7.5 Servosteuerung

Servos („Rudermaschinen“) aus dem Modellbaubereich sind trotz der Betriebsspannung von 5 Volt besonders kräftige Antriebe. Ihr hohes Drehmoment wird durch ein Untersetzungsgetriebe erreicht. Ein Servoantrieb besteht aus einem Motor, einem auf der Achse angebrachten Positions- oder Winkelsensor (im einfachsten Fall ist das ein Potentiometer) und einer Regelelektronik mit Sollwert-Eingang. Diese vergleicht den eingegebenen Sollwert mit dem Istwert des Sensors. Stimmen beide nicht überein, so lässt die Regelung den Motor zu der Position laufen, bei der Istwert und Sollwert gleich sind. Diese elektronische Motorsteuerung ermöglicht nicht nur ein sehr feinfühliges und genaues Stellen des Antriebs in eine bestimmte Position, sie sorgt auch für ein kräftiges Gegenmoment gegen Rückstellversuche der Last und hält damit die gewünschte Position.

Je nach verwendetem Servo sind Stellzeiten bis herab auf 0,08 s über einen Stellweg von 60 Grad erreichbar. Die modernste Version, der Digitalservo, ist statt der bei analog arbeitenden Servos passiv arbeitenden Servosteuerung mit einem Mikroprozessor bestückt. Durch eine hohe Taktfrequenz kann der Antriebsmotor besonders schnell und in allen Lagen mit vollem Drehmoment arbeiten, was extrem kurze Reaktionszeiten bei gleichmäßiger Kraftentfaltung erlaubt. Die Analog-Servos dagegen verlieren oft gegen Ende des Stellwegs an Drehmoment und auch Drehgeschwindigkeit. Ein weiterer Vorteil des Digitalservos ist das aktive Gegensteuern durch den Prozessor bei Rückstellversuchen der Last. Nachteilig bei beiden Typen ist der relativ kleine Stellweg und das laute Arbeitsgeräusch. Die Stellkraft (etwa 10 Newton) ist, gemessen an der Größe, beachtlich. Sie sollten aber berücksichtigen, dass die Getriebe bei vielen Modellen aus Plastikzahnrädern bestehen, die bei Überlast sehr schnell verschleißen (gegebenenfalls auf die etwas teureren Modelle mit Metallgetriebe ausweichen). Die Ansteuerung ist bei beiden Typen gleich; die Sollwert-Vorgabe erfolgt über einen längenmodulierten Impuls.

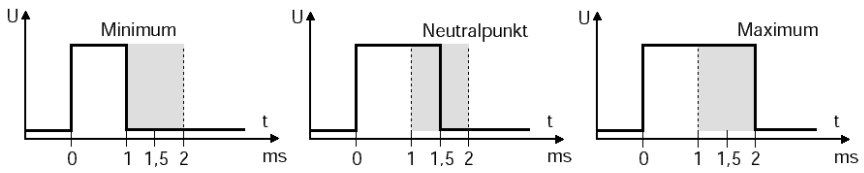

Bild 7.11: Impulstiming von Servos

Der Servo wird alle 20 ms mit kurzen Impulsen von ein bis zwei Millisekunden Dauer angesteuert und damit die Position seines Antriebs bestimmt. Die Impulse müssen im 20-ms-Abstand wiederholt werden, damit ein Servo seine Position beibehält (Bild 7.11). Praktisch alle Servos erwarten übrigens einen positiven Impuls.

Alle genannten Eigenschaften prädestinieren den Modellbau-Servo als Antrieb für andere Verwendungen. Sie lassen sich für beliebige Betätigungsfunktionen verwenden, z. B. das Betätigen eines Riegels, das Aktivieren einer Fütterungsautomatik für Fische, das Öffnen und Schließen von Lüftungsklappen, die computerisierte Einzelbild-Steuerung für Langzeitaufnahmen einer 16-mm-Kamera oder

das Schwenken einer Überwachungskamera. Modellbau-Servos werden heutzutage auch gerne bei experimentellen Robotern eingesetzt. Häufig findet man in einem Roboter eine größere Anzahl davon. Ein „Käfer" mit sechs Beinen benötigt in der Regel mindestens drei Servos pro Bein, und ein einfacher Arm hat mitunter sechs bis sieben.

