

digitalisierungen

SC5BOS

Scheduling C5 Basic Operating System

Version 0.30

Ein Projekt von
DB1OTM, Thomas Müller
DO1FJN, Jan Alte

Inhalt:

| | |
|---|----|
| Terminologie im Dokument..... | 3 |
| Einleitung..... | 3 |
| warum eine weitere Umbauanleitung? | 3 |
| Ziele..... | 3 |
| Beschreibung des Projektes | 3 |
| Lastenheft SC5BOS..... | 5 |
| Ideensammlung für Kreative..... | 5 |
| Hardware..... | 5 |
| Software..... | 6 |
| mögliche höhere Programmfunktionen..... | 6 |
| Umbau der digitalen Hardware..... | 7 |
| Umbau der analogen Hardware..... | 8 |
| Konzeptdokumentation SC5BOS..... | 9 |
| Betriebssystemerfordernisse: | 9 |
| Bedienungsanleitung | 9 |
| Einschalten..... | 9 |
| Ausschalten..... | 9 |
| Die Bedienung des Bootloaders..... | 9 |
| Erweiterte Funktionen „Programmiermodus“ | 11 |
| technische Beschreibung des SC5BOS | 12 |
| Funktionen und Fähigkeiten des SC5BOS und des Bootloaders..... | 12 |
| Kommunikationsprotokoll..... | 12 |
| well-known-ports..... | 12 |
| Application Interface..... | 12 |
| 0xh – Steuerungsfunktionen | 13 |
| 1xh – BFBUS – Zugriff und Abfrage..... | 13 |
| 2xh – Serielle I/O-Funktionen (SerIO und I ² C-Bus)..... | 13 |
| 3xh – Informations- Konvertierungsfunktionen..... | 14 |
| 4xh – DSP-Funktionen..... | 14 |
| Handler | 15 |
| Programmieranleitung..... | 16 |
| Speichermodell..... | 16 |
| Aufruf von API-Funktionen..... | 16 |
| Benutzen eines eigenen Handlers..... | 16 |
| Informationen zur C5-Hardware..... | 17 |
| Speicheraufteilung | 17 |
| Der EP200 | 17 |
| Clockgenerierung | 18 |
| EP200-Ports | 18 |
| Übersicht des Speichermapping | 18 |
| Portfunktionen des NEC und des EP200 | 18 |
| Serielle Schnittstellen | 18 |
| BFBUS..... | 18 |
| Kartenleser..... | 20 |
| externe Schnittstelle..... | 20 |
| I ² C-Bus und Slave-IC's..... | 20 |
| I ² C-Controlregister..... | 20 |
| I ² C-Timing..... | 21 |
| I ² C-Bausteine im C5 | 21 |
| Motorola DSP 56001 / 56002 | 22 |
| Host-Interface | 22 |
| SCI/SSI-Ports..... | 22 |
| Der Codec-Signalweg..... | 23 |
| Zeichensatz des Standard-Hörers | 23 |
| Zeichensatz des Standard-Hörers | 24 |
| Tastaturcode des Standard-Hörers..... | 24 |
| Der Empfänger..... | 24 |
| Die AFC Steuerung..... | 24 |

Terminologie im Dokument

- Bootloader Teil des SC5BOS der mit dem Benutzer interagiert.
- C5 Bezeichnet das Telefon Siemens C5 (auch ABB C45-5)
- SC5BOS Sheduling C5 Basic Operation System, das Betriebssystem welches hier vorgestellt wird
- SC5PRG Sheduling C5 Program, ein x86-Program das im einem C5 mit SC5BOS funktioniert.

Einleitung

warum eine weitere Umbauanleitung?

Als unbedarfter Leser und Informatik-Interessierter fragte sich der Autor vor ca. einem halben Jahr, wieso man bei einem Gerät dessen Digitalteil vollständig entfernt und eine andere Platine aufwendig einsetzt, obwohl doch auch der neue Controller auf der eingesetzten Platine programmiert werden musste. Jedoch merke der Autor schnell, das dies nicht ohne Grund geschah:

Zum original Digitalteil des C5 gehört ein GAL-Baustein mit der Bezeichnung EP200. Zu diesem zentralen Baustein gibt es keine Entwicklungsunterlagen, jedoch die Aussage, das dieser Baustein fehlerhaft arbeitet und diese Fehlfunktionen per Software ausgeglichen werden müssen. Ebenfalls eine entscheidende Rolle spielt der Motorola DSP 56001 (in den letzten C5: 56002). Es ist keine simple Aufgabe Software für einen Signalprozessor zu entwickeln. Auch der Autor hat bisher nur ein paar Erfahrungen mit einem AD2181 (EZlite-Kit).

Der Grund sich dennoch für die Entwicklung von solcher Software zu begeistern liegt zu einem in der Herausforderung dieses bewerkstelligen zu können, zum anderen in dem Gedanken, damit etwas für den Amateurfunk getan zu haben. Auch ist die Kombination Controller – DSP ein leistungsfähiges Gespann, dessen Potential hoffentlich bald nutzbar gemacht wird.

Ziele

Bei diesem Projekt geht es vorrangig darum, kostengünstig und mit minimalem Aufwand das Siemens C5 Telefon in einen FM-Amateurfunktransceiver für das 70cm Band (430-440 MHz) zu verwandeln. Eine sehr detaillierte Anleitung zu einem C5-Umbau stellt DL6INT (Uwe Henning) unter

<http://home.t-online.de/home/dl6int/c5/c5idx.htm>

zur Verfügung. Leider erfordert der Umbau des C5 nach dieser Anleitung einigen Aufwand in der Realisierung. Da der Autor Informatik studiert hat, und sich Hobbymäßig mit Amateurfunk beschäftigt, kam er auf die Idee ein kleines Betriebssystem für das C5 zu entwickeln. Dieses Vorhaben wurde durch die Veröffentlichung einiger Details des C5-Software-Innenlebens durch DB1OTM (Thomas Müller) unter

<http://www.qsl.net/db1otm/c5.html>

unterstützt. Es entstand eine Zusammenarbeit, die auf einen gegenseitigen Informationsaustausch beruhte. Basis der Entwicklung ist übrigens die von DB1OTM disassemblierte Originalsoftware, die einige Fragen beantwortet, jedoch 10mal so viele Neue aufwirft.

Leider existiert für den umbaubegeisterten Funkamateurlimmer noch keine „fertige“ Software, die den Betrieb des C5 als Funkgerät unter dem hier beschriebenen Betriebssystem ermöglicht. Größtes Hindernis ist neben dem Zeitmangel des Autors (Diplomarbeit!) die Entwicklung eigener DSP-Software. Hier kümmert sich jedoch DB1OTM und der Autor um entsprechende Lösungen für Phonie, PacketRadio und Funkruf. Für Funkamateure, die ihr C5 gleich verwenden möchten, ist daher z.Z. nur die Anleitung nach DL6INT empfehlenswert. Eine Software für unsere Umbauanleitung wird voraussichtlich in ca. einem halben Jahr fertiggestellt sein.

