Studienarbeit

Konzeption und Entwicklung des

Kiss Chip

eines

Converters zwischen RS 232 und Packet Radio

Bernhard Dwersteg 1993
Fachbereich Informatik - Universität Hamburg

Inhaltsverzeichnis

0. Einleitung

1. Protokoll zwischen PC und Chip
1.1 KISS Protokoll
1.2 RS 232 Schnittstelle

2. Protokoll auf der Funkstrecke
2.1 AX.25 Protokoll
2.1.1 Bit-Stuffing
2.1.2 Synchronisation, NRZI-Codierung
2.2 Zugriffssteuerung (CSMA-Verfahren)

3. Funkgerätspezifische Ansteuerung
3.1 FSK-Modem
3.2 Bedienung von PTT

4. Vorgehen bei der Erstellung/ Planung des Chips
4.1 Verhaltenssimulator
4.2 Strukturensimulator
4.3 Test der Simulation

5. Probleme bei der Spezifikation

6. Funktionseinheiten des Chips

7. Überlegungen zur Dimensionierung
7.1 Größe der FIFOs
7.2 Baudratengeneratoren

8. Funktion und Realisierung bestimmter Einheiten als digitale Schaltung
8.1 Realisierung von FIFOs
8.2 CRC-Generator und Prüfer
8.2.1 CRC-Berechnung
8.2.2 CRC-Generator mit Schieberegister
8.2.3 CRC-Prüfer
8.3 Zufallsgenerator
8.4 Digitale PLL
8.4.1 Funktionsweise einer PLL
8.4.2 Implementation der DPLL
8.5 RS 232-Eingangsteil

9. Danksagung

Anhang

A1 Anschlüsse des Chips
A2 Schaltungsaufbau

A3 Aufteilung in Programmteile

A4 Abkürzungsverzeichnis

A5 Literaturverzeichnis

0. Einleitung

Eine weltweit stetig wachsende Zahl von Amateurfunkern benutzt ihr Funkgerät nicht mehr nur zur Übertragung von Sprache, sondern auch zur Übertragung digitaler Daten. Damit eröffnen sich dem Funker neue Möglichkeiten, wie die Versendung von Electronic Mail über Mailboxen, schneller Programm- und Datenaustausch oder Bezug von Shareware. Erlaubt ist allerdings nur der Austausch von Programmen und Daten mit Amateurfunk-Bezug.

Seit Anfang der achtziger Jahre hat sich ein Standard für digitale Übertragung auf der Funkstrecke, das sogenannte Packet Radio, und ein Standard für den Datenaustausch zwischen Computer und Funkgerät durchgesetzt.
Der Packet Radio-Standard ermöglicht über von Amateurfunkern bereitgestellte Vermittlungsstationen (sog. Repeater) einen Datenaustausch über Entfernungen, die die Reichweite eines Funkgerätes überschreiten (s. Abb. E1).

Da ein Heimcomputer (PC) nicht direkt an ein für Sprachübertragung ausgelegtes Funkgerät angeschlossen werden kann, muß zwischen Rechner und Funkgerät ein Modem (Modulator / Demodulator) angeschlossen werden (s. Abb. E2), wie es auch zur Übertragung von Daten über herkömmliche Telefonleitungen verwendet wird.
Alle Teilnehmer, die die Dienste eines zentralen Repeaters in Anspruch nehmen wollen, müssen auf der Frequenz dieses Repeaters senden. Dies bedingt, daß immer nur einer dieser Teilnehmer zur Zeit senden darf. Hier wird eine auf das Funkgerät zugeschnittene Zugriffssteuerung notwendig.
Des weiteren wurde in der Packet Radio-Spezifikation eine Übertragung mit NRZI-codierten HDLC-Daten festgelegt. (Diese Begriffe werden später erklärt.)
Zur Realisierung der Zugriffssteuerung und der HDLC-Codierung wird zwischen PC und Modem ein kleiner Einplatinenrechner, der sogenannte Terminal Node Controller (TNC) geschaltet. Solche TNCs waren ursprünglich so konzipiert, daß sie außerdem die Ansteuerung eines ASCII-Terminals übernehmen können. Diese Funktion wird jedoch bei Verwendung eines Computers als Datenendstation überflüssig. Zum Datenaustausch zwischen Computer und Einplatinenrechner wird das sogenannte KISS-Protokoll (Keep It Simple, Stupid) verwendet.

Der im Rahmen dieser Studienarbeit zu konzipierende KISS-Chip (im Folgenden kurz als Chip bezeichnet) soll diese Einplatinenrechner ersetzen.

Da eine Mikroprozessorschaltung, wie sie zur Zeit auf den Einplatinenrechnern verwendet wird, aus den Hauptkomponenten CPU, ROM, RAM und Peripherieschaltkreisen sowie weiteren Hilfsbausteinen besteht, ist der Implementationsaufwand beträchtlich.
Der KISS-Chip kann diesen Aufwand auf insgesamt zwei integrierte Schaltkreise (KISS-Chip und Schnittstellentreiber) reduzieren.
Bei dieser Implementation bleibt die einfache Ansteuerung über eine Standardschnittstelle (RS 232) erhalten, über die praktisch jeder Rechner verfügt.
Aufgrund der Kompatibilität kann mit bereits bestehenden Softwarepaketen weitergearbeitet werden, soweit diese bereits KISS-Betrieb unterstützen. Bei anderen Softwarepaketen muß nur der Hardware-Ansteuerungstreiber dahingehend verändert werden.
Da der Chip schneller arbeitet als ein TNC, ist ein Übergang zu deutlich höheren Baudraten möglich.

Das Ziel der Studienarbeit ist, das Verhalten des Chips durch ein hardwarenahes TURBO PASCAL-Programm zu simulieren. Anhand dieses Programmes soll dann eine Umsetzung in ein Semi-Custom-IC möglich sein.

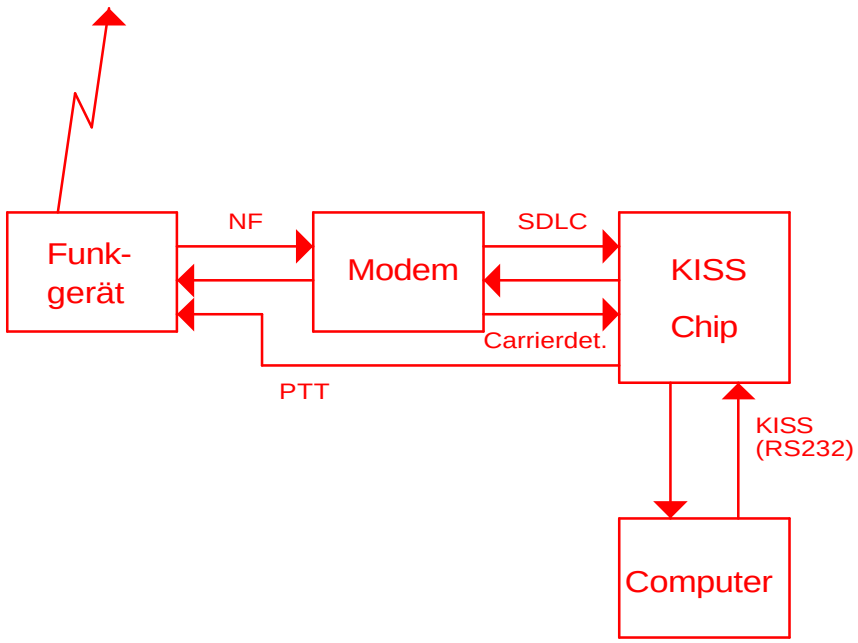

Abb. E2: Anschluß des Funkgeräts an den Computer

1. Protokoll zwischen PC und Chip

1.1 KISS Protokoll

Die Kommunikation zwischen PC und Chip findet im sog. KISS-Protokoll statt. Das KISS-Protokoll wurde in Anlehnung an das Internet SLIP-Protokoll definiert, und unterstützt zusätzliche Steuerzeichen zur Steuerung einer Funkstrecke.
Die Übertragung findet blockweise statt. Jeder Block wird von einem speziellen Zeichen, dem FEND (Frame End), eingeleitet und beendet (s. Tab. 1.1, Abb. 1.1).
Nach dem blockeinleitenden FEND folgt ein Steuerbyte (s. Tab. 3.1), das über den Zweck der folgenden Datenbytes Aufschluß gibt. Alle darauffolgenden Bytes bis zum schließenden FEND werden als Daten interpretiert.
Damit auch ein FEND-Code innerhalb der Daten übertragen werden kann, ist als zusätzlicher Spezialcode das FESC definiert:
Kommt ein FEND in den Rohdaten vor, wird es durch die Kombination FESC TFEND (Frame Escape, Transposed Frame End) ersetzt, ein FESC innerhalb der Daten durch FESC TFESC.
Erhält der Empfänger ein FESC, erwartet er als nächstes Byte nur einen der Codes TFESC oder TFEND. Diese Zwei-Byte-Kombination wird dann zu einem FESC bzw. einem FEND zurückgewandelt. Ein TFESC oder TFEND innerhalb der Daten wird nicht gesondert behandelt.
Dieses Protokoll sorgt dafür, daß innerhalb des Datenblocks nie der Code für FEND übertragen wird, und Blöcke somit eindeutig begrenzt sind. Dies hat gegenüber anderen Methoden (z. Bsp. Längenangabe für den Datenblock statt Endkennzeichen) den Vorteil, daß auch nach einem fehlerhaft empfangenen Block der Anfang des folgenden Blocks auf jeden Fall wieder korrekt erkannt werden kann.

Tab. 1.1: Codes der Spezialzeichen

FEND	frame end	300 (Oktal) = 192
FESC	frame escape	333 (Oktal) = 219
TFEND	transposed frame end	334 (Oktal) = 220
TFESC	transposed frame escape	335 (Oktal) = 221

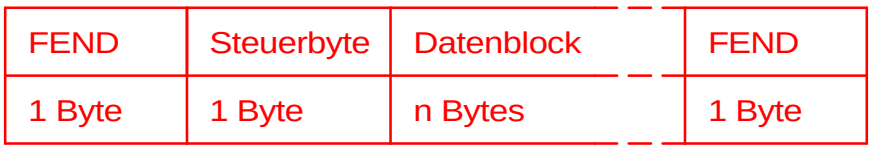

Abb. 1.1: Aufbau eines KISS-Blockes

1.2 RS 232 Schnittstelle

Die Verbindung zwischen PC und Chip geschieht über eine RS 232 C- (EIA 232) Schnittstelle. Der eigentliche Datenaustausch erfolgt dann seriell über je eine Leitung (TXD - RXD) pro Richtung. Die RS 232-Schnittstelle ist eine asynchrone Schnittstelle. Das bedeutet, daß nach der Übertragung weniger Bits jeweils wieder eine neue Synchronisation zwischen Sender und Empfänger stattfinden muß. Dazu wird vor jedem Block ein Startbit (positiv) gesendet.

