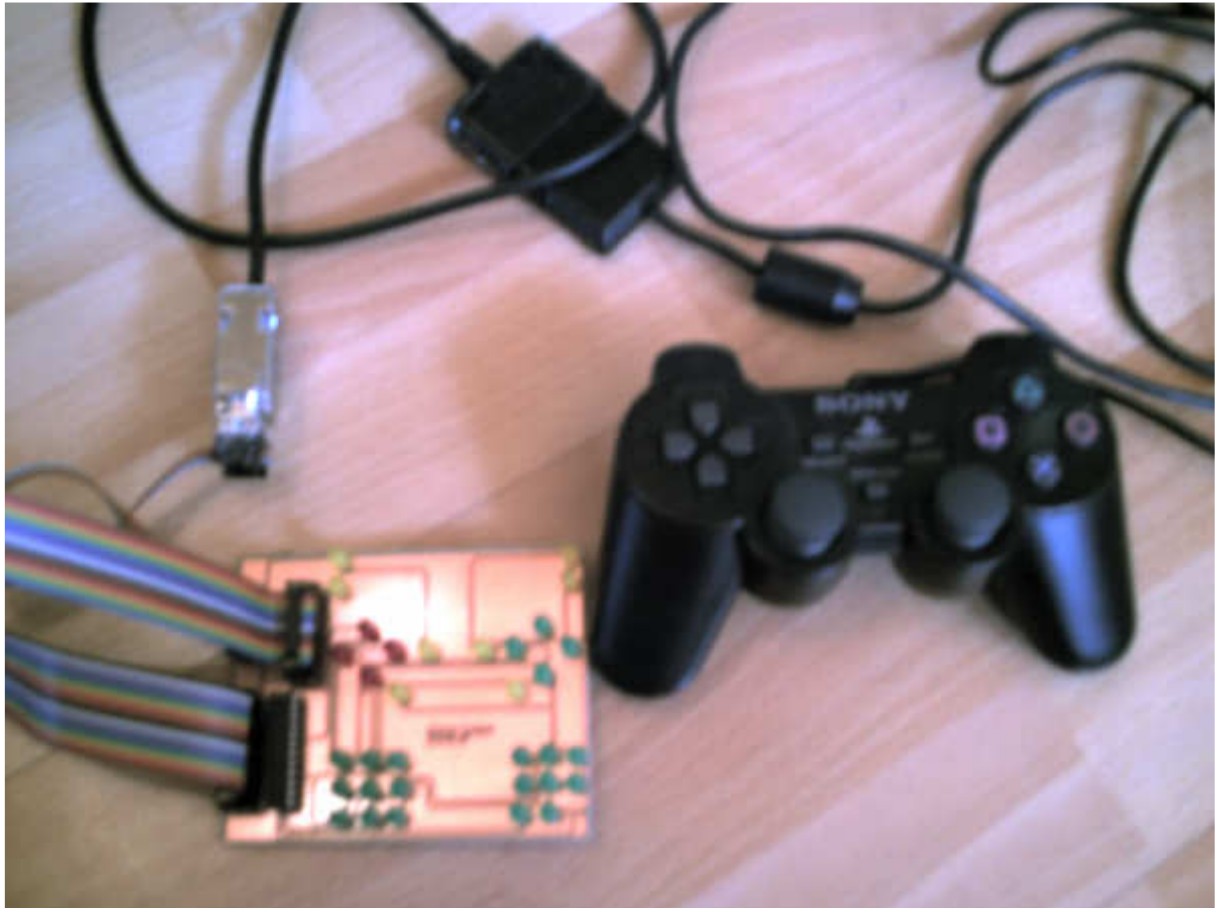


Abschlussprojekt
„Arbeitsmethoden der technischen Informatik“



Playstation 2 Controller (Digital & Analog)
am PIC16F84A
mit Zustandsausgabe über MAX7219

von
Aryan Layes

Inhaltsverzeichnis

Einleitung

- 1.0 Projektvorgaben
- 1.1 Projektbeschreibung
- 1.2 Ziel des Projekts

Hardware

- 2.0 Schaltplan des PS2-MAX-Boards
- 2.1 Bestückung / Stückliste
- 2.2 Die Schnittstelle des Playstation Controllers
- 2.3 Kommunikations-Protokoll des Playstation Controllers
- 2.4 MAX7219 I/O
- 2.5 MAX7219 Steuerung

Software

- 3.0 Neue Programmiermethoden
- 3.1 Quellcode
- 3.2 Flussdiagramme

Bilanz

- 4.0 Eigene Erfahrungen

Anhang

- 5.0 Quellen
- 5.1 Versicherung der Eigenleistung

1.0 Projektvorgaben

Im Fach „Arbeitsmethoden der technischen Informatik“ war es unsere Aufgabe, ein Projekt mit Hilfe der im Unterricht angefertigten PIC-Platine anzufertigen. Es wurden mehrere Projektvorschläge angeboten die mir alle nicht zusagten. Zu erst wollte ich einen eigenen Controller mit Hilfe eines alten PS-Controllers entwickeln, doch Herr Glaser überredete mich den Controller funktionstüchtig zu lassen und mein Projekt auf das Auslesen des Controller mit einer einfachen Anzeige zu beschränken.

Zeitliche Vorgaben:

Für das gesamte Projekt ist ein Zeitrahmen von dem Tag der Projektgenehmigung bis zum 19.04.07 vorgesehen.

1.1 Projektbeschreibung

Mein Projekt besteht aus der PIC-Platine (bestückt mit einem PIC16F84A), einem Playstation Controller und einem MAX7219 LED Display Treiber von MAXIM.

Der Playstation Controller ist über den Wannenstecker SV1 mit dem PIC an PORTA verbunden.

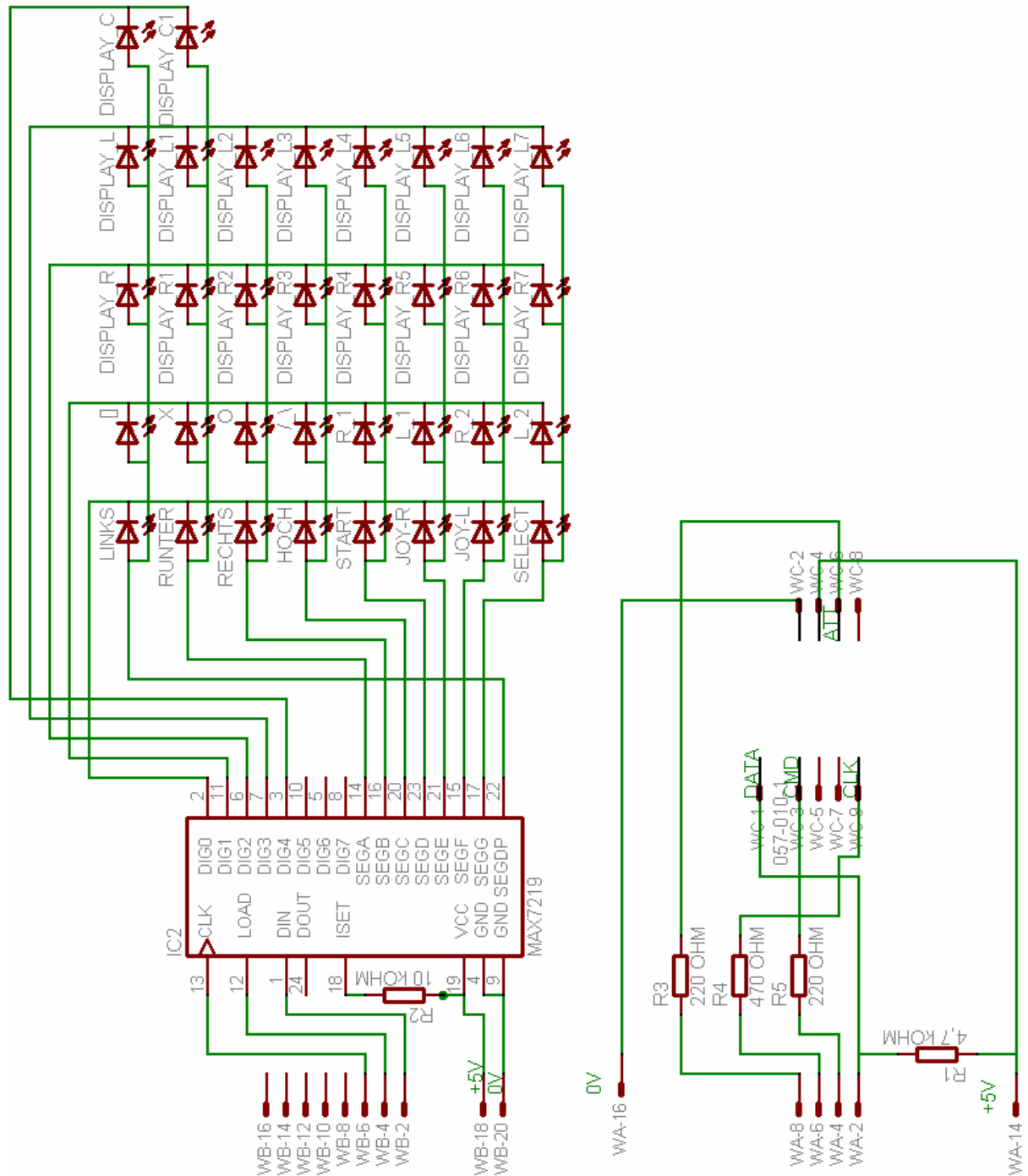
Der MAX 7219 Baustein ist über den Wannenstecker SV2 mit dem PIC an PORTB verbunden.

Der PIC kommuniziert mit dem PS-Controller und erhält so die Zustände der Tasten und Joysticks. Nachdem er diese Daten erhalten hat, überträgt er sie an den MAX7219 Display Treiber, aus dem Hause MAXIM, um die Zustände anzeigen zu lassen.

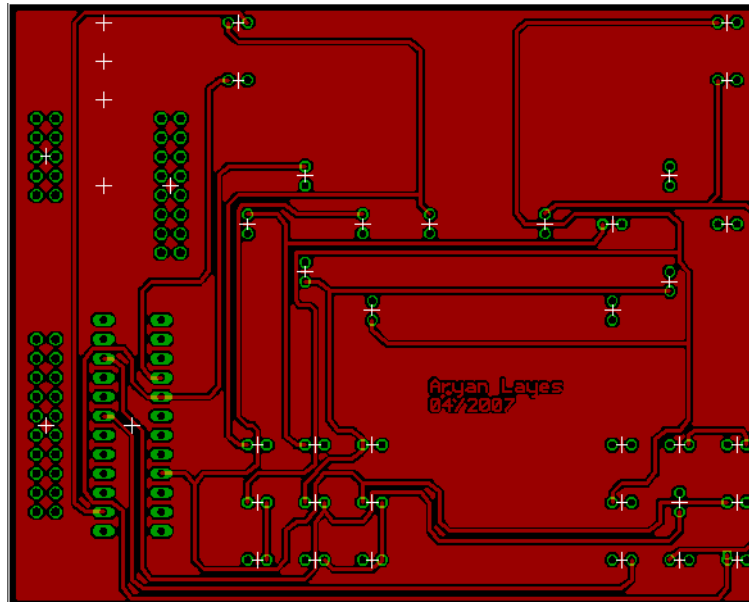
1.2 Ziel des Projekts

Mein Projekt hat in erster Linie nur das Ziel zu verstehen, wie man mit einem Playstation-Controller kommuniziert um die Zustände der Tasten und Joysticks herauszufinden. Die Zustandsanzeige der Tasten und Joysticks über den MAX-Baustein ist nur durch eigene Interesse und Neugier entstanden. Auf diesem Wissen lassen sich jetzt viele Projekte aufbauen, z.B. ein „kabelloser“ Controller oder das Steuern eines Roboters mit dem PS-Controller.

