Kap. 7.4.

4.4 Programmvarianten

Für die in Kapitel 2 vorgestellten Anlagentypen wurden vorkonfigurierte Programme zusammengestellt. Diese können direkt in die CC2 geladen und ohne Anpassung verwendet werden (z.B. mit dem Download-Tool). Eventuelle Anpassungen der Parameter (z.B. Temperaturgrenzen) stellen aber kein großes Problem dar.

Die Funktionen sind in Kap. 4.3 beschrieben.

Achtung: Werden die Sensor- / Relaiszuordnungen im Modul „control“ geändert, müssen auch die Zuweisungen in den Modulen „watch“ und „terminal“ entsprechend geändert werden (Siehe Hinweise im Quelltext).

Variante Feststoff 1

Zur Regelung der Speicherentladung über 3-Wege-Ventil oder Pumpe, zur Rücklaufanhebung des Heizkreises (Kap. 2.2).

HWCOM: 57.600 Baud

Displayformat: 4 x 20

Temperaturauswertung: Pt1000, Messbereich: -30 / +130 °C (-30, 0, +50, +100, +130)
 Ausgabewerte kontrollieren!

Fehlerüberwachungen:

- Fühlerüberwachung T1 und T5
- Solarüberwachungen deaktiviert
- Notabschaltung bei Übertemperatur Speicher (bei T1 > 95 °C)
 Es wird nur der Fehlerstatus gesetzt!
- Beeper (Tonsignale) + LED-Blinken an Port P1H.0

Regelung:

- Zeitschaltuhr deaktiviert
- Rücklaufanhebung:

Ventil / Pumpe		Relais 1
Oberer Speicherfühler	TspO	T1
HK-Rücklauffühler	ThkRI	T5
Speicherminimaltemperatur		40 °C (3 K Hysterese)
Einschalttemperaturdifferenz	ΔT_{EIN}	4 K
Ausschalttemperaturdifferenz	ΔT_{AUS}	2 K
Mindest-Einschaltdauer		1 min
Mindest-Ausschaltdauer		1 min
Pumpenkick		10 sek

Bemerkung: Bei Speicherentladung durch einen Wärmetauscher sollten die Temperaturdifferenzen erhöht werden ($\Delta T_{\text{EIN}} = 6 \text{ K}$; $\Delta T_{\text{AUS}} = 4 \text{ K}$)

Variante Feststoff 2

Zur Regelung der Speicherentladung über 3-Wege-Ventil / Pumpe (Rücklaufanhebung) und zusätzlicher Regelung der Speicherbeladung durch Festbrennstoff-Heizkessel (Kap. 2.2).

HWCOM: 57.600 Baud

Displayformat: 4 x 20

Temperaturauswertung: Pt1000, Messbereich: -30 / +130 °C (-30, 0, +50, +100, +130)
 Ausgabewerte kontrollieren!

Fehlerüberwachungen:

- Fühlerüberwachung T1, T5, T9, T13
- Solarüberwachungen deaktiviert
- Notabschaltung bei Übertemperatur Speicher (bei T1 > 95 °C)
 Ladepumpe an Relais 2 AUS
- Beeper (Tonsignale) + LED-Blinken an Port P1H.0

Regelung:

- Zeitschaltuhr deaktiviert
- Rücklaufanhebung:

Ventil / Pumpe		Relais 1
Oberer Speicherfühler	TspO	T1
HK-Rücklauffühler	ThkRI	T5
Speicherminimaltemperatur		40 °C / 3 K Hysterese
Einschalttemperaturdifferenz	ΔT_{EIN}	4 K
Ausschalttemperaturdifferenz	ΔT_{AUS}	2 K
Mindest-Einschaltdauer		1 min
Mindest-Ausschaltdauer		1 min
Pumpenkick		10 sek

Bemerkung: Bei Speicherentladung durch einen Wärmetauscher sollten die Temperaturdifferenzen erhöht werden ($\Delta T_{\text{EIN}} = 6 \text{ K}$; $\Delta T_{\text{AUS}} = 4 \text{ K}$)

- Speicherladepumpe:

Pumpe		Relais 2
Oberer Speicherfühler	TspO	T1
Unterer Speicherfühler	TspU	T9
Feststoffkesselfühler	Tk	T13
Speichermaximaltemperatur		90 °C / 5 K Hysterese
Kesselfreigabetemperatur		65 °C
Kesselsperrrtemperatur		55 °C
Einschalttemperaturdifferenz	ΔT_{EIN}	4 K
Ausschalttemperaturdifferenz	ΔT_{AUS}	2 K
Mindest-Einschaltdauer		1 min
Mindest-Ausschaltdauer		1 min
Pumpenkick		10 sek (30 sek verzögert)

Variante Solar 1

Zur Regelung der Solarkreispumpe zur solaren Beladung des Speichers (Kap. 2.3).

HWCOM: 57.600 Baud

Displayformat: 4 x 20

Temperaturauswertung: Pt1000, Messbereich: -40 / +280 °C (-40, 0, +100, +200, +280)
 Ausgabewerte kontrollieren!

Fehlerüberwachungen:

- Fühlerüberwachung T1, T5, T9
- Solarüberwachungen aktiviert:

- „dt zu hoch“

Fühler	TspU	T9
Fühler	Tkol	T5
Solarpumpe		Relais 1
Dauer		20 min
ΔT		60 K

- „Nachtumwälzung“

Fühler	Tkol	T5
Zeitraum		23:00 bis 5:00
Tkol		> 40 °C

- Notabschaltung bei Übertemperatur Speicher (bei T1 > 95 °C)
 Ladepumpe an Relais 1 AUS
- Beeper (Tonsignale) + LED-Blinken an Port P1H.0

Regelung:

- Zeitschaltuhr deaktiviert
- Solarpumpe:

Pumpe		Relais 1
Oberer Speicherfühler	TspO	T1
Unterer Speicherfühler	TspU	T9
Kollektorfühler	Tkol	T5
Speichermaximaltemperatur		60 °C / 3 K Hysterese
Kollektorgrenztemperatur		130 °C / 15 K Hysterese
Einschaltemperaturdifferenz	ΔT_{EIN}	8 K
Ausschaltemperaturdifferenz	ΔT_{AUS}	5 K
Mindest-Einschaltdauer		1 min
Mindest-Ausschaltdauer		1 min
Pumpenkick		10 sek

Variante Solar 2

Zur Regelung der Solarkreispumpe zur solaren Beladung des Speichers und zusätzlicher Regelung der Nachladepumpe (Kap. 2.3).

HWCOM: 57.600 Baud

Displayformat: 4 x 20

Temperaturauswertung: Pt1000, Messbereich: -40 / +280 °C (-40, 0, +100, +200, +280)
 Ausgabewerte kontrollieren!

Fehlerüberwachungen:

- Fühlerüberwachung T1, T5, T9
- Solarüberwachungen aktiviert:

- „dt zu hoch“

Fühler	TspU	T9
Fühler	Tkol	T5
Solarpumpe		Relais 1
Dauer		20 min
ΔT		60 K

- „Nachtumwälzung“

Fühler	Tkol	T5
Zeitraum		23:00 bis 5:00
Tkol		> 40 °C

- Notabschaltung bei Übertemperatur Speicher (bei T1 > 95 °C)
 Ladepumpe an Relais 1 AUS
- Beeper (Tonsignale) + LED-Blinken an Port P1H.0

Regelung:

- Zeitschaltuhr deaktiviert
- Solarpumpe:

Pumpe		Relais 1
Oberer Speicherfühler	TspO	T1
Unterer Speicherfühler	TspU	T9
Kollektorfühler	Tkol	T5
Speichermaximaltemperatur		60 °C / 3 K Hysterese
Kollektorgrenztemperatur		130 °C / 15 K Hysterese
Einschaltemperaturdifferenz	ΔT_{EIN}	8 K
Ausschaltemperaturdifferenz	ΔT_{AUS}	5 K
Mindest-Einschaltdauer		1 min
Mindest-Ausschaltdauer		1 min
Pumpenkick		10 sek

- Nachladepumpe:

Pumpe	Relais 2
Oberer Speicherfühler TspO	T1
Einschaltemperatur im Zeitfenster	40 °C
Ausschaltemperatur im Zeitfenster	45 °C
Einschaltemperatur außerhalb	30 °C
Ausschaltemperatur außerhalb	35 °C
Mindest-Einschaltdauer	1 min
Mindest-Ausschaltdauer	1 min
Pumpenkick	10 sek (30 sek verzögert)

5 Ausblick

Die Regelungs-Module wurden für grundlegende Aufgaben programmiert. Hier sollen Anregungen für Erweiterungen gegeben werden.

Mögliche Erweiterungen sind (Auswahl):

- Regelung der gesamten Heizungsanlage zur Erzielung einer optimalen Funktion und Energieausnutzung.
- Funkuhr bzw. DCF-Empfänger für automatische Zeitsynchronisation (min. 2 m von CC2 entfernt; ggf. geschirmtes Kabel mit Schirmung auf Masse; Anschluss: invertiert)
- Tastatureingabe am Reglergehäuse
- Speicherung der über Tastatur geänderten Einstellwerte in einem EEPROM
- Speicherung von aufgezeichneten Daten in einem EEPROM
- Fernwartung über Modem
- ...

5.1 Sonderfunktionen

Zeitschaltuhr

Zur zeitgesteuerten Regelung.

Folgende Zeitschaltuhrarten könnten sinnvoll sein:

- Wochenzeitschaltuhr: z.B. unterschiedliche Heizzeiten an Werktagen und am Wochenende
- Jahreszeitschaltuhr: z.B. Urlaubszeiten
- Sonderzeitschaltuhr: z.B. „Party“-Funktion

Beispiel einer Wochenzeitschaltuhr: siehe „zeitschaltuhr“-Modul im Ordner „Weitere Module“.

Nachheizunterdrückung (Nachladeunterdrückung)

Ist z.B. die Solarkreispumpe zur WW-Speicher-Ladung in Betrieb, kann die Nachheizung des Speichers durch den Ölheizkessel unterdrückt werden. Dadurch wird Energie eingespart. Hierbei sind das Temperaturniveau und die Größe des Speichers zu beachten. Es sollte daher die Nachheizung wieder freigegeben werden, wenn trotz laufender Solarkreispumpe die Solltemperatur nach einer bestimmten Zeit oder eine Mindesttemperatur nicht erreicht wird (siehe unten). Zusätzlich könnte die Nachheizung nur in den Zeiträumen freigegeben werden, in denen Warmwasser benötigt wird.

Möglichkeiten:

1. Die Regelung der Nachheizung erfolgt durch die CC2. Hier wird die Regelstrategie nur durch das Programm gelöst.
2. Die Regelung der Nachheizung erfolgt durch eine Fremdregelung. Hier wird der Fremdregelung eine höhere Speichertemperatur vorgetäuscht, in dem ein Zusatzwiderstand (R) in eine Ader der Speicher-Sensorleitung eingebunden wird (siehe Abbildung 14). Der Widerstand kann zusammen mit den Enden der aufgetrennten Ader an die Relaisklemmen angeschlossen werden. Diese Variante funktioniert nur mit PTC-Sensoren (z.B. KTY, Pt). Die Sensorkennlinie muss bekannt sein, um den erforderlichen Widerstand zu bestimmen. Öffnet das Relais (EIN), fließt der Messstrom über den Zusatzwiderstand und es wird eine höhere Temperatur gemessen.

Es sind 2 Varianten möglich:

1. Es wird eine um ca. 20 K höhere Temperatur vorgetäuscht. Die Nachladung wird dann von der CC2 wieder freigegeben (Relais AUS), wenn eine Mindesttemperatur unterschritten wird (oder eine andere Bedingung erfüllt ist).
2. Es wird eine um ca. 10 K höhere Temperatur vorgetäuscht. Wird auch diese Temperatur nicht erreicht, heizt der Ölkessel nach.

Achtung: Das Relais muss potentialfrei (stromlos) sein! z.B. Relais 3 mit nicht belegtem Common-Anschluss.

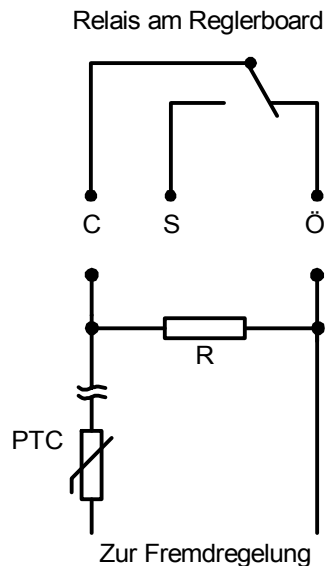


Abbildung 14 Anschluss des Zusatzwiderstands

Überschusswärmenutzung

Bei Überschreiten der Speichermaximaltemperatur kann eine Entladepumpe (und Ventil / Mischer) eingeschaltet werden, um die Überschusswärme an einen Heizkreis abzuführen. So kann z.B. bei einem Festbrennstoffkessel das Ansprechen der Ablaufsicherung verhindert werden.

Siehe auch „Kühlfunktion“.

Kühlfunktion

Bei überhöhter Temperatur im Kollektor kommt es zu verstärkter Alterung des Wärmeträgermediums. Um dies zu verhindern, kann bei Überschreiten einer Kollektormaximaltemperatur (z.B. 120 °C) die Solarpumpe trotz erreichter Speichermaximaltemperatur eingeschaltet werden. Die Pumpe wird wieder ausgeschaltet, wenn die Kollektortemperatur z.B. um 5 K gesunken ist. Dabei darf sich der Speicher bis zur Speichergrenztemperatur (90...95 °C) aufheizen. Der Speicher wird dann am Abend über den Kollektor wieder bis auf die Speichermaximaltemperatur abgekühlt.

Grundsätzlich muss das Solarsystem „eigensicher“ ausgelegt sein. D.h. bei Stillstand der Anlage dürfen keine Schäden durch Überhitzung (z.B. Verdampfen des Wärmeträgermediums) entstehen.

Siehe auch „Überschusswärmenutzung“.

Frostschutzfunktion

Bei Solarkreisen erforderlich die mit Wasser gefüllt sind. Hier wird die Solarpumpe z.B. bei unterschreiten von +4 °C am Kollektorfühler eingeschaltet und bei z.B. bei größer +5 °C ausgeschaltet. Es ist jedoch zu beachten, dass nur die begrenzte Wärmemenge des Speichers zur Verfügung steht.