Anschließend sendet der Sender in gleichmäßigen Zeitabständen seine Datenbits, zuerst das niederwertigste Bit. Da das KISS-Protokoll byteweise arbeitet und kein Paritätsbit vorsieht, werden jeweils 8 Bit übertragen. Um unabhängig von der Polarität des letzten Bits den Wechsel zum nächsten Startbit wieder erkennen zu können, wird nach dem achten Datenbit ein sogenanntes Stopbit eingefügt, das die entgegengesetzte Polarität des Startbits hat (s. Abb. 1.2). Die Gesamtlänge, über die Sender und Empfänger synchron sein müssen, entspricht also 10 Bit-Takten. D.h. schon vor Beginn der Übertragung müssen die Taktfrequenzen von Sender und Empfänger (Baudrate) auf deutlich besser als +/- 10 Prozent übereinstimmend eingestellt sein. Gängige Baudraten liegen zwischen 300 und 115000 Baud.

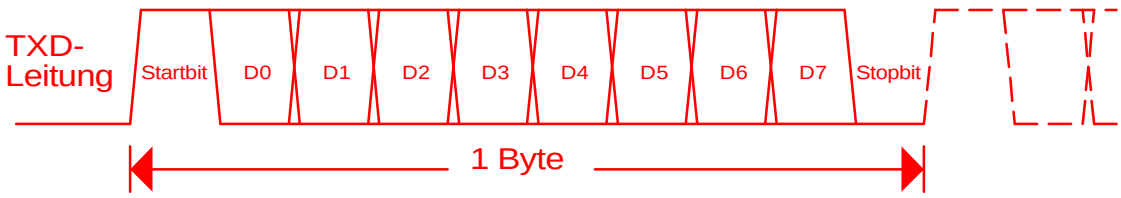

Abb. 1.2: RS 232-Datenübertragung

Der Chip arbeitet als Schnittstellenumsetzer, d.h. er empfängt Daten auf einer Schnittstelle und sendet sie über eine andere aus. Da jedoch beide Schnittstellen unterschiedliche Geschwindigkeiten aufweisen (die RS 232-Verbindung zum Computer muß schneller sein, s. Kapitel 2) und nur ein begrenzter Zwischenspeicher vorhanden sein kann, muß es möglich sein, den Sender zu unterbrechen.
Deshalb ist zusätzlich pro Richtung je eine Leitung vorgesehen (s. Abb. 1.3), um dem Partner Empfangsbereitschaft zu signalisieren. Ein Sender beginnt nur zu senden, wenn der Empfänger das CTS-Signal (Clear to Send) setzt.

Als Logikpegel einer RS 232-Schnittstelle dienen typisch +3 bis +15 V als logisch Eins und -3 bis -15 V als logisch Null.

Der Chip soll mit PC-Standardbaudraten im Bereich um 19200 Baud arbeiten. Da der Chip nicht mit den RS 232-Pegeln arbeitet, sollen extern (invertierende) Treiber (z. Bsp. Typ MAX 232 A) vorgesehen werden.

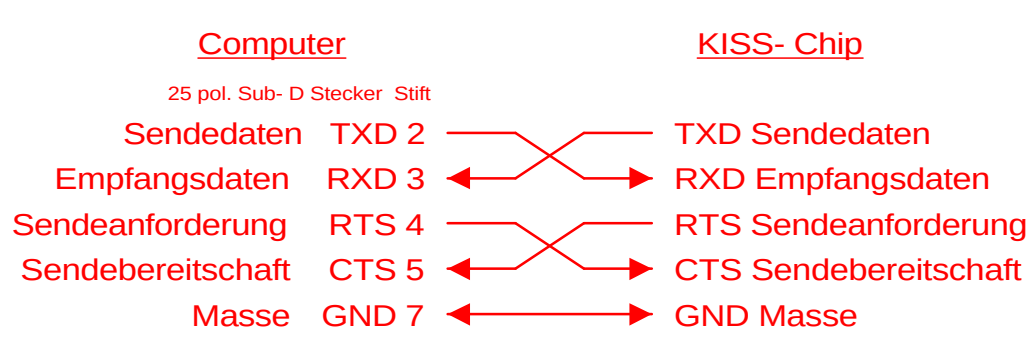

Abb. 1.3: RS 232 Verbindung

2. Protokoll auf der Funkstrecke

2.1 AX.25 Protokoll

Auf der Funkstrecke wird das AX.25 Link-Layer Protokoll verwendet. Dieses Protokoll unterstützt Halb- und Vollduplex-Betrieb: Sind Empfangs- und Sendefrequenz gleich, kann zu einem Zeitpunkt von einer Station entweder nur gesendet oder nur empfangen werden (Halbduplex), d.h. der Datenaustausch zwischen zwei verbundenen Stationen findet jeweils nur in einer Richtung statt. Bei unterschiedlichen Frequenzen kann zugleich gesendet und empfangen werden (Vollduplex). Dieses Protokoll ist eine Erweiterung von ISO-X.25. Es verwendet zu den Protokollen SDLC und HDLC kompatible Datenrahmen. Im folgenden soll deshalb bei Bezug auf die Datenübertragung auf der Funkstrecke die Abkürzung SDLC (Synchronous Data Link Control) verwendet werden.

Die Daten werden wie beim KISS-Protokoll blockweise übertragen. Ein Datenblock wird durch ein Flag eingeleitet und beendet (s. Abb. 2.1).
Nach dem blockeinleitenden Flag folgt der Datenblock. Er besteht aus der Zieladresse, Steuerbytes und den Nutzdaten. Hinter dem Datenblock folgt eine zwei Byte lange Prüfsumme. Beendet wird der Datenblock durch ein weiteres Flag. Dieses kann aber zugleich auch schon den Beginn des folgenden Datenblockes markieren.
Das Blockbegrenzungszeichen ("Flag") ist wie bei KISS wieder ein Zeichen, dessen Vorkommen innerhalb des Datenblockes und des Prüfwortes verhindert wird. Da es sich jedoch bei AX.25 um ein Protokoll für synchrone Datenübertragung handelt und die Synchronisation zwischen Sender und Empfänger durch Übertragung von Flags geschehen soll, muß das Flag eindeutig von anderen Daten unterscheidbar sein, auch wenn der Empfänger nicht weiß, bei welchem Bit es sich um Bit 0 handelt. Es muß deshalb nicht notwendig 8 Bit lang sein, AX.25 sieht jedoch eine byteweise Übertragung vor. Das Flag besteht aus einer Null, gefolgt von sechs Einsen und einer abschließenden Null (01111110).

Flag	Datenblock			Prüfs.	Flag
	Zieladr.	Steuerbytes	Nutzdaten		
01111110	112..560 Bit	16 Bit	0..256 Byte	16 Bit	01111110

Abb. 2.1: AX.25 Datenblock

2.1.1 Bit-Stuffing

Um zu verhindern, daß ein Flag innerhalb der Daten auftritt, wird das sogenannte Bit-Stuffing verwendet.
Dabei fügt der Sender nach der Übertragung von jeweils fünf aufeinanderfolgenden Eins-Bits ein Null-Bit in den seriellen Datenstrom ein. Hat der Empfänger also fünf aufeinanderfolgende Eins-Bits empfangen, und folgt ein Null-Bit, wird dieses entfernt. Folgt ein weiteres Eins-Bit, handelt es sich um ein Flag, falls darauf eine Null folgt. Eine Folge von mehr als sechs Eins-Bits wird als Fehlercode interpretiert und der empfangene Block wird verworfen.

Die Übertragung eines Prüfwortes empfiehlt sich, da eine Funkverbindung sehr stark äußeren Störeinflüssen unterliegt. Aus dem gleichen Grund ist auch die maximale Datenblocklänge begrenzt (s. Abb. 2.1).

2.1.2 Synchronisation, NRZI-Codierung

Auch die Art der Synchronisation ist auf eine synchrone Übertragung zugeschnitten. Da Sender und Empfänger über lange Zeit synchron sein müssen, korrigiert der Empfänger anhand des empfangenen Bitstromes seinen Takt. Dies ist möglich, da ein Wechsel der Datenpolarität bei optimaler Übereinstimmung nur mitten zwischen zwei Bittakten geschehen darf (genauere Beschreibung in Kap. 8.4). Da der Empfänger aber schon vor Empfang des ersten Blockes zum Sender synchron sein muß, wird vom Sender vor Aussendung des ersten Blockes und in Übertragungspausen eine Folge von Flags gesendet. Anhand dieser Flags kann sich der Empfänger synchronisieren.
Um sicherzustellen, daß der Empfänger während der Übertragung nicht zu lange auf einen Pegelwechsel warten muß (wie z. Bsp. bei einer langen Folge von Null-Bytes), wird eine sogenannte NRZI-Codierung verwendet: Anstatt eines Null-Bits wird ein Pegelwechsel gesendet, ein Eins-Bit wird durch gleichbleibenden Pegel dargestellt. Da maximal sechs Eins-Bits in Folge gesendet werden dürfen, ist somit sichergestellt, daß nach spätestens sieben Bittakten wieder ein Pegelwechsel erfolgt.

2.2 Zugriffssteuerung (CSMA-Verfahren)

Um den Zugriff mehrerer Teilnehmer auf die Übertragungsstrecke (eine Funkfrequenz) zu erlauben, wird ein Carrier-Sense-Verfahren (CSMA: Carrier Sense Multiple Access) angewendet.
Eine Station, die senden möchte, muß sich vorher vergewissern, daß keine andere Station sendet. Dies kann anhand des vom Funkgerät empfangenen Tonsignales festgestellt werden: Wird der Tonträger empfangen, sendet gerade eine andere Station, wird jedoch nur Rauschen empfangen, ist die Übertragungsstrecke frei. Das Senden darf jedoch nicht sofort beginnen, wenn festgestellt wird, daß die Strecke frei ist. Ansonsten würden alle Stationen, die während der Übertragung der zuvor aktiven Station einen Sendewunsch hatten, zeitgleich mit ihrer Übertragung beginnen. Dieses Problem wird dadurch noch verschärft, daß es einige Millisekunden dauert, bis sich ein Empfänger auf einen neuen Sender so genau eingestellt hat, daß er von ihm Daten empfangen kann.
Bevor eine Station anfängt zu senden, wartet sie, bis die Übertragungsstrecke frei ist. Sobald dies eingetreten ist, wird eine Zufallszahl (zwischen 0 und 255) erzeugt. Ist diese Zahl kleiner oder gleich dem vorwählbaren Parameter P (**Persistance** P, zwischen 0 und 255), sendet die Station ihre Daten. Ist die Zufallszahl aber größer als P, muß die Station vor dem nächsten Versuch eine gewisse Zeit (die **Slottime**) lang warten. Wird nach der Wartezeit der Tonträger einer anderen Station detektiert, wartet die Station wieder, bis der Kanal frei wird, und beginnt die Prozedur von vorne.
Die Wahl der Persistance hängt von der Anzahl der Teilnehmer und der Wichtigkeit der Station ab. Die Wahl der Slottime hängt zusätzlich noch von der Übertragungsgeschwindigkeit und den typischen Blocklängen ab.