Wo ich gerade bei der Hardware bin – es gibt leider mehrere unterschiedliche Servo-Anschlusskabel-Systeme, jeder Hersteller hat eine eigene Norm. Die verbreitetsten Steckerformen sind heute die Futaba- und die Graupner-Norm. Die diversen Steckerformen sind in Bild 7.12 zu sehen.

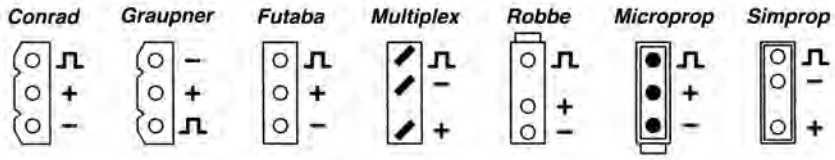

Bild 7.12: Anschluss-Stecker verschiedener Hersteller

Die äußere Form des Steckers, die man bei der Kombination des Servos mit einem Fernsteuerempfänger strikt beachten muss, tangiert beim Anschluss an das PC-Interface nicht, wohl aber die Anschlussbelegung. Denn der Stecker ist unbedingt richtig herum auf die Stiftleiste der Steuerung aufzustecken, sonst kann es zu Schäden kommen. Dabei kann man sich nach den Kabelfarben richten. Diese sind zwar auch nicht einheitlich, folgen aber zumindest einem Grundschema. So ist der Plusanschluss bei allen rot, der Minusanschluss entweder braun, schwarz oder blau. Der Signalanschluss ist entweder gelb, orange, weiß oder violett. Einziger Ausreißer aus diesem Schema ist Simprop. Hier ist der Minusanschluss blau und das Signalkabel schwarz.

Zurück zur Ansteuerung. Es muss also für jedes angeschlossene Servo alle 20 bis 25 ms der Ansteuerimpuls mit seiner Längenvariation generiert werden. Prinzipiell könnte man so eine Servo-Ansteuerung über jeden Paralleport realisieren. Noch besser wäre freilich der Ausgang eines Timers, denn dann wäre der Prozessor maximal entlastet, weil nur noch wenige Steuerwerte an den Timer ausgegeben werden müssen. Leider sind bei einem PC die Timer bereits fest vergeben. Die Ansteuerung über den Parallelport kann unseren PC bei einer größeren Anzahl von Servos überlasten. Abgesehen davon ist Linux kein Echtzeit-Betriebssystem, und so kämen die Servos aller Wahrscheinlichkeit nach ins Flattern. Dieses Problem lässt sich mit Hilfe spezieller Schaltkreise lösen.

Im Folgenden will ich Ihnen zwei Vertreter dieser Gattung vorstellen: den FT 639 von Elab Inc. (vormals FerreTronics) und den MIC 800 von Mictronics. Zu finden sind die Firmen unter folgenden Webadressen:

- Elab Inc., USA: *http://www.elabinc.com*
  Vertrieb: direkt von Elab Inc. aus den USA oder in Deutschland über Systec (*http://www.sytec-net.de*)
- Mictronics, Frankreich: *http://www.mictronics.com*
  Vertrieb: TechDesign Electronics, Belgien *http://www.techdesign.be*

Beide Bausteine werden über die serielle Schnittstelle angesteuert. Das ist von Vorteil, weil die Ansteuerung recht einfach vonstatten geht. Der Nachteil zeigt sich, wenn man schnell aufeinanderfolgend kleine Änderungen des Servoausschlags vornehmen will (Feinpositionierung, Ausregeln von externen Einflüssen), denn dann wird es gegebenenfalls zu langsam. Aber selbst beim FT 639 schafft man es, alle fünf Servos 25 mal pro Sekunde anzusteuern.

