Teutloff-Schule
Braunschweig
Fachschule Technik

Datenverarbeitungstechnik: ETZ02/2

Projektarbeit: Programmierbare Tastatur und Joystick
für Personal-Computer (ProggyKeyJoy)

Andreas Pernau
Ratiborstraße 31
D-38124 Braunschweig

Andreas.Pernau@arcor.de

Projektbetreuungsteam: Herr Heuer und Herr Gröper

Hiermit bestätige ich, dass ich diese Projektarbeit eigenständig, und nur unter zu Hilfenahme der angegebenen Quellen, erstellt habe.

Braunschweig, 03.09.2006

_____

Andreas Pernau

Inhaltsverzeichnis

1. Einleitung

Im Rahmen meiner Weiterbildung zum staatlich geprüften Techniker Fachbereich Elektrotechnik mit Schwerpunkt Datenverarbeitung musste ich zum Ende hin eine Projektarbeit machen.

Dabei setzte ich mir die Vorgaben, möglichst viele Aspekte der Elektronik zu berücksichtigen. Am geeignetsten schien mir dabei die Entwicklung eines Gerätes, möglichst mit Anschluss an einen Computer. Es sollte dabei sowohl einen Analog-, als auch einen Digitalteil enthalten. Der Programmierung eines Microcontrollers war dabei der Schwerpunkt der Projektarbeit angedacht. Auch ein Platinen-Layout sollte bei solch einer Arbeit natürlich nicht fehlen.

Folgende vier Probleme, vor denen ich im Laufe der Jahre immer wieder stand, sollten durch das Gerät zu lösen sein.

- Viele Anwenderprogramme und Spiele auf einem Personal-Computer lassen die Bedienung mit einer großen Zahl von Tastenkombinationen (auch Short-Cuts genannt) zu. Leider sind diese Tastenkombinationen, gerade für diejenigen, die solch ein Programm seltener benutzen, nur schwer zu merken.

- Bei der Arbeit an einem Computer kommt es immer wieder vor, dass man die gleichen Texte (z.B. Adressdaten) eingeben muss. Dies ist auf Dauer sehr zeitaufwendig und fehleranfällig.

- Ein weiteres Problem ist, dass sich heutzutage, gerade durch die Nutzung des Internets, sehr viele Passwörter und Benutzernamen ansammeln. Da diese meist unterschiedlich sind (und zur Sicherheit auch sein sollten) taucht wieder das Problem der Merkbarkeit auf. Es gibt zwar Programme, die die Passwörter auf dem Computer verwalten, dabei besteht jedoch immer ein Restrisiko durch ein Schädlingsprogramm an die Passwortdatei zu kommen, diese zu entschlüsseln und damit an alle Passwörter zu kommen. Die Notierung der Passwörter und Benutzernamen auf Papier stellt dabei nur eine unbefriedigende Lösung dar.

- Es gibt viele Spiele, bei denen sich leider viele Funktionen nur mit der Tastatur steuern lassen, obwohl ein Joystick vorhanden ist. Die Reaktionszeit, gerade von nicht so geübten Spielern, ist beim Spielen über die Tastatur deutlich geringer als durch die Benutzung eines Joysticks oder Joypads.

Das Gerät muss also in der Lage sein, Tastenfolgen und –kombinationen auf eine Taste, bzw. eine Joystick-Funktion zu programmieren um diese dann, bei Bedarf wieder an den Computer zu senden. Hieraus ergab sich dann der Name ProggyKeyJoy, der soviel bedeutet wie Programmierbare(r) Tastatur/Joystick. Der Bereich für die Passwörter muss dabei natürlich durch eine PIN, bzw. ein Passwort gesichert sein.

2. Beschreibung von ProggyKeyJoy

Das Gerät besitzt eine Tastatur mit 16 Tasten, von denen 10 Tasten programmierbar sind. Die restlichen 6 Tasten sind Steuertasten mit den Funktionen Keyset, Joyset, Passwordset, Setup, OK und Cancel.

Oberhalb der Tastatur befindet sich eine 2-zeilige LCD-Anzeige mit jeweils 16 Zeichen, über die dem Benutzer Anweisungen und Statusmeldungen gegeben werden.

Auf der Rückseite befinden sich Anschlüsse für Computer (PS/2-Stecker an 2,4m Leitung), Tastatur (PS/2-Buchse) und Joystick (15pol. SUB-D Buchse). Die Spannungsversorgung wird über den Computer hergestellt. Die Stromaufnahme liegt bei ca. 90mA.

Es lassen sich im Normal- und Passwort-Modus jeweils zehn Sätze mit jeweils zehn Tasten speichern, von denen jeder mit 200 Bytes belegt werden kann, was in etwa 40 bis 66 Zeichen entspricht, je nachdem, ob die Taste einen herkömmlichen Scan-Code mit 1Byte (+ 2Byte Break-Code = 3Byte) oder einen erweiterten Scan-Code mit 2Byte (+ 3Byte Break-Code = 5Byte) hat (200Byte : 3Byte pro Zeichen = 66,7 Zeichen bzw. 200Byte : 5Byte pro Zeichen = 40 Zeichen). Beim Joystick sind es 12 Tasten, bzw. Funktionen (4 Knöpfe + 4 Achsen mit jeweils 2 Richtungen = 12 Funktionen).

3. Die Tastatur-Schnittstelle

3.1 Geschichte

Es gibt viele verschiedene Arten von Tastaturen. Die drei gebräuchlichsten Arten sind heute:

- USB-Tastatur : Dies ist die aktuellste Tastatur, die von nahezu allen neuen Computern (Macintosh und IBM-kompatibel) unterstützt werden. Diese Schnittstelle ist relativ kompliziert und wird hier nicht weiter behandelt.

- IBM-kompatible Tastatur : Sie ist auch bekannt als „AT-Tastatur“ oder „PS/2-Tastatur“. Alle modernen PCs unterstützen diese Tastatur. Sie haben die einfachste Schnittstelle und werden im Folgenden genauer beschrieben.

- ADB-Tastatur : Sie ist für den Apple Desktop Bus älterer Macintosh-Systeme. Sie wird hier nicht weiter behandelt.

Die IBM-kompatible Tastatur hat seit Einführung des ersten PCs, vor über 20 Jahren, nur zwei grundlegende Änderungen erfahren.

Der original IBM PC und später auch der IBM XT benutzten die sogenannte "XT-Tastatur". Diese sind aber heute nicht mehr gebräuchlich und unterscheiden sich gravierend von modernen Tastaturen. Sie werden deshalb hier nicht weiter behandelt.

Danach kam das IBM AT System. Diese Tastaturen (auch MF II-Tastatur genannt) können auch heute noch, mittels eines einfachen Adapters, an die meisten IBM-kompatiblen Computer angeschlossen werden.

Dann folgte das IBM PS/2 System. Diese Tastaturen sind obigen AT-Tastaturen sehr ähnlich. Sie besitzen jedoch einen kleineren Stecker und unterstützen eine Reihe von Zusatzfunktionen. Auch sie können mittels eines einfachen Adapters an ältere IBM ATs angeschlossen werden, da sie elektrisch miteinander kompatibel sind.

Folgende Tabelle gibt einen kurzen Überblick über die wichtigsten Merkmale der drei Tastatur-Systeme.

	IBM PC/XT	*IBM AT*	*IBM PS/2*
Einführung	1981	1984	1987
Tasten	83	84 – 101	84 – 101
Stecker	5-pol. DIN	5-pol. DIN	6-pol. Mini-DIN
Scan-Code Set	1	2	2, 3
Serielles Protokoll	uni-direktional	bi-direktional	bi-direktional
Host-Befehle	0	8	17

Tabelle 3.1.1

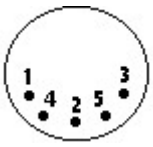
IBM PC/XT-Tastaturen sind trotz des selben Steckverbinders nicht an AT-Systeme oder neuer anschließbar, da sie ein anderes Protokoll besitzen. In der Übergangszeit von IBM PC/XT nach IBM AT gab es sowohl Computer, die mit beiden Tastatur-Systemen arbeiteten, als auch Tastaturen, die man an beide Computer-Systeme anschließen konnte. Die Angabe der Tasten bezieht sich auf den jeweiligen IBM-Standard bei Markteinführung. Tastaturen können jedoch auch eine abweichende Anzahl von Tasten besitzen. Die Scan-Code Sets werden weiter unten genauer beschrieben (siehe Kapitel 3.3.1). Mit Host-Befehlen sind die Befehle gemeint, die der Computer an die Tastatur senden kann (siehe Kapitel 3.3.1). Das Ein- und Ausschalten der Caps Lock-, Num Lock und Scroll Lock-LEDs geschieht zum Beispiel durch den Computer. Die Angabe der Host-Befehle in Tabelle 3.1.1 muss dabei als IBM-Standard gesehen werden, an den sich viele, wenn nicht sogar die meisten, Tastatur-Hersteller nicht gebunden fühlen. Es gibt also sowohl AT-Tastaturen, die mehr als 8 Host-Befehle unterstützen, als auch PS/2-Tastaturen, die wiederum weniger als 17 Host-Befehle unterstützen. Es wird nur die Unterstützung von Scan-Code Set 2 garantiert, ob eine moderne Tastatur auch Scan-Code Set 3 unterstützt, ist mehr oder weniger Glückssache.

3.2 Aufbau

Das Anschlusskabel der Tastaturen ist vier- bis sechspolig abgeschirmt vom Typ 26 AWG (ca. 0,13mm^2) und gewöhnlich 2m lang. Bei den meisten Tastaturen ist das Kabel fest verbunden, es gibt jedoch auch einige mit einem abnehmbaren Kabel. Diese Kabel haben an der einen Seite den gewohnten 6-poligen Mini-DIN- oder 5-poligen DIN-Stecker (zum Computer) und an der anderen Seite einen sogenannten SDL-Steckverbinder (Sheilded Data Link). SDL wurde von der Firma AMP eingeführt. Der Stecker besitzt eine gewisse Ähnlichkeit mit einem Western-Telefonstecker, da er Federzungen statt Stiften besitzt und mit einer Federzunge einrastet.

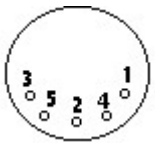
Die folgenden Abbildungen zeigen die Pinbelegungen für jeden Stecker:

Male



(Stecker)

Female

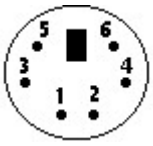


(Buchse)

5-pin DIN (AT/XT):
1 - Clock
2 - Data
3 - Nicht belegt
4 - GND
5 - Vcc (+5V)

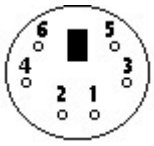
Abbildung 3.2.1

Male



(Stecker)

Female



(Buchse)

6-pin Mini-DIN (PS/2):
1 - Data
2 - Nicht belegt
3 - GND
4 - Vcc (+5V)
5 - Clock
6 - Nicht belegt

Abbildung 3.2.2

6-pin SDL:
A - Nicht belegt
B - Data
C - GND
D - Clock
E - Vcc (+5V)
F - Nicht belegt

Abbildung 3.2.3

Über Vcc und GND wird die Tastatur vom Computer mit Strom versorgt. Vcc ist dabei im Bereich von 4,5V bis 5,5V angesiedelt. Die Tastatur sollte nicht mehr als 275mA verbrauchen. Besondere Vorsicht ist gegenüber Spannungsspitzen angebracht. Diese entstehen, wenn die Tastatur ein- oder ausgestöpselt wird, während der Computer eingeschaltet ist. Ältere Mainboards hatten SMD-Sicherungen als Schutz für die Tastaturanschlüsse. Wenn diese durchbrannten, war das Board in der Regel unbrauchbar und durch einen durchschnittlichen Techniker nicht zu reparieren. Die meisten neueren Mainboards haben deshalb selbstheilende (Polyswitch-) Sicherungen. Es ist jedoch nicht Standard, weshalb es nicht anzuraten ist die Tastatur unter Spannung zu stecken.

Die Data- und Clock-Leitungen sind vom Typ Open Collector mit Pull-Up-Widerständen. Eine Open-Collector-Schnittstelle hat zwei mögliche Zustände: Low oder hochohmig. Im Zustand Low schaltet ein Transistor gegen Masse (GND) durch. Im hochohmigen Zustand wird die Leitung weder High noch Low getrieben. Zusätzlich ist ein Pull-Up-Widerstand zwischen der Leitung und Vcc geschaltet. Dieser sorgt dafür, dass die Leitung High-Potential hat, wenn sie von keinem der Busteilnehmer auf Low gezogen wird. Der Wert dieses Widerstandes ist unkritisch und sollte zwischen 1kΩ und 10kΩ liegen. Höhere Werte reduzieren den Stromverbrauch, kleinere Werte reduzieren die Anstiegszeiten.

Ein universelles Open-Collector-Interface ist auf der folgenden Abbildung gezeigt:

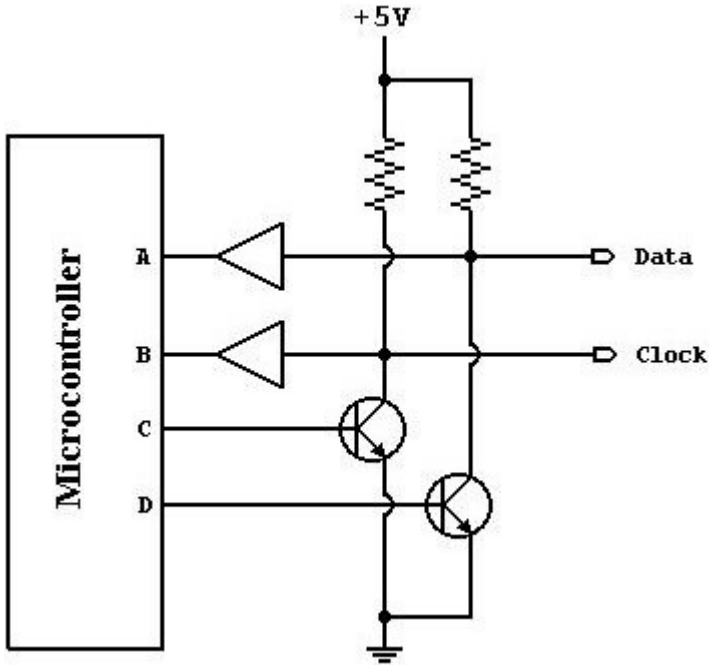


Abbildung 3.2.4

Data und Clock werden an Pin A bzw. B des Microcontrollers eingelesen. Beide Leitungen haben im Ruhezustand +5V und können auf Masse gezogen werden, indem logisch 1 an C oder D angelegt wird. Folglich hat Data den invertierten Zustand von D, und Clock den invertierten Zustand von C.

Eine Tastatur ist aus einer Matrix von Tasten aufgebaut, die durch einen eingebauten Microcontroller, den sogenannten Keyboard-Encoder überwacht werden. Als Microcontroller wurde dabei ursprünglich der Intel 8048 verwendet, heute werden natürlich auch Andere eingesetzt. Sie registrieren, welche Taste(n) gedrückt oder losgelassen wurde(n) und senden die entsprechenden Daten zum Host (Computer). Sie sorgen außerdem noch für die Tastenentprellung und speichern gegebenenfalls die Daten in ihrem 16-Byte Puffer. Die Computer-Hauptplatine hat einen Tastatur-Controller, der alle Daten von der Tastatur dekodiert und an die Software weitergibt. Ursprünglich wurde hierfür ein Microcontroller vom Typ Intel 8042 eingesetzt, inzwischen wird diese Aufgabe von den Chipsätzen der Hauptplatinen erledigt. Zum Datenaustausch zwischen Tastatur und Host wird ein IBM-Protokoll verwendet, welches auch bei PS/2-Mäusen Verwendung findet.

3.3 Protokoll

3.3.1 Software

Der Keyboard-Encoder in der Tastatur verbringt die meiste Zeit damit die einzelnen Tasten zu überwachen. Wird eine Taste gedrückt, festgehalten oder losgelassen, so sendet er an den Computer ein Datenpaket, welches auch als Scan-Code bezeichnet wird. Die Scan-Codes unterscheiden sich in Make-Codes und Break-Codes. Ein Make-Code wird gesendet, wenn eine Taste gedrückt oder festgehalten wird. Ein Break-Code wird gesendet, wenn eine Taste losgelassen wird. Jede Taste besitzt einen eigenen Make-Code und Break-Code, so dass der Computer exakt bestimmen kann, was mit jeder einzelnen Taste passiert ist. Es gibt drei Standard-Scan-Code-Sets, die als Set 1, 2 und 3 bezeichnet werden. Alle modernen Tastaturen verwenden standardmäßig das Scan-Code-Set 2. Die Make-Codes und Break-Codes jeder Taste müssen in einer Tabelle nachgeschlagen werden (siehe Anhang 10.7). Der Tastatur-Controller im Computer konvertiert, aus Gründen der Kompatibilität zu IBM PC/XT, alle ankommenden Daten zu Scan-Code-Set 1.

Bei jeder Betätigung einer Taste wird der Make-Code dieser Taste zum Computer gesendet, wobei zu beachten ist, dass der Make-Code nur eine Taste auf der Tastatur repräsentiert, und nicht das Zeichen, das auf der Taste aufgedruckt ist. Es gibt also keinen Zusammenhang zwischen einem Scan-Code und einem ASCII-Code. Der Computers hat letztendlich die Aufgabe die Scan-Codes in Buchstaben oder Befehle umzusetzen. Die meisten Make-Codes bestehen aus einem Byte, es gibt jedoch auch erweiterte Make-Codes, die aus zwei oder vier Bytes bestehen. Diese Make-Codes erkennt man daran, dass ihr erstes Byte 0xE0 ist. Beim Loslassen einer Taste wird der Break-Code gesendet. Dieser steht im direktem Zusammenhang zu dem Make-Code. Die meisten Break-Codes im Set 2 sind zwei Byte lang, wobei das erste Byte 0xF0 und das zweite Byte der Make-Code der Taste ist. Die Break-Codes der erweiterten Tasten sind gewöhnlich drei Bytes lang, wobei die ersten beiden Bytes 0xE0 und 0xF0 sind. Das dritte Byte ist das letzte Byte des Make-Codes. Die Pause/Unterbr-Taste bildet dabei eine Ausnahme, da sie in den Scan-Code Sets 1 und 2 nur einen Make-Code, nicht jedoch einen Break-Code besitzt. Es ist also nicht möglich zu bestimmen, wann diese Taste losgelassen wurde. Als Beispiel soll einmal der Buchstabe „A“ an den Computer gesendet werden.
Folgende Vorgänge sind dazu abzuarbeiten:

- Drücken der Umschalt-Taste
- Drücken der A-Taste
- Loslassen der A-Taste
- Loslassen der Umschalt-Taste

Die Scan-Codes dieser Vorgänge sind:

- Make-Code für die Umschalt-Taste (0x12)
- Make-Code für die A-Taste (0x1C)
- Break-Code für die A-Taste (0xF0, 0x1C)
- Break-Code für die Umschalt-Taste (0xF0, 0x12)

Folgende Daten werden an den Computer gesendet:

- 0x12
- 0x1C
- 0xF0, 0x1C
- 0xF0, 0x12

Wird eine Taste festgehalten, so setzt die Wiederholfunktion (Typematic) ein, wobei fortlaufend der Make-Code der Taste gesendet wird, solange bis die Taste losgelassen oder eine andere Taste gedrückt wird. Diese Wiederholfunktion setzt sich aus folgenden zwei Parametern zusammen: Die Verzögerung (typematic delay), die angibt, nach welcher Zeit eine Taste als festgehalten gilt, und die Wiederholrate (typematic rate), die angibt, wie viele Zeichen pro Sekunde gesendet werden. Die Verzögerung kann zwischen 0,25s und 1,00s liegen, und die Wiederholrate zwischen 2,0cps (Zeichen pro Sekunde) und 30,0cps. Diese beiden Parameter können mit dem Befehl 0xF3 ("Set Typematic Rate/Delay") eingestellt werden (siehe Seite 11). Wiederholte Daten werden nicht in der Tastatur zwischengespeichert. Wenn mehrere Tasten festgehalten werden, so setzt die Wiederholfunktion nur für die Taste ein, die als letztes heruntergedrückt wurde. Die Wiederholfunktion endet, wenn diese Taste losgelassen wird, selbst wenn noch weitere Tasten gedrückt sein sollten.

Nach einem Einschalt- oder Software-Reset (siehe Reset-Befehl) führt die Tastatur einen Selbsttest, genannt BAT (Basic Assurance Test) durch und lädt folgende Grundeinstellungen:

- Verzögerung 500ms,
- Wiederholrate 10,9cps,
- Scan-Code Set 2
- Alle Tasten Typematic/Make/Break

Zu Beginn des BAT schaltet die Tastatur ihre drei LEDs ein und am Ende wieder aus. Dann wird ein BAT-Ende-Code gesendet, der entweder 0xAA (BAT erfolgreich) oder 0xFC (Fehler) ist. Der BAT-Ende-Code muss 500ms bis 750ms nach dem Einschalten der Versorgungsspannung gesendet werden.

Die Tastatur löscht ihren Ausgabepuffer bei jedem empfangenen Befehl. Wenn die Tastatur einen ungültigen Befehl oder Parameter empfängt, muss sie mit Resend (0xFE) antworten. Die Tastatur darf keine Scan-Codes senden, während sie einen Befehl verarbeitet. Wenn die Tastatur auf ein Parameter-Byte wartet und stattdessen einen Befehl empfängt, sollte sie den letzten Befehl verwerfen und den neuen auswerten.
Folgende Befehle kann der Host (Computer) zur Tastatur senden:

- 0xFF (Reset) : Die Tastatur antwortet mit "Acknowledge" (0xFA) und führt dann den Reset aus.

- 0xFE (Resend) : Die Tastatur wiederholt das letzte gesendete Byte. Eine Ausnahme besteht, wenn das letzte gesendete Byte "Resend" (0xFE) war. Dann wiederholt die Tastatur das letzte Byte, das nicht 0xFE war. Mit diesem Befehl zeigt der Host an, dass ein Empfangsfehler aufgetreten ist.

- 0xF7 (Set All Keys Typematic) : Die Tastatur antwortet mit "Acknowledge" (0xFA). Ähnlich wie 0xFB, gilt jedoch für alle Tasten.

- 0xF6 (Set Default) : Lädt Grundeinstellungen: Wiederholfunktion 10.9cps / 500ms; Aktiviert MakeCode, BreakCode und Wiederholfunktion für alle Tasten; stellt Scancode-Set 2 ein.

- 0xF5 (Disable) : Stoppt die Tastenabfrage, lädt die Grundeinstellung (siehe "Set Default"-Befehl) und wartet auf weitere Befehle.

- 0xF4 (Enable) : Aktiviert die Tastenabfrage, nachdem sie mit dem vorherigen Befehl deaktiviert wurde.

- 0xF3 (Set Typematic Rate/Delay) : Die Tastatur antwortet mit "Acknowledge" (0xFA) und liest dann ein Parameterbyte vom Host, mit dem die Wiederholrate und Verzögerung eingestellt werden. Die Tastatur antwortet dann wieder mit "Acknowledge" (0xFA). Das Parameterbyte ist wie folgt definiert:

  Wiederholrate:

Bits 0-4 [hex]	*Wiederholrate [cps]*		*Bits 0-4 [hex]*	*Widerholrate [cps]*
00	30,0		10	7,5
01	26,7		11	6,7
02	24,4		12	6,0
03	21,8		13	5,5
04	20,7		14	5,0
05	18,5		15	4,6
06	17,1		16	4,3
07	16,0		17	4,0
08	15,0		18	3,7
09	13,3		19	3,3
0A	12,0		1A	3,0
0B	10,9		1B	2,7
0C	10,0		1C	2,5
0D	9,2		1D	2,3
0E	8,6		1E	2,1
0F	8,0		1F	2,0

Tabelle 3.3.1.1

Verzögerung:

Bit 7	*Bit 6*	*Bit 5*	*Verzögerung [s]*
0	0	0	0,25
0	0	1	0,50
0	1	0	0,75
0	0	1	1,00

Tabelle 3.3.1.2

Soll zum Beispiel eine Wiederholrate von 10,9cps und eine Verzögerung von 0,5s eingestellt werden, so geschieht dies nach folgenden Vorgängen:

- Host sendet Befehl zum Einstellen der Wiederholrate und der Verzögerung (0xF3).
- Tastatur sendet Acknowledge (0xFA).
- Host sendet Parameter-Byte 0010 1011b = (0x2B).
- Tastatur sendet Acknowledge (0xFA).

- 0xF2 (Read ID) : Die Tastatur antwortet mit einem Zwei-Byte ID-Code 0xAB, 0x83.

- 0xF0 (Set Scan Code Set) : Die Tastatur antwortet mit "Acknowledge" (0xFA) und liest dann ein Parameterbyte vom Host. Der Parameter kann 0x01, 0x02, oder 0x03 sein, womit Scancode-Set 1, 2 oder 3 ausgewählt wird. Die Tastatur antwortet auf den Parameter mit "Acknowledge" (0xFA). Wenn der Parameter 0x00 ist, antwortet die Tastatur mit "Acknowledge" (0xFA), gefolgt vom aktuellen Scancode-Set.

- 0xEE (Echo) : Die Tastatur antwortet mit "Echo" (0xEE).

- 0xED (Set/Reset LEDs) : Die Tastatur antwortet mit "Acknowledge" (0xFA) und liest dann ein Parameterbyte vom Host, mit dem die Num Lock-, Caps Lock- und Scroll Lock-LEDs eingestellt werden. Die Tastatur antwortet dann wieder mit "Acknowledge" (0xFA). Das Parameterbyte ist wie folgt definiert:

MSB							*LSB*
0	0	0	0	0	Caps Lock	Num Lock	Scroll Lock

Tabelle 3.3.1.3

Soll eine LED eingeschaltet werden, so ist das entsprechende Bit gleich 1. Wenn eine LED hingegen nicht leuchten soll, so ist das entsprechende Bit gleich 0. Soll zum Beispiel nur die Num Lock-LED leuchten, so geschieht dies nach folgenden Vorgängen:

  - Host sendet Befehl zum Setzen der LEDs (0xED).
  - Tastatur sendet Acknowledge (0xFA).
  - Host sendet Parameter-Byte 0000 0010b = (0x02).
  - Tastatur sendet “Acknowledge” (0xFA).

Die nächsten sechs Befehle können in jedem Modus an die Tastatur gesendet werden. Sie wirken sich jedoch nur im Modus 3 aus, d.h. wenn die Tastatur auf Scan-Code-Set 3 gestellt ist.

- 0xFD (Set Key Type Make) : Unterbindet Break-Codes und Wiederholfunktion für spezifizierte Tasten. Die Tastatur antwortet mit "Acknowledge" (0xFA) und stoppt die Tastenabfrage (sofern aktiv) und liest eine Liste von Tasten vom Host ein. Diese Tasten werden durch ihren Make-Code im Scan-Code-Set 3 spezifiziert. Die Tastatur antwortet auf jeden Make-Code mit "Acknowledge". Der Host beendet die Übertragung der Liste durch einen im Scan-Code-Set 3 nicht vorhandenen Make-Code (z.B. einen gültigen Befehl). Die Tastatur setzt dann mit der Tastenabfrage fort (sofern vorher deaktiviert).

- 0xFC (Set Key Type Make/Break) : Wie vorheriger Befehl, es wird jedoch nur die Wiederholfunktion unterbunden.

- 0xFB (Set Key Type Typematic) : Wie die beiden vorherigen Befehle, es werden jedoch nur die Break-Codes unterbunden.

- 0xFA (Set All Keys Typematic/Make/Break) : Die Tastatur antwortet mit "Acknowledge" (0xFA) und stellt die Normaleinstellung für alle Tasten her (MakeCodes, BreakCodes und Wiederholfunktion ein).

- 0xF9 (Set All Keys Make) : Die Tastatur antwortet mit "Acknowledge" (0xFA). Ähnlich wie 0xFD, gilt jedoch für alle Tasten.

- 0xF8 (Set All Keys Make/Break) : Die Tastatur antwortet mit "Acknowledge" (0xFA). Ähnlich wie 0xFC, gilt jedoch für alle Tasten.

Nachfolgend sind noch mal die Rückgabecodes, die die Tastatur zum Computer sendet aufgeführt:

- 0x00 (Überlauffehler) : Der Tastatur-Puffer ist übergelaufen.

- 0xFF (Tastenfehler) : Der empfangene Code ist der Tastatur unbekannt.

- 0xAA (BAT-Abschluss-Code) : Wird am erfolgreichen Ende des Tastatur-Selbsttests BAT (Basic Assurance Test) gesendet.

- 0xFC (BAT-Fehler) : Wird bei einem misslungenem Ende des Tastatur-Selbsttests BAT (Basic Assurance Test) gesendet.

- 0xEE (Echo) : Die Antwort auf einen Echo-Befehl (0xEE).

- 0xFA (Acknowledge) : Die Tastatur hat den letzten Befehl oder Parameter erfolgreich empfangen.

- 0xFE (Resend-Anforderung) : Die Tastatur möchte das letzte gesendete Byte vom Host noch einmal haben.

Nachfolgend die Kommunikation zwischen dem Host (Computer) und der Tastatur beim Bootvorgang aufgeführt. Da es hier keine Norm gibt oder im BIOS andere Werte eingestellt sein könnten (z.B. Num Lock aus), kann dies im einzelnen abweichen.