3. Funkgerätspezifische Ansteuerung

Bei den verwendeten Funkgeräten handelt es sich im allgemeinen um für Sprachübertragung ausgelegte Geräte. Der Anschluß des Chips an das Funkgerät geschieht über einen Modembaustein, welcher das NRZI-Ausgangssignal des Chips als Sinuswellen unterschiedlicher Frequenzen codiert und Sinuswellen unterschiedlicher Frequenzen wieder in einen digitalen Datenstrom umsetzt. Dazu wird das Modem mit dem Mikrofoneingang und einem (internen ungefilterten) NF-Ausgang verbunden.
Um ein Funkgerät im Halbduplexbetrieb betreiben zu können, gibt es meist am Mikrofon einen Schalter, der zwischen Senden und Empfangen umschaltet. Dies ist der PTT-Taster (Push To Talk). Über die PTT-Leitung kann der Chip den Sender ein- und ausschalten.

3.1 FSK-Modem

Zur Übertragung wird die sogenannte FSK-Technik (Frequency Shift Keying) genutzt. Idee hierbei ist, ein Null-Bit durch eine andere Frequenz als ein Eins-Bit zu codieren. Beide Frequenzen werden durch Sinuswellen dargestellt, da diese im Gegensatz zu allen anderen Wellenformen über keine weiteren Frequenzkomponenten als ihre Grundfrequenz verfügen. Da jedoch beim Umtasten zwischen zwei Sinuswellen zumindest keine stetig differenzierbare Funktion mehr vorliegt, entstehen dabei auch andere, unerwünschte Frequenzkomponenten. Die Erzeugung und Erkennung der zwei Frequenzen und die Ausfilterung unerwünschter Frequenzen sind Aufgaben des Modems. Des weiteren muß insbesondere zur Übertragung höherer Baudraten (> 4800 Baud) der auf Sprachsignale ausgelegte Frequenzgang des Funkgeräts kompensiert werden. Diese Kompensation muß an das Modell des Funkgeräts angepaßt sein.
Aufgrund der Komplexität dieser Aufgaben ist eine Integration des Modems auf dem Chip nicht vorgesehen.

3.2 Bedienung von PTT

Für die Bedienung der PTT-Leitung sind zwei Parameter von Bedeutung: die Zeit, die vom Anschalten des Senders bis zum Aussenden des ersten Blockes gewartet wird (**TXDelay**) und die Zeit, die der Sender nach Aussendung des letzten Bytes mindestens noch angeschaltet bleibt (**TXTail**, s. Tab. 3.1). Die TXDelay-Zeit ist nötig, um abzuwarten bis der Sender stabil arbeitet und es dem Empfänger des Gesprächspartners zu ermöglichen, sich genau auf die Sendefrequenz abzustimmen. Die TXTail-Zeit soll den durch das CSMA-Verfahren entstehenden Zeitverlust verringern, indem sie die Häufigkeit des Übergabeverfahrens reduziert. So hat die aktive Station nach der Aussendung eines Blockes die Möglichkeit, sofort einen weiteren Block zu senden, falls der Sendewunsch noch innerhalb der TXTail-Zeit entsteht.
Dieses Verfahren wird beispielsweise auch bei Diskettenlaufwerken verwendet: Dort läuft der Drehmotor nach einem Zugriff noch einige Sekunden nach, damit bei einem eventuellen Zugriff, der noch innerhalb dieser Zeit folgt, nicht erneut die Hochlaufzeit abgewartet werden muß.

Tab. 3.1: Codes für Datenblock-Typen (vgl. Tab. 1.1)

Steuerbyte	Funktion	Kommentar
0	Benutzerdaten	Der Rest des Blockes besteht aus Nutzdaten
1	TXDelay	Das folgende Byte gibt die Auftastzeit als Vielfaches von 10 ms an; bei TXDelay=0 erfolgt Auftaststeuerung durch Computer
2	Persistance	Das folgende Byte ist der Persistance-Parameter (Default: 63)
3	Slottime	Das folgende Byte gibt die Zeitscheibendauer als Vielfaches von 10 ms an
4	TXTail	Das folgende Byte gibt die Nachlaufzeit für den Sender als Vielfaches von 10 ms an
16 (*)	PC-Baudrate	Das folgende Byte enthält den Baudratendivisor für die RS 232-Schnittstelle
17 (*)	Funk-Baudrate	Das folgende Byte enthält den Baudratendivisor für die SDLC-Schnittstelle

(*): Speziell für den KISS-Chip definierte Codes

4. Vorgehen bei der Erstellung/ Planung des Chips

Das Vorgehen zum Entwurf eines Chips mit einem Semi-Custom-Entwurfssystem wie dem an der Uni Hamburg gebräuchlichen SOLO 1400 gliedert sich üblicherweise grob in die Teile

- Anforderungsermittlung und Teil-Spezifikation
- Verhaltenssimulation als vollständige Spezifikation
- Transfer auf Register- / Gatterebene
- Übersetzung in Chip durch Silicon-Compiler.

Dabei werden immer wieder Iterationen zwischen den einzelnen Stufen notwendig, da bei der Implementation der späteren Stufen häufig noch Probleme oder Verbesserungsmöglichkeiten in den vorhergehenden Schritten entdeckt werden. Insbesondere muß nach jedem dieser Schritte durch Simulation ein Vergleich mit den Ergebnissen des vorigen Schrittes erfolgen.

Da die Anforderungen an den KISS-Chip zu Beginn der Studienarbeit nur grob natürlich-sprachlich bezüglich seiner Funktion als Ersatz bestehender Terminal Node Controller (TNC) umrissen waren, mußten vor und während der Erstellung des Verhaltenssimulators noch viele Einzelheiten im Sinne einer Spezifikation geklärt werden. Dazu wurden die Anforderungen in Gesprächen mit dem Betreuer ermittelt.
Als Grundlagen für die Protokolle auf PC- und Modemschnittstelle dienten von Amateur-funkern erstellte Beschreibungen des KISS- und des AX.25-Protokolls.
Für die genaue Spezifikation der PC-Schnittstelle wurden Daten der häufig eingesetzten RS 232-Schnittstellenbausteine NS 16450 zugrunde gelegt und die übliche Verwendung der Steuerleitungen berücksichtigt.
Als Spezifikation der SDLC-Schnittstelle diente außerdem ein Datenblatt des in den meisten TNCs eingesetzten Universal-Schnittstellenbausteins Z8530, da keine genauen Unterlagen zu TNCs verfügbar waren. Der Aufbau des verwendeten CCITT-16 Prüf-Polynoms wurde aus Furrer [4] entnommen.
Die Spezifikation zur Funkgerät-Ansteuerung lieferte ein Text von Phil Karn [3], dem Urhebers des KISS Protokolls.
Die Anforderungen an die Modem-Schnittstelle wurden den Daten zu einem in TNCs verwendeten Modem-IC, dem AM 7911, entnommen. Für die Ansteuerung von 9600 Baud Modems wurde das Buch von Kneip [5] zugrundegelegt.

4.1 Verhaltenssimulator

Aufgrund der ermittelten Anforderungen und der oben genannten Unterlagen ist ein Verhaltenssimulator in Turbo Pascal erstellt worden, der die Funktion des Chips simuliert. Bei der Programmierung des Verhaltenssimulators wurde bereits auf wesentliche Aspekte einer späteren Hardware-Implementation Rücksicht genommen. Zusätzlich wurde auf gute Überprüfbarkeit und geringe Laufzeit Wert gelegt, um einen Test einzelner Programmteile unter (minimalen) Echtzeitanforderungen zu ermöglichen. So sollte es ermöglicht werden, direkt über die Ein-/ Ausgabeleitungen eines PCs Teile eines TNCs zu simulieren.
Aus diesem Grund wurden besonders zeitintensive Teile, wie die Erzeugung der Baudraten aus der Quarzfrequenz (hoher Teilerfaktor) und die Taktrückgewinnung aus dem SDLC-Datenstrom in einem separaten Programm untergebracht.

4.2 Struktursimulator

Um einen schrittweisen Übergang vom Verhaltenssimulator zur Beschreibung auf der Register-Transfer-Ebene zu ermöglichen, kann ein Struktursimulator als Zwischenschritt sinnvoll sein. Er beinhaltet:

- Auflösung komplexer Operationen wie Multiplikation und Division in im Entwurfssystem vorhanden Grundstrukturen
- Begrenzung der Register und Rechengenauigkeit auf die wirklich benötigte Breite
- Zerteilung des Funktionsablaufes in Takte
- Beachtung der parallelen Abläufe innerhalb des Chips im Gegensatz zur sequentiellen Abarbeitung eines Programmes
- Umwandlung lokaler (Prozedur-) Variablen in globale Variablen
- Minimierung der Anzahl der benötigten Hilfsvariablen

Bei der Implementation des Verhaltenssimulators wurden bereits die wesentlichen Aspekte eines Struktursimulators berücksichtigt. Da bei der Erstellung eines zusätzlichen Programmes als Zwischenschritt auch Fehler auftreten können und sich die Wartung mehrerer Simulatorprogramme bis zur endgültigen Fertigstellung des Chips aufwendiger gestaltet, wird auf die Erstellung eines Struktursimulators verzichtet. Dies erscheint auch sinnvoll, weil der KISS-Chip keine komplexen Operationen enthält, die in Pascal bereits vorhanden sind, jedoch bei der Eingabe in ein Chip-Entwurfssystem ungleich höheren Aufwand verursachen würden.

4.3 Test der Simulation

Nach Fertigstellung des Verhaltenssimulators ist ein Testprogramm erstellt worden, das die meisten Funktionen des Chips am Verhaltenssimulator austestet. Besonders kritische Funktionen sind auch isoliert in jeweils einem weiteren Testprogramm untersucht worden. Über diese Testprogramme kann die Übereinstimmung der Funktion mit der Spezifikation überprüft werden.
Des weiteren ist es sinnvoll, die getesteten Simulationsstimuli auch zur Überprüfung des Chipdesigns nach Eingabe in das Entwurfssystem zu verwenden. Ein schrittweiser Vergleich ermöglicht die Erkennung und Lokalisierung von Fehlern.