Beschreibung des Projektes

Das Projekt „SC5BOS“ beschäftigt sich mit der Softwareentwicklung für das alte C-Netztelefon Siemens C5. Später werden wahrscheinlich noch SC3BOS / PP5BOS parallel und „softwarekompatibel“ mitentwickelt (für die Telefone Siemens C3 / Philips Porty FG51 bis FG53). Jedoch fehlen für dieses Unterfangen noch Schaltungsunterlagen und vor allem Zeit. Alle Telefone verfügen über einen NEC V25 oder V40 Microcontroller (µC). Der NEC V25 / V40 ist softwarekompatibel zu einem i80186 und besitzt zusätzlich noch erweiterte Befehle. Er ist ansonsten ein klassischer Controller (keine CPU) mit UART, Ports, DMA etc.). Software kann deshalb mit „alten“ PC-Assemblern und Compilern erstellt werden. Das C3 und das C5 haben neben dem µC noch einen DSP 56001 von Motorola onboard. Dieser ist über das Host-Interface und der „glue logic“ (EP200 im C5) mit dem µC verbunden.

Der Autor hat die hier vorgestellte Software mit dem A86-Assembler und dem jloc-Linker erstellt. Der A86-Assembler kennt ein paar der zusätzlichen NEC-Befehle (leider nicht alle) und verfügt über eine simples Segmentierungs-System. Der jloc-Linker wertet die von A86 erstellte OBJ-Datei aus und erzeugt mit Hilfe einer Kontrolldatei ein binäres Image des Programmes. Als „Entwicklungsumgebung“ diente der Ultraedit (<http://www.ultraedit.com/>), der Sysntaxhighlighting und das Einbinden von Programmen (hier z.B. make) beherrscht. Leider ist dieses Programm Shareware, dessen Registrierung mit \$35 noch human ausfällt.

Neben der Softwareentwicklung konnte sich der Autor auch nicht um Hardwareentwicklung drücken. Um das C5 mit dem PC zu Verbinden, wurde ein serieller Pegelwandler (TTL \leftrightarrow RS232) benötigt. Als einfache Lösung wurde ein Max202 und ein kleiner 5V-Stabi auf einer Lochrasterplatine zusammen mit 2 Pfostensteckern gelötet. Die Beschaltung entspricht der im Max202-Datenblatt. Angeschlossen wird diese Baugruppe an die 26pol. Buchse des C5 (HDB26 benötigt). Über diese Buchse erfolgt auch gleich die externe Stromversorgung. Folgende Pins müssen mit dem Max202 verbunden werden: V24-TXD (Pin13) und V24-RXD (Pin20).

Neben dem Seriellen Anschluss ist leider auch ein Programmiergerät erforderlich gewesen, um einen leeren Flash-Baustein mit der Software zu beschreiben. Erst mit dem programmierten Flash konnte erstmalig „eigene“ Software ins C5 gelangen. Als Flash verwendet der Autor den PLCC32-Typ 29F040 der Firmen AMD, TI und Hyundai. Alternativ einsetzbar sind auch die Typen 29F002TC / BC. In jedem Fall muss das Original-ROM aus dem C5 entfernt und eine PLCC32-Fassung eingelötet werden (wenn nicht schon vorhanden).

Software:

Die Software SC5BOS ist nicht das einzige was im Laufe der Entwicklungsarbeit entstand. Neben SC5BOS sind eine Reihe von Testprogrammen, die im C5 ausführbar sind entstanden. Dazu zählen u.a. FlashWRT, DSPTest, C5Demo, I2CTest und C5Test. Nur letzteres wird weiterentwickelt, da die anderen Programme ihre (Test-) Aufgabe erfüllt haben. Einige Features in Kurzform:

- C5 könnte komplett per PC gesteuert werden
- neue Software wird einfach bei geschlossenem Gerät ins C5 übertragen
- selbst SC5BOS ist so updatefähig
- mehrere Dienste können parallel laufen (z.B. PacketRadio + Debugger + Bediensoftware)
- PacketRadio wird ohne Modem möglich sein (1k2 sicher, 9k6 und 19k2 muss überprüft werden)
- Entwickler brauchen sich nur noch um die eigentliche Aufgabe des Programms kümmern, den Rest erledigt dann SC5BOS (I/O, Tastenabfrage etc.)

Unweigerlich entstand auch PC (Windows-) software: Diese unter Delphi5 geschriebenen Tools sind für die Kommunikation mit dem C5 (oder „kompatiblen“ Geräten) gedacht. Kern der Kommunikation ist ein als PCP (Packet Communication Protocol) bezeichnetes Übertragungsprotokoll. Es spezifiziert eine auf Paketen aufgebaute Kommunikation zwischen 2 Geräten (hier C5 und PC). Eine nähere Beschreibung folgt später. Die Fähigkeiten der Windows-Software in Kurzform:

- OLE-Server für die Kommunikation (PCP) über eine serielle (Standart-) Schnittstelle-
- Einsehen vom C5 RAM und ROM per PCPMemoryViewer
- Ausgabe von Textmeldungen des C5 per PCPMessageViewer
- Übertragen und ausführen von Programmen im C5 per C5MemoryLoader
- Steuern von C5Test (noch im C5MemoryLoader)
- Beschreiben des Flash-ROM per PCPMemoryViewer

(Diese Software wird in einem extra Dokument behandelt.)

Der eigentlicher Kern der Entwicklung – SC5BOS – wird im Folgenden beschrieben.

Lastenheft SC5BOS

- das SC5BOS soll in einem 16Kbyte großen geschützten Sektor des Flash-Roms Platz finden
- es initialisiert die nötige Hardware, die für eine Programmierung des Flash-Bausteins nötig ist dazu zählen:
 - Prozessorregister
 - externe serielle Schnittstelle (57600baud, 8N1)
 - Interrupt-Vektortabelle
- es soll ein einfaches Kommunikationsprotokoll implementiert sein, das eine sichere Datenübertragung gewährleistet. Daher ist eine CRC16 wünschenswert.
- Der Telefonhörer soll bereits Meldungen ausgeben können
- Über die Tastatur des Hörers sind im Bootloader Informationen abrufbar und Testfunktionen ausführbar. darunter fallen:
 - Versionsausgabe
 - CRC16 des Flashbausteins
 - Flash löschen (bis auf SC5BOS)
 - Test der Seriellen Schnittstelle 0 (für ext. Kommunikation wichtig)
 - Einstellen der Übertragungsparameter, abweichend vom Standard (z.B. 9600baud).
 - Speichertest
 - Stacktest (Größe anzeigen)
- der Funktionsumfang ist einfach zu erweitern
- Über das Kommunikationsprotokoll kann ein Programm in den Speicher geladen und ausgeführt werden.
- Über das Kommunikationsprotokoll kann ein Programm in den Flash geschrieben werden.
- Über das Kommunikationsprotokoll kann ein beliebiger Datenblock in den Flash geschrieben werden.
- Über ein API werden Basisfunktionen den Anwendungsprogrammen zugänglich gemacht.
- Eintreffende Daten sind über eine Event-Schnittstelle dem laufenden Programm zu übergeben.
- ...