7.5.1 Der FT 639

Der FT 639 ermöglicht mit nur acht Pins die Steuerung von fünf Servos. Die Schaltung in Bild 7.13 zeigt, wie einfach der Anschluss eigentlich ist. Das IC benötigt nur einen Entkopplungskondensator und eine Pegelanpassung an serielle Pegel mittels Spannungsteiler. Die Diode schützt das IC vor den negativen Spannungen der V.24-Schnittstelle. Bei Mikrokontrollern, die über eine TTL-kompatible serielle Schnittstelle verfügen, können diese Bauteile entfallen, dafür ist gegebenenfalls ein Inverter nötig, weil die V.24-Signale ja genau „andersherum“ ankommen. Eine entsprechende Schaltung finden Sie beim MIC 800 im folgenden Abschnitt. Zur Versorgung der Schaltung ist im Schaltplan noch ein 7805 vorgesehen. Auf diese Weise läuft die Steuerung mit Betriebsspannungen zwischen 9 V und 24 V.

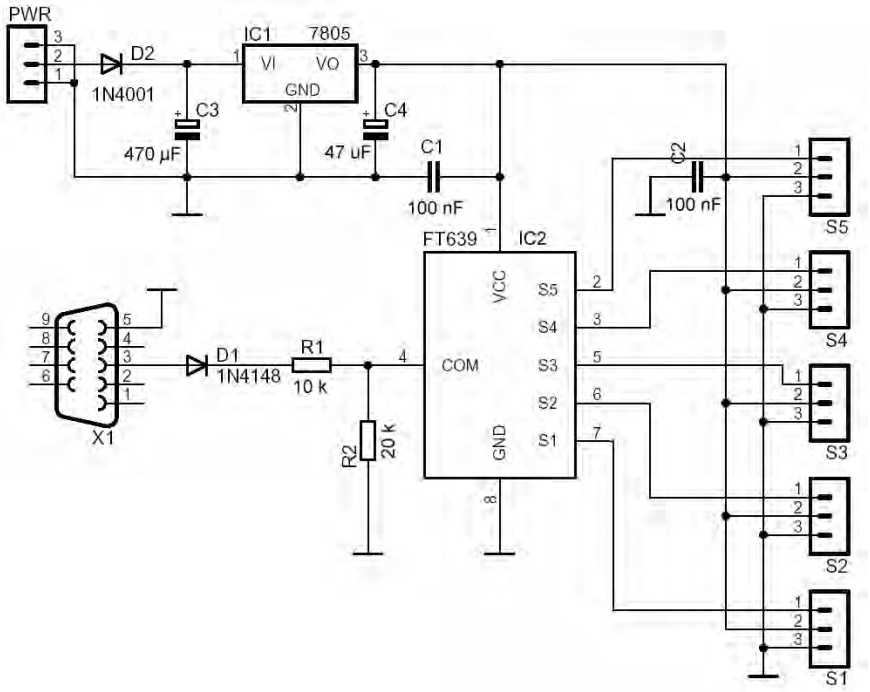

Bild 7.13: Schaltung mit dem FT 639

Das Datenformat ist einfach: Die Schnittstelle wird auf 2400 bps, kein Paritätsbit und 1 Stoppbit eingestellt. Der Chip startet immer im Setup-Modus und schaltet dann in den Active-Modus.

Tabelle 7.3: Startposition (Header Length) beim FT639

Header	Short Pulse	Long Pulse	Befehl binär	Befehl dezimal
0	0.147 ms	0.237 ms	01100000	96
1	0.219 ms	0.357 ms	01100001	97
2	0.291 ms	0.477 ms	01100010	98
3	0.363 ms	0.597 ms	01100011	99
4	0.435 ms	0.717 ms	01100100	100
5	0.507 ms	0.837 ms	01100101	101
6	0.579 ms	0.957 ms	01100110	102
7	0.651 ms	1.077 ms	01100111	103
8	0.723 ms	1.197 ms	01101000	104
9	0.795 ms	1.317 ms	01101001	105
10	0.867 ms	1.437 ms	01101010	106
11	0.939 ms	1.557 ms	01101011	107
12	1.011 ms	1.677 ms	01101100	108
13	1.083 ms	1.797 ms	01101101	109
14	1.155 ms	1.917 ms	01101110	110
15	1.227 ms	2.037 ms	01101111	111

Im Setup-Modus kann der Bediener zwischen zwei Auslenkungen wählen: mittels „Short Pulse“ kann eine von 256 möglichen Positionen in einem Bereich von 0° bis 90° gesetzt werden, 0° bis 180° in 256 Stufen werden mit Hilfe von „Long Pulse“ erreicht. Die Einstellung hängt vom verwendeten Servo ab – bei kürzerem Weg ist so eine feinere Auflösung möglich. Über „Header Length“ kann die Startposition individueller Servos erreicht werden (siehe Tabelle 7.3). Mein Rat ist, die Voreinstellung zu belassen, denn die maximale Impulslänge wird ebenfalls beeinflusst, und manche Servos laufen dann an die Begrenzung (schlecht fürs Getriebe).