5. Probleme bei der Spezifikation

Als ein für die Implementation wesentlicher Aspekt ergab sich, daß zwar momentan Packet Radio in Deutschland zumeist nur mit Baudraten von 1200 Baud betrieben wird, jedoch aufgrund steigender Teilnehmerzahl eine Migration zu höheren Baudraten wünschenswert ist. Bereits seit einigen Jahren sind insbesondere in den USA Übertragungsgeschwindigkeiten von 9600 Baud zu einem neuen Standard geworden. Dies ist jedoch nur durch eine aufwendige, aus Standardbausteinen aufgebaute Modemschaltung möglich, da die bisher verwendeten Telefonmodem-ICs diese Baudrate mit herkömmlichen Funkgeräten nicht erreichen konnten. Diese Modemschaltung ist von dem englischen Funkamateur (mit Rufzeichen G3RUH) entworfen worden. Es existiert mittlerweile eine neuere, leichter aufzubauende Version (s. Buch von Kneip [5]). Der Einsatz dieser Modemschaltung erfordert jedoch einige Eingriffe im Funkgerät und in Software und Hardware des TNCs.
Deshalb ist diese Schaltung speziell auf den Einsatz mit gängigen TNCs zugeschnitten und führt zu einem Großteil Funktionen aus, die zwar innerhalb des TNCs zu Verfügung stehen,

jedoch nicht extern zugänglich sind. Dazu wird der TNC u.a. in einem Modus betrieben, dessen Existenz nur zufällig aufgrund der Verwendung eines hochintegrierten Universal-Schnittstellenbausteins (Z8530) gegeben ist. Da jedoch die Implementation dieses Modus im Chip einen nicht unerheblichen Aufwand bereiten würde, erscheint es sinnvoll, die Signale, die bei der Signalaufbereitung innerhalb des Chips anfallen, und die von dem Modem benötigt und zur Zeit selbst generiert werden, über zusätzliche Pins zugängig zu machen. Die Modemschaltung kann dann entsprechend vereinfachet werden.
Weil jedoch die Dokumentation der Modemschaltung (in [5]) nicht in ausreichender Weise auf diese Interna eingeht, besteht hier noch Klärungsbedarf.

6. Funktionseinheiten des Chips

Zur Implementation des Verhaltenssimulators ist es sinnvoll, das Gesamtproblem zuerst in einige in sich stark zusammenhängende Teile zu zerlegen. Aus der gewünschten Funktion des Chips ergibt sich eine Aufteilung in folgende Funktionsblöcke (s. Abb. 6.1):

- RS 232 Eingangsteil
- KISS Decoder
- SDLC Wandler
- NRZI Wandler
- NRZI Decoder
- SDLC Decoder
- KISS Wandler
- RS 232 Ausgangsteil
- Funkgerätsteuerung (PTT-Bedienung)
- Taktgeneratoren / Frequenzteiler

Da die beiden seriellen Schnittstellen mit unterschiedlichen Geschwindigkeiten arbeiten und die SDLC-Schnittstelle aufgrund fehlender Verzögerungsmöglichkeiten kontinuierlich mit Daten versorgt werden muß bzw. Daten liefert, ist innerhalb des Chips ein ausreichender Speicherplatz in der Verbindung zwischen beiden Schnittstellen vorzusehen. Für solche Anwendungen benutzt man sogenannte FIFO-Speicher (First In First Out), die Daten in der gleichen Reihenfolge ausgeben, wie sie empfangen wurden. Deshalb sind als Kopplung zwischen RS 232- und SDLC-Teil FIFOs vorzusehen.

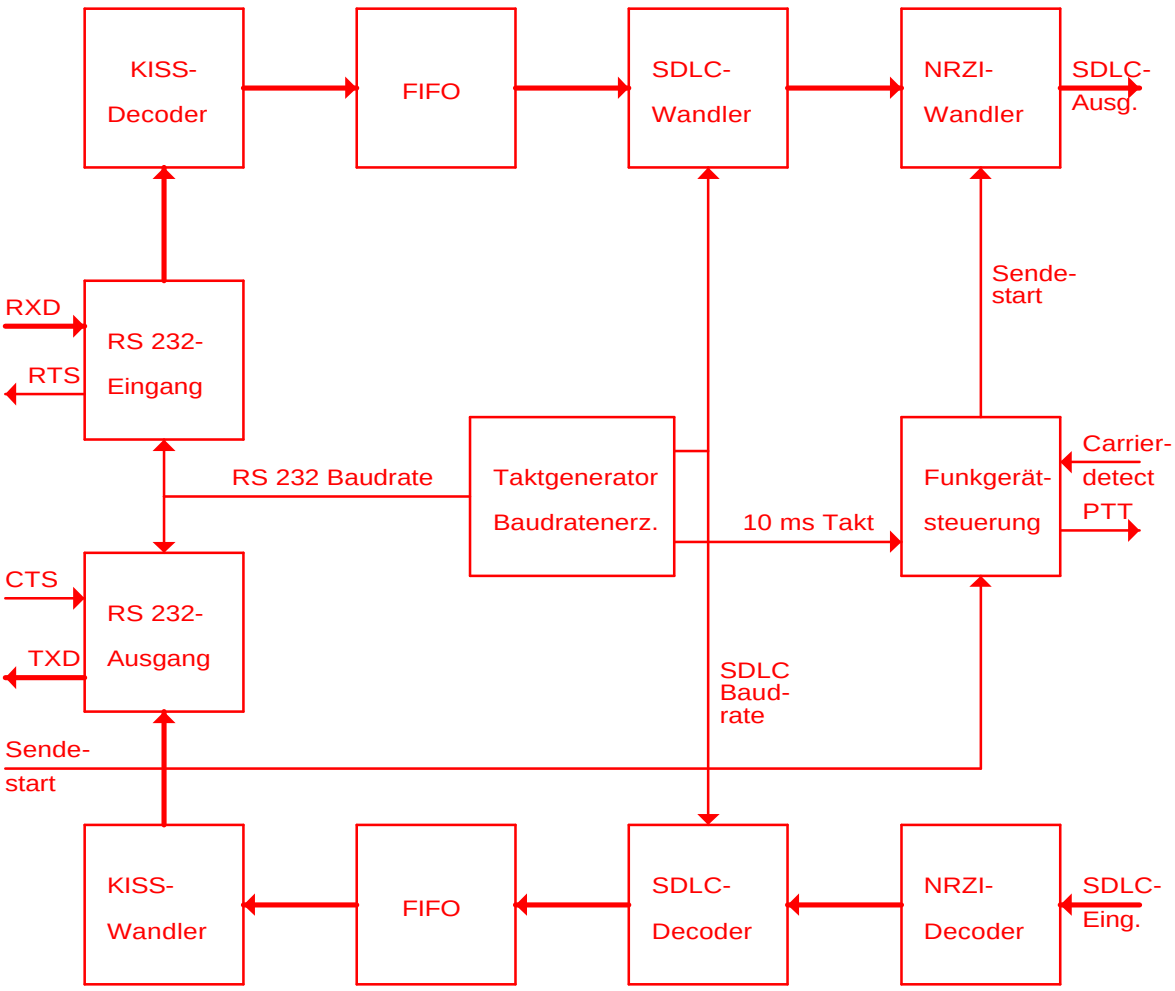

Abb. 6.1: Funktionsblöcke des Chips

7. Überlegungen zur Dimensionierung:

In diesem Abschnitt soll untersucht werden, wie einige der in Abschnitt 6 aufgeführten Teile, deren Funktion nicht vollständig spezifiziert ist, zu dimensionieren sind. Da sich bei dieser Überlegung Wechselwirkungen mit dem gewünschten Verhalten dieser Einheiten ergeben, wird die Spezifikation dieser und anderer Teile dadurch diesbezüglich erweitert.

7.1 Größe der FIFOs

Zur Sicherung der übertragenen SDLC-Blöcke ist je Block ein 16 Bit langes Prüfwort vorgesehen. Bei Empfang eines Blockes soll der Chip anhand des Prüfwortes prüfen, ob der Block korrekt empfangen wurde. Der Block wird nur bei korrektem Empfang über die RS 232-Schnittstelle an den Computer übertragen. Das bedeutet, daß der Chip in der FIFO-Warteschlange, die den **SDLC Empfangsteil** mit dem RS 232-Sendeteil verbindet, mindestens einen maximal großen Block speichern können muß. Die maximale Länge beträgt 328 Bytes (256 Datenbytes, 70 Adressbytes, 2 Steuerbytes). Da jedoch nach korrektem Empfang eines Blockes sofort der nächste Block empfangen werden kann und dem Computer eine gewisse

Reaktionszeit eingeräumt werden muß, bis er bereit ist, Daten entgegenzunehmen, muß die Speicherkapazität höher sein.
Folgen auf einen maximal großen Block viele kleine Blöcke, ist es sogar möglich, daß mehrere kleine Blöcke empfangen werden, während der große Block noch an den PC übertragen wird.
Da die Blöcke im KISS-Protokoll an den Computer geschickt werden müssen, gibt es zwei mögliche Stellen die Umwandlung vorzunehmen: entweder beim Einschreiben in die FIFO oder beim Auslesen aus der FIFO. Werden die empfangenen Blöcke gleich als KISS-Daten gespeichert, ist jedoch zu beachten, daß sich die Datenmenge durch die ESC-Codes vergrößert und daher die Speicherkapazität größer sein muß. Im schlechtesten Fall verdoppelt sich die Datenmenge.
Soll die Wandlung beim Auslesen stattfinden, müssen Blockanfänge markiert oder separat gespeichert werden. Letztere Möglichkeit würde einen nicht unerheblichen Verwaltungsaufwand erfordern. Also bleibt nur die Möglichkeit der Markierung. Da alle 256 mit einem Byte darstellbaren Codes im Datenstrom vorkommen können, muß also ein weiteres Bit pro Speicherstelle vorgesehen werden. Dieses kann dann zum Beispiel beim ersten Byte eines Blocks gesetzt werden.
Mit der gleichen Methode ließen sich aber auch ESC-Codes in der FIFO platzsparender realisieren. Ein gesetztes neuntes Bit würde dann ein mit ESC-Code zu sendendes Byte markieren.
Bei dieser Organisation bietet sich eine Größe von 512 * 9 Bit (512= 2^9) an, da 512 Speicherstellen über 9 Bit adressiert werden können, so daß jedem Code eine Speicherstelle zugeordnet ist. Diese Größe bietet auch im schlechtesten Fall genügend Reserve.