Ideensammlung für Kreative

Hardware

- Als Programmspeicher sollte ein Flash eingesetzt werden, das ein Bootloader enthält. Ein Update der Software ist dann über eine serielle Verbindung zum C5 möglich.
- Statt Hörer Anschluss von Bedienteilen
 - Display 4x20 Zeichen oder Grafikpanel (320x240 dots)
 - Impulsdrehgeber für Volume – Frequenz (Umschaltung durch Drücken, Timeout)
 - 10er Tastatur mit aufgedruckten Buchstaben (wie „Handys“)
 - Buchstabeneingabe über Drehgeber, Tasten
 - 4 Softkeys am Display-Rand
 - Mikrofonanschluss 2,5 Klinke
 - NF-Out oder kleine NF-PA für 4Ohm-Lautsprecher
- Benutzung der „Office“-Platine (TAE-Anschluss)
- Ethernet-Anschluss optional, sonst Seriell mit 115k2-Tauglichkeit
Es hat sich herausgestellt, das durch die Taktfrequenz des NEC nur 57600baud möglich sind. Höhere Taktraten sind auch möglich, jedoch keine, die ein Standard-UART unterstützt.
- Ethernet-Controller -> Voice over IP, Fernbedienung, PacketRadio etc.
- IrDA optional
- Zusatzanschlüsse auf wechselbarer Platine, von Software Erkennbar (Adresse etc.)
- Stationäres oder ins Auto installiertes Gerät ohne Akkus, dafür mit Festplatte für MP3, Anrufe, PR-Mails
Statt Festplatte wäre eine Smart Media Card (SMC) Kartenadapter realisierbar. Die Ankopplung an den Bus ist leicht und ein FAT12, FAT16 – Dateisystem einfach zu implementieren
- Kopplung eines anderen Funkgerätes 2m, CB etc. statt Hörer: Relaisbetrieb!
- 4 Cinch-Eingänge auf 4 Cinch-Ausgänge zum Durchschleifen von AutoRadio-NF
 - Analoger NF-Mischer

Software

- Beibehaltung des Digitalteiles rund um den NEC V25 und dem DSP
- FM-Empfang 12,5 / 25 kHz – Raster
- HUB umschaltbar
- HF-Ausgangsleistung veränderbar
- Abschaltbare digitale VOX
- Erzeugen und Verarbeiten von
 - CTCSS, DTMF, 1750Hz Tönen
- PacketRadio Modem-Funktion für 1k2 / 9k6
- SSTV-Unterstützung
- Kartenschlitz für (Krankenkassen-)Speicherkarten nutzbar
- Ablage von Parametern auf o.g. Karten
- Timemultiplexer Betrieb von PR und Voice
- Klingelfunktion
 - Über PR-Packet
 - Über CTCSS-Ton
 - Über DTMF-Folge
- Funkruf (Pager) Empfangsfunktion
- DSP-Sprachverbesserungsfähigkeiten
 - Rauschunterdrückung, -Sperrung á la FM-Select
 - Bandpassfilter, Notchfilter
 - Sprachinvertierung (C-Netz-Altlast ☺)
- Aushandeln von Sendeleistungen mit Simplexbetrieb per PR-Pakete
- Aussenden von APRS (Serieller Eingang für GPS-Antenne)
- HDS (HamDataSystem) – RDS ähnlich implementieren
- Kanal Scheduling der einzelnen digitalen Funktionen
- 2m MiniEmpfänger (aus Funkamateure 9/01 S.999) eingebaut
 - ohne ATME1 und NF-PA, Potis NF an DSP-Eingang!
- FFT-Berechnung (für Version mit Grafikdisplay)
- Automatik um Hörer, Bedienteilversionen oder PC zu Unterscheiden
- DTMF-Gesteuerter Anrufbeantworter -> Über ext. Speichermedium (SMC, Smartcard, HDD)
- Fernabfrage Anrufbeantworter über anderes Funkgerät
- Bei Funkgerätekopplung als Relais konfigurierbar, Fernbedienung über DTMF (Ausgabe QSY)
- CW-Dekodierung und Darstellung im Display (Kennungen von Relais)
- Über Ser. Schnittstelle Nordlinks TF-Emulation mit erweitertem Spezialbefehlssatz (@C5 ...) als Standard, 6Pack wählbar)
- Relais und DigipeaterListe (Updatebar) von EU/DL
 - durch Eingabe des Locators sind 70cm Relais auswählbar (im Umkreis erreichbare)

mögliche höhere Programmfunktionen

- | | |
|--|--|
| • Empfangsfrequenz einstellen | 1 Byte Frequenz (12,5 kHz Raster) |
| • Sendefrequenz einstellen | 1 Byte Frequenz (12,5 kHz Raster) |
| • Hub-Begrenzer einstellen | 1 Bit : (eng: 12,5kHz weit: 25kHz) |
| • Sendeleistung einstellen | 1 Byte |
| • RSSI abfragen | → 1 Byte |
| • Lautstärke Hörer einstellen | 1 Byte skaliert (0=Mute) |
| • Lautstärke LS-Gerät einstellen | 1 Byte skaliert (0=Mute) |
| • Lautstärke Freisprech (Extern) | 1 Byte skaliert (0=Mute) |
| • Mikrofonempfindlichkeit Hörer Kapsel | 1 Byte |
| • Mikrofonempfindlichkeit Hörer Freisprech | 1 Byte |
| • DSP-Grundprogramme aktivieren | 1 Byte Aufzählungstyp (0="IN to OUT" ...) |
| • DSP-Programm ausführen | SEG:OFS des Programmes |
| • Serielle Schnittstelle Ausgabe | SEG:OFS Datablock, 1 Word Länge |
| • Serielle Schnittstelle Mode | 1 Byte Baudrate, 1 Mode-Byte, 1 Byte FIFO-Rate |
| • Serielle Schnittstelle Zeichenausgabe | 1 Byte Zeichen |
| • Serielle Schnittstelle Zeichen einlesen | → 1 Byte |
| • Serielle Schnittstelle Status | → 1 Status-Byte |

Eine alternative Möglichkeit besteht darin, ein Stück Kupferdraht (1-1,5mm) als Schlaufe um die Beinchen des IC's zu legen und reichlich zu verzinnen. Durch reichlich Wärme können so alle Pad's gleichmäßig erhitzt werden und der IC ist mit einer Zange, Haken o. ä. abnehmbar.

Umbau der analogen Hardware

Dieser Umbau erfolgt weitestgehend nach DL6INT und berücksichtigt die dort gesammelten Erfahrungen.

Oszillator der PLL verstimmen

Für das Amateurfunk - Frequenzband muss die Resonanzfrequenz des Oszillators um etwas 33 MHz verringert werden. Der originale Oszillator schwingt bei etwas 415 MHz und soll für unsere Zwecke bei etwa 395MHz arbeiten. Dies erreicht man, indem man ca. 10mm 0,8 CuAg-Draht über einem 4mm-Bohrerschaft zu einem Halbkreis biegt und diesen anstatt des Anschlussbeins zum Verbinden des Kreises mit der Platine benutzt. Das alte Anschlussbein sollte so vollständig wie möglich abgesägt werden (so das nur noch der Steg übrigbleibt – Höhe des Innenleiters). Ich rate ausdrücklich zum Absägen mit einer kleinen Trennscheibe, da die Methode das Bein mit einem kleinen Seitenschneider zu durchtrennen schiefgegangen ist: Durch die mechanische Belastung ist der Leitungskreis von der Platine abgerissen – siehe Achtung. Eine andere Alternative wäre das Bauteil mit einem Draht, der U-förmig über den Oszillator gelegt und mit der Platine verlötet wird, zu sichern. Um einen größeren Abstimmbereich zu bekommen wird noch C136 von 8p2 auf 22pF vergrößert.

Nun kann die LO-Frequenz mit dem Trimmer links neben dem Leitungskreis abgestimmt werden. Leider hat dieser Trimmer einen sehr unschönen Hacken: Der Abgleichstift benötigt eine quadratische Spitze mit nur 0,63 mm (**messen!**) Kantenlänge. So was hat keine Hobby- und auch kaum eine Profiwerkstatt. Es wurde daher für den Umbau extra ein solches Teil aus einem Messingbolzen gefräst.