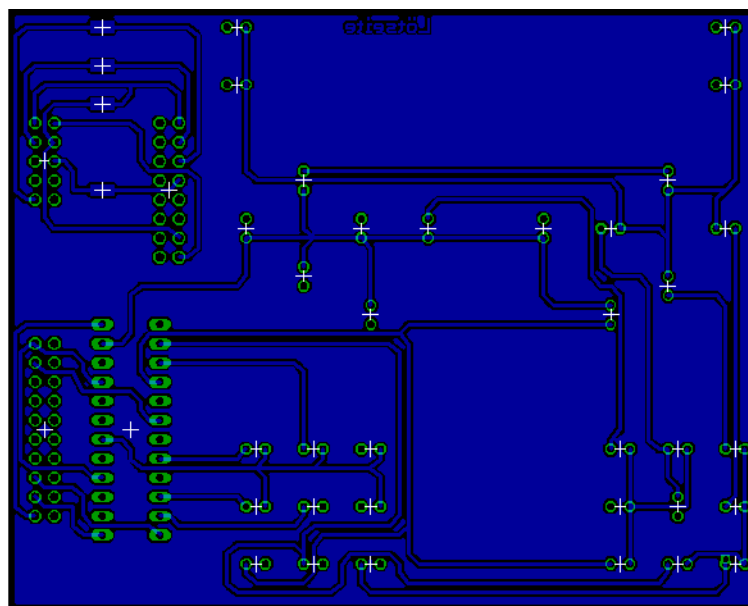
2.0 Schaltplan des PS2-MAX-Boards



Vorderseite des PSC-MAX-Boards



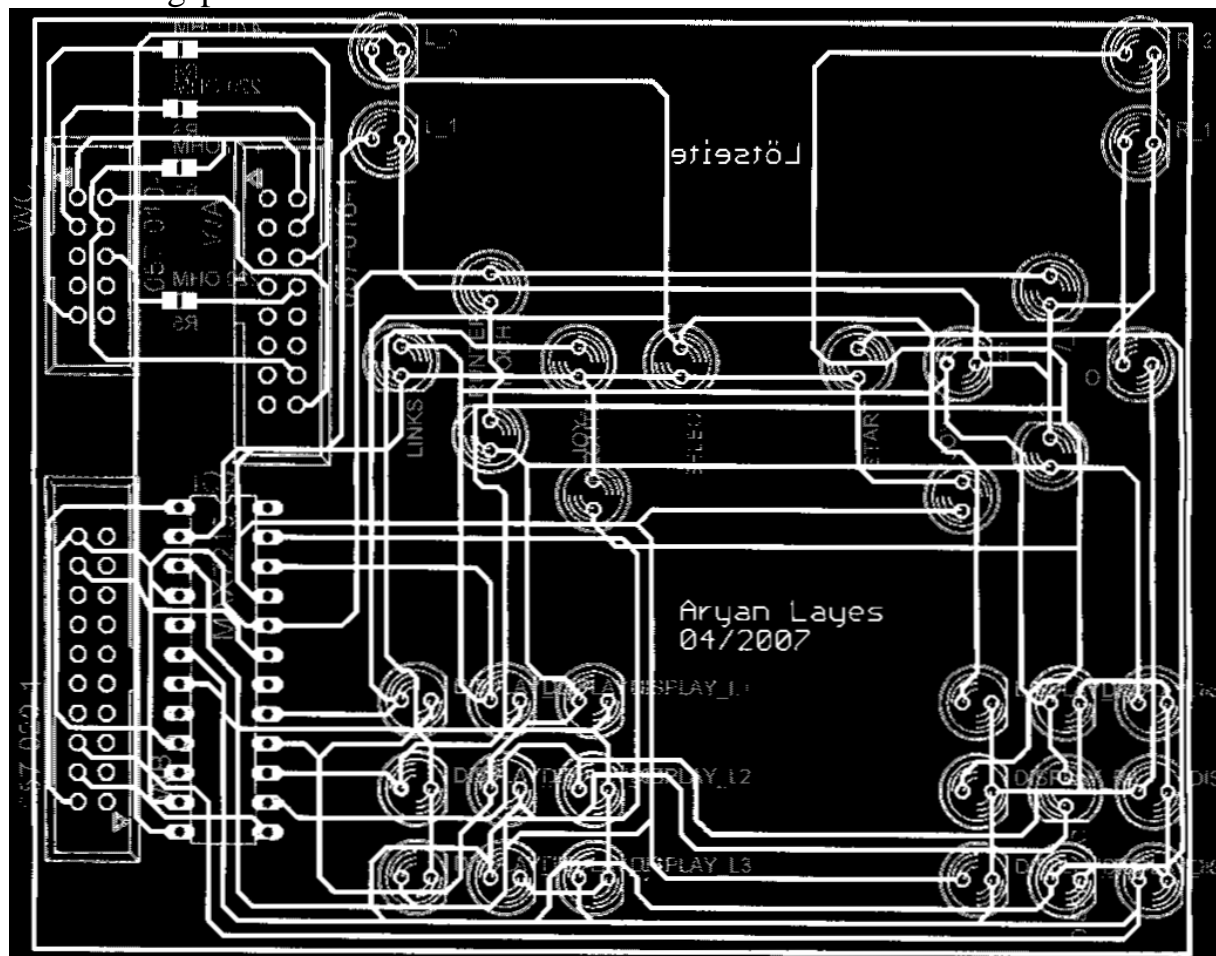
Rückseite und damit Lötseite des PSC-MAX-Boards



2.1 Bestückung / Stückliste

Bauteil	Wert
alle LED's	besitzen einen integrierter Vorwiderstand, der für eine Spannung von 5 Volt vorgesehen ist
R1	4,7 kOHM
R2	10 kOHM
R3,R5	220 OHM
R4	470 OHM
R6	10 kOHM
IC2	MAX 7219

Bestückungsplan



- In der Schaltung kommen vier verschiedene Widerstandswerte vor:

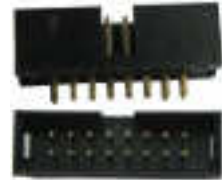
220 OHM	ROT ROT BRAUN	GOLD
470 OHM	GELB VIOLETT SCHWARZ SCHWARZ	GOLD
4,7 kOHM	GELB VIOLETT SCHWARZ BRAUN	GOLD
10 kOHM	BRAUN SCHWARZ ORANGE	GOLD



- Beim einsetzen des IC-Sockel (24 Pins) auf die Beinchen achten, da sie leicht beschädigt werden können!



- Bei den Wannenstecker auf die Richtung achten!



- Bei den LEDs auf die Polung achten!
In einem Digit sind die Kathoden (flache Seite) miteinander verbunden. Die Anoden gehen an die entsprechenden Segmente.



Nur noch den MAX7219 vorsichtig auf den Sockel drücken und der Hardwareaufbau ist erledigt.

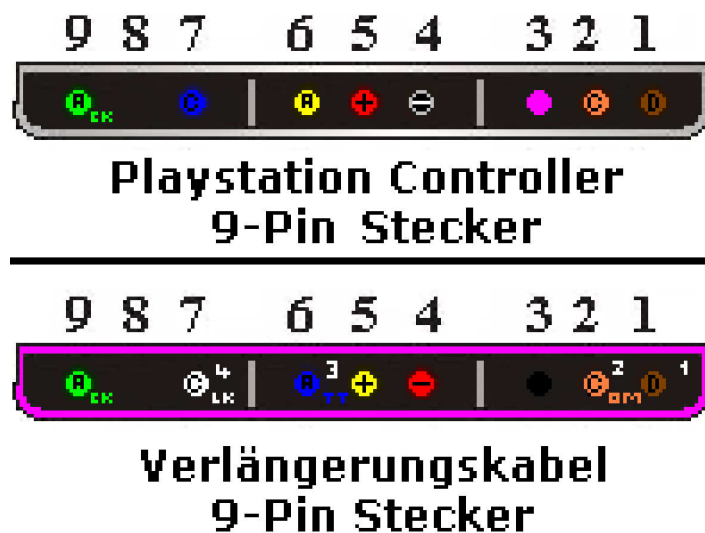
2.2 Die Schnittstelle des Playstation Controllers

Das wohl größte Problem in meinem Projekt lag an dieser Schnittstelle! In den ersten Wochen der Projektphase verwendete ich, durch falsche Informationen aus dem Internet, die Schnittstelle umgekehrt. Nach einem Öffnen des Steckers eines original Controllers bemerkte ich das Problem.

Ich benutzte ein PS-Verlängerungskabel von EAXUS um den PS-Controller anzuschließen (ohne ihn unbrauchbar für die Playstation zu machen). Vom Verlängerungskabel benutzte ich die PS-Female Schnittstelle mit ca. 20 cm Kabel. Das andere Ende des Kabels habe ich auf einen SUB-D gelötet.

Jetzt aber noch zu einem anderen Grund des Problems: Wie auf dem Bild zu sehen ist, verwendet der Hersteller des Verlängerungskabels nicht die gleiche Reihenfolge der Adernfarben. Zwar konnte ich mit dem Messgerät erkennen welche Ader zu welchem Pin gehört, aber anhand der Farbe war es mir nicht möglich zu erkennen das ich die Schnittstelle falsch verwende.

Danach kam ich zu folgendem Ergebnis:



← Bitte beachten, dass die **Clock**-Ader eine **schwarze Schirmung** hat, nicht wie abgebildet eine weiße. Wenn ich es schwarz dargestellt hätte, hätte man nichts erkennen können. ;)

1. **(Ausgang) DATA**, serielle Datenübertragung
 - bei negativer Flanke von **CLOCK** sendet der Controller ein Bit über die **DATA** Leitung
 - die Datenübertragung beginnt mit dem niederwertigsten Bit (LSB-Verfahren).
2. **(Eingang) COMMAND**, serielle Datenübertragung
 - bei negativer Flanke von **CLOCK** liest der Controller das Bit auf der **COMMAND** Leitung
3. *Nicht belegt*
4. **(Eingang) GND** - Pin an Masse
5. **(Eingang) VCC** - Pin zur +5V Spannungsquelle
6. **(Eingang) ATT** - bevor der Controller Daten erhält, muss hier 0V (Masse) anliegen.
7. **(Eingang) CLOCK** - zur synchronisierten Übertragung
8. *Nicht belegt*
9. **(Ausgang) ACK** - sendet ein Signal, wenn ein Byte über **COMMAND** angekommen ist

2.3 Kommunikations-Protokoll des Playstation Controllers

Bevor der Controller überhaupt weiß, dass er jetzt „dran“ ist und die **COMMAND** Leitung abhört, muss **ATT** auf Masse liegen (0V). Da der PS-Controller nicht gerade der schnellste ist, sollte man hier ca. 50µs warten, bevor es mit der seriellen Übertragung losgeht.






