



**MNI-Fonds für Unterrichts- und Schulentwicklung
S 6 „Anwendungsorientierung und Berufsbildung“**

AUTOMATISIERUNG EINER FERMENTATIONSANLAGE

**SELBST- UND FREMDBEWERTUNG DURCH SCHÜLER IN EINER PROJEKT-
ARBEIT ZUR WEITERENTWICKLUNG DES NATURWISSENSCHAFTLICH-
TECHNISCHEN UNTERRICHTS AN HTLS**

Dipl.-Ing. Wolfgang Bernhofer

**Schüler der 5HI der Abteilung Chemie-Informatik
HBLVA für chemische Industrie
Wien 17, Rosensteingasse 79**

Wien im Juli 2005

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS	2
ABSTRACT	5
1 EINLEITUNG	6
1.1 Leistungsbewertung.....	6
1.1.1 Voraussetzungen	6
1.1.2 Selbstbestimmung und Leistungserziehung.....	6
1.2 Naturwissenschaftlicher Unterricht.....	7
2 AUFGABENSTELLUNG	8
3 METHODEN UND DURCHFÜHRUNG	9
3.1 METHODEN	9
3.1.1 Projektmethode.....	9
3.1.2 Leistungsbewertung.....	9
3.1.3 Firmenähnliche Struktur	9
3.1.4 Pflichtenheft	9
3.1.5 Bewerbungsverfahren.....	11
3.2 DURCHFÜHRUNG	11
3.2.1 Projekt.....	11
3.2.2 Firmenstruktur.....	13
3.2.3 Ausschreibungs- und Bewerbungsverfahren	13
3.2.4 Präsentationen.....	14
3.2.5 Selbst- und Fremdbewertung.....	14
3.2.6 Schülerbefragungen.....	15
4 ERGEBNISSE	17
4.1 Gesamtergebnis.....	17
4.2 Selbstbeobachtungen (wie ich den Unterricht gesehen habe).....	17
4.3 Schülerbefragungen.....	18
4.3.1 Der geschlossene Schülerfragebogen	18
4.3.2 Der offene Schülerfragebogen	18
4.3.3 Interviews.....	18
4.4 Selbst- und Fremdbewertung.....	19

4.5	Beurteilung.....	19
5	DISKUSSION UND AUSBLICK	20
5.1	DISKUSSION.....	20
5.2	AUSBLICK	20
6	LITERATUR.....	22
7	ANHANG - DER GESCHLOSSENE SCHÜLERFRAGEBOGEN.....	23
8	ANHANG - TECHNISCHE BESCHREIBUNG DES PROJEKTS	24
8.1	Das Steuerprogramm.....	24
8.1.1	Anforderungen an das Steuerprogramm.....	24
8.2	Die Benutzeroberfläche.....	24
8.2.1	Die Statusleiste	25
8.2.2	Die Buttonleiste	25
8.2.3	Das Hauptfenster	25
8.3	Das Design der Benutzeroberfläche	25
8.3.1	Programmtechnische Implementierung der Buttonleiste.....	26
8.3.2	Kommunikation mit dem Mikrocontroller	27
8.3.3	Implementierung der Queues.....	28
8.4	Die pH-Regelung	30
8.4.1	Ermittlung des pH-Wertes über ein externes pH-Meter.....	30
8.4.2	Der PID Regler.....	31
8.5	Die Temperaturregelung	33
8.5.1	Erstellung einer „Debug“ Liste.....	34
8.6	Kommunikation der einzelnen Subsysteme untereinander	35
8.6.1	Starten der Subsysteme.....	35
8.7	Der Mikrocontroller.....	36
8.7.1	Anforderungen an den Mikrocontroller	36
8.7.2	Programmierung	37
8.7.3	Variablendeklaration	38
8.7.4	Interrupt Handling.....	39
8.7.5	String Conversion	41
8.7.6	Serielle Kommunikation	44
8.7.7	Potentiometer.....	49
8.7.8	Temperaturmesszelle.....	51

8.8	Die Elektronik.....	52
8.8.1	Das Mikroprozessor-Board	52
8.8.2	Schaltplan Mikroprozessor und Taktgenerator.....	53
8.8.3	Gerätemodifikationen	56
8.8.4	Datenblätter	56
	Datasheet TCN75	57
	Datasheet TCN75	58
	MAX-232	59
	TC74 Datasheet.....	60
	Optokoppler PC847.....	61
	MIKROPROZESSOR SOURCE CODE:	62

ABSTRACT

*In dieser Arbeit wird ein Projekt beschrieben, in dem das gesamte Unterrichtskonzept für eine Klasse so umgestellt wurde, dass mehrere Ideen moderner Lehr- und Lernmethoden wie **selbst bestimmtes Lernen** nach der **Projektmethode**, **Bewerbungs- und Aufnahmeverfahren** für bestimmte Aufgabenbereiche, **gemeinschaftliche Festlegung der Ziele** in einem **Pflichtenheft** und **Selbst- und Fremdbewertung** unter Einbeziehung der Teammitglieder, gleichzeitig umgesetzt werden konnten.*

Eine Klasse von 15 Schülern der 13. Schulstufe im Alter von 18-20 Jahren sollte im Rahmen von selbst bestimmtem Lernen nach der Projektmethode eine computergesteuerte Fermentationsanlage aufbauen. Dieses ungewöhnlich umfangreiche Projekt wurde in mehrere Arbeitsbereiche aufgeteilt. Die Schüler konnten sich in einem Bewerbungs- und Aufnahmeverfahren diesen Arbeitsbereichen zuordnen lassen. Anforderungen, Fristen und zu erreichende Ziele wurden gemeinschaftlich in einem Pflichtenheft festgelegt. Die Beurteilung erfolgte in einer mehrstufigen Bewertungsphase nach dem Prinzip der Selbst- und Fremdbewertung unter Beteiligung der Teammitglieder.

Die Ergebnisse der Projektarbeit auf technisch wissenschaftlicher Seite waren hervorragend.

Didaktisch war dieses Projekt aufgrund der Vielfältigkeit der eingesetzten Methoden sehr interessant und Ausgangspunkt für weitere Überlegungen Schüler intensiver in die Leistungsbeurteilung einzubeziehen.

Schulstufe: 13
Fächer: Labor für Automatisierungstechnik
Kontaktperson: Dipl.-Ing. Wolfgang Bernhofer
Kontaktadresse: HBLVA 17, 1170 Wien, Rosensteingasse 79
E-Mailadresse wolfgang.bernhofner@telering.at

1 EINLEITUNG

Zu den Folgen technischer Entwicklung gehören höhere und veränderte Anforderungen in der beruflichen Ausbildung und ein hoher Grad an Flexibilität in der Berufspraxis. Hohe Fachkompetenz ist notwendig aber nicht mehr ausreichend, um sich im Berufsleben zu etablieren und zu bestehen. Flexibilität und Selbstverantwortung gehören zu den Forderungen unserer Zeit.

Heute akzeptiert man individuelle Leistungsunterschiede und begründet so einen an der Person orientierten Leistungsbegriff. Chancengleichheit wird auf das individuelle Leistungsvermögen bezogen. Jedes Individuum sollte die Gelegenheit erhalten, in der Lerngruppe gemäß seiner individuellen Leistungsfähigkeit an seine Grenzen herangeführt zu werden.

Der Schüler sollte an der Leistungserziehung selbstverantwortlich beteiligt werden und den Lernprozess, die Leistungsbewertung und die Feststellung des individuellen Lernfortschritts mit dem Lehrer gemeinsam beurteilen oder an der Gestaltung dieser Prozesse zumindest in zunehmendem Maße der Selbstverantwortlichkeit beteiligt werden.

Dabei muss der Lehrer von der beherrschenden und moralisierenden Rolle in die Rolle des Beraters und Anbieters von Möglichkeiten wechseln.

1.1 Leistungsbewertung

1.1.1 Voraussetzungen

Im heutigen gesellschaftlichen Umfeld sind Lehrer im Allgemeinen bestrebt, Schüler zu Eigenverantwortlichkeit, Mitverantwortlichkeit und Handlungskompetenz zu erziehen. Sie streben mehrheitlich Schüler an, die qualitätsbewusst und selbstverantwortlich an der Gestaltung des Unterrichts mitarbeiten können.

Um das zu erreichen sollten Schülerinnen und Schüler lernen, objektive Qualitätsstandards, Lernstandards und Leistungsanforderungen auszuhandeln, zu akzeptieren, einzuhalten und diese auch selbst zu überprüfen. Dabei kann Teamfähigkeit projektbezogen praktiziert werden. Selbstkompetenz sollte sich über den aktuellen Lernprozess hinaus ausbilden.

1.1.2 Selbstbestimmung und Leistungserziehung

Das Team, die Lerngruppe, wird zunehmend an der Findung der Lernziele und Beurteilungskriterien beteiligt. Bei diesem Prozess kommt der Selbsteinschätzung eine große Bedeutung zu. Die Leistungsbeurteilung muss hierbei als objektivierende Komponente einfließen. Unter pädagogischen Gesichtspunkten ist gleichzeitig die Anerkennung der persönlichen Leistung jedes Einzelnen, wie auch die Anerkennung der Leistung der ganzen Gruppe wichtig, da sich die Leistungsbeurteilung nur so auch leistungssteigernd entfalten kann.

Diese Aktivitäten öffnen den Blick für gemeinschaftsbildende Momente. Wenn das Lernklima auf Erfolg und nicht auf Konkurrenz abgestimmt ist, überlagern sich sequentiell die angesprochenen pädagogischen Intentionen und Voraussetzungen mit Momenten zur Einübung in Schlüsselqualifikationen der Wirtschaft.

In der Praxis des Technikunterrichts ist es weitgehend üblich, dass den Schülern die Bewertungskriterien vor Arbeitsbeginn bekannt sind. Teils gibt der Lehrer/die Lehrerin die Bewertungskriterien vor, teils werden die Kriterien im Klassengespräch oder in Gruppenarbeit von den Schülern selbst erarbeitet. Im Sinne einer Erziehung zur Eigenverantwortlichkeit und im Sinne des Erwerbs von Individualkompetenz war es das Ziel, die Schüler in den Entstehungs- und Begründungsprozess der Bewertungskriterien einzubinden.

1.2 Naturwissenschaftlicher Unterricht

An unserer Schule, der Höheren Technischen Lehr- und Versuchsanstalt für chemische Industrie, Wien 17, Rosensteingasse 79, kommt dem naturwissenschaftlichen Unterricht eine tragende Rolle zu.

Ein wesentlicher Teil der Ausbildung in den vierten und fünften Jahrgängen der Abteilung Chemie-Informatik ist die praktische Arbeit an technischen Projekten.

Diese werden üblicherweise in Kleingruppen von zwei bis maximal vier Schülern durchgeführt, wobei die gesamte Projektvor- und Nachbereitung gemeinsam mit dem Betreuungslehrer stattfindet.

Größere Projektgruppen konnten sich bisher nicht durchsetzen, weil der Begriff **TEAM**, der modernes und effizientes Lernen in Gruppen geprägt hat, von manchem Schüler als Abkürzung für „**Toll Ein Anderer Macht's**“ empfunden wird. In größeren Teams konnten sich bisher Einzelne der Erfüllung ihrer Aufgaben entziehen, weil die Arbeiten zur Erreichung des Projektzieles ihrer Gruppe von anderen übernommen wurden.

2 AUFGABENSTELLUNG

Ziel der vorliegenden Arbeit war es, die Maximalzahl der Teammitglieder einer Gruppe zu erhöhen und damit umfangreichere Projekte zu ermöglichen, ohne den Nachteil der ungleichen Verteilung der Arbeitsanteile innerhalb der Gruppe in Kauf nehmen zu müssen.

Dazu sollte ein Umfeld geschaffen werden, das die Schüler motiviert mit vollem Einsatz an der Erreichung des Projektzieles mitzuarbeiten.

Es sollte zu einer verstärkten Einbindung der Schüler in den Gruppenbildungsprozess, die Arbeitsaufteilung sowie die Beurteilung kommen.

3 METHODEN UND DURCHFÜHRUNG

3.1 METHODEN

Dieser Teil gibt einen Überblick über die verwendeten Methoden, die detaillierte Durchführung ist in einem eigenen Unterpunkt separat angeführt.

3.1.1 Projektmethode

Einer relativ großen Gruppe von Schülern wurde eine extrem umfangreiche Aufgabe zur gemeinschaftlichen Lösung gestellt. Um diese Aufgabe zu lösen standen den Schülern vier Wochenstunden über den Zeitraum eines ganzen Schuljahres zur Verfügung.

Die Gruppe war zur Zusammenarbeit gezwungen, da die daraus isolierten unabhängigen Teilaufgaben schlussendlich zu einem Gesamtergebnis zusammengefügt werden mussten.

Die Problembewältigung erfolgte dabei in der Verantwortung der Schüler.

Sie setzten im Team Schwerpunkte, organisierten den Projektablauf und beschafften weitgehend selbst Materialien und Geräte. Sie suchten selbst Informationen im Internet und in der Literatur. Die Schüler überprüften ihren Arbeitsfortschritt selbst und waren für die Bewertung ihrer Projektergebnisse verantwortlich. Hierzu gehörte auch, dass der Projektfortschritt im Dialog zwischen Lehrer und Schüler analysiert und gesichert wurde.

3.1.2 Leistungsbewertung

Zur Leistungsbewertung wurde das Prinzip der Selbst- und Fremdbewertung herangezogen, das es den Schülern ermöglichte, am Beurteilungsprozess aktiv beteiligt zu sein.

3.1.3 Firmenähnliche Struktur

Der Umfang des Projektes erforderte eine Aufteilung der Arbeiten in Teilgebiete, die selbständig von Kleingruppen bearbeitet werden konnten. Unter Aufbau einer Struktur, bestehend aus Vorstand, Projektleiter, Bereichsleitern und Mitarbeitern, wurde die Arbeit nach genau festgelegten Kriterien auf Abteilungen aufgeteilt und die vereinbarten Projektziele in einem Pflichtenheft festgehalten.

3.1.4 Pflichtenheft

Innerhalb der Entwicklung hin zu einer größeren Gruppe von Schülern musste behutsam aber konsequent mit Vereinbarungen und Regeln gearbeitet werden.

In der Technik wird zu einem Projekt ein Pflichtenheft geführt. In ihm sind Anforderungen klar fixiert. Die Nichteinhaltung von Vereinbarungen und Forderungen muss begründet und möglicherweise auch individuell interpretiert werden.

Deshalb wurden die Projektziele analog den Forderungen der Technik in einem Pflichtenheft festgelegt.

Dabei konnten sich die Schüler mit ihren Ideen einbringen, wodurch gemeinsame Ziele gefunden werden konnten.

3.1.4.1 Technisch wissenschaftlicher Teil

In diesem Teil des Pflichtenheftes wurde festgehalten, welche Funktionen die Automatisierung der Fermentationsanlage erfüllen sollte, welche Materialien und Geräte verwendet werden müssten und wie die einzelnen Teile der Anlage zusammenarbeiten würden. Darüber hinaus wurden Abgabetermine für einzelne Teilbereiche festgelegt.

3.1.4.2 Teamarbeitsteil

In diesem Teil wurde versucht möglichst alle sozialen Aspekte der Gruppenzusammenarbeit festzulegen. Die Aufteilung der Arbeitsanteile auf die einzelnen Teammitglieder und die eigenverantwortliche Bearbeitung der zugeteilten Bereiche genauso, wie die Kooperation zwischen den Schülern während der einzelnen Arbeitsschritte.

3.1.4.3 Bewertungsteil

In der Praxis sind unterschiedliche „Spielarten“ denkbar. Im Normalfall erhält jeder Schüler die gleiche Note oder Gruppennote. Diese Note ist das Ergebnis von Gruppengesprächen und Gesprächen zwischen einzelnen Schülern mit dem Lehrer.

Wie in der vorliegenden Arbeit dargestellt, wäre es aber auch denkbar, dass man die Gruppenleistung mit einer Gesamtnote bewertet und innerhalb der Gruppe eine differenzierte Bewertung vornimmt, deren Gesamtdurchschnitt der Gruppennote entspricht.

Die Beurteilungsgrundlagen wurden im Vorfeld festgelegt und mit den Schülern gemeinsam objektiviert um Unstimmigkeiten möglichst auszuräumen.

Durch die genau festgelegte Arbeitsaufteilung konnten die Beurteilungskriterien für jeden Schüler anhand seines verfolgbaren Arbeitsanteils festgeschrieben und der zeitliche Rahmen abgestimmt werden.

3.1.4.4 Arbeitsteilungsfestlegung

3.1.4.4.1 Abteilungen

Innerhalb der Abteilungen wurden die Aufgaben einzelnen Mitarbeitern zur eigenverantwortlichen Bearbeitung übergeben. Ergebnisse wurden reflektiert und bewertet. Die dem Bereichsleitern übergebenen Konsenspapiere der Abteilungen wurden in Projektbesprechungen vorgestellt und diskutiert.

3.1.4.4.2 Bereichsleiter

Die Bereichsleiter koordinierten ihre Mitarbeiter und arbeiteten auch selbst mit.

3.1.4.4.3 Projektleiter

Der Projektleiter beriet sich mit seinen Bereichsleitern und holte sich Feedback vom Vorstand.

3.1.4.4.4 Vorstand

Der Vorstand des gesamten Projektes war der betreuende Lehrer.

3.1.5 Bewerbungsverfahren

Um die Aufteilung der Verantwortlichkeitsbereiche zu erleichtern und gleichzeitig den Schülern zu ermöglichen, sich Bereiche auszuwählen wurde die Möglichkeit geschaffen sich zu „bewerben“.

Das Ziel dabei war es, für den Schüler folgende Vorteile zu schaffen:

- Fühlt sich ein Schüler zu Führungsaufgaben berufen kann er sich als Bereichsleiter bewerben.
- Schüler, die lieber im Hintergrund bleiben wollen, können vermeiden Führungsaufgaben übernehmen zu müssen, denn wer sich nicht bewirbt wird auch nicht bestellt.
- Sieht ein Schüler ein interessantes Arbeitsgebiet, so kann er sich seiner Neigung nach bewerben und muss nicht „nehmen was übrig bleibt“.

3.2 DURCHFÜHRUNG

3.2.1 Projekt

3.2.1.1 Ausführende

Die Klasse, die diese Aufgabe erfüllen sollte, war eine Maturaklasse der 13. Schulstufe, die aus 15 Schülern, alle männlich, im Alter von 18-20 Jahren bestand. Diese Klasse hatte eine ausgeprägte Klassengemeinschaft und zeichnete sich durch überdurchschnittliche Leistungsbereitschaft aus. Die Durchschnittsnote der gesamten Klasse in dem Gegenstand, in dem die Projektarbeit durchgeführt wurde war im Vorjahr 2,15.

3.2.1.2 Pflichtenheft

3.2.1.2.1 Technisch wissenschaftlicher Teil

Als Gesamtaufgabe war die Automatisierung einer Fermentationsanlage durchzuführen. Bereits vorhandene Baugruppen sollten dabei so umgebaut werden, dass eine Steuerung durch einen Computer möglich wurde. Diese Aufgabenstellung wurde den Schülern vorgestellt und die zu verwendenden Geräte vorgeführt. Weiters wurden die zur Verfügung stehenden finanziellen Mittel besprochen.

Anschließend wurde ein Labornachmittag dazu verwendet um:

- Informationen über die tatsächliche Funktion einer solchen Anlage einzuholen
- Die Komplexität der einzelnen Baugruppen zu besprechen
- Mögliche Trennungen der Arbeitsbereiche zu finden
- Die gewünschte zu erreichende Funktionalität festzulegen
- Preise für benötigte Baugruppen festzustellen
- Arbeitsabfolgen zu überlegen

Diesem Findungsprozess folgte die Festlegung der gemeinsam erarbeiteten Qualitätskriterien unter Berücksichtigung der eben erarbeiteten Details. Diese Kriterien wurden als technisch wissenschaftlicher Teil des Pflichtenheftes festgehalten.