- Tastatur: (0xAA) : Selbsttest bestanden
- Computer: (0xED) : Setzen der Status-LEDs
- Tastatur: (0xFA) : Acknowledge
- Computer: (0x00) : Alle LEDs aus
- Tastatur: (0xFA) : Acknowledge
- Computer: (0xF2) : Lese ID
- Tastatur: (0xFA) : Acknowledge
- Tastatur: (0xAB) : Erstes Byte der ID
- Tastatur: (0x41) : Zweites Byte der ID
- Computer: (0xED) : Setzten der Status-LEDs
- Tastatur: (0xFA) : Acknowledge
- Computer: (0x02) : Num Lock LED an
- Tastatur: (0xFA) : Acknowledge
- Computer: (0xF3) : Wiederholrate und Verzögerung einstellen
- Tastatur: (0xFA) : Acknowledge
- Computer: (0x20) : Wiederholrate 30,0cps und Verzögerung 0,5s
- Tastatur: (0xFA) : Acknowledge
- Computer: (0xF4) : Aktiviert Tastenabfrage
- Tastatur: (0xFA) : Acknowledge

3.3.2 Hardware

Tastaturen benutzen ein bidirektionales, synchrones serielles Protokoll. Im Ruhezustand sind beide Busleitungen High. Nur in diesem Zustand darf die Tastatur mit der Datenübertragung beginnen. Der Computer hat die absolute Kontrolle über den Bus und darf die Kommunikation jederzeit unterbrechen, indem er die Clock-Leitung auf Low zieht.
Das Clock-Signal wird immer vom Eingabegerät erzeugt. Nachfolgend sind die möglichen Bus-Zustände aufgeführt:

Clock	*Data*	*Beschreibung*
1	1	Ruhezustand (Idle) : Der Computer ist bereit Daten von der Tastatur zu empfangen
1	0	Sendeanforderung (Request to Send) : Der Computer zeigt mit Startbit an, dass er Daten zur Tastatur senden will
0	1	Kommunikation gestoppt (Inhibit) : Der Computer ist beschäftigt und kann zur Zeit nichts empfangen
0	0	Zurücksetzen (Reset) : Der Computer wird gerade zurückgesetzt und die Tastatur führt einen Selbsttest durch

Tabelle 3.3.2.1

Alle Daten werden byteweise gesendet. Wenn die Tastatur Daten zum Computer senden will, dann besteht jedes Byte aus einem Frame zu 11 Bits. Will hingegen der Computer Daten zur Tastatur senden, so kommt am Ende noch ein Acknowledge-Bit hinzu und der Frame besteht aus 12 Bits:

- 1 Startbit : Dieses ist immer 0
- 8 Datenbits mit LSB zuerst
- 1 Paritäts-Bit (ungerade Parität)
- 1 Stopbit : Dieses ist immer 1
- 1 Acknowledge-Bit (nur bei Datenübertragung vom Computer zur Tastatur)

Das Paritäts-Bit ist 1, wenn das Datenbyte eine gerade Anzahl von Einsen enthält, und 0 bei einer ungeraden Anzahl. Die Anzahl der Einsen im Datenbyte plus das Paritäts-Bit ist immer ungerade (ungerade Parität). Dieses Verfahren dient zur Fehlererkennung. Die Tastatur muss das Paritäts-Bit prüfen und im Fehlerfall wie auf ein ungültiges Kommando reagieren. Daten, die von der Tastatur zum Computer gesendet werden, werden mit der fallenden Flanke des Clock-Signals gelesen; Daten vom Computer zur Tastatur mit der steigenden Flanke. Die Clock-Frequenz muss im Bereich von 10kHz bis 16,7kHz liegen. Das heißt, das Clock-Signal muss für 30µs bis 50µs High und für 30µs bis 50µs Low sein. Bei der Übertragung von der Tastatur zum Computer ist die Data-Leitung idealerweise in der Mitte jedes positiven Clock-Impulses zu ändern, d.h. 15µs bis 25µs nach der positiven Flanke (siehe Abbildung 3.3.2.2). Bei der Übertragung vom Computer zur Tastatur hingegen ist die Data-Leitung idealerweise in der Mitte jedes negativen Clock-Impulses zu ändern, d.h. 15µs bis 25µs nach der negativen Flanke (siehe Abbildung 3.3.2.4).

Wenn die Tastatur Daten senden will, prüft sie zunächst, ob die Clock- und Data-Leitung High sind. Falls nicht, blockiert der Computer gerade die Übertragung, und alle zu sendenden Zeichen müssen zwischengespeichert werden, bis der Computer die Clock- und Data-Leitung wieder freigibt. Die Clock- und Data-Leitung müssen für mindestens 50µs durchgehend High sein, bevor die Tastatur mit der Übertragung beginnen kann. Der Computer kann die Übertragung jederzeit unterbrechen, indem er die Clock-Leitung für mindestens 100µs auf Low zieht. Wenn eine Übertragung vor dem 11. Clock-Impuls unterbrochen wird, muss die Tastatur den Rest der Übertragung verwerfen und die Neuübertragung des aktuellen "Chunks" vorbereiten, wenn der Computer die Clock-Leitung wieder freigibt. Ein "Chunk" ist z.B. ein Make-Code, ein Break-Code oder eine Device-ID. Wird z.B. eine Tastatur bei der Übertragung des zweiten Byte eines Zwei-Byte Break-Codes unterbrochen, so muss der komplette Break-Code noch einmal gesendet werden, und nicht nur das zweite Byte. Wenn der Computer die Clock-Leitung vor dem ersten Clock-Impuls (fallende Flanke) oder nach der fallenden Flanke des letzten Clock-Impulses auf Low zieht, ist keine Wiederholung erforderlich. Fallen aber neue Daten an, die gesendet werden müssen, so müssen diese solange gespeichert werden, bis der Computer die Clock-Leitung freigibt. Tastaturen haben zu diesem Zweck einen 16 Byte Puffer. Fallen mehr als 16 Byte durch Tastaturanschläge an, werden diese ignoriert, bis im Puffer wieder freier Platz ist. Zur Verdeutlichung hier noch einmal die Schritte, die bei der Datenübertragung von der Tastatur zum Computer eingehalten werden müssen:

- Warten, bis die Tastatur Clock- und Data-Leitung freigegeben hat (Idle).
- Start-Bit (= 0) senden:
  - Die Data-Leitung auf Low bringen.
  - Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
  - Warten, bis die Tastatur die Clock-Leitung auf High bringt.
- Acht Daten-Bits (LSB zuerst) und ein Paritäts-Bit senden:
  - Die Data-Leitung entsprechend dem Wert des jeweiligen Daten-Bit auf 0 oder 1 setzen.
  - Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
  - Warten, bis die Tastatur die Clock-Leitung auf High bringt.
  - Wiederholen der letzten drei Schritte bis alle acht Daten-Bits und das Paritäts-Bit übertragen wurden.
- Stop-Bit (= 1) senden:
  - Die Data-Leitung freigeben.
  - Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
  - Warten, bis die Tastatur die Clock-Leitung auf High bringt.
- Warten, bis der Computer Clock- und Data-Leitung wieder freigibt (Idle).

Übertragung vom Computer (Host) zur Tastatur (Device):

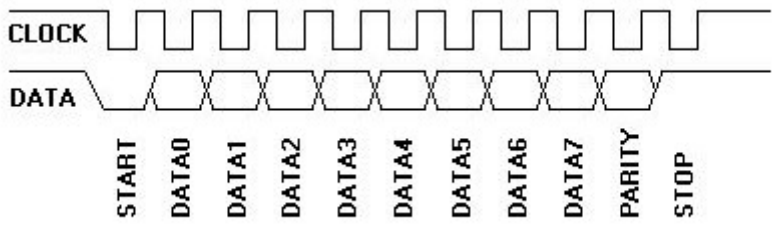


Abbildung 3.3.2.2

Scancode für die Taste "Q" (0x15), von der Tastatur zum Computer gesendet. Kanal A ist der Clock, Kanal B sind die Daten:

Abbildung 3.3.2.3

Will der Computer Daten zur Tastatur senden, sehen die Datenpakete etwas anders aus. Zunächst einmal werden die Clock- und Data-Leitung in den "Request to send"-Zustand gebracht, indem die Clock-Leitung für mindestens 100µs auf Low gezogen wird. Dann wird die Data-Leitung auf Low gezogen und anschließend die Clock-Leitung wieder freigegeben. Die Tastatur sollte den Bus in Abständen von maximal 100ms auf diesen Zustand prüfen. Erkennt sie diesen Zustand, beginnt sie Clock-Impulse für 8 Daten-Bits, Paritäts-Bit und ein Stop-Bit auszugeben. Nachdem das Stop-Bit empfangen wurde, bestätigt die Tastatur, indem Data Low gesetzt wird (Acknowledge) und ein letzter Clock-Impuls ausgegeben wird. Sollte der Computer die Data-Leitung nach dem 11. Clock-Impuls nicht freigeben, gibt die Tastatur solange Clock-Impulse aus, bis die Data-Leitung freigegeben wird (die Tastatur erzeugt dabei einen Fehler). Der Computer kann die Übertragung jederzeit vor dem 11. Clock-Impuls abbrechen, indem er Clock für mindestens 100µs Low hält. Zur Verdeutlichung hier noch einmal die Schritte, die bei der Datenübertragung vom Computer zur Tastatur eingehalten werden müssen:

- "Request to send" bzw. Start-Bit (= 0) senden:
  - Die Clock-Leitung für mindestens 100µs auf Low bringen.
  - Die Data-Leitung auf Low bringen.
  - Die Clock-Leitung wieder High werden lassen.
- Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
- Acht Daten-Bits (LSB zuerst) und ein Paritäts-Bit senden:
  - Die Data-Leitung entsprechend dem Wert des jeweiligen Daten-Bit auf 0 oder 1 setzen.
  - Warten, bis die Tastatur die Clock-Leitung auf High bringt.
  - Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
  - Wiederholen der letzten drei Schritte bis alle acht Daten-Bits und das Paritäts-Bit übertragen wurden.
- Stop-Bit (= 1) senden:
  - Die Data-Leitung freigeben.
  - Warten, bis die Tastatur die Clock-Leitung auf High bringt.

- "Acknowledge-Bit" wird vom Computer gesendet:
  - Warten, bis die Tastatur die Data-Leitung auf Low zieht.
  - Warten, bis die Tastatur die Clock-Leitung auf Low zieht.
- Warten, bis die Tastatur Clock- und Data-Leitung wieder freigibt (Idle).

Abbildung 3.3.2.4 zeigt diesen Vorgang und Abbildung 3.3.2.5 zeigt noch einmal getrennt, welche Signale vom Computer und welche von der Tastatur erzeugt werden. Dabei ist zu beachten, dass sich beim "Acknowledge-Bit" die Datenleitung, anders als üblich, ändert, während Clock High ist.

Übertragung von der Tastatur (Device) zum Computer (Host):

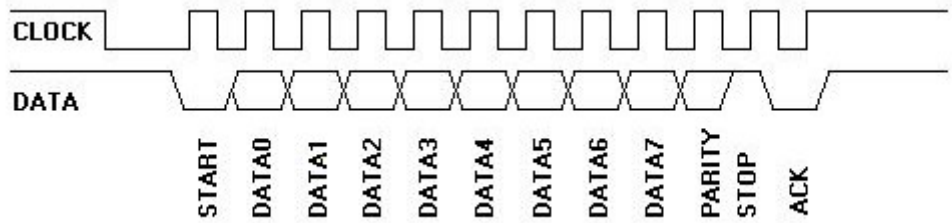


Abbildung 3.3.2.4

Übertragung von der Tastatur (Device) zum Computer (Host), getrennt für beide Geräte:

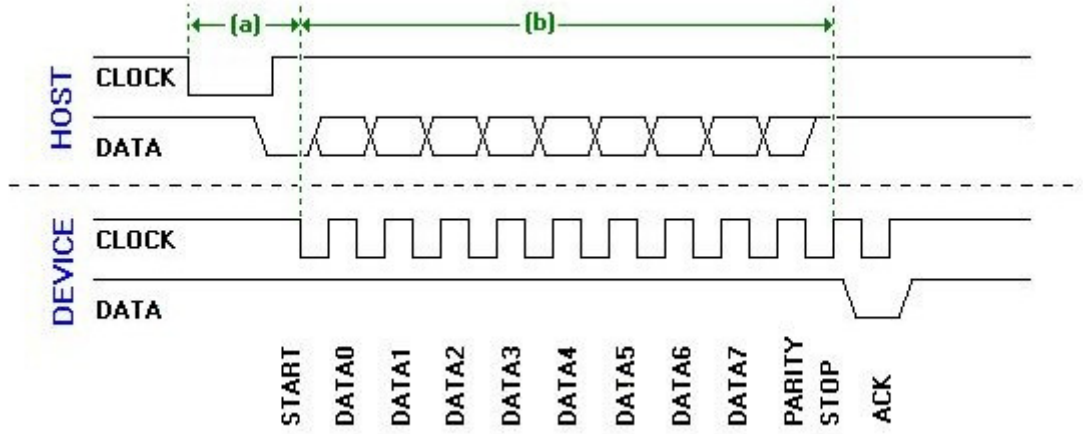


Abbildung 3.3.2.5

Bezugnehmend auf obige Abbildung 3.3.2.5 muss der Computer (Host) zwei Zeitabschnitte beachten:

- (a) ist die Zeit innerhalb der die Tastatur (Device) beginnt, Clock-Impulse auszugeben, nachdem der Computer erstmals die Clock-Leitung auf Low gesetzt hat. Sie darf nicht länger als 15ms sein.
- (b) ist die Zeit für die Übertragung des gesamten Datenpakets, die nicht größer als 2ms sein darf. Wird eine dieser Vorgaben nicht eingehalten, sollte der Computer einen „Error“ generieren. Unmittelbar nach dem „Acknowledge-Bit“ darf der Computer die Clock-Leitung auf Low bringen um die Kommunikation zu unterbrechen, während er die Daten verarbeitet. Wenn das vom Computer gesendete Kommando eine Quittierung verlangt, muss diese innerhalb von 20ms gesendet werden, nachdem die Clock-Leitung freigegeben wurde. Fehlt diese Quittierung, generiert der Computer einen Fehler.

4. Die Joystick-Schnittstelle

Viele Computer sind mit einer 15-poligen Sub-D-Buchse (siehe Abbildung 4.1) für den Anschluss von Joysticks ausgestattet. Dieser Anschluss hat vier quasi-analoge Eingänge, die über die Ladezeiten von vier Kondensatoren die Messung von Widerständen ermöglichen. Außerdem stehen vier digitale Eingänge zur Verfügung. Der Joystick-Anschluss führt darüber hinaus die PC-Betriebsspannung von 5V nach außen. Mittels eines Y-Kabels ist es möglich an einen Anschluss zwei Joysticks mit jeweils zwei Achsen und zwei Knöpfen anzuschließen.

Abbildung 4.1

Die quasi-analogen Eingänge dienen im Normalfall zur Messung des Widerstandes der Potentiometer von 0kΩ bis 100kΩ im Joystick. Nach einer Initialisierung durch die Software lädt jeder Widerstand einen Kondensator von 10nF auf. Sobald eine Ladespannung von 2/3 der Betriebsspannung erreicht ist, kippt ein interner Komperator um, und der Kondensator wird wieder entladen. Diese Schwellspannung berechnet sich wie folgt:

$$U_{Schwell} = \frac{2}{3} \cdot Vcc = \frac{2}{3} \cdot 5V = \underline{\underline{3{,}33V}}$$

Die Ladezeit des Kondensators ist proportional zum gemessenen Widerstand. Intern werden meist vierfach Timer vom Typ 558 (4x 555) verwendet. Die Eingänge sind durch Serienwiderstände mit 2,2kΩ geschützt.

Die vier digitalen Eingänge sind TTL-Eingänge. Zusätzlich werden im Computer Pull-Up-Widerstände von 1kΩ und Störschutzkondensatoren von 47pF verwendet. Die Eingänge sind für den Anschluss von Schaltern gegen Masse ausgelegt, können jedoch auch durch TTL-Ausgänge oder Transistoren angesteuert werden. Wegen der verwendeten Pull-Up-Widerstände fließt bei heruntergeschalteten Eingängen ein relativ großer Strom von etwa 5mA. Das kann beim Anschluss an CMOS-Ausgänge zu Problemen führen, wenn diese nur für geringere Ströme ausgelegt sind.

Der Joystick-Port belegt die Adresse 0x201 (siehe Tabelle 4.2) unter der die vier quasi-analogen Eingänge und die vier digitalen Eingänge gelesen werden können. Bei einem Schreibzugriff werden die vier quasi-analog-Eingänge für eine neue Messung zurückgesetzt. Eine zurückgelesene 0 an den Eingängen A0 bis A3 bedeutet, dass die Schwellspannung von 3,33V erreicht wurde und man durch die gemessene Zeit den entsprechenden Widerstandswert des Potentiometers im Joystick bestimmen kann. Bei den vier digitalen Eingängen D0 bis D3 bedeutet eine zurückgelesene 0, dass der entsprechende Knopf betätigt wurde.

Die Anschlussbelegung der Joystick-Schnittstelle geht aus folgender Tabelle hervor.

Pin	*Anschluss*		*Bit im Register 0x201*
	1 Joystick	*2 Joysticks*	
1, 8, 9, 15	5V	5V	/
4, 5, 12	GND	GND	/
3	A0 (Joystick X-Achse 1)	A0 (Joystick 1 X-Achse)	0
6	A1 (Joystick Y-Achse 1)	A1 (Joystick 1 Y-Achse)	1
11	A2 (Joystick X-Achse 2)	A2 (Joystick 2 X-Achse)	2
13	A3 (Joystick Y-Achse 2)	A3 (Joystick 2 Y-Achse)	3
2	D0 (Joystick Knopf 1)	D0 (Joystick 1 Knopf 1)	4
7	D1 (Joystick Knopf 2)	D1 (Joystick 1 Knopf 2)	5
10	D2 (Joystick Knopf 3)	D2 (Joystick 2 Knopf 1)	6
14	D3 (Joystick Knopf 4)	D3 (Joystick 2 Knopf 2)	7

Tabelle 4.2

5. Schaltungsbeschreibung

Der Microcontroller IC1 vom Typ Atmel ATmega32 ist das Herz der Schaltung. Er steuert die LCD-Anzeige an, fragt die 16er-Tastatur und den Joystick ab und regelt das Zusammenspiel mit der Computer-Tastatur und dem Computer. Er verfügt über 32 I/O-Pins, von denen 31 wie folgt benötigt werden:

Verwendung	*Anzahl I/O-Pins*
Joystick-Anschluss	8
LCD-Anzeige	8
Externes EEPROM IC2	2
Schalten von Analog-Schalter IC3	1
16er-Tastatur	8
Computer	2
Tastatur	2
Summe:	*31*

Tabelle 5.1

Er verfügt außerdem über 32KByte Flash-Speicher für das Programm, 1KByte EEPROM-Speicher und 2KByte SRAM-Speicher. Darüber hinaus besitzt er unter anderem acht Analog-Digital-Wandler mit jeweils 10Bit Auflösung. Über den Quarz X1 mit 16MHz und den zwei Keramik-Kondensatoren C1 und C2 mit jeweils 22pF wird seine Taktfrequenz mit 16MHz festgelegt. Dies ist auch die maximale Taktfrequenz, für die der ATmega32 ausgelegt ist. Es handelt sich bei ihm um einen RISC-Controller (Reduced Intruction Set = Reduzierter Befehls-Satz), bei dem die meisten Befehle innerhalb eines Taktzyklus abgearbeitet werden. Damit ergeben sich bis zu 16 Millionen Befehle pro Sekunde. Die Abarbeitungszeit pro Befehl liegt dann bei:

$$T = \frac{1}{f} = \frac{1}{16MHz} = \underline{\underline{62{,}5ns}}$$

Der Widerstand R1 sorgt zusammen mit dem Kondensator C3 für einen sauberen Einschalt-Reset. Der Kondensator C3 hält dabei auch noch HF-Störungen vom Reset-Pin fern, die zu einem unerwartetem Zurücksetzen des Gerätes führen würden. Zur Abfrage der Joystick-Achsen wird der interne Analog-Digital-Wandler eingesetzt. Hierfür wird die Versorgungsspannung von ca. 5V, über den AVcc-Pin, als Referenz-Spannungsquelle benutzt. Kondensator C9 hält dabei HF-Störungen von dem AREF-Pin fort, die zu einer Verfälschung der Analog-Digital-Wandlung führen würden (siehe ATmega32-Datenblatt Seite 206). Die Kondensatoren C9, C10, C11, C12 und C13 filtert HF-Störungen an den Versorgungsspannungs-Anschlüssen von IC1 heraus.

Das serielle I²C-EEPROM IC2 vom Typ 24C512 speichert die Tastatur-Sequenzen, die PIN, die Werte für LCD-Hintergrundbeleuchtung und –Kontrast und die Joystick-Kalibrierwerte. Es besitzt eine Speicherkapazität von 64KByte. Angeschlossen ist es an die I²C-Anschüsse des Microcontrollers IC1. Die beiden Pull-Up-Widerstände R2 und R3 sorgen für einen definierten High-Pegel an den I²C-Bus-Leitungen SDA (serial data = Datenleitung) und SCL (serial clock = Taktleitung). Über die Adress-Eingänge A0 und A1 wird das EEPROM adressiert, da es möglich ist mehrere Bausteine an den I²C-Bus zu hängen. Der WP-Eingang (Write-Protect) ermöglicht einen Schreibschutz, der jedoch durch Anlegen von Masse ausgeschaltet ist. Der Kondensator C7 filtert HF-Störungen am Versorgungsspannungs-Anschluss von IC2 heraus.

Der Analog-Schalter IC3 vom Typ 74HCT4066 verbindet im Normalbetrieb die Clock- und Data-Leitungen von Computer-Tastatur und Computer. Dadurch ist eine Bedienung des Computers über die Computer-Tastatur so möglich, als wäre sie direkt am Computer angeschlossen. Im Programmier-Modus werden die Schalter geöffnet, um eventuelle Rückmeldungen des Computers nicht als Tasten-Code von der Computer-Tastatur zu speichern. Auch wenn eine Tastatur-Sequenz an den Computer gesendet werden soll, werden die Schalter geöffnet, um zu verhindern, dass die Tasten-Codes, die ja nur für den Computer bestimmt sind, auch an die Computer-Tastatur gesendet werden. Die Computer-Tastatur würde sonst so reagieren, dass sie die empfangenen Codes mit sehr großer Wahrscheinlichkeit nicht als gültigen Befehlscode für die Tastatur erkennt und ihrerseits versucht ein 0xFE als Zeichen für einen Fehler an den Computer zu senden. Da jedoch schon der Microcontroller IC1 Daten an den Computer sendet würde die gesamte Übertragung gestört werden. Das Öffnen und Schließen der Schalter wird vom Microcontroller IC1 gesteuert. Bei den zwei überflüssigen Gatter von IC3 (IC3C und IC3D) wird jeweils der Schalt-Eingang und ein Schalter-Anschluss mit Masse verbunden, um Schwingungen zu vermeiden, die den Stromverbrauch erhöhen und sogar zur Zerstörung des ICs führen können. Es gibt zwar auch Analog-Schalter mit nur zwei Schaltern (z.B. MAX323), diese sind aber um ein vielfaches teurer als das hier gewählte Exemplar. Die Widerstände im Widerstands-Netzwerk RN2 sorgen als Pull-Up-Widerstände für einen definierten High-Pegel an den Clock- und Data-Leitungen der Computer-Tastatur- und der Computer-Schnittstelle. Die Pins PD0 bis PD3 am Microcontroller IC1 bilden per Programm die Open-Collector-Schnittstelle des Tastatur-Anschlusses nach, in dem zwar Low-Pegel ausgegeben wird, High-Pegel jedoch, durch Schalten des entsprechenden Pins als Eingang, über einen Pull-Up-Widerstand aus dem Widerstandsnetzwerk RN2 definiert ist. Die Taktleitung der Computer-Tastatur ist an den INT0-Anschluss des Microcontrollers IC1 angeschlossen, um schnell per Interrupt-Routine auf eingehende Daten reagieren zu können, ohne die Computer-Tastatur zyklisch abfragen zu müssen. Der Kondensator C8 filtert HF-Störungen am Versorgungsspannungs-Anschluss von IC3 heraus.

Der DC/DC-Wander IC4 liefert aus einer Eingangsspannung von 5V an seinem Ausgang eine unstabilisierte Spannung von ca. 12V. Diese wird wiederum dazu benutzt mit Hilfe des Spannungsreglers VR1 vom Typ 78L05 eine stabile 5V-Spannung zu erzeugen, die als Spannungsversorgung für die LCD-Hintergrundbeleuchtung und –Kontrast dient. Die zwei Kondensatoren C5 und C6 sorgen dafür, dass der Spannungsregler VR1 nicht anfängt zu schwingen. Da die Spannungsversorgung der gesamten Schaltung und der Computer-Tastatur über den Computer erfolgt fällt bei erhöhtem Strombedarf (z.B. alle drei Tastatur-LEDs an) an der internen Polyfuse im Computer zusätzliche Spannung ab. Dadurch sinkt die Versorgungsspannung geringfügig ab. Würde man die Spannungsversorgung für die LCD-Hintergrundbeleuchtung und –Kontrast über die normale Spannungsversorgung herstellen, so würde sich bereits dieser geringe Spannungsunterschied in einer deutlich sichtbaren Veränderung der Helligkeit auswirken. Da an den LEDs der LCD-Hintergrundbeleuchtung bereits eine Spannung von ca. 3,4V abfällt war für eine Konstantstromquelle nicht mehr genug Spannung übrig, so dass zu dieser vielleicht etwas ungewöhnlichen Methode gegriffen wurde. Die übrige Schaltung auch mit einer stabilisierten Spannung zu betreiben ist nicht nötig, da die Spannungsunterschiede zu gering und noch deutlich in den, laut Bauteil-Datenblätter (IC1, IC2, IC3 und LCD), erlaubten Tolleranzen sind. Außerdem würde man in dem Fall einen größeren Spannungsregler VR1 und einen größeren DC/DC-Wandler benötigen. Da ein DC/DC-Wandler nur einen schlechten Wirkungsgrad (hier 70%) hat würde dies zu einem Anstieg der Stromaufnahme führen, was bei einem maximalen Strom von gerade mal 275mA, die ein Tastatur-Anschluss liefern kann (siehe Kapitel 3.2), nicht gerade vorteilhaft wäre.

Die LCD-Anzeige kann zwei Zeilen mit jeweils 16 Zeichen darstellen und hat eine grüne Hintergrundbeleuchtung. Sie hat einen KS0066-Controller (oder kompatibel), der auch in vielen anderen LCD-Anzeigen verwendet wird und über den eine einfache Ansteuerung möglich ist. Sie wird im 4-Bit-Modus betrieben, also sind die Anschlüsse DB0, DB1, DB2 und DB3 auf Masse gelegt. Da der R/W-Anschluss nicht vom Microcontroller IC1 angesteuert wird, sondern dauerhaft auf Masse gelegt ist, kann man an die LCD-Anzeige nur Schreiben. Ein Lesen z.B. des Busy-Flag ist nicht möglich, aber auch nicht nötig, wenn man sich an die Timing-Angaben (LCD-Datenblatt Seite 5) hält. Der Microcontroller IC1 steuert über ein PWM-Signal an Widerstand R8 und Transistor T1 die Helligkeit der Hintergrundbeleuchtung. Über den Widerstand R7 wird dabei über den Strom die maximale Helligkeit der Hintergrundbeleuchtung bestimmt (I = U : R). Der Microcontroller IC1 steuert außerdem über ein PWM-Signal an Widerstand R6 und Transistor T2 den Kontrast der LCD-Anzeige. Die Widerstände R4 und R5 bilden einen Spannungsteiler, an dessen Mitte die LCD-Kontrast-Spannung abgegriffen wird. Die Anschlüsse DB4, DB5, DB6 und DB7 sind die Datenleitungen, über die der Microcontroller IC1 Befehle und Daten an die LCD-Anzeige sendet. Über den Anschluss RS teilet der Microcontroller IC1 der LCD-Anzeige mit, ob es sich bei den anliegenden Daten um Befehle (RS = 0) oder Nutzdaten (RS = 1) handelt. Über einen kurzen negativen Impuls am Anschluss E werden die anliegenden Daten bzw. Befehle von der LCD-Anzeige übernommen. Der Kondensator C4 filtert HF-Störungen am Versorgungsspannungs-Anschluss der LCD-Anzeige heraus.

Die Joystick-Signale werden dem Microcontroller IC1 über den Steckverbinder K3 zugeführt. Die vier Joystick-Knöpfe, die ja gegen Masse schalten (siehe Kapitel 4), sind über die internen Pull-Up-Widerstände des Microcontrollers IC1 mit den Pins PA0 bis PA3 verbunden, die per Programm als digitale Eingänge geschaltet sind. Die Auswertung der Joystick-Potentiometer erfolgt nicht wie in Kapitel 4 beschrieben, da dies zu erheblich mehr Hard- und Softwareaufwand geführt hätte. Die vier Potentiometer von ca. 100kΩ, die je für eine Joystick-Achse zuständig sind, bilden mit den Widerständen (je 100kΩ) im Widerstandsnetzwerk RN1 Spannungsteiler, deren Spannungen auf die Pins PA4 bis PA7, die per Programm als analoge Eingänge geschaltet sind, geführt werden. Da die Joystick-Potentiometer leider nicht an Masse angeschlossen sind, ist also jeweils ein Spannungsteiler notwendig, da sonst unabhängig der Potentiometer-Stellung immer nur 5V gemessen würden. Durch den 1:1-Spannungsteiler misst der Microcontroller IC1 statt von 0V bis 5V nur von 2,5V bis 5V, die Hälfte des Messbereichs ist also verloren. Das macht jedoch gar nichts, da ein angeschlossener Analog-Joystick sowieso nur als Digital-Joystick betrachtet wird und von einer Joystick-Achse nur fünf Werte benötigt werden (siehe Abbildung 6.2).

Die 16er-Tastatur ist aus einer Matrix zu vier Zeilen und vier Spalten aufgebaut. Beim Druck auf eine Taste wird eine Zeile mit einer Spalte verbunden, aus denen sich dann die gedrückte Taste bestimmen lässt.

An Steckverbinder K1 wird die LCD-Anzeige, an K2 die 16er-Tastatur, an K3 die Joystick-Buchse, an K4 über ein Kabel mit PS/2-Stecker der Computer, an K5 die Computer-Tastatur-Buchse und an K6 ggf. der Programmieradapter zum Aufspielen der Software angeschlossen.

6. Programmbeschreibung

Die Zeilennummern gehören nicht zum eigentlichen Programm-Quelltext und dienen nur der besseren Orientierung.

In den Zeilen 1 bis 7 werden die externen Bibliotheken für den C-Compiler eingebunden.

In den Zeilen 9 bis 102 stehen die Definitionen. Die gesamten Signal-Leitungen für LCD-Anzeige, 16er-Tastatur, Tastatur- und Computer-Anschluss, Joystick-Anschluss, Analogschalter IC3 und externes EEPROM IC2 sind wie im Schaltplan definiert. Außerdem sind hier die Steuer-Tasten der 16er-Tastatur, Zeitwerte z.B. für das Entprellen der Tasten, die Startadressen im externen EEPROM IC2, die Reihenfolge der Setup-Einträge und noch ein paar andere Sachen definiert. Dies alles erhöht die Lesbarkeit des Programms und vereinfacht deren eventuelle Änderung.

In den Zeilen 104 bis 140 werden die globalen Strukturen definiert.

In den Zeilen 142 bis 154 werden die globalen Variablen definiert.