Die serielle Übertragung besteht bei einem Analog Controller aus 9 Bytes, bei einem Digital Controller nur aus 5 Bytes. Eine Übertragung mit 9 Bytes bei einem Digital Controller funktioniert dennoch, dazu aber später mehr.

Die Übertragung erfolgt nach dem LSB-Verfahren!

In der folgenden Tabelle ist der Übertragungsablauf dargestellt. Dieses Beispiel bezieht sich auf die Kommunikation mit einem „**Analog Red Mode Controller**“.

Byte	COMMAND		DATA		empfangene Information
	Hex	Binär	Hex	Binär	
1	0x01	0000 0001	-	-	-
2	0x42	0100 0010	0x73	0111 0011	Controller-Typ*
3	0x00	0000 0000	0x5A	0101 1010	Status*
4	0x00	0000 0000	0xXX	xxxx xxxx	Zustände der linken Tasten*
5	0x00	0000 0000	0xXX	xxxx xxxx	Zustände der rechten Tasten*
6	0x00	0000 0000	0xXX	xxxx xxxx	Position des rechten Joysticks x-Achse*
7	0x00	0000 0000	0xXX	xxxx xxxx	Position des rechten Joysticks y-Achse*
8	0x00	0000 0000	0xXX	xxxx xxxx	Position des linken Joysticks x-Achse*
9	0x00	0000 0000	0xXX	xxxx xxxx	Position des linken Joysticks y-Achse*

*Controller-Typ

Controller	Controller	empfangener Wert
	Standard Digital Controller	0x41
	Analog Controller in Red Mode	0x73
	Analog Controller in Green Mode	0x53
	NegCon (Racing Controller)	0x23
	PSX Maus	0x12

Quelle: http://www.technick.net/public/code/cp_dp.php?aiocp_dp=pinconjoy_psx

*Status

Wenn das empfangene Byte ungleich 01011010 ist, ist der Controller nicht bereit oder sogar defekt! Die Zustände der Tasten, die in der folgenden Übertragung empfangen werden, sind wahrscheinlich unbrauchbar.

Allgemeines zum Zustand der Tasten:

Wenn eine Taste gedrückt ist, sendet der Controller an der Stelle eine 0.

Im englischen heißt das, dass die Tasten „active low“ sind !

*linke Tasten

Zu den linken Tasten gehören:



Die Zustände der **gelb** markierten Tasten sind bei einem Standard Digital Controller, der keine Joysticks hat, immer auf 1.

*rechte Tasten

Zu den rechten Tasten gehören:



*Position des rechten/linken Joysticks [x-Achse]/[y-Achse]

Die Joysticks sind so hergestellt, dass sie auf der x-Achse und y-Achse jeweils 256 Zustände annehmen können. Diese Genauigkeit reicht völlig aus und braucht auch nur ein Byte für eine Achse.

Wenn das Joystick ganz centriert ist (nicht bewegt)

x-Achse = 1000 0000 b = 128 dez (genau die Hälfte der möglichen Zustände)

y-Achse = 1000 0000 b = 128 dez (genau die Hälfte der möglichen Zustände)

x-Achse < 1000 0000 b => Joystick ist nach links gedrückt

x-Achse > 1000 0000 b => Joystick ist nach rechts gedrückt

y-Achse < 1000 0000 b => Joystick ist nach oben gedrückt

y-Achse > 1000 0000 b => Joystick ist nach unten gedrückt

Jetzt sollte jedem klar sein, was die empfangenen Daten bedeuten. Deshalb gehen wir jetzt zu der Übertragung selbst.

Zur Übertragung brauchen wir die **COMMAND**, die **DATA**, die **CLOCK** und die **ATT**-Leitung.

1. Ausgangssituation:

Zustand vor der Übertragung:

COMMAND = 0

CLOCK = 1

ATT = 1

2. Übertragungsvorbereitungen

Die **ATT**-Leitung (hab ich vorhin schon erwähnt) muss auf 0 geschaltet werden und wegen des langsamen Controllers sollte danach eine Wartezeit von ca. 50µs eingehalten werden.

Nach den 50µs legen wir das erste Bit, das wir übertragen möchten, auf die **COMMAND**-Leitung.

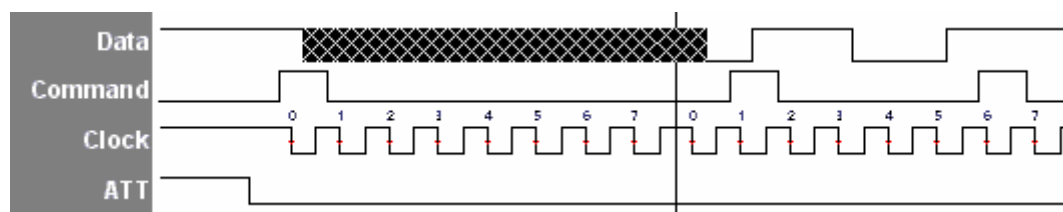
3. Übertragung

Danach setzen wir **CLOCK** von 1 auf 0 um eine negative Flanke zu erzeugen. Jetzt braucht der Controller ca. 15µs um das Bit zu lesen und zu verarbeiten, also muss auch hier eine Wartezeit von 15µs eingehalten werden. **CLOCK** wird danach wieder auf 1 gesetzt und das nächste Bit kommt auf die **COMMAND**-Leitung. Gehe zu 3. Übertragung.

4. Empfangen während der Übertragung

Ein bisschen „komplizierter“ wird es bei dem Übertragen und gleichzeitigen Empfangen.

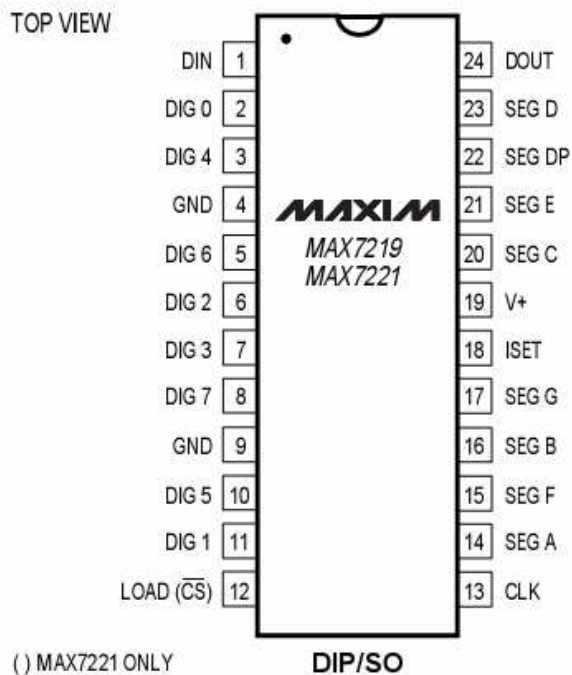
Das Bit, das übertragen werden soll, kommt auf die **COMMAND**-Leitung, **CLOCK** wird von 1 auf 0 gesetzt. Jetzt muss 15µs gewartet werden, danach wird überprüft ob auf der **DATA**-Leitung eine 1 oder 0 liegt und der entsprechende Wert gespeichert.



Hier ist das Zeit-Signal Diagramm der Übertragung der ersten beiden Bytes an einen Analog Red Mode Controller dargestellt. Der vertikale Strich trennt die beiden Bytes.

2.4 MAX7219 I/O

Mit Hilfe der schematischen Abbildung des MAX 7219 LED Display Treibers werde ich dessen einzelnen Ein- und Ausgänge beschreiben.



Spannungs- und Stromversorgung

19. **V+** (+5V) Spannungsquelle

4 & 9. **GND**

5. **ISET** Helligkeitsregelung der LED Anzeige über extern geschalteter Widerstand.

I/O Kommunikation/Steuerung des MAX-IC's

1. **(Eingang) DIN**, serielle Datenübertragung

- die Datenübertragung beginnt mit dem letzten Bit 15 (MSB-Verfahren).
- bei positiver Flanke von **CLK** übernimmt der MAXIM das anstehende Bit an DIN.

12. **(Eingang) LOAD** - die Übertragung beginnt mit dem Schalten von **LOAD** auf 0V (Masse). Die zwei Bytes werden transferiert und danach wird LOAD wieder auf 1(+5V) geschaltet.

13. **(Eingang) CLK** - zur synchronisierten Übertragung der Bits über **DIN**

LED Steuerung

6. **(Ausgang) DIG 0 bis DIG 7** – Jede LED wird über dazugehörigen Digitausgang angesteuert, dies geschieht über die Kathoden der LEDs

7. **(Ausgang) SEG DP - G** – Die einzelnen Segmente der Digits werden mit den gleichen Segmenten der anderen Digits verbunden. Zum Beispiel:

Anode der SEG DP LED im Digit 0 ist verbunden mit der Anode der SEG DP LED im Digit 1.

2.5 MAX7219 Steuerung

Die Steuerung des MAX7219 erfolgt über zwei Byte lange Befehle.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
X	X	X	X	ADDRESS				DATA								LSB

Das zweite Byte (links) gibt an auf welches Register des MAX7219 zugegriffen wird. Mit dem ersten Byte (rechts) werden die Werte in dem Register festgelegt. Die folgende Tabelle aus dem Datenblatt des MAX7219 zeigt welche Register zur Verfügung stehen.

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	X0
Digit 0	X	0	0	0	1	X1
Digit 1	X	0	0	1	0	X2
Digit 2	X	0	0	1	1	X3
Digit 3	X	0	1	0	0	X4
Digit 4	X	0	1	0	1	X5
Digit 5	X	0	1	1	0	X6
Digit 6	X	0	1	1	1	X7
Digit 7	X	1	0	0	0	X8
Decode Mode	X	1	0	0	1	X9
Intensity	X	1	0	1	0	XA
Scan Limit	X	1	0	1	1	XB
Shutdown	X	1	1	0	0	XC
Display Test	X	1	1	1	1	XF

Digit 0-7:

speichert die Zustände der Segmente

Decode Mode:

Der MAX7219 wurde so entwickelt, dass er neben der ganz normalen Ausgabe der Digits einen integrierten BCD-Decoder bereitstellt. Auf diesen werde ich in meiner Dokumentation aber nicht eingehen.

Intensity:

Auch hier gibt es eine spezielle Funktion des Display Treibers. Die Helligkeitssteuerung kann über den ISET-Eingang, oder durch dieses Register erfolgen. Da ich in meinem Projekt keine Helligkeitssteuerung verwende und das Datenblatt die Informationen zu diesem Feature beinhaltet, gehe ich nicht näher darauf ein.