Festgelegt wurden:

- Aufteilung der Arbeit in drei Teilgebiete die nahezu gleichwertigen Arbeitsaufwand erwarten ließen.
- Qualitätskriterien für die Funktion der Einzelteile sowie der Gesamtanlage
- Zu verwendende Materialien und Geräte
- Einhaltung des finanziellen Rahmens
- Genaue terminliche Abfolge der einzelnen Teilergebnisse, da andere Arbeiten einzelne Zwischenergebnisse voraussetzten

3.2.1.2.2 Teamarbeitsteil

Nach einer ausführlichen Präsentation der zuvor gefundenen Arbeitsaufteilung machten sich die Schüler klar, dass eine intensive Zusammenarbeit nötig sein würde, um dieses Projekt koordinieren zu können. Gemeinsam wurden Richtlinien zur Zusammenarbeit festgelegt, die das Miteinander innerhalb der Klasse während des Projektes regeln sollten.

Festgelegt wurde, dass:

- Jeder für seinen Bereich verantwortlich ist
- Jederzeit jedem alle Information zur Verfügung gestellt werden muss
- Die Schüler sich gegenseitig weiterhelfen
- Tägliche Teamsitzungen stattfinden
- Probleme in der Zusammenarbeit sofort besprochen werden.

3.2.1.2.3 Bewertungsteil

Nach Vorstellung der Methode der Selbst- und Fremdbewertung, legte ich die Beurteilungsvariante fest. Die Gruppe sollte eine Gesamtbewertung erhalten, die die Leistung aller gemeinsam beurteilt. Individualnoten sollten durch gegenseitige Bewertung der Schüler untereinander so gefunden werden, dass sich aus den Einzelnoten im Mittel die Gesamtnote ergibt.

Nach einer längeren Diskussion konnten folgende Beurteilungskriterien festgelegt werden:

- Erreichung der Zielvorgaben bei der Funktion: 25%
- Einhaltung der Abgabetermine: 25%
- Leistungsbereitschaft: 20%
- Hilfsbereitschaft: 20%
- Den Zusammenhalt fördernde Leistungen: 10%

Diese Kriterien sollten für die Gesamtbewertung der Projektarbeit durch mich, wie auch für die gegenseitigen Bewertungen der Schüler untereinander Gültigkeit haben.

3.2.1.3 Gliederung

Das Projekt gliederte sich in drei Teilbereiche:

- Umrüstung und Umbau der Anlagenkomponenten
- Bau einer SPS und Programmierung der Software zur Steuerung der SPS
- Programmierung der Steuersoftware am Computer mit LabVIEW

3.2.2 Firmenstruktur

Um eine den Arbeitsanforderungen gerechte Gruppenaufteilung zu erreichen, schlug ich vor, die Arbeit an den drei Teilbereichen auch physisch in drei Abteilungen durchzuführen. Um etwas Realitätsbezug im Arbeitsumfeld zu schaffen, stellte ich den Vorstand einer Firma dar, die den Auftrag angenommen hat, eine Automatisierung einer Fermentationsanlage vorzunehmen.

Ich setzte den Projektleiter für das gesamte Projekt nach einem Ausschreibungs- und Bewerbungsverfahren ein.

Der Projektleiter klärte mit dem Vorstand, mir, grob die Vorgehensweise ab und traf die Entscheidung, nicht nur wie von mir vorgeschlagen die Bereichsleiter, sondern auch alle anderen Positionen mittels Ausschreibungs- und Bewerbungsverfahren zu besetzen.

Für die drei oben erwähnten Bereiche wurden die Positionen der Bereichsleiter mit einem entsprechenden Anforderungsprofil ausgeschrieben und im Zuge des anschließenden Aufnahmeverfahrens durch den Projektleiter in Abstimmung mit dem Vorstand besetzt.

Diese drei Bereichsleiter nahmen ihrerseits eine weitere Aufgliederung ihres Zuständigkeitsbereiches vor. Sie definierten die einzelnen Arbeiten in Abstimmung mit dem Projektleiter exakt.

Diese Arbeitsdefinitionen dienten als Beschreibung der weiteren durch die Bereichsleiter zur Ausschreibung kommenden Positionen für Mitarbeiter, die in einem Bewerbungs- und Auswahlverfahren durch die Bereichsleiter in Abstimmung mit dem Projektleiter bestellt wurden.

Diese doch relativ komplizierte und mehrstufige Arbeitsaufteilung, die gewährleisten sollte, dass jeder nur in der Position und Ebene zum Einsatz kommt, die ihm entsprach, stellte sich als viel weniger langwierig heraus, als von mir befürchtet, da nur wenige Überschneidungen bei den Wunschpositionen vorlagen und eine Zuteilung der Aufgaben rasch und einfach erfolgen konnte.

Damit hatte jeder Schüler eine Position und einen Aufgabenbereich innerhalb des Projektteams.

3.2.3 Ausschreibungs- und Bewerbungsverfahren

Der jeweilige Arbeitsbereich jeder Position innerhalb der Arbeitsgruppen wurde kurz beschrieben und die Anforderungen an den Bewerber festgelegt. So waren reine Programmieraufgaben ebenso wie stark handwerklich orientierte Problemstellungen oder koordinative Bereiche ausgeschrieben. Nach der Beschreibung der Position wurde kurz im engen Kreis der Interessierten abgeklärt, ob das Fremd- mit dem

Selbstbild jedes Bewerbers seine Fähigkeiten diese Position abzudecken übereinstimmt. Da sehr stark divergierende Aufgaben und Interessen zu einer gleichmäßige Verteilung der Bewerber führte, konnte nach kürzester Zeit jedem eine befriedigende Aufgabe zugewiesen werden.

3.2.4 Präsentationen

In regelmäßigen Abständen wurden Präsentationen der einzelnen Gruppen durchgeführt um ihren Arbeitsfortschritt darzulegen und Feedback zu erfahren. Nach einigen Problemen, die sich dadurch ergeben hatten, dass die Einzelleistungen der Gruppenmitglieder bei den Präsentationen nicht gewürdigt wurden, wurden die Präsentationen dahingehend erweitert, dass jeder seinen eigenen Arbeitsfortschritt darstellen durfte.

3.2.5 Selbst- und Fremdbewertung

Gemeinsam wurde vor Arbeitsbeginn der Rahmen der Projektarbeit aufgespannt. Dies geschah in Gesprächen anlässlich der Verfassung des Pflichtenheftes. Dabei wurden Organisations- und Aktionsrahmen, Ort, Zeitablauf, Verantwortlichkeiten, finanzieller Rahmen, personelle Ressourcen und die Bewertungskriterien festgelegt.

Die Beurteilungen der Schüler untereinander fanden monatlich nach den Präsentationen der laufenden Arbeitsfortschritte statt. Den Anfang machte für jeden Schüler eine verbale Selbsteinschätzung seiner Leistungen, die er an seinen Vortrag anknüpfte. Anschließend wurde diese Selbsteinschätzung kurz diskutiert, meist aber nicht von den Mitschülern angezweifelt. Nach dieser Diskussion gab jeder der Zuhörer einen Zettel mit seiner Bewertung des Vortragenden ab.

Um nach den monatlich stattfindenden Präsentationen vernünftige Beurteilungen der Arbeiten nach den Bewertungskriterien zu bekommen, war es notwendig, dass sich die Schüler und der Lehrer während der Arbeit Notizen machten, weil schnell klar wurde, dass die Einzelleistungen über einen so großen Zeitraum von den anderen vergessen wurden.

Nach einiger Zeit stellte sich aber das Problem, dass die Klassendynamik diese Beurteilungsprozesse dahingehend verschob, dass sich alle selbst mit „gut“ beurteilten und durch die anderen mit „sehr gut“ bedacht wurden.

Da von vergangenen Diskussionen die Klassengemeinschaft schon belastet war, wurden diese Beurteilungsrunden beendet und auf eine Gesamtbeurteilungsrunde am Jahresende verschoben.

Da stellte sich das Problem dann nicht mehr, weil ich mich in der Zwischenzeit entschlossen hatte die Gesamtbeurteilung allen zukommen zu lassen, das bedeutete, dass alle Schüler die gleiche Note erhielten. Ich empfand es den Schülern gegenüber als unfair, wenn in ihrem Abschlusszeugnis Noten aufscheinen würden, die gegen den Willen der Schüler von einer noch sehr experimentellen Beurteilungsmethode beeinflusst würden.

3.2.6 Schülerbefragungen

Unmittelbar nach der Einteilung der Schüler fand eine erste einfach gehaltene Schülerbefragung statt. Sie bestand nur aus Fragen, die mit JA, NEIN und KANN NOCH NICHTS SAGEN zu beantworten waren.

- Gefällt dir eine firmenähnliche Aufteilung des Klassenverbandes?
- Hast du die Aufgabe bekommen, die du dir gewünscht hast?
- Fühlst du dich durch das neue Konzept besonders motiviert?

3.2.6.1 Der geschlossene Schülerfragebogen

Nach der Hälfte des Projektes wurde eine umfassende anonyme Befragung mittels Fragebogens durchgeführt um einen ersten Zwischenstand festzuhalten.

(Fragebogen im Anhang)

3.2.6.2 Der offene Schülerfragebogen

Zur laufenden Erhebung der Befindlichkeiten der Schüler wurde ein offener Fragebogen so gestaltet, dass Raum für selbst formulierte Antworten blieb.

Diese Fragebögen wurden in unregelmäßigen Abständen, die aber nie größer als ein Monat waren, von den Schülern ausgearbeitet und anschließend diskutiert.

Die Fragen waren analog jenen des geschlossenen Fragebogens, richteten sich in erster Linie auf die soziale Komponente des Projektes und sollten erheben, wie die Zusammenarbeit zwischen den Schülern und zwischen den Schülern und dem Lehrer empfunden wurde.

3.2.6.3 Interviews

Wie sich im Laufe der Zeit herausstellte, waren die Diskussionen nach Ausfüllen der offenen Fragebögen viel aussagekräftiger als die Auswertung der Antworten, sodass zur Evaluation des Gesamterfolges und der aktuellen Befindlichkeiten Interviews herangezogen wurden.

3.2.6.4 Technische Ausführung der Arbeiten

Näheres ist dem ANHANG zu entnehmen.

3.2.6.4.1 Ausstattung der Pumpen mit digitalen Potentiometern

Die Förderleistung der analog steuerbaren Pumpen konnte mittels Drehpotentiometers stufenlos geregelt werden. Um den Eingriff in dieses Steuerungssystem mittels Computer zu ermöglichen, mussten die Drehpotentiometer durch digitale Potentiometer gleichen Regelbereichs ersetzt und Anschlussstecker für die externe Steuerung installiert werden. Die digitalen Potentiometer konnten nun über eine Steuereinheit in 255 Schritten über ihren Regelbereich verstellt werden, wodurch die Veränderung der Förderleistung über eine Schnittstelle möglich wurde.

3.2.6.4.2 Automatisierung der pH-Messung

Ein pH-Meter mit einer RS-232 Schnittstelle wurde so installiert, dass alle Messdaten an den Computer übergeben wurden.

3.2.6.4.3 Automatisierung der Temperaturmessung

Es gab zwei Temperaturmessstellen in dem System. Eine befand sich im Thermostat, die andere im Reaktor. Der verwendete Thermostat hatte eine selbsttätige Temperaturregelung, welche die Wassertemperatur auf einer manuell vorgegebenen Größe hielt. Um diese Regelung durch den Computer steuerbar zu machen wurde der Regelkreis des Temperaturfühlers unterbrochen und die Messsignale dem analogen Eingang der Messkarte des Computers zugeführt. Umgekehrt wurden entsprechend veränderte Signale an den Thermostaten zurück geschickt um das Ein- und Ausschalten der Heizung beliebig zu beeinflussen. Die Temperaturmessung im Reaktor wurde mit einem digitalen Temperatursensor durchgeführt, dessen Signale direkt an den Computer übermittelt werden konnten.

3.2.6.4.4 Bau einer SPS

Es hat sich als äußerst umständlich erwiesen, alle Steuerungen und Schnittstellen direkt über den Computer zu verwalten. Die Aufgabenstellungen würden zu viele analoge und serielle Eingänge voraussetzen. Aus diesem Grund wurde der Bau einer SPS (Speicherprogrammierbaren Steuerung), die zwischen den zu steuernden Komponenten und dem Computer so geschaltet werden sollte, dass nur mehr eine einzige Kommunikationsschnittstelle notwendig war, beschlossen. Diese SPS beruhte auf einem PIC und den umgebenden Erweiterungsbausteinen.

Ein 187615 PIC wurde mit einem LCD-Display, zwei USB sowie einer seriellen RS232C Schnittstelle auf einer Standardplatine verbaut.

3.2.6.4.5 Programmierung der Steuersoftware des PIC

Unter Zuhilfenahme der proprietären Programmiersprache des PIC wurden mehrere Funktionen in die SPS integriert, die selbständig die Schnittstellen der Geräte sowie die digitalen Potentiometer bedienen konnten.

3.2.6.4.6 Programmierung der Steuersoftware mittels LabVIEW

Zur Steuerung des gesamten Prozesses und zum Ansprechen und Auslesen der SPS wurde eine umfangreiche Steuerungssoftware am PC mittels LabVIEW programmiert. Diese Software stellte sich als einer der komplexesten und aufwendigsten Teile der Gesamtanlage dar.

4 ERGEBNISSE

4.1 Gesamtergebnis

Da in unserer Abteilung schon immer Projektarbeiten stattgefunden haben, ist es relativ einfach die Ergebnisse dieses Projektes mit Ergebnissen vorheriger Arbeiten zu vergleichen.

Die wissenschaftlich-technischen Ergebnisse dieses Projektes waren sehr gut. Der größte Unterschied war in der Größe des Projektes zu sehen, denn vergleichbar umfangreiche Aufgaben wurden noch nie in dieser Qualität durchgeführt.

Die soziale und menschliche Komponente dieses Projektes war interessant und aufschlussreich.

Die gegenseitige Beurteilung der Schüler führte schlussendlich zu keinem Aufbrechen der Klassengemeinschaft.

Das Ziel, dass sich kein Schüler aus seiner Verantwortung zurückziehen und anderer Schüler Arbeit für sich reklamieren kann, wurde voll erfüllt.

Aufgrund der Aufteilung der Zuständigkeitsbereiche und der exakten Planung der Zielvorgaben konnte eine fristgerechte Lösung aller Aufgabenstellungen erreicht werden.

Der Unterricht wurde von den Schülern als sehr attraktiv empfunden, die eigene Position im Klassenverband sahen die meisten als gestärkt an.

Die Leistungsfeststellung und Beurteilung stellte ein größeres Problem dar, weil die anfängliche Zustimmung der Schüler sich durch Selbst- und Fremdbewertung beurteilen zu lassen, im Laufe der Zeit schwand.

4.2 Selbstbeobachtungen (wie ich den Unterricht gesehen habe)

Die mit großem Elan begonnen Arbeiten führten rasch zu brauchbaren und verwertbaren Ergebnissen.

Die geforderten Zwischenberichte zu den einzelnen Arbeitsschritten waren pünktlich fertig und von hoher Qualität.

Nach den anfänglich doch extrem positiven Reaktionen der Schüler wurde rasch klar, dass eine völlige Übergabe der Leistungsfeststellung in die Eigenverantwortung der Schüler zu Schwierigkeiten führen würde, weil die Beurteilungspraxis anlässlich der Präsentationen immer weniger objektiv wurde.

Mein Arbeitsaufwand während des Projektes hielt sich, nach Abschluss der vorbereitenden Arbeiten, im Rahmen des Üblichen. Zusätzlicher Aufwand, der sich aus laufenden Anpassungen der Methodik sowie beratenden Gesprächen ergab, wurde durch geringeren Aufwand bei der Kontrolle des Arbeitsfortschrittes mehr als kompensiert.

Die Vorbereitenden Überlegungen und die Beobachtungen der Reaktionen der Schüler bei der Umsetzung dieses Projektes machten wirklich Spaß und es war erfreulich den Erfolg eines neuen Konzeptes zu sehen.

4.3 Schülerbefragungen

Unmittelbar nach der Einteilung der Schüler fand eine erste einfach gehaltene Schülerbefragung statt:

Gefällt dir eine firmenähnliche Aufteilung des Klassenverbandes?

JA: 57% NEIN: 30% KANN NOCH NICHTS SAGEN: 13%

Hast du die Aufgabe bekommen, die du dir gewünscht hast?

JA: 86% NEIN: 14%

Fühlst du dich durch das neue Konzept besonders motiviert?

JA: 86% NEIN: 7% SO WIE IMMER: 7%

4.3.1 Der geschlossene Schülerfragebogen

Nach der Hälfte des Projektes wurde eine umfassende anonyme Befragung mittels Fragebogens durchgeführt um einen ersten Zwischenstand festzuhalten.

Die Beantwortung des geschlossenen Schülerfragebogens ergab kein eindeutiges Bild, weil fast alle Fragen extrem positiv beantwortet wurden. Offensichtlich wollten mir die Schüler einen Gefallen mit ihrer Beurteilung machen. Auf eine Wiederholung dieser Form der Befragung wurde deshalb verzichtet.

(Fragebogen im Anhang)

4.3.2 Der offene Schülerfragebogen

Zur laufenden Erhebung der Befindlichkeiten der Schüler wurde ein offener Fragebogen so gestaltet, dass Raum für selbst formulierte Antworten blieb.

Diese Fragebögen wurden in unregelmäßigen Abständen, die aber nie größer als ein Monat waren, von den Schülern ausgearbeitet und anschließend diskutiert.

Die Fragen richteten sich in erster Linie auf die soziale Komponente des Projektes und sollten erheben, wie die Zusammenarbeit zwischen den Schülern und zwischen den Schülern und dem Lehrer empfunden wurde.

Die Auswertung dieser Fragebögen war aufschlussreicher als die des geschlossenen Fragebogens. Da aber je nach Projektphase andere Fragen interessant waren, stellte sich heraus, dass diese Form der Befragung zu unflexibel war und die Schüler langweilte. Die Antworten, die bei den ersten Bögen noch ausführlich ausfielen wurden immer spärlicher. Auf Rückfrage nach den Ursachen dieser wortkargen Aussagen gaben die Schüler an, ihre Meinung sowieso schon kundgetan zu haben und nicht immer dieselben Fragen beantworten zu wollen. Auf einen weiteren Einsatz dieser Form der Befragung wurde deshalb ebenfalls verzichtet.

4.3.3 Interviews

Wie sich im Laufe der Zeit herausstellte, waren die Diskussionen nach ausfüllen der offenen Fragebögen viel Aussagekräftiger als die Auswertung der Antworten, sodass zur Evaluation des Gesamterfolges und der aktuellen Befindlichkeiten Interviews herangezogen wurden.

Es zeichnet sich in diesen Gesprächen eine große Zufriedenheit mit dem Gesamtkonzept ab. Offensichtlich waren die Schüler davon überzeugt, dass die Ziele des Projektes erreicht wurden, weil keiner das Gefühl äußerte, dass Mitschüler gar nichts oder sehr viel weniger als andere zum Erfolg beigetragen hatten.

Der Hauptkritikpunkt, der aus den frei formulierten Meldungen abgelesen werden konnte, war, dass manche Schüler überzeugt waren, dass ihr Anteil am Gesamterfolg durch die gemeinsame Präsentation der Gruppe geschmälert wurde. Diese Problematik wurde durch Einzelpräsentationen gelöst.

4.4 Selbst- und Fremdbewertung

In diesem Projekt nahm ich eine andere Rolle als im normalen Unterrichtsprozess ein. In diesem Fall war ich zumeist der Beobachter, der in die eigentliche Arbeit der Gruppe nicht mehr eingebunden war.