Bypassfunktion

Bei langen Rohrleitungen und / oder großem Durchmesser der Rohrleitungen kann der Kollektorkreis über einen Bypass vorgewärmt werden, um den Speicher nicht auszukühlen.

Legionellenschutzfunktion

Bei Warmwasser-Speichern > 400 l oder einem Inhalt der Warmwasserleitung > 3 l muss die Temperatur am Austritt des Wassererwärmers mindestens 60 °C erreichen (thermische Desinfektion). Zusätzlich muss der gesamte Speicherinhalt min. 1-mal am Tag auf mindestens 60 °C aufgeheizt werden. Dazu wird zur Durchmischung des Speicherinhalts eine Zirkulationsleitung mit Pumpe zwischen oberem und unterem Speicheranschluss eingebunden (Bypass). Die Nachheizfunktion muss auf mindestens 60 °C eingestellt werden. Die Aufheizung sollte am späten Nachmittag vor einer großen Wasserentnahme erfolgen damit sich der Speicher bis zur nächsten Ladung größtmöglich abkühlen kann (energetisch besser). Vgl. [1] und DVGW W551 Richtlinie.

Zirkulationsfunktion

Bei vorhandener Zirkulationspumpe in der Warmwasserleitung kann diese zeitgesteuert werden (nur bei Warmwasserbedarf EIN) um nicht unnötig Energie zu verbrauchen und den Speicher nicht auszukühlen.

Röhrenkollektorfunktion

Damit die Solarpumpe durch die ungünstige Sensorposition bei Röhrenkollektoren nicht zu spät eingeschaltet wird, kann abhängig von der Kollektortemperatur (oder Strahlungsmesser) eine Intervallfunktion für die Pumpe verwendet werden, z.B. alle 30 Minuten für 30 Sekunden EIN, damit der Sensor die richtige Temperatur misst.

Wärmengenzählung

Um die genutzte Wärmemenge einer Solaranlage zu messen, ist ein Volumenzähler im Solarkreis erforderlich. Da auch bei konstanter Pumpendrehzahl der Volumenstrom durch Viskositätsschwankungen und nichtstationäre hydraulische Verhältnisse um bis zu 30% schwanken kann, muss dieser für genaue Ergebnisse kontinuierlich gemessen werden [1]. Hierzu werden üblicherweise Volumenzähler verwendet, die nach Durchströmen eines bestimmten Volumens einen elektrischen Impuls ausgeben. Die Impulse können über einen Digitalport der CC2 gezählt werden kann. Der Volumenzähler wird im Rücklauf des Kollektorkreises installiert. Die Vor- und Rücklauftemperatur werden unmittelbar vor dem Eintritt in den WT bzw. nach dem Austritt aus dem WT des Solarspeichers gemessen (vgl. [1]).

Die Wärmemenge wird mit folgender Formel berechnet:

$$Q = m \cdot c \cdot \Delta T \cdot t = V \cdot \rho \cdot c \cdot \Delta T \cdot t$$

Q - Wärmemenge

V - Volumenstrom im Kollektorkreis

ρ - Dichte des Wärmeträgermediums

c - spezifische Wärmekapazität des Wärmeträgermediums

ΔT - Temperaturdifferenz zwischen Kollektorkreisvor- und Rücklauf ($T_{VL} - T_{RL}$)

Dichte und spez. Wärmekapazität sind temperaturabhängig und müssen für genaue Ergebnisse auch temperaturabhängig verwendet werden.

Heizungsmischer-Ansteuerung

Stellmotoren für Mischer sind in zwei Ausführungen erhältlich. Stetig bzw. proportional (Stellwinkel abhängig vom Eingangssignal z.B. 0...10 V) oder unstetig (Auf / Zu). Proportionale Stellmotoren sind sehr teuer. Mit der CC2 lässt sich jedoch ein Mischer mit unstetigem Stellmotor über variable Einschalt- / Pausendauern regeln. Hierbei „tastet“ sich die Regelung an den Sollwert heran, indem bei einer Abweichung der Mischer für eine bestimmte Zeit auf- oder zugefahren wird. Um eine Temperaturänderung zu ermöglichen wird für eine bestimmte Pausendauer gewartet und dann die Abweichung erneut überprüft usw. Zusätzlich könnte noch eine Pumpenlogik verknüpft werden, die die Heizkreispumpe ausschaltet, wenn der Mischer für eine bestimmte Zeit zugefahren wurde.

Es werden 2 Relais benötigt (Auf / Zu). Die Abschaltung an den Endlagen erfolgt über Endlagenschalter des Stellmotors.

Vorraussetzung für eine korrekte Funktion ist eine (nahezu) lineare Temperaturkennlinie des Mischers. D.h. bei einer bestimmten Änderung des Öffnungswinkels erfolgt immer die gleiche Temperaturerhöhung / -verringern. Dies ist nicht bei jedem Mischer gegeben.

Damit der Mischer die gewünschte Vorlauftemperatur auch einstellen kann, sollte bei Heizkreisen eine Kesseltemperaturüberhöhung (Wärmeverluste, ca. 5 K) eingestellt werden.

Zur Ansteuerung wurde ein „mischer“-Modul programmiert (siehe Ordner „Weitere Module“).

Heizkurve

Berechnung der Vorlauftemperatur in Abhängigkeit der Außentemperatur (witterungsgeführte Heizkreisregelung). Die Außentemperatur sollte über einen längeren Zeitraum (ca. 1 Std.) gemittelt werden, um kurzfristige Schwankungen auszugleichen. Hier könnte noch eine Pumpenlogik verknüpft werden, die die Heizkreispumpe ausschaltet, wenn die Außentemperatur z.B. 1 K über der Raumsolltemperatur liegt.

Siehe „heizkurve“-Modul im Ordner „Weitere Module“.

Speicherladeregung

Zur Beladung eines Schichtspeichers in mehreren Ebenen. Hier wird der Kollektorvorlauf über ein 3-Wege-Ventil entsprechend seiner Temperatur in die Ebene (Temperaturschicht) geleitet, deren Temperatur am ähnlichsten ist.

Drehzahlregelung

Zur Optimierung der Heizungsanlage kann die Durchflussmenge über eine Drehzahlregelung den Betriebsbedingungen kontinuierlich angepasst werden.

In Heizkreisen ist die Pumpenleistung für den max. Bedarf ausgelegt. Im Normalbetrieb muss der Durchfluss dann gedrosselt werden (Energieverlust). Besser geeignet ist hier eine Drehzahlregelung.

Der Betrieb einer Solaranlage kann durch 2 übliche Regelstrategien optimiert werden (vgl. [1], auch [5]):

1. **Temperaturdifferenzregelung:** Hier wird die Temperaturdifferenz durch Durchflussmengenänderung möglichst nah an der Mindesttemperaturdifferenz gehalten. Dadurch werden die Wärmeverluste minimiert.
2. **Zieltemperaturregelung:** Hier wird die Durchflussmenge so geregelt, dass sich eine Zieltemperatur (z.B. Warmwassersolltemperatur) im Vorlauf einstellt. Dadurch steht schon nach kurzer Zeit Warmwasser mit hoher Temperatur zur Verfügung. Diese Strategie ist jedoch nur mit einem Schichtenspeicher mit entsprechender Ladevorrichtung bzw. Regelung sinnvoll bzw. möglich. Auch die Regelung der Nachheizung sollte darauf abgestimmt sein.

Eine weitere Anwendung ist die Warmwasseraufbereitung nach dem Durchlaufprinzip. Hier wird über einen externen Wärmetauscher zur Speicherentladung immer soviel Warmwasser durch den WT geleitet, wie zur Aufheizung des abgenommenen Frischwassers erforderlich ist. Hierzu ist eine sehr schnell reagierende Regelroutine mit entsprechendem Temperaturfühler erforderlich [1].

Es gibt verschiedenen Möglichkeiten die Pumpe anzusteuern bzw. zu regeln. Im Wesentlichen sind folgende Verfahren üblich:

1. Frequenzumrichter: Teuer! Nur bei großen dreiphasigen Motoren sinnvoll.
2. Differenzdruckregelung:
Interne Pumpenregelung. Geeignet wenn ein Mindestdruck im System erforderlich ist. Nicht optimal wenn nach Wärmebedarf geregelt werden soll. Kommt nur für den Heizkreislauf in Betracht (üblich).
3. Pulsweitenmodulation (Pulse Width Modulation, PWM):
Auch (Im-)Puls-Paket-Steuerung genannt. Das Relais bzw. die Pumpe wird in sehr kurzen Abständen EIN / AUS geschaltet. Durch die Trägheit des Pumpenrotors/-Welle läuft diese nach, bevor der nächste Impuls kommt. Dadurch kommt es so, bei richtiger Regelung, nie zum Stillstand. Es ergibt sich eine mittlere Drehzahl bzw. Leistung. Da nur Energie während des „EIN-Signals“ verbraucht wird, ist der Wirkungsgrad sehr gut. Die Pumpe darf nur über ein „null-spannung-schaltendes“, elektronisches Lastrelais (ELR) angesteuert werden.

Geeignet für einphasige Nassläuferpumpen ohne interne Schaltelektronik (z.B. im Heizungsbau übliche Standard-Umwälzpumpen). Es sollten nur Qualitätspumpen verwendet werden (Grundfos, Wilo, KSB, DAB usw.).

Eine Mindestdrehzahl von i.d.R. 30 % sollte nicht unterschritten werden um eine Mangelschmierung zu verhindern.

Dieses Verfahren ist in der Solartechnik weit verbreitet.

Die CC2 stellt 2 PWM-Ports zur Verfügung. Mit den Relaisplatinen „ELR 1“ oder „ELR 4 (SR)“ von CC-TOOLS lassen sich so max. 2 Pumpen drehzahl geregelt ansteuern.

6 Fazit

Die hier vorgestellte Lösung ist sehr leistungsfähig und für (fast) jede, nicht nur heizungstechnische, Regelungsaufgabe geeignet. So kann die Regelung z.B. parallel zur Heizungsregelung auch noch weitere Aufgaben übernehmen (z.B. „Haussteuerung“). Es ist jedoch der unter Umständen nicht unerhebliche Programmieraufwand zu berücksichtigen. Für viele Fälle sollte die Anpassung durch Erweiterung der hier vorgestellten Programm-Module aber keine größeren Schwierigkeiten bereiten. Eine (zeitintensive) Einarbeitung in die Programmierung ist dann aber unumgänglich.

Preislich bietet die Lösung kaum Vorteile. Insbesondere für einfache Aufgaben, wie den Beispielen in Kap. 2.2 und 2.3, sind die einfachen Regelungen der Systemanbieter deutlich kostengünstiger.

Die Vorteile der hier vorgestellten Lösung steigen mit der Komplexität der Aufgabe. Außerdem lässt sich das System nahezu beliebig erweitern, was bei Standard-Regelungen i.d.R. nicht gegeben ist.

Nachteilig ist der nicht unerhebliche Mehraufwand bei der Installation und Konfiguration, fehlende Garantie und Support für das Gesamtsystem und die relativ unsichere Verfügbarkeit der Komponenten.

Abschließend lässt sich festhalten, dass die hier vorgestellte Lösung hauptsächlich für interessierte „Bastler“ oder für Aufgaben geeignet ist, die nicht mit Standard-Regelungen zu lösen sind.

7 Anhang

7.1 Anlagen

Dieser Ausarbeitung liegt eine CD-ROM mit allen benötigten / verwendeten Dateien bei.

7.2 Hinweise zur Programmierung in C2

Hinweis:

Beim kopieren von Quellcode aus einem Dokument kann es beim Einfügen in die IDE bei eingerückten Zeilen zu Fehlermeldungen kommen. Die Zeilenanfänge müssen dann an den linken Seitenrand gesetzt werden.

Siehe Kap. 4 dieser Arbeit für einführende Hinweise.

Es sollten unbedingt die CC2Net-FAQs, das Kapitel 4.4.3 und die kompletten Kapitel 5, 6 und 7 im CC2-Handbuch gelesen werden (mehrmals)!

Es sollen hier ergänzende Hinweise gegeben werden, die nicht oder nur unzureichend im Handbuch beschrieben sind.

Module und Projekte

Die Programmiersprache C2 basiert auf Modulen. In einem Modul werden zusammengehörende Abläufe zusammengefasst, um diese leichter wieder verwenden zu können und das Programm übersichtlicher zu gestalten.

Als Grundlage dienen die Systemmodule, die die Zugriffe auf die Systemressourcen der CC2 übernehmen.

Module bauen aufeinander auf. So greifen Funktionen in einem Modul auf Funktionen in einem anderen zu. Das Modul, auf das zugegriffen werden soll, muss aktiviert und bekannt sein (d.h. vor dem aufrufenden Modul stehen).

Mehrere Module werden in einem Projekt zusammengefasst. In einem Projekt sind alle Module enthalten, die aktiviert wurden (zum Aktivieren eines Moduls muss dieses mit der linken Maustaste markiert werden und dann mit der rechten Maustaste „Modul in Projekt verwenden“ auswählen).

Ein neues Projekt enthält nur einige wichtige Systemmodule. Es müssen noch Anwendermodule („Projektmodule“) hinzugefügt bzw. erstellt werden (Modulreihenfolge beachten).

Bei mehreren geöffneten Projekten ist immer nur eins aktiv (mit einem Klick mit der rechten Maustaste auf ein Projekt kann dieses aktiviert werden).

Vor dem Laden des Projekts in die CC2 muss dieses kompiliert werden (angezeigte Fehler vorher korrigieren).

Threads

Threads sind immer Schleifen die dauerhaft durchlaufen werden.

Es gibt zwei Arten von Threads:

- Normale Threads: Diese „stehen“ zu Beginn (Priorität ,0') und müssen von einem anderen Thread gestartet werden.
- Main-Threads: Diese werden automatisch gestartet (Priorität ,32').

Jedes Modul kann einen Main-Thread enthalten (muss ganz am Ende des Moduls stehen).