Da das C5 wegen den C-Netz Duplexbetrieb eine um konstant 10MHz geringere Sendefrequenz gegenüber der Empfangsfrequenz erzeugt, muß die PLL bei Empfangsfrequenzen von 428 bis 452 MHz sauber einrasten. Gemessen wird das über die Regelspannung (TP 70), die zwischen -3,5 bis +3,5 Volt liegen sollte. Alternativ gibt der TBB200 ein pulswiden moduliertes Signal auf LD (Pin 14) aus: Je kürzer die LOW-Pulse, desto weniger muss der Baustein die Frequenz nachregeln.

Achtung:

Der Leitungskreis besteht aus einem Dielektrikum mit einer sehr hohen Dielektrizitätskonstante und darauf aufgedampfter Silberschicht. Bei groben Vorgehen (Anschlussbein abzwicken) reist diese Bedampfung von der Platine und der Leitungskreis ist erst mal hin. Abhilfe hier: Ausbauen, reinigen und mit dünner Kupferfolie sorgfältig umwickeln. Darauf achten, das am Anschlussende überstehende Folie mit einem scharfen Messer sauber abgetrennt wird.

Austausch der Sendefilter

Die SMD-Induktivität L304 vom Ausgang des zweiten Filters nach Masse ist zu entfernen (angeklebt!). Um Schwingneigungen vorzubeugen, ist der Widerstand R316 von 18Ohm auf 51Ohm zu vergrößern. Das Entfernen der alten Sendefilter gestaltet sich dann leicht, wenn man sich eine Kupferdrahtschlaufe biegt, die auf der Lötseite alle Pins eines Filters berührt und diese reichlich verzinnt. Hält ein Helfer den Filter mit einer Zange und erhitzt man die Kupferdrahtschlaufe, so fällt der Filter nach kurzer Zeit fast von selbst heraus.

Die Vorschläge zur Montage der beiden Helix-Filter sind in der Meinung des Autors „Bastellösungen“. Ein besserer Vorschlag ist die Anfertigung zweier Adapterplatinen, die je einen Filter aufnimmt.

Kleinigkeiten:

Um über den gesamten Bereich des 70cm-Bandes konstante maximal Leistung abgeben zu können, sind noch 2 „Durchlass“-Kondensatoren im Sendertreiber (C300, 2p7 und C424, 3p9) um je ein 1pF zu vergrößern (1p2 Huckepack z.B.).

Spannungsmessung:

Da am Port T des NEC die Leitungen LOW_POWER und BAT_OUTZ angeschlossen sind, kann die Spannung bereits grob bestimmt werden (der Port T ist ein Vergleichs-Port, der sich auch zur 8bit AD-Messung per Software nötigen läßt). Ansonsten wäre noch die Modifikation (Schritt 13, Uwe Henning) denkbar.

Konzeptdokumentation SC5BOS

Das SC5BOS ist in dem Flash in einem kleinen Block am oberen Adressende untergebracht. Die für den Einsatz im C5 in Frage kommenden Flashtypen (z. B. 29F002TC, 29F040) besitzen einen 16Kbyte-Sektor (29F002TC) oder einen 64Kbyte-Sektor (29F040) am oberen Adressende. Da Sektoren in einem externen Programmiergerät komplett gegen Überschreiben geschützt werden können, kann ein solch geschütztes SC5BOS-Flash auf fest ins C5 eingelötet werden. Da jedoch wichtige Aufgaben wie die komplette serielle Kommunikation vom SC5-API gekapselt werden (müssen), ist jedoch erst einmal ein gesockelter Flash mit ungeschützten SC5BOS vorzusehen. Bei Bedarf kann hier das gesamte Inhalt neugeschrieben werden, was jedoch kritische Momente erfordert (SC5BOS einen Moment lang nur noch im SRAM)!

Betriebssystemerfordernisse:

- Einfaches Multitasking über den Timer2 (10ms-Systick)
- Application Interface (API) über Softints (ähnlich MSDOS) mit Parameterübergabe in den Registern
- integrierter Bootloader
- Funktionen:
 - Test, ob Benutzerprogramm (SC5PRG) im Flash geladen ist
 - Kopiert bei Bedarf SC5BOS vom Flash ins SRAM und führt es von dort aus
 - PCP: Update SC5BOS über externe serielle Schnittstelle
 - PCP: Neue Software / Update -> Flash paketweise neu beschreiben
 - PCP: Programme in SRAM laden und ausführen
 - PCP: Aktivierung bei laufender Anwendersoftware über die serielle Schnittstelle möglich.
- Unterteilung
 - Bootloader (BL) als Bestandteil des SC5OS
 - Application Interface (API) als Bestandteil des SC5BOS
 - Signalisierung über Soft-Interrupts (Handler) durch das SC5BOS
 - C5-Programm (SC5PRG)
 - Texte und Funktionen, Menü, Ser. IO etc.

Bedienungsanleitung

Das SC5BOS ist vorrangig nicht für eine Benutzerinteraktion gedacht. Ist jedoch kein Programm (SC5PRG) im Flash vorhanden, startet der Bootloader um dem Benutzer einige Funktionen anzubieten. Diese sind beim Einsatz entsprechender PC-Software und einer funktionierenden seriellen Verbindung eigentlich nicht nötig, da diese auch durch den PC ausgelöst werden könnten. Jedoch ist nicht immer ein PC zur Stelle.

Einschalten

Durch den Tastendruck auf **⊙** (On/Off-Taste) oder durch Einschalten der Zündung wird das C5 eingeschaltet. Das Einschalten durch die Zündung erfolgt nur, wenn das Telefon beim Abschalten der Zündung in Betrieb war (*noch nicht implementiert*).

Ausschalten

Durch den Tastendruck auf **⊙** (On/Off-Taste) oder durch Ausschalten der Zündung wird das C5 abgeschaltet. Der Ausschaltvorgang durch die Zündung wird durch SC5BOS verzögert, wobei als Standart 5min eingestellt sind. Die Länge des Gedrückt-Haltens der **⊙** (On/Off-Taste) kann ebenso durch das Programm (zw. 0 und 2,5s) eingestellt werden. In kritischen Phasen kann ein Programm das Abschalten gänzlich unterdrücken. Startet der Bootloader, so schaltet das C5 nach ca. 0,5s Tastendruck ab.

Die Bedienung des Bootloaders

Um auch bei einem vorhandenen SC5PRG den Bootloader zu starten, reicht es, die ON/OFF-Taste beim Einschalten solange weiter gedrückt zu halten, bis ein Piepton ertönt (2,5 Sekunden). Dies ist sehr sinnvoll, falls versehentlich ein fehlerhaftes Programm in den Flash geschrieben wurde.

Es erscheint die Meldung: "**SC5BOS vx.xx**" (x.xx ist die Versionsnummer des SC5BOS). Die Meldung wechselt sich eventuell im Sekundentakt mit einer Status- oder Fehlermeldung ab.

Folgende Statusmeldungen kann die Version 0.19 ausgeben:

| | |
|--------------------------|---|
| Bootload invoked | Durch Gedrückt-Halten der ⊙ -Taste wurde der Start des Bootloaders erzwungen. |
| Flashrom is empty | SC5BOS hat im Flash kein Programm entdeckt |
| Serial malfunc. | Die serielle Schnittstelle funktioniert nicht (Hardwarefehler). Somit kann keine Kommunikation mit dem PC stattfinden! |
| SC5BOS in RAM | Das SC5BOS startete aus dem Speicher des C5 und ist dadurch befähigt, auf den Flash-ROM zuzugreifen (Programmiermodus). |

Durch Drücken einer beliebigen Taste wechselt man in ein Menü. Es erscheint der erste Eintrag („**Show Info**“) im Display.