Am Ende wusste ich nicht genau, welche Probleme innerhalb der Gruppe bewältigt werden mussten und welche Schüler substantielle Beiträge geliefert hatten.

Während der Gespräche anlässlich der Beurteilungsphasen nach den Präsentationen kam es häufig zu unterschiedlichen Einschätzungen durch die Klassenkameraden oder zu einer unterschiedlichen Bewertung einer Qualitätsfrage durch die Schüler und den Lehrer.

Trotzdem entwickelte sich bei diesen Findungsprozessen eine erstaunlich sachliche Gesprächskultur. Die Schüler argumentierten ruhig und überzeugend, ließen einander ausreden und hörten zu. Positives konnte herausgestellt, persönliche Vorbehalte hintangehalten werden. Kennzeichen dieser Gesprächskultur war, dass in erster Linie Sachargumente sprechen gelassen wurden.

Nachdem die Klasse aber nach drei Monaten dahingehend übereingekommen war, sich gegenseitig nur mehr mit „sehr gut“ zu beurteilen, fanden diese konstruktiven Beurteilungsdiskussionen nicht mehr statt.

Der Grund für diese Verweigerung der Selbst- und Fremdbewertung war einfach, dass die Schüler sich um ihre Abschlussnoten zu sorgen begannen, weil die bisherigen Runden der Beurteilung aufgrund der Ernsthaftigkeit mit der die eigene Arbeit beleuchtet wurde teilweise doch recht schlechte Noten hervorgebracht hatten. Die Sorge in ihrem Überschwang über das Ziel der gerechten Bewertung hinausgeschossen zu sein, konnte ich auch durch ausführliche Besprechungen nicht ganz ausräumen

4.5 Beurteilung

Am Ende des Projektes gab ich eine Gruppenbewertung, weil der Bewertungsgegenstand das Ergebnis der gesamten Gruppe darstellte. Entgegen meiner ursprünglichen Intention durch stattgefundene Selbst- und Fremdbewertung der Schüler die Beurteilung weiter zu diversifizieren, erhielt jeder Projektteilnehmer dieselbe Note.

Die Gesamtleistung der Gruppe war weit höher als die Summe der Einzelleistungen. Die individuellen Leistungsbeiträge hatten sich im Gestaltungsprozess so optimal beeinflusst, dass Fehler vermieden und durch die Streuung der individuell verschiedenen Leistungsstärken bessere Lösungen gefunden werden konnten als der Einzelnen zu leisten im Stande gewesen wäre.

5 DISKUSSION UND AUSBLICK

5.1 DISKUSSION

Da ich als Lehrer während der Durchführung dieses Projektes nicht mehr Zentrum des unterrichtlichen Geschehens war, konnte ich auch nicht mehr der alleine Zuständige für die Bewertung der Leistung sein. Die Verantwortung dafür behielt ich allerdings.

Das ist ein für mich noch nicht gelöstes Problem!

Wie sich aufgrund der Spannungen während der Bewertungsphase gezeigt hat, ist die Leistung dann nicht mehr das die Gruppe verbindende Element, wenn die Schüler gezwungen werden, durch gegenseitige Beurteilung Abstufungen in der Bewertung der Leistung des Einzelnen durchzuführen. Die Gruppe stößt dabei durch die notwendige Diskussion leicht an die Grenzen ihrer Belastbarkeit. Der Lehrer sollte bei diesen Bewertungsgesprächen in jedem Fall als Diskussionsleiter agieren.

Durch diese Form des Unterrichtes hat sich meine Beurteilungspraxis geändert. Ich beachte folgende Punkte:

- Abstimmung der Bewertungskriterien mit den Schülern
- Abstimmung des Bewertungsschlüssels mit den Schülern
- Zunehmende Beteiligung der Schüler an der Bewertung der Schülerleistungen
- Produktbezogene Bewertung wird durch prozessbezogene Bewertung ergänzt
- Beachtung der gesamten Schülerpersönlichkeit
- Beachtung von Sozialverhalten, Arbeitsverhalten und Lernverhalten
- Interpretation und Würdigung der Leistung unter Beachtung des individuellen Leistungsvermögens und persönlicher Bedingtheit

Die Selbst- und Fremdbewertung zählt für mich zu den integrierenden Bestandteilen des neuen Lernens.

Es fällt mir schwer abschließend eine Empfehlung an andere abzugeben, diese oder ähnliche Methoden in ihren Unterricht einfließen zu lassen. Ich kann nur ganz persönlich feststellen, dass es mir wirklich Spaß gemacht hat, dieses Projekt durchzuführen. All der Aufwand, der mit den vorbereitenden Überlegungen, den Befragungen und den Diskussionen die Beurteilung betreffend zusammenhing, ist nichts im Vergleich dazu, den Schülern bei ihrer Entwicklung hin zu einer eigenverantwortlichen Gruppe zuschauen zu können und zu sehen, dass man als Lehrer ruhig auch einmal das Zepter aus der Hand geben kann, ohne dass das Chaos ausbricht.

5.2 AUSBLICK

Als Befürworter und energischer Verfechter von Projektarbeit und Gruppenarbeit werde ich meinen Unterricht auf Selbsttätigkeit und Kooperation in der Gruppe abstimmen.

Da zur Förderung der Selbständigkeit auch die zunehmende eigenverantwortliche Planung, Durchführung und Bewertung von Lernprozessen gehört, werde ich diese Prinzipien auf meine Arbeit in der Gruppe anwenden.

Als Techniklehrer werde ich nach Möglichkeiten suchen, wie ich eine individuelle Leistungsorientierung aufbauen kann. Gleichzeitig erfordert die Gruppenbewertung noch mehr Überlegungen, da hierbei das Gruppen- und das individuelle Leistungsverhalten durch Fehlentscheidungen nachhaltig gestört werden können.

Die Lösung scheint mir in offenen Gesprächen zwischen dem Lehrer und seinen Schülern bzw. den Schülern untereinander zu liegen. Fragen des Teamverhaltens gilt es im Vorfeld zu diskutieren. Fragen der Leistungsbewertung ergeben sich dann häufig als Konsequenz konkreter Unterrichtssituationen sehr anschaulich.

So können Schüler für die Beobachtung und Wertung ihres eigenen Verhaltens und des Fremdverhaltens sensibilisiert werden. Über diese Gespräche gilt es zunehmend objektive Beobachtungskriterien zu erschließen.

Es geht darum, den Mittelweg zu finden zwischen notwendiger konstruktiver Kritik, die häufig für den Betroffenen mit Frustration verbunden ist, und dem notwendigen Respekt vor dem Anderen. Takt und Respekt vor dem Schüler bzw. der Mitschüler setzen Grenzen in der Bewertung des anderen.

6 LITERATUR

BASTIAN, JOHANNES. (1996). Leistung im Projektunterricht. In: Prüfen und Beurteilen. [Jahresheft] 26-30. Seelze: Friedrich Verlag.

FAST, LUDGER (1997). Lernerfolgskontrolle, Leistungsbeurteilung und Notengebung im Technikunterricht. In: LUDGER FAST & HARALD SEIFERT (Hrsg.), Technische Bildung. Geschichte, Probleme, Perspektiven. Weinheim.

FAST, LUDGER (1999). Prüfen und Bewerten im Technikunterricht. In: Unterricht/Arbeit+Technik.

GOETSCH, KARLHEINZ (1993). Projektunterricht bewerten. In: GUDJONS, HERBERT (Hrsg.) Das Projektbuch II. Hamburg.

HOHLOCH, MARTIN (2001). Selbstbewertung und Fremdbewertung. In: Unterricht/Arbeit+Technik.

7 ANHANG - DER GESCHLOSSENE SCHÜLERFRAGEBOGEN

<i>Ist der Lehrplanbezug ersichtlich?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Können Sie im Projekt erarbeitete Fertigkeiten fachübergreifend verwenden?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Glauben Sie, dass die Erreichung des fachlichen Zieles durch die Projektarbeit einfacher wird?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Ist der Unterricht weniger streng?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Sehen Sie sich in einer völlig neuen Situation?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Empfinden Sie den Unterricht Interessant?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Verstehen Sie sich mit Ihren Mitschülern besser?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Werden Wissen und Kompetenzen gefördert?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Fühlen Sie sich im Unterricht wohl?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Ist die Verteilung der Aufgaben gerecht?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Gibt es für Sie genügend Unterstützung im Unterricht?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Wie sind Sie mit den Rahmenbedingungen zufrieden?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Fühlen Sie sich durch den Lehrer betreut?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Helfen Ihnen Ihre Mitschüler weiter?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Wird Ihre Arbeit gewürdigt?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Wird den Ansprüchen der Schüler Rechnung getragen?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Sind Sie mit Ihrer Arbeit zufrieden?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Wird auf Rückmeldungen reagiert?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Empfinden Sie den Projektunterricht weniger wichtig?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Sind die Leistungsanforderungen zu hoch?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>
<i>Ist die Leistungsbeurteilung nachvollziehbar?</i>	<i>Sehr</i>	<i>eher</i>	<i>-</i>	<i>wenig</i>	<i>nicht</i>

8 ANHANG - TECHNISCHE BESCHREIBUNG DES PROJEKTS

8.1 Das Steuerprogramm

8.1.1 Anforderungen an das Steuerprogramm

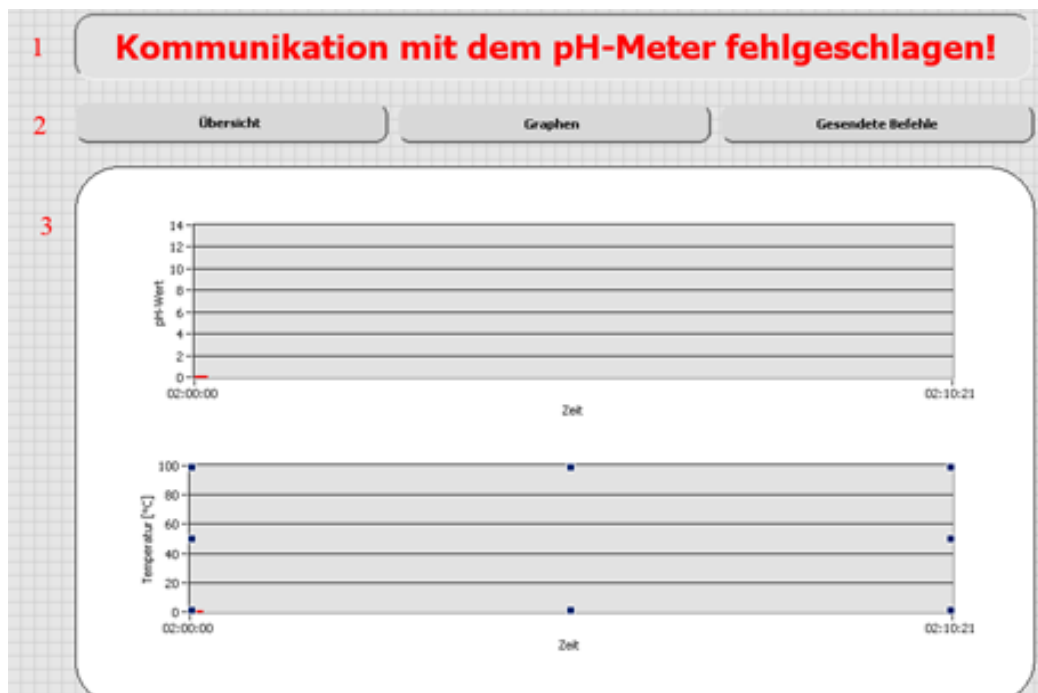
Um alle Bedienelemente des Fermenters per Computer bedienen zu können muss das Steuerprogramm folgende Funktionen besitzen:

- Eine Benutzeroberfläche
- Kommunikationsfunktionen um dem Mikrokontroller Befehle zu senden und Messdaten zu empfangen.
- Eine Regelungsfunktion für das Einstellen des pH-Wertes in der Fermentationskammer.
- Eine Regelungsfunktion für ein Thermostat um die Temperatur in der Fermentationskammer konstant zu halten.
- Eine Pumpensteuerungsfunktion um Nährlösung in die Fermentationskammer zu pumpen.

Weiters sollen alle Messwerte graphisch aufbereitet in Diagrammen dargestellt werden.

8.2 Die Benutzeroberfläche

Die Benutzeroberfläche setzt sich im Wesentlichen aus 3 Teilen zusammen:



8.2.1 Die Statusleiste

In der Statusleiste werden wichtige Meldungen angezeigt wie z.B. dass die Kommunikation mit dem Mikrokontroller oder dem pH-Messgerät fehlgeschlagen ist.

8.2.2 Die Buttonleiste

Mit der Buttonleiste kann der Benutzer zwischen den einzelnen Funktionen des Programms navigieren. Nach dem Drücken einer Taste wird der dementsprechende Inhalt im Feld Nr. 3 angezeigt.

8.2.2.1 Der Button Übersicht

Nach dem Drücken des Buttons „Übersicht“ kann der Benutzer alle Funktionen des Programms mittels eines Flussdiagramms des Fermenters steuern.

8.2.2.2 Der Button Diagramme

Hier werden die Temperatur und der pH-Wert in der Fermentationskammer graphisch angezeigt. Hierbei wird der Ist- Wert in grün und der Soll- Wert in rot dargestellt.

8.2.2.3 Der Button Gesendete Befehle

Nach dem Drücken dieses Buttons gelangt der Benutzer zu einem Feld in dem alle, dem Mikrocontroller gesendeten Befehle angezeigt werden. Hier kann gegebenenfalls überprüft werden, ob eine Fehlfunktion in einem der Steuerprogramme vorliegt.

8.2.3 Das Hauptfenster

Im Hauptfenster werden alle wichtigen Werte und Funktionen dem Benutzer zugänglich gemacht. Der Inhalt des Fensters lässt sich mit den Feldern in der Buttonleiste verändern.

8.3 Das Design der Benutzeroberfläche

Da die Benutzeroberfläche nicht dem Standarddesign eines LabVIEW Programms entsprechen sollte, mussten verschiedene Modifikationen an den Front Panel Elementen vorgenommen werden.


Kommunikation mit dem pH-Meter fehlgeschlagen!

Die Statusleiste besteht aus einer „Recessed Rounded Box“ Dekoration mit dem Farbwert „226:226:226“. Darüber wurde eine String Control platziert, deren Hintergrund mit dem „Set Color“ Werkzeug transparent gemacht wurde. Weiters wurde der Rahmen der Control mittels des Control Editors von LabVIEW so klein gemacht, dass er unsichtbar erscheint. Dies

musste so gelöst werden, da LabVIEW keine Funktion besitzt, um auch den Rahmen einer Control transparent zu machen.



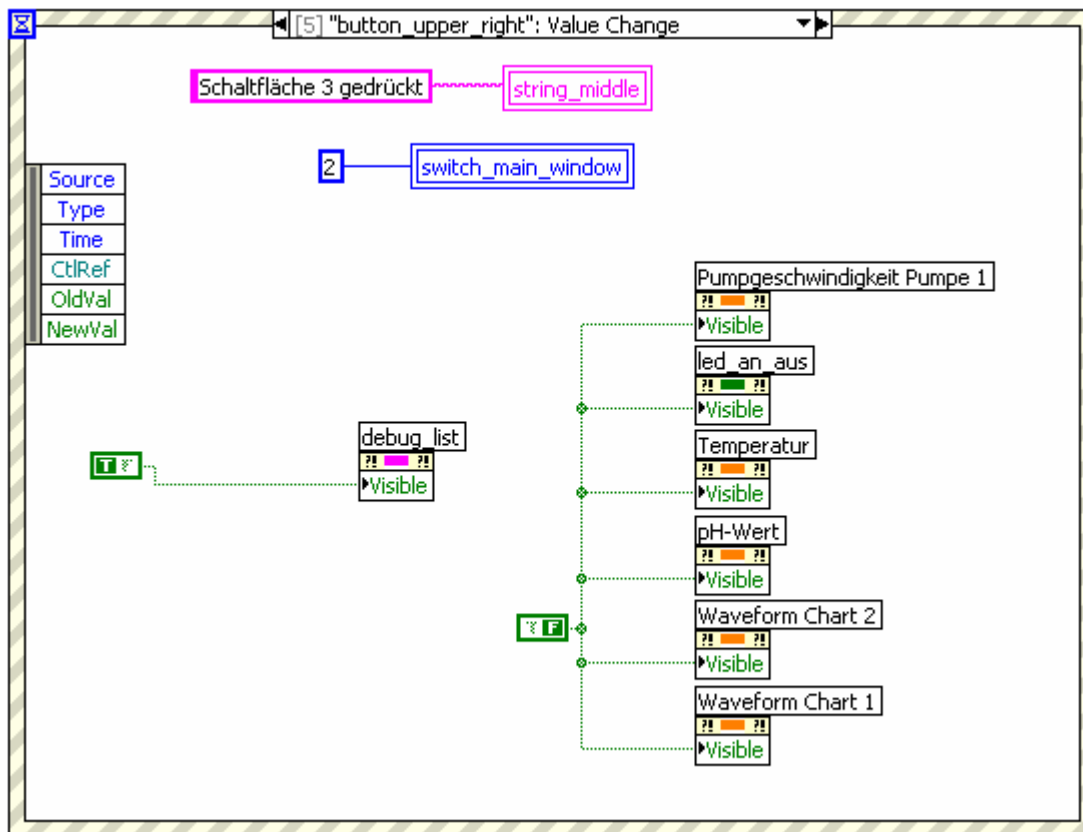
Die einzelnen Felder der Buttonleiste bestehen aus einer „Raised Rounded Box“ Dekoration mit dem Farbwert „216:216:216“. Darüber wurde ein „Labeled Square Button“ aus den Classic Controls gesetzt, dessen Hintergrund transparent gemacht wurde.

Das Hauptfenster besteht aus 2-mal einer „Flat Rounded Box“ und einer „Flat Box“ Dekoration. Alle diese Elemente haben die Hintergrundfarbe „255:255:255“ und sind nebeneinander angeordnet. Dabei wurde die Position der einzelnen Elemente mit dem „Rekorder“ Werkzeug () so verändert, dass eine einzelnen Box mit leicht abgerundeten Ecken entstand.

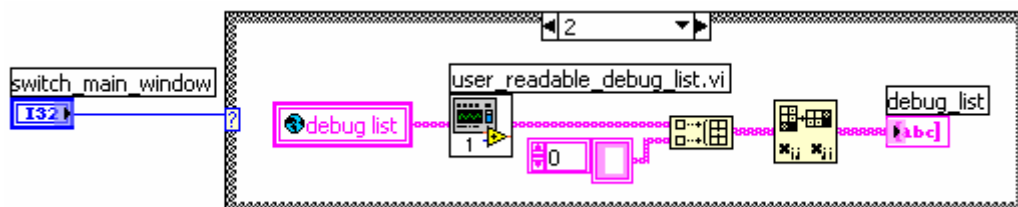
8.3.1 Programmtechnische Implementierung der Buttonleiste

Hierbei wurde von dem „Event Loop“ Gebrauch gemacht. Diese konfigurierbare Schleife führt einen bestimmten Programmteil aus, wenn ein vorher bestimmtes Event auftritt. Dies kann z.B. das Drücken eines Buttons oder die Veränderung eines Wertes sein. Nachdem die Schleife im Block Diagramm aufgezogen wurde, konnte in dem Fenster „Edit Events“ jedem Button der Event „Value Change“ zugewiesen werden. Das bedeutet, jedes Mal wenn sie der Boolean Wert der Button ändert wird der zugehörige Programmteil in der Event Schleife ausgeführt. Mittels der Eigenschaft Visible der jeweiligen Property Node der Bedienelemente können diese nun Ein – und Ausgeblendet werden.