Da das Prüfwort erst am Blockende übertragen wird, muß der **SDLC-Sendeteil** keinen vollständigen Block zwischenspeichern. Weil im KISS-Protokoll keine Prüfmöglichkeit vorgesehen ist, muß auch im KISS-Empfangsteil kein vollständiger Block zwischengespeichert werden können. Da jedoch dem Computer auch beim Senden eine gewisse Reaktionszeit eingeräumt werden sollte, sollte die FIFO einen gewissen Zeitraum während des Sendens eines SDLC-Blockes überbrücken können, währenddessen keine neuen Daten vom PC kommen. Wird nämlich während der Übertragung eines Blockes die FIFO (kurzzeitig) leer, kann der synchrone Datenstrom nicht mehr aufrechterhalten werden, und der Chip muß einen Break-Code (mehr als 6 aufeinanderfolgende Eins-Bits) senden, um dem Empfänger den Fehler zu melden. Alle folgenden Daten, die der Computer bis zum Blockende sendet, muß der Chip dann ignorieren.
Um zu verhindern, daß es schon bei Blockanfang zu einem Unterlauf der FIFO kommen kann, soll der Chip einen Sendewunsch überhaupt erst anerkennen, wenn mindestens 14 Bytes vom PC gekommen sind. Dies ist weniger als die minimale Blockgröße.
Die wirklich benötigte Größe hängt von der PC-Software und dem innerhalb des Computers verwendeten Schnittstellenbaustein ab: Neuere UART-Bausteine (Universal Asynchronous Receiver Transmitter) wie sie in PCs eingesetzt werden, enthalten ihrerseits eine 2 bis 16 Byte große FIFO. Dadurch kann die Reaktion des PCs auf eine Rücknahme des CTS-Signales (Clear To Send) durch den Chip um bis zu 16 Bytes verzögert geschehen. Das hat zur Konsequenz, daß der Chip bereits 16 Bytes bevor seine interne FIFO überläuft, das CTS-Signal löschen muß.
Mit diesen Überlegungen muß die Sende-FIFO mindestens 14+16, also 32 (=2^5) Bytes groß sein. Ein Vorteil eines größeren Speichers ist, daß der Computer weniger häufig auf das RTS-Signal warten muß, und, während die Übertragung eines Blockes noch in Gang ist, schon den nächsten Block an den Chip senden kann. Deshalb wird die Sende-FIFO 64 Byte groß gewählt.

7.2 Baudratengeneratoren

Die Aufgabe eines Baudratengenerators ist es, einen Takt zur Verfügung zu stellen, mit dem Bits gesendet oder empfangen werden sollen. Ein Baudratengenerator besteht im allgemeinen aus einem einstellbaren Frequenzteiler, der eine hohe Frequenz durch einen variablen Divisor teilt.
Bisher sind als Übertragungsbaudrate auf der Funkstrecke nur 1200 bzw. 9600 Baud etabliert. Dies sind Standardbaudraten, die von den meisten seriellen Schnittstellen mit einstellbarer Geschwindigkeit unterstützt werden. Für die Übertragung auf der RS 232-Schnittstelle ist die Verwendung einer möglichst hohen Baudrate wünschenswert, damit sich die Übertragung hier nicht verzögernd auf den gesamten Netzdurchsatz auswirkt. Um einen Überlauf bzw. Unterlauf der FIFOs zu verhindern, muß die Übertragungsgeschwindigkeit höher sein, als die auf der Funkstrecke verwendete Geschwindigkeit. Da jedoch mit dem KISS-Protokoll bestimmte Codes nur über ESC-Codes übertragen werden können, ist sogar mindestens die doppelte Baudrate wünschenswert.
Um die Auswahl zwischen möglichst vielen Baudraten zu haben und gleichzeitig die gängigen Standardbaudraten abzudecken, bietet es sich an, die Frequenzteilung auf genau die gleiche Weise vorzunehmen, wie Standard-UARTs in PCs es tun. Diese teilen eine Frequenz von 1,8432 MHz durch einen einstellbaren 16-Bit Teiler und die resultierende Frequenz über einen festen Teiler durch 16. Als maximale Baudrate ergibt sich 115200 *1/s. Da eine Verwendung von Frequenzen unterhalb 500 Baud nicht sinnvoll erscheint und das KISS-Protokoll auch für andere Parameter nur ein Byte große Steuerworte verwendet, ist für den Chip die Verwendung eines einstellbaren 7-Bit Teilers für die Erzeugung der RS 232-Baudrate und die eines 8-Bit Teilers für die Erzeugung der SDLC-Baudrate ausreichend. Da intern (für die später beschriebene DPLL) ein gegenüber der SDLC-Baudrate 32-fach höherer Takt verwendet werden muß, soll als externe Taktfrequenz 3,68 MHz verwendet werden. Dies ist ebenfalls eine Frequenz, für die es Standardquarze gibt.

8. Funktion und Realisierung bestimmter Einheiten als digitale Schaltung

In diesem Abschnitt soll die Realisierung bestimmter Teile des Chips beschrieben werden. Dabei sind insbesondere solche Teilschaltungen von Interesse, deren Realisierung nicht nur in der Formulierung einer komplexen Logik besteht, sondern bei denen das Vorgehen selbst entscheidend die Komlexität und Realisierbarkeit der Schaltung bestimmt.

8.1 Realisierung von FIFOs

Ein FIFO-Speicher ist die technische Realisierung einer endlich langen Warteschlange. Seine Aufgabe ist es, Daten an einem Eingang in Empfang zu nehmen, und bei Bedarf in unveränderter Reihenfolge am Ausgang abzuliefern. Eine FIFO realisiert also eine variable Verzögerung. Die maximale Anzahl von Daten innerhalb der FIFO-Warteschlange ist durch ihre Länge gegeben. Der Informationsgehalt jedes Datenwortes bestimmt die Breite der FIFO. Bei 8-Bit Datenworten ist also jeder Warteplatz 8 Bit breit.
Im einfachsten Fall enthält die FIFO nur einen Warteplatz der Breite b: Eine solche FIFO ist technisch als b-Bit Register realisierbar. Die für diese einfache FIFO benötigte Steuerlogik enthält bereits die grundlegenden Elemente der für größere FIFOs benötigten Steuerlogik:

- Aufwärts-/ Abwärts-Zähler: enthält die Anzahl der belegten Warteplätze (hier 0 oder 1)
- Komparator für Endwert: meldet dem Sender, ob noch Platz in der FIFO ist
- FIFO-Leer-Detektor: meldet dem Empfänger, ob Daten für ihn da sind

Zur Herstellung größerer FIFOs reicht es also aus, solche grundlegenden FIFOs zu kaskadieren. Nachteilig ist jedoch, daß Daten dann immer alle FIFO-Elemente durchlaufen müssen und der Anteil der für die Steuerlogik benötigten Elemente proportional mit der Größe wächst. Auch ein Aufbau aus kaskadierten Registern und gemeinsamem Steuerwerk wäre denkbar.
Eine Implementation solcher Logik mit einem Semi-Custom-Entwurfssystem führt jedoch erfahrungsgemäß zu einem äußerst chipflächen-ineffizienten Design. Deshalb ist dieses Vorgehen nur für **kleine FIFOs** sinvoll.

Eine gute Chipflächenausnutzung ergibt sich bei Verwendung leistungsfähiger Standardzellen. Zur Speicherung **großer Datenmengen** bietet sich daher ein als Standardzelle vorhandenes RAM an. Dieses kann analog einem Array in PASCAL verwendet werden. Die Anzahl der Adreß- und Datenbits kann (relativ) frei vorgewählt werden.
Mit einem RAM als Speicher kann eine **FIFO als Ringpuffer** (s. Abb. 8.1) realisiert werden. Die Anzahl der Datenbits wird gleich der benötigten Breite gewählt und die Anzahl der Adreßbits so, daß die mindestens benötigte Anzahl N von Speicherplätzen in der FIFO adressiert werden kann. Zur Steuerung werden folgende grundlegenden Elemente benötigt:

- Schreibzähler (S) als modulo Speichergröße- Aufwärtszähler
- Lesezähler (L) als modulo Speichergröße- Aufwärtszähler
- Subtrahierer zur Berechnung der Füllung F = (S+N-L) mod N
- Komparator für FIFO leer
- ggf. Komparator für minimale Füllung (vgl. Kap. 7.1)
- ggf. Komparator für maximale Füllung

Die zwei Haupt-FIFOs sind nach diesem Prinzip aufgebaut.

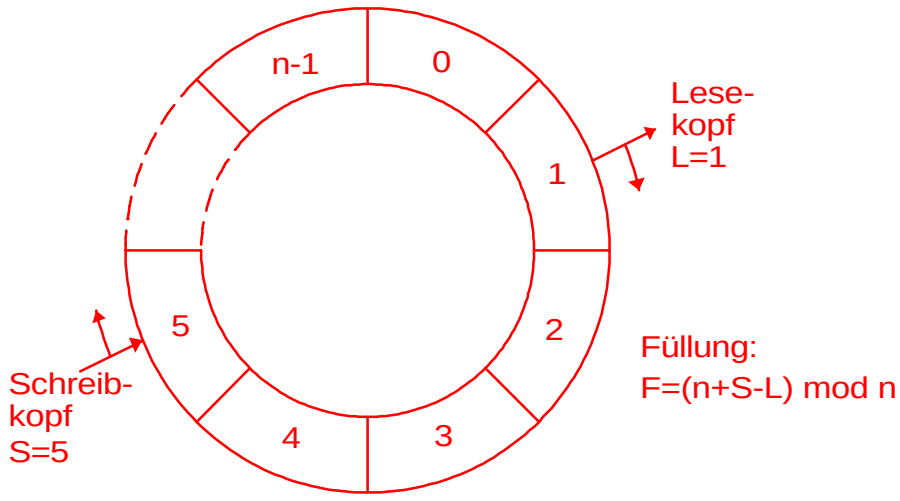

Abb. 8.1: Prinzip der FIFO

8.2 CRC-Generator und Prüfer

8.2.1 CRC-Berechnung

In synchronen Datenübertragungssystemen werden die seriellen Daten durch einen fehlererkennenden Code gesichert. Zu diesem Zweck wird jedem Informationsblock ein Prüfblock angehängt. Der Prüfblock entsteht aus der fortlaufenden binären Division der Datenbits durch ein Generator- oder Prüfpolynom g(z). Im Sender ergibt diese Division den Prüfblock, der anschließend an die zu übertragenden Daten angehängt wird.
Im Empfänger wird dieselbe Division mit den empfangenen Daten durchgeführt und das Ergebnis wird mit dem empfangenen Prüfblock verglichen. Stimmen der empfangene Prüfblock und der neu errechnete Prüfblock überein, so wird eine fehlerfreie Übertragung angenommen. Unterscheiden sich die beiden Prüfworte, so sind Übertragungsfehler aufgetreten und der fehlerhafte Block wird als Ganzes noch einmal übermittelt. (Dafür sorgt der als Programm realisierte Teil des Protokolls; der Chip muß einen als fehlerhaft erkannten Block nur ignorieren.)
Datenblock und Prüfwort zusammen werden auch als Codewort bezeichnet. Der **Hamming-Abstand** bezeichnet die Anzahl unterschiedlicher Bits zwischen zwei Codeworten. Ein Prüfpolynom wird geschickt so gebildet, daß ein möglichst großer Hamming-Abstand zwischen gültigen Codeworten besteht. Das bedeutet, daß nach Verfälschung eines oder mehrerer Bits eines gültigen Codewortes noch möglichst viele weitere Bits geändert werden müssen, um wieder ein gültiges Codewort zu erreichen. Zusätzlich ist es möglich, das Prüfpolynom so auszuwählen, daß es insbesondere für bestimmte für das gewählte Übertragungsverfahren charakteristische Fehlertypen unempfindlich ist.
Die für synchrone Datenübertragung verwendeten Prüfpolynome sind normiert worden. In AX.25 wird das CRC-Polynom (Cyclic Redundancy Check) der CCITT verwendet. Es sieht einen 16-Bit Prüfblock vor.