In den Zeilen 158 bis 168 steht die Funktion „wait_us“, die eine als Parameter anzugebende Zeit in µs wartet. Sie ruft wiederum die Funktion „_delay_us“ auf, die hier 1µs wartet. Die Funktion „wait_us“ kann nicht einfach durch „_delay_us“ ersetzt werden, da „_delay_us“ nur eine geringe Wartezeit zulässt und öfters größere Pausen eingelegt werden müssen. Die maximale Wartezeit bei „_delay_us“ ist laut Beschreibung (AVR-Bibliothek Seite 15):

$$t = \frac{768\mu s}{f_{CPU}[MHz]} = \frac{768\mu s}{16} = \underline{\underline{48\mu s}}$$

Die zusätzliche Zeit, die durch die Schleife verloren geht kann nicht ganz vernachlässigt werden. Deshalb wird in Zeile 162 ein Korrektur-Faktor berechnet, mit dem die Schleife läuft. Dieser Korrektur-Faktor von 10/15 wurde experimentell ermittelt. Die eingelegten Pausen sind dadurch natürlich nicht ganz so genau.

In den Zeilen 171 bis 180 steht die Funktion „wait_ms“, die eine als Parameter anzugebende Zeit in ms wartet. Sie ruft wiederum die Funktion „_delay_ms“ auf, die hier 1ms wartet. Die Funktion „wait_ms“ kann nicht einfach durch „_delay_ms“ ersetzt werden, da „_delay_ms“ nur eine geringe Wartezeit zulässt und öfters größere Pausen eingelegt werden müssen. Die maximale Wartezeit bei „_delay_ms“ ist laut Beschreibung (AVR-Bibliothek Seite 14):

$$t = \frac{262{,}14\,ms}{f_{CPU}[MHz]} = \frac{262{,}14\,ms}{16} = \underline{\underline{16{,}38\,ms}}$$

Die zusätzliche Zeit, die durch die Schleife verloren geht kann vernachlässigt werden.

In den Zeilen 183 bis 196 steht die Funktion „lcd_hex_4“, die ein Nibble „z“ an die LCD-Anzeige sendet. Mit dem zweiten Parameter „rs“ wird angegeben, ob es sich bei „z“ um einen Befehl (rs = 0) oder um Daten (rs = 1) handelt. Mit einem kurzen negativen Impuls an LCD_E (siehe Datenblatt Seite 3) wird das Nibble von der LCD-Anzeige übernommen.

In den Zeilen 199 bis 205 steht die Funktion „lcd_hex_8“, die ein Byte „z“ an die LCD-Anzeige sendet. Mit dem zweiten Parameter „rs“ wird angegeben, ob es sich bei z um einen Befehl (rs = 0) oder um Daten (rs = 1) handelt. Über die Funktion lcd_hex_4 (siehe oben) wird zuerst das High- und dann das Low-Nibble an die LCD-Anzeige gesendet.

In den Zeilen 208 bis 214 steht die Funktion lcd_clear, die die LCD-Anzeige löscht.

In den Zeilen 217 bis 261 steht die Funktion „lcd_init“, die die LCD-Anzeige initialisiert. Zuerst werden die entsprechenden Port-Pins als Ausgang gesetzt und LCD_E zusätzlich High gesetzt. Dann wird die Initialisierungs-Sequenz (siehe LCD-Datenblatt Seite 5 + 6) gestartet, in der folgende Einstellungen vorgenommen werden:

- 4-Bit Interface
- 2 Zeilen LCD-Anzeige
- 5x7 Zeichensatz
- LCD-Anzeige anschalten
- LCD-Anzeige löschen
- Adresspointer inkrementieren
- Displayinhalt nicht schieben

Zum Schluss werden noch die Timer-Zähler für die Hintergrundbeleuchtung und den Kontrast auf 0 gesetzt.

In den Zeilen 264 bis 275 steht die Funktion „ps2_init“, in der die Schnittstellen von Tastatur und Computer initialisiert werden. Zuerst werden die entsprechenden Port-Pins als Eingang geschaltet. Dann werden die mit Data- und Clock-Signal belegten Schalter des Analog-Schalters IC3 geschlossen, wodurch die Tastatur mit dem Computer verbunden ist. Das ist wichtig, damit die Tastatur beim Hochfahren des Computers gefunden wird und man normal mit der Tastatur arbeiten kann. Da die Schnittstellen für Tastatur und Computer als Open-Collector-Schnittstellen (siehe Kapitel 3.2) arbeiten müssen, sind Data- und Clock-Leitungen zukünftig wie folgt anzusteuern, PC_CLOCK ist dann ggf. zu ersetzen:

Signal-Zustand	*Pin-Zustand*	*Programm-Sequenz*
Eingang	Eingang	`PS2_DDR=PS2_DDR & ~(1<<PC_CLOCK);`
High	Eingang	`PS2_DDR=PS2_DDR & ~(1<<PC_CLOCK);`
Low	Ausgang = Low	`PS2_PORT=PS2_PORT & ~(1<<PC_CLOCK);` `PS2_DDR=PS2_DDR | (1<<PC_CLOCK);`

Tabelle 6.1

Der High-Pegel kommt dann durch einen Pull-Up-Widerstand in RN1 zustande.

In den Zeilen 278 bis 304 steht die Funktion „key16_init“, in der die 16er-Tastatur initialisiert wird. Zuerst werden die Leitungen für die Zeilen und Spalten der 16er-Tastatur als Eingang geschaltet und die internen Pull-Up-Widerstände eingeschaltet. Danach werden die zur 16er-Tastatur gehörenden Variablen initialisiert.

In den Zeilen 307 bis 345 steht die Funktion „joy_init“, in der der Joystick-Anschluss initialisiert wird. Zuerst werden die internen Pull-Up-Widerstände für die Joystick-Knöpfe eingeschaltet und die Joystick-Port-Pins als Eingang geschaltet. Dann wird die Spannung am AVcc-Pin von IC1 als Referenzspannung definiert und das Ergebnis der AD-Wandlung auf 8 Bit beschränkt. Zuletzt werden die zum Joystick gehörenden Variablen initialisiert.

In den Zeilen 348 bis 361 steht die Funktion „ext_eeprom_init“, die das externe I²C-EEPROM IC2 initialisiert. Zuerst werden die SDA- und SCL-Leitungen als Eingang gesetzt. Dann wird der Vorteiler für die Geschwindigkeit auf 1 (= kein Vorteiler) (siehe Atmega32-Datenblatt Seite 177) gesetzt. Zuletzt wird die Bit-Rate für die Geschwindigkeit laut Beschreibung (Atmega32-Datenblatt Seite 173) wie folgt berechnet:

$$f_{SCL} = \frac{f_{CPU}}{16 + 2 \cdot TWBR \cdot 4^{TWPS}}$$

umgestellt zu:

$$TWBR = \frac{\frac{f_{CPU}}{f_{SCL}} - 16}{2 \cdot 4^{TWPS}}$$

mit TWPS=0 (Vorteiler-Bits) ergibt sich:

$$TWBR = \frac{\frac{f_{CPU}}{f_{SCL}} - 16}{2}$$

In den Zeilen 364 bis 392 steht die Funktion „uc_init“, in der bisher noch nicht initialisierte Teile initialisiert werden. Zuerst werden die Vorteiler für Timer0 (zuständig für LCD-Hintergrundbeleuchtung und –Kontrast) und Timer2 (zuständig für die Abfrage der 16er-Tastatur und des Joysticks) bestimmt. Die Frequenz, mit der einer dieser Timer ausgeführt wird, wird wie folgt berechnet:

$$f_{TOF} = \frac{f_{CPU}}{N \cdot 256}$$

für Timer0 ergibt sich dann:

$$f_{TOF} = \frac{16\,MHz}{1 \cdot 256} = \underline{\underline{62{,}5\,kHz}}$$

$$T_{TOF} = \frac{1}{f_{TOF}} = \frac{1}{62{,}5\,kHz} = \underline{\underline{16\,\mu s}}$$

für Timer2 ergibt sich dann:

$$f_{TOF} = \frac{16\,MHz}{8 \cdot 256} = \underline{\underline{7{,}8\,kHz}}$$

$$T_{TOF} = \frac{1}{f_{TOF}} = \frac{1}{7{,}8\,kHz} = \underline{\underline{128\,\mu s}}$$

Danach wird der Timer2-Zeit-Wert „debounce“ für die Entprellung der 16er-Tastatur und des Joysticks wie folgt berechnet:

$$debounce = \frac{t_{debounce} \cdot f_{CPU}}{N \cdot 256} = \frac{20\,ms \cdot 16\,MHz}{8 \cdot 256} = \underline{\underline{156}}$$

Danach wird der Timer2-Zeit-Wert „repeat_start“ (Verzögerung – siehe Tabelle 3.3.1.2) für die Erkennung einer lang gedrückten Taste auf der 16er-Tastatur bzw. des Joysticks wie folgt berechnet:

$$repeat\_start = \frac{t_{repeat\_start}}{t_{debounce}} = \frac{500ms}{20ms} = \underline{\underline{25}}$$

Danach wird der Timer2-Zeit-Wert „repeat_next“ (Wiederholrate – siehe Tabelle 3.3.1.1) für die Wiederholung einer lang gedrückten Taste auf der 16er-Tastatur bzw. des Joysticks wie folgt berechnet:

$$repeat\_next = \frac{t_{repeat\_next}}{t_{debounce}} = \frac{200ms}{20ms} = \underline{\underline{10}}$$

Dann werden die Zähler für die beiden benutzten Timer initialisiert und die Interrupts eingeschaltet.

In den Zeilen 396 bis 411 steht die Funktion „lcd_print“, mit der man, an einer anzugebenden Position, eine Zeichenkette auf der LCD-Anzeige ausgeben kann.

In den Zeilen 416 bis 480 steht die Funktion „key16_scan“, die die 16er-Tastatur abfragt. Dazu werden nacheinander eine der vier Zeilen-Leitungen von hochohmig auf Low geschaltet und überprüft, welche Spalte Low-Potential hat. Aus der getriebenen Zeile und der eingelesenen Spalte wird dann die gedrückte Taste bestimmt und zurückgegeben.

In den Zeilen 483 bis 499 steht die Funktion „ext_eeprom_start“, mit der eine Übertragung zum externen EEPROM IC2 gestartet wird. Dabei wird jedoch nicht, wie bei der Funktion „ext_eeprom_start_wait“ (siehe unten), geschaut, ob das EEPROM tatsächlich für eine Übertragung bereit ist. Diese Funktion wird nur in der Funktion „ext_eeprom_read_buffer“ (siehe unten) verwendet, um das EEPROM, nach erfolgreichem Start, auf Lesen zu stellen.

In den Zeilen 503 bis 544 steht die Funktion „ext_eeprom_start_wait“, mit der eine Übertragung zum externen EEPROM IC2 gestartet wird. Danach wird geschaut, ob das EEPROM tatsächlich für eine Übertragung bereit ist.

In den Zeilen 547 bis 556 steht die Funktion „ext_eeprom_stop“, mit der die Übertragung zum externen EEPROM IC2 beendet wird.

In den Zeilen 559 bis 569 steht die Funktion „ext_eeprom_write_byte“, mit der ein Byte zum externen EEPROM IC2 gesendet wird.

In den Zeilen 572 bis 581 steht die Funktion „ext_eeprom_read_ack“ mit der ein Byte vom externen EEPROM IC2 gelesen wird und mit einem „Acknowledge“ quittiert wird. Ein Lesen von weiteren Bytes, ohne die Übertragung neu zu starten, ist dadurch möglich.

In den Zeilen 584 bis 594 steht die Funktion „ext_eeprom_read_nack“, mit der ein Byte vom externen EEPROM IC2 gelesen wird ohne dies allerdings, wie bei „ext_eeprom_read_ack“ (siehe oben), mit einem „Acknowledge“ zu quittieren. Ein Lesen von weiteren Bytes, ohne die Übertragung neu zu starten, ist nicht möglich. Hiermit sollte also das letzte gewünschte Byte gelesen werden.

In den Zeilen 597 bis 637 steht die Funktion „ext_eeprom_write_buffer“, mit der ein zu übergebender Puffer-Speicher, ab einer ebenfalls anzugebender EEPROM-Startadresse, gespeichert wird. Das letzte zu speichernde Byte ist im Puffer-Speicher durch eine 0 zu kennzeichnen. Da das externe EEPROM IC2 intern zu Seiten je 128 Byte organisiert ist (siehe 24C512-Datenblatt Seite 7), ist dies zu berücksichtigen und ggf. eine neue Startadresse zu senden.

In den Zeilen 641 bis 667 steht die Funktion „ext_eeprom_read_buffer“, mit der aus dem externen EEPROM IC2, ab einer anzugebenden EEPROM-Startadresse, in einen ebenfalls anzugebenden Puffer-Speicher gelesen wird. Das Ende wird dabei durch eine 0 oder ein 0xFF gekennzeichnet.

In den Zeilen 670 bis 684 steht die Funktion „pc_lowhigh“, mit der ein 40µs langer negativer Impuls auf die Clock-Leitung des Computers gegeben wird. Am Anfang und am Ende sind nochmals je 20µs Pause eingebaut. Diese Funktion wird dazu benutzt, wenn Daten an den Computer gesendet werden und ProggyKeyJoy die Tastatur simulieren, und dabei den Takt erzeugen muss (siehe Kapitel 3.3.2).

In den Zeilen 687 bis 752 steht die Funktion „ch_to_pc“, die ein Byte an den Computer sendet. Zuerst wird dazu der Timer2-Interrupt abgeschaltet, der für die Abfrage der 16er-Tastatur und des Joysticks zuständig ist, um dadurch Störungen in der Übertragung zu vermeiden. Dann wird gewartet, bis Data- und Clock-Leitung für mindestens 50µs High-Pegel haben, was notwendig ist, um die Empfangsbereitschaft des Computers sicherzustellen. Dann wird zuerst das Start-Bit, die acht Daten-Bits, das Paritäts-Bit und das Stop-Bit an den Computer gesendet. Zur Takterzeugung wird dabei die Funktion „pc_lowhigh“ (siehe oben) eingesetzt und die Daten werden in der Mitte des positiven Takt-Impulses geändert. Das Hardware-Protokoll ist in Kaptitel 3.3.2 beschrieben. Am Ende wird der Timer2-Interrupt, und damit die Abfrage der 16er-Tastatur und des Joysticks, wieder eingeschaltet.

In den Zeilen 755 bis 799 steht die Funktion „send_to_pc“, die eine ganze Tasten-Sequenz an den Computer sendet. Als Parameter wird ihr entweder eine der Tasten 0 bis 9 auf der 16er-Tastatur (bei Keyset oder Passwordset) oder die Nummer (0 bis 11) einer Joystick-Funktion (bei Joyset) übergeben. Aus ihr, und dem jeweils eingestellten Satz (0 bis 9), wird die EEPROM-Startadresse berechnet, an der die Tasten-Sequenz gespeichert ist (siehe Anhang 10.5). Dann wird die Tasten-Sequenz aus dem externen EEPROM IC2 gelesen und die Verbindung zwischen Tastatur und Computer getrennt. Danach wird die Tasten-Sequenz solange an den Computer gesendet, bis die entsprechende Taste bzw. Joystick-Funktion gelöst wurde. Nach jedem Senden wird dabei eine Pause von 20ms eingelegt, in der der Timer2-Interrupt eingeschaltet ist, um die 16er-Tastatur und den Joystick abzufragen. Am Ende wird die Verbindung zwischen Tastatur und Computer wieder hergestellt.

In den Zeilen 802 bis 825 steht die Funktion „show_set“, die den eingestellten Modus (Keyset, Joyset oder Passwordset) und den jeweils eingestellte Satz (0 bis 9) auf der LCD-Anzeige anzeigt.

In den Zeilen 829 bis 846 steht die Funktion „set_keyset“, mit der der Satz (0 bis 9) des aktuellen Modus (Keyset, Joyset oder Passwordset) eingestellt werden kann. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 849 bis 868 steht die Funktion „enter_pin“, mit der man eine 5-stellige PIN (für den Modus Passwordset), über die Tasten 0 bis 9 der 16er-Tastatur, eingeben kann. Jede Ziffer der PIN wird dabei um eins erhöht (0 -> 1, ... ,9 -> 10), damit es später beim Speichern der PIN im externen EEPROM IC2 keine Probleme gibt. Die verwendete Funktion ext_eeprom_write_buffer benutzt ja die 0 als Ende-Markierung (siehe oben). Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 871 bis 916 steht die Funktion „set_keypasswordset“, mit der man in den Modus Passwordset gelangen kann und den zugehörigen Satz einstellen kann. Dazu wird zuerst die 5-stellige PIN aus dem externen EEPROM IC2 gelesen. Ist dort eine PIN abgelegt worden und befindet man sich nicht schon im Modus Passwordset, so muss diese über die 16er-Tastatur eingegeben werden. Zur Eingabe der PINs dient die Funktion enter_pin (siehe oben). Gibt man die PIN falsch ein, so erscheint für 0,5s die Meldung „Error!“ und man gelangt wieder in den vorigen Modus und Satz. Gibt man die PIN korrekt ein oder ist noch keine PIN im externen EEPROM IC2 abgelegt oder befindet man sich bereits im Modus Passwordset, so wird man nach dem gewünschten Passwort-Satz gefragt. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 919 bis 1002 steht die Funktion „setup_backlight_contrast“, mit der man die LCD-Hintergrundbeleuchtung, bzw. –Kontrast einstellen kann. Dazu gibt man über eine der Tasten 0 bis 9 der 16er-Tastatur den neuen gewünschten Wert ein. Bestätigt man die Eingabe mit der Taste OK und ist der neue eingegebene Wert ein anderer als der Alte, dann wird der neue Wert im externen EEPROM IC2 gespeichert und man bekommt für 0,5s die Meldung, dass der entsprechende Wert gespeichert wurde. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 1005 bis 1052 steht die Funktion „enter_new_pins“, in der man zwei mal aufgefordert wird eine neue PIN einzugeben. Die zweite Eingabe dient dabei zur Sicherheit, dass man sich nicht vertippt hat. Gibt man die PINs korrekt ein, so wird die neue PIN im externen EEPROM IC2 gespeichert und eine 0,5s lange Meldung „PIN saved!“ auf der LCD-Anzeige ausgegeben. Macht man hingegen bei der zweiten Eingabe der PIN einen Fehler, so wird eine 0,5s lange Meldung „Error!“ auf der LCD-Anzeige ausgegeben und die PIN nicht gespeichert. Zur Eingabe der PINs dient die Funktion enter_pin (siehe oben). Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 1055 bis 1106 steht die Funktion „setup_pin“, die zur Eingabe einer neuen PIN dient. Dazu muss, falls vorhanden, erst die alte PIN richtig eingegeben werden. Danach kann man die neue PIN eingeben (Funktion enter_new_pins – siehe oben). Macht man bei der Eingabe der alten PIN einen Fehler, so wird eine 0,5s lange Meldung „Error!“ auf der LCD-Anzeige ausgegeben. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

In den Zeilen 1109 bis 1228 steht die Funktion „setup_joystick“, in der ein angeschlossener Joystick kalibriert werden kann. Dazu muss man zuerst die Joystick-Achsen in Mittelstellung bringen und einen Joystick-Knopf betätigen. Dann muss man alle Joystick-Achsen bis zum jeweiligen Vollausschlag bewegen und wieder einen Joystick-Knopf betätigen. Mit den daraus ermittelten Daten für Mittelstellung und Vollausschlag werden die Auslösewerte, die bei der Hälft des Vollausschlags liegen sollen, bestimmt. Da die gemessenen Werte nicht linear zu den Joystick-Potentiometern und damit zu den zurückgelegten Werten des Joystick-Hebels sind, kann man nicht einfach die Hälfte zwischen Mitte und Minimum bzw. Maximum als Auslösepunke nehmen. Der Auslösepunkt für einen kleiner werdenden Widerstandswert des Joystick-Potentiometers wäre sonst viel weiter von der Mitte entfernt, als der Auslösepunkt für einen größer werdenden Widerstandswert des Joystick-Potentiometers (siehe Abbildung 6.2). Um hier eine Symmetrie zu erzeugen werden dazu zuerst die Widerstandswerte aus den gemessenen Werten wie folgt zurückgerechnet:

$$R_{mid} = \frac{100k\Omega \cdot 255}{X_{mid}} - 100k\Omega$$

$$R_{\min} = \frac{100k\Omega \cdot 255}{X_{\max}} - 100k\Omega$$

$$R_{\max} = \frac{100k\Omega \cdot 255}{X_{\min}} - 100k\Omega$$

Mit diesen Widerstandswerten werden jetzt die Auslösewerte, bei denen eine Bewegung des Joysticks erkannt werden soll, wie folgt bestimmt:

$$X_{mid\_\max} = \frac{100k\Omega \cdot 255}{R_{\min} + \left(\frac{R_{mid} - R_{\min}}{2} \right) + 100k\Omega}$$

$$X_{mid\_\min} = \frac{100k\Omega \cdot 255}{R_{mid} + \left(\frac{R_{\max} - R_{mid}}{2} \right) + 100k\Omega}$$

Danach werden die Auslösewerte im externen EEPROM IC2 gespeichert, und die Meldung „Joystick calibrated!“ auf der LCD-Anzeige ausgegeben. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden.

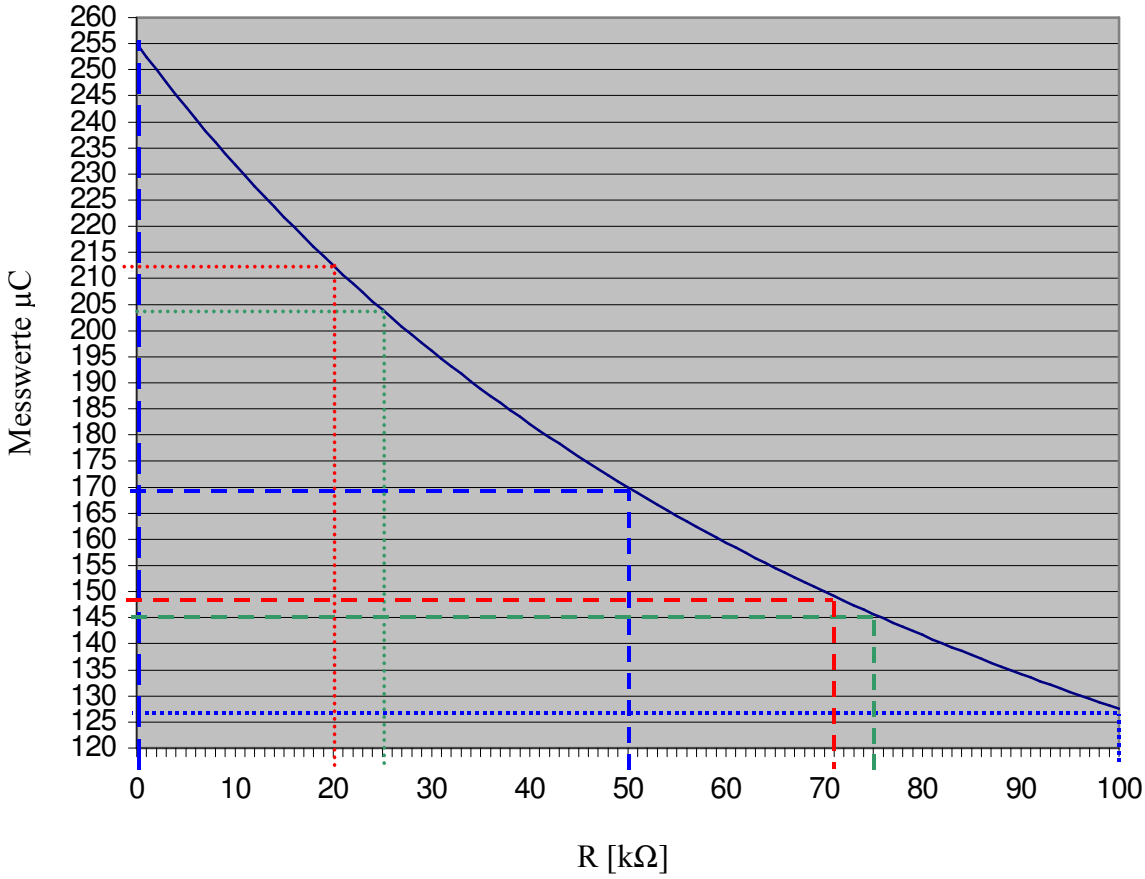


Abbildung 6.2

Das obige Diagramm zeigt die vom Microcontroller IC1 gemessenen Werte in Abhängigkeit vom Joystick-Potentiometer im Idealfall (Joystick-Potentiometer und RN1 ohne Toleranz) an. Die blauen Linien geben die Mittelstellung und die beiden Vollausschläge an. Die grünen Linien geben die berechneten Auslösepunkte, bei jeweils der Hälfte des Vollausschlags, wieder. Die roten Linien markieren die Punkte, die man erhält, wenn man einfach die Hälfte der Messwerte zwischen Mitte und Minimum, bzw. Maximum als Auslösepunkte gewählt hätte. Die Abweichungen zwischen den roten und den grünen Linien und das dadurch entstehende unsymmetrische Verhalten ist dabei klar zu sehen.

In den Zeilen 1231 bis 1246 steht die Funktion „joypress“, die erkennt, ob ein Joystick-Knopf gedrückt oder eine Joystick-Achse über den Auslösepunkt hinaus bewegt wurde.

In den Zeilen 1250 bis 1486 steht die Funktion „program_keyset“, mit der man eine Tasten-Sequenz auf eine Taste der 16er-Tastatur oder eine Joystick-Funktion programmieren kann. Zuerst werden die Variablen initialisiert, die Verbindung zwischen Tastatur und Computer getrennt, der Interrupt zur Abfrage der Tastatur eingeschaltet und der jeweilige Modus (Keyset, Joyset oder Passwordset) und die noch zur Verfügung stehenden Bytes, für die Tasten-Sequenz, angezeigt. Nun kann man über die Tastatur eine Tasten-Sequenz eingeben. Um den Speicher nicht zu schnell zu füllen, ist die Wiederholfunktion der Tastatur außer Kraft gesetzt, in dem geschaut wird, ob der aktuelle Make-Code gleich dem vorigen Make-Code entspricht und dieser dann verworfen wird. Hat man die Tasten-Sequenz komplett eingegeben, so kann man sie in den Modi „Keyset“ und „Passwordset“ durch Drücken einer Taste 0 bis 9 auf der 16er-Tastatur speichern. Im Modus „Joyset“ muss man die entsprechende Joystick-Funktion betätigen unter der die Tasten-Sequenz gespeichert werden soll. Die eigentliche Speicherung geschieht dabei erst nach Betätigen der OK-Taste. Solange hat man noch Gelegenheit die Joystick-Funktion, die auf der LCD-Anzeige angezeigt wird, zu ändern. Die Eingabe kann jederzeit durch Drücken der Cancel-Taste abgebrochen werden. Die Speicherung wird mit der Meldung „Data saved!“ bestätigt. Zuletzt wird noch der Interrupt zur Abfrage der Tastatur abgeschaltet und die Verbindung zwischen Tastatur und Computer wieder hergestellt.

In den Zeilen 1489 bis 1537 steht die Funktion „setup_toggle“, die im Setup-Menü zum nächsten Setup-Eintrag wechselt.

In den Zeilen 1540 bis 1604 steht die Funktion „setup“, die das Setup-Menü regelt. Bei Betätigen der Setup-Taste wird über die Funktion „setup_toogle“ (siehe oben) zum nächsten Setup-Eintrag gewechselt. Bei Betätigen der OK-Taste wird der gewählte Setup-Eintrag aufgerufen und mit der Cancel-Taste wird das Setup-Menü wieder verlassen.

In den Zeilen 1607 bis 1642 steht die Interrupt-Routine „SIGNAL (SIG_INTERRUPT0)“, die für die Abfrage der Computer-Tastatur zuständig ist. Dazu wird bei jeder fallenden Flanke des Clock-Signals über den Zähler „keyat.bitcount“ geschaut, ob es sich bei den, an der Data-Leitung anliegenden Daten, um das eigentlich zu übertragende Byte handelt (Start-, Paritäts- und Stop-Bit sind für die Speicherung uninteressant), welches in der Variablen „keyat.data“ gespeichert wird. Die Funktion Interrupt-Routine wird immer abwechselnd zuerst auf fallender und dann auf steigender Flanke ausgeführt. Bei der steigenden Flanke wird geschaut, ob schon alle Bits übertragen worden sind und dann ggf. der Zähler „keyat.bitcount“ wieder auf seinen Startwert von 11 gesetzt.

In den Zeilen 1646 bis 1673 steht die Interrupt-Routine „SIGNAL (SIG_OVERFLOW0)“, die für die Steuerung der LCD-Hintergrundbeleuchtung und des –Kontrastes zuständig ist. Sie wird bei jedem Überlauf von Timer0 alle 16µs aufgerufen (Funktion „uc_init“ – siehe oben). Die eigentliche Steuerung der LCD-Hintergrundbeleuchtung und des –Kontrastes geschieht dabei über das Tastverhältnis eines PWM-ähnlichen Signals (ähnlich, da die Frequenz nicht fest ist).

In den Zeilen 1676 bis 1825 steht die Interrupt-Routine „INTERRUPT (SIG_OVERFLOW2)“, die für die Abfrage und Entprellung der 16er-Tastatur und der Joystick-Knöpfe zuständig ist. Sie wird bei jedem Überlauf von Timer2 alle 128µs aufgerufen (Funktion „uc_init“ - siehe oben). Die eigentliche Abfrage wird dabei jedoch nur ca. alle 20ms ausgeführt, wenn „timer2count“ gleich „debounce“ ist. Die Abfrage der 16er-Tastatur geschieht nach folgender Vorgehensweise:

- Abfrage der 16er-Tastatur mittels der Funktion „key16_scan“.
- Ist der aktuelle Tasten-Code gleich dem Tasten-Code von vor 20ms?
    - Wenn ja, dann betrachte Taste als entprellt an.
    - Erkennung, ob Taste schon bereits länger gedrückt ist.

Die Abfrage und Entprellung des Joysticks geschieht dabei analog zur Vorgehensweise bei der 16er-Tastatur.

In den Zeilen 1828 bis 1847 steht die Interrupt-Routine „SIGNAL (SIG_ADC)“, die für die Abrage der Joystick-Achsen zuständig ist. Sie wird automatisch aufgerufen, wenn eine AD-Wandlung einer Joystick-Achse beendet worden ist. Der gemessene Wert wird gespeichert und die Messung der nächsten Joystick-Achse wird gestartet.