Zur Veranschaulichung ein Auszug aus dem Programmcode:



Mit der hier sichtbare Variable „switch_main_window“ lässt sich bestimmen, welches Anzeigefeld im Main Window gerade ausgewählt ist. Somit ist es möglich bestimmte Rechenintensive Programmfunktionen nur dann auszuführen, wenn der Benutzer sie auch tatsächlich sieht. Dies wird z.B. bei dem Auslesen der gesendeten Befehle angewandt:

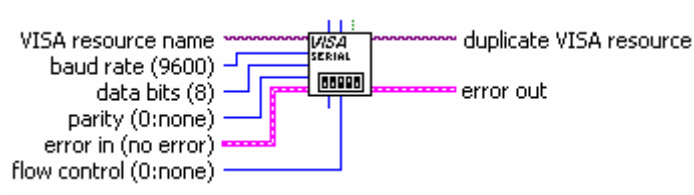


Der in der Case Schleife Programmcode wird nur dann ausgeführt, wenn auch das entsprechende Fenster gewählt wurde.

8.3.2 Kommunikation mit dem Mikrocontroller

Die Kommunikation mit dem Mikrocontroller wurde über die Serielle Schnittstelle realisiert. Diese lässt sich mit den VISA Funktionen in ein Programm integrieren. Die 3 wichtigsten Funktionen sind „VISA Configure Serial Port“, „VISA Write“ und „VISA Read“.

8.3.2.1 VISA Configure Serial Port



Mit diesem VI lassen sich die verschiedenen Eigenschaften wie „Baud Rate“, „Data Bits“, „Parity“ und „Flow Control“ der Seriellen Schnittstelle festlegen.

8.3.2.2 VISA Write



Mit diesem VI können Strings an den Seriellen Port gesendet werden. Wenn daraufhin Werte empfangen werden sollen, kann der Umfang dieser mit dem Output „Return Count“ ermittelt werden.

8.3.2.3 VISA Read



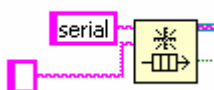
Mit diesem VI können Daten vom Seriellen Port eingelesen werden. Als Parameter erwartet diese Funktion wie viele Daten gelesen werden sollen (Byte Count).

Da verschiedene Unterfunktionen des Programms gleichzeitig die Serielle Schnittstelle benötigen musste der Zugriff auf diese geregelt werden. Dies erfolgt mit Queues.

Eine Queue symbolisiert einen Stapel zu dem Werte hinzugefügt und entfernt werden können. Das besondere an den Queues ist, dass sie Systemweit zu Verfügung stehen, dass heißt jedes im Speicher befindliche VI kann auf ein und dieselbe Queue zugreifen.

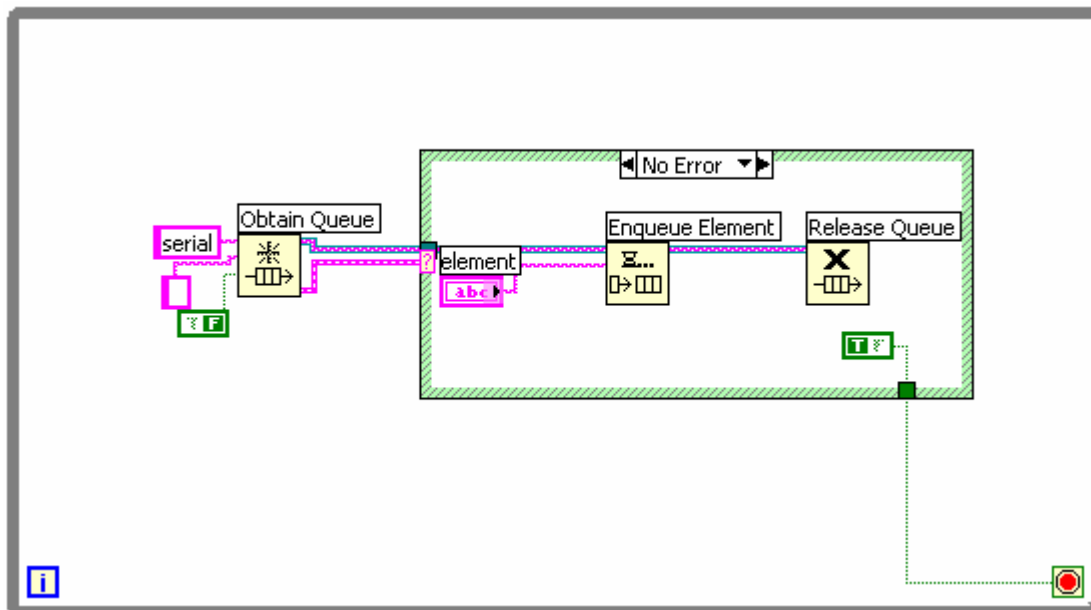
8.3.3 Implementierung der Queues

Zuerst wird in dem VI „read_and_send_queue“ eine Queue erstellt:



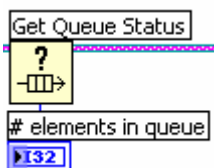
Diese hat den Namen „serial“ und den Datentyp „String“.

Nun kann durch Aufruf des VI „add_new_elements_to_queue“ ein neuer Befehl in die Queue geschrieben werden:



Das VI „Obtain Queue“ sichert sich den alleinigen Zugriff auf die Queue mit dem Namen „serial“, sollte dabei ein Fehler auftreten, schaltet die Case Structure auf Error und dem Abbruchsbedingungsfeld der Schleife wird das Element „False“ übergeben. Dies hat zur Folge, dass noch ein Schleifendurchlauf startet und neuerlich versucht wird den alleinigen Zugriff auf die Queue zu erhalten. Sollte dies gelingen, wird mit dem „Enqueue Element“ VI der Wert, welcher in dem String „Element“ enthalten ist in die Queue geschrieben und die Queue mit dem VI „Release Queue“ wieder freigegeben.

Das VI „read_and_send_queue“, welches dauernd läuft überprüft bei jedem Durchlauf ob sich ein Element in der Queue befindet. Dies wird mit der Funktion „Get Queue Status“ realisiert. Diese stellt fest, wie viele Elemente in der Queue sind und schreibt sie in das Feld „# Elements in Queue“.



Mit einer Case Schleife wird nun überprüft, ob das Feld größer 0 ist, wenn ja wird mit der Funktion „Preview Queue Element“ der oberste Wert aus der Queue ausgelesen und mit der „VISA Write“ Funktion an den Mikrocontroller gesendet. Nur wenn dieser Vorgang fehlerfrei durchgeführt wird, wird mittels der Funktion „Dequeue Element“ der Wert aus der Queue gelöscht. Anschließend wird die Queue wieder freigegeben.

Somit kann sichergestellt werden, dass alle Befehle ohne Kollisionen oder Verluste an den Mikrocontroller weitergegeben werden.

8.4 Die pH-Regelung

Die Nährlösung im Fermenter muss auf einem konstanten pH-Wert gehalten werden, um bestmöglichen Wachstumsbedingungen für die Bakterien zu schaffen. Dies wird dadurch realisiert, dass Säure und Base durch Membranpumpen in die Fermentationskammer gepumpt werden. Hier bot sich die Möglichkeit an, einen PID Regler zu verwenden.

Der pH-Wert wird über ein separates pH-Meter welches über eine serielle Schnittstelle verfügt eingelesen.

Die pH-Regelung besteht aus 2 wesentlichen Teilen:

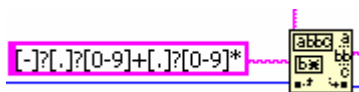
- Einlesen des pH-Wertes über ein externes pH-Meter
- Ausgabe der Pumpgeschwindigkeiten an den Mikrocontroller

8.4.1 Ermittlung des pH-Wertes über ein externes pH-Meter

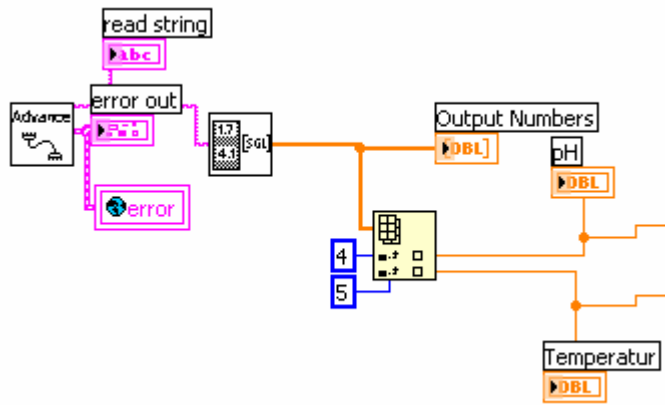
Das an den COM 1 Port des PC angeschlossene PH Meter wird über das „Advanced Serial Read“ VI ausgelesen. Dabei werden an der Seriellen Schnittstelle folgende Parameter eingestellt:

- Baud Rate: 9600
- Data Bits: 8
- Parity: None
- Stop Bits: 1.0
- Flow Control: RTS/CTS
- XON Character: 11
- XOFF Character: 13
- Termination Char: 10
- Timeout: 5000ms
- Bytes to Read: 57

Aus dem empfangenen String wird mittels des „Extract Values“ VI alle Zahlen extrahiert. Dieses VI erkennt mittels einer Regular Expression alle Zahlen in einem String.



Alle gefundenen Zahlen werden in ein Array geschrieben und aus diesem Array kann nun der pH-Wert extrahiert werden:

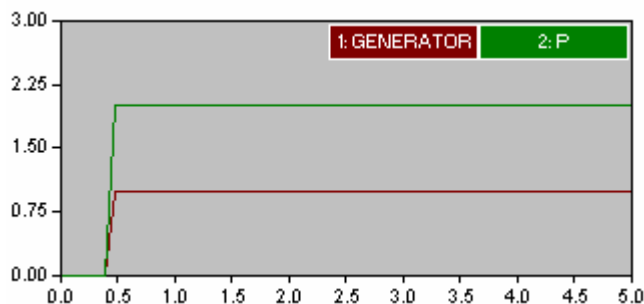


8.4.2 Der PID Regler

PID steht für Proportional – Integral – Differential Regler.

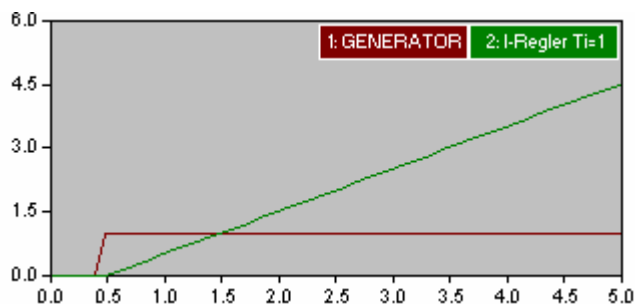
8.4.2.1 Der Proportionalregler (P-Regler)

Beim P-Regler ist das Ausgangssignal des Reglers proportional der Abweichung des Ist- Wertes vom Sollwert. Dieser Regler reagiert schnell auf Veränderungen der Regelgröße und ist stabil, er kann aber Störungen der Regelstrecke nicht gänzlich ausregeln und ist daher ungenau.



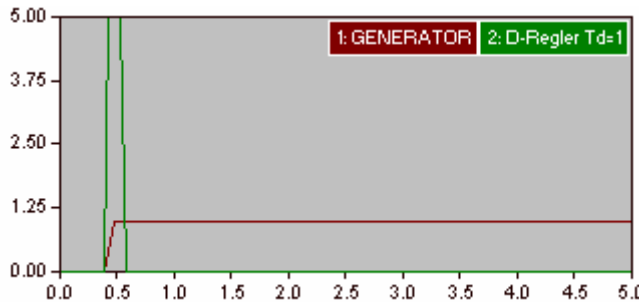
8.4.2.2 Der Integralregler

Beim I Regler ist das Ausgangssignal proportional zur zeitlichen Integration der Regelabweichung. Durch die Integration bekommt der Regler ein „Gedächtnis“. Somit hängt der Regelausgang von der Vergangenheit der Regelabweichung ab. Nur dadurch lassen sich Abweichungen vollständig kompensieren.



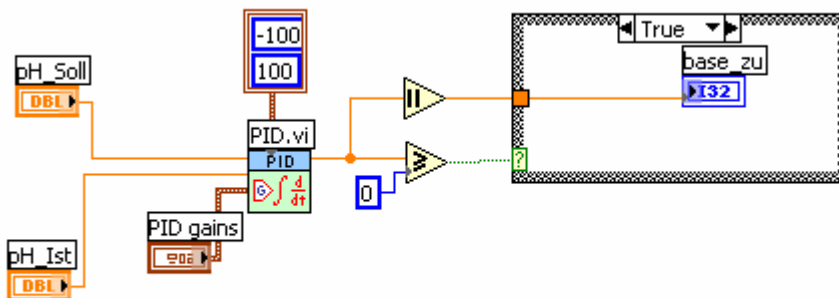
8.4.2.3 Der Differentialregler

Beim D-Regler ist das Ausgangssignal des Reglers proportional zur zeitlichen Änderung der Regelabweichung. Bei einer schnellen Abweichung vom Sollwert erfolgt eine starke Reaktion des D-Reglers. Bei einer kleinen Abweichung hingegen ist auch die Reaktion des Reglers geringer.



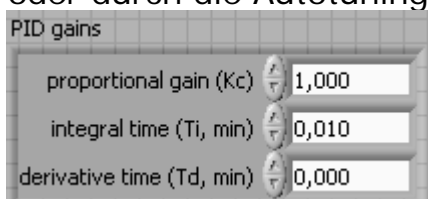
Der PID Regler kombiniert nun diese 3 Reglertypen und kann somit ihre gemeinsamen Vorteile nutzen. Die Nachteile der einzelnen Regler werden so durch die beiden anderen Regler ausgeglichen.

In LabVIEW können PID Regler mit dem PID Toolkit implementiert werden.



Hier werden dem PID VI 4 Parameter übergeben:

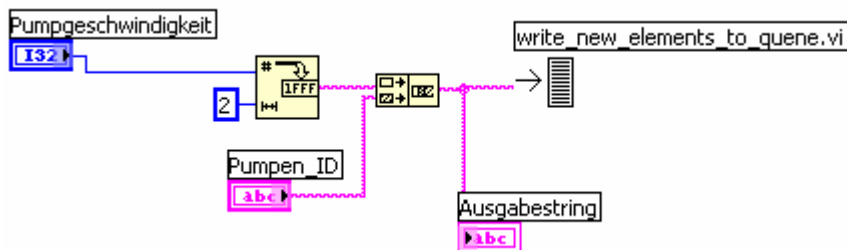
- Der Ist- Wert („pH_Ist“)
- Der Soll- wert („pH_Soll“)
- Ein Cluster, welches die Ober- und Untergrenzen des Ausgangssignals festlegt.
- Die „PID Gains“ welche den Proportional-, Integral- und Differentialfaktor des Reglers festlegen. Diese können entweder durch Versuche oder durch die Autotuning Funktion ermittelt werden.



Daraufhin wird überprüft ob das Ausgangssignal positiv oder negativ daher ist ob Lauge oder Säure zugegeben werden soll und anschließend wird dieser Wert zum Mikrocontroller gesendet.

Die Ober- und Untergrenzen des Ausgangssignals wurden so gewählt, dass 0 der kleinste und 100 der größte Wert ist. Dies ist insofern nützlich, da diese Werte nun nicht mehr aufbereitet werden müssen, da die Pumpe durch das digitale Potentiometer in 100 Geschwindigkeitsstufen betrieben werden kann.

Damit der Mikrocontroller die Werte auch verarbeiten kann, müssen diese noch ins HEX Format konvertiert werden.



Hierbei wird die Pumpgeschwindigkeit in einen zweistelligen Hexstring umgewandelt. Sollte die Ausgangszahl in HEX angegeben weniger als 2 Stellen haben wird mit Nullen aufgefüllt. Daraufhin wird noch die Pumpenkennung zum String hinzugefügt und der fertige Befehl der Queue übergeben.

8.5 Die Temperaturregelung

Die Temperaturregelung besteht aus 2 Teilen:

- Einlesen der Temperatur
- Senden des Steuersignals an den Mikrocontroller

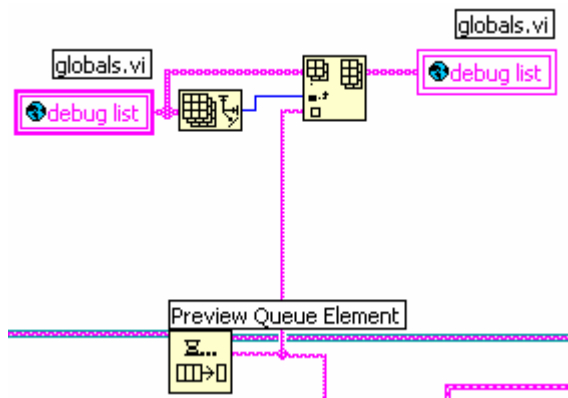
Das Einlesen der Temperatur erfolgt dadurch, dass der Mikrocontroller nach dem Senden eines bestimmten Steuerbefehls die Temperatur als Hexdezimalzahl zurückgibt. Diese muss nun in das Dezimalformat konvertiert werden.

Der so erhaltene Istwert wird laufen mit dem Sollwert verglichen. Die Regelung der Temperatur erfolgt nun mittels eines Ein/Aus Reglers. Dieser schaltet die Heizung des Thermostates Ein, sobald die Ist-Temperatur unter die Solltemperatur gefallen ist und deaktiviert die Heizspirale, wenn die Ist-Temperatur höher als die Solltemperatur ist.

8.5.1 Erstellung einer „Debug“ Liste

Um verschiedenen Fehlfunktionen der einzelnen Regeleinrichtungen schneller auf die Sprünge kommen zu können, werden diese einzeln bei jedem Sendevorgang in ein Array gespeichert.

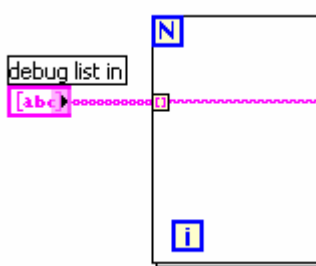
Dies geschieht in dem VI „read_and_send_queue“:



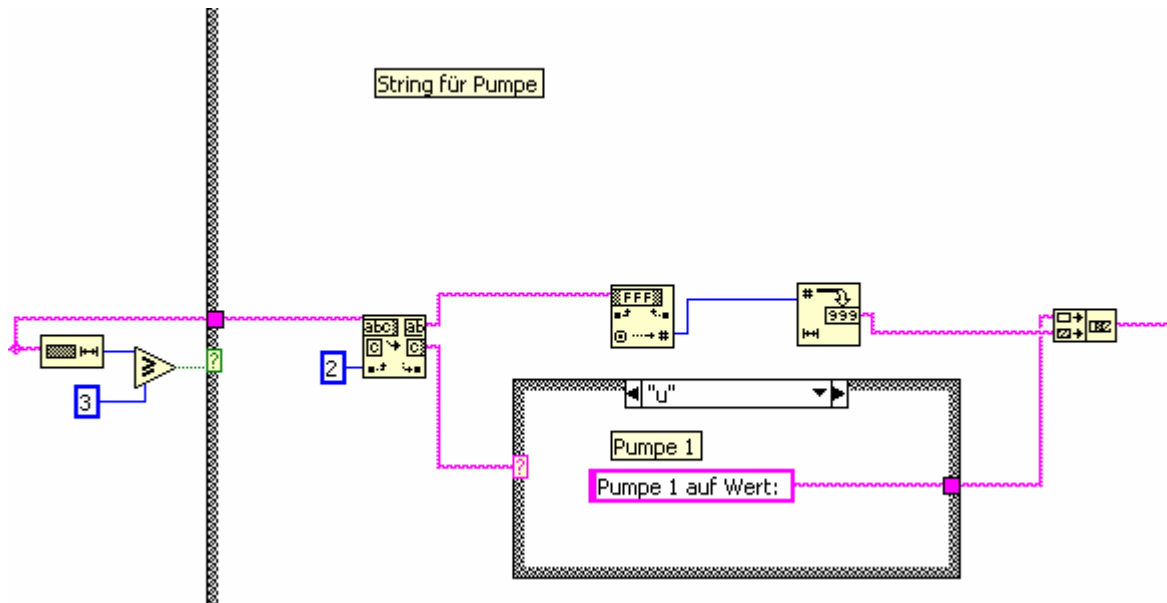
Zuerst wird mit der „Array Size“ Funktion ermittelt, wie viele Elemente sich im Array bereits befinden. Danach wird der zu sendende Befehl mit der „Insert Into Array“ Funktion an die letzte Stelle des Arrays eingetragen und das alte Array mit dem neuen überschrieben.