Tabelle 7.4: Befehle des FT 639

Befehl	binär	dezimal
Aktiv-Mode	01110101	117
Short Pulse	01010101	85
Long Pulse	01011010	90
Setup-Mode	01111010	122
Header length	0110xxx	96 + xxxx

Der vierte mögliche Setup-Befehl, 01110101, schaltet den Baustein in den aktiven Modus. Und das war es auch schon mit dem Setup, mehr gibt es nicht. Alle Kommandos sind in Tabelle 7.4 aufgeführt.

Nach dem Einschalten werden in der Regel einige wenige Setup-Befehle geschickt, wobei der letzte in den aktiven Modus umschaltet. Vom Aktivmodus zurück in den Setup-Modus gelangt man mit dem Befehlswort 01111010.

Im aktiven Modus werden immer zwei Bytes an den FT 639 geschickt, wobei das erste Byte immer eine 0 an der höchstwertigen Stelle besitzt und das zweite immer eine 1. Der Aufbau eines solchen 2-Byte-Befehls ist:

0SSSLLLL **1**SSSHHHH

- SSS ist die Nummer des Servos (Servo 1: SSS = 000, Servo 2: SSS = 001, Servo3: SSS = 010, Servo 4: SSS = 011, Servo 5: SSS = 100),
- LLLL der niederwertige Teil der Position und
- HHHH der höherwertige Teil der Position.

Um beispielsweise den Positionswert 185 (1011 1001) an Servo 3 auszugeben, sendet man 00101001 10101011 (dezimal 41 171) an den Chip – und schon dreht der Teller in die gewünschte Position. Diese Position wird gehalten, solange kein neuer Befehl eintrifft, und natürlich auch nur, solange die Schaltung Betriebsspannung erhält.