In den Zeilen 1850 bis 1986 steht das Hauptprogramm „main“. Hier werden zunächst Komponenten der Schaltung, wie LCD-Anzeige, Tastatur- und Computer-Anschluss, 16er-Tastatur, Joystick-Anschluss, externes EEPROM IC2 und der Microcontroller IC1 initialisiert. Dann werden die Werte für LCD-Hintergrundbeleuchtung und –Kontrast und Joystick-Kalibration aus dem externen EEPROM IC2 gelesen. Nach der Initialisierung der Variablen für den eingestellten Modus und der dazugehörigen Sätze wird noch eine 1s lange Meldung mit Gerätname und Softwareversion ausgegeben. Alles nachfolgende befindet sich in einer Endlosschleife. Zuerst wird geschaut, ob die Keyset-, Joyset-, Passwordset- oder Setup-Taste gedrückt wurde und ggf. die entsprechende Funktion aufgerufen. Dann wird geschaut, ob im Keyset- oder Passworset-Modus eine der Tasten 0 bis 9, bzw. im Joyset-Modus eine von 12 Joystick-Funktionen betätigt wurden und dann ggf. die entsprechende Tastatur-Sequenz an den Computer gesendet.

7. Mögliche Erweiterungen

Bei einer Weiterentwicklung des Gerätes sind noch einige mögliche Erweiterungen denkbar, die in der aktuellen Version nicht berücksichtigt wurden, meist aus Zeitmangel, aber auch um diese Projektarbeit nicht unnötig aufzublähen. Die Programm-Version ist deshalb auch erst mal auf 0.89 und damit kleiner eins festgelegt worden. Mögliche Erweiterungen wären wie folgt:

- Die Verwendung einer kleineren 16er-Tastatur mit entsprechenden Aufdrucken für die Steuertasten.
- Die Verwendung eines kleineren Gehäuses.
- Durch ein größeres oder weiteres externes EEPROM stünden für die einzelnen Tasten-Sequenzen mehr Speicher zur Verfügung.
- Alle bisher im externen EEPROM gespeicherten Daten, die keine Tasten-Sequenzen sind, können auch im internen EEPROM gespeichert werden, um mehr Speicher für die Tasten-Sequenzen zur Verfügung zu haben.
- Die erweiterten Tasten-Codes könnten so gespeichert werden, dass sie nur noch so viel Speicherplatz wie normale Tasten-Codes beanspruchen.

- Die PIN und die Tasten-Sequenzen sollten verschlüsselt im Speicher abgelegt werden.
- Nach dreimaliger Eingabe der falschen PIN sollte eine Weiterbedienung nur noch durch Eingabe einer „Master-PIN“ mit mehr Stellen möglich sein. Bei dreimaliger falscher Eingabe dieser „Master-PIN“ könnte das Gerät ganz gesperrt werden.
- Die PIN könnte durch ein Passwort ersetzt werden.
- Für die einzelnen Sätze könnten Bezeichnungen (z.B. Word oder Spiel XY) gespeichert werden.
- Es könnten sich mehrere Sprachen, für die Meldungen auf der LCD-Anzeige, auswählen lassen.
- Die Werte für die Verzögerung und die Wiederholrate beim Start vom Computer lesen oder auch per Setup einstellbar machen.
- Es könnten sich Tasten-Sequenzen auch ohne eine Tastatur erzeugen lassen, um z.B. Windows-Multimedia-Tasten (z.B. Taste für E-Mail-Programm) auch ohne eine Tastatur, die diese besitzt, zu programmieren.
- Der Joystick könnte auch in den Modi „Keyset“ oder „Passwordset“ funktionieren. Der spezielle Modus „Joyset“ könnte dann eventuell entfallen.
- Durch die Erkennung einer schrägen Stellung (z.B. links-oben) des Joystick-Hebels könnte man abwechselnd zwei Tasten-Sequenzen senden.
- Der Auslösepunkt für die Betätigung eines Joystick-Hebels könnte sich einzustellen lassen.
- Die Einführung einer Hysterese für die Bewegung eines Joystick-Hebels. Der Auslösepunkt, der eine Bewegung von der Mittelstellung zum Vollausschlag hin markiert, wäre dann ein anderer, als bei einer Bewegung vom Vollausschlag zurück zur Mittelstellung hin. Auch diese Werte könnten einstellbar sein.
- Joystick-Dauerfeuer wird zur Zeit nicht unterstützt. Es kommt dabei zwischendurch immer wieder zu Aussetzern bei der Wiedergabe.
- Wenn eine Taste der Tastatur gedrückt ist, wenn man in den Programmiermodus geht, dann kann es passieren, dass nur ein Teil des Tasten-Codes erkannt wird.
- Das Tastatur-Protokoll ist bisher erst auf das nötigste umgesetzt. Im normalen Betrieb fällt dies aber nicht auf. Probleme könnte es geben, wenn während der Widergabe einer Tastatur-Sequenz an den Computer dieser die Übertragung unterbrechen will. Dies ist bisher nicht implementiert und kommt wohl auch so gut wie gar nicht vor. Ein Fall, in dem es jedoch vorkommen könnte wäre zum Beispiel der Apple II-Emulator „Oasis“, der die Scroll-Lock-LED als virtuelle Laufwerks-LED benutzt und dazu ggf. eine gerade laufende Übertragung zum Computer hin unterbrechen müsste.
- Die Einstellung von LCD-Hintergrundbeleuchtung und –Kontrast kann noch optimiert werden, da in einem Einstell-Bereich größere Änderungen sichtbar sind, als in einem anderen Bereich.
- Nach Umsetzung der obigen Programmänderungen kann, falls das Programm dann immer noch weniger als 16KB Speicher beansprucht, der Microcontroller IC1 durch einen ATmega16 ausgetauscht werden, der weitestgehend kompatibel zum hier verwendeten ATmega32 ist und dabei weniger kostet. Es wäre wahrscheinlich nur eine neue Übersetzung des Programms mit dem C-Compiler notwendig.
- Wenn man das Gerät auf den Markt bringen will, dann braucht es eine CE-Zulassung. Eine Prüfung diesbezüglich ist also vorher durchzuführen.

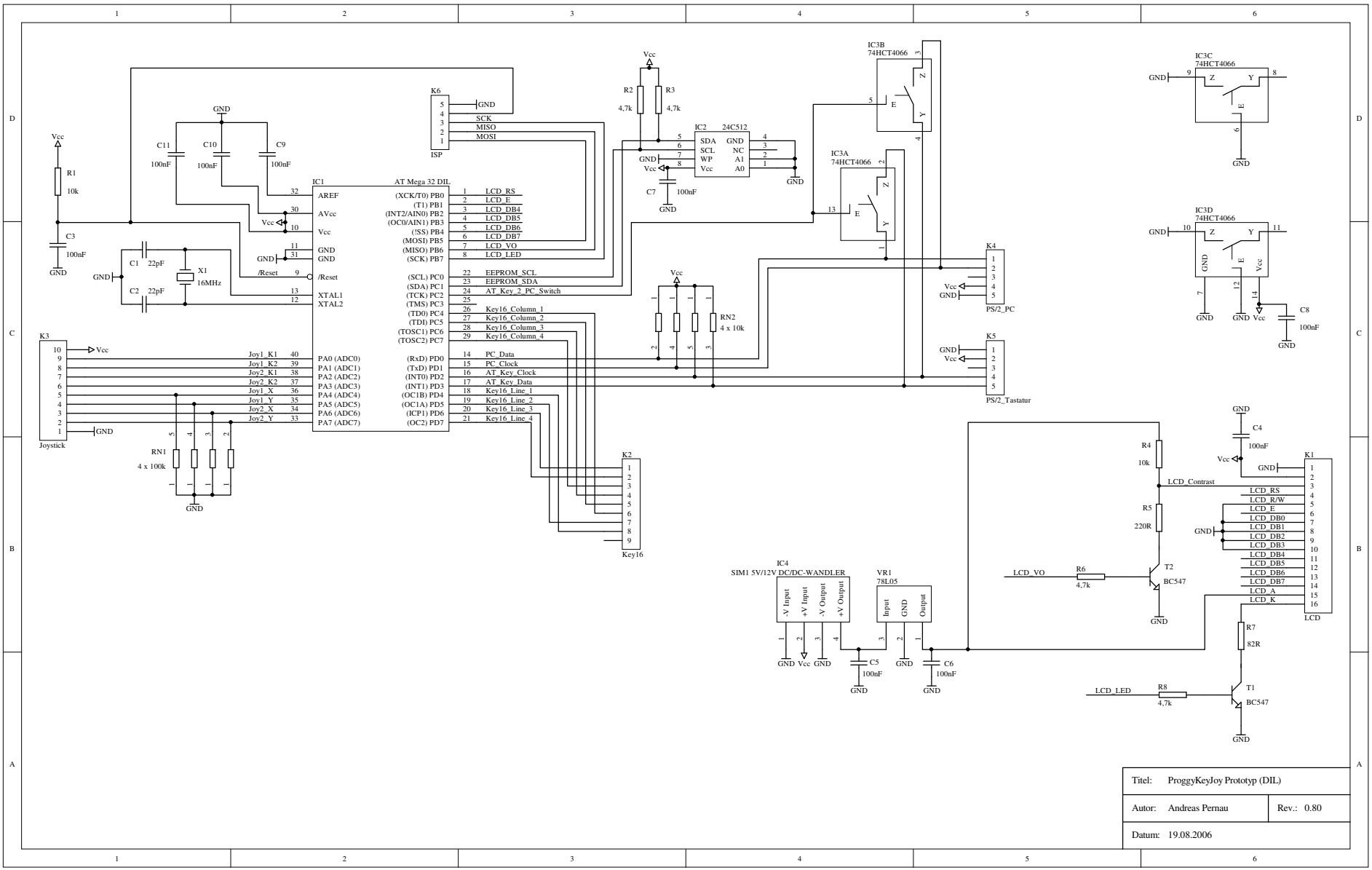
8. Hilfsmittel

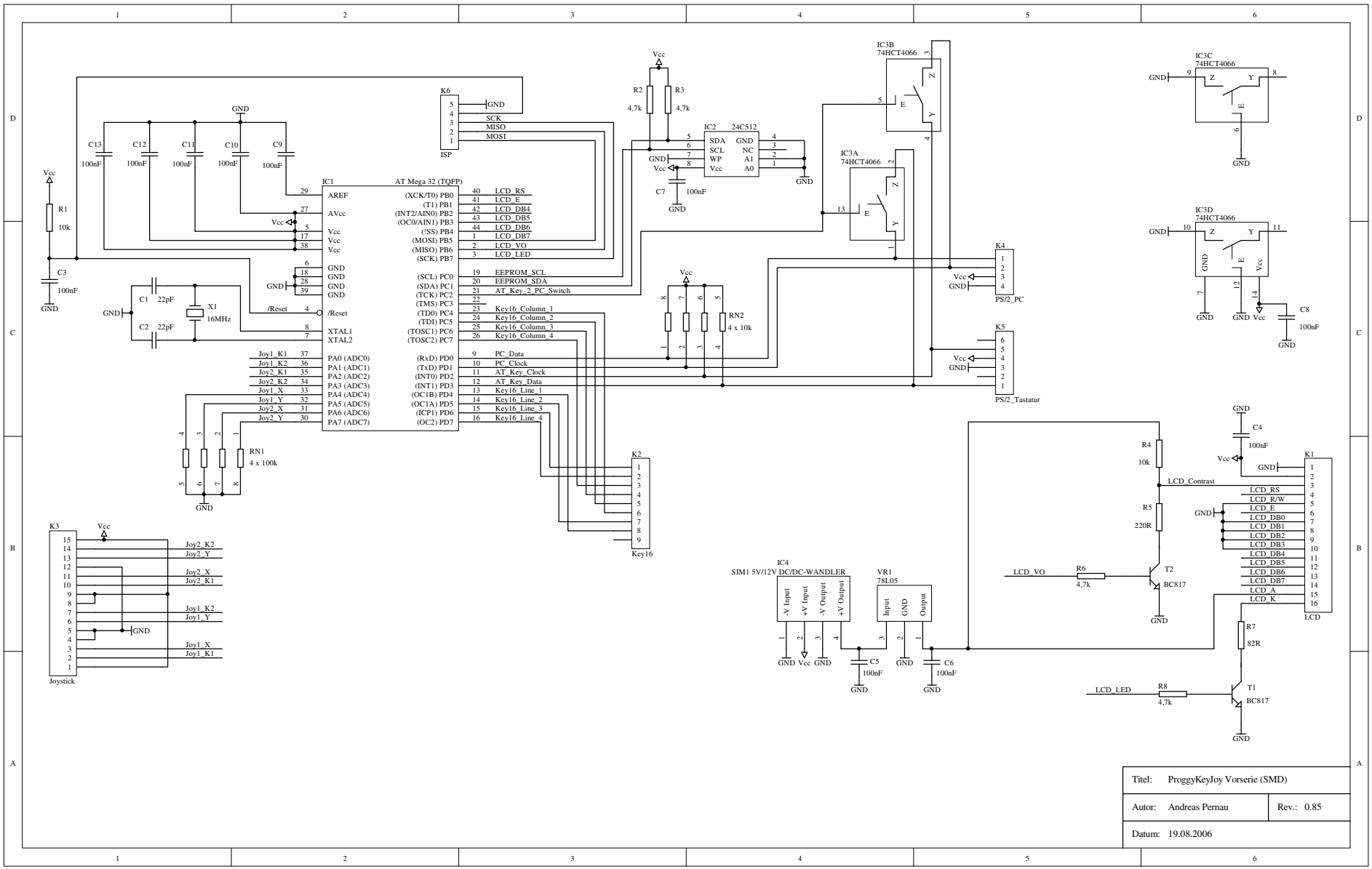
Als Hilfsmittel sind neben Werkzeug zur Fertigung des Prototyps besonders folgende Programme zu nennen:

- MS Word 2000 : Schreiben von Texten (z.B. diese Projektarbeit)
- Highlight 2.4.6 : Konvertieren von Programm-Quelltext in Word-kompatibles Format und Hinzufügen von Zeilennummern
- MS Excel 2000 : Erstellen von Tabellen und Diagrammen
- Programmers Notepad 2.0.5.48 : Schreiben des C-Programms und starten des AVR-GCC C-Compilers
- AVR-GCC 3.4.3 : C-Compiler
- PonyProg2000 2.06f Beta : Dient zum Überspielen des Programms auf das Gerät
- Protel 99 SE SP6 : Erstellen von Schaltplänen und Platinen-Layouts
- Lochmaster 2.0 : Erstellen von Platinen-Layouts für Lochrasterkarten
- Firefox 1.5.0.3 : Internet-Browser für Informationsbeschaffung

9. Quellenverzeichnis

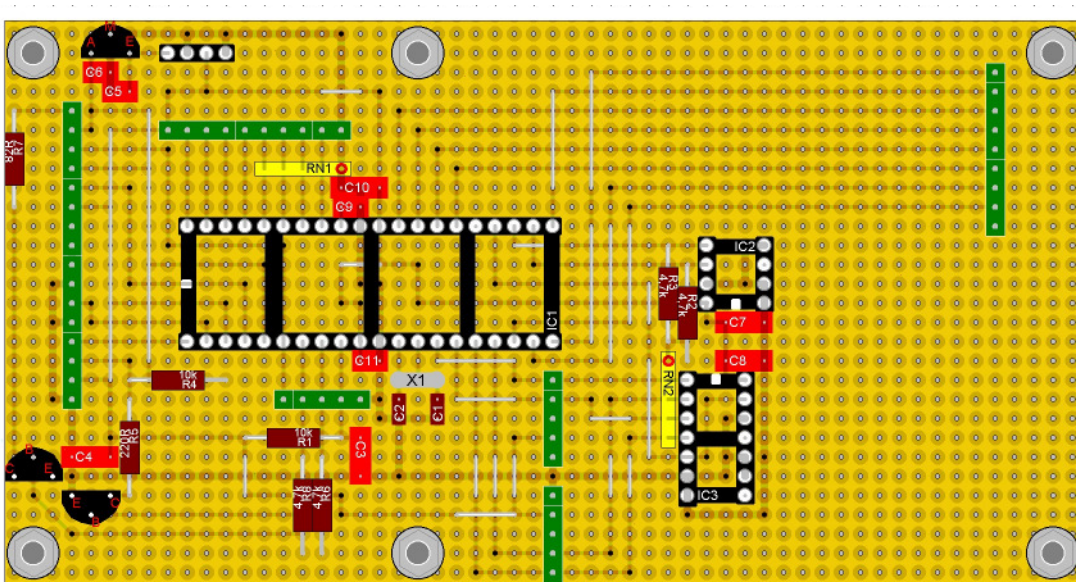
- C-Grundlagen:
  - Buch: C/C++ Kompendium; Autor: Dirk Louis; Verlag: Markt & Technik; ISBN: 3-8272-6812-5
- Tastatur-Schnittstelle:
  - Buch: Mensch-Maschine-Schnittstellen: Leitfaden für Design und Schaltungstechnik; Verlag: Springer; ISBN: 3-540-63601-3
  - Internet: http://www.atmel.com/dyn/resources/prod_documents/DOC1235.PDF
  - Internet: http://www.marjorie.de/ps2/start.htm
  - Internet: http://de.wikipedia.org/wiki/PS2-Schnittstelle
- Joystick-Schnittstelle:
  - Buch: PC-Schnittstellen Angewandt; Autor: Burkhard Kainka; Verlag: elektor; ISBN: 3-928051-42-3
- I²C-Schnittstelle:
  - Internet: http://jump.to/fleury


IC1
AT Mega 32 DIL
AREF
AVcc
Vcc
GND
GND
/Reset
XTAL1
XTAL2
(XCK/T0) PB0
(T1) PB1
(INT2/AIN0) PB2
(OC0/AIN1) PB3
(!SS) PB4
(MOSI) PB5
(MISO) PB6
(SCK) PB7
(SCL) PC0
(SDA) PC1
(TCK) PC2
(TMS) PC3
(TDO) PC4
(TDI) PC5
(TOSC1) PC6
(TOSC2) PC7
(RxD) PD0
(TxD) PD1
(INT0) PD2
(INT1) PD3
(OC1B) PD4
(OC1A) PD5
(ICP1) PD6
(OC2) PD7
LCD_RS
LCD_E
LCD_DB4
LCD_DB5
LCD_DB6
LCD_DB7
LCD_VO
LCD_LED
EEPROM_SCL
EEPROM_SDA
AT_Key_2_PC_Switch
Key16_Column_1
Key16_Column_2
Key16_Column_3
Key16_Column_4
PC_Data
PC_Clock
AT_Key_Clock
AT_Key_Data
Key16_Line_1
Key16_Line_2
Key16_Line_3
Key16_Line_4
Joy1_K1
Joy1_K2
Joy2_K1
Joy2_K2
Joy1_X
Joy1_Y
Joy2_X
Joy2_Y
K6
ISP
SCK
MISO
MOSI
IC2
24C512
SDA
SCL
WP
Vcc
GND
NC
A1
A0
IC3A
74HCT4066
IC3B
74HCT4066
IC3C
74HCT4066
IC3D
74HCT4066
K4
PS/2_PC
K5
PS/2_Tastatur
K3
Joystick
K2
Key16
RN1
4 x 100k
RN2
4 x 10k
K1
LCD
LCD_Contrast
LCD_R/W
LCD_A
LCD_K
IC4
SIM1 5V/12V DC/DC-WANDLER
-V Input
+V Input
-V Output
+V Output
VR1
78L05
Input
GND
Output
T1
T2
BC547
R1
10k
R2
4,7k
R3
4,7k
R4
10k
R5
220R
R6
4,7k
R7
82R
R8
4,7k
C1
22pF
C2
22pF
C3
100nF
C4
100nF
C5
100nF
C6
100nF
C7
100nF
C8
100nF
C9
100nF
C10
100nF
C11
100nF
X1
16MHz
Titel: ProggyKeyJoy Prototyp (DIL)
Autor: Andreas Pernau
Rev.: 0.80
Datum: 19.08.2006

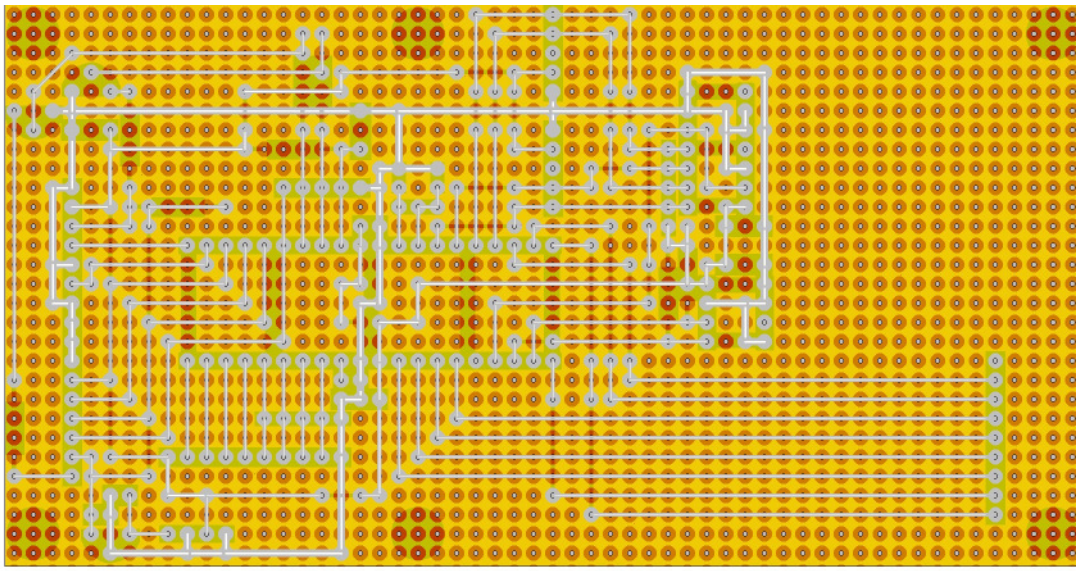


Titel:	ProggyKeyJoy Vorserie (SMD)	
Autor:	Andreas Pernau	Rev.: 0.85
Datum:	19.08.2006	

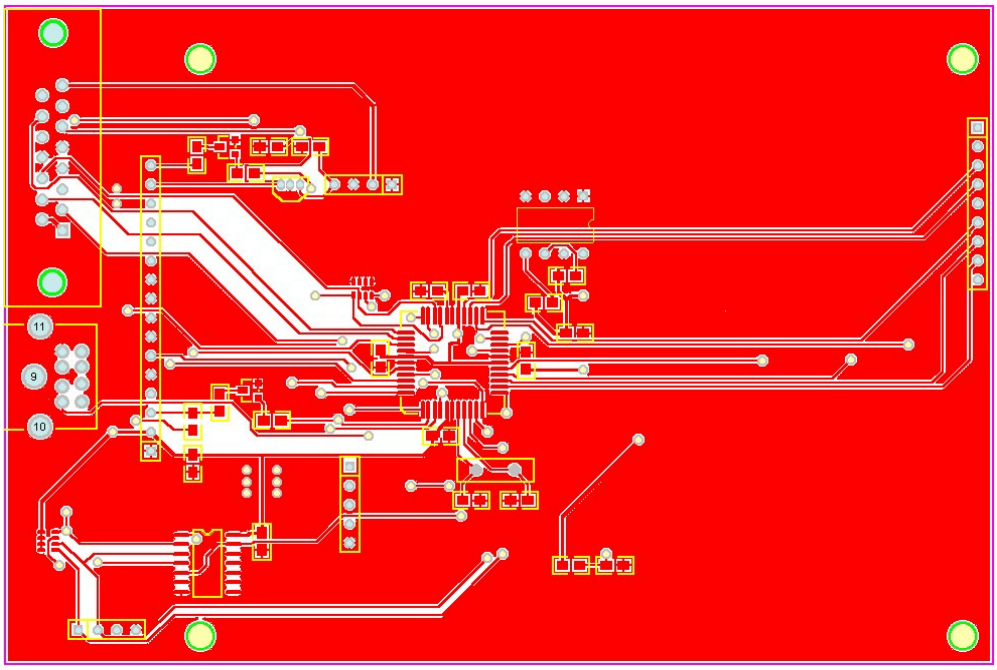
Bauteilseite:



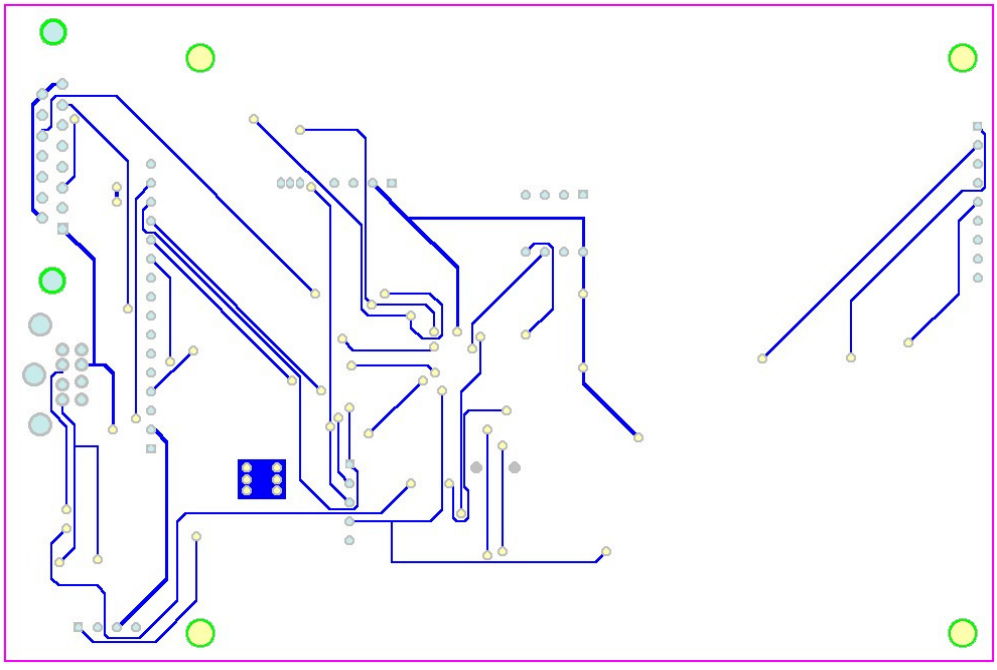
Lötseite:



Bauteilseite:



Unterseite:



Anzahl	*Bezeichnung*	*Lieferant*	*Bestellnummer*	*Preis/St.*	*Preis/ges.*
	Platine				
1	Lochrasterplatine 160x100	Reichelt	H25PR160	1,65 €	1,65 €
1	IC-Sockel 40-pol.	Reichelt	GS 40P	0,45 €	0,45 €
1	IC-Sockel 14-pol.	Reichelt	GS 14P	0,17 €	0,17 €
1	IC-Sockel 8-pol.	Reichelt	GS 8P	0,09 €	0,09 €
1	Buchsenleiste RM 2,54 4-pol.	Reichelt	SPL 20	0,05 €	0,05 €
1	Stiftleiste RM 2,54 16-pol.	Reichelt	SL 1X40G 2,54	0,07 €	0,07 €
1	Folienbuchse 9-pol.	Conrad	732003	0,99 €	0,99 €
1	Platinenstecker 10-pol.	Reichelt	PS 25/10G BR	1,10 €	1,10 €
3	Platinenstecker 5-pol.	Reichelt	PS 25/5G BR	0,61 €	1,83 €
1	ATmega32 DIL40	Reichelt	ATMega 32-16 DIP	3,50 €	3,50 €
1	24C512 DIL8	Reichelt	ST24C512 BN6	2,55 €	2,55 €
1	74HCT4066 DIL14	Reichelt	74HCT4066	0,27 €	0,27 €
1	DC/DC-Wandler 1W 5V/12V	Reichelt	SIM1-0512 SIL4	4,95 €	4,95 €
1	Spannungsregler 78L05	Reichelt	µA 78L05	0,12 €	0,12 €
2	Transistor BC547C	Reichelt	BC 547C	0,03 €	0,06 €
5	Kondensator 100nF RM5,08	Reichelt	MKS-2 100n	0,07 €	0,35 €
4	Kondensator 100nF RM2,54	Reichelt	MKS-02 100n	0,15 €	0,60 €
2	Kondensator 22pF	Reichelt	Kerko 22p	0,04 €	0,08 €
1	Quarz 16MHz	Reichelt	16-HC18	0,44 €	0,44 €
1	Widerstandsnetzwerk SIL5-4 100k	Reichelt	SIL5-4 100k	0,08 €	0,08 €
1	Widerstandsnetzwerk SIL5-4 10k	Reichelt	SIL5-4 10k	0,08 €	0,08 €
4	Widerstand 4,7k	Reichelt	1/4W 4,7k	0,02 €	0,08 €
1	Widerstand 82	Reichelt	1/4W 82	0,02 €	0,02 €
1	Widerstand 10k	Reichelt	1/4W 10k	0,02 €	0,02 €
1	Widerstand 220	Reichelt	1/4W 220	0,02 €	0,02 €
				Summe:	19,61 €
	LCD-Anzeige				
1	LCD-Anzeige 2x16 grün	Reichelt	LCD 162C LED	7,50 €	7,50 €
1	Stiftleiste RM 2,54 16-pol.	Reichelt	SL 1X40G 2,54	0,07 €	0,07 €
				Summe:	7,57 €
	Anschlüsse und Verkabelung				
1	Sub-D-Buchse 15-pol. Lötanschluss	Reichelt	D-SUB BU 15	0,10 €	0,10 €
1	Mini-DIN-Buchse 6-pol.	Conrad	734209	1,75 €	1,75 €
0,5	Tastaturkabel 6-pol. St/St 5m (2,5m)	Reichelt	AK 3235	0,98 €	0,49 €
1	Knickschutz	Reichelt	BOPLA BFK7	0,50 €	0,50 €
1	Mutter für Knickschutz	Reichelt	BOPLA GM7	0,12 €	0,12 €
23	Schrumpfschlauch 1,6/0,8 8mm	Reichelt	SDB 1,6 SW	0,00 €	0,05 €
2	Pfostensteckverbinder 34-pol.	Reichelt	PFL 34	0,12 €	0,24 €
1	Flachbandkabel 34-pol. 0,1m	Reichelt	AWG 28-34G	0,05 €	0,05 €
				Summe:	3,30 €
	16er-Tastatur				
1	16er-Telefon-Tastatur	Conrad	470082	0,46 €	0,46 €
				Summe:	0,46 €
	Gehäuse und Montage				
1	Gehäuse	Reichelt	TEKO 104	10,15 €	10,15 €
1	Rahmen für LCD-Anzeige	Reichelt	LCD Front 2	6,20 €	6,20 €
4	Distanzrolle 8mm	Reichelt	DK 8mm	0,05 €	0,20 €
6	Abstandsbolzen Innen/Außen M3x10	Reichelt	DA 10mm	0,09 €	0,54 €
4	Zylinderkopfschraube M3x8	Reichelt	SZK M3x8-200	0,01 €	0,03 €