Initialisierungen sollten immer nur einmal beim Start ausgeführt werden. Daher sollten Threads immer folgende Struktur haben:

```
thread xy
{
    // Definition der Variablen
    // Initialisierungen
    // Starten weiterer threads

    loop
    {
        // Hauptroutinen
    }
}
```

Sollen keine Hauptroutinen ausgeführt werden bzw. nur einen Durchlauf, sollte der Thread nach dem Starten weiterer Threads mit dem Befehl ,halt;' gestoppt werden.

Achtung:

Bei mehreren Threads kann ein einzelner Thread während des Ablaufs unterbrochen und die Rechenzeit an den nächsten übergeben werden (je nach Priorität). Das sollte man bei der Programmierung, insbesondere bei Abfragen von (Uhr-) Zeiten, berücksichtigen.

Auch muss berücksichtigt werden, dass die Threads nicht wirklich parallel laufen, sondern diese immer nur für eine bestimmte Zahl von Arbeitsschritten Rechenzeit erhalten (z.B. Timer, hier sollte nicht auf ,==' geprüft werden, da der Zeitpunkt evtl. verstreicht, während der abfragende Thread gerade keine Rechenzeit hat)!

Berechnungen

Bei Berechnungen sind die verwendeten Datentypen zu beachten. Es sollte möglichst auf die Verwendung des Datentyps „float“ (Fließkommazahlen) verzichtet werden, auch wenn die Berechnung dadurch übersichtlicher würde. Durch die Verwendung von Floatvariablen wird erheblich mehr Rechenzeit und Speicherplatz verbraucht. Außerdem können Fehlermeldungen bzw. Programmfehler, in Bezug auf Berechnungen, nur bei Floatvariablen auftreten.

Bei den anderen Datentypen werden Nachkommastellen ignoriert (es wird nicht gerundet!). Daher sollte eine Division möglichst am Schluss stehen! Hier kann es aber trotzdem zu Abweichungen vom exakten Ergebnis kommen (wg. ignorierte Nachkommastellen)!

Wenn in einer Berechnung der Wertebereich des Datentyps überschritten werden kann, muss mindestens eine Variable in der Berechnung vom höheren Datentyp sein (es reicht nicht, wenn die zu füllende Variable vom nötigen Datentyp ist)!

Unbedingt die Hinweise zu Berechnungen in den FAQs beachten!

Strings

Bei der Ausgabe von Strings ist folgendes zu beachten. Die max. String-Länge beträgt 30 Zeichen. Sollen längere Zeilen ausgegeben werden, muss die Zeichenfolge aufgeteilt werden. Bei der Ausgabe über HWCOM muss bei der Ausgabe mehrerer Stringvariablen ohne Zeilenwechsel (mit `hwcom.ret()`!) die Schnittstelle nach jedem String (außer dem letzten) auf Bereitschaft geprüft werden. z.B.:

```
s="Teil 1";  
hwcom.print(s); wait hwcom.ready();  
s="Teil 2";  
hwcom.print(s);
```

Alternativ kann auch abwechselnd mit 2 Stringvariablen gearbeitet werden. z.B.:

```
s1="Teil 1";  
hwcom.print(s1);  
s2="Teil 2";  
hwcom.print(s2);  
s1="Teil 3";  
hwcom.print(s1);
```

Die Ausgabegeschwindigkeit ist hier höher als bei der 1. Variante.

Siehe auch Hinweise in den FAQs.

Zeitverzögerungen

Bei Verwendung des „sleep“-Befehls ist der Wertebereich zu beachten (siehe Handbuch)! Der negative Wert von `-,5536'` entspricht dabei einer Minute (60000 ms). Für längere Zeitverzögerungen gibt es mehrere Möglichkeiten (siehe unten).

Es sei noch mal darauf hingewiesen, dass ein "sleep" die anderen Threads nicht „ausbremst“, da der Thread, in dem das "sleep" steht, sofort die Rechenzeit weitergibt, solange die Wartezeit noch nicht verstrichen ist.

1. Variante: Mehrere „sleeps“ hintereinander, z.B.:

```
sleep -5536; // 1 Minute  
sleep -5536; // noch 1 Minute
```

2. Variante: Durch eine "for"-Schleife, z.B.:

```
for i=0...<5 sleep -5536; // entspricht 5 Minuten
```

- 3. Variante:** Zählen der verstrichenen Minuten, als Alternative zu Variante 1 und 2.
Nur für längere, ungenaue Wartezeiten, da eine Wartezeitverkürzung von bis zu einer Minute entstehen kann (Abfrage startet beim Minutenwechsel mit ,1')!

```
thread xy
{
    byte WaitTime, timer, minute;

    WaitTime = 30; // Zeit in Minuten

    timer=0;
    minute=system.minute();
    while timer<WaitTime
    {
        wait minute!=system.minute();
        minute=system.minute();
        timer=timer+1;
    }

    // Anweisungen werden nach Ablauf der Zeit ausgeführt
}
```

- Hinweis:** Diese genannten Varianten blockieren den Thread, in dem sie stehen, für die Dauer der Wartezeit.

- 4. Variante:** Ähnlich wie Variante 3. Hier wird jedoch ein eigener Timer-Thread verwendet und ein „timerflag“ gesetzt. Der Zustand des „timerflag“ kann dann von einem anderen Thread abgefragt werden (so kann der abfragende Thread weiterlaufen, bis der vorgesehene Zustand des „timerflag“ eintritt).

```
byte timerflag; // globale Variable (ganz oben im Modul)

...

thread timer
{
    byte WaitTime, timer, minute;

    WaitTime = 30; // Zeit in Minuten

    timerflag=1;
    timer=0;
    minute=system.minute();
    while timer<WaitTime
    {
        wait minute!=system.minute();
        minute=system.minute();
        timer=timer+1;
    }
    timerflag=0;
    reset;
}
```

Dieser Thread wird dann vom aufrufenden Thread gestartet. Nach dem Start ist ‚timerflag‘ = ‚1‘ und wird nach Ablauf der Zeit wieder auf ‚0‘ und der Thread zurückgesetzt (reset).

Auch hier kann zu einer Wartezeitverkürzung kommen (siehe Variante 3).

Für genaue, längere Wartezeiten (im Millisekundentakt) kann das Modul „timer“ von Andreas Sperling verwendet werden (Ordner „Weitere Module“).

Soll eine Bedingung über eine bestimmte Zeit bestehen, bevor eine Aktion ausgeführt wird, siehe Beispiel im Modul „watch“ (Thread „dThoch“).

Für eine Zeitschaltuhr-Funktion siehe Modul „zeitschaltuhr“ (Ordner „Weitere Module“).

Weitere Hinweise

Viele Bezeichner sind ‚0‘-basiert (d.h. das 1. Element wird mit ‚0‘ bezeichnet), z.B. Ports, Arrayelemente, Anzahl der Durchläufe bei „for“-Schleifen, usw.

Reihenfolge beim Aufruf von Variablen oder Threads, Funktionen beachten (muss vor dem Aufruf bekannt sein).

Vor einem Anweisungsblock darf kein Semikolon stehen (siehe Kap. 5.3.4 im Handbuch) Der Block würde sonst unabhängig von der „if“-Abfrage immer ausgeführt, z.B.:

```
if x != 0      // hier darf kein Semikolon stehen!  
{  
    Anweisung 1;  
    Anweisung 2;  
}
```

Es ist auf die richtige Verwendung der „else“-Bedingung bei „if“-Abfragen zu achten (Kap. 5.10.1 im Handbuch). Wird kein „else“ verwendet, werden alle folgenden „if“-Abfragen geprüft. Bei Verwendung von „else“ wird (nur) die nächste Abfrage übersprungen, wenn die davor stehende Abfrage erfüllt wurde. Hier kann es bei verketteten Bedingungen in den Abfragen zu Fehlern kommen!

Bei Zugriffen auf Arrayelemente muss der Indexbereich beachten werden (‚0‘-basiert !). Die Gültigkeit der Zugriffe wird in der IDE nicht überprüft und es kann bei Missachtung des Bereichs zu Fehlern im Programm kommen. Siehe auch Kap. 5.5.3 im Handbuch.

Statusabfragen sollten immer wie folgt durchgeführt werden:

```
if x == 0    für Prüfung auf AUS  
if x != 0    für Prüfung auf EIN
```

Bei Warteschleifen mit „wait min != system.minute()“ kann es zu einer Abweichung von bis zu einer Minute kommen, da die Abfrage beim Minutenwechsel startet.

Werden mehrere Zeit- / Datumselemente abgefragt, ist auf den Minuten-, Stunden-, Tageswechsel zwischen den Einzelabfragen zu achten. Hier kann es sonst zu falschen

Ergebnissen kommen. Dies kann z.B. mit folgender Befehlszeile direkt vor den Abfragen verhindert werden (siehe auch Kap. 7.11 Handbuch):

```
if system.second()==59 and system.minute()==59 wait  
system.second()==0;  
  
// ab hier die Abfragen
```

Die Abfragen sollten oben im Thread erfolgen.

Häufige Fehler

- Ein erforderliches Modul ist nicht aktiviert
- Falsche Reihenfolge der (Anwender-) Module, Threads, Funktionen, Variablen (Name nicht bekannt)
- Bezeichner ist im aufgerufenen Modul nicht definiert
- Anweisung nicht mit Semikolon abgeschlossen bzw. fehlende Endklammer bei Anweisungsblöcken
- Falsche oder fehlende Klammersetzung bei logischen Verknüpfungen
- Gültigkeit des Arraybereichs verletzt (Achtung: es wird keine Fehlermeldung in der IDE angezeigt)
- Zugriff auf ein falsches Element wenn dieses ,0'-basiert ist

Fehler werden links unten im IDE-Fenster und durch ein rotes Kreuz vor der Befehlszeile angezeigt. Mit einem Doppelklick auf die Fehlermeldung springt der Cursor zur Fehlerstelle.

Achtung:

Oft ist der verursachende Fehler nicht an der Stelle, an der ein Fehler angezeigt wird (Folgefehler). Meist verursacht durch fehlenden Abschluss einer Anweisung (','') oder eines Anweisungsblocks (','').

7.3 Programm-Quelltexte

7.3.1 Modul „temp“ (Temperatursensor-Auswertung)

```
/* **** */
/* Modul zur Auswertung der Temperatursensoren am CC2-ReglerBoard */
/* Vorlage : André Helbig */
/* Geändert : Malte Alpers (malte.alpers@gmx.de) */
/* Version : 1.0 */
/* Datum : 05.07.2005 */
/* **** */
/* WICHTIG: */
/* - Verwendete Auswertfunktionen müssen angepasst werden, wenn die */
/* Messadapter für einen anderen Messbereich abgeglichen werden! */
/* - Im thread "main" muss ggf. der Name der zu verwendende */
/* Auswertfunktion geändert werden! */
/* - ggf. individuelle Offsetwerte der einzelnen Sensoren setzen */
/* (Abweichung von einer Bezugstemperatur) */
/* - Ausgabewerte T[1]...T[16] der Sensoren 1...16 am Reglerboard */
/* in 1/10°C (z.B. 21.6°C = 216)! */
/* - Erkennung von Fühlerbruch und -kurzschluss: */
/* - bei Wert = -999 : Fühlerkurzschluss */
/* - bei Wert = 9999 : Fühler unterbrochen oder nicht angeschlossen*/
/* **** */

int T[17]; // Anzahl der Sensoren (Ports) + 1 (16 + 1 = 17) !!!

// Setzen der individuellen Offset-Werte in +/- 1/10°C (z.B. Offset 2K =
20)
// Sensor:      1,2,3,4 ,      5,6,7,8 ,      9,10,11,12 ,      13,14,15,16
const OFFSET[] = 0,0,0,0 ,      0,0,0,0 ,      0,0,0,0 ,      0,0,0,0;

/* **** */
/* Auswertfunktion für AD592 */
/* Funktion: y = (y * Messbereich)/1023 +(-) Offset */
/* Messbereich in K (z.B. -20 bis +115°C = 135 K) */
/* Offset in 1/10 K (z.B. Messbereich ab -20°C : Offset = -200) */
/* Rückgabewert in 1/10°C (z.B. 21.6°C = 216)! */
/* **** */
/* WICHTIG: */
/* - Für andere Messbereiche Werte in der Berechnung ändern! */
/* - Klammern und Reihenfolge nicht ändern (wg. int, long)! */
/* - Ausgabewerte mit vorgegebenen Temperaturwerten am Sensoreingang */
/* mit 'rbkalib' kontrollieren um Tipp-/Rechenfehler zu bemerken! */
/* **** */
function getAD592(int x, byte i) returns int
{
    long y;
    y=x; // Datentypwechsel von 'int' auf 'long'
    if x==10230 return -999; // Fühler Kurzschluß
    if x==0 return 9999; // Fühler unterbrochen

    y=(y*135)/1023-200; // Messbereich = -20 ... +115°C

    y=y+OFFSET[i]; // Addieren des individuellen Offsets
    return y;
}
```

```

/*****
/* Auswertfunktion für Pt1000 (linearisiert) */
/* Funktion: y=((y-AD)*TempBereich)/ADBereich +(-) Offset */
/* TempBereich in K (z.B. +60 bis +80°C = 20 K) */
/* ADBereich = (oberer AD-Wert MINUS unterer AD-Wert)/10 (integer!) */
/* Offset in 1/10 K (z.B. Bereich ab -20°C : Offset = -200) */
/* Rückgabewert in 1/10°C (z.B. 21.6°C = 216)! */
*****/
/* WICHTIG: */
/* - Linearisierung für andere Messbereiche: AD-Werte müssen neu ermit- */
/* telt werden! Soll keine Linearisierung erfolgen, siehe 'getAD592'. */
/* - Werte in den Berechnungen unten entsprechend ändern! */
/* - Klammern und Reihenfolge nicht ändern (wg. int, long)! */
/* - Ausgabewerte mit vorgegebenen Temperaturwerten am Sensoreingang */
/* für jeden Bereich kontrollieren um Tipp-/Rechenfehler zu bemerken! */
*****/
function getPT1(int x, byte i) returns int
/* Temperaturbereiche für Linearisierung
    AD-Wert
    +280°C = 10230
    +200°C = 7760
    +100°C = 4600
    0°C = 1340
    -40°C = 0
*/
{
    long y;
    y=x; // Datentypwechsel von 'int' auf 'long'
    if x==0 return -999; // Fühler Kurzschluß
    if x==10230 return 9999; // Fühler unterbrochen

    if x>=7760 y=((y-7760)*80)/247+2000;
    else
    if x>=4600 y=((y-4600)*100)/316+1000;
    else
    if x>=1340 y=((y-1340)*100)/326;
    else
    y=(y*40)/134-400;

    y=y+OFFSET[i]; // Addieren des individuellen Offsets
    return y;
}

/*****
/* Auswertfunktion für Pt1000 (linearisiert) */
/* Funktion: y=((y-AD)*TempBereich)/ADBereich +(-) Offset */
/* TempBereich in K (z.B. +60 bis +80°C = 20 K) */
/* ADBereich = (oberer AD-Wert MINUS unterer AD-Wert)/10 (integer!) */
/* Offset in 1/10 K (z.B. Bereich ab -20°C : Offset = -200) */
/* Rückgabewert in 1/10°C (z.B. 21.6°C = 216)! */
*****/
/* WICHTIG: */
/* - Linearisierung für andere Messbereiche: AD-Werte müssen neu ermit- */
/* telt werden! Soll keine Linearisierung erfolgen, siehe 'getAD592'. */
/* - Werte in den Berechnungen unten entsprechend ändern! */
/* - Klammern und Reihenfolge nicht ändern (wg. int, long)! */
/* - Ausgabewerte mit vorgegebenen Temperaturwerten am Sensoreingang */
/* für jeden Bereich kontrollieren um Tipp-/Rechenfehler zu bemerken! */
*****/
```

```
function getPT2(int x, byte i) returns int
/* Temperaturbereiche für Linearisierung
    AD-Wert
    +130°C = 10230
    +100°C = 8360
    +50°C = 5180
    0°C = 1960
    -30°C = 0
*/
{
    long y;
    y=x; // Datentypwechsel von 'int' auf 'long'
    if x==0 return -999; // Fühler Kurzschluß
    if x==10230 return 9999; // Fühler unterbrochen

    if x>=8360 y=((y-8360)*30)/187+1000;
    else
    if x>=5180 y=((y-5180)*50)/318+500;
    else
    if x>=1960 y=((y-1960)*50)/322;
    else
    y=(y*30)/197-300;

    y=y+OFFSET[i]; // Addieren des individuellen Offsets
    return y;
}

/*****
/* Abfrage und Auswertung der Sensorwerte und Speicherung als */
/* Temperaturen in T[1]...T[16] */
/* Rückgabewert in 1/10°C (z.B. 21.6°C = 216)! */
/*****
/* WICHTIG: */
/* - Für anderen Sensortyp Funktionsname ändern (s.u.)! */
/* - Bei Mischbestückung muss eine Abfrage mit 'if' erfolgen, z.B.: */
/*     for i=0...15 */
/*     { */
/*         if i%4==2 or i%4==3 T[i+1]=getAD592(rbports.ad(i),i); */
/*         else T[i+1]=getPT1(rbports.ad(i),i); */
/*     } */
/* Im Beispiel werden Messadapter 'C' + 'D' mit der Auswertfunktion */
/* für AD592 und die anderen Messadapter mit 'PT1' ausgewertet. */
/*     i%4==0 (Messadapter A) */
/*     i%4==1 (Messadapter B) */
/*     i%4==2 (Messadapter C) */
/*     i%4==3 (Messadapter D) */
/*****
thread main
{
    byte i;
    rbports.init(1);
    loop
    {
        for i=0...15 // Multiplexer-Port 0...15
        {
            T[i+1]=getPT2(rbports.ad(i),i); // ggf. Funktionsname ändern
            (AD592,PT1,PT2)
        }
        sleep 250;
    }
}
```