Man wählt mit den Cursortasten **▲** oder **▼** einen der Menüpunkte und ruft die Funktion mit **▶** auf. In Version 0.16 sind folgende Funktionen sind abrufbar:

| | |
|-------------------------|--|
| Show Info | <p>Zeigt die Seriennummer (obere Zeile) und die Version (untere Zeile) an. Die Version xxyyzzz besteht aus Folgendem:</p> <p>xx = „02“, Jahr-2000 yyy = „455“, Gerätebezeichnung (aus ABB C45-5) zzz = laufende Nummer „001“ bis „zzz“</p> <p><i>Ausgabebsp:</i> 52455001 v0.30</p> |
| Serial0 adjust | Einstellen der Baudrate durch die Tasten ◀ ▶ (57600, 19200, 9600 Baud). Mit ⓘ (Hörer) wird bestätigt – mit ⊙ wird die Baudrate nicht verändert |
| Serial0 test | Überprüft die Funktion der seriellen Schnittstelle mittels internen Loopback (im EP200-IC). Dieser Test findet auch bei jedem Einschalten statt. Meldet entweder „ Serial0 is ok “ oder „ Serial0 malfunc “. Letzteres deutet auf einen Defekt im NEC V25- μ C oder an den Leiterbahnen zu Diesem hin oder die geschaltete 13,2V-Spannung ist abgeschaltet. |
| Memory test | Testet den SRAM des Telefons, ob alle Speicherzellen korrekt arbeiten. Jedoch werden die unteren 2Kbyte nicht getestet, da sich dort wichtige Daten des SC5BOS befinden. Der Test besteht aus Schreiben 2 komplementärer Bitmuster und zählt die fehlerhaften Wörter. Bei 0 Fehlern meldet der Test „ Memory is ok “. Bei 1 oder mehr Fehlern „ Memory malfunc. “ ausgegeben. In späteren Versionen sollen auch die Adressleitungen getestet werden. Befindet sich SC5BOS im RAM, so wird „ not possible “ ausgegeben. |
| EPROM size | Ermittelt die Größe des eingesetzten Flash- oder Eprom-Bausteins. Der Test überprüft, ob sich das SC5BOS 128Kbyte oder 256KByte tiefer spiegelt. Das passiert, wenn kleinere Flashtypen (128/256 KByte) eingesetzt werden. Gibt „ EPROMsiz 128Kbyte “ bis „ EPROMsiz 512Kbyte “ auf dem Display aus. Hinweis: Dieser Test funktioniert dann nicht, wenn die A18 beim Einsatz des 29F040 nicht verbunden wurde. Dafür detektiert diese Funktion auch die Größe eines EProms. mit dieser Funktion wird der Programmiermodus aktiviert. SC5BOS wird in den RAM des C5 kopiert und von dort gestartet. Erst dadurch ist es möglich den Flash zu manipulieren (Löschen, Beschreiben). |
| Execute from RAM | <i>Hinweis:</i> Das RAM ist nach ausführen dieser Funktion schreibgeschützt. Befindet sich SC5BOS schon im RAM, so wird „ not possible “ ausgegeben. |
| SC5BOS: show CRC | Berechnet eine Prüfsumme (CRC nach CCITT) von SC5BOS und stellt diese neben der programmierten Prüfsumme dar. Diese Funktion dient dem Vergleich mit der durch die Update-Software errechneten und an die Stelle 0xFFFFAh programmierten Prüfsumme. Die ersten 4 Stellen in der Anzeige sind das Ergebnis der Berechnung und sollten mit den nächsten 4 Stellen übereinstimmen (programmierte CRC). Hinweis: Die letzten 4 Bytes werden nicht mitgerechnet, da im C5 an reservierter Stelle. <i>Ausgabebsp:</i> FlashCRC EC39EC39 |

0xh – Steuerungsfunktionen

| Funktion | AH | AL | Beschreibung | I-Status |
|---|-----|-----|---|-----------------|
| Programm beenden oder Gerät abschalten | 00h | - | Schaltet das C5 aus (INT3Bh-Aufruf) | v0.14 |
| | 01h | - | Hardware-Reset des C5 (INT3Bh-Aufruf) | v0.14 |
| | 02h | - | Softwarereset normal | v0.14 |
| | 03h | - | Softwarereset, Start des Bootloaders | v0.14 |
| | 04h | - | Startet Bootloader ohne Reset | v0.14 |
| Interruptvector / Behandlungsroutine setzen | 05h | Vec | IN: AL=Interruptvector, DS:[SI]=INT-Routine, bei Behandlungsroutine ES = benutztes Datensegment | v0.15 Ä0.17b |
| Stack in den SRAM verlagern | 06h | - | Die Funktion kopiert einmalig den Stack an das obere SRAM-Ende (für Programme, die mehr als 64Words Stack benötigen). | v0.14 |
| Stackwarnung, Fehler | 07h | 00h | Stackwarnung ausschalten | v0.14 |
| | | 01h | Stackwarnung einschalten (default) | |
| Gerät abschalten steuern | 08h | 00h | Ausschalten erlaubt (default) | v0.14 |
| | | 01h | Ausschalten verboten (Taste keine Wirkung) | |
| Unterbrechung durch BL | 09h | 00h | Unterbrechungen zulassen (default) | v0.14 |
| | | 01h | Unterbrechungen verbieten (für kritischen Code) | |
| AUS-Taste Drucklänge | 0Ah | Val | IN: AL = Länge (x10ms), die der Benutzer die Aus-Taste gedrückt halten muss. | v0.14 |
| Behandlungsroutine entfernen | 0Bh | Vec | IN: AL = Vektor der Behandlungsroutine | v0.17b |
| Zündungs-Timer einstellen | 0Ch | Tmr | IN: AL = Anz. Minuten, die das C5 noch an bleibt | v0.30 |
| Warten (passives Warten) | 0Eh | - | IN: CX: Wartezeit in (CX*10ms) | |
| Warten (aktive Warteschleife) | 0Fh | - | IN: CX: Wartezeit (CX=256 → 30ms Warten) | v0.14 |

1xh – BFBus – Zugriff und Abfrage

| Funktion | AH | AL | Beschreibung | I-Status |
|----------------------------|-----|-----|--|----------|
| BF-Bussteuerung | 10h | 00h | BFBus-Behandlung abschalten (→ INT 3Ah wird nicht ausgeführt) | v0.14 |
| | | 01h | BFBus-Behandlung einschalten (→ INT 3Ah ist aktiviert, default) | |
| Sende Paket zum Hörer | 11h | - | IN: Paket in DS:[SI] mit der Länge CL, | v0.15 |
| QuittungsTon | 12h | - | Gibt einen leisen einzelnen Ton aus | v0.30 |
| FehlerTon | 13h | - | Gibt einen leisen Doppelton aus | v0.30 |
| Warte auf Hörer-Taste | 14h | - | OUT: AL = Tastencode | v0.15 |
| Letzte Hörertaste abfragen | 15h | - | OUT: AL = Tastencode, AH = 0h, wenn Taste neu | v0.15 |
| Setze Wiederholrate | 16h | Val | IN: AL = Tastenwiederholrate in 10ms | v0.15 |
| Setze Startverzögerung | 17h | Val | IN: AL = Verzögerungszeit erste Wiederh. (10ms) | v0.15 |