Da nun der Benutzer mit Befehlen wie „64u“ oder „W“ nicht viel anfangen kann, müssen diese in ein lesbares Format umgewandelt werden. Dies wird in dem VI „user_readable_debug_list“ durchgeführt.

Hierbei werden zuerst alle Array Elemente mittels einer For Schleife ausgelesen:



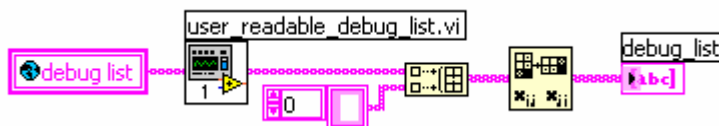
Danach wird überprüft, ob der jeweilige String mehr als 3 Stellen hat, wenn ja ist er ein Befehl für die Pumpe, wenn nein, ist es ein Befehl für das Einschalten bzw. Ausschalten eines Relais.



Bei einem Steuerbefehl für eine der Pumpen wird der String zuerst zerlegt, um die Pumpen ID und den Stellwert separat zu erhalten. Der Stellwert wird nun wieder ins Dezimalformat umgewandelt. Die Pumpenken- nung dient als Selektor für eine Case Schleife, in der der jeweiligen ID ein String wie „Pumpe 1 auf Wert: “ zugeordnet ist. Mit der „Merge String“ Funktion werden die beiden Strings noch kombiniert und ausgegeben.

Dieses VI dient als SubVI und erwartet als Eingabeparameter ein 1D Array vom Typ String und gibt denselben Datentyp wieder aus.

Um dieses Array nun als Liste im Hauptprogramm anzeigen zu können muss dem erhaltenen Array eine zweite Dimension hinzugefügt werden und danach mit der „Reshape 2D Array“ Funktion noch umgedreht werden.

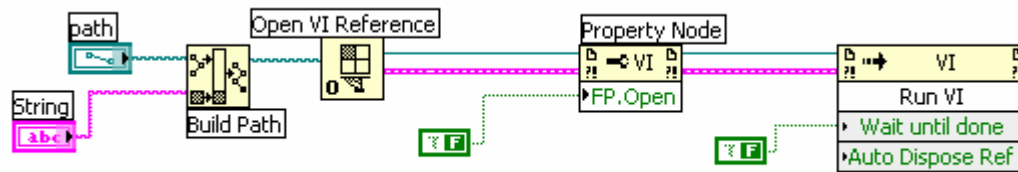


8.6 Kommunikation der einzelnen Subsysteme untereinander

8.6.1 Starten der Subsysteme

Da die einzelnen Regelungsprogramme asynchron zueinander laufen wurden sie als eigenständige Programme implementiert um eine Beeinflussung untereinander zu vermeiden.

Zuerst müssen die Subsysteme automatisch gestartet werden. Dies erfolgt mittels des VI Servers:



Hierbei wird zuerst aus dem Pfad des aktuellen VI's und dem im String übergebenen Dateinamen ein Pfad zu dem zu öffnenden VI erstellt und mit der „Open VI Reference“ Funktion eine Referenz zu diesem VI erzeugt. Mittels einer Property Node wird das Öffnen des Front Panels deaktiviert und mittels einer Invoke Node wird das VI gestartet. Nach dem Beenden des Hauptprogramms werden automatisch alles so geöffneten VI's gestoppt und aus dem Speicher entladen.

8.7 Der Mikrocontroller

8.7.1 Anforderungen an den Mikrocontroller

Der Mikrocontroller dient als Schnittstelle zwischen dem Steuerprogramm und der anzusteuern Hardware. Dabei hat er folgende Aufgaben zu erledigen:

- Kommunikation mit dem Steuerprogramm über die serielle Schnittstelle.
- Ansteuern von zwei digitalen Potentiometern und sechs Schaltrelais, die zum steuern der Pumpen benötigt werden.
- Messen der Temperatur des Fermenters über einen angeschlossenen Temperaturmessfühler.

Dazu wurde ein PIC16F877 Mikrocontroller der Firma Microchip verwendet. Dieser 40-Pin Chip ist ein Mikrocontroller der PIC16 Mikrocontroller-Familie und verfügt über eine Vielzahl von integrierten Fähigkeiten. Dazu zählen:

- „In-Circuit“ Programmierfähigkeit
- Debuggen während des Betriebes
- Interrupt – Fähigkeit
- Eingebaute Timer
- Analog – Digital Konverter
- Serielle Kommunikation

Der Hauptgrund warum ein Design mit Mikrocontrollern gewählt wurde, war: Mikrocontroller sind im Gegensatz zur anderen Alternative, dem Ansteuern mittels einer Controllerkarte der Firma National Instruments, die man mit LabView ansteuern kann, wesentlich billiger. Während eine LabView-Karte, die die Anforderungen erfüllt mehrere hundert Euro kosten würde, kostet ein Mikrocontroller, wie verwendet um die 10 Euro. Vermutlich könnte man zusätzlich sparen, wenn statt eines 16F877, der sozusagen das Luxusmodell der PIC16-Familie ist, einen Mikrocontroller verwenden.

den würde, der nur die Features unterstützt, die benötigt werden. Hier würde sich zum Beispiel ein PIC16F873 oder ein PIC16F876 anbieten.

8.7.2 Programmierung

Ein PIC-Mikrocontroller kann auf drei Arten programmiert werden:

- PIC-Assembler
- C/C++
- PIC-Basic

Zur Programmierung wurde Assembler ausgewählt, da man mit Assembler am besten vorhersagen kann, wie sich das Gerät verhalten wird und Assembler die einzige nicht kostenpflichtige Variante darstellt.

Die PIC-Assembler-Sprache basiert auf 35 Befehlen. Hier sind einige wichtige:

<i>nop</i>		<i>Keine Aktion</i>
<i>movf</i> <i>f, d</i>		<i>Kopiere den Wert in Adresse f nach W oder nach f</i>
<i>movwf</i>	<i>f</i>	<i>Kopieren den Wert in W nach f</i>
<i>movlw</i>	<i>l</i>	<i>Kopiere den konstanten Wert l nach W</i>
<i>bcf</i> <i>f, b</i>		<i>Setze Bit b von f auf 1</i>
<i>bsf</i> <i>f, b</i>		<i>Setze Bit b von f auf 0</i>
<i>addlw</i>	<i>l</i>	<i>Addiere W und die Konstante l</i>
<i>addwf</i> <i>f, d</i>		<i>Addiere f und W</i>
<i>sublw</i>	<i>l</i>	<i>Subtrahiere W von der Konstante l</i>
<i>subwf</i> <i>f, d</i>		<i>Subtrahiere f von W</i>
<i>btfss</i> <i>f, b</i>		<i>Teste Bit b von f und überspringe den nächsten Befehl, wenn es 1 ist</i>
<i>btfsc</i> <i>f, b</i>		<i>Teste Bit b von f und überspringe den nächsten Befehl, wenn es 1 ist</i>
<i>goto</i>	<i>l</i>	<i>Gehe zur Adresse l</i>
<i>call</i>	<i>l</i>	<i>Speichere die aktuelle Adresse im Stack und gehe</i>

zu /

return

Gehe zurück zur letzten Adresse im Stack

Beim Programmieren von PICs muss mit Registern gearbeitet werden. Register sind bestimmte Speicherbereiche, die eine spezielle Funktion erfüllen. So kann man zum Beispiel einzelne Pins des Mikrocontrollers unter Strom setzen, indem man im dementsprechenden Register ein Bit setzt.

```
bsf    PORTB,0    ; aktiviert den ersten Pin auf Port B
```

Beim Entwickeln der Software musste in erster Linie darauf Acht gegeben werden, dass der Mikrocontroller möglichst immer für eine Vielzahl von Signalen erreichbar ist. Würde man zum Beispiel während der Übertragung von Daten an die serielle Schnittstelle warten, so wäre der Controller für andere Peripherie unerreichbar. Daher wurde ein Design gewählt, das vollständig durch Hardware-Interrupts gelöst wurde.

Ein Interrupt ist ein Event, das dazu führt, dass der Mikrocontroller seine derzeitigen Aktionen abbricht, um einen bestimmten Programmteil auszuführen.

Das Grunddesign des Assemblerprogrammes:

```
Variablendeklarationen
Main-Code
    Initialisierungen
    Leere Endlosschleife
Interrupt-Code
Funktionen
```

8.7.3 Variablendeklaration

Die Variablendeklaration in Assembler:

```
TEMP_VAR          UDATA
hexstr            RES      2          ; 2 byte-register für
hex-zahl
hexval            RES      1          ; wert v. hex-zahl
hexerr            RES      1          ; Ist ein Fehler auf-
getreten?
hextmp            RES      1          ; temp. speicher
rcbuff            RES     16          ; 16 byte reception
buffer
rcpos             RES      1          ; position in rcbuff
txbuff            RES     16          ; 16 byte transmission
```

```

buffer
txpos          RES      1          ; position in txbuff
txleft        RES      1          ; bytes left for
transmission
byte          RES      1          ; byte for reading and
writing
resval_b      RES      1          ; sollwert v. Poti 1
restodo_b     RES      1          ; ausstehende steps
für Poti 1
resbit_b      RES      1          ; div.flags v. Poti 1
resval_a      RES      1          ; sollwert v. Poti 2
restodo_a     RES      1          ; überbleibende Steps
für Poti 2
resbit_a      RES      1          ; div. flags für Poti
2

```

8.7.4 Interupt Handling

Anschließend wurden die Hardware-Register des Mikrocontrollers eingestellt und der PIC in einer Endlosschleife gefangen, aus der er nur im Falle eines Interupts erwacht.

```

main
    ;sollte im idealfall leer bleiben
    goto    main          ; main loop

```

Bei einem Interupt wird folgende Funktion ausgeführt:

```

INT_VECTOR    CODE      0x004    ; interrupt vector location
    movwf    w_temp          ; save off current W register
contents
    movf    STATUS,w        ; move status register into W
register
    movwf    status_temp    ; save off contents of STATUS
register

; isr code can go here or be located as a call subroutine
elsewhere
    banksel PIR1
    btfsc   PIR1,TXIF      ; test for transmission interupt
    call    txhandle

    banksel PIR1

```

```

btfsc   PIR1,RCIF           ; test for reception interupt
call    rchandle

banksel  PIR1                 ; test for timer interupt
btfsc   PIR1,TMR1IF
call    tmr1handle

movf    status_temp,w       ; retrieve copy of STATUS regis-
ter
movwf   STATUS              ; restore pre-isr STATUS register
contents
swapf   w_temp,f
swapf   w_temp,w           ; restore pre-isr W register con-
tents
retfie                                ; return from interrupt

```

Der erste Teil speichert die Inhalte der Arbeitsregister W und STATUS in temporären Variablen. Im zweiten Teil wird nacheinander überprüft, wo der Interupt stattgefunden hat. Das wird über die Interupt-Flags in PIR1 gemacht. Sollte ein Interupt an der überprüften Stelle stattgefunden haben, wird mit **call** in eine entsprechende Funktion zu deren Bearbeitung gewechselt.

Folgende Interupts können auftreten:

- Sende-Interupt
Tritt auf, wenn die Übertragung eines Bytes über die serielle Schnittstelle abgeschlossen wurde. Das ist insofern praktisch, da man nun sofort das Senden eines weiteren Bytes in der Warteschlange in Auftrag geben kann.
- Empfangs-Interupt
Tritt auf, wenn ein Byte über die serielle Schnittstelle abgeschlossen wurde. Im Interupt kann man dieses Byte nun auslesen, abspeichern und auswerten.
- Timer1-Interupt
Tritt auf, wenn der unabhängige Timer 1 überläuft. Das geschieht in regelmäßigen Abständen. Der Timer wird für das Ansteuern der digitalen Potentiometer benötigt.

Anschließend werden die ursprünglichen Werte wieder in die Arbeitsregister geschrieben und der Interupt beendet.

Für die Kommunikation zwischen dem Mikrocontroller und dem Steuerprogramm einigte man sich auf ein sehr einfaches Protokoll. Dabei war es wichtig, dass dieses Protokoll für den Mikrocontroller möglichst einfach zu interpretieren ist. Deshalb wurden einzelne Buchstaben als Kommandos gewählt. Da auch Hexadezimalzahlen übertragen werden sollten und diese neben Nummern auch die Buchstaben A bis F enthalten, mussten für die Kommandos Buchstaben aus dem oberen Alphabet genommen werden.

<i>Command</i>	<i>Verwendung</i>	<i>Wirkung</i>
<i>t</i>	<i>t</i>	<i>Fragt die Temperatur ab</i>
<i>u</i>	<i>XXu</i>	<i>Steuert Potentiometer 1</i>
<i>v</i>	<i>XXv</i>	<i>Steuert Potentiometer 2</i>
<i>w/W</i>	<i>w oder W</i>	<i>Steuert Relais 1</i>
<i>x/X</i>	<i>x oder X</i>	<i>Steuert Relais 2</i>
<i>y/Y</i>	<i>y oder Y</i>	<i>Steuert Relais 3</i>
<i>z/Z</i>	<i>z oder Z</i>	<i>Steuert Relais 4</i>

Nachdem ‚t‘ übermittelt wurde, antwortet der PIC mit zwei Hexadezimal-Ziffern, die die derzeitige Temperatur angeben.

Bei den Potentiometer-Befehlen muss vor dem eigentlichen Befehl noch ein Wert mit dem gewünschten Widerstand übertragen werden. Dieser muss aus zwei Hexadezimal-Ziffern bestehen. Hexadezimal deshalb, weil dieses Zahlensystem besonders angenehm in einen binären Wert konvertierbar ist.

Zum steuern der Relais muss man entweder einen Groß-, oder Kleinbuchstaben übertragen. Mit einem Kleinbuchstaben deaktiviert man das Relais und mit einem Großbuchstaben aktiviert man es.

8.7.5 String Conversion

Zur Konvertierung eines Strings wurde ein eigener Satz von Funktionen und Variablen angelegt. Dazu gehören die Variablen `hexval` und `hexstr`, die einen Hexadezimal-Wert jeweils als Zeichenfolge oder als Zahl enthalten, und die beiden Funktionen `str2val` bzw. `val2str` die von der einen Form in die andere umwandeln und im jeweiligen Gegenstück speichern.

Beispiel:

```

movlw    ,A'           ; hexstr = ,AB'
movwf    hexstr
movlw    ,B'
movwf    hexstr+1
call     str2val       ; Konvertiere hexstr in hexval
btfsc    hexerr,0     ; Teste auf mögliche Fehler
goto     error        ; Fehlerbehandlung
movfw    hexval       ; hexval ist nun 0xAB

```

Die Konvertierung von einer Zahl in eine Zeichenkette ist relativ einfach. Dabei wird das Byte zuerst in zwei 4-bit Teile geteilt und diese werden in Strings konvertiert indem man sie solange verringert, bis sie Null sind. An der Stelle an der sie Null sind, wird das dementsprechende Zeichen in die Ziffer des Strings eingetragen.

```

val2str                                     ; konvertiert einen Hex-Wert in einen
Hex-String
banksel hexval
swapf    hexval,w
andlw    0x0f
movwf    hextmp           ; Erster Teil
btfss    STATUS,Z
goto     $+4
movlw    '0'
movwf    hexstr
goto     vs_next
decfsz   hextmp
goto     $+4
movlw    '1'
movwf    hexstr
goto     vs_next
decfsz   hextmp
goto     $+4
movlw    '2'
movwf    hexstr
goto     vs_next
decfsz   hextmp
goto     $+4
movlw    '3'
movwf    hexstr

```

```

goto    vs_next
decfsz  hextmp
goto    $+4
movlw   '4'
..
...
```

Die Konvertierung von einer Zeichenfolge in einen Wert ist schwieriger, weil es eine Lücke zwischen ‚9‘ und ‚A‘ gibt. Außerdem kann es sein, dass ein Wert fehlerhaft ist und andere Zeichen als 0-9 und A-F enthält, worauf auch Acht gegeben werden muss. Deshalb muss man mehrere ineinander verschachtelte Überprüfungen machen:

chkchar

```

                                nop
                                movfw   INDF           ; smaller than '0'
then error
                                sublw   '0'           ; '0' - x
                                btfsc   STATUS,C       ; skip if 0 -> nega-
tive
                                goto    cberror
                                movfw   INDF           ; smaller than '9'
then number
                                sublw   '9'           ; '9' - x
                                btfsc   STATUS,C       ; skip if 0 -> nega-
tive
                                goto    cbnum
                                movfw   INDF           ; smaller than 'A'
then error
                                sublw   'A'           ; 'A' - x
                                btfsc   STATUS,C       ; skip if 0 -> nega-
tive
                                goto    cberror
                                movfw   INDF           ; smaller than 'F'
then character
                                sublw   'F'           ; 'F' - x
                                btfsc   STATUS,C       ; skip if s -> nega-
tive
                                goto    cbchar

cberror
                                ; else error
                                movlw   0xff
                                movwf   hexerr       ; FEHLER
                                return
```

cbchar

```
movlw    'A'
subwf    INDF,W           ; f-w+10 -> char-
'A'+10

addlw    0xA
movwf    hextmp
movlw    0x00
movwf    hexerr         ; kein fehler
return
```

cbnum

```
movlw    '0'
subwf    INDF,W           ; f-w -> char-'0'
movwf    hextmp
movlw    0x00
movwf    hexerr         ; kein fehler
return
```

8.7.6 Serielle Kommunikation

Bei der Implementierung der seriellen Kommunikation musste auch darauf Acht gegeben werden, dass es zu keinem Verlust von Daten kommen kann. Daher wurde sowohl für die Sender-, als auch für die Empfänger-Einheit des Programms ein 16 Byte lange Puffer angelegt, der mit speziellen Funktionen beschrieben und ausgelesen wird. Dadurch können im Normalfall Engpässe vermieden werden. Sollte es einmal dazu kommen, dass einer der Puffer voll ist, so wird die seine Entleerung erzwungen, indem Interrupts zwischenzeitig deaktiviert werden und der Inhalt des Puffers abgearbeitet wird.

Für die Puffer-Bearbeitung wurden folgende Funktionen angelegt:

<i>rcbufferwrite</i>	<i>Ein Byte wird in den Empfangs-Puffer geschrieben</i>
<i>rcbufferread</i>	<i>Ein Byte wird aus dem Empfangs-Puffer gelesen</i>
<i>rcbufferdec</i>	<i>Der Empfangs-Puffer wird ein byte zurückgespult</i>
<i>rcbufferinc</i>	<i>Der Empfangspuffer wird ein Byte vorgepult</i>
<i>txbufferwrite</i>	<i>Ein Byte wird in den Sende-Puffer ge-</i>

geschrieben und, falls möglich, gesendet

txbufferread

Ein Byte wird aus dem Empfangspuffer gelesen und gesendet.

Um in den Puffer schreiben und aus ihm lesen zu können, muss man eine zusätzliche Funktion verwenden, da PIC-Assembler keine Zeiger-Arithmetik unterstützt. Will man also auf eine bestimmte Adresse plus eines gewissen Offsets zugreifen, muss man indirekte Adressierung verwenden. Indirekte Adressierung benötigt zwei Hardware-Register. Einerseits das FSR-Register, in das die Adresse geschrieben wird, und andererseits das INDF-Register, das den Wert der Adresse im FSR-Register enthält, und ausgelesen und beschrieben werden kann.