7.3.2 Modul „watch“ (Fehlerüberwachung)

```

/*****
/* Modul zur Fehlerüberwachung */
/* Autor : Malte Alpers (malte.alpers@gmx.de) */
/* Version : 1.0 */
/* Datum : 01.07.2005 */
/* Geändert : */
/*****
/* Massnahmen bei z.B. Fühlerdefekt. Es kann sonst schwerwiegenden */
/* Fehlfunktionen kommen (z.B. Speicherentladung über Kollektor) */
/*****
/* WICHTIG: */
/* - Der 'error'-thread muss vom 'main'-thread (Modul 'control') ge- */
/* startet werden!!! */
/* - Nicht zutreffende oder benötigte Überwachungen deaktivieren !!! */
/* - Programm anpassen! Sensoren anpassen, ggf. weitere hinzufügen! */
/* - Es sollten nur Sensoren überwacht werden, die regelnd in den */
/* Programmablauf eingreifen, um diesen nicht unnötig zu stoppen */
/* - Die entsprechende Fehlervariable bzw. Status muss in den zu */
/* reagierenden threads/functions abgefragt werden! (s. 'Relais'- */
/* threads)! */
/* (Wert = 1 => Fehler) */
/* - Ausgaben im Modul 'display' anpassen! */
/*****

//----- Variablen für Fehlerstatus -----

byte F0, // Status für Handbetrieb
    F1, // Fühlerdefekt
    F2, // Übertemperatur Speicher 1
    F10, // Solarüberwachung "dT zu hoch"
    F11; // Nachtumwälzung

/*****
/* Solarüberwachung - "dT zu hoch" */
/* Fehlerstatus wird gesetzt, wenn bei eingeschalteter Solarpumpe die */
/* Temperaturdifferenz zwischen Kollektor und Speicher unten für eine */
/* festgelegte Zeit einen bestimmten Wert (z.B. 60 K) überschritten */
/* hat. Dies deutet z.B. auf eine defekte Solarpumpe hin (die verfüg- */
/* bare Wärme wird nicht abgenommen). */
/*****
/* WICHTIG: */
/* - Relais Nr. und überwachte Fühler anpassen! */
/* - Der Fehlerstatus wird nicht zurückgesetzt! Es muss ein RESET */
/* durchgeführt werden. */
/* - Wird vom 'error'-thread gestartet (s.u.) */
/*****
thread dThoch // F10
{
    byte R, t, timer, min;
    int Tkol, TspU;

    R = 0; // Relais Nr. Solarpumpe, '0'-basiert! (Relais 1 = '0')
    t = 20; // Bedingung muss für (ca.) 't' Minuten erfüllt sein

    wait rbports.get(R) != 0; // Solarpumpe auf EIN prüfen

```

```
timer = 0;
min = system.minute();

while timer < t
{
    wait min != system.minute();
    min = system.minute();
    timer = timer + 1;

    TspU = temp.T[9]; // Sensor Speicher unten
    Tkol = temp.T[5]; // Sensor Kollektor

    if Tkol - TspU < 600 // in 1/10 K (z.B. 60 K = 600)
        or rbports.get(R) == 0 reset; // Solarpumpe auf AUS prüfen
}
F10 = 1; // Fehlerstaus setzen
halt; // CC2 muss durch 'RESET' neu gestartet werden
}

/*****
/* Solarüberwachung - "Nachtumwälzung"
/* Fehlerstatus wird gesetzt, wenn die Kollektortemperatur Nachts im
/* festgelegten Zeitraum einen festgelegten Wert (z.B. 40 °C) über-
/* steigt. Dies deutet darauf hin, dass durch eine Schwerkraftumwäl-
/* zung der Speicher entladen wird (defekte Schwerkraftbremse). Es
/* sind dabei jedoch die Umgebungstemperaturen zu beachten.
*****/
/* WICHTIG:
/* - Kollektorfühler und ggf. Zeitraum und Temperatur anpassen!
/* - Der Fehlerstatus wird nicht zurückgesetzt! Es muss ein RESET
/* durchgeführt werden.
/* - Wird vom 'error'-thread gestartet (s.u.)
*****/
thread TkolNacht // F11
{
    byte min;
    int time, Tkol;

    min = system.minute();
    wait system.minute() != min;
    time=system.hour()*100 + system.minute();

    Tkol = temp.T[5]; // Sensor Kollektor

    if (time >= 2300 and time <= 2359 // zwischen 23:00
        or
        time >= 0 and time < 500) // und 5:00
        and Tkol > 400 and Tkol < 9999 // Wenn Kollektortemp > 40 °C (400)
    {
        F11 = 1; // Fehlerstatus setzen
    }
}
```

```
/* ***** */
/* Hauptüberwachungen */
/* 1. Fühlerüberwachungen: Fehlerstatus wird gesetzt wenn ein Sensor- */
/* defekt festgestellt wird (Bruch/Kurzschluss). Signalton über */
/* beeper und LED-Blinken. Nach Behebung des Defekts wird der */
/* Staus zurückgesetzt. */
/* 2. Speicher 1 wird auf Übertemperatur überwacht. Die Ladepumpe */
/* wird abgeschaltet und Fehlerstatus gesetzt. Der Fehlerstatus */
/* wird nicht zurückgesetzt! Es muss ein RESET durchgeführt werden.*/
/* 3. Solarüberwachung wird gestartet ('dThigh', s.o.). */
/* ***** */
/* WICHTIG: */
/* - Muss vom 'main'-thread (Modul 'control') gestartet werden! */
/* - Nicht zutreffende oder benötigte Überwachungen deaktivieren !!! */
/* - Sensoren, Relais und Maßnahmen anpassen, ggf. weitere hinzufügen! */
/* - Es sollten nur Sensoren überwacht werden, die regelnd in den */
/* Programmablauf eingreifen, um diesen nicht unnötig zu stoppen */
/* - Die entsprechende Fehlervariable bzw. Status muss in den zu */
/* reagierenden threads/functions abgefragt werden (s. 'Relais'- */
/* threads)! */
/* (Wert = 1 => Fehler) */
/* - Ausgaben im Modul 'display' anpassen! */
/* ***** */
thread error
{
    byte z;
    sleep 3000; // Wichtig! (muss < der Startverzögerung der Relais-Threads
sein)

    loop
    {
        //----- F1 Fühlerüberwachung T[1], T[5], T[9] -----

        if temp.T[1] == -999 or temp.T[1] == 9999 // Kurzschluss oder Bruch
        or temp.T[5] == -999 or temp.T[5] == 9999 // Kurzschluss oder Bruch
        or temp.T[9] == -999 or temp.T[9] == 9999 // Kurzschluss oder Bruch
        {
            F1 = 1; // Fehlerstatus F1 setzen

            if z == 0
            {
                rbports.set(0,0); // Relais 1 ('0') ausschalten
                sleep 300;
                rbports.set(1,0); // Relais 2 ('1') ausschalten
                sleep 300;
                rbports.set(2,0); // Relais 3 ('2') ausschalten
                sleep 10000;
                z = 1;
            }
        }
        else
        {
            F1 = 0; // Fehlerstatus zurücksetzen
            z = 0;
        }
    }
}
```

```
//----- F2 Übertemperatur Speicher 1 -----  
  
// Notabschaltung wenn Sensor Speicher oben > 95 °C (950)  
if temp.T[1] > 950 and temp.T[1] < 9999 // Sensor Speicher oben  
{  
    rbports.set(0,0); // Ladepumpe an Relais 1 ('0') ausschalten  
                      // Abfrage bei 'Handbetrieb' in 'Terminal'  
anpassen!  
    F2 = 1; // Fehlerstatus F1 setzen  
}  
  
//----- Solarüberwachung -----  
  
run dThoch; // Warnung "dT zu hoch" starten, ggf. deaktivieren  
run TkolNacht; // Warnung "Nachtumwälzung" starten, ggf. deaktivieren  
  
//----- Maßnahmen bei Fehler -----  
  
if F1 != 0 or F2 != 0 or F10 != 0 or F11 != 0 // ggf. weitere  
hinzufügen  
{  
    // Signaltonausgabe (beeper)  
    plm.beep(10); sleep 500; plm.beep(-1); sleep 50;  
  
    // LED-Blinken (z.B. 8 = Port 'PlH.0')  
    ports.set(8,-1); sleep 500; ports.set(8,0); sleep 500;  
}  
}  
}
```

7.3.3 Modul „control“ (Regelungsfunktionen)

```

/*****
/* Hauptmodul zur Heizungsregelung mit dem CC2-ReglerBoard */
/* Autor : Malte Alpers (malte.alpers@gmx.de) */
/* Version : 1.0 */
/* Datum : 02.07.2005 */
/* Geändert : */
/*****
/* WICHTIG: */
/* - Relais-threads anpassen! */
/* - Relais-threads müssen über den 'main'-thread gestartet werden! */
/*****

const ONEMIN = -5536; // entspricht 1 Minute bei 'sleep'
byte ZSUL; // flag für Zeitschaltuhr

/*****
/* Ansteuerung 3-Wege-Ventil / Pumpe zur Pufferspeicherentladung */
/* - Rücklaufanhebung, mit Sockeltemperatur */
/*****
/* - Relais EIN wenn Differenz zwischen Speichertemp. und Heizkreis- */
/* rücklauftemp >= dTein und Speichertemp. >= Sockeltemp. + Hysterese */
/* (3 K). Fehlerstatus F0 und F1 müssen = '0' sein. */
/* - Relais AUS wenn Differenz <= dTaus oder Speichertemp. <= Sockel- */
/* temp. */
/*****
/* WICHTIG: */
/* - Folgende Einstellungen müssen im oberen Bereich kontrolliert und */
/* ggf. geändert werden: */
/* - Relais Nr. */
/* - Min/Max Temperaturen, Temperaturdifferenzen */
/* - Sensorzuweisung (am Anfang der 'loop'-Schleife) */
/* - Es werden Mindest-Ein-/Ausschaltdauer festgelegt um z.B. bei */
/* Regelschwingungen Pumpe und Relais zu schonen (s. Hinweise im */
/* Quelltext). Voreinstellung = 1 Minute. Für Pausen < 1 min muss */
/* die 'for'-Schleife entfernt werden (max. 32767 Millisekunden!) */
/* - Es können Ein- und Ausschalverzögerungen festgelegt werden */
/* (s. Hinweise Quelltext) */
/* - Auf "Relais-Logik" achten (Schließer oder Öffner) */
/* - Pumpenkick: Die Pumpe wird nach 23h Stillstand für eine fest- */
/* gelegte Dauer eingeschaltet (Verhindern des Festsetzens). */
/* (für Ventile ggf. Laufzeit anpassen) */
/* - ACHTUNG: Wenn die EinschaltRoutine startet, ist die Abfrage der */
/* Ausschaltbedingungen für die Mindest-Einschaltdauer */
/* blockiert (auch bei z.B. Übertemperatur)! Ist die Min- */
/* Einschaltdauer sehr groß, sollte die Übertemperatur- */
/* abschaltung über einen anderen thread direkt erfolgen. */
/* (siehe Modul 'watch' !) */
/*****
/* dTein = Einschalt-Temperaturdifferenz, min. dTaus + 2 K (20)!!! */
/* dTaus = Ausschal-Temperaturdifferenz (Wärmeverluste) */
/* dTein = dTaus + Hysterese */
/*****