Anzahl	*Bezeichnung*	*Lieferant*	*Bestellnummer*	*Preis/St.*	*Preis/ges.*
6	Zylinderkopfschraube M3x10	Reichelt	SZK M3x10-200	0,01 €	0,05 €
4	Senkkopfschraube M3x10	Reichelt	SSK M3x10-200	0,01 €	0,03 €
4	Senkkopfschraube M2,5x16	Reichelt	SSK M2,5x16-200	0,01 €	0,04 €
2	Zylinderkopfschraube M2x10			0,02 €	0,03 €
16	Mutter M3	Reichelt	SK M3-100	0,01 €	0,14 €
4	Mutter M2,5	Reichelt	SK M2,5-100	0,01 €	0,06 €
2	Mutter M2	Conrad	815608	0,01 €	0,02 €
10	U-Scheibe für M3	Reichelt	SKU 3,2-100	0,01 €	0,09 €
12	U-Scheibe für M3 Nylon			0,10 €	1,20 €
2	U-Scheibe für M2			0,01 €	0,02 €
				Summe:	18,80 €
				Total:	49,75 €

Anzahl	*Bezeichnung*	*Lieferant*	*Bestellnummer*	*Preis/St.*	*Preis/ges.*
	Platine				
1	Platine (*)	Ilfa		10,00 €	10,00 €
1	Sub-D-Buchse 15-pol. für Platine	Reichelt	D-SUB BU 15EU	0,29 €	0,29 €
1	Mini-DIN-Buchse 6-pol. für Platine	Reichelt	K-DIO M06	0,56 €	0,56 €
1	Folienbuchse 9-pol.	Conrad	732003	0,99 €	0,99 €
1	Platinenstecker 5-pol.	Reichelt	PS 25/5G BR	0,61 €	0,61 €
1	ATmega32 TQFP	Reichelt	ATMega 32-16 TQ	2,55 €	2,55 €
1	24C512 DIL8	Reichelt	ST24C512 BN6	2,55 €	2,55 €
1	74HCT4066 SMD	Conrad	147699	0,33 €	0,33 €
1	DC/DC-Wandler 1W 5V/12V	Reichelt	SIM1-0512 SIL4	4,95 €	4,95 €
1	Spannungsregler 78L05	Reichelt	µA 78L05	0,12 €	0,12 €
2	Transistor BC817-40	Reichelt	BC 817-40 SMD	0,05 €	0,10 €
11	Kondensator 100nF SMD 0805	Reichelt	X7R-G0805 100N	0,05 €	0,55 €
2	Kondensator 22pF SMD 0805	Reichelt	NPO-G0805 22P	0,05 €	0,10 €
1	Quarz 16MHz	Reichelt	16-HC18	0,44 €	0,44 €
1	Widerstandsnetzwerk SMD 100k	Reichelt	BCN16 100k	0,02 €	0,02 €
1	Widerstandsnetzwerk SMD 10k	Reichelt	BCN16 10k	0,02 €	0,02 €
4	Widerstand 4,7k SMD 0805	Reichelt	SMD-0805 4,7k	0,01 €	0,04 €
1	Widerstand 82 SMD 0805	Reichelt	SMD-0805 82	0,01 €	0,01 €
1	Widerstand 10k SMD 0805	Reichelt	SMD-0805 10k	0,01 €	0,01 €
1	Widerstand 220 SMD 0805	Reichelt	SMD-0805 220	0,01 €	0,01 €
				Summe:	24,25 €
	LCD-Anzeige				
1	LCD-Anzeige 2x16 grün	Reichelt	LCD 162C LED	7,50 €	7,50 €
1	Stiftleiste RM 2,54 16-pol. Lang	Conrad	739499	0,68 €	0,68 €
				Summe:	8,18 €
	Anschlüsse und Verkabelung				
0,5	Tastaturkabel 6-pol. St/St 5m (2,5m)	Reichelt	AK 3235	0,98 €	0,49 €
1	Knickschutz	Reichelt	BOPLA BFK7	0,50 €	0,50 €
1	Mutter für Knickschutz	Reichelt	BOPLA GM7	0,12 €	0,12 €
				Summe:	1,11 €
	16er-Tastatur				
1	16er-Tastatur (*)			5,00 €	5,00 €
				Summe:	5,00 €
	Gehäuse und Montage				
1	Gehäuse (*)			5,00 €	5,00 €
1	Rahmen für LCD-Anzeige	Reichelt	LCD Front 2	6,20 €	6,20 €
1	Montagematerial (*)			0,20 €	0,20 €
				Summe:	11,40 €
				Total:	49,94 €