Beispiel:

```
rcbufferread    ; read the content of rcbuff at rcpos to byte
                banksel rcpos
                movlw  0x1
                subwf  rcpos,W           ; byte =
rcbuff[(rcpos-1) & 0xf]
                andlw  0xf
                addlw  rcbuff
                movwf  FSR
                movfw  INDF
                banksel byte
                movwf  byte
                return
```

Hier wird zuerst rcpos mit der Adresse von rcbuff addiert und dieser Wert anschließend in FSR kopiert. Danach kann über INDF auf den Wert in rcpos+rcbuff zugegriffen werden.

Das eigentliche Senden und Empfangen von Daten erfolgt über die im Mikrocontroller integrierte RX/TX Funktion. Dafür muss man am Anfang die für eine Übertragung wichtigen Werte (Baud-Rate, Stop-Bits, Synchron/Asynchron,...) in einem Register angeben. Außerdem müssen Interrupts für die serielle Kommunikation aktiviert werden:

```
;serIO
                banksel SPBRG           ;SPBRG: .25 -> 9.615 kBaud
                movlw  .25
                movwf  SPBRG
                banksel TXSTA           ;TXSTA: BRGH -> high Baud
rate
                clrf  TXSTA
                bsf  TXSTA,5
```

```

    bsf      TXSTA, 2
banksel RCSTA          ;RCSTA: SPEN
    clrf    RCSTA
    bsf      RCSTA, 7
banksel PIE1          ;PIE1: RCIE, TXIE, TMR1IE
    clrf    PIE1
    bsf      PIE1, RCIE
    bsf      PIE1, TXIE
    bsf      PIE1, TMR1IE
banksel RCSTA          ;RCSTA: CREN -> enable the
serialIO
    bsf      RCSTA, 4

```

Einzelne Bytes werden dann innerhalb der Interrupts an das Steuerprogramm gesendet und von ihm empfangen. Dazu wird ebenfalls ein spezielles Hardware-Register genutzt:

```

rchandle
banksel RCREG          ; read byte
movfw  RCREG
banksel byte
movwf  byte
call   rcbufferwrite ; byte -> rcbuffer
banksel RCSTA
btfs   RCSTA, OERR   ; overrun error
goto   rcfailure
btfs   RCSTA, FERR   ; framing error
goto   rcfailure
call   rcinterpreter ; start the rc Command-Interpreter
goto   rcend

```

Diese Funktion ist die Funktion, die vom Empfangs-Interrupt ausgelöst wird. Zuerst wird das empfangene Byte aus dem Register RCREG ausgelesen und in der Variable byte gespeichert. Diese wird dann mit der Funktion rcbufferwrite in den Empfangs-Puffer kopiert. Anschließend wird eine Fehler-Überprüfung durchgeführt, die testet, ob während der Übertragung ein Fehler aufgetreten ist. Sollte dies der Fall sein, wird in die Funktion rcfailure gewechselt, in der das gerade gelesene Byte verworfen und der Empfang neu gestartet wird. Wenn kein Fehler auftritt, wird das empfangene Byte an den Kommando-Interpreter übergeben.

Die Interpretation der Einzelnen Kommandos wird in einer eigenen Funktion, die durch den Empfangs-Interrupt ausgelöst wird, erledigt. Dabei gibt

es das Problem, dass es keinen Maschinen-Befehl zum direkten Vergleichen von zwei Werten gibt. Daher muss man einen Trick anwenden.

Wenn das Ergebnis einer Rechnung Null ergibt, wird das so genannte Z Bit im STATUS-Register aktiviert. Das bedeutet, dass man, wenn man zwei Werte auf Gleichheit prüfen will, zuerst einen Wert vom anderen subtrahiert, und anschließend überprüft, ob das Ergebnis dieser Subtraktion Null ist:

```
movlw    1                ; 2 - 1 = -1
sublw    2
btfsc    STATUS,Z        ; Z = 0, weil nicht 0
call     ungleich        ; 1 ungleich 2
```

Die vollständige Testfunktion sieht so aus:

```
rcinterpreter                ; check for commands
over COM
                                ; read the most recent
                                ; test for u and set potentiometer 1 if suc-
byte from rcbuff                ; cess
                                ; test for v and set potetniometer 2 if suc-
                                ; cess
                                banksel byte
                                movfw    byte
                                sublw    'u'
                                btfss    STATUS,Z
                                goto     rcinterpreter_1
                                call     fillhexstr        ; copy the 2 bytes be-
                                ; fore the command-byte to hexstr
                                call     poti_1            ; start potentiometer
1
rcinterpreter_1
                                ; test for v and set potetniometer 2 if suc-
                                ; cess
                                banksel byte
                                movfw    byte
                                sublw    'v'
                                btfss    STATUS,Z
                                goto     rcinterpreter_2
                                call     fillhexstr        ; copy the 2 bytes be-
                                ; fore the command-byte to hexstr
                                call     poti_2            ; start potentiometer
2
rcinterpreter_2
                                ; test for w and clear port B3 if success
                                banksel byte
```

```

movfw   byte
sublw   'w'
btfss   STATUS,Z
goto    $+4
banksel  PORTB
bcf     PORTB,3
; test for x and clear port B4 if success
banksel  byte
movfw   byte
sublw   'x'
btfss   STATUS,Z
goto    $+4
banksel  PORTB
bcf     PORTB,4
; test for y and clear port A3 if success
banksel  byte
movfw   byte
sublw   'y'
btfss   STATUS,Z
goto    $+4
banksel  PORTA
bcf     PORTA,3
; test for z and clear port A4 if success
banksel  byte
movfw   byte
sublw   'z'
btfss   STATUS,Z
goto    $+4
banksel  PORTA
bcf     PORTA,4
; test for W and set port B3 if success
banksel  byte
movfw   byte
sublw   'W'
btfss   STATUS,Z
goto    $+4
banksel  PORTB
bsf     PORTB,3
; test for X and set port B4 if success

```



```

bankssel byte
movfw   byte
sublw   'X'
btfss   STATUS,Z
goto    $+4
bankssel PORTB
bsf     PORTB,4
; test for Y and set port A3 if success
bankssel byte
movfw   byte
sublw   'Y'
btfss   STATUS,Z
goto    $+4
bankssel PORTA
bsf     PORTA,3
; test for Z and set port A4 if success
bankssel byte
movfw   byte
sublw   'Z'
btfss   STATUS,Z
goto    $+4
bankssel PORTA
bsf     PORTA,4
return

```

8.7.7 Potentiometer

Die Potentiometer werden über die Funktion poti_x gesteuert. Wenn sie ausgeführt wird, werden zuerst die beiden Hexadezimal-Zahlen zu einem Wert umgerechnet. Dieser wird in ein dazu gehöriges Register geschrieben. Anschließend werden die einzelnen Werte, die nötig sind initialisiert, und die Funktion beendet. Die eigentliche Ansteuerung der Potentiometer findet dann im Timer Interrupt statt.

```

poti_1
call    str2val           ; konvertiere den hex-String zu einem
hex-Wert
btfsc   hexerr,0         ; hex-Wert fehlerhaft -> Abbruch
goto    poti_1_end
btfsc   resbit_b,2       ; Potentiometer gerade im Betrieb ->
Abbruch
goto    poti_1_end

```

```

movfw   hexval           ; hex-Wert wird in resval_b beschrie-
ben
banksel  resval_b
movwf   resval_b
bcf     PORTB,2
bcf     PORTB,1
bsf     resbit_b,0       ; start INC with high
bsf     resbit_b,1       ; start with setting the resistance to
0
bsf     resbit_b,2       ; reserve potentiometer
movlw   0x64
movwf   restodo_b
return

```

Das Potentiometer wird über 3 Pins gesteuert. Diese heißen INC, U/D und CS. Um das Potentiometer überhaupt ansteuerbar zu machen muss CS auf 0 gestellt sein. Mit INC kann man entscheiden, ob man sich nach oben oder nach unten bewegt und mit einem Puls von INC (low -> high -> low) macht man einen Schritt.

Zuerst wird der Widerstand auf 0 gesetzt, um von dort aus den entsprechenden Wert einstellen zu können. Das wird alles innerhalb des Timer-Interrupts gemacht:

```

banksel  restodo_a         ; communication with potentiometer 1
        movfw   restodo_a     ; ist restodo 0?
        sublw   0
        btjsc   STATUS,Z
tmr1h_p2_2      goto         tmr1h_p2_2           ; dann gehe zu
tmr1h_p2_2
        banksel  resbit_a         ; high oder low?
        btfs   resbit_a,0
        goto   tmr1h_p2_u         ; gehe zu tmr1h_p2_u
tmr1h_p2_d
        banksel  PORTA           ; INC -> low
        bcf     PORTA,0
        banksel  resbit_a
        bcf     resbit_a,0
        banksel  restodo_a
        decf   restodo_a
        goto   tmr1h_p1
tmr1h_p2_u
        banksel  PORTA           ; INC -> high
        bsf     PORTA,0

```

```

banksel resbit_a
bsf    resbit_a,0
goto   tmr1h_p1

tmr1h_p2_2

banksel resbit_a
btfss  resbit_a,1      ; phase 1 or 2?
goto   tmr1h_p2_3      ; phase 2 -> no action
                          ; phase 2:

banksel resval_a
movfw  resval_a        ; restodo=resval
banksel restodo_a
movwf  restodo_a
banksel PORTA
bsf    PORTA,1         ; set U/D to UP
banksel resbit_a
bcf    resbit_a,1      ; set to phase 2
goto   tmr1h_p1

tmr1h_p2_3                          ; nothing left to do:

banksel PORTA
bsf    PORTA,2         ; deactivate potenti-
ometer

banksel resbit_a
bcf    resbit_a,2      ; end reservation

```

Hier wird zwischen Phase 1 und 2 unterschieden. Phase 1 ist das Nullstellen des Widerstandes und Phase 2 ist das eigentliche Einstellen des gewünschten Widerstandes. Zwischen diesen beiden Phasen wird resval, die den gewünschten Widerstand enthält nach restodo kopiert und U/D wird auf UP gestellt. Wenn Phase 2 beendet ist, kann die Reservierung des Ports aufgehoben und CS wieder auf 1 und somit auf inaktiv gestellt werden.

8.7.8 Temperaturmesszelle

Der Ansteuerungscode für eine Temperaturmesszelle konnte nicht mehr implementiert werden. Hier würden sich allerdings mehrere Ansätze anbieten.

Einerseits könnte man einen Temperaturfühler mit analogem Ausgang verwenden, der über den im PIC integrierten Analog/Digital Wandler gemessen wird. Hier wäre es allerdings ein Nachteil, dass es zu Spannungsverlusten im Kabel kommen kann.

Eine andere Methode ist es, einen seriellen Messfühler zu benutzen. Hier gab es auch bereits Erfolge, bei der Kommunikation mit einem TC74 der

Firma Microchip, der über das Protokoll I²C kommuniziert. Allerdings war dieser Temperaturfühler nicht mehr erhältlich.

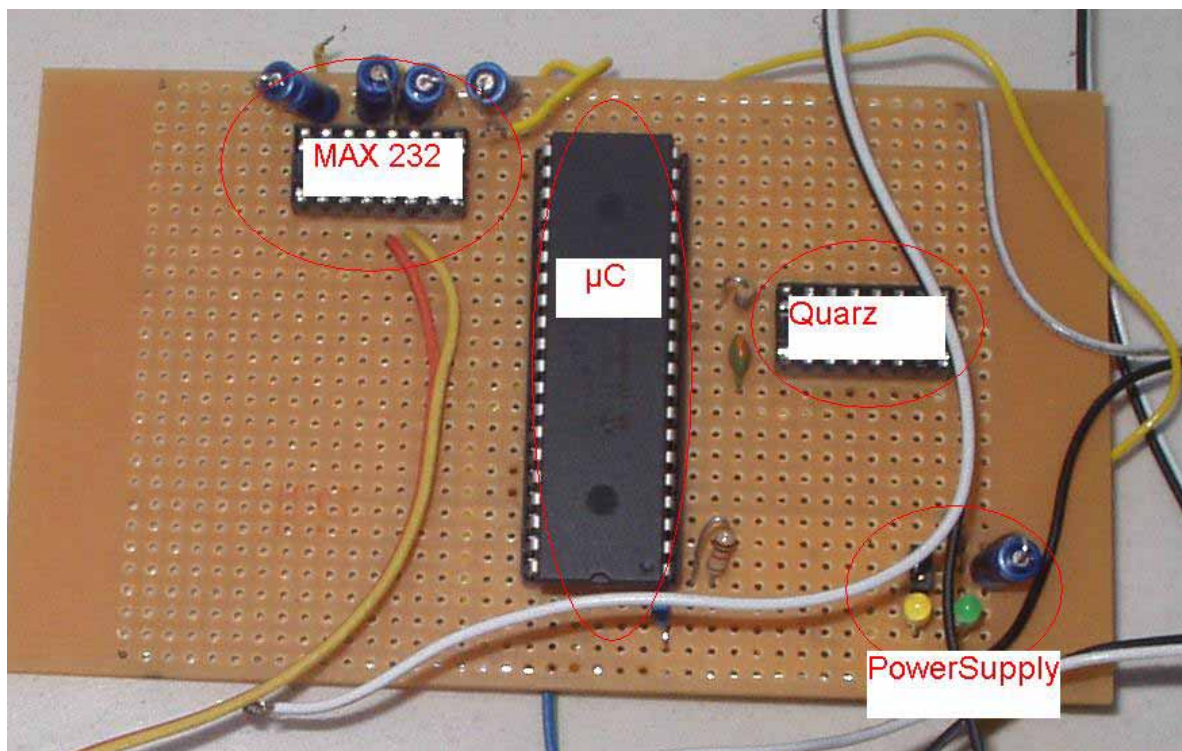
Deshalb wurde erst die Kommunikation mit dem verwandten Temperaturfühler TCN75 ausprobiert und nachdem diese nicht erfolgreich war auf ein Konkurrenzprodukt der Firma Dallas Semiconductors, einen DS1820, umgestiegen. Dieser verwendet einen one-wire bus zur Übertragung. Leider blieb die Ansteuerung beider Geräte erfolglos. Ob es sich darum um einen Software-, oder Hardware-Fehler gehandelt hat, konnte nicht geklärt werden.

8.8 Die Elektronik

Es sollte eine geeignete Schaltung entwickelt werden auf der die Messumformer und die nötigen Elektronikkomponenten Platz finden.

Entwicklungsschaltung mit μ Prozessor

8.8.1 Das Mikroprozessor-Board

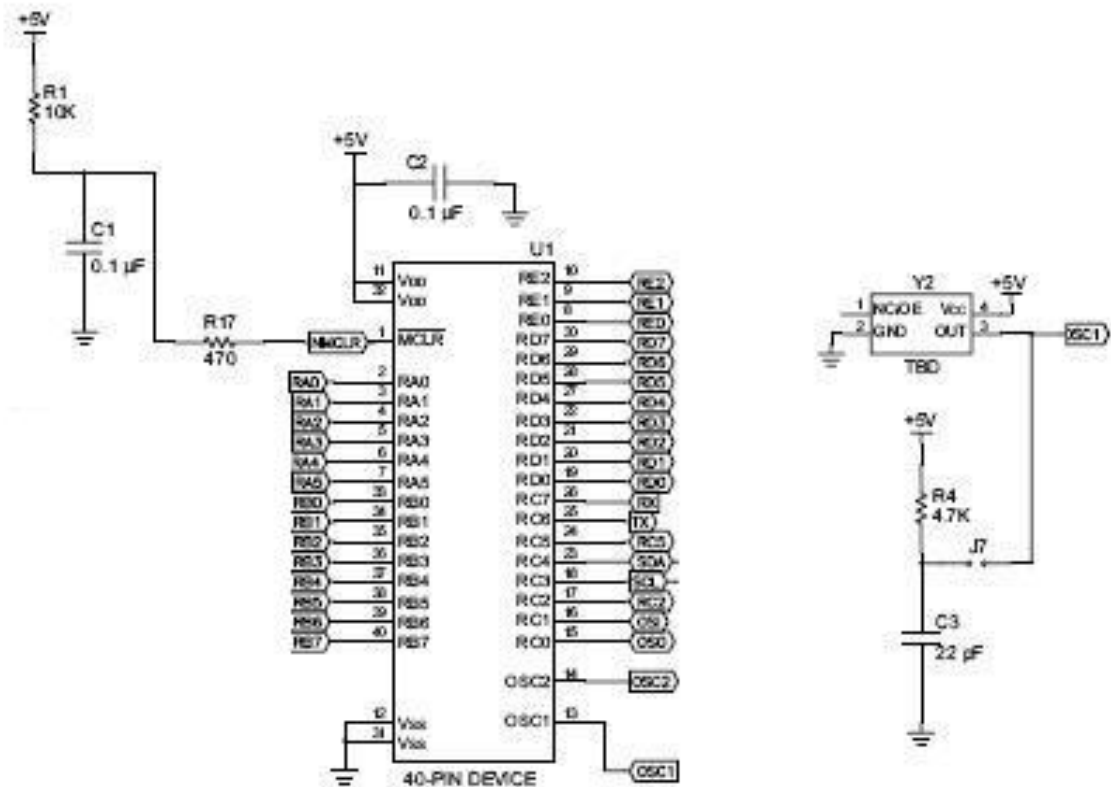


8.8.1.1 Anforderungen

- 40-Pin Mikroprozessor
- RS232 Treiber Baustein
- Spannungsstabilisierung
- Optokoppler
- Taktgenerator

- Digitales Potentiometer
- Relais → Auslagerung auf 2 Platine

8.8.2 Schaltplan Mikroprozessor und Taktgenerator



8.8.2.1 Taktgenerator

Es kam ein 4 MHz Oszillator-Modul (Schaltplan TB0) zum Einsatz da dieses in der Anwendung einfacher ist und weniger Verdrahtungsaufwand in der Schaltung erzwingt. Dieses Modul musste mit +5V und GND Spannung versorgt werden um am OUTPUT Pin eine 4 MHz Taktfrequenz zu erhalten.

Es waren jedoch 2 zusätzliche Bauteile notwendig R4 im obigen Schaltbild 4,7 kOhm zwischen TB0_Out und +5V und ein Kondensator C3 22pF zwischen TB0_Out und GND als Pullup-Schaltung der Taktrate.

8.8.2.2 Spannungsversorgung

Es kam ein Wechselstromtransformator zum Einsatz der den Strom für einen 6A – Brückengleichrichter von AC 230V auf AC 12V transformierte diese nach dem Brückengleichrichter 12V DC wurden durch Einbringung von 2 höher kapazitiven Kondensatoren weitergeglättet. Diese 12V Versorgung konnte nun auch für diverse Relais als Spannungsquelle genutzt werden.

Messungen konstatierten eine sehr konstante +12V Spannungsversorgung, die um den Mikroprozessor zu versorgen noch auf 5 V begrenzt werden musste. Dies gelang mit so genannten Linear-Spannungswandlern. Dieser musste in anbetracht der hohen Spannungsdifferenz bereits mit einem Kühlblech verbunden werden um die auftretende Verlustleistung abführen zu können. Danach wurde wieder ein Kondensator (Elektrolyt 1 μ F) in den Stromkreis als Puffer und Spannungsstabilisator eingebracht.

8.8.2.3 Mikroprozessor

Die Spannungsversorgung musste an insgesamt 4 Pins des Mikroprozessors erfolgen. Pin 11 Vdd und Pin 32 Vdd wurden mit +5 Volt versorgt wobei direkt vor dem Mikroprozessor ein 0,1 μ F Kondensator eingesetzt wurde um Lokale Spannungsabfälle auf der Platine und damit einen unsauberen Reset des Mikroprozessors zu verhindern. Pin 12 und 31 wurden mit Ground verbunden.