```



```
thread Rueckklopfanhebung
{
    byte R, i, d;
    int Tsp0, ThkR1, TspMin, dTein, dTaus;

    R = 0; // Relais Nr., '0'-basiert! (Relais 1 = '0')

    TspMin = 400; // Sockeltemperatur im Speicher in 1/10 °C (z.B. 40 °C =
400)
    dTein = 40; // min. dTaus + 2 K !!!, in 1/10 K (z.B. 4 K = 40)
    dTaus = 20; // in 1/10 K (z.B. 2 K = 20)

    rbports.set(R,0); // Relais AUS
    sleep 10000; // Startverzögerung nach RESET (größer Pause in
'watch.error'!)

    loop
    {
        //----- Sensorzuweisung -----

        Tsp0 = temp.T[1]; // Sensor Speicher oben
        ThkR1 = temp.T[5]; // Sensor HK-Rücklauf

        //----- Bedingungen für Pumpe AUS -----

        if (Tsp0 <= TspMin or (Tsp0 - ThkR1) <= dTaus)
            and rbports.get(R)!=0 // Zustand von Relais auf EIN prüfen
            {
                // for i=0...<1 sleep onemin; // Ausschaltverzögerung bzw. Nachlauf
(z.B. 1 = 1 min)
                rbports.set(R,0); // Relais AUS
                for i=0...<1 sleep ONEMIN; // Mindest-Ausschaltdauer (z.B. 1 = 1 min)
            }

        else

            //----- Bedingungen für Pumpe EIN -----

            if watch.F0 == 0 and watch.F1 == 0 // Fehlerstatus prüfen
                and Tsp0 >= (TspMin + 30) // + Hysterese (z.B. 3 K = 30)
                and (Tsp0 - ThkR1) >= dTein
                and rbports.get(R)==0 // Zustand von Relais auf AUS prüfen
                {
                    // sleep 10000; // Einschaltverzögerung in Millisekunden (max. 32767
!)
                    rbports.set(R,-1); // Relais EIN
                    for i=0...<1 sleep ONEMIN; // Mindest-Einschaltdauer (z.B. 1 = 1 min)
                }

            if rbports.get(R)!=0 d=system.day(); // Tag speichern wenn Relais EIN

            //----- Pumpenkick wenn >= 23h AUS -----

            if rbports.get(R)==0 and system.day()!=d and system.hour()==23
            {
                // sleep 10000; // Einschaltverzögerung in Millisekunden (max. 32767
!)
                rbports.set(R,-1); // Relais EIN
            }
    }
}
```

```
        sleep 10000; // für Ventile evtl. anpassen
        d=system.day(); // Tag speichern für Relais EIN
        rports.set(R,0); // Relais AUS
        sleep 10000;
    }
}
}

/****
/* Ansteuerung Pufferspeicher-Ladepumpe
/* - Zur Pufferspeicher-Beladung durch Feststoffkessel, mit Speicher-
/*   Maximaltemp.-Begrenzung, Kesselmindesttemp.-Überwachung und
/*   Temperaturdifferenzregelung
/****
/* - Relais EIN wenn Kesseltemp. >= TkEin und Differenz zwischen
/*   Kesseltemp. und Speichertemp. unten >= dTein und Speichertemp.
/*   <= Speichermaximaltemp. - Hysterese (5 K). Fehlerstatus F0 und F1
/*   müssen = '0' sein.
/* - Relais Aus wenn Kesseltemp. <= TkAus oder Differenz zwischen
/*   Kesseltemp. und Speichertemp. unten <= dTaus oder Speichertemp.
/*   >= Speichermaximaltemp.
/****
/* WICHTIG:
/* - Folgende Einstellungen müssen im oberen Bereich kontrolliert und
/*   ggf. geändert werden:
/*     - Relais Nr.
/*     - Min/Max Temperaturen, Temperaturdifferenzen
/*     - Sensorzuweisung (am Anfang der 'loop'-Schleife)
/* - Es werden Mindest-Ein-/Ausschaltdauer festgelegt um z.B. bei
/*   Regelschwingungen Pumpe und Relais zu schonen (s. Hinweise im
/*   Quelltext). Voreinstellung = 1 Minute. Für Pausen < 1 min muss
/*   die 'for'-Schleife entfernt werden (max. 32767 Millisekunden!)
/* - Es können Ein- und Ausschalverzögerungen festgelegt werden
/*   (s. Hinweise Quelltext)
/* - Auf "Relais-Logik" achten (Schließer oder Öffner)
/* - Pumpenkick: Die Pumpe wird nach 23h Stillstand für eine fest-
/*   gelegte Dauer eingeschaltet (Verhindern des Festsetzens).
/*   (für Ventile ggf. Laufzeit anpassen)
/* - ACHTUNG: Wenn die EinschaltRoutine startet, ist die Abfrage der
/*   Auschaltbedingungen für die Mindest-Einschaltdauer
/*   blockiert (auch bei z.B. Übertemperatur)! Ist die Min-
/*   Einschaltdauer sehr groß, sollte die Übertemperatur-
/*   abschaltung über einen anderen thread direkt erfolgen.
/*   (siehe Modul 'watch' !)
/****
/* dTein = Einschalt-Temperaturdifferenz, min. dTaus + 2 K (20)!!!
/* dTaus = Ausschalt-Temperaturdifferenz (Wärmeverluste)
/* dTein = dTaus + Hysterese
/****
thread Feststoffkessel
{
    byte R, i, d;
    int Tsp0, TspU, TspMax, Tk, TkEin, TkAus, dTein, dTaus;

    R = 1; // Relais Nr., '0'-basiert! (Relais 2 = '1')

    TspMax = 900; // Speichermaximaltemp. in 1/10 °C (z.B. 90 °C = 900)
```

```
TkEin = 650; // Kesseltemp. für Ladepumpe EIN in 1/10 °C (z.B. 65 °C = 650)
TkAus = 550; // Kesseltemp. für Ladepumpe AUS in 1/10 °C (z.B. 55 °C = 550)
dTein = 40; // min. dTaus + 2 K !!!, in 1/10 K (z.B. 4 K = 40)
dTaus = 20; // in 1/10 K (z.B. 2 K = 20)

rbports.set(R,0); // Relais AUS
sleep 10000; // Startverzög. nach RESET (größer Pause in 'watch.error'!)

loop
{
    //----- Sensorzuweisung -----

    TspO = temp.T[1]; // Sensor Speicher oben
    TspU = temp.T[9]; // Sensor Speicher unten
    Tk = temp.T[13]; // Sensor Kessel

    //----- Bedingungen für Pumpe AUS -----

    if (TspO >= TspMax or Tk <= TkAus or (Tk - TspU) <= dTaus)
        and rbports.get(R)!=0 // Zustand von Relais auf EIN prüfen
    {
        // for i=0...<1 sleep ONEMIN; // Ausschaltverzögerung bzw. Nachlauf
        (z.B. 1 = 1 min)
        rbports.set(R,0); // Relais AUS
        for i=0...<1 sleep ONEMIN; // Mindest-Ausschaltdauer (z.B. 1 = 1 min)
    }

    else

    //----- Bedingungen für Pumpe EIN -----

    if watch.F0 == 0 and watch.F1 == 0 // auf Fehlerstatus prüfen
        and TspO <= TspMax - 50 // - Hysterese (z.B. 5 K = 50)
        and Tk >= TkEin
        and (Tk - TspU) >= dTein
        and rbports.get(R)==0 // Zustand von Relais auf AUS prüfen
    {
        // sleep 10000; // Einschaltverzög. in Millisekunden (max. 32767 !)
        rbports.set(R,-1); // Relais EIN
        for i=0...<1 sleep ONEMIN; // Mindest-Einschaltdauer (z.B. 1 = 1 min)
    }

    if rbports.get(R)!=0 d=system.day(); // Tag speichern wenn Relais EIN

    //----- Pumpenkick wenn >= 23h AUS -----

    if rbports.get(R)==0 and system.day()!=d and system.hour()==23
    {
        sleep 30000; // Einschaltverzögerung in Millisekunden (max. 32767 !)
        rbports.set(R,-1); // Relais EIN
        sleep 10000; // für Ventile evtl. anpassen
        d=system.day(); // Tag speichern für Relais EIN
        rbports.set(R,0); // Relais AUS
        sleep 10000;
    }
}
}
```

```

/*****
/* Ansteuerung Solarpumpe */
/* - Zur solaren Speicherbeladung, mit Speicher-Maximaltemp.-Begrenzung*/
/* Kollektor-Grenztemp.-Begrenzung und Zeitschaltuhr (diese muss ge- */
/* startet werden -> main-thread) */
/*****
/* - Pumpe EIN wenn Differenz zwischen Kollektortemp. und Speichertemp.*/
/* unten >= dTein und Speichertemp. <= Speichermx. - Hysterese (3 K)*/
/* und Kollektortemp. <= Kollektorgrenztemp. - Hysterese (15 K). */
/* Fehlerstatus F0 und F1 müssen = '0' sein. */
/* - Pumpe AUS wenn Differenz <= dTaus oder Speichertemp. >= Speicher- */
/* maximaltemp. oder Kollektortemp. >= Kollektorgrenztemp. */
/*****
/* WICHTIG: */
/* - Folgende Einstellungen müssen im oberen Bereich kontrolliert und */
/* ggf. geändert werden: */
/* - Relais Nr. */
/* - Min/Max Temperaturen, Temperaturdifferenzen */
/* - Sensorzuweisung (am Anfang der 'loop'-Schleife) */
/*
/* - Es werden Mindest-Ein-/Ausschaltdauer festgelegt um z.B. bei */
/* Regelschwingungen Pumpe und Relais zu schonen (s. Hinweise im */
/* Quelltext). Voreinstellung = 1 Minute. Für Pausen < 1 min muss */
/* die 'for'-Schleife entfernt werden (max. 32767 Millisekunden!) */
/* - Es können Ein- und Ausschalverzögerungen festgelegt werden */
/* (s. Hinweise Quelltext) */
/* - Auf "Relais-Logik" achten (Schließer oder Öffner) */
/* - Pumpenkick: Die Pumpe wird nach 23h Stillstand für eine fest- */
/* gelegte Dauer eingeschaltet (Verhindern des Festsetzens). */
/* (für Ventile ggf. Laufzeit anpassen) */
/*
/* - ACHTUNG: Wenn die Einschalttroutine startet, ist die Abfrage der */
/* Auschaltbedingungen für die Mindest-Einschaltdauer */
/* blockiert (auch bei z.B. Übertemperatur)! Ist die Min- */
/* Einschaltdauer sehr groß, sollte die Übertemperatur- */
/* abschaltung über einen anderen thread direkt erfolgen. */
/* (siehe Modul 'watch' !) */
/*****
/* dTein = Einschalt-Temperaturdifferenz, min. dTaus + 2 K (20)!!! */
/* dTaus = Ausschalt-Temperaturdifferenz (Wärmeverluste) */
/* dTein = dTaus + Hysterese */
/*****
thread Solarpumpe
{
    byte R, i, d, ZSU;
    int Tsp0, TspU, Tkol, TspMax, TkolGrz, dTein, dTaus;

    R = 0; // Relais Nr., '0'-basiert! (Relais 1 = '0')

    TspMax = 600; // Speichermaximaltemp. in 1/10 °C (z.B. 60 °C = 600)
    TkolGrz = 1300; // Kollektorgrenztemp. in 1/10 °C (z.B. 130 °C = 1300)
    dTein = 80; // min. dTaus + 2 K !!!, in 1/10 K (z.B. 8 K = 80)
    dTaus = 50; // in 1/10 K (z.B. 5 K = 50)

    rbports.set(R,0); // Relais AUS
    sleep 10000; // Startverzögerung nach RESET (größer Pause in
'watch.error'!)