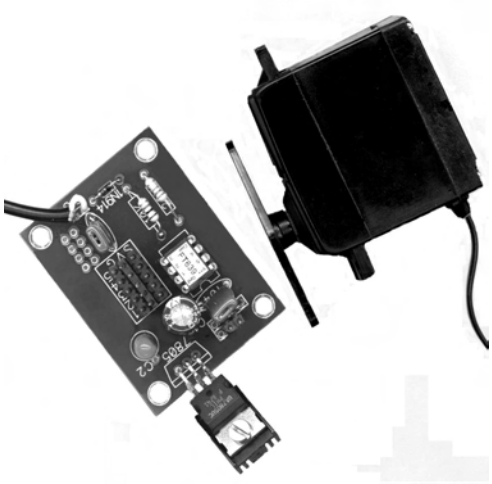

Bild 7.14: Testplatine für den FT639 mit Servo

Im folgenden Listing sind die Funktionen `OpenPort()`, `SendByte()`, `SetServo()` und `CenterAll()` für allgemeine Anwendungen vorgesehen. Mit ihnen lassen sich alle Steueraufgaben mit dem Baustein erledigen. Zum Testen und als Beispiel für eine Anwendung dienen neben dem Hauptprogramm die Funktionen `Handbetrieb()` und `Scanbetrieb()`. Mit der ersten Funktion lassen sich zum Testen der ans Servo angeschlossenen Mechanik beliebige Stellungen anfahren. Mit der Scan-Funktion könnten Sie z. B. eine kleine Webcam um einen definierten Winkel hin- und herschwenken – fertig ist das Überwachungssystem.

```
/*
* Linux_Servo_FT639 is a Linux servo control program for the FT639 chip.
*/

#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termio.h>
#include <termios.h>
#include <signal.h>

/* Debug output */
#define DEBUG 1

/* Microseconds to pause for one byte at 2400 bps */
#define pause 4200

/* Command codes fÃ¼r FT639 */
#define setup 122
#define shortpulse 85
#define longpulse 90
#define header 96
#define active 117

/* store terminal setting */
struct termios old_options, new_options;

/* Devicehandle */
int SerialDevice;


int SendByte(char byte, int SerialDevice)
  {
  /* send one byte and wait */
  int result;

  result = write(SerialDevice, &byte, 1);
  usleep(pause);
  return(result);
  }

int OpenPort(int PortNumber)
  {
  /* Open the serial port */
  switch (PortNumber)
    {
    case 0:
      SerialDevice = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
      break;
    case 1:
      SerialDevice = open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NDELAY);
      break;
    case 2:
      SerialDevice = open("/dev/ttyS2", O_RDWR | O_NOCTTY | O_NDELAY);
      break;
    case 3:
      SerialDevice = open("/dev/ttyS3", O_RDWR | O_NOCTTY | O_NDELAY);
      break;
    default: SerialDevice = -1;
    }
  if (SerialDevice < 0)
    {
    printf("\nCan't open the serial port: %d\n",SerialDevice);
```

```
    return(0);
    }

  /* Get the Original port settings */
  tcgetattr(SerialDevice, &old_options);

  /* Specify port settings: 2400 Baud, 8 Bits, No Parity, 1 Stoppbit */
  new_options.c_cflag = (B2400 | CS8 | CLOCAL);  // raw input = no ICANON

  /* Specify raw data (deselect some settings) */
  new_options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);

  /* Activate the new port settings */
  tcsetattr(SerialDevice, TCSANOW, &new_options); // TCSAFLUSH

  /* Put the chip in setup mode */
  SendByte(setup,SerialDevice);

  /* if needed ... */
  //  SendByte(shortpulse,SerialDevice);
  //  SendByte(longpulse,SerialDevice);

  /* Activate the chip */
  SendByte(active,SerialDevice);
#ifdef DEBUG
  printf("\nSerial Device bereit.\n");
#endif
  sleep(1);
  return(1);
  }


void SetServo(int Servo, int Pos, int SerialDevice)
  {
  unsigned char LowByte, HiByte;
  /* Check borders */
  if (Pos < 0) Pos = 0;
  if (Pos > 255) Pos = 255;
  /* Calculate command data */
  LowByte = ((Servo - 1) << 4) + (Pos & 0x0f);
  HiByte  = ((Servo - 1) << 4) + (Pos >> 4) + (1 << 7);
  SendByte(LowByte,SerialDevice);
  SendByte(HiByte,SerialDevice);
#ifdef DEBUG
  printf("Low byte = %02X, High Byte = %02X\r\n", LowByte, HiByte);
#endif
  }


void CenterAll(int SerialDevice)
  {
  int i;
  for (i=1; i <= 5; i++)
    {
    SetServo(i, 128, SerialDevice);
    /* Give them 50 ms to set Position */
    usleep(50000);
    }
  }
```

```
int Scanbetrieb(int nservo, int low, int high, int SerialDevice)
  {
  /* Servo im Scanningmode betreiben */
  int i;

  printf("\nZum Beenden CTRL-C druecken\n");

  while(1)
    {    /* Endless loop */
    for (i = low; i <= high; i++)
      {
      SetServo(nservo, i, SerialDevice);
      usleep(50000);
      }
    for(i = high; i >= low; i--)
      {
      SetServo(nservo, i, SerialDevice);
      usleep(50000);
      }
    }
  }


void Handbetrieb(int nservo, int SerialDevice)
  {
  /* Servo von Hand steuern */
  int step = 0;

  printf("\nZum Beenden -1 eingeben ...\n");

  while(step >= 0)
    {
    printf("Zielposition 0 to 255: ");
    scanf("%d", &step);
    if (step < 0) return;
    if (step > 255) step = 255;
    SetServo(nservo, step, SerialDevice);
    usleep(50000);
    }
  }

void ClosePort(void)
  {
  /* FT639 in den Setup mode schalten and Device schliessen */
#ifdef DEBUG
  printf("\nSetup-Mode setzen und Schnittstelle schliessen.\n");
#endif
  SendByte(setup,SerialDevice);
  tcsetattr(SerialDevice, TCSANOW, &old_options);
  close(SerialDevice);
  exit(0);
  }


int main (int argc, char *argv[])
  {
```

```
char control_mode = '1';
int low, high, nservo;

/* Seriellen Port oeffnen */
if(! OpenPort(0)) exit(1);

/* Signal-Handler setzen */
signal(SIGINT, ClosePort);
signal(SIGTERM, ClosePort);
/* Modus waehlen */
printf("Überwachungsmodus [1] oder Handsteuerung [2]:\n");
scanf("%c", &control_mode);
switch (control_mode)
  {
  case '1':
    printf("\nStarte Überwachungsmodus\n");
    /* Center all servos */
    CenterAll(SerialDevice);

    printf("Untere Auslenkung eingeben [0 to 100]:\n");
    scanf("%d", &low);
    if (low > 100)
       {
       printf("Wert > 100, verwende 100\n");
       low = 100;
       }
    printf("Obere Auslenkung eingeben [0 to 255]:\n");
    scanf("%d", &high);
    if(high > 255) high = 255;
    if (high < low)
       {
       printf("Wert kleiner %d, verwende 255\n", low);
       high=255;
       }
    printf("Servo auswählen: 1 - 5: ");
    scanf("%d", &nservo);

    Scanbetrieb(nservo, low, high, SerialDevice);
    break;
 case '2':
    printf("Servo auswaehlen: 1 - 5: ");
    scanf("%d", &nservo);
    printf("\nStarte Handsteuerung\n");

    Handbetrieb(nservo, SerialDevice);
    break;
 default: printf("Bitte 1 oder 2 eingeben.\n");
 }
ClosePort();
}
```