(*) Diese Preise sind nur geschätzt. Der Gesamtpreis sollte sich bei einer Serienfertigung durch die größere Stückzahl sicher noch unter 1/3 drücken lassen.

```
 1  #include <avr/io.h>
 2  #include <avr/delay.h>
 3  #include <avr/interrupt.h>
 4  #include <avr/signal.h>
 5  #include <compat/twi.h>
 6  #include <stdio.h>
 7  #include <string.h>
 8
 9  #define LCD_DDR   DDRB
10  #define LCD_PORT  PORTB
11  #define LCD_RS    0
12  #define LCD_E     1
13  #define LCD_DB4   2
14  #define LCD_DB5   3
15  #define LCD_DB6   4
16  #define LCD_DB7   5
17  #define LCD_LED   6
18  #define LCD_VO    7
19
20  #define KEY16_LINE_DDR      DDRD
21  #define KEY16_LINE_PORT     PORTD
22  #define KEY16_LINE_IN       PIND
23  #define KEY16_COLUMN_DDR    DDRC
24  #define KEY16_COLUMN_PORT   PORTC
25  #define KEY16_COLUMN_IN     PINC
26  #define KEY16_LINE_1        4
27  #define KEY16_LINE_2        5
28  #define KEY16_LINE_3        6
29  #define KEY16_LINE_4        7
30  #define KEY16_COLUMN_1      4
31  #define KEY16_COLUMN_2      5
32  #define KEY16_COLUMN_3      6
33  #define KEY16_COLUMN_4      7
34
35  #define PS2_DDR       DDRD
36  #define PS2_PORT      PORTD
37  #define PS2_IN        PIND
38  #define PC_DATA       0
39  #define PC_CLOCK      1
40  #define AT_KEY_DATA   3
41  #define AT_KEY_CLOCK  2
42
43  #define AT_KEY_2_PC_SWITCH_DDR    DDRC
44  #define AT_KEY_2_PC_SWITCH_PORT   PORTC
45  #define AT_KEY_2_PC_SWITCH        2
46
47  #define JOY_DDR       DDRA
48  #define JOY_PORT      PORTA
49  #define JOY_IN        PINA
50  #define JOY_BUTTON_0  0
51  #define JOY_BUTTON_1  1
52  #define JOY_BUTTON_2  2
53  #define JOY_BUTTON_3  3
54  #define JOY_POTI_0    4
55  #define JOY_POTI_1    5
56  #define JOY_POTI_2    6
57  #define JOY_POTI_3    7
58
59  #define EXT_EEPROM_DDR  DDRC
60  #define EXT_EEPROM_SCL  0
61  #define EXT_EEPROM_SDA  1
62
```

```
 63  #define KEY_KEYSET    13
 64  #define KEY_JOYSET    14
 65  #define KEY_PASSWORD  15
 66  #define KEY_SETUP     16
 67  #define KEY_OK        4
 68  #define KEY_CANCEL    12
 69
 70  #define TIMER2PRESCALER     8
 71  #define DEBOUNCE_TIME       20E-3
 72  #define REPEAT_START_TIME   500E-3
 73  #define REPEAT_NEXT_TIME    200E-3
 74  #define MS2WAIT             500
 75
 76  #define SCL_CLOCK           100000
 77  #define EXT_EEPROM_READ     1
 78  #define EXT_EEPROM_WRITE    0
 79  #define EXT_EEPROM_ADDRESS  0xA0
 80
 81  #define NO        0
 82  #define OK        1
 83  #define CANCEL    2
 84
 85  #define SET_KEY       0
 86  #define SET_JOY       1
 87  #define SET_PASSWORD  2
 88
 89  #define MAX_KEY_DATA        201
 90  #define JOYSET_ADDRESS      0x9D08
 91  #define PASSWORD_ADDRESS    0x4E84
 92  #define PIN_ADDRESS         0xFB40
 93  #define BACKLIGHT_ADDRESS   0xFB46
 94  #define CONTRAST_ADDRESS    0xFB48
 95  #define JOYCAL_ADDRESS      0xFB4A
 96
 97  #define SETUP_PROGRAM     0
 98  #define SETUP_JOYSTICK    1
 99  #define SETUP_PIN         2
100  #define SETUP_BACKLIGHT   3
101  #define SETUP_CONTRAST    4
102  #define LAST_SETUP_ITEM   4
103
104  struct lcdview
105  {
106    unsigned char count,trig;
107  };
108
109  struct lcdelement
110  {
111    struct lcdview led,vo;
112  };
113
114  struct at_keyboard
115  {
116    unsigned char clockedge,bitcount,finish,data;
117  };
118
119  struct a16_keyboard
120  {
121    unsigned char key,oldkey,prevkey,longpress,repeat;
122  };
123
124  struct joystick_calibration
125  {
126    unsigned char trig_mid_max,trig_mid_min;
127  };
```

```
128
129   struct joystick
130   {
131     unsigned char button[4],oldbuttons,prevbuttons,btn_repeat,btn_longpress,
132                   poti[4],potiaxis,oldpotiaxis,prevpotiaxis,
133                   poti_repeat,poti_longpress,potipointer;
134     struct joystick_calibration cal[4];
135   };
136
137   struct buffertype
138   {
139     unsigned char pointer,data[MAX_KEY_DATA];
140   };
141
142   volatile struct lcdelement lcd;
143   volatile struct at_keyboard keyat;
144   volatile struct a16_keyboard key16;
145   volatile struct joystick joy;
146   volatile struct buffertype keybuffer;
147
148   volatile unsigned char timer0count,timer2count,debounce,
149                          repeat_start,repeat_next,*ledvo_trig,
150                          keyset,joyset,passwordset,set_mode;
151
152   char lcdstr[17];
153
154   signed char digit16[17]={-1,7,4,1,-1,8,5,2,0,9,6,3,-1,-1,-1,-1,-1};
155
156
157   /* wartet eine Zeit von ca. (us) µs */
158   void wait_us (unsigned int us)
159   {
160   unsigned int a;
161
162     us=(us*10)/15;
163     for (a=1;a<=us;a++)
164     {
165       _delay_us(1);
166     }
167
168   }  // wait_us
169
170   /* wartet eine Zeit von ca. (ms) ms */
171   void wait_ms (unsigned int ms)
172   {
173   unsigned int a;
174
175     for (a=1;a<=ms;a++)
176     {
177       _delay_ms(1);
178     }
179
180   }  // wait_ms
181
182   /* sendet 4-Bit Daten (rs=1) oder Befehle (rs=0) an die LCD-Anzeige */
183   void lcd_hex_4 (unsigned char z, unsigned char rs)
184   {
185
186     if (rs)
187     {
188       rs=(1<<LCD_RS);
189     }
190     LCD_PORT=(z<<LCD_DB4) | (1<<LCD_E) | (rs);
191     _delay_us(1);
192     LCD_PORT=LCD_PORT & ~(1<<LCD_E);
```

```
193     _delay_us(1);
194     LCD_PORT=LCD_PORT | (1<<LCD_E);
195
196   }  // lcd_hex_4
197
198   /* sendet 8-Bit Daten (rs=1) oder Befehle (rs=0) an die LCD-Anzeige */
199   void lcd_hex_8 (unsigned char z,unsigned char rs)
200   {
201
202     lcd_hex_4((z>>4) & 0x0F,rs);
203     lcd_hex_4(z & 0x0F,rs);
204
205   }  // lcd_hex_8
206
207   /* löscht die LCD-Anzeige */
208   void lcd_clear (void)
209   {
210
211     lcd_hex_8(0x01,0);
212     _delay_ms(2);
213
214   }  // lcd_clear
215
216   /* Initialisiert die LCD-Anzeige */
217   void lcd_init (void)
218   {
219
220     // E High setzen
221     LCD_PORT=(1<<LCD_E);
222
223     // DB4-DB7, E, RS, LED und VO als Ausgang setzen
224     LCD_DDR=(1<<LCD_DB4) | (1<<LCD_DB5) | (1<<LCD_DB6) | (1<<LCD_DB7) |
225             (1<<LCD_RS) | (1<<LCD_E) | (1<<LCD_LED) | (1<<LCD_VO);
226
227     wait_ms(40);
228     lcd_hex_4(0x3,0);
229
230     wait_ms(6);
231     lcd_hex_4(0x3,0);
232
233     wait_us(150);
234     lcd_hex_4(0x3,0);
235     wait_us(100);
236
237     // 4-Bit Interface setzen
238     lcd_hex_4(0x2,0);
239     wait_us(50);
240
241     // 2 Zeilen LCD-Anzeige und 5x7 Zeichensatz
242     lcd_hex_8(0x28,0);
243     wait_us(50);
244
245     // LCD-Anzeige an
246     lcd_hex_8(0x0C,0);
247     wait_us(50);
248
249     // LCD-Anzeige löschen
250     lcd_clear();
251
252     // Entry mode set
253     // Adresspointer inkrementieren und Displayinhalt nicht schieben
254     lcd_hex_8(0x06,0);
255     wait_us(50);
256
```

```
257     // Initialisierung der Timer-Werte für Hintergrundbeleuchtung und Kontrast
258     lcd.led.count=0;
259     lcd.vo.count=0;
260
261   }  // lcd_init
262
263   /* Initialisierung der PS/2-Schnittstellen für Tastatur und PC */
264   void ps2_init (void)
265   {
266
267     // AT_KEY_DATA, AT_KEY_CLOCK, PC_DATA und PC_CLOCK als Eingang setzen
268     PS2_DDR=PS2_DDR & ~((1<<AT_KEY_DATA) | (1<<AT_KEY_CLOCK) |
269                         (1<<PC_DATA) | (1<<PC_CLOCK));
270
271     // Normal-Modus, Tastatur mit PC verbunden
272     AT_KEY_2_PC_SWITCH_DDR=AT_KEY_2_PC_SWITCH_DDR | (1<<AT_KEY_2_PC_SWITCH);
273     AT_KEY_2_PC_SWITCH_PORT=AT_KEY_2_PC_SWITCH_PORT | (1<<AT_KEY_2_PC_SWITCH);
274
275   }  // ps2_init
276
277   /* Initialisierung der 16er-Tastatur */
278   void key16_init (void)
279   {
280
281     // Line1-Line4 und Column1-Column4 Pull-Ups
282     KEY16_LINE_PORT=KEY16_LINE_PORT | (1<<KEY16_LINE_1) | (1<<KEY16_LINE_2) |
283                                       (1<<KEY16_LINE_3) | (1<<KEY16_LINE_4);
284     KEY16_COLUMN_PORT=KEY16_COLUMN_PORT | (1<<KEY16_COLUMN_1) |
285                                           (1<<KEY16_COLUMN_2) |
286                                           (1<<KEY16_COLUMN_3) |
287                                           (1<<KEY16_COLUMN_4);
288
289     // Line1-Line4 und Column1-Column4 als Eingang setzen
290     KEY16_LINE_DDR=KEY16_LINE_DDR & ~((1<<KEY16_LINE_1) | (1<<KEY16_LINE_2) |
291                                       (1<<KEY16_LINE_3) | (1<<KEY16_LINE_4));
292     KEY16_COLUMN_DDR=KEY16_COLUMN_DDR & ~((1<<KEY16_COLUMN_1) |
293                                           (1<<KEY16_COLUMN_2) |
294                                           (1<<KEY16_COLUMN_3) |
295                                           (1<<KEY16_COLUMN_4));
296
297     // Initialisierung der Variablen
298     key16.key=0;
299     key16.oldkey=0;
300     key16.prevkey=0;
301     key16.longpress=0;
302     key16.repeat=0;
303
304   }  // key16_init
305
306   /* Initialisierung des Joysticks */
307   void joy_init (void)
308   {
309   unsigned char i;
310
311     // Initialisierung der Variablen
312     joy.potipointer=0;
313     joy.prevbuttons=0;
314     joy.oldpotiaxis=0;
315     joy.prevpotiaxis=0;
316     joy.potiaxis=0;
317     joy.btn_longpress=0;
318     joy.poti_longpress=0;
319     joy.btn_repeat=0;
320     joy.poti_repeat=0;
321     joy.oldbuttons=0x0F;
```

```
322     for (i=0;i<=3;i++)
323     {
324       joy.button[i]=1;
325       joy.poti[i]=0;
326     }
327
328     // PUll-Ups für Joystick-Knöpfe einschalten und alle Pins
329     // als Eingang setzen
330     JOY_PORT=(1<<JOY_BUTTON_0) | (1<<JOY_BUTTON_1) |
331              (1<<JOY_BUTTON_2) | (1<<JOY_BUTTON_3);
332     JOY_DDR=0x00;
333
334     // AVcc (5V) als Referenz gewählt und Ergebnis Linksbündig (8 Bit)
335     ADMUX=(1<<REFS0) | (1<<ADLAR);
336
337     // Einzulesendes Joystick-Poti bestimmen
338     ADMUX=(ADMUX & 0xE0) | (JOY_POTI_0+joy.potipointer);
339
340     // ADC einschalten, ADC-Vorteiler=128, AD-Wandlung starten und
341     // ADC-Interrupt einschalten
342     ADCSRA=ADCSRA | (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) |
343                     (1<<ADPS0) | (1<<ADSC) | (1<<ADIE);
344
345   }  // joy_init
346
347   /* Initialisierung des externen EEPROM */
348   void ext_eeprom_init (void)
349   {
350
351     // SCL und SDA als Eingang setzen
352     EXT_EEPROM_DDR=EXT_EEPROM_DDR & ~((1<<EXT_EEPROM_SCL) |
353                                       (1<<EXT_EEPROM_SDA));
354
355     // keinen Vorteiler
356     TWSR=0;
357
358     // Berechnung der Bit-Rate
359     TWBR=((F_CPU/SCL_CLOCK)-16)/2;
360
361   }  // ext_eeprom_init
362
363   /* µC Initialisieren */
364   void uc_init (void)
365   {
366
367     // Unbenutzter Pin PC3: als Eingang mit Pull-Up definieren
368     PORTC=PORTC | (1<<PC3);
369     DDRC=DDRC & ~(1<<PC3);
370
371     // Timer0-Init
372     // keinen Vorteiler
373     TCCR0=(1<<CS00);
374
375     // Timer2-Init
376     // Vorteiler F_CPU/8
377     TCCR2=(1<<CS21);
378
379     // Werte für Entprellung und Wiederholfunktion berechnen
380     debounce=(DEBOUNCE_TIME*F_CPU)/(256*TIMER2PRESCALER);
381     repeat_start=REPEAT_START_TIME/DEBOUNCE_TIME;
382     repeat_next=REPEAT_NEXT_TIME/DEBOUNCE_TIME;
383
384     // Timer-Werte initialisieren
385     timer0count=0;
386     timer2count=0;
```

```
387
388     // Timer-Interrupt Enable
389     TIMSK=(1<<TOIE0) | (1<<TOIE2);
390     sei();
391
392   }  // uc_init
393
394   /* druckt die Zeichenkette (s[]) in Spalte (x) und Zeile (y)
395      auf die LCD-Anzeige */
396   void lcd_print (unsigned char x,unsigned char y,char s[])
397   {
398   unsigned char a;
399
400     // Zeile und Spalte bestimmen
401     lcd_hex_8(0x80+(y-1)*0x40+x-1,0);
402     wait_us(50);
403
404     // Text ausgeben
405     for (a=0;s[a]!=0;a++)
406     {
407       lcd_hex_8(s[a],1);
408       wait_us(50);
409     }
410
411   }  // lcd_print
412
413   /* Abfrage der 16er-Tastatur, Rückgabe der gedrückten Taste
414      0 = keine Taste gedrückt, 1 = links oben, 2 = links 2. von oben
415      ... 16 = rechts unten */
416   unsigned char key16_scan (void)
417   {
418   unsigned char column,line,key,keyin;
419
420     key=0;
421
422     for (line=0;line<=3 && key==0;line++)
423     {
424       // Zeile als Ausgang schalten und auf Low setzen
425       KEY16_LINE_DDR=KEY16_LINE_DDR | (1<<(KEY16_LINE_1+line));
426       KEY16_LINE_PORT=KEY16_LINE_PORT & ~(1<<(KEY16_LINE_1+line));
427
428       // kleine Pause
429       asm("NOP");
430       asm("NOP");
431
432       // Spalte einlesen
433       column=((KEY16_COLUMN_IN>>KEY16_COLUMN_1)&0xF);
434
435       // Zeile einlesen
436       keyin=((KEY16_LINE_IN>>KEY16_LINE_1)&0xF);
437
438       // Zeile wieder als Eingang setzten
439       KEY16_LINE_DDR=KEY16_LINE_DDR & ~(1<<(KEY16_LINE_1+line));
440       KEY16_LINE_PORT=KEY16_LINE_PORT | (1<<(KEY16_LINE_1+line));
441
442       // Ist in jeder Zeile und Spalte nur eine Taste gedrückt?
443       // Mehrere Tasten diagonal werden nicht erkannt
444       if (((column==0xE) || (column==0xD) || (column==0xB) || (column==0x7)) &&
445             ((keyin==0xE) || (keyin==0xD) || (keyin==0xB) || (keyin==0x7)))
446       {
```

```
447         // Berechnung der gedrückten Taste
448         switch (column)
449         {
450           case 0xE :
451           {
452         key=line+1;
453         break;
454           }
455       case 0xD :
456           {
457             key=line+5;
458             break;
459           }
460       case 0xB :
461           {
462         key=line+9;
463         break;
464           }
465       case 0x7 :
466           {
467         key=line+13;
468         break;
469           }
470         }  // switch (column)
471
472       }  // if (((column==0xE) || (column==0xD) || (column==0xB) ||
473          //      (column==0x7)) && ((keyin==0xE) || (keyin==0xD) ||
474          //      (keyin==0xB) || (keyin==0x7)))
475
476     }  // for (line=0;line<=3 && key==0;line++)
477
478     return key;
479
480   }  // key16_scan
481
482   /* Startet Übertragung mit externem EEPROM und sendet die HW-Adresse */
483   void ext_eeprom_start (unsigned char address)
484   {
485
486     // sende Start-Bedingung
487     TWCR=(1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
488
489     // warte bis Übertragung abgeschlossen ist
490     while (!(TWCR & (1<<TWINT))) {}
491
492     // sende HW-Adresse
493     TWDR=address;
494     TWCR=(1<<TWINT) | (1<<TWEN);
495
496     // warte bis Übertragung abgeschlossen ist
497     while (!(TWCR & (1<<TWINT))) {}
498
499   }  // ext_eeprom_start
500
501   /* Startet Übertragung mit externem EEPROM und sendet die HW-Adresse
502      wartet dabei bis EEPROM ansprechbar ist */
503   void ext_eeprom_start_wait (unsigned char address)
504   {
505   unsigned char twst;
506
507     while (1)
508     {
509       // sende Start-Bedingung
510       TWCR=(1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
511
```

```
512       // warte bis Übertragung abgeschlossen ist
513       while (!(TWCR & (1<<TWINT))) {}
514
515       // Übertragung zum EEPROM erfolgreich?
516       twst=TW_STATUS & 0xF8;
517       if ((twst!=TW_START) && (twst!=TW_REP_START))
518       {
519         continue;
520       }
521
522       // sende HW-Adresse
523       TWDR=address;
524       TWCR=(1<<TWINT) | (1<<TWEN);
525
526       // warte bis Übertragung abgeschlossen ist
527       while (!(TWCR & (1<<TWINT))) {}
528
529       // Übertragung zum EEPROM erfolgreich?
530       twst=TW_STATUS & 0xF8;
531       if ((twst==TW_MT_SLA_NACK) || (twst==TW_MR_DATA_NACK))
532       {
533         // EEPROM nicht ansprechbar, sende Stop-Bedingung
534         TWCR=(1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
535
536         // warte bis stop-Bedingung übertragen wurde
537         while (TWCR & (1<<TWSTO)) {}
538
539         continue;
540       }  // if ((twst==TW_MT_SLA_NACK) || (twst==TW_MR_DATA_NACK))
541       break;
542     }
543
544   }  // ext_eeprom_start_wait
545
546   /* Beendet die Übertragung zum externen EEPROM */
547   void ext_eeprom_stop (void)
548   {
549
550     // sende Stop-Bedingung
551     TWCR=(1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
552
553     // warte bis stop-Bedingung übertragen wurde
554     while (TWCR & (1<<TWSTO)) {}
555
556   }  // ext_eeprom_stop
557
558   /* Sendet ein Byte zum externen EEPROM */
559   void ext_eeprom_write_byte (unsigned char data)
560   {
561
562     // sende data zum externen EEPROM
563     TWDR=data;
564     TWCR=(1<<TWINT) | (1<<TWEN);
565
566     // warte bis Übertragung abgeschlossen ist
567     while (!(TWCR & (1<<TWINT))) {}
568
569   }  // ext_eeprom_write_byte
570
```

```
571   /* Liest ein Byte vom externen EEPROM, weitere Bytes können gelesen werden */
572   unsigned char ext_eeprom_read_ack (void)
573   {
574
575     TWCR=(1<<TWINT) | (1<<TWEN) | (1<<TWEA);
576
577     // warte bis Übertragung abgeschlossen ist
578     while (!(TWCR & (1<<TWINT))) {}
579
580     return TWDR;
581   }  // ext_eeprom_read_ack
582
583   /* Liest ein letztes Byte vom externen EEPROM */
584   unsigned char ext_eeprom_read_nak (void)
585   {
586
587     TWCR=(1<<TWINT) | (1<<TWEN);
588
589     // warte bis Übertragung abgeschlossen ist
590     while (!(TWCR & (1<<TWINT))) {}
591
592     return TWDR;
593
594   }  // ext_eeprom_read_nak
595
596   /* Speichert (data[]) im externem EEPROM ab (address), letztes Byte ist 0 */
597   void ext_eeprom_write_buffer (unsigned int address, unsigned char data[])
598   {
599   unsigned char i;
600
601     i=0;
602     while (data[i]!=0)
603     {
604       // EEPROM-Adresse senden
605       ext_eeprom_start_wait(EXT_EEPROM_ADDRESS+EXT_EEPROM_WRITE);
606
607       // Zuerst High- und dann Low-Byte der Speicher-Adresse senden
608       ext_eeprom_write_byte((address>>8) & 0xFF);
609       ext_eeprom_write_byte(address & 0xFF);
610
611       // Daten senden bis EEPROM-Seitengrenze erreicht oder keine Daten mehr
612       do
613       {
614         ext_eeprom_write_byte(data[i]);
615         address++;
616         i++;
617       }
618       while (address%128!=0 && data[i]!=0);
619
620       // Datenübertragung beenden
621       ext_eeprom_stop();
622     }  // while (data[i]!=0)
623
624     // EEPROM-Adresse senden
625     ext_eeprom_start_wait(EXT_EEPROM_ADDRESS+EXT_EEPROM_WRITE);
626
627     // Zuerst High- und dann Low-Byte der Speicher-Adresse senden
628     ext_eeprom_write_byte((address>>8) & 0xFF);
629     ext_eeprom_write_byte(address & 0xFF);
630
631     // End-Markierung senden
632     ext_eeprom_write_byte(0);
633
634     // Datenübertragung beenden
635     ext_eeprom_stop();
```

```
636
637   }  // ext_eeprom_write_buffer
638
639   /* Liest aus dem externem EEPROM ab (address) bis eine 0 oder FF gelesen
640      wird und speichert in (data[]) */
641   void ext_eeprom_read_buffer (unsigned int address, unsigned char data[])
642   {
643   unsigned char i,d;
644
645     // EEPROM-Adresse senden
646     ext_eeprom_start_wait(EXT_EEPROM_ADDRESS+EXT_EEPROM_WRITE);
647
648     // Zuerst High- und dann Low-Byte der Speicher-Adresse senden
649     ext_eeprom_write_byte((address>>8) & 0xFF);
650     ext_eeprom_write_byte(address & 0xFF);
651
652     // EEPROM auf Lesen stellen
653     ext_eeprom_start(EXT_EEPROM_ADDRESS+EXT_EEPROM_READ);
654
655     // Daten empfangen
656     d=1;
657     for (i=0;d!=0xFF && d!=0;i++)
658     {
659       d=ext_eeprom_read_ack();
660       data[i]=d;
661     }
662     ext_eeprom_read_nak();
663
664     // Datenübertragung beenden
665     ext_eeprom_stop();
666
667   }  // ext_eeprom_read_buffer
668
669   /* Sendet an die PC-Clockleitung einen Low-High-Impuls */
670   void pc_lowhigh (void)
671   {
672
673     _delay_us(20);
674
675     // Clockleitung Low
676     PS2_PORT=PS2_PORT & ~(1<<PC_CLOCK);
677     PS2_DDR=PS2_DDR | (1<<PC_CLOCK);
678     _delay_us(40);
679
680     // Clockleitung High
681     PS2_DDR=PS2_DDR & ~(1<<PC_CLOCK);
682     _delay_us(20);
683
684   }  // pc_lowhigh
685
686   /* Sendet (sdata) an den PC */
687   void ch_to_pc (unsigned char sdata)
688   {
689   unsigned char parity,datbit,in;
690
691     // Timer2-Interrupt ausschalten
692     TIMSK=TIMSK & ~(1<<TOIE2);
693
694     // Warten bis Clock- und Datenleitung für mindestens 50µs high sind
695     in=PS2_IN;
696     timer0count=0;
697     while (!(in & (1<<PC_CLOCK)) || !(in & (1<<PC_DATA)) || timer0count<5)
698     {
```

```
699       in=PS2_IN;
700       if (!(in & (1<<PC_CLOCK)) || !(in & (1<<PC_DATA)))
701       {
702         timer0count=0;
703       }
704     }  // while (!(in & (1<<PC_CLOCK)) || !(in & (1<<PC_DATA)) || timer0count<5)
705
706     parity=0;
707
708     // Startbit, Datenleitung Low
709     PS2_PORT=PS2_PORT & ~(1<<PC_DATA);
710     PS2_DDR=PS2_DDR | (1<<PC_DATA);
711     pc_lowhigh();
712
713     // 8 Datenbits senden
714     for (datbit=0;datbit<=7;datbit++)
715     {
716       // wenn Bit = 1
717       if (sdata & (1<<datbit))
718       {
719         PS2_DDR=PS2_DDR & ~(1<<PC_DATA);
720         parity++;
721       }
722       // wenn Bit = 0
723       else
724       {
725         PS2_PORT=PS2_PORT & ~(1<<PC_DATA);
726         PS2_DDR=PS2_DDR | (1<<PC_DATA);
727       }
728       pc_lowhigh();
729     }  // for (datbit=0;datbit<=7;datbit++)
730
731     // ungerade Parität senden
732     // gerade Anzahl von Einsen
733     if (!(parity % 2))
734     {
735       PS2_DDR=PS2_DDR & ~(1<<PC_DATA);
736     }
737     // ungerade Anzahl von Einsen
738     else
739     {
740       PS2_PORT=PS2_PORT & ~(1<<PC_DATA);
741       PS2_DDR=PS2_DDR | (1<<PC_DATA);
742     }
743     pc_lowhigh();
744
745     //Stopbit senden
746     PS2_DDR=PS2_DDR & ~(1<<PC_DATA);
747     pc_lowhigh();
748
749     // Timer2-Interrupt wieder einschalten
750     TIMSK=TIMSK | (1<<TOIE2);
751
752   }  // ch_to_pc
753
754   // Tasten-Sequenz an PC senden
755   void send_to_pc (unsigned char digit)
756   {
757   unsigned char i;
758   unsigned int startaddress;
759
```

```
760     // EEPROM-Adresse bestimmen
761     if (set_mode==SET_PASSWORD)
762     {
763       startaddress=PASSWORD_ADDRESS+MAX_KEY_DATA*(passwordset*10+digit);
764     }
765     if (set_mode==SET_KEY)
766     {
767       startaddress=MAX_KEY_DATA*(keyset*10+digit);
768     }
769     if (set_mode==SET_JOY)
770     {
771       startaddress=JOYSET_ADDRESS+MAX_KEY_DATA*(joyset*12+digit);
772     }
773
774     // Tasten-Sequenz aus externem EEPROM lesen
775     ext_eeprom_read_buffer(startaddress,keybuffer.data);
776
777     // Verbindung zwischen Tastatur und PC getrennt
778     AT_KEY_2_PC_SWITCH_PORT=AT_KEY_2_PC_SWITCH_PORT & ~(1<<AT_KEY_2_PC_SWITCH);
779     _delay_us(1);
780
781     do
782     {
783       // Tasten-Sequenz an PC senden
784       for (i=0;keybuffer.data[i]!=0x00 && keybuffer.data[i]!=0xFF;i++)
785       {
786         ch_to_pc(keybuffer.data[i]);
787       }
788       wait_ms(30);
789     }
790     while ((set_mode!=SET_JOY && key16.longpress) ||
791            (set_mode==SET_JOY && digit<=3 && joy.btn_longpress) ||
792            (set_mode==SET_JOY && digit>=4 && joy.poti_longpress));
793     // solange bis Taste bzw. Joystick nicht mehr betätigt wird
794
795     // Normal-Modus, Tastatur mit PC verbunden
796     AT_KEY_2_PC_SWITCH_PORT=AT_KEY_2_PC_SWITCH_PORT | (1<<AT_KEY_2_PC_SWITCH);
797     _delay_us(1);
798
799   }  // send_to_pc
800
801   /* Anzeige des aktuellen Modus (Keyset, Joyset oder Passwordset) */
802   void show_set (void)
803   {
804     switch (set_mode)
805     {
806       case SET_KEY :
807       {
808         sprintf(lcdstr,"Keyset: %u",keyset);
809         break;
810       }
811       case SET_JOY :
812       {
813         sprintf(lcdstr,"Joyset: %u",joyset);
814         break;
815       }
816       case SET_PASSWORD :
817       {
818         sprintf(lcdstr,"Passwordset: %u",passwordset);
819         break;
820       }
821     }
822     lcd_clear();
823     lcd_print(1,1,lcdstr);
824
```

```
825   }  // show_set
826
827   /* Bestimmen des Satzes (0-9) im aktuellen Modus
828      (Keyset, Joyset oder Passwordset) */
829   void set_keyset (unsigned char x,unsigned char y,unsigned char *set)
830   {
831
832     show_set();
833     lcd_print(x,y,"?");
834
835     // Warten bis Ziffern-Taste gedrückt wurde oder Abbruch
836     while (digit16[key16.key]==-1 && key16.key!=KEY_CANCEL) {}
837     if (key16.key!=KEY_CANCEL)
838     {
839       *set=digit16[key16.key];
840     }
841     show_set();
842
843     // Warten bis Taste losgelassen wurde
844     while (key16.key!=0) {}
845
846   }  // set_keyset
847
848   /* Eingabe der PIN */
849   void enter_pin (unsigned char pin[])
850   {
851   unsigned char i;
852
853     for (i=0;i<5 && key16.key!=KEY_CANCEL;i++)
854     {
855       // Warten bis Ziffern-Taste gedrückt wurde oder Abbruch
856       while (digit16[key16.key]==-1 && key16.key!=KEY_CANCEL) {}
857       if (key16.key!=KEY_CANCEL)
858       {
859         pin[i]=digit16[key16.key]+1;
860         lcd_print(5+i,2,"*");
861
862         // Warten bis Taste losgelassen wurde
863         while (key16.key!=0) {}
864
865       }  // if (key16.key!=KEY_CANCEL)
866     }  // for (i=0;i<5 && key16.key!=KEY_CANCEL;i++)
867
868   }  // enter_pin
869
870   /* Eintritt in den Passwort-Modus */
871   void set_keypasswordset (void)
872   {
873   unsigned char i,pin[6],oldpin[6];
874
875     // PIN aus EEPROM lesen
876     ext_eeprom_read_buffer(PIN_ADDRESS,oldpin);
877
878     // wenn Modus ungleich Passwort-Modus und PIN vorhanden
879     if (set_mode!=SET_PASSWORD && oldpin[0]!=0xFF)
880     {
881       lcd_clear();
882       lcd_print(3,1,"Enter PIN!");
883       lcd_print(5,2,"-----");
884
885       enter_pin(pin);
886
```

```
887       // PIN-Eingabe abgeschlossen
888       if (key16.key!=KEY_CANCEL)
889       {
890         for (i=0;i<5 && oldpin[i]==pin[i];i++) {}
891         // PIN nicht korrekt eingegeben
892         if (i!=5)
893         {
894           lcd_clear();
895           lcd_print(5,1,"Error!");
896           wait_ms(MS2WAIT);
897           show_set();
898           return;
899         }  // if (i!=5)
900       }  // if (key16.key!=KEY_CANCEL)
901
902       // PIN-Eingabe wurde durch CANCEL abgebrochen
903       else
904       {
905         show_set();
906         return;
907       }
908     }  // if (set_mode!=SET_PASSWORD && oldpin[0]!=0xFF)
909
910     //  Bestimmen des Passwort-Satzes
911     set_mode=SET_PASSWORD;
912     lcd_clear();
913     lcd_print(1,1,"Passwordset:");
914     set_keyset(14,1,&passwordset);
915
916   }  // set_keypasswordset
917
918   /* LCD-Hintergrundbeleuchtung und -Kontrast verändern */
919   void setup_backlight_contrast (unsigned char setupitem)
920   {
921   unsigned char exitmenu,oldvalue,buffer[2];
922   unsigned int startaddress;
923
924     exitmenu=NO;
925     oldvalue=*ledvo_trig;
926     lcd_clear();
927     lcd_print(1,1,lcdstr);
928
929     // Warten bis Taste losgelassen wurde
930     while (key16.key!=0) {}
931     while (exitmenu==NO)
932     {
933       while (digit16[key16.key]==-1 &&
934              key16.key!=KEY_OK && key16.key!=KEY_CANCEL) {}
935
936       // Wurde eine der Zifferntasten gedrückt?
937       if (digit16[key16.key]!=-1)
938       {
939         // LCD-Hintergrundbeleuchtung bzw. -Kontrast verändern
940         *ledvo_trig=digit16[key16.key];
941         buffer[0]=*ledvo_trig;
942
943         // neuen Wert anzeigen
944         sprintf(lcdstr,"%u",*ledvo_trig);
945         lcd_print(16,1,lcdstr);
946
947         // Warten bis Taste losgelassen wurde
948         while (key16.key!=0) {}
949       }  // if (digit16[key16.key]!=-1)
950
```

```
951       // Abbruch
952       if (key16.key==KEY_CANCEL)
953       {
954         // LCD-Hintergrundbeleuchtung bzw. -Kontrast wieder alten Wert zuweisen
955         *ledvo_trig=oldvalue;
956         exitmenu=CANCEL;
957       }
958
959       // Neuen Wert speichern
960       if (key16.key==KEY_OK)
961       {
962         if (oldvalue!=*ledvo_trig)
963         {
964           if (setupitem==SETUP_BACKLIGHT)
965           {
966             startaddress=BACKLIGHT_ADDRESS;
967             sprintf(lcdstr,"Backlight saved!");
968           }
969           else
970           {
971             startaddress=CONTRAST_ADDRESS;
972             sprintf(lcdstr,"Contrast saved!");
973           }
974
975           // Neuen Wert in externem EEPROM speichern
976           buffer[1]=0;
977           ext_eeprom_write_buffer(startaddress,buffer);
978           lcd_clear();
979           lcd_print(1,1,lcdstr);
980           wait_ms(MS2WAIT);
981         }  // if (oldvalue!=*ledvo_trig)
982         exitmenu=OK;
983       }  // if (key16.key==KEY_OK)
984     }  // while (exitmenu==NO)
985
986     // Setup wieder anzeigen
987     lcd_clear();
988     lcd_print(1,1,"Setup:");
989     if (setupitem==SETUP_BACKLIGHT)
990     {
991       sprintf(lcdstr,"LCD-Backlight: %u",*ledvo_trig);
992     }
993     else
994     {
995       sprintf(lcdstr,"LCD-Contrast:  %u",*ledvo_trig);
996     }
997     lcd_print(1,2,lcdstr);
998
999     // Warten bis Taste losgelassen wurde
1000    while (key16.key!=0) {}
1001
1002  }  // setup_backlight_contrast
1003
1004  /* Zweifache Eingabe einer neuen PIN */
1005  void enter_new_pins (void)
1006  {
1007  unsigned char i,pin[6],oldpin[6];
1008
1009    lcd_clear();
1010    lcd_print(2,1,"Enter New PIN!");
1011    lcd_print(5,2,"-----");
1012
1013    // PIN das erste mal eingeben
1014    enter_pin(oldpin);
1015
```

```
1016      // PIN-Eingabe abgeschlossen
1017      if (key16.key!=KEY_CANCEL)
1018      {
1019        lcd_clear();
1020        lcd_print(2,1,"Re-Enter PIN!");
1021        lcd_print(5,2,"-----");
1022
1023        // PIN das zweite mal zur Sicherheit eingeben
1024        enter_pin(pin);
1025
1026        // zweite PIN-Eingabe abgeschlossen
1027        if (key16.key!=KEY_CANCEL)
1028        {
1029          // zwei mal der gleiche PIN eingegeben
1030          for (i=0;i<5 && oldpin[i]==pin[i];i++) {}
1031          if (i==5)
1032          {
1033            pin[5]=0;
1034
1035            // neue PIN speichern
1036            ext_eeprom_write_buffer(PIN_ADDRESS,pin);
1037
1038            lcd_clear();
1039            lcd_print(4,1,"PIN saved!");
1040            wait_ms(MS2WAIT);
1041            return;
1042          }  // if (i==5)
1043
1044          // Fehler bei der zweiten Eingabe der PIN
1045          lcd_clear();
1046          lcd_print(5,1,"Error!");
1047          wait_ms(MS2WAIT);
1048
1049        }  // if (key16.key!=KEY_CANCEL)
1050      }  // if (key16.key!=KEY_CANCEL)
1051
1052    }  // enter_new_pins
1053
1054    /* Neue PIN eingeben */
1055    void setup_pin (void)
1056    {
1057    unsigned char i,pin[6],oldpin[6];
1058
1059      // alte PIN aus EEPROM lesen
1060      ext_eeprom_read_buffer(PIN_ADDRESS,oldpin);
1061
1062      // Es steht noch keine PIN im EEPROM
1063      if (oldpin[0]==0xFF)
1064      {
1065        enter_new_pins();
1066      }
1067
1068      // Es steht bereits eine PIN im EEPROM
1069      else
1070      {
1071        lcd_clear();
1072        lcd_print(2,1,"Enter Old PIN!");
1073        lcd_print(5,2,"-----");
1074
1075        enter_pin(pin);
1076
```

```
1077        // PIN-Eingabe abgeschlossen
1078        if (key16.key!=KEY_CANCEL)
1079        {
1080          // alte PIN korrekt eingegeben
1081          for (i=0;i<5 && oldpin[i]==pin[i];i++) {}
1082          if (i==5)
1083          {
1084            enter_new_pins();
1085          }
1086
1087          // alte PIN nicht korrekt eingegeben
1088          else
1089          {
1090            lcd_clear();
1091            lcd_print(5,1,"Error!");
1092            wait_ms(MS2WAIT);
1093          }
1094
1095        }  // if (key16.key!=KEY_CANCEL)
1096      }  // if (oldpin[0]!=0xFF)
1097
1098      // Setup wieder anzeigen
1099      lcd_clear();
1100      lcd_print(1,1,"Setup:");
1101      lcd_print(1,2,"PIN");
1102
1103      // Warten bis Taste losgelassen wurde
1104      while (key16.key!=0) {}
1105
1106    }  // setup_pin
1107
1108    /* Joystick Kalibrieren */
1109    void setup_joystick (void)
1110
1111    {
1112    struct joy_min_max
1113    {
1114      unsigned char min,mid,max;
1115    };
1116
1117    unsigned char i,r_max,r_min,r_mid,joycalbuffer[9];
1118    struct joy_min_max joytotal[4];
1119
1120      lcd_clear();
1121      lcd_print(1,1,"Center Joystick");
1122      lcd_print(1,2,"and press button");
1123
1124      // Mittelstellung bestimmen
1125      while (joy.button[0]==1 && joy.button[1]==1 && joy.button[2]==1 &&
1126             joy.button[3]==1 && key16.key!=KEY_CANCEL) {}
1127
1128      for (i=0;i<=3;i++)
1129      {
1130        if (joy.poti[i]>20)
1131        {
1132          joytotal[i].min=joy.poti[i];
1133          joytotal[i].max=joy.poti[i];
1134          joytotal[i].mid=joy.poti[i];
1135        }
1136        else
1137        {
1138          joytotal[i].min=5;
1139          joytotal[i].max=5;
1140          joytotal[i].mid=5;
1141        }
```

```
1142        joy.cal[i].trig_mid_max=5;
1143        joy.cal[i].trig_mid_min=5;
1144      }  // for (i=0;i<=3;i++)
1145
1146      lcd_clear();
1147      lcd_print(2,1,"Move Joystick");
1148      lcd_print(1,2,"and press button");
1149
1150      // Warten bis kein Joystick-Knopf mehr gedrückt ist
1151      while (joy.button[0]==0 || joy.button[1]==0 ||
1152             joy.button[2]==0 || joy.button[3]==0) {}
1153
1154      // Minimale und Maximale Werte bestimmen
1155      while (joy.button[0]==1 && joy.button[1]==1 && joy.button[2]==1 &&
1156             joy.button[3]==1 && key16.key!=KEY_CANCEL)
1157      {
1158        for (i=0;i<=3;i++)
1159        {
1160          if (joy.poti[i]>20)
1161          {
1162            if (joy.poti[i]>joytotal[i].max)
1163            {
1164              joytotal[i].max=joy.poti[i];
1165            }
1166            if (joy.poti[i]<joytotal[i].min)
1167            {
1168              joytotal[i].min=joy.poti[i];
1169            }
1170          }  // if (joy.poti[i]>20)
1171        }  // for (i=0;i<=3;i++)
1172      }  // while (joy.button[0]==1 && joy.button[1]==1 && joy.button[2]==1 &&
1173         //        joy.button[3]==1 && key16.key!=KEY_CANCEL)
1174
1175      // Warten bis kein Joystick-Knopf mehr gedrückt ist
1176      while (joy.button[0]==0 || joy.button[1]==0 ||
1177             joy.button[2]==0 || joy.button[3]==0) {}
1178
1179      if (key16.key!=KEY_CANCEL)
1180      {
1181        // Auslöse-Werte bestimmen, bei denen eine Bewegung des Joysticks
1182        // erkannt wird,
1183        // von der Mitte nach Außen bei 1/2 Vollausschlag
1184        for (i=0;i<=3;i++)
1185        {
1186          // Joystick-Poti angeschlossen?
1187          if (joytotal[i].mid>20)
1188          {
1189            // Zuerst Rückrechnung auf Widerstand, da AD-Werte nicht linear
1190            // zum Widerstand sind
1191            r_min=(100*255/joytotal[i].max)-100;
1192            r_max=(100*255/joytotal[i].min)-100;
1193            r_mid=(100*255/joytotal[i].mid)-100;
1194
1195            // Bestimmung der Auslösepunkte
1196            joy.cal[i].trig_mid_max=100*255/(r_min+((r_mid-r_min)/2)+100);
1197            joy.cal[i].trig_mid_min=100*255/(r_mid+((r_max-r_mid)/2)+100);
1198
1199          }  // if (joytotal[i].mid>20)
1200        }  // for (i=0;i<=3;i++)
1201
```

```
1202       // Anlegen des Feldes zum Speichern der Kalibrier-Werte
1203       for (i=0;i<=3;i++)
1204       {
1205         joycalbuffer[i*2]=joy.cal[i].trig_mid_max;
1206         joycalbuffer[i*2+1]=joy.cal[i].trig_mid_min;
1207       }
1208       joycalbuffer[8]=0;
1209
1210       // Kalibrier-Werte speichern
1211       ext_eeprom_write_buffer(JOYCAL_ADDRESS,joycalbuffer);
1212
1213       lcd_clear();
1214       lcd_print(4,1,"Joystick");
1215       lcd_print(3,2,"calibrated!");
1216       wait_ms(MS2WAIT);
1217
1218     }  // if (key16.key!=KEY_CANCEL)
1219
1220     // Setup wieder anzeigen
1221     lcd_clear();
1222     lcd_print(1,1,"Setup:");
1223     lcd_print(1,2,"Joystick");
1224
1225     // Warten bis Taste losgelassen wurde
1226     while (key16.key!=0) {}
1227
1228   }  // setup_joystick
1229
1230   // Bestimmen ob Joystick betätigt wurde (Rückgabe=1)
1231   unsigned char joypress (void)
1232   {
1233   unsigned char i;
1234
1235     for (i=0;i<=3;i++)
1236     {
1237       if (joy.button[i]==0 || (joy.poti[i]>20 &&
1238          (joy.poti[i]<joy.cal[i].trig_mid_min ||
1239           joy.poti[i]>joy.cal[i].trig_mid_max)))
1240       {
1241         return 1;
1242       }
1243     }
1244     return 0;
1245
1246   }  // joypress
1247
1248   /* Programmieren einer Tastatur-Sequenz auf eine Taste der 16er-Tastatur
1249      bzw. einer Joystick-Funktion */
1250   void program_keyset (void)
1251   {
1252   unsigned char i,byteleft,savebyteleft,old,breakcode,extendet,atkey,
1253                 joyfunc,oldjoyfunc,store;
1254   unsigned int startaddress;
1255   char joyfuncstr[][9]={"Button 1","Button 2","Button 3","Button 4",
1256                         "Left 1","Up 1","Left 2","Up 2",
1257                         "Right 1","Down 1","Right 2","Down 2"};
1258
1259     // Variablen initialisieren
1260     for (i=0;i<MAX_KEY_DATA;i++)
1261     {
1262       keybuffer.data[i]=0;
1263     }
1264     keybuffer.pointer=0;
1265     byteleft=MAX_KEY_DATA-1;
1266     savebyteleft=byteleft;
```

```
1267     joyfunc=0;
1268     oldjoyfunc=1;
1269     old=0;
1270     breakcode=0;
1271     extendet=0;
1272     store=0;
1273     keyat.bitcount=11;
1274     keyat.finish=0;
1275
1276     // Verbindung zwischen Tastatur und PC getrennt
1277     AT_KEY_2_PC_SWITCH_PORT=AT_KEY_2_PC_SWITCH_PORT & ~(1<<AT_KEY_2_PC_SWITCH);
1278     _delay_us(1);
1279
1280     // Setze Interrupt auf fallende Flanke
1281     keyat.clockedge=0;
1282     MCUCR=MCUCR & ~(1<<ISC00);
1283
1284     // PC-Tastatur-Interrupt einschalten
1285     GICR=GICR | (1<<INT0);
1286
1287     // Anzeige des richtigen Modus
1288     switch (set_mode)
1289     {
1290       case SET_KEY :
1291       {
1292         sprintf(lcdstr,"Program KS: %u",keyset);
1293         break;
1294       }
1295       case SET_JOY :
1296       {
1297         sprintf(lcdstr,"Program JS: %u",joyset);
1298         break;
1299       }
1300       case SET_PASSWORD :
1301       {
1302         sprintf(lcdstr,"Program PS: %u",passwordset);
1303         break;
1304       }
1305     }  // switch (set_mode)
1306     lcd_clear();
1307     lcd_print(1,1,lcdstr);
1308
1309     // Anzeige der freien Bytes
1310     sprintf(lcdstr,"Byte left: %u",byteleft);
1311     lcd_print(1,2,lcdstr);
1312
1313     // Warten bis Taste losgelassen wurde
1314     while (key16.key!=0) {}
1315
1316     while (key16.key!=KEY_CANCEL &&
1317           ((set_mode!=SET_JOY && digit16[key16.key]==-1) ||
1318            (set_mode==SET_JOY && joypress()==0)))
1319     {
1320       // Anzeige der freien Bytes
1321       if (byteleft!=savebyteleft)
1322       {
1323         sprintf(lcdstr,"%u  ",byteleft);
1324         lcd_print(12,2,lcdstr);
1325         savebyteleft=byteleft;
1326       }
1327
1328       while (!keyat.finish && key16.key!=KEY_CANCEL &&
1329             ((set_mode!=SET_JOY && digit16[key16.key]==-1) ||
1330              (set_mode==SET_JOY && joypress()==0))) {}
1331
```

```
1332       // Neues Zeichen von der PS/2-Tastatur empfangen
1333       if (keyat.finish && byteleft>0)
1334       {
1335         atkey=keyat.data;
1336
1337         if (breakcode==2)
1338         {
1339           breakcode=0;
1340           old=0;
1341         }
1342         if (breakcode==1)
1343         {
1344           breakcode=2;
1345         }
1346         // Breakcode eingeleitet
1347         if (atkey==0xF0)
1348         {
1349           breakcode=1;
1350         }
1351
1352         if (extendet==2)
1353         {
1354           extendet=0;
1355           old=0;
1356         }
1357         if (extendet==1)
1358         {
1359           // Widerholfunktion bei Extendet-Keycode ausschalten
1360           if (keybuffer.pointer>2 &&
1361               keybuffer.data[keybuffer.pointer-3]==0xE0 &&
1362               keybuffer.data[keybuffer.pointer-2]==atkey)
1363           {
1364             keybuffer.pointer--;
1365             keybuffer.data[keybuffer.pointer]=0;
1366             byteleft++;
1367             old=atkey;
1368           }
1369           extendet=2;
1370         }  // if (extendet==1)
1371
1372         // Extendet Keycode eingeleitet
1373         if (atkey==0xE0)
1374         {
1375           extendet=1;
1376         }
1377
1378         // Neues empfangenes Zeichen ist ungleich dem vorigen Zeichen
1379         if (old!=atkey)
1380         {
1381           // Zeichen speichern
1382           keybuffer.data[keybuffer.pointer]=atkey;
1383           keybuffer.pointer++;
1384           old=atkey;
1385           byteleft--;
1386         }  // if (old!=atkey)
1387         keyat.finish=0;
1388
1389       }  // if (keyat.finish && byteleft>0)
1390     }  // while (key16.key!=KEY_CANCEL &&
1391        //       ((set_mode!=SET_JOY && digit16[key16.key]==-1) ||
1392        //        (set_mode==SET_JOY && joypress()==0)))
1393
```

```
1394    // Es wurden Zeichen auf der PS/2-Tastatur eingegeben
1395    if (keybuffer.pointer>0)
1396    {
1397      keybuffer.data[keybuffer.pointer]=0;
1398
1399      // Joystick-Modus
1400      if (set_mode==SET_JOY && key16.key!=KEY_CANCEL)
1401      {
1402        lcd_clear();
1403        lcd_print(1,1,"Show Joy-Func");
1404        lcd_print(13,2,"[OK]");
1405
1406        // Warten bis Taste losgelassen wurde
1407        while (key16.key!=0) {}
1408
1409        // Bestimmen der Joystick-Funktion
1410        while (key16.key!=KEY_OK && key16.key!=KEY_CANCEL)
1411        {
1412          // Joystick-Knopf
1413          for (i=0;i<=3;i++)
1414          {
1415            if (joy.button[i]==0)
1416            {
1417              joyfunc=i;
1418            }
1419          }  // for (i=0;i<=3;i++)
1420
1421          // Joystick_Achse
1422          if (joy.potiaxis>0)
1423          {
1424            joyfunc=joy.potiaxis+3;
1425          }
1426
1427          // Anzeigen der Joystick-Funktion
1428          if (oldjoyfunc!=joyfunc)
1429          {
1430            oldjoyfunc=joyfunc;
1431            lcd_print(1,2,"          ");
1432            lcd_print(1,2,joyfuncstr[joyfunc]);
1433          }
1434        }  // while (key16.key!=KEY_OK && key16.key!=KEY_CANCEL)
1435        if (key16.key==KEY_OK)
1436        {
1437          // Tasten-Sequenz im externen EEPROM speichern
1438          startaddress=JOYSET_ADDRESS+MAX_KEY_DATA*(joyset*12+joyfunc);
1439          store=1;
1440        }
1441      }  // if (set_mode==SET_JOY && key16.key!=KEY_CANCEL)
1442
1443      // Tastatur- bzw. Passwort-Modus
1444      else
1445      {
1446        if (digit16[key16.key]!=-1)
1447        {
1448          // Tasten-Sequenz im externen EEPROM speichern
1449          if (set_mode==SET_PASSWORD)
1450          {
1451            startaddress=PASSWORD_ADDRESS+
1452                         MAX_KEY_DATA*(passwordset*10+digit16[key16.key]);
1453          }
1454          else
1455          {
1456            startaddress=MAX_KEY_DATA*(keyset*10+digit16[key16.key]);
1457          }
1458          store=1;
```

```
1459          }  // if (digit16[key16.key]!=-1)
1460        }  // if !(set_mode==SET_JOY && key16.key!=KEY_CANCEL)
1461      }  // if (keybuffer.pointer>0)
1462
1463      if (store)
1464      {
1465        ext_eeprom_write_buffer(startaddress,keybuffer.data);
1466        lcd_clear();
1467        lcd_print(3,1,"Data saved!");
1468        wait_ms(MS2WAIT);
1469      }  // if (store)
1470
1471      // PC-Tastatur-Interrupt ausschalten
1472      GICR=GICR & ~(1<<INT0);
1473
1474      // Normal-Modus, Tastatur mit PC verbunden
1475      AT_KEY_2_PC_SWITCH_PORT=AT_KEY_2_PC_SWITCH_PORT | (1<<AT_KEY_2_PC_SWITCH);
1476      _delay_us(1);
1477
1478      // Setup wieder anzeigen
1479      lcd_clear();
1480      lcd_print(1,1,"Setup:");
1481      lcd_print(1,2,"Program");
1482
1483      // Warten bis Taste losgelassen wurde
1484      while (key16.key!=0) {}
1485
1486    }  // program_keyset
1487
1488    /* Wechselt den aktuellen Setup-Eintrag und zeigt ihn an */
1489    void setup_toggle (unsigned char *setupitem)
1490    {
1491
1492      // Setup-Eintrag erhöhen und bei Überlauf auf Anfang setzen
1493      (*setupitem)++;
1494      if (*setupitem==LAST_SETUP_ITEM+1)
1495      {
1496        *setupitem=0;
1497      }
1498
1499      // Zeichenkette für neuen Setup-Eintrag bestimmen
1500      switch (*setupitem)
1501      {
1502        case SETUP_PROGRAM :
1503        {
1504          sprintf(lcdstr,"Program         ");
1505          break;
1506        }
1507        case SETUP_JOYSTICK :
1508        {
1509          sprintf(lcdstr,"Joystick        ");
1510          break;
1511        }
1512        case SETUP_PIN :
1513        {
1514          sprintf(lcdstr,"PIN             ");
1515          break;
1516        }
1517        case SETUP_BACKLIGHT :
1518        {
1519          ledvo_trig=&lcd.led.trig;
1520          sprintf(lcdstr,"LCD-Backlight: %u",*ledvo_trig);
1521          break;
1522        }
```

```
1523        case SETUP_CONTRAST :
1524        {
1525          ledvo_trig=&lcd.vo.trig;
1526          sprintf(lcdstr,"LCD-Contrast:  %u",*ledvo_trig);
1527          break;
1528        }
1529      }  // switch (*setupitem)
1530
1531      // Neuen Setup-Eintrag anzeigen
1532      lcd_print(1,2,lcdstr);
1533
1534      // Warten bis Taste losgelassen wurde
1535      while (key16.key!=0) {}
1536
1537    }  // setup_toggle
1538
1539    /* Setup-Modus */
1540    void setup (void)
1541    {
1542    unsigned char setupitem;
1543
1544      setupitem=0;
1545
1546      lcd_clear();
1547      lcd_print(1,1,"Setup:");
1548      lcd_print(1,2,"Program");
1549
1550      // Warten bis Taste losgelassen wurde
1551      while (key16.key!=0) {}
1552
1553      while (key16.key!=KEY_CANCEL)
1554      {
1555        switch (key16.key)
1556        {
1557          // Setup-Eintrag wechseln
1558          case KEY_SETUP :
1559          {
1560            setup_toggle(&setupitem);
1561            break;
1562          }
1563
1564          // Setup-Eintrag auswählen
1565          case KEY_OK :
1566          {
1567            switch (setupitem)
1568            {
1569              case SETUP_PROGRAM :
1570              {
1571                program_keyset();
1572                break;
1573              }
1574              case SETUP_JOYSTICK :
1575              {
1576                setup_joystick();
1577                break;
1578              }
1579              case SETUP_PIN :
1580              {
1581                setup_pin();
1582                break;
1583              }
```

```
1584            case SETUP_BACKLIGHT :
1585            case SETUP_CONTRAST :
1586            {
1587              setup_backlight_contrast(setupitem);
1588              break;
1589            }
1590
1591          }  //switch (setupitem)
1592        }  //case KEY_OK
1593
1594        break;
1595
1596      }  // switch (key16.key)
1597    }  // while (key16.key!=KEY_CANCEL)
1598
1599    show_set();
1600
1601    // Warten bis Taste losgelassen wurde
1602    while (key16.key!=0) {}
1603
1604  }  // setup
1605
1606  /* Interrupt-Routine zum Einlesen der PS/2-Tastatur */
1607  SIGNAL (SIG_INTERRUPT0)
1608  {
1609
1610    // Fallende Flanke
1611    if (!keyat.clockedge)
1612    {
1613      // Bits 1 bis 8 enthalten die Daten
1614      if ((keyat.bitcount<11) && (keyat.bitcount>2))
1615      {
1616        keyat.data=(keyat.data>>1);
1617        if (PS2_IN & (1<<AT_KEY_DATA))
1618        {
1619          keyat.data=keyat.data | 0x80;
1620        }
1621      }  // if ((keyat.bitcount<11) && (keyat.bitcount>2))
1622
1623      // reagiere das nächste mal auf die steigende Flanke
1624      MCUCR=MCUCR | (1<<ISC00);
1625      keyat.clockedge=1;
1626    }  // if (!keyat.clockedge)
1627
1628    // Steigende Flanke
1629    else
1630    {
1631      // reagiere das nächste mal auf die fallende Flanke
1632      MCUCR=MCUCR & ~(1<<ISC00);
1633      keyat.clockedge=0;
1634      keyat.bitcount--;
1635      if (keyat.bitcount==0)
1636      {
1637        keyat.bitcount=11;
1638        keyat.finish=1;
1639      }
1640    }  // if (keyat.clockedge)
1641
1642  }  // SIGNAL (SIG_INTERRUPT0)
1643
```

```
1644  /* Interrupt-Routine zur Steuerung der LCD-Hintergrundbeleuchtung und
1645     des -Kontrasts */
1646  SIGNAL (SIG_OVERFLOW0)
1647  {
1648
1649    if (lcd.led.count==0)
1650    {
1651      LCD_PORT=LCD_PORT & ~(1<<LCD_LED);
1652    }
1653    lcd.led.count++;
1654    if (lcd.led.count==(lcd.led.trig+1))
1655    {
1656      LCD_PORT=LCD_PORT | (1<<LCD_LED);
1657      lcd.led.count=0;
1658    }
1659
1660    if (lcd.vo.count==0)
1661    {
1662      LCD_PORT=LCD_PORT | (1<<LCD_VO);
1663    }
1664    lcd.vo.count++;
1665    if (lcd.vo.count==(lcd.vo.trig+1))
1666    {
1667      LCD_PORT=LCD_PORT & ~(1<<LCD_VO);
1668      lcd.vo.count=0;
1669    }
1670
1671    timer0count++;
1672
1673  }  // SIGNAL (SIG_OVERFLOW0)
1674
1675  /* Interrupt-Routine zur Abfrage der 16er-Tastatur und der Joystick-Knöpfe */
1676  INTERRUPT (SIG_OVERFLOW2)
1677  {
1678  static unsigned char actval,i;
1679
1680    timer2count++;
1681
1682    // 156 = ca. 20ms
1683    if (timer2count==debounce)
1684    {
1685      // 16er-Tastatur abfragen
1686      actval=key16_scan();
1687
1688      // Taste entprellt?
1689      if (actval==key16.oldkey)
1690      {
1691        // aktuelle Taste ungleich voriger Taste?
1692        if (actval!=key16.prevkey)
1693        {
1694          key16.repeat=repeat_start;
1695          key16.longpress=0;
1696        }
1697
1698        // aktuelle Taste gleich voriger Taste?
1699        else
1700        {
1701          key16.repeat--;
1702          if (key16.repeat==0)
1703          {
1704            key16.repeat=repeat_next;
```

```
1705            if (actval!=0)
1706            {
1707              // aktuelle Taste wurde länger gedrückt
1708              key16.longpress=1;
1709            }
1710          }  // if (key16.repeat==0)
1711        }  // if (actval==key16.prevkey)
1712        key16.key=actval;
1713        key16.prevkey=actval;
1714
1715      }  // if (actval==key16.oldkey)
1716
1717      key16.oldkey=actval;
1718
1719      if (!key16.key)
1720      {
1721        actval=(JOY_IN>>JOY_BUTTON_0) & 0x0F;
1722
1723        // Joystick-Knöpfe entprellt?
1724        if (actval==joy.oldbuttons)
1725        {
1726          // aktuelle Joystick-Knöpfe ungleich vorigen Joystick-Knöpfen?
1727          if (actval!=joy.prevbuttons)
1728          {
1729            joy.btn_repeat=repeat_start;
1730            joy.btn_longpress=0;
1731          }
1732
1733          // aktuelle Joystick-Knöpfe gleich vorigen Joystick-Knöpfen?
1734          else
1735          {
1736            joy.btn_repeat--;
1737            if (joy.btn_repeat==0)
1738            {
1739              joy.btn_repeat=repeat_next;
1740              if (actval!=0x0F)
1741              {
1742                // aktuelle Joystick-Knöpfe wurden länger gedrückt
1743                joy.btn_longpress=1;
1744              }
1745            }  // if (joy.btn_repeat==0)
1746          }  // if (actval==joy.prevbuttons)
1747          for (i=0;i<=3;i++)
1748          {
1749            joy.button[i]=(actval>>i) & 1;
1750          }
1751          joy.prevbuttons=actval;
1752
1753        }  // if (actval==joy.oldbuttons)
1754
1755        joy.oldbuttons=actval;
1756
1757        if (actval==0x0F)
1758        {
1759          for (i=0;i<=3 && actval==0x0F;i++)
1760          {
1761            if (joy.poti[i]>20 && (joy.poti[i]<joy.cal[i].trig_mid_min ||
1762                                   joy.poti[i]>joy.cal[i].trig_mid_max))
1763            {
1764              // Speichere die benutzte Joystick-Achse und Joystick-Richtung
1765              if (joy.poti[i]>joy.cal[i].trig_mid_max)
1766              {
1767                actval=i+1;
1768              }
```

```
1769              else
1770              {
1771                actval=i+5;
1772              }
1773            }  // if (joy.poti[i]>20 && (joy.poti[i]<joy.cal[i].trig_mid_min ||
1774               //                       joy.poti[i]>joy.cal[i].trig_mid_max))
1775          }  // for (i=0;i<=3 && actval==0x0F;i++)
1776
1777          // Keine Joystick-Achse über Schaltpunkt bewegt
1778          if (actval==0x0F)
1779          {
1780            actval=0;
1781            joy.poti_longpress=0;
1782            joy.potiaxis=0;
1783            joy.prevpotiaxis=0;
1784            joy.oldpotiaxis=0;
1785          }  // if (actval==0x0F)
1786
1787          // Joystick-Achse "entprellt" (eher für Joypad)
1788          if (actval==joy.oldpotiaxis)
1789          {
1790            // aktuelle Joystick-Achse ungleich voriger Joystick-Achse?
1791            if (actval!=joy.prevpotiaxis)
1792            {
1793              joy.poti_repeat=repeat_start;
1794              joy.poti_longpress=0;
1795            }
1796
1797            // aktuelle Joystick-Achse gleich voriger Joystick-Achse?
1798            else
1799            {
1800              joy.poti_repeat--;
1801              if (joy.poti_repeat==0)
1802              {
1803                joy.poti_repeat=repeat_next;
1804                if (actval!=0)
1805                {
1806                  // aktuelle Joystick-Achse wurde länger betätigt
1807                  joy.poti_longpress=1;
1808                }
1809              }  // if (joy.poti_repeat==0)
1810            }  // if (actval==joy.prevpotiaxis)
1811            joy.potiaxis=actval;
1812            joy.prevpotiaxis=actval;
1813
1814          }  // if (actval==joy.oldpotiaxis)
1815
1816          joy.oldpotiaxis=actval;
1817
1818        }  // if (actval==0x0F)
1819      }  // if (!key16.key)
1820
1821      timer2count=0;
1822
1823    }  // if (timer2count==debounce)
1824
1825  }  // INTERRUPT (SIG_OVERFLOW2)
1826
```

```
1827   /* Interrupt-Routine zur Abfrage der Joystick-Achsen */
1828   SIGNAL (SIG_ADC)
1829   {
1830
1831     // Ergebnis der AD-Wandlung speichern
1832     joy.poti[joy.potipointer]=ADCH;
1833
1834     // zu nächstem AD-Kanal schalten
1835     joy.potipointer++;
1836     if (joy.potipointer==4)
1837     {
1838       joy.potipointer=0;
1839     }
1840
1841     // Einzulesendes Joystick-Poti bestimmen
1842     ADMUX=(ADMUX & 0xE0) | (JOY_POTI_0+joy.potipointer);
1843
1844     // AD-Wandlung starten
1845     ADCSRA=ADCSRA | (1<<ADSC);
1846
1847   }  // SIGNAL (SIG_ADC)
1848
1849   // Hauptprogramm
1850   int main (void)
1851   {
1852   unsigned char i,buffer[9],joyfunc;
1853
1854     // Gerät initialisieren
1855     lcd_init();
1856     ps2_init();
1857     key16_init();
1858     joy_init();
1859     ext_eeprom_init();
1860     uc_init();
1861
1862     // Wert für LCD-Hintergrundbeleuchtung aus externem EEPROM lesen
1863     ext_eeprom_read_buffer(BACKLIGHT_ADDRESS,buffer);
1864     if (buffer[0]==0xFF)
1865     {
1866       buffer[0]=9;
1867     }
1868     lcd.led.trig=buffer[0];
1869
1870     // Wert für LCD-Kontrast aus externem EEPROM lesen
1871     ext_eeprom_read_buffer(CONTRAST_ADDRESS,buffer);
1872     if (buffer[0]==0xFF)
1873     {
1874       buffer[0]=8;
1875     }
1876     lcd.vo.trig=buffer[0];
1877
1878     // Joystick-Kalibrierwerte aus externem EEPROM lesen
1879     ext_eeprom_read_buffer(JOYCAL_ADDRESS,buffer);
1880     for (i=0;i<=3;i++)
1881     {
1882       joy.cal[i].trig_mid_max=buffer[i*2];
1883       joy.cal[i].trig_mid_min=buffer[i*2+1];
1884     }
1885
1886     set_mode=SET_KEY;
1887     keyset=0;
1888     joyset=0;
1889     passwordset=0;
1890
```

```
1891      lcd_print(3,1,"ProggyKeyJoy");
1892      lcd_print(5,2,"SW: 0.89");
1893      wait_ms(1000);
1894
1895      show_set();
1896
1897      while (1)
1898      {
1899        switch (key16.key)
1900        {
1901          // Bestimmen des Keyset
1902          case KEY_KEYSET :
1903          {
1904            set_mode=SET_KEY;
1905            set_keyset(9,1,&keyset);
1906            break;
1907          }
1908
1909          // Bestimmen des Joystickset
1910          case KEY_JOYSET :
1911          {
1912            set_mode=SET_JOY;
1913            set_keyset(9,1,&joyset);
1914            break;
1915          }
1916
1917          // Bestimmen des Passordset
1918          case KEY_PASSWORD :
1919          {
1920            set_keypasswordset();
1921            break;
1922          }
1923
1924          // Aufrufen des Setup
1925          case KEY_SETUP :
1926          {
1927            setup();
1928            break;
1929          }
1930        }  // switch (key16.key)
1931
1932        // Tasten-Sequenz aus Keyset oder Passwordset an PC senden
1933        if (set_mode!=SET_JOY)
1934        {
1935          // Ziffern-Taste auf 16er-Tastatur gedrückt?
1936          if (digit16[key16.key]!=-1)
1937          {
1938            // Tasten-Sequenz an PC senden
1939            send_to_pc(digit16[key16.key]);
1940
1941            // Warten bis Taste losgelassen wurde oder länger gedrückt ist
1942            while (key16.key!=0 && !key16.longpress) {}
1943
1944          }  // if (digit16[key16.key]!=-1)
1945        }  // if (set_mode!=SET_JOY)
1946
1947        // Tasten-Sequenz aus Joystickset an PC senden
1948        else
1949        {
1950          // Bestimmen der Joystick-Funktion
1951          joyfunc=0xFF;
1952          for (i=0;i<=3 && joyfunc==0xFF;i++)
1953          {
```

```
1954            // Joystick-Knopf
1955            if (joy.button[i]==0)
1956            {
1957              joyfunc=i;
1958            }
1959          }  // for (i=0;i<=3 && joyfunc==0xFF;i++)
1960          if (joyfunc==0xFF && joy.potiaxis>0)
1961          {
1962            joyfunc=joy.potiaxis+3;
1963          }
1964
1965          if (joyfunc!=0xFF)
1966          {
1967            // Tasten-Sequenz an PC senden
1968            send_to_pc(joyfunc);
1969            i--;
1970
1971            // Warten bis Joystick-Funktion (Knopf oder Poti) losgelassen wurde
1972            // oder länger gedrückt ist
1973            while (joyfunc<=3 && joy.button[i]==0 && !joy.btn_longpress) {}
1974            while (joyfunc>=4 && joyfunc<=7 && joy.potiaxis!=0 &&
1975                   !joy.poti_longpress) {}
1976            while (joyfunc>=8 && joyfunc<=11 && joy.potiaxis!=0 &&
1977                   !joy.poti_longpress) {}
1978
1979          }  // if (joyfunc!=0xFF)
1980        }  // if (set_mode==SET_JOY)
1981
1982      }  // while (1)
1983
1984      return 1;
1985
1986    }  // main
```