```

```
loop
{
    //----- Sensorzuweisung -----

    ZSU = ZSU1; // hier Zeitschaltuhr (flag) angeben
    Tsp0 = temp.T[1]; // Sensor Speicher oben
    TspU = temp.T[9]; // Sensor Speicher unten
    Tkol = temp.T[5]; // Sensor Kollektor

    //----- Bedingungen für Pumpe AUS -----

    if (ZSU == 0 or Tsp0 >= TspMax or Tkol >= TkolGrz
        or (Tkol - TspU) <= dTaus)
        and rbports.get(R)!=0 // Zustand von Relais auf EIN prüfen
    {
        // for i=0...<1 sleep ONEMIN; // Ausschaltverzögerung bzw. Nachlauf
        (z.B. 1 = 1 min)
        rbports.set(R,0); // Relais AUS
        for i=0...<1 sleep ONEMIN; // Mindest-Ausschaltdauer (z.B. 1 = 1 min)
    }

    else

        //----- Bedingungen für Pumpe EIN -----

        if watch.F0 == 0 and watch.F1 == 0 // Fehlerstatus prüfen
            and ZSU == 1
            and Tsp0 <= (TspMax - 30) // - Hysterese (z.B. 3 K = 30)
            and Tkol <= (TkolGrz - 150) // - Hysterese (z.B. 15 K = 150)
            and (Tkol - TspU) >= dTein
            and rbports.get(R)==0 // Zustand von Relais auf AUS prüfen
        {
            // sleep 10000; // Einschaltverzögerung in Millisekunden (max. 32767
            !)
            rbports.set(R,-1); // Relais EIN
            for i=0...<1 sleep ONEMIN; // Mindest-Einschaltdauer (z.B. 1 = 1 min)
        }

        if rbports.get(R)!=0 d=system.day(); // Tag speichern wenn Relais EIN

        //----- Pumpenkick wenn >= 23h AUS -----

        if rbports.get(R)==0 and system.day()!=d and system.hour()==23
        {
            // sleep 10000; // Einschaltverzögerung in Millisekunden (max. 32767
            !)
            rbports.set(R,-1); // Relais EIN
            sleep 10000; // für Ventile evtl. anpassen
            d=system.day(); // Tag speichern für Relais EIN
            rbports.set(R,0); // Relais AUS
            sleep 10000;
        }
    }
}
```

```

/*****/
/* Ansteuerung Nachladepumpe */
/* - Zur Nachladung des WW-Speichers (Thermostatregelung) */
/* mit Zeitschaltuhr (diese muss gestartet -> main-thread) */
/*****/
/* - Pumpe EIN wenn Speichertemp. <= Einschalttemp. (je nach Zeitraum).*/
/* Fehlerstatus F0 und F1 müssen '0' sein. */
/* - Pumpe AUS wenn Speichertemp. >= Ausschalttemp. (je nach Zeitraum).*/
/*****/
/* WICHTIG: */
/* - Folgende Einstellungen müssen im oberen Bereich kontrolliert und */
/* ggf. geändert werden: */
/* - Relais Nr. */
/* - Min/Max Temperaturen, Temperaturdifferenzen */
/* - Sensorzuweisung (am Anfang der 'loop'-Schleife) */
/* */
/* - Es werden Mindest-Ein-/Ausschaltdauer festgelegt um z.B. bei */
/* Regelschwingungen Pumpe und Relais zu schonen (s. Hinweise im */
/* Quelltext). Voreinstellung = 1 Minute. Für Pausen < 1 min muss */
/* die 'for'-Schleife entfernt werden (max. 32767 Millisekunden!) */
/* - Es können Ein- und Ausschaltverzögerungen festgelegt werden */
/* (s. Hinweise Quelltext) */
/* - Auf "Relais-Logik" achten (Schließer oder Öffner) */
/* - Pumpenkick: Die Pumpe wird nach 23h Stillstand für eine fest- */
/* gelegte Dauer eingeschaltet (Verhindern des Festsetzens). */
/* (für Ventile ggf. Laufzeit anpassen) */
/* */
/* - ACHTUNG: Wenn die Einschaltroutine startet, ist die Abfrage der */
/* Ausschaltbedingungen für die Mindest-Einschaltdauer */
/* blockiert (auch bei z.B. Übertemperatur)! Ist die Min- */
/* Einschaltdauer sehr groß, sollte die Übertemperatur- */
/* abschaltung über einen anderen thread direkt erfolgen. */
/* (siehe Modul 'watch' !) */
/*****/
/* dTein = Einschalt-Temperaturdifferenz, min. dTaus + 2 K (20)!!! */
/* dTaus = Ausschalt-Temperaturdifferenz (Wärmeverluste) */
/* dTein = dTaus + Hysterese */
/*****/
thread Thermostat
{
    byte R, i, d, ZSU;
    int Tsp0, Tein, Taus, TeinMin, TausMin;

    R = 1; // Relais Nr., '0'-basiert! (Relais 2 = '1')

    Tein = 400; // EIN im Zeitfenster in 1/10 K (z.B. 40 °C = 400)
    Taus = 450; // AUS im Zeitfenster in 1/10 K (z.B. 45 °C = 450)
    TeinMin = 300; // EIN außerhalb des Zeitfenster in 1/10 K (z.B. 30 °C =
300)
    TausMin = 350; // AUS außerhalb des Zeitfenster in 1/10 K (z.B. 35 °C =
350)

    rbports.set(R,0); // Relais AUS
    sleep 10000; // Startverzögerung nach RESET (größer Pause in
'watch.error'!)

    loop
    {
        //----- Sensorzuweisung -----

```

```
ZSU = ZSU1; // hier Zeitschaltuhr (flag) angeben
TspO = temp.T[1]; // Sensor Speicher oben

//----- Bedingungen für Pumpe AUS -----

if ((ZSU == 1 and TspO >= Taus) or (ZSU == 0 and TspO >= TausMin))
    and rbports.get(R)!=0 // Zustand von Relais auf EIN prüfen
{
    // for i=0...<1 sleep ONEMIN; // Ausschaltverzögerung bzw. Nachlauf
    (z.B. 1 = 1 min)
    rbports.set(R,0); // Relais AUS
    for i=0...<1 sleep ONEMIN; // Mindest-Ausschaltdauer (z.B. 1 = 1 min)
}

else

//----- Bedingungen für Pumpe EIN -----

if watch.F0 == 0 and watch.F1 == 0 // Fehlerstatus prüfen
    and ((ZSU == 1 and TspO <= Tein) or (ZSU == 0 and TspO <= TeinMin))
    and rbports.get(R)==0 // Zustand von Relais auf AUS prüfen
{
    // sleep 10000; // Einschaltverzögerung in Millisekunden (max. 32767
!)
    rbports.set(R,-1); // Relais EIN
    for i=0...<1 sleep ONEMIN; // Mindest-Einschaltdauer (z.B. 1 = 1 min)
}

if rbports.get(R)!=0 d=system.day(); // Tag speichern wenn Relais EIN

//----- Pumpenkick wenn >= 23h AUS -----

if rbports.get(R)==0 and system.day()!=d and system.hour()==23
{
    sleep 30000; // Einschaltverzögerung in Millisekunden (max. 32767 !)
    rbports.set(R,-1); // Relais EIN
    sleep 10000; // für Ventile evtl. anpassen
    d=system.day(); // Tag speichern für Relais EIN
    rbports.set(R,0); // Relais AUS
    sleep 10000;
}
}
}

/*****
/* Haupt-thread zum starten der Relais-threads und der Zeitschaltuhr */
/*****
/* WICHTIG: */
/* - Zu startende Relais-threads anpassen, ggf weitere hinzufügen. */
/* darauf achten, dass nicht mit mehreren threads das gleiche Relais */
/* angesteuert wird! */
/* - Wird die Zeitschaltuhr nicht benötigt, kann die 'loop'-Schleife */
/* entfernt werden. Thread muss dann nach den Initialisierungen ge- */
/* stoppt werden (s.u.)! */
/* - Zeitschaltuhr ist nur minutengenau (ggf. ändern). */
/* - Nicht über den Tageswechsel schalten! Wenn über den Tageswechsel */
```

```
/* geschaltet werden soll, Schaltanweisung umkehren (z.B. im Tages- */
/* zeitfenster AUS statt EIN). */
/* - Für komplexere Zeitschaltaufgaben sollte ein eigenes Modul ver- */
/* wendet werden (bessere Übersichtlichkeit). */
/*****/
thread main
{
    byte min;
    int time;

    pcf8583.init(1,1,1,0,0); // Uhrenbaustein PCF8583 als Gangreserve der
    Systemuhr

    run watch.error; // Fehlerüberwachung starten

    //----- Relais-threads starten -----

    run Rueckklaufenhebung;
    run Feststoffkessel;
    // run Solarpumpe;
    // run Thermostat;

    //----- Zeitschaltuhr -----

    /* DEAKTIVIERT
    loop
    {
        min = system.minute();
        wait system.minute() != min; // Abfrage bei jedem Minutenwechsel
        time=system.hour()*100 + system.minute();

        if time >= 600 and time < 2200 // zwischen 6:00 und 22:00
            // nicht über Tageswechssel schalten
(s.o.)

            ZSU1 = 1; // EIN
        else
            ZSU1 = 0; // AUS
    }
    DEAKTIVIERT */

    ZSU1 = 1; // Zeitschaltuhr-Status EIN wenn Zeitschaltuhr deaktiviert
    halt; // thread anhalten (wenn Zeitschaltuhr deaktiviert)
}
```


7.3.4 Modul „display“ (LC-Display-Ausgabe)

```

/*****
/* Modul zur Ausgabe über externes Display
/* Autor : Malte Alpers (malte.alpers@gmx.de)
/* Version : 1.0
/* Datum : 01.07.2005
/* Geändert :
/*****
/* WICHTIG:
/* - Für 4x20 Displays. Bei anderen Formaten muss der Ablauf (Seiten-
/* wechsel) und ggf. Zeilenlängen + Positionen angepasst werden.
/* - Es wird zuerst ein Startbildschirm angezeigt (s. 'main'-thread).
/* Danach werden Datum, Zeit und ausgewählte Temperaturen angezeigt.
/* - Ausgaben bei Fehlern werden im 'main'-thread definiert.
/* - Ausgaben ggf. anpassen und weitere hinzufügen.
/*****

const wochentag[] = "So", "Mo", "Di", "Mi", "Do", "Fr", "Sa";

/*****
/* Standard Ausgaben
/* Die Anzeige ist in Seiten aufgeteilt, die jeweils für eine festge-
/* legte Zeit angezeigt werden. Datum und Zeit werden immer in der 1.
/* Zeile angezeigt.
/*****
/* WICHTIG:
/* - Für 4x20 Displays. Bei anderen Formaten muss der Ablauf (Seiten-
/* wechsel) und ggf. Zeilenlängen + Positionen angepasst werden.
/* - ggf. Ausgaben anpassen und weitere Seiten hinzufügen.
/* - Bei den Seiten darauf achten, dass immer alle Felder überschrie-
/* ben werden! Ist der Text an einer Position auf der Folgeseite
/* kürzer, bleiben die Zeichen der vorherigen Seite an den 'Leer-
/* stellen' stehen (ggf. Zeilen vor dem Seitenwechsel löschen).
/* - Zeilennummern ('line') beginnen mit 1. Bei Feldbezeichnern
/* ('goto') sind die Spalten '0'-basiert, z.B. (1,5) = 1. Zeile, 4.
/* Spalte.
/*****
thread ausgabe
{
    byte i, sec;
    string s;

    //----- Seite 1 -----

    for i=0...<10 // Anzeigedauer Seite 1 in Sekunden
    {
        sec = system.second();
        wait system.second() != sec;

        rblcd.line(1);
        s=wochentag[system.dow()] + " "; // Wochentag
        rblcd.line(1);
        rblcd.print(s);
        rblcd.date(5); // Datum
        rblcd.goto(1,15);
        rblcd.time(3); // Zeit (Stunden + Minuten)
    }
}
```

```
s="Tsp0 :"; // gleiche Länge aller Bezeichner (ggf. mit Leerzeichen
füllen)!
rblcd.line(2);
rblcd.print(s);
rblcd.zahl4n1(temp.T[1]); // Sensor
rblcd.put(0x20); // Space
rblcd.put(0xDF); // Grad
rblcd.put('C');
s="ThkRl:"; // gleiche Länge aller Bezeichner (ggf. mit Leerzeichen
füllen)!
rblcd.line(3);
rblcd.print(s);
rblcd.zahl4n1(temp.T[5]); // Sensor
rblcd.put(0x20); // Space
rblcd.put(0xDF); // Grad
rblcd.put('C');

// Uhr - Blinkereffekt
sleep 500;
rblcd.goto(1,17);
rblcd.put(0x20);
}
```

//----- Seite 2 -----

```
for i=0...<10 // Anzeigedauer Seite 1 in Sekunden
{
    sec = system.second();
    wait system.second() != sec;

    rblcd.line(1);
    s=wochentag[system.dow()] + " "; // Wochentag
    rblcd.line(1);
    rblcd.print(s);
    rblcd.date(5); // Datum
    rblcd.goto(1,15);
    rblcd.time(3); // Zeit (Stunden + Minuten)

    s="TspU :"; // gleiche Länge aller Bezeichner (ggf. mit Leerzeichen
füllen)!
    rblcd.line(2);
    rblcd.print(s);
    rblcd.zahl4n1(temp.T[9]); // Sensor
    rblcd.put(0x20); // Space
    rblcd.put(0xDF); // Grad
    rblcd.put('C');
    s="Tk   :"; // gleiche Länge aller Bezeichner (ggf. mit Leerzeichen
füllen)!
    rblcd.line(3);
    rblcd.print(s);
    rblcd.zahl4n1(temp.T[13]); // Sensor
    rblcd.put(0x20); // Space
    rblcd.put(0xDF); // Grad
    rblcd.put('C');

    // Uhr - Blinkereffekt
    sleep 500;
    rblcd.goto(1,17);
    rblcd.put(0x20);
}
}
```

```
/* ***** */
/* Startbildschirm und Fehlerausgaben */
/* Es wird zuerst der Startbildschirm für eine festgelegte Zeit ange- */
/* zeigt. Danach wird der 'ausgabe'-thread gestartet. Zusätzlich wird */
/* der Staus von Fehlervariablen überwacht, ggf. der 'ausgabe'-thread */
/* gestoppt und die Fehlermeldung(en) angezeigt, bis alle Fehler be- */
/* hoben sind. */
/* ***** */
/* WICHTIG: */
/* - ggf. Angaben für Variante und Version ändern (auch in 'terminal!') */
/* - ggf. weitere Fehlermeldungen hinzufügen. Zeilennummern anpassen! */
/* ***** */
thread main
{
    string s;
    rblcd.init();
    rblcd.clear();

    //----- Startbildschirm Anfang -----

    s="Betriebssystem";
    rblcd.line(1); rblcd.print(s);
    system.getOSInfo(s);
    rblcd.line(3); rblcd.print(s);
    sleep 5000; rblcd.clear();

    s="HEIZUNGSREGELUNG"; // ggf. ändern (auch in 'terminal!')

    rblcd.line(1); rblcd.print(s);

    s="Feststoff 2"; // ggf. ändern (auch in 'terminal!')

    rblcd.line(3); rblcd.print(s);

    s="Version: 1.0"; // ggf. ändern (auch in 'terminal!')

    rblcd.line(4); rblcd.print(s);
    sleep 8000; rblcd.clear();

    //----- Startbildschirm Ende -----

    run ausgabe;

    //----- Anzeige bei Fehlermeldung -----

    loop
    {
        if watch.F1 != 0 or watch.F2 != 0
        or watch.F10 != 0 or watch.F11 != 0 // ggf. weitere
        {
            halt ausgabe; // Ausgabe stoppen
            rblcd.clear();

            while watch.F1 != 0 or watch.F2 != 0
            or watch.F10 != 0 or watch.F11 != 0 // ggf. weitere
            {
                if watch.F1 != 0
                {
                    s="F1 : Fuehlerdefekt";
                    rblcd.line(1); // Zeile 1
                    rblcd.print(s);
                }
            }
        }
    }
}
```

```
    }  
    else rblcd.delline(1); // Zeile 1  
  
    if watch.F2 != 0  
    {  
        s="F2 : Uebertemperatur";  
        rblcd.line(2); // Zeile 2  
        rblcd.print(s);  
    }  
    else rblcd.delline(2); // Zeile 2  
  
    if watch.F10 != 0  
    {  
        s="F10: dT zu hoch";  
        rblcd.line(3); // Zeile 3  
        rblcd.print(s);  
    }  
    else rblcd.delline(3); // Zeile 3  
  
    if watch.F11 != 0  
    {  
        s="F11: Nachtumwaelzung";  
        rblcd.line(4); // Zeile 4  
        rblcd.print(s);  
    }  
    else rblcd.delline(4); // Zeile 4  
    }  
    rblcd.clear();  
    resume ausgabe; // Ausgabe wieder starten  
    }  
}
```