Weiters wurde Pin 1 MCLR über einen 10KOhm Widerstand an +5V gelegt um ein Reset des Mikroprozessors beim Einschalten des Gerätes zu garantieren, dabei wurde ein weiterer 100nF Kondensator zwischen MCLR und GND geschaltet um eine saubere Taktentkoppelung zu gewährleisten.

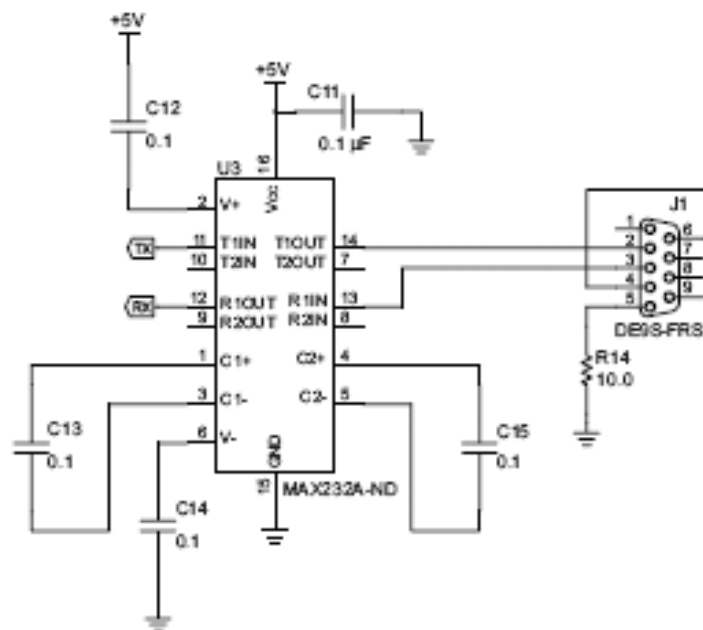
Die Taktfrequenz nahm der Mikroprozessor am Pin 13 vom Oszillator-Modul entgegen.

8.8.2.4 Optokoppler

Um den Mikroprozessor vor zerstörerischer Spannung und zu hohen Strömen zu schützen verwendeten wir Optokoppler die im Falle eines Fehlstromes oder einer Überspannung als schwacher Bauteil den Mikroprozessor vor Schaden zu bewahren.

Alle relaischaltende Ausgänge wurden auch im Gedanken an die zu schaltenden 230 V und die unwahrscheinliche aber dennoch mögliche Gefahr einer Kopplung des Mikroprozessor Board mit dem 230 V Netz mit Optokoppler ausgerüstet um eine galvanische Trennung zu erreichen.

8.8.2.5 RS-232 Pegelanpassung



Da der RS-232 Standard mit +12V und -12V Pegeln arbeitet und die serielle Schnittstelle im Mikroprozessor mit +2,4 bis +5V für eine logische Eins musste ein Treiberbaustein eingesetzt werden um diese Spannungen für die Kommunikation mit dem PC zu erzeugen. Dabei kam ein Maxim MAX232CPE zum Einsatz der nur 4 Externe Kondensatoren (100nF) benötigt. Die Kommunikation zwischen Mikroprozessor und PC über diese Serielle Schnittstelle stellte sich als nicht funktionstüchtig heraus.

Es wurden deshalb auch Transistor Pegelanpassungen aufgebaut die jedoch auch nicht den gewünschten Erfolg erzielten.

8.8.2.6 Die Relais Platine

Diese Schaltung wurde auf einer weiteren Platine aufgebaut da auf ihr Relais 230 V Schalten und diese Spannung unter keinen Umständen mit dem Mikroprozessor und in weiterer Folge mit dem Computer in Kontakt kommen sollte. Dazu wurden die Optokoppler auf dem Mikroprozessor-Board angebracht um eine völlige Galvanische Trennung zu erreichen.

Die Relais besitzen Werkseitig leider keine optische Zustandskontrolle die deshalb durch eine LED nachgerüstet wurde.

Da die Ausgänge des PIC nun durch Optokoppler geschützt waren konnte aus denselben nicht der notwendige Strom zum Schalten der Relais bezogen werden. Dadurch wurde ein Transistor PULL-UP Schaltung implementiert die durch +12V gespeist wurde und den nötigen Strom zum Schalten

der Relais liefern konnte. Durch diese Schaltung ist nur eine sehr geringe Spannung $>2\text{ V}$ notwendig um ein Relais auszulösen.

8.8.3 Gerätemodifikationen

8.8.3.1 Pumpen

Es wurde in den 2 Pumpen die für die pH-Regelung eingesetzt werden Schnittstellen eingebaut um über eine 2 Drahtleitung und die beiden digitalen Potentiometern die auf der Mikroprozessorplatine angebracht waren den Widerstand respektive die Pumpgeschwindigkeit zu steuern.

Eine weitere Pumpe, aus Kostengründen eine ausrangierte Autowaschanlagenpumpe (12V), wurde durch ein Relais schaltbar und als Nährlösungspumpe eingesetzt.

8.8.3.2 Thermostat

Es mussten umfangreichere Modifikationen an dem Gerät durchgeführt werden denn es bestand der Wunsch dieses Gerät auch ohne Fermenter zu benutzen. Deshalb musste eine Art Dongle entwickelt werden der die Funktionen des Gerätes je nach eingestecktem Dongle bestimmt. So konnten wir einerseits über den Mikroprozessor die Temperatur regeln, andererseits konnte mit eingestecktem Dongle die normale Analoge Steuerungsfunktion des Gerätes genutzt werden.

Die Digitalisierung des Thermometers gestaltete sich nicht weiter schwierig da nur 2 Bauteile hinzugefügt werden mussten. Ein Digitales Thermometer sowie ein Transistor der die Schaltung des Heizregisterrelais übernahm.

8.8.3.3 Rührer

Der Rührer konnte nicht steuerbar gemacht werden da die elektrische Leistung zu hoch war. Es wäre mit digital-steuerbaren Motortreibern möglich gewesen doch wäre der Kosten/Nutzen Faktor sehr gering. Es kann jedoch über ein Relais der Rührer An und Aus geschaltet werden.

8.8.4 Datenblätter

8.8.4.1 L7800 Series Datasheet

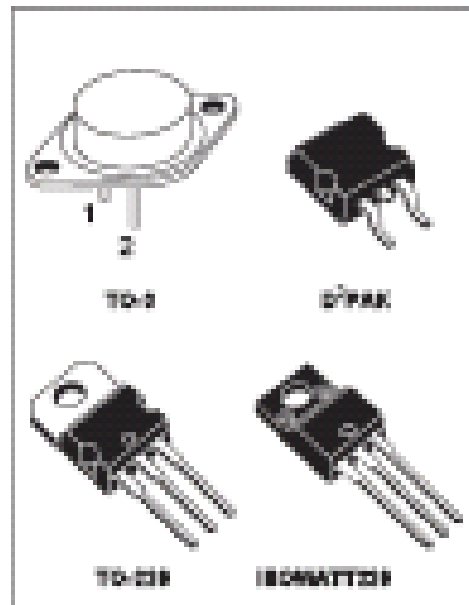


POSITIVE VOLTAGE REGULATORS

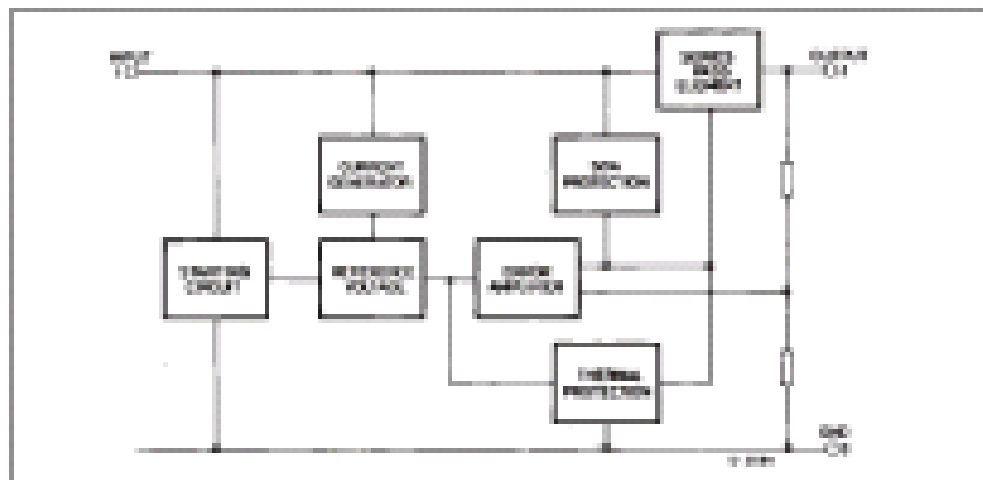
- OUTPUT CURRENT UP TO 1.5A
- OUTPUT VOLTAGES OF 5; 6.2; 8; 9; 10; 12; 15; 18; 24V
- THERMAL OVERLOAD PROTECTION
- SHORT CIRCUIT PROTECTION
- OUTPUT TRANSIENT SOA PROTECTION

DESCRIPTION

The L7800 series of three-terminal positive regulators is available in TO-220 (SOT223), TO-18 and DIPAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distributed problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



BLOCK DIAGRAM





TCN75

2-Wire Serial Temperature Sensor and Thermal Monitor

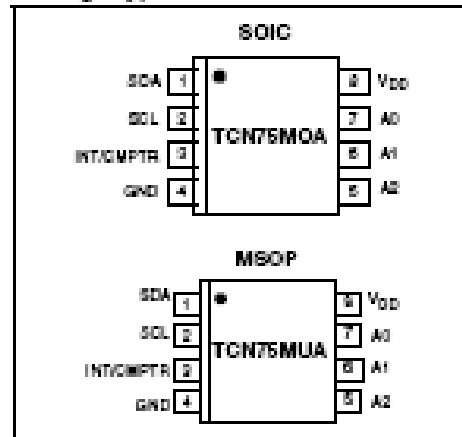
Features

- Solid-State Temperature Sensing; 0.5°C Accuracy (Typ.)
- Operates from -55°C to +125°C
- Operating Supply Range: 2.7V to 5.5V
- Programmable Trip Point and Hysteresis with Power-up Defaults
- Standard 2-Wire Serial Interface
- Thermal Event Alarm Output Functions as Interrupt or Comparator / Thermostat Output
- Up to 8 TCN75s may Share the Same Bus
- Shutdown Mode for Low Standby Power Consumption
- 5V Tolerant I/O at V_{DD} = 3V
- Low Power:
 - 250µA (Typ.) Operating
 - 1µA (Typ.) Shutdown Mode
- 8-Pin SOIC and MSOP Packaging

Applications

- Thermal Protection for High Performance CPUs
- Solid-State Thermometer
- Fire/Heat Alarms
- Thermal Management in Electronic Systems:
 - Computers
 - Telecom Racks
 - Power Supplies / UPS / Amplifiers
- Copiers / Office Electronics
- Consumer Electronics
- Process Control

Package Type



General Description

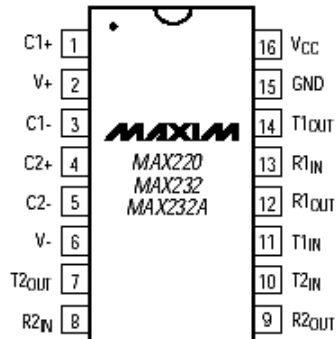
The TCN75 is a serially programmable temperature sensor that notifies the host controller when ambient temperature exceeds a user programmed set point. Hysteresis is also programmable. The INT/CMPTR output is programmable as either a simple comparator for thermostat operation or as a temperature event interrupt. Communication with the TCN75 is accomplished via a two-wire bus that is compatible with industry standard protocols. This permits reading the current temperature, programming the set point and hysteresis, and configuring the device.

The TCN75 powers up in Comparator mode with a default set point of 80°C with 5°C hysteresis. Defaults allow independent operation as a stand-alone thermostat. A shutdown command may be sent via the 2-wire bus to activate the low power Standby mode. Address selection inputs allow up to eight TCN75s to share the same 2-wire bus for multizone monitoring.

All registers can be read by the host and the INT/CMPTR output's polarity is user programmable. Both polled and interrupt driven systems are easily accommodated. Small physical size, low installed cost, and ease of use make the TCN75 an ideal choice for implementing sophisticated system management schemes.

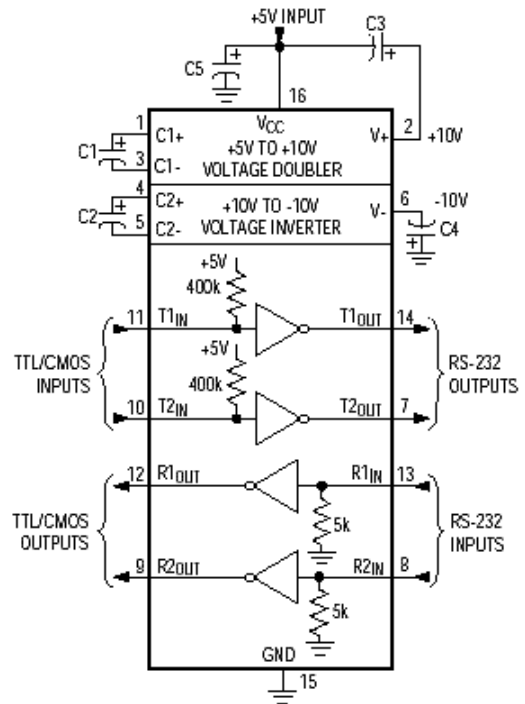
MAX-232

TOP VIEW



DIP/SO

CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1





TC74

Tiny Serial Digital Thermal Sensor

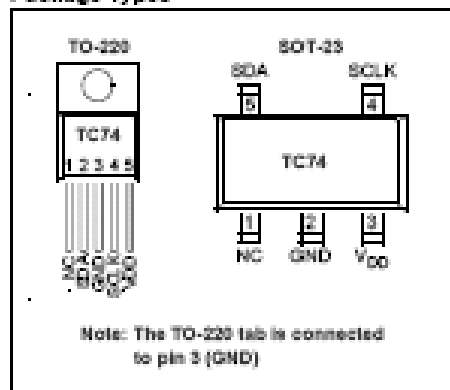
Features

- Digital Temperature Sensing in SOT-23-5 or TO-220 Packages
- Outputs Temperature as an 8-Bit Digital Word
- Simple SMBusTM Serial Port Interface
- Solid-State Temperature Sensing:
 - ±2°C (max.) Accuracy from +25°C to +85°C
 - ±3°C (max.) Accuracy from 0°C to +125°C
- Supply Voltage of 2.7V to 5.5V
- Low Power:
 - 200 µA (typ.) Operating Current
 - 5 µA (typ.) Standby Mode Current

Applications

- Thermal Protection for Hard Disk Drives and other PC Peripherals
- PC Card Devices for Notebook Computers
- Low Cost Thermal Control
- Power Supplies
- Thermistor Replacement

Package Types



General Description

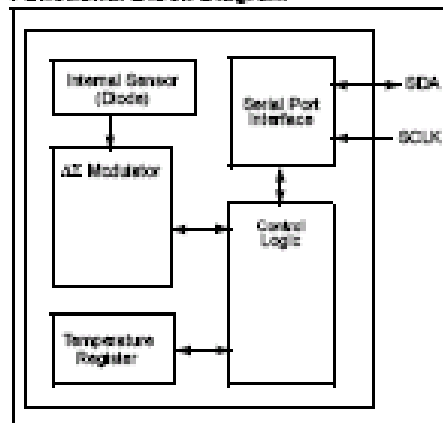
The TC74 is a serially accessible, digital temperature sensor particularly suited for low cost and small form-factor applications. Temperature data is converted from the onboard thermal sensing element and made available as an 8-bit digital word.

Communication with the TC74 is accomplished via a 2-wire SMBusTM compatible serial port. This bus also can be used to implement multi-drop/multi-zone monitoring. The SHDN bit in the CONFIG register can be used to activate the low power Standby mode.

Temperature resolution is 1°C. Conversion rate is a nominal 8 samples/sec. During normal operation, the quiescent current is 200 µA (typ). During standby operation, the quiescent current is 5 µA (typ).

Small size, low installed cost and ease of use make the TC74 an ideal choice for implementing thermal management in a variety of systems.

Functional Block Diagram



PC827/PC847

High Density Mounting Type Photocoupler

■ Lead forming type (I type) and tapping lead type (P type) are also available.
 ■ A TOY (VDE8836) approved type is also available as an option.

■ Features

1. Current transfer ratio (CTR): MIN. 30% at $I_{C(ON)}$, $V_{CE(ON)}$
2. High isolation voltage between input and output ($V_{CE(ON)}$: 5kV)
3. Compact dual-in-line package
 - PC827 2-channel type
 - PC847 4-channel type
4. Recognized by UL, file No. 24489D

■ Applications

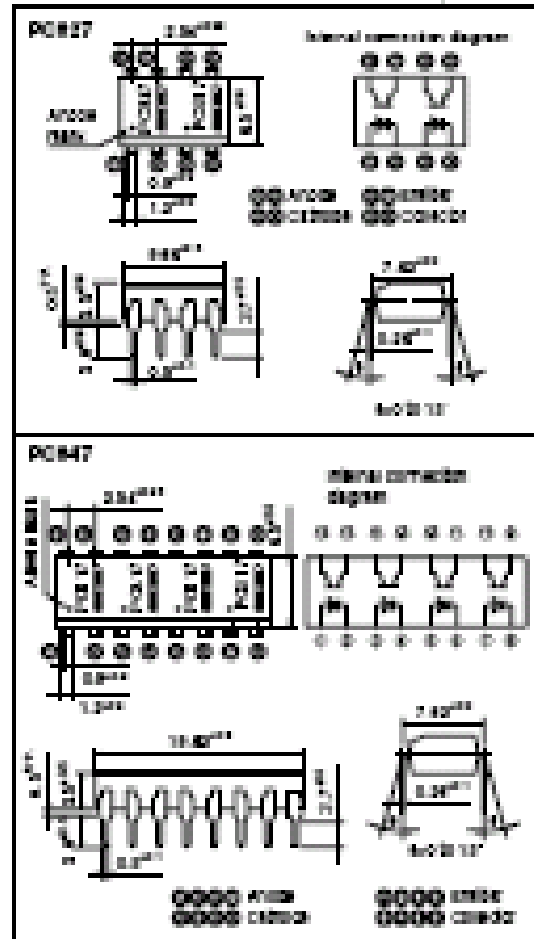
1. OA equipment
2. Copiers
3. Home appliances

■ Absolute Maximum Ratings (T_{case} 25°C)

	Parameter	Symbol	Rating	Unit
IN	Forward current	I_F	50	mA
	Peak forward current	I_{FM}	1	A
	Forward voltage	V_F	1	V
OUT	Forward signal	I_F	30	mA
	Collector-emitter voltage	V_{CE}	33	V
	Emitter-collector voltage	V_{EC}	6	V
	Collector current	I_C	50	mA
	Collector-emitter dissipation	P_C	150	mW
	Total power dissipation	P_{TOT}	200	mW
	Isolation voltage	$V_{CE(ON)}$	5	kV
	Operating temperature	T_{OP}	-30 to +100	°C
	Storage temperature	T_{ST}	-55 to +125	°C
	Soldering temperature	T_{SD}	260	°C

*1 Pulse width of 100µs, duty ratio 1:10
 *2 I_{FM} is at 100µs, 100 times release
 *3 See 102

■ Outline Dimensions (Unit: mm)