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
0	200	0000	00C8	Keyset	0	0
201	401	00C9	0191	Keyset	0	1
402	602	0192	025A	Keyset	0	2
603	803	025B	0323	Keyset	0	3
804	1004	0324	03EC	Keyset	0	4
1005	1205	03ED	04B5	Keyset	0	5
1206	1406	04B6	057E	Keyset	0	6
1407	1607	057F	0647	Keyset	0	7
1608	1808	0648	0710	Keyset	0	8
1809	2009	0711	07D9	Keyset	0	9
2010	2210	07DA	08A2	Keyset	1	0
2211	2411	08A3	096B	Keyset	1	1
2412	2612	096C	0A34	Keyset	1	2
2613	2813	0A35	0AFD	Keyset	1	3
2814	3014	0AFE	0BC6	Keyset	1	4
3015	3215	0BC7	0C8F	Keyset	1	5
3216	3416	0C90	0D58	Keyset	1	6
3417	3617	0D59	0E21	Keyset	1	7
3618	3818	0E22	0EEA	Keyset	1	8
3819	4019	0EEB	0FB3	Keyset	1	9
4020	4220	0FB4	107C	Keyset	2	0
4221	4421	107D	1145	Keyset	2	1
4422	4622	1146	120E	Keyset	2	2
4623	4823	120F	12D7	Keyset	2	3
4824	5024	12D8	13A0	Keyset	2	4
5025	5225	13A1	1469	Keyset	2	5
5226	5426	146A	1532	Keyset	2	6
5427	5627	1533	15FB	Keyset	2	7
5628	5828	15FC	16C4	Keyset	2	8
5829	6029	16C5	178D	Keyset	2	9
6030	6230	178E	1856	Keyset	3	0
6231	6431	1857	191F	Keyset	3	1
6432	6632	1920	19E8	Keyset	3	2
6633	6833	19E9	1AB1	Keyset	3	3
6834	7034	1AB2	1B7A	Keyset	3	4
7035	7235	1B7B	1C43	Keyset	3	5
7236	7436	1C44	1D0C	Keyset	3	6
7437	7637	1D0D	1DD5	Keyset	3	7
7638	7838	1DD6	1E9E	Keyset	3	8
7839	8039	1E9F	1F67	Keyset	3	9
8040	8240	1F68	2030	Keyset	4	0
8241	8441	2031	20F9	Keyset	4	1
8442	8642	20FA	21C2	Keyset	4	2
8643	8843	21C3	228B	Keyset	4	3
8844	9044	228C	2354	Keyset	4	4
9045	9245	2355	241D	Keyset	4	5
9246	9446	241E	24E6	Keyset	4	6
9447	9647	24E7	25AF	Keyset	4	7
9648	9848	25B0	2678	Keyset	4	8
9849	10049	2679	2741	Keyset	4	9
10050	10250	2742	280A	Keyset	5	0
10251	10451	280B	28D3	Keyset	5	1
10452	10652	28D4	299C	Keyset	5	2
10653	10853	299D	2A65	Keyset	5	3
10854	11054	2A66	2B2E	Keyset	5	4
11055	11255	2B2F	2BF7	Keyset	5	5
11256	11456	2BF8	2CC0	Keyset	5	6
11457	11657	2CC1	2D89	Keyset	5	7

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
11658	11858	2D8A	2E52	Keyset	5	8
11859	12059	2E53	2F1B	Keyset	5	9
12060	12260	2F1C	2FE4	Keyset	6	0
12261	12461	2FE5	30AD	Keyset	6	1
12462	12662	30AE	3176	Keyset	6	2
12663	12863	3177	323F	Keyset	6	3
12864	13064	3240	3308	Keyset	6	4
13065	13265	3309	33D1	Keyset	6	5
13266	13466	33D2	349A	Keyset	6	6
13467	13667	349B	3563	Keyset	6	7
13668	13868	3564	362C	Keyset	6	8
13869	14069	362D	36F5	Keyset	6	9
14070	14270	36F6	37BE	Keyset	7	0
14271	14471	37BF	3887	Keyset	7	1
14472	14672	3888	3950	Keyset	7	2
14673	14873	3951	3A19	Keyset	7	3
14874	15074	3A1A	3AE2	Keyset	7	4
15075	15275	3AE3	3BAB	Keyset	7	5
15276	15476	3BAC	3C74	Keyset	7	6
15477	15677	3C75	3D3D	Keyset	7	7
15678	15878	3D3E	3E06	Keyset	7	8
15879	16079	3E07	3ECF	Keyset	7	9
16080	16280	3ED0	3F98	Keyset	8	0
16281	16481	3F99	4061	Keyset	8	1
16482	16682	4062	412A	Keyset	8	2
16683	16883	412B	41F3	Keyset	8	3
16884	17084	41F4	42BC	Keyset	8	4
17085	17285	42BD	4385	Keyset	8	5
17286	17486	4386	444E	Keyset	8	6
17487	17687	444F	4517	Keyset	8	7
17688	17888	4518	45E0	Keyset	8	8
17889	18089	45E1	46A9	Keyset	8	9
18090	18290	46AA	4772	Keyset	9	0
18291	18491	4773	483B	Keyset	9	1
18492	18692	483C	4904	Keyset	9	2
18693	18893	4905	49CD	Keyset	9	3
18894	19094	49CE	4A96	Keyset	9	4
19095	19295	4A97	4B5F	Keyset	9	5
19296	19496	4B60	4C28	Keyset	9	6
19497	19697	4C29	4CF1	Keyset	9	7
19698	19898	4CF2	4DBA	Keyset	9	8
19899	20099	4DBB	4E83	Keyset	9	9
20100	20300	4E84	4F4C	Password	0	0
20301	20501	4F4D	5015	Password	0	1
20502	20702	5016	50DE	Password	0	2
20703	20903	50DF	51A7	Password	0	3
20904	21104	51A8	5270	Password	0	4
21105	21305	5271	5339	Password	0	5
21306	21506	533A	5402	Password	0	6
21507	21707	5403	54CB	Password	0	7
21708	21908	54CC	5594	Password	0	8
21909	22109	5595	565D	Password	0	9
22110	22310	565E	5726	Password	1	0
22311	22511	5727	57EF	Password	1	1
22512	22712	57F0	58B8	Password	1	2
22713	22913	58B9	5981	Password	1	3
22914	23114	5982	5A4A	Password	1	4
23115	23315	5A4B	5B13	Password	1	5

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
23316	23516	5B14	5BDC	Password	1	6
23517	23717	5BDD	5CA5	Password	1	7
23718	23918	5CA6	5D6E	Password	1	8
23919	24119	5D6F	5E37	Password	1	9
24120	24320	5E38	5F00	Password	2	0
24321	24521	5F01	5FC9	Password	2	1
24522	24722	5FCA	6092	Password	2	2
24723	24923	6093	615B	Password	2	3
24924	25124	615C	6224	Password	2	4
25125	25325	6225	62ED	Password	2	5
25326	25526	62EE	63B6	Password	2	6
25527	25727	63B7	647F	Password	2	7
25728	25928	6480	6548	Password	2	8
25929	26129	6549	6611	Password	2	9
26130	26330	6612	66DA	Password	3	0
26331	26531	66DB	67A3	Password	3	1
26532	26732	67A4	686C	Password	3	2
26733	26933	686D	6935	Password	3	3
26934	27134	6936	69FE	Password	3	4
27135	27335	69FF	6AC7	Password	3	5
27336	27536	6AC8	6B90	Password	3	6
27537	27737	6B91	6C59	Password	3	7
27738	27938	6C5A	6D22	Password	3	8
27939	28139	6D23	6DEB	Password	3	9
28140	28340	6DEC	6EB4	Password	4	0
28341	28541	6EB5	6F7D	Password	4	1
28542	28742	6F7E	7046	Password	4	2
28743	28943	7047	710F	Password	4	3
28944	29144	7110	71D8	Password	4	4
29145	29345	71D9	72A1	Password	4	5
29346	29546	72A2	736A	Password	4	6
29547	29747	736B	7433	Password	4	7
29748	29948	7434	74FC	Password	4	8
29949	30149	74FD	75C5	Password	4	9
30150	30350	75C6	768E	Password	5	0
30351	30551	768F	7757	Password	5	1
30552	30752	7758	7820	Password	5	2
30753	30953	7821	78E9	Password	5	3
30954	31154	78EA	79B2	Password	5	4
31155	31355	79B3	7A7B	Password	5	5
31356	31556	7A7C	7B44	Password	5	6
31557	31757	7B45	7C0D	Password	5	7
31758	31958	7C0E	7CD6	Password	5	8
31959	32159	7CD7	7D9F	Password	5	9
32160	32360	7DA0	7E68	Password	6	0
32361	32561	7E69	7F31	Password	6	1
32562	32762	7F32	7FFA	Password	6	2
32763	32963	7FFB	80C3	Password	6	3
32964	33164	80C4	818C	Password	6	4
33165	33365	818D	8255	Password	6	5
33366	33566	8256	831E	Password	6	6
33567	33767	831F	83E7	Password	6	7
33768	33968	83E8	84B0	Password	6	8
33969	34169	84B1	8579	Password	6	9
34170	34370	857A	8642	Password	7	0
34371	34571	8643	870B	Password	7	1
34572	34772	870C	87D4	Password	7	2
34773	34973	87D5	889D	Password	7	3

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
34974	35174	889E	8966	Password	7	4
35175	35375	8967	8A2F	Password	7	5
35376	35576	8A30	8AF8	Password	7	6
35577	35777	8AF9	8BC1	Password	7	7
35778	35978	8BC2	8C8A	Password	7	8
35979	36179	8C8B	8D53	Password	7	9
36180	36380	8D54	8E1C	Password	8	0
36381	36581	8E1D	8EE5	Password	8	1
36582	36782	8EE6	8FAE	Password	8	2
36783	36983	8FAF	9077	Password	8	3
36984	37184	9078	9140	Password	8	4
37185	37385	9141	9209	Password	8	5
37386	37586	920A	92D2	Password	8	6
37587	37787	92D3	939B	Password	8	7
37788	37988	939C	9464	Password	8	8
37989	38189	9465	952D	Password	8	9
38190	38390	952E	95F6	Password	9	0
38391	38591	95F7	96BF	Password	9	1
38592	38792	96C0	9788	Password	9	2
38793	38993	9789	9851	Password	9	3
38994	39194	9852	991A	Password	9	4
39195	39395	991B	99E3	Password	9	5
39396	39596	99E4	9AAC	Password	9	6
39597	39797	9AAD	9B75	Password	9	7
39798	39998	9B76	9C3E	Password	9	8
39999	40199	9C3F	9D07	Password	9	9
40200	40400	9D08	9DD0	Joystick	0	Knopf 1
40401	40601	9DD1	9E99	Joystick	0	Knopf 2
40602	40802	9E9A	9F62	Joystick	0	Knopf 3
40803	41003	9F63	A02B	Joystick	0	Knopf 4
41004	41204	A02C	A0F4	Joystick	0	Links 1
41205	41405	A0F5	A1BD	Joystick	0	Oben 1
41406	41606	A1BE	A286	Joystick	0	Links 2
41607	41807	A287	A34F	Joystick	0	Oben 2
41808	42008	A350	A418	Joystick	0	Rechts 1
42009	42209	A419	A4E1	Joystick	0	Unten 1
42210	42410	A4E2	A5AA	Joystick	0	Rechts 2
42411	42611	A5AB	A673	Joystick	0	Unten 2
42612	42812	A674	A73C	Joystick	1	Knopf 1
42813	43013	A73D	A805	Joystick	1	Knopf 2
43014	43214	A806	A8CE	Joystick	1	Knopf 3
43215	43415	A8CF	A997	Joystick	1	Knopf 4
43416	43616	A998	AA60	Joystick	1	Links 1
43617	43817	AA61	AB29	Joystick	1	Oben 1
43818	44018	AB2A	ABF2	Joystick	1	Links 2
44019	44219	ABF3	ACBB	Joystick	1	Oben 2
44220	44420	ACBC	AD84	Joystick	1	Rechts 1
44421	44621	AD85	AE4D	Joystick	1	Unten 1
44622	44822	AE4E	AF16	Joystick	1	Rechts 2
44823	45023	AF17	AFDF	Joystick	1	Unten 2
45024	45224	AFE0	B0A8	Joystick	2	Knopf 1
45225	45425	B0A9	B171	Joystick	2	Knopf 2
45426	45626	B172	B23A	Joystick	2	Knopf 3
45627	45827	B23B	B303	Joystick	2	Knopf 4
45828	46028	B304	B3CC	Joystick	2	Links 1
46029	46229	B3CD	B495	Joystick	2	Oben 1
46230	46430	B496	B55E	Joystick	2	Links 2
46431	46631	B55F	B627	Joystick	2	Oben 2

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
46632	46832	B628	B6F0	Joystick	2	Rechts 1
46833	47033	B6F1	B7B9	Joystick	2	Unten 1
47034	47234	B7BA	B882	Joystick	2	Rechts 2
47235	47435	B883	B94B	Joystick	2	Unten 2
47436	47636	B94C	BA14	Joystick	3	Knopf 1
47637	47837	BA15	BADD	Joystick	3	Knopf 2
47838	48038	BADE	BBA6	Joystick	3	Knopf 3
48039	48239	BBA7	BC6F	Joystick	3	Knopf 4
48240	48440	BC70	BD38	Joystick	3	Links 1
48441	48641	BD39	BE01	Joystick	3	Oben 1
48642	48842	BE02	BECA	Joystick	3	Links 2
48843	49043	BECB	BF93	Joystick	3	Oben 2
49044	49244	BF94	C05C	Joystick	3	Rechts 1
49245	49445	C05D	C125	Joystick	3	Unten 1
49446	49646	C126	C1EE	Joystick	3	Rechts 2
49647	49847	C1EF	C2B7	Joystick	3	Unten 2
49848	50048	C2B8	C380	Joystick	4	Knopf 1
50049	50249	C381	C449	Joystick	4	Knopf 2
50250	50450	C44A	C512	Joystick	4	Knopf 3
50451	50651	C513	C5DB	Joystick	4	Knopf 4
50652	50852	C5DC	C6A4	Joystick	4	Links 1
50853	51053	C6A5	C76D	Joystick	4	Oben 1
51054	51254	C76E	C836	Joystick	4	Links 2
51255	51455	C837	C8FF	Joystick	4	Oben 2
51456	51656	C900	C9C8	Joystick	4	Rechts 1
51657	51857	C9C9	CA91	Joystick	4	Unten 1
51858	52058	CA92	CB5A	Joystick	4	Rechts 2
52059	52259	CB5B	CC23	Joystick	4	Unten 2
52260	52460	CC24	CCEC	Joystick	5	Knopf 1
52461	52661	CCED	CDB5	Joystick	5	Knopf 2
52662	52862	CDB6	CE7E	Joystick	5	Knopf 3
52863	53063	CE7F	CF47	Joystick	5	Knopf 4
53064	53264	CF48	D010	Joystick	5	Links 1
53265	53465	D011	D0D9	Joystick	5	Oben 1
53466	53666	D0DA	D1A2	Joystick	5	Links 2
53667	53867	D1A3	D26B	Joystick	5	Oben 2
53868	54068	D26C	D334	Joystick	5	Rechts 1
54069	54269	D335	D3FD	Joystick	5	Unten 1
54270	54470	D3FE	D4C6	Joystick	5	Rechts 2
54471	54671	D4C7	D58F	Joystick	5	Unten 2
54672	54872	D590	D658	Joystick	6	Knopf 1
54873	55073	D659	D721	Joystick	6	Knopf 2
55074	55274	D722	D7EA	Joystick	6	Knopf 3
55275	55475	D7EB	D8B3	Joystick	6	Knopf 4
55476	55676	D8B4	D97C	Joystick	6	Links 1
55677	55877	D97D	DA45	Joystick	6	Oben 1
55878	56078	DA46	DB0E	Joystick	6	Links 2
56079	56279	DB0F	DBD7	Joystick	6	Oben 2
56280	56480	DBD8	DCA0	Joystick	6	Rechts 1
56481	56681	DCA1	DD69	Joystick	6	Unten 1
56682	56882	DD6A	DE32	Joystick	6	Rechts 2
56883	57083	DE33	DEFB	Joystick	6	Unten 2
57084	57284	DEFC	DFC4	Joystick	7	Knopf 1
57285	57485	DFC5	E08D	Joystick	7	Knopf 2
57486	57686	E08E	E156	Joystick	7	Knopf 3
57687	57887	E157	E21F	Joystick	7	Knopf 4
57888	58088	E220	E2E8	Joystick	7	Links 1
58089	58289	E2E9	E3B1	Joystick	7	Oben 1

Dezimal		Hexadezimal				
Start	*Stop*	*Start*	*Stop*	*Keyset / Inhalt*	*Nr.*	*Taste / Joystick-Funktion / Wert*
58290	58490	E3B2	E47A	Joystick	7	Links 2
58491	58691	E47B	E543	Joystick	7	Oben 2
58692	58892	E544	E60C	Joystick	7	Rechts 1
58893	59093	E60D	E6D5	Joystick	7	Unten 1
59094	59294	E6D6	E79E	Joystick	7	Rechts 2
59295	59495	E79F	E867	Joystick	7	Unten 2
59496	59696	E868	E930	Joystick	8	Knopf 1
59697	59897	E931	E9F9	Joystick	8	Knopf 2
59898	60098	E9FA	EAC2	Joystick	8	Knopf 3
60099	60299	EAC3	EB8B	Joystick	8	Knopf 4
60300	60500	EB8C	EC54	Joystick	8	Links 1
60501	60701	EC55	ED1D	Joystick	8	Oben 1
60702	60902	ED1E	EDE6	Joystick	8	Links 2
60903	61103	EDE7	EEAF	Joystick	8	Oben 2
61104	61304	EEB0	EF78	Joystick	8	Rechts 1
61305	61505	EF79	F041	Joystick	8	Unten 1
61506	61706	F042	F10A	Joystick	8	Rechts 2
61707	61907	F10B	F1D3	Joystick	8	Unten 2
61908	62108	F1D4	F29C	Joystick	9	Knopf 1
62109	62309	F29D	F365	Joystick	9	Knopf 2
62310	62510	F366	F42E	Joystick	9	Knopf 3
62511	62711	F42F	F4F7	Joystick	9	Knopf 4
62712	62912	F4F8	F5C0	Joystick	9	Links 1
62913	63113	F5C1	F689	Joystick	9	Oben 1
63114	63314	F68A	F752	Joystick	9	Links 2
63315	63515	F753	F81B	Joystick	9	Oben 2
63516	63716	F81C	F8E4	Joystick	9	Rechts 1
63717	63917	F8E5	F9AD	Joystick	9	Unten 1
63918	64118	F9AE	FA76	Joystick	9	Rechts 2
64119	64319	FA77	FB3F	Joystick	9	Unten 2
64320	64325	FB40	FB45	PIN		
64326	64327	FB46	FB47	LCD-Beleuchtung		
64328	64329	FB48	FB49	LCD-Kontrast		
64330	64330	FB4A	FB4A	Joystick-Kalibrierwerte		trig_mid_max X1
64331	64331	FB4B	FB4B	Joystick-Kalibrierwerte		trig_mid_min X1
64332	64332	FB4C	FB4C	Joystick-Kalibrierwerte		trig_mid_max Y1
64333	64333	FB4D	FB4D	Joystick-Kalibrierwerte		trig_mid_min Y1
64334	64334	FB4E	FB4E	Joystick-Kalibrierwerte		trig_mid_max X2
64335	64335	FB4F	FB4F	Joystick-Kalibrierwerte		trig_mid_min X2
64336	64336	FB50	FB50	Joystick-Kalibrierwerte		trig_mid_max Y2
64337	64337	FB51	FB51	Joystick-Kalibrierwerte		trig_mid_min Y2
64338	64338	FB52	FB52	Joystick-Kalibrierwerte		0

ProggyKeyJoy

Bedienungsanleitung

Andreas Pernau
Ratiborstraße 31
D-38124 Braunschweig

ProggyKeyJoy@arcor.de

Inhaltsverzeichnis

1. Vorwort

Mit ProggyKeyJoy sind Sie in der Lage Tastenfolgen und –Kombinationen zu speichern und bei Bedarf wieder abzurufen. Dies ist zum Beispiel dort hilfreich, wo man immer wieder die gleichen Texte (z.B. Adressdaten) eingeben muss oder für die Bedienung eines Programms komplizierte Tastenkombinationen nötig sind.