7.3.5 Modul „terminal“ („Wartung“ über PC)

```

/*****
/* Modul zur Wartung über PC (HyperTerminal) */
/* Autor : Malte Alpers (malte.alpers@gmx.de) */
/* Version : 1.0 */
/* Datum : 01.07.2005 */
/* Geändert : */
/*****
/* WICHTIG: */
/* - Im 'main'-thread muss die Verbindungsgeschwindigkeit der seriel- */
/* len Schnittstelle (HCOM) ggf. angepasst werden. */
/* - In 'menu' ggf. Variante und Version ändern. */
/* - In der 'ausgabe'-function ggf. weitere Ausgaben hinzufügen. */
/* - In der 'logger'-function ggf. weitere aufzuzeichnende Variablen */
/* hinzufügen. */
/* - Für die korrekte Funktion des Handbetriebs muss in allen, Relais */
/* schaltenden, threads und functions auf 'F0' geprüft werden! */
/* (siehe 'Relais'-threads) */
/* Werden mehr als die 3 Relais des Reglerboards verwendet, müssen */
/* diese hinzugefügt werden. */
/* - Hinweis: ESC drücken wenn das Menü im Hyperterminal nicht ange- */
/* zeigt wird. */
/*****

const wochentag[] = "So", "Mo", "Di", "Mi", "Do", "Fr", "Sa";
byte r;

/*****
/* Hauptmenü, mit Angaben über das Betriebssystem und den Funktionen: */
/* 1 = Datum und Uhrzeit stellen */
/* 2 = Ausgabe */
/* 3 = Sensorkontrolle */
/* 4 = Datenaufzeichnung */
/* 5 = Handbetrieb */
/*****
/* WICHTIG: */
/* - ggf. Angaben für Variante und Version ändern (auch in 'display!') */
/*****
function menu()
{
    string s;
    hwcom.clr();

    s=" -----";
    hwcom.print(s);wait hwcom.ready();
    s="-----";
    hwcom.print(s);
    hwcom.ret();
    s=" Betriebssystem : ";
    hwcom.print(s);wait hwcom.ready();
    system.getOSInfo(s);
    hwcom.print(s);
    hwcom.ret();

    s=" Heizungsregelung: Feststoff 2"; // ggf. ändern (auch in 'display!')

    hwcom.print(s);wait hwcom.ready();

```

```
s=", v1.0"; // ggf. ändern (auch in 'display'!)

hwcom.print(s);
hwcom.ret();
s=" -----";
hwcom.print(s);wait hwcom.ready();
s="-----";
hwcom.print(s);
hwcom.ret();
hwcom.ret();

s=" Hauptmen"+129;
hwcom.print(s);
hwcom.ret();
s=" -----";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
s=" 1 = Datum und Uhrzeit stellen";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
s=" 2 = Ausgabe";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
s=" 3 = Sensorkontrolle";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
s=" 4 = Datenaufzeichnung";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
s=" 5 = Handbetrieb";
hwcom.print(s);
}

/*****/
/* 1 = Datum und Uhrzeit stellen */
/*****/
function datum()
{
    int y;
    byte d, mth, h, min, get;
    string s;
    r=1;
    hwcom.clr();
    hwcom.ret();
    s=" Tag (1- bzw. 2-stellig): ";
    hwcom.print(s);
    d=hwcom.input(1);
    hwcom.ret();
    s=" Monat (1- bzw. 2-stellig): ";
    hwcom.print(s);
    mth=hwcom.input(1);
    hwcom.ret();
    s=" Jahr (4-stellig): ";
    hwcom.print(s);
```

```
y=hwcom.input(1);
hwcom.ret();
s=" Stunde (1- bzw. 2-stellig): ";
hwcom.print(s);
h=hwcom.input(1);
hwcom.ret();
s=" Minute (1- bzw. 2-stellig): ";
hwcom.print(s);
min=hwcom.input(1);
hwcom.ret();
hwcom.ret();
s=" ";
str.putint(s,d);
s=s+".";
str.putint(s,mth);
s=s+".";
str.putint(s,y);
s=s+" / ";
str.putint(s,h);
if min < 10 s=s+":0";
else s=s+": ";
str.putint(s,min);
hwcom.print(s);wait hwcom.ready();
s=" (o)k / (a)bbrechen ?";
hwcom.print(s);

loop // Schleife notwendig falls andere Taste gedrückt wird
{
    wait hwcom.rxd();
    get=hwcom.get();
    if get=='o' or get=='O'
    {
        system.setdate(y,mth,d);
        system.settime(h,min,0);
        pcf8583.syncpcf(); // Uhrenbaustein PCF8583 synchronisieren
        r=0;
        return;
    }
    else
    if get=='a' or get=='A'
    {
        r=0;
        return;
    }
}
}

/*****
/* 2 = Ausgabe
/* Anzeigen der Informationen über Datum, Uhrzeit, Temperaturen und
/* Schaltzustände der Relais.
/* - ggf. anpassen und weitere Ausgaben hinzufügen (siehe Quelltext)
/* - Bei den Temperaturengaben müssen jeweils 2 Zuweisungen (T[i])
/* angepasst werden !!!
*****/
function ausgabe()
{
    string s;
    r=1;
```

```
loop
{
    if hwcom.rxd() and hwcom.get()==27 // mit ESC beenden
    {
        r=0;
        return;
    }
    hwcom.clr();
    hwcom.ret();
    // Fehlabfrage bei Stunden- bzw. Tageswechsel abfangen:
    if system.second()==59 and system.minute()==59 wait system.second()==0;
    s=" ";
    s=s+wochentag[system.dow()]+ " ";
    str.putintf(s,system.day(),2);
    s=s+".";
    str.putintf(s,system.month(),2);
    s=s+".";
    str.putintf(s,system.year(),2);
    s=s+" / ";
    hwcom.print(s);
    system.TIME Zeit;
    system.gettime(Zeit);
    if Zeit.hour < 10 s="0";
    else s="";
    str.putint(s,Zeit.hour);
    if Zeit.minute < 10 s=s+":0";
    else s=s+": ";
    str.putint(s,Zeit.minute);
    if Zeit.second < 10 s=s+":0";
    else s=s+": ";
    str.putint(s,Zeit.second);
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();

    //----- Anfang angezeigte Temperaturen, Relaiszustände usw. -----

    s=" Tsp0 (1): ";
    str.putint(s,temp.T[1]/10); // T[1]
    s=s+", "+((math.abs(temp.T[1])%10)+0x30)+" "+248+"C"; // T[1]
    hwcom.print(s);
    hwcom.ret();
    s=" ThkR1 (5): ";
    str.putint(s,temp.T[5]/10); // T[5]
    s=s+", "+((math.abs(temp.T[5])%10)+0x30)+" "+248+"C"; // T[5]
    hwcom.print(s);
    hwcom.ret();
    s=" TspU (9): ";
    str.putint(s,temp.T[9]/10); // T[9]
    s=s+", "+((math.abs(temp.T[9])%10)+0x30)+" "+248+"C"; // T[9]
    hwcom.print(s);
    hwcom.ret();
    s=" Tk (13): ";
    str.putint(s,temp.T[13]/10); // T[13]
    s=s+", "+((math.abs(temp.T[13])%10)+0x30)+" "+248+"C"; // T[13]
    hwcom.print(s);
    hwcom.ret();

    s=" Relais 1: ";
    if rbports.get(0)==0 s=s+"AUS"; // 'rbports.get(0)' = Relais 1
    else s=s+"EIN";
```



```
hwcom.print(s);
hwcom.ret();
s=" Relais 2: ";
if rbports.get(1)==0 s=s+"AUS"; // 'rbports.get(1)' = Relais 2
else s=s+"EIN";
hwcom.print(s);
hwcom.ret();
s=" Relais 3: ";
if rbports.get(2)==0 s=s+"AUS"; // 'rbports.get(2)' = Relais 3
else s=s+"EIN";
hwcom.print(s);
hwcom.ret();

s=" F1 : ";
str.putint(s,watch.F1);
hwcom.print(s);
hwcom.ret();
s=" F2 : ";
str.putint(s,watch.F2);
hwcom.print(s);
hwcom.ret();
s=" F10: ";
str.putint(s,watch.F10);
hwcom.print(s);
hwcom.ret();
s=" F11: ";
str.putint(s,watch.F11);
hwcom.print(s);

//----- Ende angezeigte Temperaturen, Relaiszustände usw. -----

hwcom.ret();
hwcom.ret();
s=" -----";
hwcom.print(s);
hwcom.ret();
s=" zur"+129+"ck mit ESC";
hwcom.print(s);
sleep 500;
}
}

/*****/
/* 3 = Sensorkontrolle */
/* Zur Anzeige der Sensorzustände, z.B. Leitungsbruch oder Kurzschluss*/
/*****/
function sensoren()
{
  byte i;
  int x;
  string s;
  r=1;
  loop
  {
    if hwcom.rxd() and hwcom.get()==27 // mit ESC beenden
    {
      r=0;
      return;
    }
  }
}
```

```
hwcom.clr();
hwcom.ret();

for i=1...16 // T1...T16
{
    s=" S";
    str.putint(s,i);
    x=temp.T[i];
    if i < 10 s=s+" : ";
    else s=s+": ";
    if x == 9999 s=s+"Bruch / n.a.";
    else
    if x == -999 s=s+"Kurzschluss";
    else
    str.putint(s,x);
    hwcom.print(s);
    hwcom.ret();
}
hwcom.ret();
s=" -----";
hwcom.print(s);
hwcom.ret();
s=" zur"+129+"ck mit ESC";
hwcom.print(s);
sleep 2000;
}
}

/*****
/* 4 = Datenaufzeichnung
/* Zur Aufzeichnung der Temperaturen und Schaltzustände mit PC
/* (HyperTerminal)
/* - ggf. anpassen und weitere aufzuzeichnende Variablen hinzufügen
/* (siehe Quelltext)
*****/
function logger()
{
    byte get, modus, intervall, i, timer;
    string s;
    r=1;
    hwcom.clr();
    hwcom.ret();
    s="Aufzeichnungstaktung: ";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();
    s="1 = Sekunden";
    hwcom.print(s);
    hwcom.ret();
    s="2 = Minuten";
    hwcom.print(s);
    wait hwcom.rxd();
    get=hwcom.get();
    if get=='1' modus=1;
    if get=='2' modus=2;
    hwcom.clr();
    hwcom.ret();
    s="Intervall: ";
    hwcom.print(s);
```

```
intervall=hwcom.input(1);
hwcom.clr();
s="\Text aufzeichnen...\ w";
hwcom.print(s);wait hwcom.ready();
s=132+"hlen und eine Taste dr";
hwcom.print(s);wait hwcom.ready();
s=129+"cken";
hwcom.print(s);
hwcom.ret();
s="Aufzeichnung beenden mit ESC";
hwcom.print(s);
hwcom.ret();
hwcom.ret();
wait hwcom.rxd();
hwcom.flush();

//----- Spaltenbezeichner, ggf. erweitern (max. 30 Zeichen pro string!)

s="Datum;Zeit;T1;T5;T9;T13;R1;R2;";
hwcom.print(s);wait hwcom.ready(); // aktivieren bei 2 strings !
s="R3"; // 2. string

hwcom.print(s);
hwcom.ret();
loop
{
    if hwcom.rxd() and hwcom.get()==27 // mit ESC beenden
    {
        r=0;
        break;
    }
    if modus==1
    {
        for i=0...<intervall
        {
            timer = system.second();
            wait system.second() != timer;
        }
    }
    else
        for i=0...<intervall
        {
            timer = system.minute();
            wait system.minute() != timer;
        }
    s=wochentag[system.dow()]+ " ";
    str.putintf(s,system.day(),2);
    s=s+ ".";
    str.putintf(s,system.month(),2);
    s=s+ " ";
    hwcom.print(s);
    system.TIME Zeit;
    system.gettime(Zeit);
    if Zeit.hour < 10 s="0";
    else s="";
    str.putint(s,Zeit.hour);
    if Zeit.minute < 10 s=s+":0";
    else s=s+ ":";
    str.putint(s,Zeit.minute);
    if modus==1
```

```
{
    if Zeit.second < 10 s=s+":0";
    else s=s+": ";
    str.putint(s,Zeit.second);
}
s=s+": ";
hwcom.print(s);

//----- Anfang aufzuzeichnende Temperaturen, Relaiszustände usw. -----

s="";
str.putint(s,temp.T[1]/10); // T[1]
s=s+", "+((math.abs(temp.T[1])%10)+0x30); // T[1]
s=s+": ";
hwcom.print(s);wait hwcom.ready();
s="";
str.putint(s,temp.T[5]/10); // T[5]
s=s+", "+((math.abs(temp.T[5])%10)+0x30); // T[5]
s=s+": ";
hwcom.print(s);wait hwcom.ready();
s="";
str.putint(s,temp.T[9]/10); // T[9]
s=s+", "+((math.abs(temp.T[9])%10)+0x30); // T[9]
s=s+": ";
hwcom.print(s);wait hwcom.ready();
s="";
str.putint(s,temp.T[13]/10); // T[13]
s=s+", "+((math.abs(temp.T[13])%10)+0x30); // T[13]
s=s+": ";
hwcom.print(s);wait hwcom.ready();

if rbports.get(0)==0 s="AUS"; // 'rbports.get(0)' = Relais 1
else s="EIN";
s=s+": ";
hwcom.print(s);wait hwcom.ready();
if rbports.get(1)==0 s="AUS"; // 'rbports.get(1)' = Relais 2
else s="EIN";
s=s+": ";
hwcom.print(s);wait hwcom.ready();
if rbports.get(2)==0 s="AUS"; // 'rbports.get(2)' = Relais 3
else s="EIN";
hwcom.print(s);

//----- Ende aufzuzeichnende Temperaturen, Relaiszustände usw. -----

hwcom.ret();
}
hwcom.ret();
s="\nText aufzeichnen...\n beenden ";
hwcom.print(s);wait hwcom.ready();
s="und eine Taste dr";
hwcom.print(s);wait hwcom.ready();
s=129+"cken";
hwcom.print(s);
wait hwcom.rxd();
}
```

```
/* ***** */
/* 5 = Handbetrieb */
/* Um die Anlage von Automatik auf Handbetrieb umzustellen. */
/* Bietet die Möglichkeit die Relais von Hand zuschalten. */
/* ***** */
/* WICHTIG: */
/* - Die Relaiszustände werden nicht laufend abgefragt (nur nach Um- )*/
/* schalten. Wird das Relais z.B. durch Übertemperaturüberwachung */
/* ausgeschaltet, bleibt die Anzeige auf EIN. */
/* - Für die korrekte Funktion des Handbetriebs muss in allen, Relais */
/* schaltenden, threads und functions auf 'F0' geprüft werden! */
/* (siehe 'Relais'-threads) */
/* - Werden mehr als die 3 Relais des Reglerboards verwendet, müssen */
/* diese hinzugefügt werden (siehe Hinweise im Quelltext). */
/* - Wird eine Speicherladepumpe angesteuert, muss bei der Relaisum- */
/* schaltung Fehler 'F2' (Übertemp.) auf '0' geprüft werden! */
/* (im Quelltext unter 'Relais manuell ansteuern') */
/* Das Relais kann dann bei Übertemp. auch nicht im Handbetrieb */
/* eingeschaltet werden. */
/* z.B. Ladepumpe an Relais 2: */
/* */
/* if get=='2' and watch.F2==0 rbports.toggle(1);sleep 300; */
/* */
/* ***** */
function manuell()
{
    byte get;
    string s;
    r=1;
    hwcom.clr();
    hwcom.ret();
    s=" Alle Relais-Threads werden ";
    hwcom.print(s);wait hwcom.ready();
    s="angehalten!";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();
    s=" (o)k / (a)bbrechen ?";
    hwcom.print(s);
    wait hwcom.rxd();
    get=hwcom.get();
    if get=='o' or get=='O'
    {
        watch.F0=1; // Handbetriebstaus setzen

        //----- Hier alle Relais auf 'AUS' schalten! -----
        sleep 300;
        rbports.set(0,0);
        sleep 300;
        rbports.set(1,0);
        sleep 300;
        rbports.set(2,0);
    }
    else
    {
        r=0;
        return;
    }
    hwcom.clr();
    hwcom.ret();
}
```

```
loop
{
    hwcom.clr();
    hwcom.ret();
    s=" Handbetrieb";
    hwcom.print(s);
    hwcom.ret();
    s=" -----";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();
    s=" Durch Dr"+129+"cken der Nr. wird ";
    hwcom.print(s);wait hwcom.ready();
    s="das Relais umgeschaltet";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();

    //----- Relaisstatus abfragen -----

    s=" 1 = Relais 1: ";
    if rbports.get(0)==0 s=s+"AUS"; // 'rbports.get(0)' = Relais 1
    else s=s+"EIN";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();
    s=" 2 = Relais 2: ";
    if rbports.get(1)==0 s=s+"AUS"; // 'rbports.get(1)' = Relais 2
    else s=s+"EIN";
    hwcom.print(s);
    hwcom.ret();
    hwcom.ret();
    s=" 3 = Relais 3: ";
    if rbports.get(2)==0 s=s+"AUS"; // 'rbports.get(2)' = Relais 3
    else s=s+"EIN";
    hwcom.print(s);
    hwcom.ret();

    hwcom.ret();
    s=" -----";
    hwcom.print(s);
    hwcom.ret();
    s=" zur"+129+"ck mit ESC (Automatik";
    hwcom.print(s);wait hwcom.ready();
    s="betrieb)";
    hwcom.print(s);

    //----- Relais manuell ansteuern -----

    wait hwcom.rxd();
    get=hwcom.get();

    if get=='1' rbports.toggle(0);sleep 300; // Zustand von Relais 1
umschalten

    // Zustand von Relais 2 umschalten (F2 muss '0' sein)
    if get=='2' and watch.F2==0 rbports.toggle(1);sleep 300;

    if get=='3' rbports.toggle(2);sleep 300; // Zustand von Relais 3
umschalten
```

```
if get==27 // mit ESC beenden
{
    hwcom.clr();
    hwcom.ret();
    s=" bitte warten...";
    hwcom.print(s);

    //----- Hier alle Relais auf 'AUS' schalten! -----
    sleep 300;
    rbports.set(0,0);
    sleep 300;
    rbports.set(1,0);
    sleep 300;
    rbports.set(2,0);

    watch.F0=0; // Handbetriebstaus zurücksetzen

    sleep 10000;
    r=0;
    return;
}
}

//-----
thread main
//-----
{
    byte get;
    hwcom.init();
    hwcom.setspeed(8); // 57.600 Baud
    hwcom.put(12);

    loop
    {
        menu();
        wait hwcom.rxd();
        get=hwcom.get();
        if get=='1' datum(); wait r!=1;
        if get=='2' ausgabe(); wait r!=1;
        if get=='3' sensoren(); wait r!=1;
        if get=='4' logger(); wait r!=1;
        if get=='5' manuell(); wait r!=1;
    }
}
```