Scan Limit:

Über dieses Register lässt sich einstellen wie viele Digits genutzt werden. Nicht benutzte Digits sollten nicht aktiviert werden, da diese die Helligkeit der Anzeige „dimmen“ würden.

Shutdown:

Mit diesem Register wird die Anzeige ein- und ausgeschaltet.

Shutdown:

Mit diesem Register kann die Anzeige geprüft werden. Im Testmodus sollten alle LEDs an sein.

3.0 Neue Programmiermethoden

Im Laufe der Programmierung vergrößerte sich mein Programmcode. Deshalb habe ich gegen Ende der Projektfertigstellung mir überlegt, wie ich einige Funktionen verändern kann um den Programmcode zu minimieren.

Ich bin zu dem Entschluss gekommen, das die Kommunikation mit dem PS-Controller und dem MAX7219 über Tabelle eine starke Optimierung des Codes erreichen kann.

Problem:

Wie kann ich aus einer Tabelle einen nicht konstanten Wert laden?

Lösung:

Erstellung „dynamischer Tabellen“ durch indirekte Adressierung.

Die indirekte Adressierung wird in den Flussdiagrammen der Unterprogramme UP_PSC_send_tabelle und UP_MAX_send_tabelle genauestens erklärt.

3.1 Quellcode

[illegible]

```

;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;      A      Joy-L-X
;
;      -----
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;      A      Joy-L-Y
;
;      -----
;
;      *****
;      *                  Changelog beta_4_1                  *
;      *****
;      Überprüfung ob Controller bereit
;      wenn nicht bereit-> MAX Digits löschen
;      und dann warten bis Controller wieder bereit
;      *****
;      *                  Changelog beta_4_0                  *
;      *****
;      Register Invertierung
;      gLEFT und gRIGHT, sodass nur noch die
;      LED's der Taste an sind, die auch
;      gedrückt sind.
;
;
;      *****
;      * Bestimmung des Prozessortyps für den Assembler und das Programmiergerät *
;      *****
;
LIST p=16F84A

;      *****
;      * Includedatei für den 16F84A einbinden (vordef. Reg. und Konst.) *
;      *****
;
#include <p16f84A.INC>

; Diese Datei enthält Vordefinitionen für wichtige Register und Konstanten.
; (Z.B. gibt es die Konstante PORTB mit der sich ohne Angabe der
; absoluten Adresse H'0006' der Port B des Prozessors ansprechen lässt)

;      *****
;      * Konfigurationseinstellungen für IC-Prog vordefinieren *
;      *****
;
__CONFIG _PWRTE_ON & _CP_OFF & _HS_OSC & _WDT_OFF

; Hier werden verschiedene Prozesseigenschaften festgelegt:
; _PWRTE_ON schaltet den Power Up Timer ein, d.h. der Prozessor wartet nach
; dem Einschalten ca. 70ms mit dem Programmstart, um sicher zu sein,
; dass alle angeschlossene Peripherie bereit ist.

```



```
; _CP_OFF schaltet die Code-Protection des Prozessor aus. Damit ist das im Prozessor
;   befindliche Programm jederzeit auslesbar und überschreibbar.
; _HS_OSC spezifiziert einen Quarzoszillator (Highspeed) als Zeitbasis für den Prozessor.
; _WDT_OFF schaltet den Watchdog-Timer des Prozessor aus.
```

```
*****
;* Register / Variablen festlegen *
*****
; hier werden Adressen von Registern / Variablen festgelegt. Diese werden beginnend
; mit der Adresse H'20' aufsteigend vergeben.
```

CBLOCK H'20'

;Register des PS2 Controller

```

sPSC          ;Übertragungsregister (dynamische Tabelle)
sIDLE          ;IDLE-Register (= 0)

sSTART          ;START-Befehl  (00000001)
gSTART          ;um später die PS2 Kommunikation per
                ;dynamische Tabelle ab zu arbeiten

sTYPE          ;GET TYPE-Befehl (01000010)
gTYPE          ;Empfangene Daten nach dem GET TYPE Befehl

gSTATUS        ;Status des Controllers (0x5A wenn bereit)
rSTATUS        ;Register um gSTATUS mit 0x5A zu vergleichen

gLEFT          ;Status der Buttons:
                ;<- | V | -> | ^ | Start | JL | JR | Select

gRIGHT         ;Status der Buttons:
                ;[] | X | O | / _ \ | R1 | L1 | R2 | L2

gRJoyX         ;X Position des rechten Joysticks
gRJoyY         ;Y Position des rechten Joysticks

gLJoyX         ;X Position des linken Joysticks
gLJoyY         ;X Position des linken Joysticks
```

;14 Register

;Warteschlangen

```

;wait_20us     ;Wartezeit nach dem a_ATT auf 0 geht
wait_50us      ;Wartezeit nach dem a_ATT auf 0 geht      ;NEW

;wait_5us      ;Wartezeit zwischen CLOCK LOW und HIGH
wait_25us      ;Wartezeit zwischen CLOCK LOW und HIGH    ;NEW
```

;2 Register

;Register des MAX-7219

```

zaehler_bit    ;Zaehler der einzelnen Übertragenene Bits
zaehler_tab    ;Zaehler um in der MAX-Tabelle zu springen

sMAX           ;Register das zur Kommunikation zwischen PIC
                ;und MAX-7219 dient
```

;3 Register

;zur Initialisierung:

```
;MAX Digit Kodierungs
;kMODUS_adr ;Modusbefehlsadresse
;m_MODUS ;MAX Digit Kodierungsmodus

m_D_MODUS_adr ; 0x09 Digit Kodierungsmodus Adresse
m_D_MODUS ; 0x00 MAX Digit Kodierungsmodus ->keine Kodierung<-
```

```
;MAX Digit Helligkeit
m_HELLIGKEIT_adr ; 0x0A MAX Helligkeitssteuerung Adresse
m_HELLIGKEIT ; 0x15 MAX Helligkeitssteuerung ->Digit 31 / 32 (max
ON)<-
```

```
;MAX Digit Anzahl
m_DIGITS_adr ; 0x0B Digitsanzahl Adresse
m_DIGITS ; 0x01 Anzahl der aktivierten Digits ->Digit 0 und 1<-
```

```
;MAX Shutdown
;kSTART_adr ; 0x0C Startbefehlsadresse
;m_START ; 0x01 MAX Shutdownbefehl ->Normal
Operation<-
```

```
m_SHUTDOWN_adr ; 0x0C SHUTDOWN Adresse
m_SHUTDOWN ; 0x01 MAX Shutdownbefehl ->Normal Operation<-
```

```
;MAX Test
m_TEST_adr ; 0x0F Testbefehlsadresse
m_TEST ; 0x00 MAX Testmodus ->KEIN
TESTMODUS<-
;10 Register
```

;zur Digitauswahl

```
m_TYPE_adr ; Type - Digit-Adresse
m_STATUS_adr ; Status - Digit-Adresse

m_LEFT_adr ; LINKS - Digit-Adresse
m_RIGHT_adr ; RECHTS - Digit-Adresse

m_RJoyX_adr ; Joystick rechts x-Achse - Digit-Adresse
m_RJoyY_adr ; Joystick rechts y-Achse - Digit-Adresse

m_LJoyX_adr ; Joystick links x-Achse - Digit-Adresse
m_LJoyY_adr ; Joystick links y-Achse - Digit-Adresse

m_display_r
m_display_c
m_display_l
```

;8 Register

;Register um zu Überprüfen ob Controller bereit
zaehler

;Register für die debugging Funktionen.....

```
wait_05s
wait_05s_1
wait_05s_2
```

;3 Register

;40 Register insgesamt -> Bank 0 reicht dafuer aus
ENDC

```

;*****
;
;      * Konstanten festlegen *
;*****

;*****
;
;      * Konstanten des MAX-7219 *
;*****

;*****
;
; * Definition von einzelnen Bits in einem Register / in einer Variable *
;*****

;##### Ports zum Controller #####
;#####
;#DEFINE      e_DATA      PORTA, 0      ; Data      Eingang
;#DEFINE      a_COMMAND   PORTA, 1      ; Command   Ausgang
;#DEFINE      a_CLOCK     PORTA, 2      ; Takt      Ausgang
;#DEFINE      a_ATT       PORTA, 3      ; ATT       Ausgang
;#DEFINE      a_ACK       PORTA, 4      ; ACK      Eingang NICHT BENUTZT

;##### Ports zur Anzeige #####
;#####
;#DEFINE      a1          PORTB, 0      ; DEBBUGING Ausgang
;#DEFINE      b1          PORTB, 1      ;           Ausgang
;#DEFINE      c1          PORTB, 2      ;           Ausgang
;#DEFINE      d1          PORTB, 3      ;           Ausgang
;#####
;#DEFINE      a2          PORTB, 4      ;           Ausgang
;#DEFINE      b2          PORTB, 5      ;           Ausgang
;#DEFINE      c2          PORTB, 6      ;           Ausgang
;#DEFINE      d2          PORTB, 7      ;           Ausgang
;#####
;#DEFINE      a_MAX_DATA   PORTB, 0      ; MAX-7219-DATA Ausgang
;#DEFINE      a_MAX_LOAD   PORTB, 1      ; MAX-7219-LOAD Ausgang
;#DEFINE      a_MAX_CLOCK  PORTB, 2      ; MAX-7219-CLOCK Ausgang
;#####
;#DEFINE      bank1       STATUS, RP0
;#####

;*****
;
; * Programmstart *
;*****

ORG    H'00'    ; Das Programm wird ab Speicherstelle 0 in den
                ; Speicher geschrieben
GOTO   init     ; Springe zur Grundinitialisierung der Ports A und B

;*****
;
; * Initialisierung *
;*****

init      BSF      bank1    ; wechsele zu Registerbank 1 (spezielle Register)

                MOVLW      B'00000001'
                MOVWF      TRISA ; RA0 Eingang (RA1 bis RA7 sind Ausgänge)
                MOVLW      B'00000000'
```

MOVWF TRISB ; RB0 bis RB7 sind Ausgänge

; Die Register TRISA und TRISB legen fest, welche Bits in den jeweiligen Ports
; Ein- bzw. Ausgänge sind. Eine '1' an der entsprechenden Stelle setzt das Bit
; des Ports als Eingang eine '0' setzt das Bit als Ausgang.

BCF bank1 ; wechsele zu Registerbank 0
; (normaler Speicherbereich)

CLRF PORTA ; Port A löschen
CLRF PORTB ; Port B löschen

;PS2 - Übertragungsregister

;

CLRF sIDLE

MOVLW B'00000001'
MOVWF sSTART ;START-Befehl (00000001) = H'01'

MOVLW B'01000010'
MOVWF sTYPE ;GET-TYPE-Befehl (01000010) = H'42'

;PS2 - Vergleichsregister

;

MOVLW B'01011010' ; = 0x5A = Status des Controller wenn bereit
MOVWF rSTATUS ;Register wird zum Vergleichen benutzt
;wenn Controller nicht bereit ist soll von
;er zurück zu main springen und die
;Anzeige löschen

;MAX - Register zur Initialisierung

;

;Ich haette auch alle (hab ich auch in einer früheren Programmversion)
;Initialisierungs Adressen/Befehle in Konstanten schreiben können. Da ich aber
;bei der Programmminimierung mir eine Senderoutine überlegt habe, die mit einer
;dynamischen Tabelle die komplette Übertragung (Initialisierung + Anzeige) ab
;arbeitet verwende ich jetzt Register.