Mikroprozessor Source Code:

```
,*****
;
;   Filename:      final.asm
;   Date:         27/05/05
;   File Version: 0.1
;
;   Author:       Thomas Weber
;   Company:      HBLVA 17
;
;*****
;
;   Files required: 16f877.lkr
;
;*****
;
;   Notes: Das sollte jetzt endgültig die finale ASM-File sein, bei
;          der nun nach und nach alle Elemente eingebaut werden.
;
;*****

list      p=16f877          ; list directive to define processor
#include <p16f877.inc>      ; processor specific variable definitions

__CONFIG _CP_OFF & _WDT_ON & _BODEN_ON & _PWRTE_ON & _RC_OSC & _WRT_ENABLE_ON & _LVP_ON & _DEBUG_OFF & _CPD_OFF

; '__CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.

;***** VARIABLE DEFINITIONS (examples)

INT_VAR      UDATA_SHR
w_temp      RES    1          ; variable used for context saving
status_temp RES    1          ; variable used for context saving

TEMP_VAR     UDATA
hexstr       RES    2          ; 2 byte-register für hex-zahl
hexval       RES    1          ; wert v. hex-zahl
hexerr       RES    1          ; Ist ein Fehler bei hex-konvertierung aufgetreten?
hextmp       RES    1          ; temp. speicher für hex-konvertierung
robuff       RES    16         ; 16 byte reception buffer
ropos        RES    1          ; position in rcbuff
txbuff       RES    16         ; 16 byte transmission buffer
txpos        RES    1          ; position in txbuff
txleft       RES    1          ; bytes left for transmission
byte         RES    1          ; byte for reading and writing
```

```

resval_b      RES    1          ; sollwert v. Poti 1
restodo_b     RES    1          ; ausstehende steps für Poti 1
resbit_b      RES    1          ; div.flags v. Poti 1
resval_a      RES    1          ; sollwert v. Poti 2
restodo_a     RES    1          ; überbleibende Steps für Poti 2
resbit_a      RES    1          ; div. flags für Poti 2

;*****
RESET_VECTOR  CODE    0x000      ; processor reset vector
              movlw  high start  ; load upper byte of 'start' label
              movwf  PCLATH      ; initialize PCLATH
              goto   start       ; go to beginning of program

INT_VECTOR    CODE    0x004      ; interrupt vector location
              movwf  w_temp      ; save off current W register contents
              movf   STATUS,w     ; move status register into W register
              movwf  status_temp  ; save off contents of STATUS register

; isr code can go here or be located as a call subroutine elsewhere

              banksel PIR1
              btfsc  PIR1,TXIF    ; test for transmission interupt
              call   txhandle

              banksel PIR1
              btfsc  PIR1,RCIF    ; test for reception interupt
              call   rchandle

              banksel PIR1        ; test for timer interupt
              btfsc  PIR1,TMR1IF
              call   tmr1handle

              movf   status_temp,w ; retrieve copy of STATUS register
              movwf  STATUS        ; restore pre-isr STATUS register contents
              swapf  w_temp,f
              swapf  w_temp,w      ; restore pre-isr W register contents
              retfie               ; return from interrupt

MAIN          CODE
start
              ; initialisation stuff

              ;interupts

              banksel INTCON      ; INTCON: GIE, PIE
              clrf   INTCON
              bsf    INTCON,7
              bsf    INTCON,6

              ;serIO
              banksel SPBRG        ;SPBRG: .25 -> 9.615 kBaud
              movlw  .25
              movwf  SPBRG

              banksel TXSTA        ;TXSTA: BRGH -> high Baud rate

```

```

    clrf    TXSTA
    bsf    TXSTA,5
    bsf    TXSTA,2

    banksel RCSTA          ;RCSTA: SPEN
    clrf    RCSTA
    bsf    RCSTA,7

    banksel PIE1          ;PIE1: RCIE, TXIE, TMR1IE
    clrf    PIE1
    bsf    PIE1,RCIE
    bsf    PIE1,TXIE
    bsf    PIE1,TMR1IE

    banksel RCSTA          ;RCSTA: CREN -> enable the serialIO
    bsf    RCSTA,4

    banksel PORTB          ; clear PORT A & B
    clrf    PORTB
    banksel PORTA
    clrf    PORTA

    movlw   0x07           ; all RA-pins to digital
    banksel ADCON1
    movwf   ADCON1

    ;Port A & Port B

    banksel TRISB          ; set all pins to out
    clrf    TRISB
    banksel TRISA
    clrf    TRISA

    movlw   0x06           ; initialize PORT A & B
    banksel PORTB
    movwf   PORTB
    banksel PORTA
    movwf   PORTA

    ;timer

    movlw   0xA0           ; set 0xA000 as start-value for the timer
    banksel TMR1H
    movwf   TMR1H
    movlw   0x00
    banksel TMR1L
    movwf   TMR1L

    banksel T1CON          ; clear TMR1CS and set TMR1ON
    clrf    T1CON
    bcf    T1CON,TMR1CS
    bsf    T1CON,TMR1ON    ; timer 1 activated

    ;potentiometer

    banksel resval_b       ; initialize the values for the potentiometers
    clrf    resval_b
    banksel resbit_b

```



```

    clr    resbit_b
    bank   restodo_b
    clr    restodo_b

;common          ; set txleft, rcpos & txpos to 0
    bank   rcpos
    clr    rcpos
    bank   txpos
    clr    txpos
    bank   txleft
    clr    txleft

main
    ;sollte im idealfall leer bleiben
    goto   main          ; main loop

rchandle

    bank   RCREG          ; read byte
    movfw RCREG
    bank   byte
    movfw byte
    call   rcbufferwrite ; byte -> rcbuffer

    bank   RCSTA
    btfsc RCSTA,OERR      ; overrun error
    goto   rcfailure

    btfsc RCSTA,FERR      ; framing error
    goto   rcfailure

    call   rcinterpreter ; start the rc Command-Interpreter

    goto   rcend

rcfailure

    bcf    RCSTA,CREN      ; restart serial IO
    bsf    RCSTA,CREN

rcend

    return

txhandle

    bank   PIR1          ; clear TXIF
    bcf    PIR1,TXIF
    bank   txleft        ; check for remaining bytes to send
    movfw txleft
    sublw 0x0            ; 0 - txleft
    btfsc STATUS,Z       ; return if 0-txleft==0 -> txleft==0
    return

    call   txbufferread   ; read one byte from the buffer and send it

    return

tmr1handle          ; timer 1 interrupt

```

```

bankssel restodo_a      ; communication with potentiometer 1
movfw restodo_a
sublw 0
btfs STATUS,Z
goto tmr1h_p2_2

bankssel resbit_a
btfs resbit_a,0
goto tmr1h_p2_u

tmr1h_p2_d
bankssel PORTA
bcf PORTA,0
bankssel resbit_a
bcf resbit_a,0
bankssel restodo_a
decf restodo_a
goto tmr1h_p1

tmr1h_p2_u
bankssel PORTA
bsf PORTA,0
bankssel resbit_a
bsf resbit_a,0
goto tmr1h_p1

tmr1h_p2_2
bankssel resbit_a
btfs resbit_a,1      ; phase 1 or 2?
goto tmr1h_p2_3      ; phase 2 -> no action

                        ; phase 2:

bankssel resval_a
movfw resval_a      ; restodo=resval
bankssel restodo_a
movwf restodo_a

bankssel PORTA
bsf PORTA,1      ; set U/D to UP
bankssel resbit_a
bcf resbit_a,1      ; set to phase 2
goto tmr1h_p1

tmr1h_p2_3
                        ; nothing left to do:

bankssel PORTA
bsf PORTA,2      ; deactivate potentiometer
bankssel resbit_a
bcf resbit_a,2      ; end reservation

tmr1h_p1
bankssel restodo_b      ; communication with potentiometer 2
movfw restodo_b
sublw 0
btfs STATUS,Z
goto tmr1h_p1_2

```

```

        banksel resbit_b
        btfss resbit_b,0
        goto tmr1h_p1_u

tmr1h_p1_d
        banksel PORTB
        bcf PORTB,0
        banksel resbit_b
        bcf resbit_b,0
        banksel restodo_b
        decf restodo_b
        goto tmr1h_end

tmr1h_p1_u
        banksel PORTB
        bsf PORTB,0
        banksel resbit_b
        bsf resbit_b,0
        goto tmr1h_end

tmr1h_p1_2
        banksel resbit_b
        btfss resbit_b,1 ; phase 1 or 2?
        goto tmr1h_p1_3 ; phase 2 -> no action

                                ; phase 2:

        banksel resval_b
        movfw resval_b ; restodo = resval
        banksel restodo_b
        movwf restodo_b

        banksel PORTB
        bsf PORTB,1 ; set U/D to UP
        banksel resbit_b
        bcf resbit_b,1 ; set to phase 2
        goto tmr1h_end

tmr1h_p1_3 ; nothing left to do:
        banksel PORTB
        bsf PORTB,2 ; deactivate potentiometer
        banksel resbit_b
        bcf resbit_b,2 ; end reservation

tmr1h_end
        movlw 0xFA
        banksel TMR1H
        movwf TMR1H

        banksel PIR1
        bcf PIR1,TMR1IF ; clear TMR1IF
        nop ; erst mal irgendwas, damit man den Timer auf seine Funktion prüfen kann.

        return ; end timer - interrupt

;## RC-Buffer Functions

rcbufferwrite ; write the content of byte to rcbuff

```

```

banksel rcpos
movfw rcpos          ; rcbuff[rcpos] = byte
addlw rcbuff
movwf FSR
banksel byte
movfw byte
movwf INDF

incf rcpos          ; rcpos++

movlw 0xF          ; rcpos & 0b00001111
andwf rcpos,F

return

rcbufferread          ; read the content of rcbuff at rcpos to byte
banksel rcpos
movlw 0x1
subwf rcpos,W       ; byte = rcbuff[(rcpos-1) & 0xf]
andlw 0xf
addlw rcbuff
movwf FSR
movfw INDF
banksel byte
movwf byte
return

rcbufferdec          ; move rcpos back
banksel rcpos
decf rcpos          ; rcpos--

movlw 0xF          ; rcpos & 0b00001111
andwf rcpos,F
return

rcbufferinc          ; move rcpos forward
banksel rcpos
incf rcpos          ; rcpos++

movlw 0xF          ; rcpos & 0b00001111
andwf rcpos,F
return

;### TX-Buffer Functions

txbufferwrite        ; write byte into txbuff and send it, if possible
banksel txpos
movfw txpos          ; txbuff[txpos] = byte
banksel txbuff
addlw txbuff
movwf FSR
banksel byte
movfw byte
movwf INDF

banksel txpos
incf txpos          ; txpos++

```

```

movlw 0xF          ; txpos & 0b00001111
andwf txpos,F

banksel txleft
incf txleft       ; txleft++

movfw txleft
sublw 0xe        ; 0xf - txleft
btfss STATUS,C   ; skip if positive
goto txbor

btfss TXSTA,1    ; check if the PIC is currently transmitting
return          ; end function if shift register is full

call txbufferread ; if shift register is empty, start transmission

return

txbor
; almost Buffer overrun
; force buffer cleaning

banksel INTCON   ; disable interrupts
bcf INTCON,GIE

txbor01

banksel TXSTA
btfss TXSTA,1    ; wait
goto $-1

call txbufferread ; write 1 byte

banksel txleft
movfw txleft     ; is txleft 0?
sublw 0
btfss STATUS,Z
goto txbor01

banksel INTCON   ; reenale interrupts
bsf INTCON,GIE

return

txbufferread
; read one byte from txbuff and send it

banksel txleft
movfw txleft     ; byte = txbuff[(txpos-txleft) & 0xF]
subwf txpos,W    ; txpos - txleft
andlw 0xF        ; W & 0xF
addlw txbuff     ; txbuff + W
movwf FSR
movfw INDF
banksel byte
movwf byte

banksel TXREG
movwf TXREG

banksel txleft

```

```

    decf    txleft

    return

rcinterpreter                                ; check for commands over COM

    call   rcbufferread                      ; read the most recent byte from rcbuff

    ; test for u and set potentiometer 1 if success
    banksel byte
    movfw  byte
    sublw  'u'
    btfss  STATUS,Z
    goto   rcinterpreter_1
    call   fillhexstr                        ; copy the 2 bytes before the command-byte to hexstr
    call   poti_1                            ; start potentiometer 1

rcinterpreter_1

    ; test for v and set potetniometer 2 if success
    banksel byte
    movfw  byte
    sublw  'v'
    btfss  STATUS,Z
    goto   rcinterpreter_2
    call   fillhexstr                        ; copy the 2 bytes before the command-byte to hexstr
    call   poti_2                            ; start potentiometer 2

rcinterpreter_2

    ; test for w and clear port B3 if success
    banksel byte
    movfw  byte
    sublw  'w'
    btfss  STATUS,Z
    goto   $+4
    banksel PORTB
    bcf    PORTB,3

    ; test for x and clear port B4 if success
    banksel byte
    movfw  byte
    sublw  'x'
    btfss  STATUS,Z
    goto   $+4
    banksel PORTB
    bcf    PORTB,4

    ; test for y and clear port A3 if success
    banksel byte
    movfw  byte
    sublw  'y'
    btfss  STATUS,Z
    goto   $+4
    banksel PORTA
    bcf    PORTA,3

```

```
; test for z and clear port A4 if success
```

```
banksel byte  
movfw byte  
sublw 'z'  
btfs STATUS,Z  
goto $+4  
banksel PORTA  
bcf PORTA,4
```

```
; test for W and set port B3 if success
```

```
banksel byte  
movfw byte  
sublw 'W'  
btfs STATUS,Z  
goto $+4  
banksel PORTB  
bsf PORTB,3
```

```
; test for X and set port B4 if success
```

```
banksel byte  
movfw byte  
sublw 'X'  
btfs STATUS,Z  
goto $+4  
banksel PORTB  
bsf PORTB,4
```

```
; test for Y and set port A3 if success
```

```
banksel byte  
movfw byte  
sublw 'Y'  
btfs STATUS,Z  
goto $+4  
banksel PORTA  
bsf PORTA,3
```

```
; test for Z and set port A4 if success
```

```
banksel byte  
movfw byte  
sublw 'Z'  
btfs STATUS,Z  
goto $+4  
banksel PORTA  
bsf PORTA,4
```

```
return
```

poti_1

```
call str2val ; konvertiere den hex-String zu einem hex-Wert
```

```
btfs hexerr,0 ; hex-Wert fehlerhaft
```

```
goto poti_1_end
```

```

btfscl resbit_b,2      ; Potentiometer gerade im Betrieb -> Abbruch
goto pot1_1_end

movfw hexval          ; hex-Wert wird in resval_b beschrieben
banksel resval_b
movwf resval_b

bcf PORTB,2
bcf PORTB,1

bsf resbit_b,0      ; start INC with high
bsf resbit_b,1      ; start with setting the resistance to 0
bsf resbit_b,2      ; reserve potentiometer

movlw 0x64
movwf restodo_b

pot1_1_end

return

pot1_2

call str2val          ; konvertiere den hex-String zu einem hex-Wert

btfscl hexerr,0      ; hex-Wert fehlerhaft
goto pot1_2_end

btfscl resbit_a,2    ; Potentiometer gerade im Betrieb -> Abbruch
goto pot1_2_end

movfw hexval          ; hex-Wert wird in resval_a beschrieben
banksel resval_a
movwf resval_a

bcf PORTA,2
bcf PORTA,1

bsf resbit_a,0      ; start INC with high
bsf resbit_a,1      ; start with setting the resistance to 0
bsf resbit_a,2      ; reserve potentiometer

movlw 0x64
movwf restodo_a

pot1_2_end

return

fillhexstr

call rcbufferdec     ; move rcpes two positions back
call rcbufferdec

call rcbufferread    ; copy rcbuff at rcpes to byte

movlw hexstr+0      ; copy byte to hexstr+0
movwf FSR

```



```

banksel byte
movfw byte
movwf INDF

call rcbufferinc ; move rcpes one position forward
call rcbufferread ; copy rcbuff at rcpes to byte

movlw hexstr+1 ; copy byte to hexstr+1
movwf FSR
banksel byte
movfw byte
movwf INDF

call rcbufferinc ; move rcpes to the initial position
call rcbufferread ; read in the command-byte again
return

str2val
; konvertiert einen hex-String in einen hex-Wert

movlw hexstr ; konvertiert das erste byte und testet anschliessend auf fehler
movwf FSR
call chkchar
btfs hexerr,0
return

speichert
swapf hextmp,W ; das erste eingelesene byte wird mit swapf nach vorne geschoben und in hexval ge-
movwf hexval

movlw hexstr+1 ; konvertiert das zweite byte und testet anschliessend auf fehler
movwf FSR
call chkchar
btfs hexerr,0
return

movfw hextmp ; das zweite eingelesene byte wird mit dem ersten verodert.
iorwf hexval,F
return

chkchar

nop

movfw INDF ; smaller than '0'-2 then error
sublw '0'-2 ; ('0'-2) - x
btfs STATUS,C ; skip if 0 -> negative
goto cberror

movfw INDF ; smaller than '9' then number
sublw '9' ; '9' - x
btfs STATUS,C ; skip if 0 -> negative
goto cbnum

movfw INDF ; smaller then 'A'-2 then error
sublw 'A'-2 ; ('A'-2) - x
btfs STATUS,C ; skip if 0 -> negative
goto cberror

movfw INDF ; smaller than 'F' then character

```

```

    sublw  'F'          ; 'F' - x
    btfsc STATUS,C     ; skip if s -> negative
    goto  cbchar

cberror

    movlw  0xff
    movwf  hexerr     ; FEHLER
    return

cbchar

    movlw  'A'
    subwf  INDF,W     ; f-w+10 -> char-'A'+10
    addlw  0xA
    movwf  hextmp
    movlw  0x00
    movwf  hexerr     ; kein fehler
    return

cbnum

    movlw  '0'
    subwf  INDF,W     ; f-w -> char-'0'
    movwf  hextmp
    movlw  0x00
    movwf  hexerr     ; kein fehler
    return

val2str

    ; konvertiert einen Hex-Wert in einen Hex-String
banksel hexval
    swapf  hexval,w
    andlw  0x0f

    movwf  hextmp     ; Erster Teil
    btfss STATUS,Z
    goto  $+4
    movlw  '0'
    movwf  hexstr
    goto  vs_next
    decfsz hextmp
    goto  $+4
    movlw  '1'
    movwf  hexstr
    goto  vs_next
    decfsz hextmp
    goto  $+4
    movlw  '2'
    movwf  hexstr
    goto  vs_next
    decfsz hextmp
    goto  $+4
    movlw  '3'
    movwf  hexstr
    goto  vs_next
    decfsz hextmp
    goto  $+4
    movlw  '4'
    movwf  hexstr
    goto  vs_next
    decfsz hextmp

```

```

goto    $+4
movlw   '5'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   '6'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   '7'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   '8'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   '9'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'A'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'B'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'C'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'D'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'E'
movwf   hexstr
goto    vs_next
decfsz  hextmp
goto    $+4
movlw   'F'
movwf   hexstr
goto    vs_next

```

vs_next

```

banksel hexval
movf   hexval,w
andlw  0x0f

movwf  hextmp           ; Zweiter Teil
btfss  STATUS,Z
goto   $+4
movlw  '0'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '1'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '2'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '3'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '4'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '5'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '6'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '7'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '8'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  '9'
movwf  hexstr+1
goto   vs_end
decfsz hextmp
goto   $+4
movlw  'A'

```

```

movwf hexstr+1
goto vs_end
decfsz hextmp
goto $+4
movlw 'B'
movwf hexstr+1
goto vs_end
decfsz hextmp
goto $+4
movlw 'C'
movwf hexstr+1
goto vs_end
decfsz hextmp
goto $+4
movlw 'D'
movwf hexstr+1
goto vs_end
decfsz hextmp
goto $+4
movlw 'E'
movwf hexstr+1
goto vs_end
decfsz hextmp
goto $+4
movlw 'F'
movwf hexstr+1
goto vs_end

vs_end

return

END ; directive 'end of program'

```