7.4 Bedienungsanleitung - Modul „terminal“

Die Ausgabe erfolgt über einen PC über ein Terminal-Programm (z.B. HyperTerminal). Es muss die gleiche Schnittstellen-Geschwindigkeit wie im „terminal“-Modul eingestellt werden. Wird das HyperTerminal gestartet erscheint das Hauptmenü. Hier können durch drücken der Tasten 1...5 die entsprechenden Funktionen aufgerufen werden. Erscheint das Hauptmenü nicht im HyperTerminal, ESC-Taste drücken!

```
Hauptmenü
-----

1 = Datum und Uhrzeit stellen

2 = Ausgabe

3 = Sensorkontrolle

4 = Datenaufzeichnung

5 = Handbetrieb
```

Datum und Uhrzeit stellen (,1' drücken)

Um über das HyperTerminal das Datum und die Uhrzeit einzustellen, werden nach einander folgende Angaben abgefragt:

```
Tag (1- bzw. 2-stellig):      10
Monat (1- bzw. 2-stellig):    6
Jahr (4-stellig):             2005
Stunde (1- bzw. 2-stellig):   12
Minute (1- bzw. 2-stellig):   20
```

Jede Eingabe ist mit ENTER zu bestätigen!

```
10.6.2005 / 12:20 (o)k / (a)bbrechen?
```

Sind alle Werte korrekt eingeben, "o" für o.k. drücken um ins Hauptmenü zurückzukehren.

Wichtig: Die Eingaben können nicht mit der „BackSpace“-Taste korrigiert werden! Bei einer Falscheingabe mit ENTER bestätigen, bis die Abfrage „ok / abbrechen“ erscheint. Dann "a" drücken um wieder ins Hauptmenü zurückzukehren. Erneut die 1 drücken und das Datum und die Uhrzeit erneut eingeben.

Anzeige der Ausgabedaten (,2' drücken)

Alle wichtigen Informationen über den momentanen Zustand der Heizungsanlage werden im HyperTerminal angezeigt und laufend aktualisiert.

```
Fr 10.06.2005 / 12:20:30      Datum / Uhrzeit

TspO (1): 36,7 °C             Temp. Warmwasser-Speicher Oben (*)
Tkol (5): 34,2 °C             Temp. Solarkollektor (*)
TspU (9): 30,6 °C             Temp. Warmwasser-Speicher Unten(*)
T13      : 999,9 °C           Sensor unterbrochen, nicht
                              angeschlossen!
Relais 1: AUS
Relais 2: AUS
Relais 3: AUS
F1 : 1                        Status 1: Fehler (**)
F2 : 0                        Status 0: kein Fehler
F10: 0
F11: 0
-----
zurück mit ESC
```

* Sensor-Nummer bzw. Sensorport

** Fehlererklärung: F1 : Fühlerdefekt
F2 : Übertemperatur Speicher 1
F10: Solarüberwachung „dT zu hoch“
F11: Solarüberwachung „Nachtumwälzung“

Sensorkontrolle (,3' drücken)

Der Zustand der Sensoren wird im HyperTerminal angezeigt und laufend aktualisiert.

```
S1: Kurzschluss
S2: 95
S3: 95                        Sensorports S1 ... S16
S4: 95                        zeigen die aktuelle Temperatur an
S5: 640                        (in 1/10 °C, z. B. S5: 640 = 64,0 °C).
S6: 95
S7: 95
S8: 95
S9: Bruch / n.a.              Sensor unterbrochen oder nicht
S10: 95                        angeschlossen!
S11: 95
S12: 95
S13: Bruch / n.a.              Sensor unterbrochen oder nicht
S14: 95                        angeschlossen!
S15: 95
S16: 95
-----
zurück mit ESC
```

Wichtig: Die Sensoreingänge der nicht bestückten Messadapter zeigen einen undefinierten Wert an (z.B. 95), wenn die Messadaptersteckplätze nicht auf Masse gelegt sind (vgl. Kap. 3.4)

Datenaufzeichnung (,4' drücken)

Aufzeichnungstaktung:	Datenerfassung
1 = Sekunden	im Sekunden-Takt
2 = Minuten	im Minuten-Takt

Es ist das gewünschte Intervall anzugeben, z.B. ,10' für eine Aufzeichnung im 10 Sekunden-Takt.

"*Text aufzeichnen...*" wählen und eine Taste drücken.

Beenden mit ESC.

In der Menüleiste des HyperTerminal „Übertragung“ wählen, dann auf „*Text aufzeichnen...*“ klicken. Es erscheint ein Dialogfenster in dem der Dateiordner für die Aufzeichnung angegeben werden kann. Auf „*Starten*“ klicken um die Aufzeichnung zu starten und im HyperTerminal eine beliebige Taste drücken, die Aufzeichnung beginnt.

Um die Aufzeichnung zu beenden die ESC-Taste drücken. In der Menüleiste wieder „Übertragung“, „*Text aufzeichnen...*“ und „*Beenden*“ wählen. Im HyperTerminal eine beliebige Taste drücken, die Aufzeichnung ist beendet.

Die aufgezeichneten Daten können mit Hilfe eines Tabellenkalkulationsprogramms bearbeitet werden.

Regelung auf Handbetrieb stellen (,5' drücken)

Alle Relais-Threads werden angehalten!

(o)k / (a)bbrechen ?

"o" drücken, um in den Handbetrieb zu wechseln.

Handbetrieb

Durch Drücken der Nr. wird das Relais umgeschaltet

1 = Relais 1: AUS

2 = Relais 2: AUS

3 = Relais 3: AUS

zurück mit ESC (Automatikbetrieb)

Durch drücken der Tasten 1...3 kann das entsprechende Relais beliebig EIN / AUS geschaltet werden. Durch drücken der ESC-Taste wechselt die Regelung wieder in den Automatikbetrieb.

Bei einer Übertemperatur des Speichers (Fehler F2) kann das Relais für die Ladepumpe nicht von Hand eingeschaltet werden!

Wichtig: Die Anzeige wird nur nach einer Eingabe aktualisiert (nicht laufend)! Ein Ausschalten des Relais bei z.B. Speicherübertemperatur wird dann nicht angezeigt.

8 Quellen

Bücher

- [1] Remmers, K.-H.: Große Solaranlagen. Die Solarpraxis (Hrsg.), Berlin. Uranus, Wien. 1999
- [2] Kainka, B.; Helbig, A.: Messen, Steuern und Regeln mit C-Control II. Franzis. 2002

Zeitschriften

- [3] Bachmann, S.; Drück, H.; Müller-Steinhagen, H.: Bauformen von Kombispeichern für Ein- und Zweifamilienhäuser. Erneuerbare Energien 3 (2004) 50-52
- [4] Rummel, A.: Qualitätslücken auf der Spur. Sonne Wind & Wärme 10 (2004) 26-28
- [5] Scheuren, J.; Pujiula, F.; Eisenmann, W.; Böhle, W.; Hafner, B.: Die Suche nach dem optimalen Volumenstrom in thermischen Solaranlagen. Erneuerbare Energien 10 (2004) 50-53
- [6] Krause, T.: Auslegungshinweise und Testverfahren für Solare Kombianlagen zur Brauchwassererwärmung und Heizungsunterstützung. Erneuerbare Energien 2 (2004) 55-57

Dissertationen

- [7] Drück, H.: Der Speicher - das Herz der Kombianlage. 1999. Universität Stuttgart, Institut für Thermodynamik und Wärmetechnik (ITW)
- [8] Streicher, E.; Heidemann, W.; Müller-Steinhagen, H.: Einfluss von Systemertrag und Pumpenlaufzeit auf die energetische Amortisationszeit von thermischen Solaranlagen. 2003. Universität Stuttgart, Institut für Thermodynamik und Wärmetechnik (ITW)

Weitere Quellen

Planungsunterlagen der Heizungssystemanbieter (Auswahl):

- Planungsunterlage Festbrennstoff-Heizkessel Logano. Buderus
- Planungsunterlage Solartechnik Logasol. Buderus
- Planungsanleitung Vitolog Festbrennstoffkessel. Viessmann
- Planungsanleitung Vitosol (Solartechnik). Viessmann
- Planungsinformation Solar. Vaillant

Linkliste

- CONRAD ELECTRONIC. <http://www1.conrad.de>
- CC-TOOLS, Hardware für C-Control. <http://cctools.hs-control.de>
- CC2net, Supportseite für die C-Control II. <http://www.cc2net.de>