MOVLW B'00001011' ; 0x0B Digitsanzahl Adresse
MOVWF m_DIGITS_adr
MOVLW B'00000100' ; 0x01 Anzahl der aktivierten Digits
MOVWF m_DIGITS ; ->Digit 0 bis 4<-

MOVLW B'00001100' ; 0x0C SHUTDOWN Adresse
MOVWF m_SHUTDOWN_adr
MOVLW B'00000001' ; 0x01 MAX Shutdownbefehl
MOVWF m_SHUTDOWN; ->Normal Operation<-

MOVLW B'00001111' ; 0x0F Testbefehlsadresse
MOVWF m_TEST_adr
MOVLW B'00000000' ; 0x00 MAX Testmodus
MOVWF m_TEST ; ->kein Testmodus<-

MOVLW B'00001001' ; 0x09 Digit Kodierungsmodus Adresse
MOVWF m_D_MODUS_adr
MOVLW B'00000000' ; 0x00 MAX Digit Kodierungsmodus
MOVWF m_D_MODUS ;->keine Kodierung<-

MOVLW B'00001010' ; 0x0A MAX Helligkeitssteuerung Adresse
MOVWF m_HELLIGKEIT_adr
MOVLW B'00001111' ; 0x15 MAX Helligkeitssteuerung

```
MOVWF    m_HELLIGKEIT    ; -> Digit 31 / 32 (max ON)<-
```

```
;MAX - Register zur Digitzuweisung
```

```
;
```

```
;Könnte auch mit einem Zaehler gemacht werden, aber ich möchte auswählen
```

```
;wo welches Register angezeigt werden soll und auserdem ist es jetzt möglich
```

```
;die Initialisierung und die Anzeige in einem Unterprogramm ab zu arbeiten
```

```
MOVLW    B'00000001'    ; LINKS - Digit-Adresse
MOVWF    m_LEFT_adr     ; -> Digit 0
```

```
MOVLW    B'00000010'    ; RECHTS - Digit-Adresse
MOVWF    m_RIGHT_adr    ; -> Digit 1
```

```
MOVLW    B'00000011'    ; Joystick rechts, x-Achse - Digit-Adresse
MOVWF    m_RJoyX_adr    ; -> Digit 2
```

```
MOVLW    B'00000100'    ; Joystick rechts, y-Achse - Digit-Adresse
MOVWF    m_RJoyY_adr    ; -> Digit 3
```

```
MOVLW    B'00000101'    ; Joystick links, x-Achse - Digit-Adresse
MOVWF    m_LJoyX_adr    ; -> Digit 4
```

```
MOVLW    B'00000110'    ; Joystick links, y-Achse - Digit-Adresse
MOVWF    m_LJoyY_adr    ; -> Digit 5
```

```
;MAX - Starteinstellungen
```

```
;
```

```
BSF      a_MAX_LOAD    ; MAX-Baustein soll am Anfang nicht im ...
```

```
BSF      a_MAX_CLOCK    ; ...Übertragungsmodus sein
```

```
*****
```

```
;* Hauptprogramm *
```

```
*****
```

```
main
```

```
CLRF     m_display_r    ;Display Bits löschen
CLRF     m_display_c    ;Display Bits löschen
CLRF     m_display_l    ;Display Bits löschen
```

```
BSF      a_ATT          ; ATT auf HIGH, Controller wird abgewählt
                          ; (ignoriert alle Daten)
```

```
CALL     UP_wait_25us
```

```
BSF      a_CLOCK        ; Clock HIGH
```

```
BSF      a_COMMAND      ; Command HIGH
```

```
BCF      a_ATT          ; ATT auf LOW das der Controller
                          ; die Daten annimmt
```

```
CALL     UP_wait_50us    ; Warteschlange bis Controller bereit ist
```

```
#####
```

```
BSF      a1;;;;;;;;;;----DEBUGGING
```

```
CALL     UP_PSC_send_tabelle
```

```
COMF     gLEFT,F        ;Speicherzellen werden invertiert ( 1 zu 0
```

```
COMF     gRIGHT,F       ;und 0 zu 1). Weil wenn eine Taste gedrückt
```

```
ist dann liegt ein 0 Signal am PIC an
```

```
MOVF     rSTATUS,W      ;Controller Status auf Zustand "bereit"
```

```
SUBWF    gSTATUS        ;prüfen...
```

```

    BTFSS STATUS,Z           ;Wenn breit, Zerobit = 1
    GOTO nicht_bereit        ;<- Controller nicht bereit...
                                ;<- Controller bereit...

    MOVLW    D'1'
    MOVWF    zaehler         ;falls der Controller wieder
                                ;keine Verbindung hat, das die LEDs
                                ;auf 0 gesetzt werden!

;#####
;BCF    d2           ;-----DEBUGGING
;CALL   UP_Display_r ;rechte Display LED auswählen
;CALL   UP_Display_l ;linke Display LED auswählen
;BSF    d2           ;-----DEBUGGING
;#####

;#####
max
;    BSF    d2           ;-----DEBUGGING
;    CALL   UP_MAX_send_tabelle ;Ausgabe
;    BCF    d2           ;-----DEBUGGING
;#####

GOTO main                    ;Springe wieder an den Anfang zurück

nicht_bereit
    MOVF    zaehler,F
    BTFSC STATUS,Z
    GOTO    main
    DECF    zaehler
    GOTO    max

;*****
;* Unterprogramme *
;*****
;#####
;#####
UP_PSC_send_tabelle

    MOVLW    D'0'
    MOVWF    zaehler_tab

PSC_st1 MOVLW    D'8'
    MOVWF    zaehler_bit

    BCF     STATUS, IRP    ; Bank 0 oder 1 (alle Register aus der
                                ; Tabelle sind in Bank 0)

    MOVF    zaehler_tab,W    ;wird mit dem PCL addiert

    CALL    tab_PSC          ;Öffnet die "dynamische Tabelle"
                                ;Im Workregister steht jetzt die Adresse
                                ;des Registers dessen Inhalt
                                ;Übertragen werden soll

    MOVWF    FSR             ;der Zeiger zeigt jetzt auf das Register
                                ;dessen Inhalt Übertragen werden soll

```

```

MOVF INDF,W      ;mit dem virtuellen Register INDF kann der
                  ;Inhalt des Registers ausgegeben werden,
                  ;auf das der Zeiger FSR zeigt

MOVWF    sPSC      ;Inhalt wird in sende Register kopiert,
                  ;da ein weiteres Register aus der Tabelle
                  ;benötigt wird um die empfangenen Bits zu
                  ;speichern

INCF    zaehler_tab ;nächste Tabellenreihe

MOVF    zaehler_tab,W ;wird mit dem PCL addiert
CALL    tab_PSC      ;Öffnet die "dynamische Tabelle"
                  ;Im Workregister steht jetzt die Adresse
                  ;des Registers in das geschrieben wird

MOVWF    FSR        ;der Zeiger zeigt jetzt auf das Register
                  ;in das die empfangenen Bits gespeichert
                  ;werden
                  ;-
                  ;mit dem virtuellen Register INDF kann der
                  ;Inhalt des Registers geändert werden,
                  ;auf das der Zeiger FSR zeigt

PSC_1    RRF    sPSC      ;verschiebe sende Register nach rechts...
          BTFSS STATUS,C      ;prüfe Wertigkeit des rausgeworfenen Bits
          GOTO   PSC_sende_0  ;(STATUS,C=0)-> Übertrage 0 an Controller...
          GOTO   PSC_sende_1  ;(STATUS,C=1)-> ansonsten Übertrage '1'

PSC_2    DECF    zaehler_bit
          MOVF    zaehler_bit,F ; Zähler ...
          BTFSS   STATUS,Z      ; ... auf 0 prüfen,
          GOTO    PSC_1         ; <-- Zähler != 0 --> nächstes Bit übertrag.
                  ; <-- Zähler == 0

          INCF    zaehler_tab
          MOVLW    D'18'        ;PS2-Analog-Mode Controller
          SUBWF    zaehler_tab,W
          BTFSS   STATUS,Z
          GOTO    PSC_st1      ;zaehler_tab < 18
          RETURN              ;zaehler_tab = 18
;#####
PSC_sende_0
          BCF     a_COMMAND    ;auf 0 setzten
          BCF     a_CLOCK      ;negative Flanke (Controller ließt COMMAND)
          CALL    UP_wait_25us ;aber erst nach 25µs !!!
          BTFSC   e_DATA       ;e_DATA prüfen
          GOTO    INDF_0_write_1 ;empfangenes Bit in Carry-Flag setzen
          GOTO    INDF_0_write_0 ;empfangenes Bit in Carry-Flag setzen
PSC_sende_0_next
          RRF     INDF          ;empfangenes Bit in Register hinzufügen
          BSF     a_CLOCK
          CALL    UP_wait_25us
          GOTO    PSC_2
;#####
INDF_0_write_1
          BSF     STATUS,C
          GOTO    PSC_sende_0_next

INDF_0_write_0      ;zur Übersicht, hier geschrieben!!!