CRC-CCITT Prüfpolynom: 1+z^5+z^12+z^16

Dieses Polynom nennt sich zyklisch, da nach einer zyklischen binären Verschiebung eines damit gebildeten Codewortes wieder ein gültiges Codewort entsteht. Damit aus dieser Eigenschaft keine Nachteile entstehen, wenn zum Beispiel ein Bit verloren geht, wird vor der Übertragung des Prüfwortes noch eine Addition einer Konstanten vorgenommen.

Das Vorgehen zur **Bildung des Prüfwortes** bei AX.25 umfasst folgende Schritte:

1. Der Block aus k Informationsbits wird als binäres Polynom c(z) geschrieben. Das zuerst zu sendende Zeichen erhält dabei die höchste Potenz z^(k-1).
2. Das Polynom c(z) der Informationsbits wird mit z^16 multipliziert (das entspricht einer binären Verschiebung um 16 Stellen nach links).
3. Die ersten 16 Informationsbits werden invertiert. Die Inversion entspricht der Operation:
   z^16*c(z) + z^k*(z^15 + z^14 + z^13 + ... + z^2 + z + 1)
   Das bedeutet die 16 obersten Bits des um 16 nach links verschobenen Polynoms c(z) werden invertiert.
4. Das so entstandene Polynom wird durch das Generatorpolynom g(z) binär dividiert.
   Es entsteht ein Quotient q(z) und ein Rest r(z). Der Rest besitzt als höchste Potenz maximal z^15.
5. Der Quotient q(z) wird nicht berücksichtigt. Der Rest r(z) wird invertiert.
   Die Inversion entspricht der Operation:
   r(z) + (z^15 + z^14 + z^13 + ... + z^2 + z + 1)

6. Der invertierte Rest bildet den Prüfblock und wird dem Informationsblock angehängt.

Dieses Verfahren ist mit einem Schieberegister realisierbar.

8.2.2 CRC-Generator mit Schieberegister

Zyklische Prüfcodes lassen sich sehr gut mit Hilfe von Schieberegistern codieren. Dazu gibt es mehrere Möglichkeiten. Für den KISS-Chip wurde die sogenannte systematische Codierung mit Generatorpolynom gewählt (beschrieben in [4]).
Die Coderschaltung (s. Abb. 8.2) besitzt soviele Schieberegisterzellen, wie Prüfbits in Form des Divisionsrests r(z) erzeugt werden sollen, also 16. Sie ergibt einen sog. **systematischen Code**, d.h. einen Code, bei dem die Informationsbits uncodiert bleiben. Das sog. Generatorpolynom ist das CRC-CCITT Prüfpolynom: An den Stellen im Schieberegister, die durch die im Generatorpolynom vorhandenen Potenzen gegeben sind, wird eine binäre Modulo 2-Addition des vorhergehenden Bits mit der Eingabeleitung vorgenommen. Diese Eingabeleitung wiederum wird über eine Modulo 2- Addition des Inhalts der letzten Schieberegisterzelle mit dem als nächstes zu übertragenden Datenbit gesteuert. Dabei werden die Datenbits in Sendereihenfolge, mit der höchsten Potenz in c(z) voran, angelegt. Für jedes Datenbit wird das Schieberegister einmal getaktet. Sind alle Datenbits gesendet, enthält das Register den gewünschten Divisionsrest. Dieser wird dann invertiert als Prüfblock gesendet. Das letzte Schieberegisterbit enthält dabei das erste zu sendende Bit. Um die in Schritt 3 beschriebene Inversion der ersten 16 Datenbits zu realisieren, werden alle Schieberegisterzellen vor Beginn mit Einsen vorbesetzt.

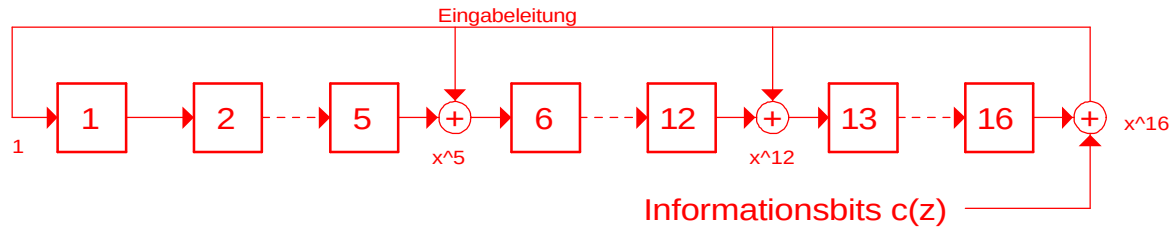

Abb. 8.2: Coderschaltung

8.2.3 CRC-Prüfer

Zur Decodierung des Codewortes im Empfänger kann genau die gleiche Schaltung wie zur Codierung verwendet werden. Da es sich um einen systematischen Code handelt, können die Datenbits in Empfangsreihenfolge in das Schieberegister eingespeist werden und der Inhalt des Schieberegisters nach Empfang des Datenblockes mit dem empfangenen Prüfwort verglichen werden.
Eine weitere Möglichkeit besteht darin, sowohl den empfangenen Datenblock als auch den empfangenen Prüfblock in das Prüfregister einzuspeisen. Wäre der Prüfblock vor dem Senden nicht invertiert worden, müßte sich, bei gleicher Vorbesetzung des Prüf-Schieberegisters mit Einsen, der Divisionsrest Null ergeben, da der gesendete Divisionsrest das Codewort durch das Generatorpolynom teilbar macht. Es würde also ausreichen, nach vollständigem Empfang von Daten- und Prüfblock den Inhalt des Prüferregisters auf Null zu testen. Aufgrund der Inversion des Prüfblockes ergibt sich jedoch nicht Null, sondern die Konstante 61624, die auch entsteht, wenn 16 Eins-Bits in das mit Null vorbesetzte Prüfregister geschoben werden.

Diese Methode ist von Vorteil gegenüber dem Vergleich der Prüfworte, da technisch der Vergleich einer Variable mit einer Konstante auf dem Chip weniger platzaufwendig zu realisieren ist als der Vergleich zwischen zwei Variablen.

8.3 Zufallsgenerator

Zur Steuerung des CSMA-Zugriffsverfahrens wird unter anderem ein Zufallsgenerator benötigt, der 8-Bit Zufallswerte erzeugt. Wesentlich für die Funktion des Zugriffsverfahrens ist, daß die Folge der Zufallswerte sich von Station zu Station unterscheidet.
Prinzipiell ist es jedoch mit einer digitalen Schaltung nicht möglich, echte Zufallswerte zu erzeugen, da sich die Schaltung immer deterministisch verhält. Es ist jedoch möglich, sehr lange berechnete Folgen von Zahlen zu erzeugen. Erst die Einbeziehung nicht genau vorhersehbarer Ereignisse macht eine solche digital erzeugte Folge von Zahlen zu einem guten Zufallsgenerator. Beispiele für solche Ereignisse sind externe Ereignisse wie Empfang eines Tonträgers oder ein Sendewunsch des Computers. Ein häufig verwendeter Trick ist es, die seit dem Einschalten eines Geräts vergangene Zeit in die Erzeugung einer Zufallszahl mit einzubeziehen. Dies kann z.B. erreicht werden, indem der Zufallsgenerator kontinuierlich eine zyklische Folge von Zahlen erzeugt. Die gelesenen Werte hängen dann von den Abfragezeitpunkten ab. Je seltener sich diese Folge wiederholt, desto geringer ist die Chance, daß zwei Stationen nacheinander die gleichen Zahlen erzeugen, falls die Abfragezeitpunkte bei beiden Stationen die gleichen Abstände besitzen.
Eine Folge von Zahlen mit guter Verteilung läßt sich über ein rückgekoppeltes Schieberegister erzeugen (s. Abb. 8.3): Der Folgewert des ersten Bits dieses 20-Bit Schieberegisters soll durch ein Exklusiv-Oder-Gatter aus Bit 3 und Bit 20 erzeugt werden. Die Periodizität der durch ein solches Schieberegister erzeugten Werte ist (2^20)-1, d.h. es generiert in fester Reihenfolge zyklisch alle mit 20 Bit binär darstellbaren Zahlen bis auf die Zahl Null. Dies wurde experimentell bestätigt. Eine Null kann nicht erzeugt werden, da ein einzelnes Eins-Bit im Schieberegister wieder zur Entstehung weiterer Eins-Bits führen muß. (Bei Entstehung einer Null bliebe das Register kontinuierlich auf Null.)

Diese Schaltung führt zyklisch folgende Rechenvorschrift aus:

x' = x div 2
Wenn entweder x ungerade oder aber (x div (2^17)) ungerade, *dann* x = x' + 2^(20-1)
sonst x = x'

Acht der 20 Bits des Registers werden als Zufallszahl interpretiert.
Ein solcher Zufallsgenerator kann auch für andere Schieberegister-Breiten realisiert werden, es ist jedoch darauf zu achten, daß die Einspeisung in das Exklusiv-Oder-Gatter an geeigneten Stellen geschieht, damit wirklich alle Codes durchlaufen werden.
Durch die Wahl einer Anzahl von 20 Bits ergibt sich eine Periodizität von einer Million Zyklen. Die Taktung des Registers soll mit Sendetakt geschehen. Damit wiederholt sich die Zahlenfolge nach frühestens 9 Sekunden (bei 115 KBaud).