2xh – Serielle I/O-Funktionen (Ser0 und I²C-Bus)

| Funktion | AH | AL | Beschreibung | I-Status |
|--|-----|------|--|----------|
| Serial0: Sende-/Empfangsmodus | 20h | | | |
| Serial0: Baudraten Ser0 | 21h | Val | IN: AL=Aufzählung Baudrate | |
| Serial0: Sende Zeichen | 22h | Val | IN: Zeichen, das direkt auf TxD0 geschickt wird | |
| Serial0: Sende Paket | 23h | - | IN: DS:[SI], CX = Paket, analog zu 22h | |
| Serial0: Send-PCP_Short | 24h | Port | IN: AL=Port, DL=Zeichen | v0.14 |
| Serial0: Send-PCP-Long | 25h | Port | IN: AL=Port, DS:[SI], CX = Paket | v0.15 |
| Serial0: Sende zur Chipkarte | 26h | - | IN: DS:[SI], CX = Paket | |
| Serial0: Lese von Chipkarte | 27h | - | IN: ES:[DI] = Buffer, CX = ReadLength (max) | |
| I ² C: Abbruch ausgeben | 28h | Char | IN: AL = Zeichen, das gesendet wird | v0.15c |
| I ² C: Daten ausgeben | 29h | Adr | IN: AL = I ² C-Geräteadresse, DS:[SI], CX = Paket | v0.15c |
| I ² C: Daten auslesen | 2Ah | Adr | IN: AL = I ² C-Geräteadresse, ES:[DI] = RXBuffer, CX = Anzahl auszulesender Bytes | v0.15c |
| I ² C: Kombi: Ausgabe, Einlesen | 2Bh | Adr | IN: AL = I ² C – Geräteadresse (Schreibadr.) DS:[SI], CL = Paket für Aussendung, ES:[DI] = RXBuffer, CH = Anzahl auszulesender Bytes | v0.15c |

| | | | | |
|-------------------------|-----|-----|--|-------|
| EEProm: Daten schreiben | 2Ch | Adr | <i>IN</i> : DS:[SI] = Zeiger auf die Daten, CL = Anzahl der Daten, AL = Word-Adresse der ersten Speicherzelle. | v0.30 |
| EEProm: Daten lesen | 2Dh | Adr | <i>IN</i> : ES:[DI] = Buffer für die Daten, CL = Anzahl, AL = Word-Adresse der ersten Speicherzelle | v0.30 |

Hinweis: Die API-Funktion I²C Acknowledge Polling wurde ersatzlos entfernt, da sie nicht benötigt wird!

3xh – Informations- Konvertierungsfunktionen

| Funktion | AH | AL | Beschreibung | I-Status |
|---|-----|-----|--|----------|
| Informationen über SC5BOS | 30h | - | <i>OUT</i> : Version in AX (Hi.LOW) | v0.14 |
| | 31h | - | <i>OUT</i> : Einschaltdauer in AX, BH, BL (H, M, S) | v0.14 |
| | 32h | - | Fehlerzähler des BFBusses (Frame, Overrun → AX) | v0.14 |
| | 33h | - | Fehlerzähler der ext. Schnittstelle (analog 02h) | v0.14 |
| Flashgröße bestimmen | 34h | - | <i>OUT</i> : AX = Flashgröße in Kbyte | v0.15 |
| Flashhersteller und Device-ID | 35h | - | <i>OUT</i> : AH = Hersteller ID, AL = Device ID | v0.15 |
| Interruptvector lesen (zum Sichern z.B.) | 36h | Nr | <i>IN</i> : AL = Interruptvektor <i>OUT</i> : DX:AX = Adresse der Interruptbehandlungsfunktion (Seg:Ofs) | v0.15e |
| Zahl → ASCII (DS:[DI]) | 3Dh | Val | <i>IN</i> : AL = Zahl, Umwandlung in 2.-Hexdarstellung <i>OUT</i> : Ascii → ES:[DI], DI = DI + 0 | v0.14 |
| | 3Eh | - | <i>IN</i> : DX = Zahl, Umwandlung in 4.-Hexdarstellung <i>OUT</i> : Ascii → ES:[DI], DI = DI + 4 | |
| | 3Fh | Val | <i>IN</i> : AL = Zahl, Umwandlung in 8.-Binärdarstellung <i>OUT</i> : Ascii → ES:[DI], DI = DI + 8 | |

4xh – DSP-Funktionen

| Funktion | AH | AL | Beschreibung | I-Status |
|------------------------|-----|-----|--|----------|
| Reset DSP | 40h | - | Führt einen Rest per Reset-Signal am DSP durch | 0.20 |
| Übertrage DSP-Programm | 41h | - | <i>IN</i> : DS:[SI], CX = Datenblock, der das DSP-Programm enthält. | 0.20 |
| Sende zum DSP | 42h | - | <i>IN</i> : DX:CH = 24bit-Wort das der DSP erhalten soll. | 0.20 |
| DSP-Kommando | 43h | Cmd | <i>IN</i> : AL = Kommando (0 bis 11), welches das DSP-Programm ausführen soll. DX:CH = 24bit-Wort das der DSP vorher erhält. | 0.20 |

Informationen zur C5-Hardware

In diesem Abschnitt ist in Stichpunkten zusammengetragen, was es so übers C5 als Entwickler wissen sollte. Leider lassen sich nicht besonders viele Informationen zur Technik des C-Netzes auffinden. Hier ist eine kleine aber gute Zusammenfassung:

<http://www.e-online.de/sites/kom/0408091.htm>

Aufgrund der kombinierten Daten- und Sprachübertragung gibt es auch in der C5-Hardware ein paar Besonderheiten (näheres siehe Codec-Signalweg).

Speicheraufteilung

Der Adressraum des NEC-Microcontrollers beträgt 1024kByte. Davon wird im C5 nur die unteren 64Kbyte und die obersten 256Kbyte (Originalsoftware, nach dem Umbau 512Kbyte) benutzt. Das C5 besitzt 32Kbyte statischen RAM an der Adresse 0000h bis 3FFFh. Hier befinden sich die Interruptvektortabelle des x86-kompatiblen NEC's, (1Kbyte) sowie beim Betrieb von SC5BOS ein weiteres KByte Daten. Die weiteren 30Kbyte sind vollständig durch Programme nutzbar, wenn der Stack im Internen RAM des NEC belassen wird. Ansonsten sollte an der oberen RAM-Grenze der Stack-Bereich nicht beschrieben werden. In dem Bereich von 4000h bis 7FFFh befinden sich verschiedene Speichereinblendungen:

- **C000h - CFFFh**: Host-Interface des DSP (Register an C101h, C202h usw.)
- **D000h - D001h**: Paralleler DAC für AFC und TXLEV_REF (Sendeleistung)
- **E000h - E01Fh**: Speichereinblendung auf Register des EP200 – welche genau was bewirken ist noch Forschungsarbeit. Der Autor bittet Jeden, der neue Infos zum EP200 hat, ihm Diese mitzuteilen.
- **FE00h - FEFFFh**: Interner RAM des NEC- μ C (der Bereich wird hier sowohl in der Originalsoftware, als auch bei SC5BOS eingeblendet, ist jedoch in 4K-Schritten wählbar)
- **FF00h - FFFFh**: Special-Function-Register des NEC- μ C (Einblendung mit int. RAM gekoppelt)

Danach folgt erst mal eine Weile nix. Ab der Adresse 80000h ist dann der Flashrom eingeblendet. Glücklicherweise generiert der EP200 sowohl /CS als auch /WE Signale für den Bereich von 80000h bis BFFFFh. Im C5 wäre eigentlich keine Veranlassung für ein solches Verhalten, da der EPROM erst ab Adresse C0000h benutzt wird (von 80000h bin BFFFFh spiegelt er sich).