```

```

        BCF    STATUS,C
        GOTO   PSC_sende_0_next
;#####
PSC_sende_1
        BSF    a_COMMAND    ;auf 1 setzen
        BCF    a_CLOCK      ;negative Flanke (Controller ließt COMMAND)
        CALL   UP_wait_25us  ;aber erst nach 4µs !!!
        BTFSC  e_DATA       ;e_DATA prüfen
        GOTO   INDF_1_write_1 ;empfangenes Bit in Carry-Flag setzen
        GOTO   INDF_1_write_0 ;empfangenes Bit in Carry-Flag setzen
PSC_sende_1_next
        RRF    gTYPE        ;empfangenes Bit in Register hinzufügen
        BSF    a_CLOCK
        CALL   UP_wait_25us
        GOTO   PSC_2
;#####
INDF_1_write_1
        BSF    STATUS,C
        GOTO   PSC_sende_1_next

INDF_1_write_0                ;zur Übersicht, hier geschrieben!!!
        BCF    STATUS,C
        GOTO   PSC_sende_1_next
;#####
UP_Display_r                  ;Positionsberechnung Joystick rechts
        MOVLW   D'85'
        SUBWFgRJoyX,W
        BTFSC  STATUS,C
        GOTO   x_groesser_85

x_kleiner_gleich_85
        MOVLW   D'85'
        SUBWFgRJoyY,W
        BTFSC  STATUS,C
        GOTO   y_groesser_85
y_kleiner_gleich_85
        BSF    m_display_r,6
        RETURN

y_groesser_85
        MOVLW   D'170'
        SUBWFgRJoyY,W
        BTFSC  STATUS,C
        GOTO   y_groesser_170
y_kleiner_gleich_170
        BSF    m_display_r,7
        RETURN

y_groesser_170
        BSF    m_display_r,0
        RETURN
;#####
;#####
x_groesser_85
        MOVLW   D'170'
        SUBWFgRJoyX,W
        BTFSC  STATUS,C
        GOTO   x_groesser_170
x_kleiner_gleich_170
        MOVLW   D'85'

```



```

        SUBWFgRJoyY,W
        BTFSC STATUS,C
        GOTO y_2_groesser_85
y_2_kleiner_gleich_85
        BSF    m_display_r,5
        RETURN

```

```

y_2_groesser_85
        MOVLW    D'170'
        SUBWFgRJoyY,W
        BTFSC STATUS,C
        GOTO y_2_groesser_170
y_2_kleiner_gleich_170
        BSF    m_display_c,0
        RETURN

```

```

y_2_groesser_170
        BSF    m_display_r,1
        RETURN

```

```

;#####
;#####

```

```

x_groesser_170
        MOVLW    D'85'
        SUBWFgRJoyY,W
        BTFSC STATUS,C
        GOTO y_3_groesser_85
y_3_kleiner_gleich_85
        BSF    m_display_r,4
        RETURN

```

```

y_3_groesser_85
        MOVLW    D'170'
        SUBWFgRJoyY,W
        BTFSC STATUS,C
        GOTO y_3_groesser_170
y_3_kleiner_gleich_170
        BSF    m_display_r,3
        RETURN

```

```

y_3_groesser_170
        BSF    m_display_r,2
        RETURN

```

```

;#####

```

```

UP_Display_1                ;Positionsberechnung Joystick links
        MOVLW    D'85'                ;links und rechts kann auch wieder mit
        SUBWFgLJoyX,W                ;Tabelle minimiert weden!
        BTFSC STATUS,C
        GOTO _x_groesser_85

```

```

_x_kleiner_gleich_85
        MOVLW    D'85'
        SUBWFgLJoyY,W
        BTFSC STATUS,C
        GOTO _y_groesser_85
_y_kleiner_gleich_85
        BSF    m_display_l,6
        RETURN

```

```
_y_groesser_85
    MOVLW    D'170'
    SUBWFgLJoyY,W
    BTFSC STATUS,C
    GOTO    _y_groesser_170
_y_kleiner_gleich_170
    BSF      m_display_1,7
    RETURN

_y_groesser_170
    BSF      m_display_1,0
    RETURN
;#####
;#####
_x_groesser_85
    MOVLW    D'170'
    SUBWFgLJoyX,W
    BTFSC STATUS,C
    GOTO    _x_groesser_170
_x_kleiner_gleich_170
    MOVLW    D'85'
    SUBWFgLJoyY,W
    BTFSC STATUS,C
    GOTO    _y_2_groesser_85
_y_2_kleiner_gleich_85
    BSF      m_display_1,5
    RETURN

_y_2_groesser_85
    MOVLW    D'170'
    SUBWFgLJoyY,W
    BTFSC STATUS,C
    GOTO    _y_2_groesser_170
_y_2_kleiner_gleich_170
    BSF      m_display_c,1
    RETURN

_y_2_groesser_170
    BSF      m_display_1,1
    RETURN

;#####
;#####
_x_groesser_170
    MOVLW    D'85'
    SUBWFgLJoyY,W
    BTFSC STATUS,C
    GOTO    _y_3_groesser_85
_y_3_kleiner_gleich_85
    BSF      m_display_1,4
    RETURN

_y_3_groesser_85
    MOVLW    D'170'
    SUBWFgLJoyY,W
    BTFSC STATUS,C
```

```

        GOTO  _y_3_groesser_170
_y_3_kleiner_gleich_170
        BSF    m_display_1,3
        RETURN

_y_3_groesser_170
        BSF    m_display_1,2
        RETURN
;#####
UP_MAX_send_tabelle

        MOVLW   D'0'
        MOVWF   zaehler_tab      ;Wenn zaehler_tab ungerade, 2.Byte gesendet
                                   ;-> LOAD auf 1

MAX_st1      MOVLW   D'8'
        MOVWF   zaehler_bit

        BCF     STATUS,IRP      ; Bank 0 oder 1

        MOVF    zaehler_tab,W   ;wird mit dem PCL addiert

        CALL    tab_MAX          ;Öffnet die "dynamische Tabelle"
                                   ;Im Workregister steht jetzt die Adresse
                                   ;des Registers dessen Inhalt
                                   ;Übertragen werden soll

        MOVWF    FSR            ;der Zeiger zeigt jetzt auf das Register
                                   ;dessen Inhalt Übertragen werden soll

        MOVFW    INDF           ;mit dem virtuellen Register INDF kann der
                                   ;Inhalt des Registers ausgegeben werden,
                                   ;auf das der Zeiger FSR zeigt
;MOVWF    PORTB      ; #####DEBUGGING#####
;CALL    UP_wait_05s ; #####DEBUGGING#####
        MOVWF    sMAX

        BCF     a_MAX_LOAD      ;Vor der Übertragung von 2 Bytes auf LOW

MAX_st2      RLF     sMAX        ;MSB zuerst...
        BTFSC   STATUS,C
        GOTO    MAX_send_tabelle_1 ;MAX-Datenleitung auf HIGH (Übertrage 1)
        BCF     a_MAX_DATA      ;MAX-Datenleitung auf LOW   (Übertrage 0)

MAX_st3      BCF     a_MAX_CLOCK ;
;CALL    UP_wait_05s ; #####DEBUGGING#####
        BSF     a_MAX_CLOCK      ;Positive Flanke erzeugen
        BCF     a_MAX_CLOCK

        DECF    zaehler_bit
        MOVF    zaehler_bit,F    ; Zähler ...
        BTFSS   STATUS,Z         ; ... auf 0 prüfen,
        GOTO    MAX_st2          ; <-- Zähler != 0 --> nächstes Bit übertrag.
                                   ; <-- Zähler == 0

        BTFSC   zaehler_tab,0    ; nach jedem 2. Byte LOAD auf 1
        BSF     a_MAX_LOAD      ; <-- Zähler_tab == ungerade -> 2.Byte
                                   ; <-- Zähler_tab == gerade -> 1.Byte

        INCF    zaehler_tab
        MOVLW    D'20'           ;Platine, mit Joystickberechnung 20Bytes

```

```

        SUBWF zaehler_tab,W
        BTFSS STATUS,Z
        GOTO MAX_st1          ;zaehler_tab < 22
        RETURN                ;zaehler_tab = 22

MAX_send_tabelle_1
        BSF    a_MAX_DATA
        GOTO   MAX_st3

;#####
;##### Warteschleifen #####
;#####

;#####*****#####
;##### ATTENTION Warteschleife #####
;UP_wait_20us
;
;          ;100 cycles
;        movlw 0x21
;        movwf wait_20us
;loop20us_0
;        decfsz wait_20us, f
;        goto  loop20us_0
;RETURN
;##### ATTENTION Warteschleife 2 #####
UP_wait_50us
;
;          ;250 cycles
;        movlw 0x53
;        movwf wait_50us
loop50us_0
;        decfsz wait_50us, f
;        goto  loop50us_0
RETURN
;#####*****#####
;##### CLOCK LOW - CLOCK HIGH Warteschleife #####
;UP_wait_5us          ;UP_wait_5us
;
;          ;25 cycles
;        movlw 0x08
;        movwf wait_5us
;loop5us_0
;        decfsz wait_5us, f
;        goto  loop5us_0
;RETURN
;##### CLOCK LOW - CLOCK HIGH Warteschleife 2 #####
UP_wait_25us          ;UP_wait_5us
;
;          ;124 cycles
;        movlw 0x29
;        movwf wait_25us
loop25us_0
;        decfsz wait_25us, f
;        goto  loop25us_0

;
;          ;1 cycle
        nop
RETURN
;#####*****#####
;##### DEBUGGING Warteschleife #####
;UP_wait_05s
;
;          ;249999 cycles
;        movlw 0x16
;        movwf wait_05s
;        movlw 0x74
;        movwf wait_05s_1

```

```

;      movlw 0x06
;      movwf wait_05s_2
;loop05s_0
;      decfsz wait_05s, f
;      goto $+2
;      decfsz wait_05s_1, f
;      goto $+2
;      decfsz wait_05s_2, f
;      goto loop05s_0
;
;      ;1 cycle
;      nop
;RETURN
#####
##### Tabellen #####
#####

#####*****#####

##### PSC-Tabelle #####

tab_PSC                                ;Tabellenlänge: 18; -> zaehler_tab max. 18!
    BCF    PCLATH,0                    ;PCLATH auf ...
    BSF    PCLATH,1                    ;... 0x2 stellen
    ADDLW  0x50
    MOVWF  PCL                         ;PCL ist jetzt 0x250 + zaehler_tab

ORG    0x250                          ; Tabelle in Speicherblock 2

    RETLW  sSTART                      ; Playstation Controller - Startbefehl
    RETLW  gSTART                      ; empfange... irgentwas

    RETLW  sTYPE                       ; Playstation Controller - Typebefehl
    RETLW  gTYPE                       ; empfange Type des Controllers

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gSTATUS                     ; empfange Status des Controllers

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gLEFT                       ; empfange Status der Tasten LINKS

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gRIGHT                      ; empfange Status der Tasten RECHTS

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gRJoyX                      ; empfange : Joystick rechts, x-Achse

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gRJoyY                      ; empfange : Joystick rechts, y-Achse

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gLJoyX                      ; empfange : Joystick links, x-Achse

    RETLW  sIDLE                       ; übertrage 0 Signal
    RETLW  gLJoyY                      ; empfange : Joystick links, y-Achse

#####*****#####
##### MAX-Tabelle #####

tab_MAX                                ;Tabellenlänge: 22; -> zaehler_tab max. 22!
    BSF    PCLATH,0                    ;PCLATH auf ...