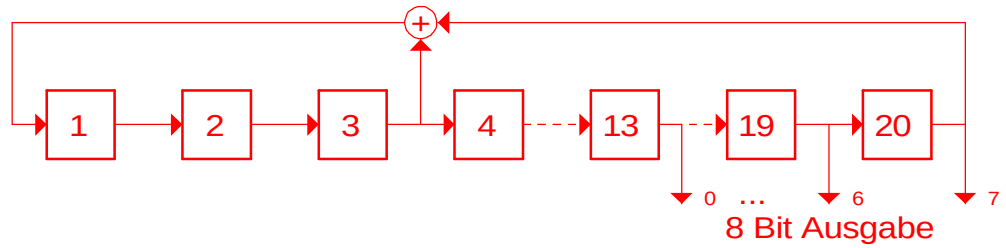

Abb 8.3: Zufallsgenerator

8.4 Digitale PLL

Wie bereits beschrieben, wird auf der Funkstrecke eine synchrone Datenübertragung vorgenommen. Sender und Empfänger müssen also über viele hundert Bittakte synchron zueinander laufen. Dies ist zwar heutzutage durch Verwendung von Quarzen möglich, besser ist es jedoch, die Lage des Bittaktes kontinuierlich genau auf die Mitte des Intervalls, in der das jeweilige Bit am Eingang anliegt, zu korrigieren. Aus diesem Grund wird im Protokoll dafür gesorgt, daß im gesendeten Signal höchstens eine bestimmte maximale Anzahl von Bits mit gleicher Polarität aufeinanderfolgt. Die maximale Anzahl gleicher Bits entsteht beim Aussenden von Flags. Durch die Verwendung des Bitstuffings und der NRZI-Codierung sind dies maximal 7 gleiche Bits (außer bei Aussendung eines Breaks, hier ist jedoch die genaue Anzahl von Eins-Bits unwichtig).
Ein Flag (01111110) ergibt NRZI-codiert: 100000001 oder 011111110.
Zur Regeneration des Bittaktes aus dem empfangenen Datenstrom kann eine sogenannte **PLL** (Phase Locked Loop) verwendet werden.

8.4.1 Funktionsweise einer PLL

Aufgabe einer PLL ist es, einen Takt zu erzeugen, dessen Frequenz durch Vergleich der Phasenlage mit einem Referenztakt phasenkonstant nachgeregelt wird. Eine solche PLL besteht im allgemeinen aus einem Phasenkomparator und einem steuerbaren Oszillator. Je nach dem, ob die Phase des erzeugten Taktes gegenüber dem Referenztakt nacheilt oder vorauseilt, wird die Frequenz des Oszillators geringfügig erhöht bzw. erniedrigt. Da normalerweise über eine analoge Spannung steuerbare Oszillatoren eingesetzt werden, soll zur Unterscheidung im folgenden die Abkürzung DPLL für eine voll digital arbeitende PLL benutzt werden.

Anwendungsgebiete für eine PLL sind:

- Frequenzvervielfachung: Hier wird ein über einen Frequenzteiler herabgeteilter Oszillatortakt auf die Referenzfrequenz geregelt.
- FM-Demodulation: Am Phasenkomparator-Ausgang entsteht das demodulierte Signal.
- Taktregeneration aus sporadischen Synchronisationsimpulsen

8.4.2 Implementation der DPLL

Zur Erzeugung des Bittaktes im SDLC-Empfangsteil muß der Sendetakt anhand der Daten regeneriert werden. Da der Bittakt von Sender und Empfänger bereits quarzgenau übereinstimmt (d.h. auf einige 100 ppm genau), muß (fast) nur die Phasenlage des Bittaktes anhand

der Polaritätswechsel am Dateneingang so korrigiert werden, daß der Takt möglichst um 180 Grad verschoben zu den Pegelwechseln liegt. Dann ist Abtastsicherheit optimal (s. Abb. 8.4.1).

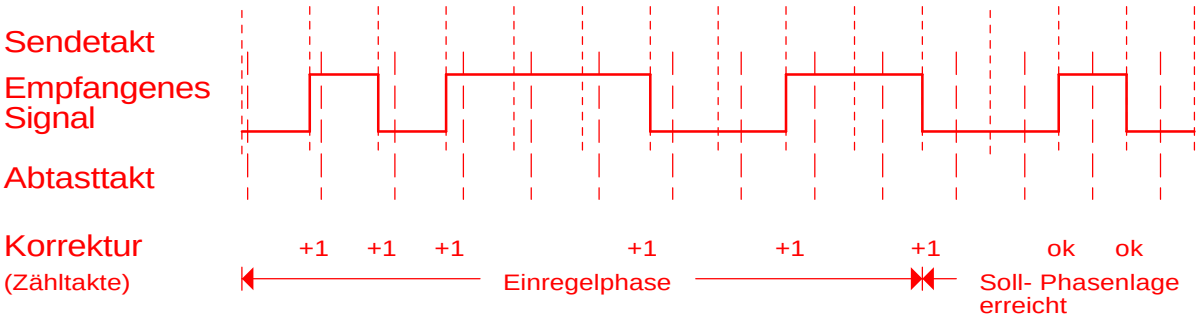

Abb. 8.4.1: Abtastung des SDLC-Signales

Die Veränderung der Phasenlage kann durch kurzzeitige geringfügige Erhöhung oder Erniedrigung der Frequenz geschehen. Eine Möglichkeit dies zu tun ist es, bei Herabteilung der Quarzfrequenz auf die Sollfrequenz, zur Phasenkorrektur wahlweise kurzzeitig den Divisor zu erhöhen oder zu erniedrigen. Damit dies immer in definierten Abständen vorgenommen werden kann, liefert der Baudratengenerator ein Vielfaches der Baudrate.
Hier soll es möglich sein, den Mittelpunkt des Bitfensters auf 1/32 genau zu treffen. Dazu wird der 32-fache Empfangstakt durch einen 5-Bit Zähler herabgeteilt. Beim Übergang des Zählers von 15 nach 16 wird der Eingangspegel eingelesen und als empfangenes SDLC-Bit interpretiert (s. Abb. 8.4.2). Um den Mittelpunkt des Bitfensters zu treffen, muß die PLL also den Takt so teilen, daß Pegelwechsel zwischen Zählerstand 31 und 0 auftreten (der Binärzähler läuft von 31 auf 0 über). Findet also ein Wechsel des Eingangssignales zwischen Zählerstand 0 und 15 statt, muß die PLL langsamer zählen, also einen Zähltakt auslassen. Tritt der Pegelwechsel zwischen Zählerstand 16 und 31 auf, muß die PLL schneller zählen, also einen Zählerstand überspringen. Das Auslassen eines Zähltaktes soll jeweils sofort nach dem dies verursachenden Pegelwechsel geschehen, während das Überspringen eines Zählerstandes einfach dadurch realisierbar ist, daß der Zähler beim nächsten Durchlauf den Zählerstand 0 ausläßt.
Dieses System ist unempfindlich gegenüber einzelnen Fehlimpulsen auf der Datenleitung, da immer nur ein Takt eingefügt oder ausgelassen wird. Zur Synchronisation werden also maximal 16 Eingangspegelwechsel benötigt.

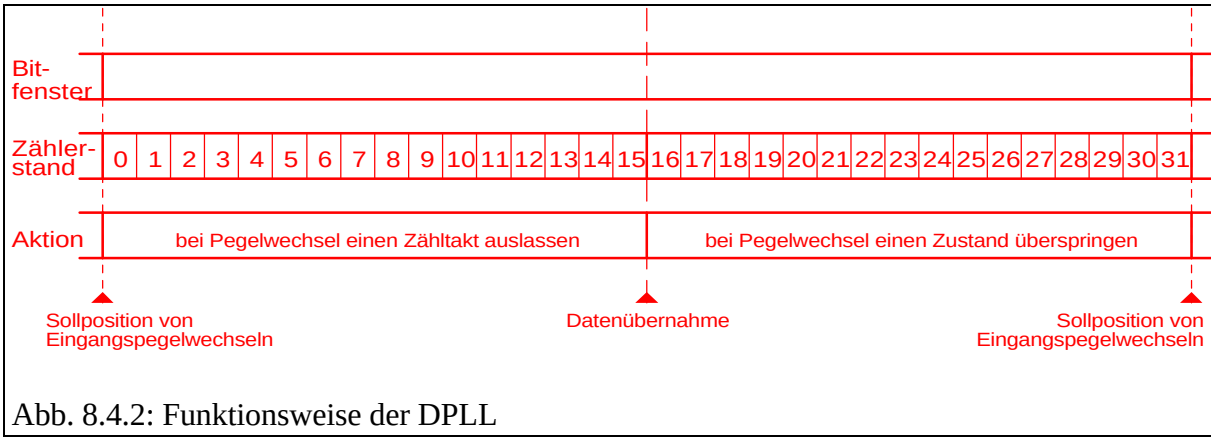

Abb. 8.4.2: Funktionsweise der DPLL

8.5 RS 232-Eingangsteil

Während der RS 232-Ausgangsteil im wesentlichen nur aus einem mit der RS 232-Baudrate getakteten Schieberegister besteht, muß sich der Empfangsteil an die Phasenlage des fremden Sendetaktes anpassen. Dazu wird wie beim SDLC-Eingangsteil mit einem Vielfachen der Baudrate als Referenzfrequenz gearbeitet. Auch hier ist es wünschenswert, den Eingang möglichst mittig bezüglich des Bitfensters abzutasten. Eine Synchronisation während des Empfanges ist jedoch nicht nötig, da die RS 232-Schnittstelle byteorientiert arbeitet. Nach jeweils 10 Bits wird wieder ein Startbit empfangen.

Der Eingangsteil ist folgendermaßen implementiert (s. Abb. 8.5):
Mit dem 16-fachen der Baudrate wird das Eingangssignal (invertiert durch den Leitungsempfängerbaustein) in ein 3-Bit Schieberegister geschoben. Zweck dieses Schieberegisters ist es, Störungen auf der Eingangsleitung zu unterdrücken.

Der Empfang startet, sobald (im Schieberegister) ein 1-nach-0-Übergang erkannt wird. Daraufhin wird der 4-Bit Zähler, der den Takt (durch 16) auf die Baudrate herabteilt, auf 0 gesetzt. Das Rücksetzen macht seine Überläufe synchron zu den Grenzen der empfangenen Bits.
Beim 9-ten Zustand der Zählers enthält das 3-Bit Schieberegister die Eingangswerte der 7-ten, 8-ten und 9-ten Abtastung. Der für ein Bit akzeptierte Wert ist der Logikpegel, der bei mindestens zwei Abtastungen auftrat. Wenn dieser Wert beim ersten Mal nicht 0 ist, wird weiter nach einem 1-nach-0-Übergang gesucht. Ist der Wert 0, so ist das Startbit als gültig erkannt worden und das RS 232-Empfangsregister (ein 9-Bit Schieberegister) wird mit 011111111 vorbesetzt. Das gültige Startbit ist dann also bereits in Bit 8 des Empfangsregisters und der Rest des Bytes kann empfangen werden.