7.5.2 Der MIC 800

Dieser Chip arbeitet prinzipiell genau so, wie der FT 693. Auch er wird über die serielle Schnittstelle angesprochen. Der MIC 800 steuert bis zu acht Servos mit einem einfachen Protokoll an. Seine externe Beschaltung besteht lediglich aus einem 10-MHz-Quarz (ideal wären 9,84 MHz – aber mit dieser Frequenz ist kein Quarz

erhältlich) und den üblichen Entkoppelkondensatoren. Seine acht Ausgänge werden wie beim FT 639 direkt mit den anzusteuernden Servos verbunden. Wenn acht Servos nicht ausreichen, kein Problem: Insgesamt lassen sich acht MIC-800-Chips an einer seriellen Schnittstelle parallel anschließen; die Basisadresse der einzelnen ICs wird über drei Adresspins eingestellt. Durch den entsprechenden Code lassen sich beim Ansteuern vom PC aus Baustein und Servo selektieren. Diese Hardwareinstellung wird nur einmal beim Einschalten ausgewertet; man kann die Chipadresse also nicht im Betrieb umschalten. Im Schaltbild (Bild 7.15) sind diese Pins fest mit Masse verdrahtet. Sie könnten alternativ Jumper vorsehen. Die Schaltung zeigt weiterhin die Anordnung der Pufferkondensatoren. Die Betriebsspannung des MIC 800 beträgt 5 Volt.