```

```
BSF    PCLATH,1          ;... 0x3 stellen
MOVWF  PCL               ;PCL ist jetzt 0x300 + zaehler_tab

ORG    0x300             ; Tabelle in Speicherblock 3

RETLW  m_DIGITS_adr      ; Anzahl der genutzten Digits
RETLW  m_DIGITS          ;

RETLW  m_SHUTDOWN_adr    ; Normalmodus oder Ausgeschaltet
RETLW  m_SHUTDOWN        ;

RETLW  m_TEST_adr        ; Normalmodus oder Displaytest
RETLW  m_TEST            ;

RETLW  m_D_MODUS_adr     ; Dekodierungsmodus
RETLW  m_D_MODUS         ;

RETLW  m_HELLIGKEIT_adr  ; Helligkeitssteuerung der LED's
RETLW  m_HELLIGKEIT      ;

RETLW  m_LEFT_adr        ; Tasten LINKS - Digit-Adresse
RETLW  gLEFT             ; Tasten LINKS

RETLW  m_RIGHT_adr       ; Tasten RECHTS - Digit-Adresse
RETLW  gRIGHT            ; Tasten RECHTS

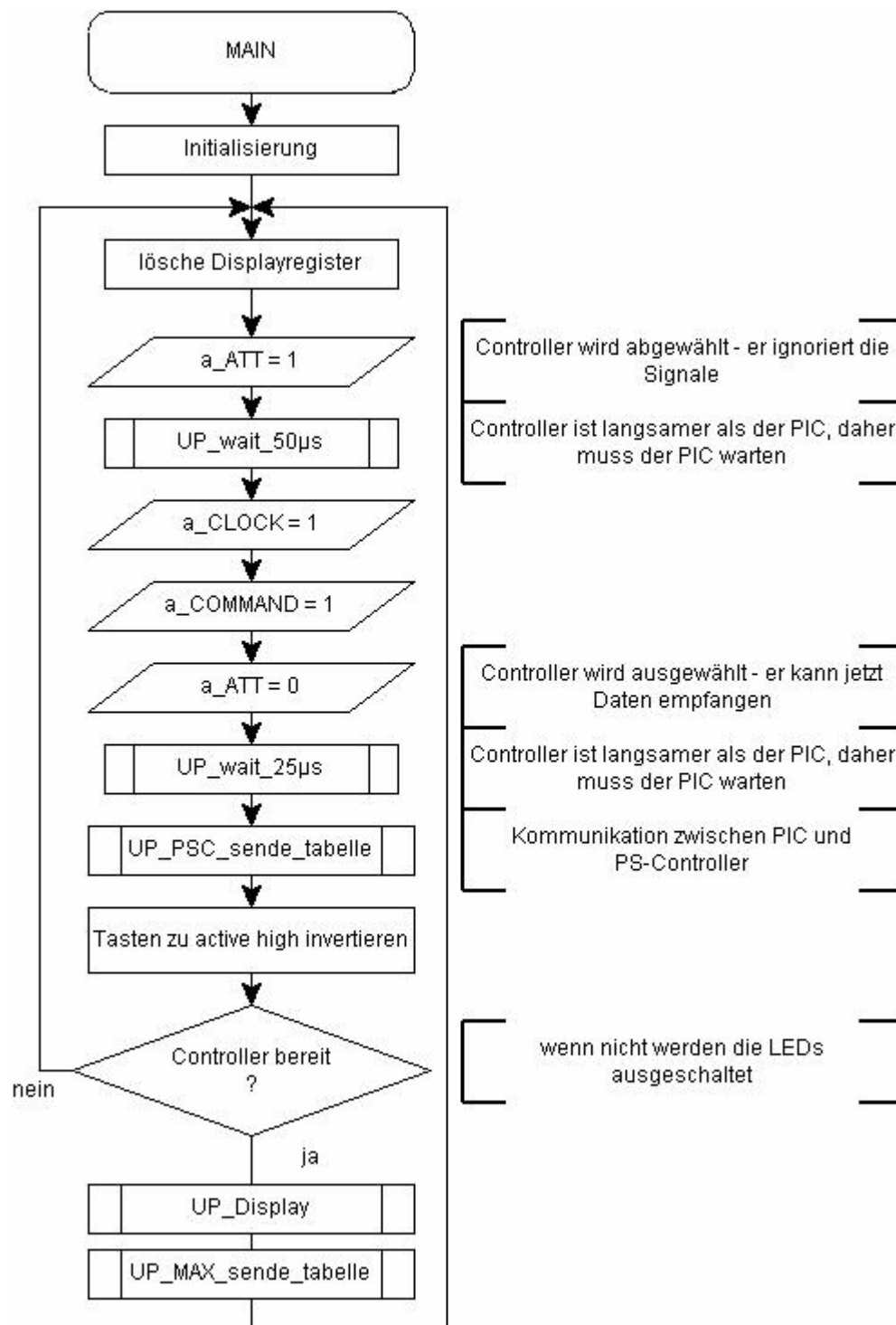
RETLW  m_RJoyX_adr       ; Joystick rechts, x-Achse - Digit-Adresse
RETLW  m_display_r       ; Joystick Positionsdisplay, rechts

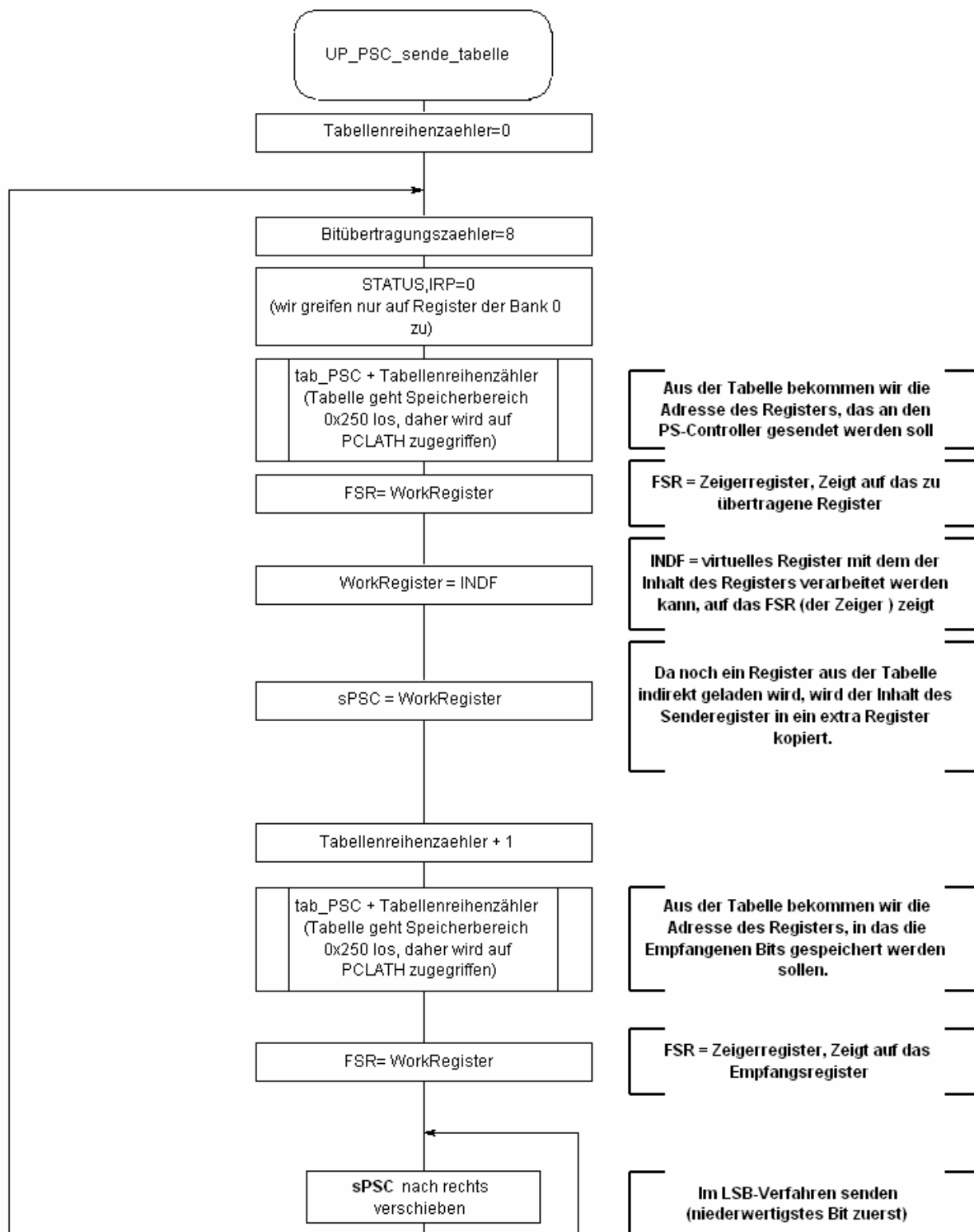
RETLW  m_RJoyY_adr       ; Joystick rechts, y-Achse - Digit-Adresse
RETLW  m_display_l       ; Joystick Positionsdisplay, links

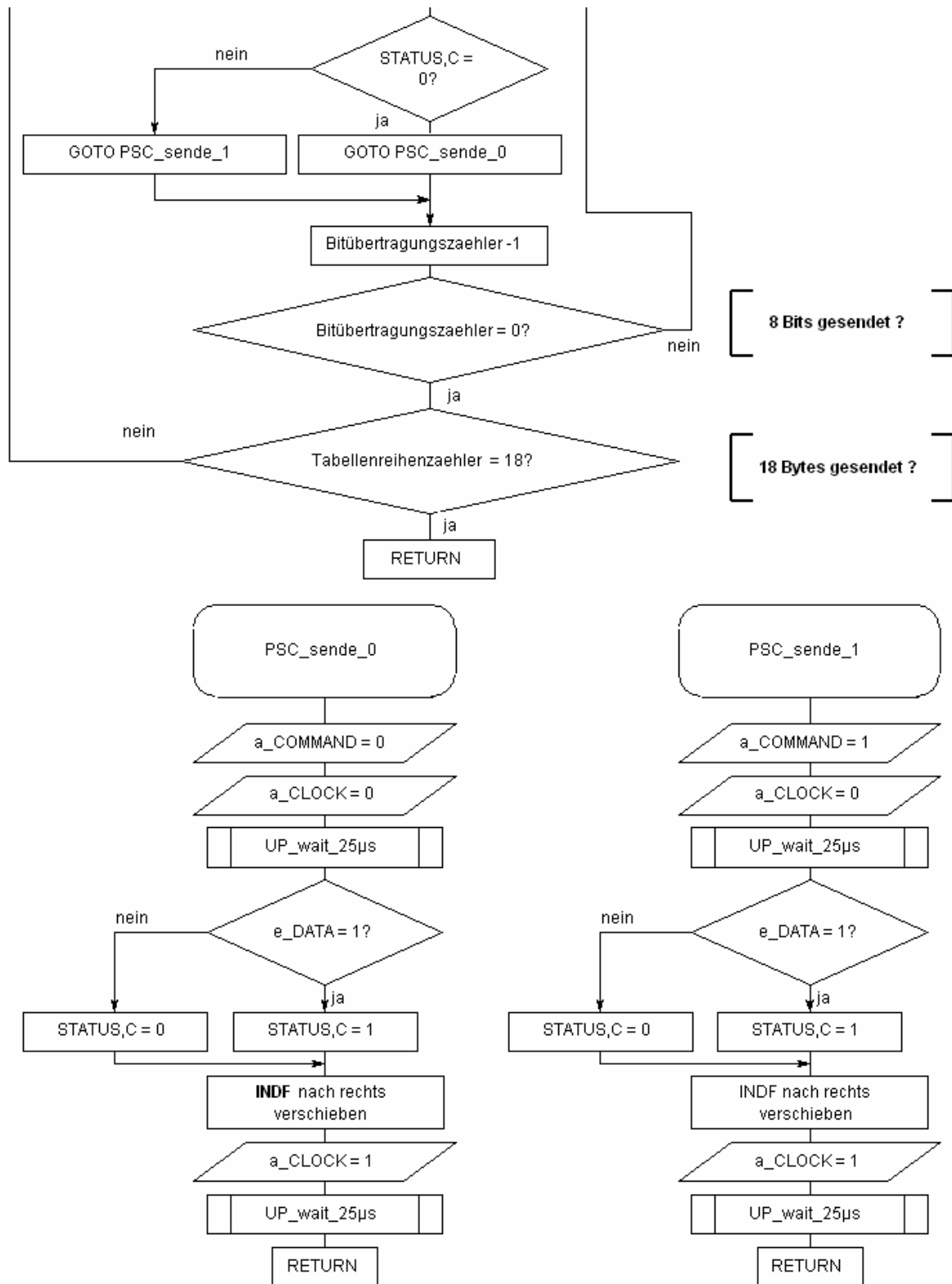
RETLW  m_LJoyX_adr       ; Joystick links, x-Achse - Digit-Adresse
RETLW  m_display_c       ; Joystick Positionsdisplay, recht&links...
                        ; mittlere LEDs

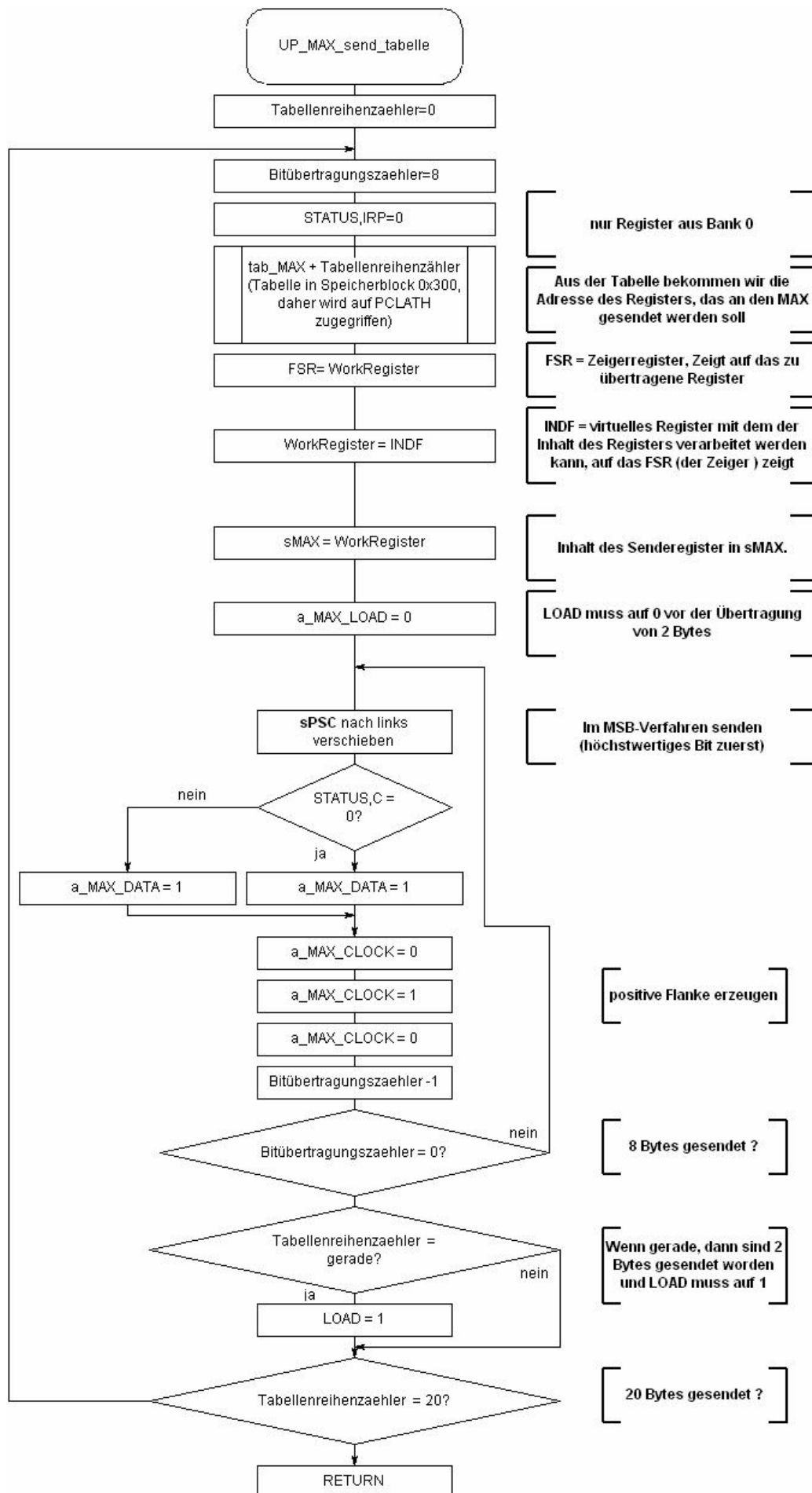
END
```