Der EP200

Leider ist der GAL mit der Bezeichnung EP200 immer noch ein großes Rätsel für sich. Er ist gehört von seiner Funktion weder zur reinen „Glue-Logic“ noch zur reinen Signalverarbeitung. Dieser Baustein vereinigt verschiedenste Funktionen in sich. Hier eine kurze Aufzählung:

- aus Quarzoszillator werden verschiedene Takt- und Clocksignale erzeugt: μ C, Combo-Clock-Signale etc. (bis auf μ C sind alle Signale ein-/abschaltbar)
- I²C realisieren.
- Bereitstellen von zusätzlichen I/O-Ports.
- „multiplexen“ der zweiten seriellen Schnittstelle.
- C-Netz-Datenpakete (zwischen den komprimierten Audiodaten) herausfiltern und verarbeiten (hier wurde der DMA des NEC benutzt).
- ... (bestimmt noch mehr Features)

Laut Aussage eines nicht näher bekannten Insiders:

„...habe mit einem Entwickler vom FP100 ASIC (EP200 , Entwicklungsprojekt 2000) gesprochen. In diesem Baustein wird nur die Signalverarbeitung vom C-Netz gemacht. Als Komprimierung, Dekomprimierung, ausfiltern der Daten usw. aber keine NF Bearbeitung. Da von den IC nur 2 Schüsse gemacht wurden ist er Fehlerhaft und stürzt öfters ab. Diese Probleme wurden Softwaremäßig kompensiert.“ (Zitat Ende)

Clockgenerierung

folgt noch...

EP200-Ports

| Mapped Adresse | Funktion | Beschreibung |
|----------------|--------------------------------|--|
| E000h | EP200-Port | Bit6 = BFBus, Bit4 = NF-Enable (Lautsprecher) |
| E001h | Combo-Steuerung | |
| E002h | PC Data | s.u. |
| E003h | PC Control | s.u. |
| E004h | Steuerung / ser. Schnittstelle | evtl. sind hier Steuerbits für Kartenzugriffe |
| E010h | Abtastintervall Combo1 | 174 (0Aeh) ergeben die (gewünschten) 104µs für 9600Sps |
| E011h | Steuerung FSR/FSK ... (?) | 085h schaltet den FSx-Takt vom Combo1 ein, welche Bits was machen -> ? |
| E01Ah | Umschaltung extern/Karte | Bit5 schalten C5 jedenfalls ab – Fehler? mit 0Ch kann ser. Schnittstelle auf Extern geschaltet werden. |

folgt noch...

Übersicht des Speichermapping

folgt noch...

Portfunktionen des NEC und des EP200

folgt noch...

Serielle Schnittstellen

Der NEC-Microcontroller verfügt über 2 unabhängige fullduplex UARTs die synchron oder asynchron betrieben werden können. An dem ersten Port (Ser0) ist im C5 der BFBus angeschlossen. Der zweite Port (Ser1) ist dagegen nur mit dem EP200 verbunden. Außerdem ist die CCData-Leitung des Kartenlesers mit der RXD-Leitung verbunden. Über Controlregister im EP200 kann die Benutzung von Ser1 gesteuert werden. Dabei kann bestimmt werden, ob die Schnittstellen an die V24-RXD, TXD Leitungen am SubD-Anschluß oder intern am Kartenleser betrieben werden. Ein Loopback durch den EP200 ist möglich (wird für die Kartenkommunikation auch benötigt) und wird zum Test der Schnittstelle verwendet (SC5BOS).

BFBus

Der BFBus ist ein serieller Eindraht-Bus, der den Telefonhörer und die Optionsbaugruppen mit dem C5-Grundgerät verbindet. Die Kommunikation erfolgt dabei paketbasiert, halb-duplex und bidirectional mit 19200baud 8N2 und TTL-Pegel (der inaktive Zustand ist High). Physikalisch ist der BFBus fest mit dem Seriellen Port 0 des NEC verbunden. Zur Kommunikation benötigt SC5BOS alle Ser0-Interrupt-Vektoren und zusätzlich den Timer0 im SingleShot-Mode.

Die Kollisionsvermeidung (da mehrere sendefähige Baugruppen) erfolgt durch einen einfachen Signalisierungsmechanismus: Die sendewillige Station signalisiert Ihren bevorstehende Bus-Benutzung mit dem Umschalten der BFBus-Leitung auf den aktiven Zustand (Low) für ca. 3-5ms. Alle anderen Baugruppen erkennen dies und halten die BFBus-Leitung ebenfalls auf Low. Die sendende Baugruppe (z.B. Höher) kann nach Umschalten auf den inaktiven Zustand erkennen, ob die Leitung weiterhin auf Low bleibt. Nachdem auch die anderen Baugruppen die Leitung wieder inaktiv geschaltet haben, beginnt die sendewillige Baugruppe (nach ca. 3-13ms) mit der Aussendung des (ersten) Paketes. Folgen noch weitere Pakete, so können diese in einem Zeitfenster auch ohne Bus-Request hinterhergesendet werden. Nach ca. 5ms erfolgt bei Bedarf vom Empfänger ein Antwort(ACK)-Paket. Das Zeitfenster für weitere Pakete beträgt ca. 10ms.

Jedes Paket ist nach folgenden Schema aufgebaut:

| | | | | |
|-----------------|------------------|-------------|---------------------------|---------------|
| Absenderadresse | Empfängeradresse | Länge Paket | Nutzdaten (Länge-4 Bytes) | XOR Prüfsumme |
|-----------------|------------------|-------------|---------------------------|---------------|

Folgende Nutzdaten kann der Hörer zum Grundgerät schicken:

- **2Eh, 01h, Key** (Key enthält den Tastencode, beim Loslassen der Taste +80h)
- **2Fh, S01, S02, S03, S04** (S0x = Antwort auf 0Fh, enthält u.a. Status den Reed-Kontakts)

Die Nutzdaten, die das Grundgerät zum Hörer schicken kann, sind wie folgt aufgebaut:

Das erste Datenbyte ist ein Kommando. Danach folgen mit variabler Länge die Parameter zu einem Kommando.

Folgende Kommandos werden vom Hörer unterstützt:

- **02h** - Textausgabe, es werden die folgenden 16 ASCII-Zeichen im Display dargestellt (s. Zeichentab.)
- **06h** - Symbolanzeige, mit 7 Steuer-Bytes werden die Symbole im Display ein, aus oder blinkend geschaltet
- **07h** - Einstellung, mit einem Steuerbyte werden Licht, Hörkapsel aus/ein, Mikrofon-Auswahl und die Hörerlautstärke gesteuert
- **08h** - Töne/Klingeln ausgeben, durch 6 Bytes wird eine Tonsequenz über den „Piepser“ ausgegeben.
- **0Fh** - Version/Statusabfrage, keine näheren Infos ☹
- **10h** - Bestätigung (ACK) senden, keine Parameter

Typ 06h: Display-Symbole

| Bsp | Beschreibung | Details (Bits) |
|-----|--------------------------|---|
| 00h | Absender | |
| 1Ah | Empfänger (Hörer) | |
| 0Ch | Länge 12 Bytes | |
| 06h | Typ 6 Paket | |
| 00h | Display-Symbole | 1=Schlüssel, 2=Pfeil heraus, 4=Pfeil hinein, 8=Lautsprecher, 16=Hörer, 32=Kreis |
| 01h | Display-Balken | 1=S-Rahmen, 2=S1, 4=S2, 8=S3, 16=S4, 32=S5, 64=TAE-Rahmen, 128=TAE-Symbol |
| 00h | ??? | |
| 20h | Display-Symbole blinkend | 1=Schlüssel, 2=Pfeil heraus, 4=Pfeil hinein, 8=Lautsprecher, 16=Hörer, 32=Kreis, Rest=??? |
| 01h | Display-Balken blinkend | 1=S-Rahmen, 2=S1, 4=S2, 8=S3, 16=S4, 32=S5, 64=TAE-Rahmen, 128=TAE-Symbol |
| 00h | ??? | |
| 00h | ??? | 1=komplettes Display blinkt |
| 30h | Prüfsumme | |