In einem Bereich sind diese Tastenfolgen durch eine PIN geschützt, so dass man dort seine Passwörter oder ähnliches hinterlegen kann.

Darüber hinaus lässt sich auch ein handelsüblicher Joystick mit 15-poligen Sub-D-Anschluss anschließen, dessen Funktionen sich dann genauso mit Tastenfolgen belegen lassen. Damit können Sie dann zum Beispiel Spiele, die sich nur mit der Tastatur steuern lassen trotzdem mit einem Joystick spielen. Ein neuer High-Score ist Ihnen damit so gut wie sicher.

Für jede der drei obigen Betriebsarten lassen sich zehn Sätze mit jeweils zehn (zwölf beim Joystick) Tastenfolgen speichern. Jede Tastenfolge kann dabei aus bis zu 66 Tasten bestehen.

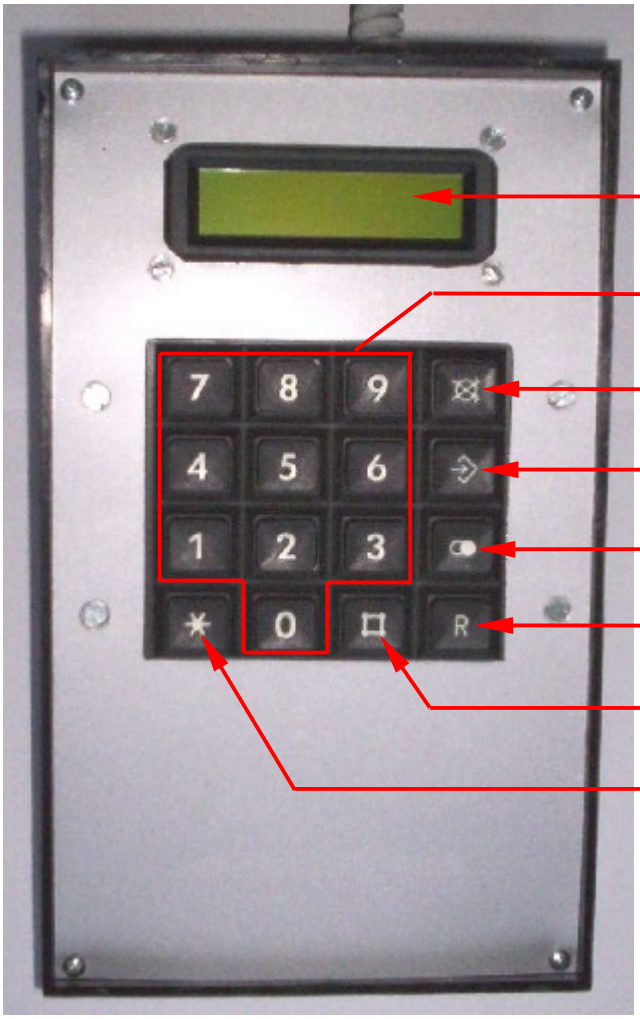
Der Betrieb und Anschluss des Geräts ist dabei kinderleicht. Eine Treiberinstallation ist nicht notwendig. Der Betrieb ist an allen handelsüblichen PCs mit PS/2-Schnittstelle, unabhängig vom verwendeten Betriebsystem, möglich. Für den Anschluss an ältere Rechner bzw. Tastaturen mit 5-poligen DIN-Anschluss werden im Handel Adapter angeboten.

2. Allgemeine Sicherheitshinweise

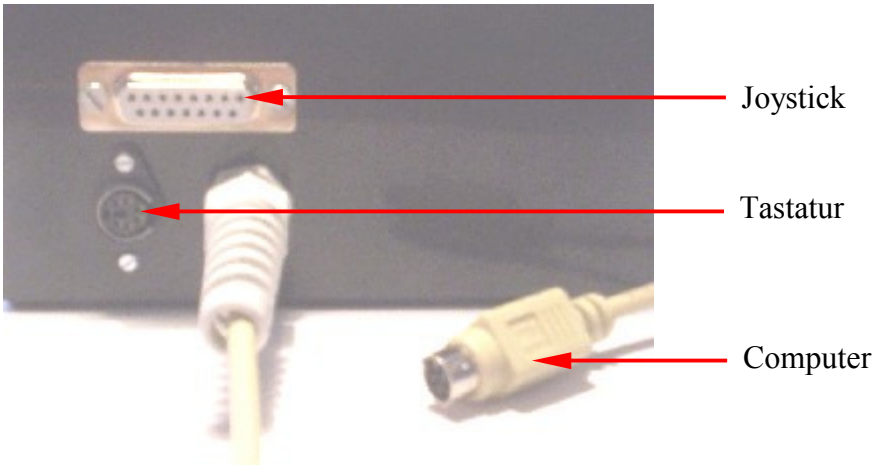
- Lesen Sie vor Inbetriebnahme dieses Gerätes sorgfältig die Bedienungsanleitung.
- Bitte bewahren Sie diese Anleitung, den Garantieschein, den Kassenbon und nach Möglichkeit den Karton mit Innenverpackung gut auf.
- Prüfen Sie das Kabel und das Gerät regelmäßig auf Schäden hin. Ein Gerät ist bei Schäden jeglicher Art nicht in Betrieb zu nehmen.
- Reparieren Sie das Gerät nicht selbst, sondern suchen Sie bitte einen autorisierten Fachmann auf.
- Halten Sie das Gerät und das Kabel fern von Hitze, direkter Sonneneinstrahlung, Feuchtigkeit, scharfen Kanten und ähnlichem.
- Das Gerät darf auf keinen Fall in Wasser oder andere Flüssigkeiten eingetaucht werden oder damit in Berührung kommen. Benutzen Sie das Gerät nicht mit nassen oder feuchten Händen.
- Das Gerät hat scharfe Kanten und hervorstehende Teile, es besteht Verletzungsgefahr bei Berührung.
- Das Gerät gehört nicht in Kinderhände.
- Gehen Sie vorsichtig mit dem Gerät um, durch Stöße, Schläge oder dem Fall aus bereits geringer Höhe wird es beschädigt.
- Benutzen Sie das Gerät nur für den vorgesehenen Zweck.

3. Übersicht

3.1 Bedien- und Anzeigeelemente



3.2 Anschlüsse



4. Inbetriebnahme

- Computer ausschalten.
- Eine ggf. an den Computer angeschlossene Tastatur entfernen.
- Den 6-poligen Stecker am Ende des Anschlusskabels von ProggyKeyJoy mit dem Tastaturanschluss des Computers verbinden.
- Die Tastatur in die 6-polige Buchse von ProggyKeyJoy stecken.
- Bei Bedarf einen Joystick in die 15-polige Sub-D-Buchse von ProggyKeyJoy stecken.
- Computer einschalten.

5. Bedienung

5.1 Wechseln des Modus und des Satzes

Der aktuelle Modus wird auf der LCD-Anzeige angezeigt. Es kann sich dabei um folgende drei Modi handeln:

- Keyset : Hier kann über eine der Tasten 0 bis 9 von ProggyKeyJoy eine Tastensequenz an den Computer gesendet werden.
- Passwordset : Hier kann ebenso wie bei Keyset über eine der Tasten 0 bis 9 von ProggyKeyJoy eine Tastensequenz an den Computer gesendet werden. Dieser Modus ist zudem über eine 5-stellige PIN abgesichert.
- Joyset : Hier kann über einen ggf. angeschlossenen Joystick eine Tastensequenz über einen Joystick-Hebel bzw. einen Joystick-Knopf an den Computer gesendet werden.

Die Ziffer nach dem Modus gibt an, welcher von jeweils zehn Sätzen gerade eingestellt ist. Zum Wechseln des Modus bzw. Einstellen des Satzes ist die entsprechende Taste auf ProggyKeyJoy zu drücken. Ein Fragezeichen nach dem Modus fordert zur Eingabe einer Ziffer von 0 bis 9 über ProggyKeyJoy auf. Will man von Keyset oder Joyset nach Passwordset wechseln, so wird man zur Eingabe einer 5-stelligen PIN aufgefordert.

5.2 Wiedergabe von Tastatursequenzen

Die Wiedergabe von Tastatursequenzen an den Computer erfolgt in den Modi Keyset und Passwordset indem man eine der Tasten 0 bis 9 auf ProggyKeyJoy drückt.

Im Modus Joyset erfolgt die Wiedergabe indem man den Richtungshebel oder einen Knopf des Joysticks betätigt. Der Richtungshebel ist dabei zu gut der Hälfte des Vollausschlags zu bewegen.

Wird eine der obigen Aktionen, die zur Widergabe führen, länger als eine ½ Sekunde betätigt, so wird ca. fünf mal in der Sekunde eine neue Widergabe durchgeführt, bis die Taste bzw. Joystick-Hebel oder –Knopf losgelassen wurde. Die tatsächliche Widerholrate kann dabei jedoch langsamer als fünf mal in der Sekunde sein, wenn der Computer mit der Verarbeitung von Tastensequenzen (z.B. Umschalttaste) längere Zeit braucht.

5.3 Setup

Im Setup können das Programmieren von ProggyKeyJoy, das Kalibrieren des Joysticks, Ändern der PIN, Einstellen von LCD-Hintergrundbeleuchtung und LCD-Kontrast vorgenommen werden.

Im Normalbetrieb (Anzeige: Keyset, Joyset oder Passwordset gefolgt von einer Ziffer) gelangt man ins Setup-Menü durch Drücken der Taste Setup. Durch wiederholtes Drücken der Taste „Setup“ wechselt man zwischen den einzelnen Setup-Menüeinträgen. Durch Drücken der Taste „OK“ wählt man den gewünschten Menüeintrag aus und durch Drücken der Taste „Cancel“ verlässt man das Setup-Menü und gelangt wieder in den Normalbetrieb.

Die folgenden fünf Anleitungen gehen davon aus, dass man sich im Normalbetrieb befindet.

5.3.1 Programmieren

Zuerst ist ggf. der richtige Modus und der richtige Satz einzustellen (siehe 5.1). Dann ist durch Drücken der Taste „Setup“ das Setup-Menü aufzurufen. Die Anzeige zeigt nun „Setup: Program“ an. Durch Drücken auf die Taste „OK“ gelangt man nun in den Programmiermodus. Die Anzeige wechselt in der ersten Zeile, je nach eingestelltem Modus, zu „Program KS X“ (KS = Keyset), „Program JS X“ (JS = Joyset) oder „Program PS X“ (PS = Passwordset) und in der zweiten Zeile zu „Bytes left: 200“. Das X steht dabei für den jeweils eingestellten Satz von 0 bis 9.

Über die Computer-Tastatur können nun Tastenfolgen und –Kombinationen eingegeben werden. In der zweiten Zeile wird dabei ständig der noch zur Verfügung stehende Speicherplatz angezeigt. Dabei ist zu beachten, dass eine Taste 3 bis 10 Byte Speicher beansprucht.
Der Speicherplatz von 200 Byte reicht also für ca. 66 (= 200 : 3) bis 20 (= 200 : 10) Zeichen aus.

Wünscht man die Eingabe abzubrechen ohne sie zu speichern, so gelangt man durch Drücken der Taste „Cancel“ wieder in das Setup-Menü.

Will man die Eingabe speichern, so ist in den Modi Keyset und Passwordset eine der Tasten 0 bis 9 auf ProggyKeyJoy zu drücken, unter der man dann später die Tastenfolge bzw. –Kombination an den Computer wiedergeben möchte.

Zum Speichern der Eingabe im Modus Joyset ist der entsprechende Joystick-Hebel bzw. -Knopf zu betätigen. Die Anzeige wechselt dann in der ersten Zeile zu „Show Joy-Func“ und in der zweiten Zeile wird die betätigte Joystick-Funktion gefolgt von „[OK]“ angezeigt. Die Joystick-Funktion kann nun ggf. noch mal korrigiert werden. Durch Drücken der Taste „OK“ wird die Eingabe dann gespeichert. Mit der Taste „Cancel“ hingegen gelangt man wieder ins Setup-Menü ohne die Eingabe zu speichern.

Nach der Speicherung erscheint auf der Anzeige die kurze Nachricht „Data saved!“. Danach befindet man sich wieder im Setup-Menü.

5.3.2 Kalibrieren des Joysticks

Wenn man einen neuen Joystick an ProggyKeyJoy anschließt, dann ist dieser zu kalibrieren, um eine reibungslose Funktion zu gewährleisten.

Um in den Kalibriermodus zu gelangen ist die Taste „Setup“ zweimal zu betätigen. Die Anzeige zeigt nun „Setup: Joystick“ an. Durch Drücken der Taste „OK“ gelangt man in den Joystick-Kalibriermodus. Auf der Anzeige erscheint „Center Joystick and press button“. Die Joystick-Achsen sind nun in Mittelstellung zu bringen und danach ein Joystick-Knopf zu betätigen. Auf der Anzeige erscheint nun „Move Joystick and press button“. Am Joystick sind nun alle Achsen von ganz rechts nach ganz links bzw. von ganz hoch nach ganz runter zu bewegen. Danach werden durch Drücken eines Joystick-Knopfes die Kalibrationsdaten gespeichert. Will man die Kalibrationsdaten hingegen nicht speichern, so ist die Taste „Cancel“ zu betätigen.

Nach der Speicherung erscheint auf der Anzeige die kurze Nachricht „Joystick calibrated!“. Danach befindet man sich wieder im Setup-Menü.

5.3.3 Ändern der PIN

Mittels einer 5-stelligen PIN wird der Passwordset-Modus gesichert.

Um die PIN zu ändern ist die Taste „Setup“ dreimal zu betätigen. Die Anzeige zeigt nun „Setup: PIN“ an. Durch Drücken der Taste OK gelangt man in den Modus, um die PIN zu ändern. Auf der Anzeige erscheint nun „Enter old PIN!“. Über die Tasten 0 bis 9 ist nun die alte PIN einzugeben. Bei Auslieferung ist die PIN auf „00000“ eingestellt. Bei fehlerhafter Eingabe gefolgt von der kurzen Anzeige „Error!“ gelangt man wieder ins Setup-Menü.

Hat man die alte PIN richtig eingegeben, so wird man zur Eingabe der neuen PIN aufgefordert. Auf der Anzeige erscheint nun „Enter new PIN!“. Über die Tasten 0 bis 9 soll man nun die neue PIN eingeben. Hat man die neue PIN eingegeben, so wird man aufgefordert diese zu wiederholen. Auf der Anzeige erscheint nun „Re-Enter PIN!“. Über die Tasten 0 bis 9 soll man nun die neue PIN wiederholen. Hat man bei der zweiten Eingabe der PIN keinen Fehler gemacht, so wird sie gespeichert und eine kurze Meldung „PIN saved!“ erscheint. Hat man bei der zweiten Eingabe der PIN hingegen einen Fehler gemacht, so erscheint die kurze Meldung „Error!“ und die alte PIN ist weiterhin gültig. Danach befindet man sich wieder im Setup-Menü.

Die Änderung der PIN kann zu jedem Zeitpunkt durch Drücken der Taste „Cancel“ abgebrochen werden. Man gelangt dann wieder ins Setup-Menü.

5.3.4 Einstellen der LCD-Hintergrundbeleuchtung

Um die LCD-Hintergrundbeleuchtung einzustellen ist die Taste Setup viermal zu betätigen. Die Anzeige zeigt nun „Setup: LCD-Backlight:“ gefolgt von einer Ziffer 0 bis 9 an. Durch Drücken der Taste „OK“ gelangt man in den Einstell-Modus für die LCD-Hintergrundbeleuchtung. Auf der Anzeige erscheint nun „LCD-Backlight:“ gefolgt von einer Ziffer 0 bis 9.

Durch Drücken einer der Tasten 0 bis 9 kann man nun die LCD-Hintergrundbeleuchtung einstellen. Durch Drücken der Taste „OK“ wird der neue Wert gespeichert. Will man hingegen den alten Wert weiterverwenden, so ist die Taste „Cancel“ zu drücken.

Nach der Speicherung erscheint auf der Anzeige die kurze Nachricht „Backlight saved!“. Danach befindet man sich wieder im Setup-Menü.

5.3.5 Einstellen des LCD-Kontrastes

Um den LCD-Kontrast einzustellen ist die Taste Setup fünfmal zu betätigen. Die Anzeige zeigt nun „Setup: LCD-Contrast: “ gefolgt von einer Ziffer 0 bis 9 an. Durch Drücken der Taste „OK“ gelangt man in den Einstell-Modus für den LCD-Kontrast. Auf der Anzeige erscheint nun „LCD-Contrast: “ gefolgt von einer Ziffer 0 bis 9.

Durch Drücken einer der Tasten 0 bis 9 kann man nun den LCD-Kontrast einstellen. Durch Drücken der Taste „OK“ wird der neue Wert gespeichert. Will man hingegen den alten Wert weiterverwenden, so ist die Taste „Cancel“ zu drücken.

Nach der Speicherung erscheint auf der Anzeige die kurze Nachricht „Contrast saved!“. Danach befindet man sich wieder im Setup-Menü.

6. Technische Daten

Die nachfolgenden technischen Daten beziehen sich auf den Prototyp und können bei einem Serienprodukt (z.B. bei Maßen und Gewicht) abweichen.

Modell:	ProggyKeyJoy 0.89
Maße (L x B x H):	213mm x 130mm x 77mm (ohne Anschlüsse)
Gewicht:	ca. 600g
Schutzklasse:	III
Spannungsversorgung:	4,5V – 5,5V (über Computer)
Stromaufnahme:	90mA (bei 5V)
Leistungsaufnahme:	0,45W (bei 5V)
Zulässige Betriebstemperatur:	0°C – 50°C
LCD-Anzeige:	2 Zeilen mit je 16 Zeichen

	Set 1		Set 2		Set 3	
Taste	Make	Break	Make	Break	Make	Break
A	1E	9E	1C	F0, 1C	1C	F0, 1C
B	30	B0	32	F0, 32	32	F0, 32
C	2E	AE	21	F0, 21	21	F0, 21
D	20	A0	23	F0, 23	23	F0, 23
E	12	92	24	F0, 24	24	F0, 24
F	21	A1	2B	F0, 2B	2B	F0, 2B
G	22	A2	34	F0, 34	34	F0, 34
H	23	A3	33	F0, 33	33	F0, 33
I	17	97	43	F0, 43	43	F0, 48
J	24	A4	3B	F0, 3B	3B	F0, 3B
K	25	A5	42	F0, 42	42	F0, 42
L	26	A6	4B	F0, 4B	4B	F0, 4B
M	32	B2	3A	F0, 3A	3A	F0, 3A
N	31	B1	31	F0, 31	31	F0, 31
O	18	98	44	F0, 44	44	F0, 44
P	19	99	4D	F0, 4D	4D	F0, 4D
Q	10	90	15	F0, 15	15	F0, 15
R	13	93	2D	F0, 2D	2D	F0, 2D
S	1F	9F	1B	F0, 1B	1B	F0, 1B
T	14	94	2C	F0, 2C	2C	F0, 2C
U	16	96	3C	F0, 3C	3C	F0, 3C
V	2F	AF	2A	F0, 2A	2A	F0, 2A
W	11	91	1D	F0, 1D	1D	F0, 1D
X	2D	AD	22	F0, 22	22	F0, 22
Y	15	95	35	F0, 35	35	F0, 35
Z	2C	AC	1A	F0, 1A	1A	F0, 1A
0	0B	8B	45	F0, 45	45	F0, 45
1	02	82	16	F0, 16	16	F0, 16
2	03	83	1E	F0, 1E	1E	F0, 1E
3	04	84	26	F0, 26	26	F0, 26
4	05	85	25	F0, 25	25	F0, 25
5	06	86	2E	F0, 2E	2E	F0, 2E
6	07	87	36	F0, 36	36	F0, 36
7	08	88	3D	F0, 3D	3D	F0, 3D
8	09	89	3E	F0, 3E	3E	F0, 3E
9	0A	8A	46	F0, 46	46	F0, 46
`	29	89	0E	F0, 0E	0E	F0, 0E
-	0C	8C	4E	F0, 4E	4E	F0, 4E
=	0D	8D	55	FO, 55	55	F0, 55
\	2B	AB	5D	F0, 5D	5C	F0, 5C
BKSP	0E	8E	66	F0, 66	66	F0, 66
SPACE	39	B9	29	F0, 29	29	F0, 29
TAB	0F	8F	0D	F0, 0D	0D	F0, 0D
CAPS	3A	BA	58	F0, 58	14	F0, 14
L SHFT	2A	AA	12	FO, 12	12	F0, 12
L CTRL	1D	9D	14	FO, 14	11	F0, 11
L GUI	E0, 5B	E0, DB	E0,1F	E0, F0, 1F	8B	F0, 8B
L ALT	38	B8	11	F0, 11	19	F0, 19
R SHFT	36	B6	59	F0, 59	59	F0, 59
R CTRL	E0, 1D	E0, 9D	E0,14	E0, F0, 14	58	F0, 58
R GUI	E0, 5C	E0, DC	E0,27	E0, F0, 27	8C	F0, 8C

	Set 1		Set 2		Set 3	
Taste	*Make*	*Break*	*Make*	*Break*	*Make*	*Break*
R ALT	E0, 38	E0, B8	E0,11	E0, F0, 11	39	F0, 39
APPS	E0, 5D	E0, DD	E0,2F	E0, F0, 2F	8D	F0, 8D
ENTER	1C	9C	5A	F0, 5A	5A	F0, 5A
ESC	01	81	76	F0, 76	08	F0, 08
F1	3B	BB	05	F0, 05	07	F0, 07
F2	3C	BC	06	F0, 06	0F	F0, 0F
F3	3D	BD	04	F0, 04	17	F0, 17
F4	3E	BE	0C	F0, 0C	1F	F0, 1F
F5	3F	BF	03	F0, 03	27	F0, 27
F6	40	C0	0B	F0, 0B	2F	F0, 2F
F7	41	C1	83	F0, 83	37	F0, 37
F8	42	C2	0A	F0, 0A	3F	F0, 3F
F9	43	C3	01	F0, 01	47	F0, 47
F10	44	C4	09	F0, 09	4F	F0, 4F
F11	57	D7	78	F0, 78	56	F0, 56
F12	58	D8	07	F0, 07	5E	F0, 5E
PRNT SCRN	E0, 2A, E0, 37	E0, B7, E0, AA	E0, 12, E0, 7C	E0, F0, 7C, E0, F0, 12	57	F0, 57
SCROLL	46	C6	7E	F0, 7E	5F	F0, 5F
PAUSE	E1, 1D, 45, E1, 9D, C5		E1, 14, 77, E1, F0, 14, F0, 77		62	F0, 62
[	1A	9A	54	FO, 54	54	F0, 54
INSERT	E0, 52	E0,D2	E0,70	E0, F0, 70	67	F0, 67
HOME	E0, 47	E0,97	E0,6C	E0, F0, 6C	6E	F0, 6E
PG UP	E0, 49	E0,C9	E0,7D	E0, F0, 7D	6F	F0, 6F
DELETE	E0, 53	E0,D3	E0,71	E0, F0, 71	64	F0, 64
END	E0, 4F	E0,CF	E0,69	E0, F0, 69	65	F0, 65
PG DN	E0, 51	E0,D1	E0,7A	E0, F0, 7A	6D	F0, 6D
U ARROW	E0, 48	E0,C8	E0,75	E0, F0, 75	63	F0, 63
L ARROW	E0, 4B	E0,CB	E0,6B	E0, F0, 6B	61	F0, 61
D ARROW	E0, 50	E0,D0	E0,72	E0, F0, 72	60	F0, 60
R ARROW	E0, 4D	E0,CD	E0,74	E0, F0, 74	6A	F0, 6A
NUM	45	C5	77	F0, 77	76	F0, 76
KP /	E0, 35	E0,B5	E0,4A	E0, F0, 4A	4A	F0, 4A
KP *	37	B7	7C	F0, 7C	7E	F0, 7E
KP -	4A	CA	7B	F0, 7B	4E	F0, 4E
KP +	4E	CE	79	F0, 79	7C	F0, 7C
KP EN	E0, 1C	E0,9C	E0,5A	E0, F0, 5A	79	F0, 79
KP .	53	D3	71	F0, 71	71	F0, 71
KP 0	52	D2	70	F0, 70	70	F0, 70
KP 1	4F	CF	69	F0, 69	69	F0, 69
KP 2	50	D0	72	F0, 72	72	F0, 72
KP 3	51	D1	7A	F0, 7A	7A	F0, 7A
KP 4	4B	CB	6B	F0, 6B	6B	F0, 6B
KP 5	4C	CC	73	F0, 73	73	F0, 73
KP 6	4D	CD	74	F0, 74	74	F0, 74
KP 7	47	C7	6C	F0, 6C	6C	F0, 6C
KP 8	48	C8	75	F0, 75	75	F0, 75

	Set 1		Set 2		Set 3	
Taste	*Make*	*Break*	*Make*	*Break*	*Make*	*Break*
KP 9	49	C9	7D	F0, 7D	7D	F0, 7D
]	1B	9B	5B	F0, 5B	5B	F0, 5B
;	27	A7	4C	F0, 4C	4C	F0, 4C
'	28	A8	52	F0, 52	52	F0, 52
,	33	B3	41	F0, 41	41	F0, 41
.	34	B4	49	F0, 49	49	F0, 49
/	35	B5	4A	F0, 4A	4A	F0, 4A
ACPI						
Power	E0, 5E	E0, DE	E0, 37	E0, F0, 37		
Sleep	E0, 5F	E0, DF	E0, 3F	E0, F0, 3F		
Wake	E0, 63	E0, E3	E0, 5E	E0, F0, 5E		
Windows Multimedia						
Next Track	E0, 19	E0, 99	E0, 4D	E0, F0, 4D		
Previous Track	E0, 10	E0, 90	E0, 15	E0, F0, 15		
Stop	E0, 24	E0, A4	E0, 3B	E0, F0, 3B		
Play/Pause	E0, 22	E0, A2	E0, 34	E0, F0, 34		
Mute	E0, 20	E0, A0	E0, 23	E0, F0, 23		
Volume Up	E0, 30	E0, B0	E0, 32	E0, F0, 32		
Volume Down	E0, 2E	E0, AE	E0, 21	E0, F0, 21		
Media Select	E0, 6D	E0, ED	E0, 50	E0, F0, 50		
E-Mail	E0, 6C	E0, EC	E0, 48	E0, F0, 48		
Calculator	E0, 21	E0, A1	E0, 2B	E0, F0, 2B		
My Computer	E0, 6B	E0, EB	E0, 40	E0, F0, 40		
WWW Search	E0, 65	E0, E5	E0, 10	E0, F0, 10		
WWW Home	E0, 32	E0, B2	E0, 3A	E0, F0, 3A		
WWW Back	E0, 6A	E0, EA	E0, 38	E0, F0, 38		
WWW Foreward	E0, 69	E0, E9	E0, 30	E0, F0, 30		
WWW Stop	E0, 68	E0, E8	E0, 28	E0, F0, 28		
WWW Refresh	E0, 67	E0, E7	E0, 20	E0, F0, 20		
WWW Favorites	E0, 66	E0, E6	E0, 18	E0, F0, 18		