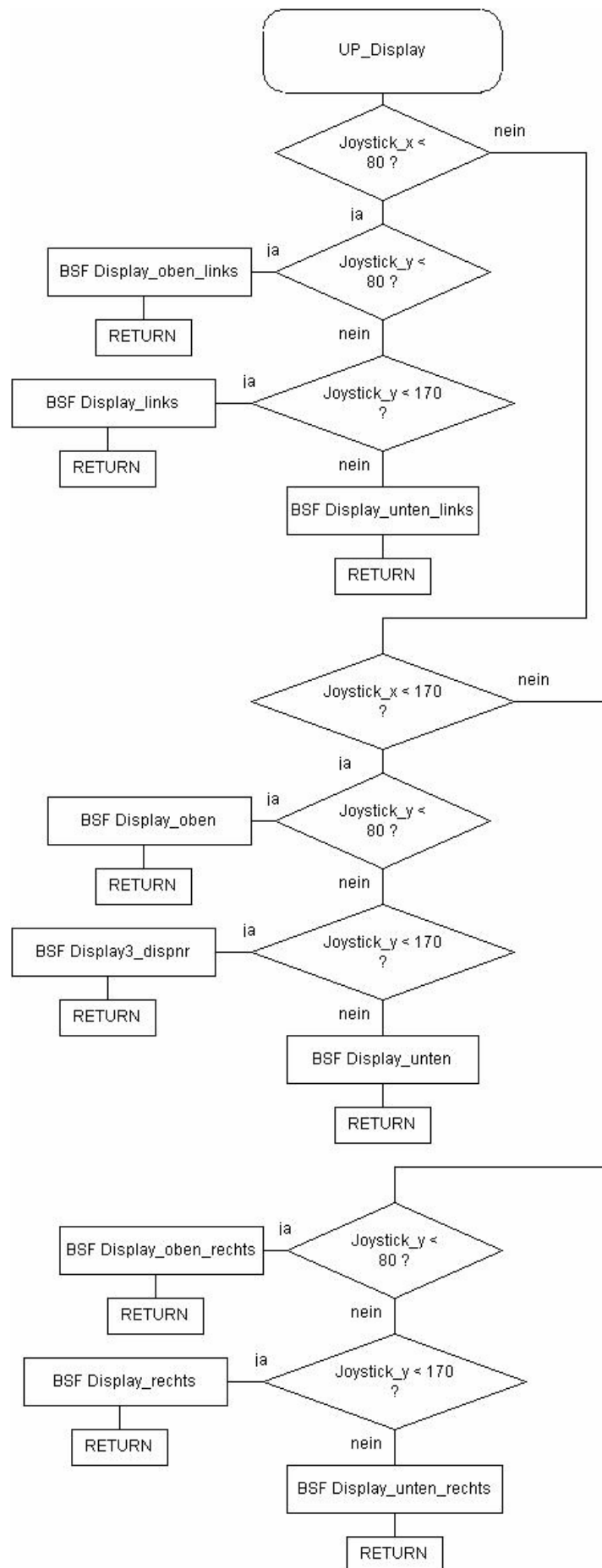
3.2 Flussdiagramme











4.0 Bilanz

Durch die Probleme, die ich in der ersten Zeit der Projektphase nicht lösen konnte, befürchtete ich, dass das Projekt nicht erfolgreich abgeschlossen werden könnte. Trotzdem versuchte ich weiterhin die Programmierlösung zu finden. Als der Fehler gefunden und beseitigt war, ging es endlich weiter.

Ich hatte mir schon im Voraus überlegt, was ich mit dem Controller alles steuern könnte. Da mein Klassenkamerad aber zu diesem Zeitpunkt Probleme mit dem MAX7219 Display Treiber hatte und mich um Hilfe fragte, wurde mir bewusst, dass die Ansteuerung dieses Bausteins dem Playstation-Controller ähnelt. So wurde vorübergehend aus meinem Projektvorschlag (Anzeige von 8 Tastenzuständen) eine LED Display Treiber geregelte 16 Tasten-Zustandsanzeige mit Positionsanzeige der Joysticks.

Leider konnte ich die Umsetzung dieser Projekterweiterung nicht auf einer Platine fertig stellen. Daher hab ich mich entschlossen die Platine ätzen zu lassen, was sich später als teurer Fehler herausgestellt hat: die Platine funktioniert nur bedingt. Aus diesem Grund bin ich auf mein ursprüngliches Projektvorhaben zurückgekommen. Dennoch kann ich mit Bestimmtheit sagen, dass ich durch diesen Projektablauf einiges hinzugelernt habe (wie man auch auf dem Video erkennen kann). Trotz aller Schwierigkeiten hat mir die Arbeit mit dem Projekt viel Spaß gemacht.

5.0 Quellen

Playstation Controller:

http://www.parallax.com/html_pages/resources/catapps/cat_ps.asp

http://www.geocities.com/digitan000/Hardware/22/e22_page.html

http://www.hardwarebook.info/Sony_Playstation_Controller_Port

http://users.ece.gatech.edu/~hamblen/489X/f04proj/USB_PSX/psx_protocol.html

indirekte Adressierung:

<http://www.roboternetz.de/phpBB2/viewtopic.php?p=264376#264376>

<http://www.sprut.de/electronic/pic/grund/adress.htm#indirekt>

Sonstige:

[http://www.staff.uni-](http://www.staff.uni-bayreuth.de/~btp918/cmt2004/PIC/Handout/06%20Fortgeschrittene%20Programmiertechniken.pdf)

[bayreuth.de/~btp918/cmt2004/PIC/Handout/06%20Fortgeschrittene%20Programmiertechniken.pdf](http://www.staff.uni-bayreuth.de/~btp918/cmt2004/PIC/Handout/06%20Fortgeschrittene%20Programmiertechniken.pdf)

<http://www.piclist.com/techref/microchip/compcon.htm>

Platine:

<http://platinen-design.de/>

- die angefertigte Platine ist nicht funktionsfähig!
- die Bohrungen waren zu klein
- die Platine wurde zu spät gesendet
- die Kontaktfläche um die Bohrungen sind zu dünn

5.1 Versicherung der Eigenleistung

Hiermit versichere ich, Aryan Layes, dass ich die vorliegende Projektarbeit selbstständig, ohne fremde Hilfe und nur mit den angegebenen Hilfsmittel erstellt habe.

Datum, Unterschrift: _____