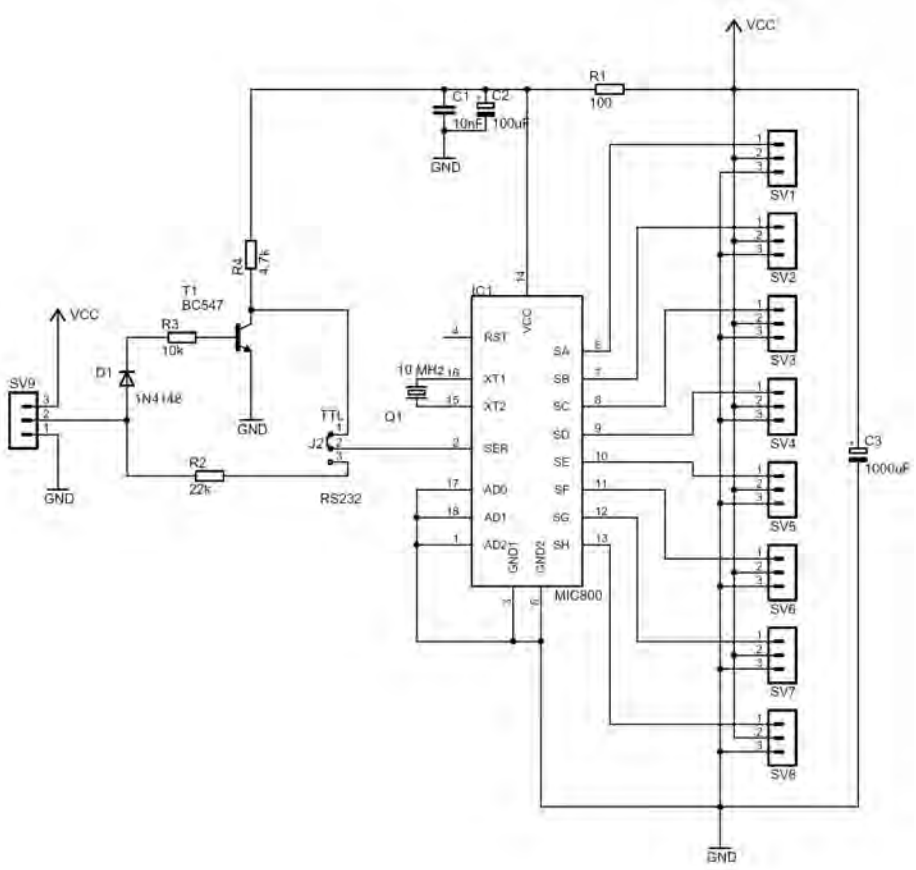

Bild 7.15: Schaltung mit dem MIC 800

Der MIC 800 erwartet an seinem Eingang ein invertiertes Logiksignal: eine logische 1 korrespondiert mit Low-Pegel (0 V), eine logische 0 mit High-Pegel (+ 5 V). Die Beschaltung des seriellen Eingangs kann auf verschiedene Arten erfolgen:

- Die serielle Ansteuerung erfolgt durch einen UART mit TTL-Pegel (und invertiert).

- Vor den Eingang wird ein Widerstand von 22 kOhm geschaltet, wenn es sich um eine serielle Verbindung mit RS232-Pegeln handelt.
- Es wird eine invertierende Transistorstufe vorgeschaltet, wenn die Ansteuerung durch einen UART mit TTL-Pegel ohne Invertierung erfolgt.

Die Schaltung ist so ausgelegt, dass über einen Jumper zwischen RS-232- und TTL-Eingang gewählt werden kann.

Der Dialog mit dem MIC 800 erfolgt mit 2400 Baud, ohne Paritätsbit und mit acht Datenbits. Die Syntax der Befehle, die zum Servo-Interface geschickt werden müssen, ist besonders einfach und wird aus fünf ASCII Zeichen in der Form „ASxxx“, gefolgt von Carriage Return, gebildet (z. B. „TB064“):

- A ist ein Buchstabe zwischen „S“ und „Z“. Er korrespondiert mit der Adresse des gewünschten MIC-800-Chips („S“ = 0, „T“ = 1, „U“ = 2 usw.).
- S ist ein Buchstabe zwischen „A“ und „H“. Er wählt das anzusteuernde Servo aus. Die Servoausgänge SV1 bis SV8 korrespondieren mit den Buchstaben „A“ bis „H“.
- xxx ist eine Zahl zwischen 001 und 128. Sie bestimmt die Position, die der Servo einnehmen soll. 001 entspricht dem Linksanschlag (Endposition entgegen dem Uhrzeigersinn), 064 der Neutralstellung und 128 dem Rechtsanschlag (Endposition im Uhrzeigersinn).

Das ist so einfach, dass man gar kein Programm dazu braucht. Der Befehl `echo "SB064" > /dev/ttyS0` würde beispielsweise das zweite Servo auf Mittelstellung bringen.

Nach dem Einschalten des MIC 800 sind alle Ausgänge inaktiv. Sobald er einen Befehl empfängt, erzeugt der im Befehl angegebene Servoausgang automatisch die Impulse, die erforderlich sind, um die im Befehl angegeben Servoposition einzustellen. Diese Position wird gehalten, solange kein anderslautender Befehl eintrifft, und natürlich auch nur, solange die Schaltung Betriebsspannung erhält.