Sowie links Daten in das Empfangsregister hereingeschoben werden, werden rechts Einsen herausgeschoben. Wenn das Startbit an der ganz rechten Position im Schieberegister angekommen ist, signalisiert es der Steuerung, noch einmal zu schieben. Das dann noch zu empfangende Bit ist das Stopbit. Nur wenn dieses ordnungsgemäß 1 ist, wird das empfangene Datenbyte, das jetzt in den Bits 0 bis 7 steht, an die weiteren Einheiten übergeben. Unabhängig davon wird jetzt wieder das nächste Startbit erwartet.

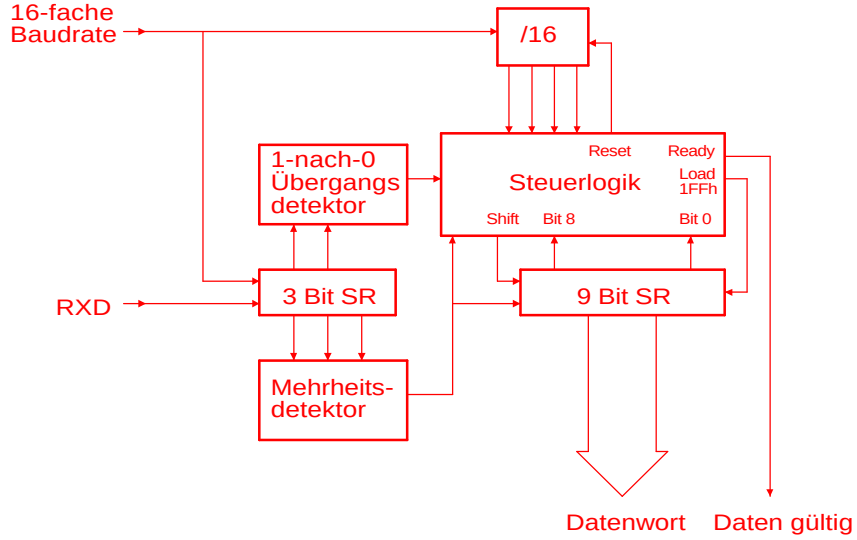

Abb. 8.5: Blockschaltbild des RS 232 Eingangsteils

9. Danksagung

Hiermit möchte ich Herrn Gunter Hille meinen Dank aussprechen für die vielen fruchtbaren Gespräche und für die Unterstützung beim Test des Simulators. Für die Unterstützung im Hinblick auf eine spätere technische Realisierung möchte ich Herrn Reinhard Rauscher danken.

A1 Anschlüsse des Chips

Der fertige KISS-Chip soll über folgende Anschlüsse verfügen:

Pin	Funktion	Kommentar
GND	Masse	Bezugspunkt für alle Signale
VCC	Versorgung	+5V Versorgungsspannung
TEST	Testmodus Eingang	Eingangssignal zu Einschalten des Testmodus (High = Testmodus an)
SCANIN	Scanpath-eingang	Eingangssignal des Test-Pfades zur Funktionsüberprüfung, bei Betrieb mit GND oder VCC verbinden
SCANOUT	Scanpath-ausgang	Ausgangssignal des Test-Pfades zur Funktionsüberprüfung, bei Betrieb offen lassen
CLK	Takteingang	Alle internen Einheiten des Chips werden mit diesem Takt getaktet. Durch Frequenzteilung werden die Baudraten für RS 232- und SDLC-Schnittstelle aus CLK gewonnen. Empfohlene Frequenz: 3,6864 MHz
RESET	Initialisierung Eingang	Ein HIGH-Pegel an diesem Eingang initialisiert den Chip auf Standardwerte und beendet alle Aktionen
TXD	RS 232 Daten Ausgang	Serieller Datenausgang für RS 232-Schnittstelle, extern mit invertierendem Leitungstreiber zu beschalten
RXD	RS 232 Daten Eingang	Serieller Dateneingang für RS 232-Schnittstelle, mit externem invertierenden Leitungsempfänger verbinden
/CTS	RS 232 Hand-shake Eingang	Handshake Eingang für RS 232-Schnittstelle (LOW = Chip darf senden), extern mit invertierendem Leitungstreiber zu beschalten
/RTS	RS 232 Hand-shake Ausgang	Handshake Ausgang für RS 232-Schnittstelle (LOW = Chip empfangsbereit), mit externem invertierenden Leitungs-empfänger verbinden
/START	Sendestart Eingang	Bedeutung nur bei Wahl von TXDelay=0: Chip startet mit dem Senden von SDLC-Daten, sobald Signal LOW
SDOUT	SDLC Ausgang	Serieller Datenausgang für SDLC-Schnittstelle (NRZI-codiert)
SDIN	SDLC Eingang	Serieller Dateneingang für SDLC-Schnittstelle (NRZI-codiert)
SDCLK16	Hilfstakt Ausgang	Hilfsausgang: 16-facher SDLC-Sendetakt
CDIN	Carrier Detect Eingang	Signal vom Modem (HIGH = Datenträger wird empfangen)
PTT	Push To Talk Ausgang	Ausgang zum Einschalten des Senders (HIGH = Sender einschalten)

A2 Schaltungsaufbau

Der folgende Schaltplan zeigt, wie der KISS-Chip eingesetzt werden soll:

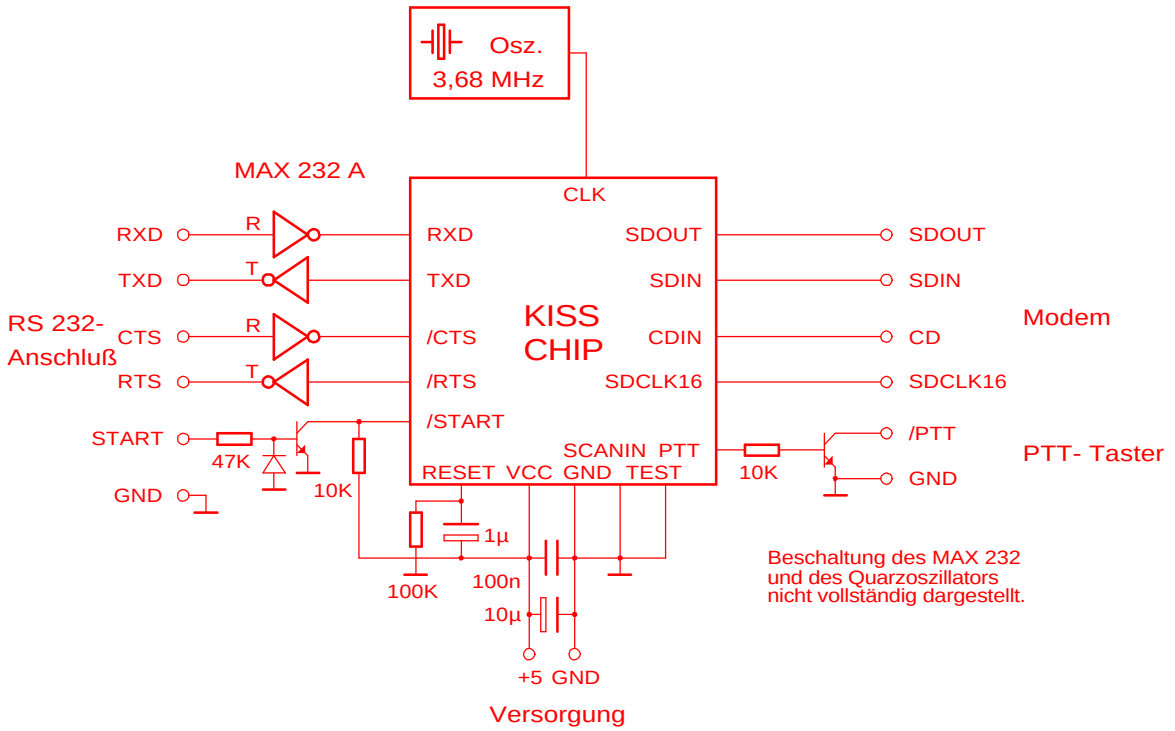

Abb. A2: Schaltungsbeispiel für KISS-Chip

A3 Aufteilung in Programmteile

Der Verhaltenssimulator ist in mehrere Teile aufgeteilt:

- Verhaltenssimulator: CHIPVERH.PAS (Unit)
- Testprogramm für Verhaltenssimulator: TESTCHIP.PAS
- Verhaltenssimulator für Taktteilung und serielle Eingangsteile: TAKTVERH.PAS
- Testprogramm für Zufallsgenerator: TESTRAND.PAS
- Testprogramm für CRC-Generator/ Prüfer: TESTCRC.PAS
- Hilfsroutinen zur Bitmanipulation usw.: BITOPS.PAS (Unit)

(Die als Unit gekennzeichneten Teile sind nicht für sich alleine lauffähig.)

Das Programm TESTCHIP.PAS testet die wesentlichen Routinen des Verhaltenssimulators. Es sendet Blöcke als KISS-Daten und speist die durch den Verhaltenssimulator nach AX.25 konvertierten Daten zurück in den SDLC-Eingang des Simulators ein. Gleichzeitig werden auf der KISS-Schnittstelle zurückgelieferte Daten empfangen und angezeigt. Durch Veränderung von Parametern oder Auskommentierung von Programmteilen kann das Verhalten in unterschiedlichen Situationen getestet werden.

A4 Abkürzungsverzeichnis

Abkürzung	Abschnitt
AX.25	2.1
CRC	8.2.1
CSMA	2.2
DPLL	8.4.1
FEND	1.1, Tab. 1.1
FESC	1.1, Tab. 1.1
FIFO	6., 8.1
HDLC	2.1
KISS	0., 1.1
Modem	0., 3.
NRZI	2.1.2
PC	0.
PLL	8.4
PTT	3.
RS 232	1.2
SDLC	2.1
SLIP	1.1
TNC	0., 4.
UART	7.1

A5 Literaturverzeichnis

[1] Grundlegende Kiss Chip-Spezifikation von Gunter Hille 1991

[2] AX.25 Amateur Packet-Radio Link-Layer Protokoll Vers.2

[3] Proposed "Raw" TNC Functional Spec - 6 August 1986, Phil Karn

[4] Fehlerkorrigierende Block-Codierung für die Datenübertragung, F.J. Furrer, Birkhäuser Verlag 1981

[5] 9600 Baud FSK Modem Technik nach dem G3RUH-Standard, Johannes Kneip (HRSG)

[6] AMD 7911 FSK Modem Datenblatt, Advanced Micro Devices 1985

[7] Zilog Z8030 / Z8530 Serial Communications Controller Manual, Zilog

[8] AMD 8051 Microcontrollers 1988 Data Book, Advanced Micro Devices 1988

[9] TI EIA-232 Interface Circuits Application and Data Book, Texas Instruments 1992

[10] Computerschnittstellen, L.Preuß, H.Musa, Hanser Verlag 1989

[11] 1993 New Releases Data Book Volume II, Maxim 1993