Typ 07h: Einstellungen

| Bsp | Beschreibung | Details (Bits) |
|-----|------------------------|---|
| 00h | Absender | |
| 1Ah | Empfänger (Hörer) | |
| 06h | Länge 6 Bytes | |
| 07h | Typ 7 Paket | |
| 71h | Einstellungen am Hörer | 1=Licht ein, 2=Hörikapsel ein, 4=Mikro ein, 8=Auswahl Freisprech-Mikro, 16, 32, 64=Hörerlautstärke (8 Stufen) |
| 6Ah | Prüfsumme | |

Typ 08h: Tonerzeugung

| Bsp | Beschreibung | Details (Bits) |
|-----|-------------------|---|
| 00h | Absender | |
| 1Ah | Empfänger (Hörer) | |
| 0Bh | Länge 11 Bytes | |
| 08h | Typ 8 Paket | |
| C1h | Tonhöhe 1 | Bei Tonhöhe2=0x4F ist Periodendauer des Tons $T=(256-Tonhöhe1)*8,138\mu s$ |
| 4Fh | Tonhöhe 2 | ??? |
| 0Fh | Tonlänge | |

| | | |
|-----|-------------|--|
| 01h | Pausenlänge | |
| 01h | Tonanzahl | |
| 13h | Steuerbits | 1=Ton ein, 2=Single shot, 4=???, 8=Portamento, 16=Tastenklick, 32=leiser Beep |
| 8Bh | Prüfsumme | |

Typ 2Eh: Tastencode

| Bsp | Beschreibung | Details (Bits) |
|-----|---------------|---|
| 1Ah | Absender BT | |
| 00h | Empfänger GG | |
| 07h | Länge 7 Bytes | |
| 2Eh | Typ 2Eh Paket | |
| 01h | ??? | |
| 30h | Tastencode | Beim Loslassen einer Taste wird der Tastencode plus 80h gesendet. |
| 02h | Prüfsumme | |

Die (Tasten-)Nachricht muss vom Grundgerät mit dem ACK-Paket (Typ 10h) bestätigt werden, ansonsten wird sie automatisch nach einigen Millisekunden wiederholt.

Kartenleser

Der Kartenleser ist mit CC_Clock und CC_Data an den EP200 angeschlossen. Wahrscheinlich erzeugt der EP200 analog zum I²C-Bus das Clocksignal automatisch, wenn Daten gesendet oder ausgelesen werden. Die Kommunikation mit der Karte erfolgt dabei in der Originalsoftware mit 9600baud über Ser1. Daher ist auch CC_Data physikalisch mit RXD1 verbunden.

Weitere Nachforschungen werden später noch vorgenommen. Ziel: Einfache Benutzung einer Standard-Speicherkarte (E²Prom) als transportable und am PC editierbare Frequenzspeicherablage.

externe Schnittstelle

Die externen Leitungen V24-RXD und V24-TXD können durch den EP200 mit Ser1 verbunden werden. Dadurch ist es möglich ohne Umbau bei geschlossenem Gerät Daten zu transportieren. In der Originalsoftware wurden im Servicebetrieb Meldungen mit 19200baud ausgegeben. SC5BOS nutzt diese Schnittstelle jedoch viel intensiver und stellt eine PCP-Kommunikation mit bis zu 57600baud bereit. Eine höhere Standardbaudrate (115k2) ist aufgrund der Taktfrequenz von 4.032MHz des NEC nicht machbar, da es dafür keine ganzen Teiler mehr gibt. Die V24-Leitungen haben jedoch einen entschiedenen Nachteil: Sie führen nur ein TTL-Signal (5V – 0V). Eine PC-RS232-Schnittstelle benötigt jedoch ein Signal mit rund ±10V. Daher ist ein einfacher Pegelwandler wie z.B. ein Max232A oder Max202 (viele Typen) nötig. Die Flusskontrollsignale V24-CTS, V24-RTS werden nicht benutzt, um keine zusätzliche Konkurrenz für den I²C-Bus (Signale gleich I²C-Clock und -Data) darzustellen.

I²C-Bus und Slave-IC's

Der I²C-Bus wird komplett durch den EP200 gesteuert. Er stellt 2 byte-große Register die an den Adressen 0E002h und 0E003h eingeblendet sind (MemoryMapped-I/O).

An 0E002h werden die Datenbytes des I²C-Busses geschrieben (oder ausgelesen), während an 0E003h Controlbits die Übertragung steuern und einen Status der Übertragung beinhalten.

Die I²C-Bus-Benutzung funktioniert jedoch erst, nachdem der EP200 initialisiert worden ist. Welches Controlbit dabei die Clocksignalgenerierung für die Ausgabe einschaltet liegt leider noch im Dunkeln. SC5BOS initialisiert den EP200 nach dem Reset (so wie Originalsoftware) so, das I²C funktioniert. Ist I²C erst mal aktiv, wird mit jedem Schreibzugriff auf das Datenregister das geschriebene Byte (+ 9tes Bit) auf den Bus geschifft. Das 9te Byte stellt bei I²C-Übertragungen das ACK-Bit dar. Weitere Infos zu I²C bitte aus Datenblättern o. ä. herausnehmen.

I²C-Controlregister

Damit eine korrekte Übertragung getätigt werden kann, muss der EP200 verschiedene Conditions auf dem I²C-Bus erzeugen können. Es handelt sich dabei um die im I²C-Protokoll spezifizierten Start- und Stop-Conditions. Auch fehlt im Datenregister noch das 9te Bit (Acknowledge). Daher existieren im Controlregister folgende Bits:

Bit0: "1" erzeuge Start-Condition (SDA H->L, bei SCL = H).

- "0" keine Start-Condition (SDA, SCL werden beim Start einen Clock-Zyklus auf LOW gesetzt).
- Bit1: "1" erzeuge Stop-Condition (SDA L->H, bei SCL = H).
 "0" keine Stop-Condition (SDA, SCL werden nach dem 9ten Bit auf LOW gesetzt).
- Bit2: "1" ACK = HIGH : Nicht bestätigen oder Slave bestätigt.
 "0" ACK = LOW : Slave bestätigen (beim Daten-Einlesen)

Damit die Programmroutine weiß, wann ein ins Datenregister geschriebenes Byte vollständig herausgeschifft ist, existiert im Controlregister ein Busy-Flag (Bit 3), das solange HIGH bleibt, wie der EP200 das Datum herausshifft. Das Bit 2 (Acknowledge) scheint beim Auslesen den Zustande der Datenleitung SDA zu beinhalten. Damit kann detektiert werden, ob ein Slave das Datum bestätigt hat.

Folgende Bitkombinationen für das Controlregister sind üblich:

TX:

- 05 101 Erstes Zeichen an Slave, Slave bestätigt mit ACK->L
 04 100 ein weiteres Zeichen wird gesendet,
 06 110 Das letzte Zeichen wird gesendet, Slave bestätigt mit ACK->L oder I²C-Reset

RX:

- 00 000 Ein Zeichen wird empfangen und vom Master bestätigt
 06 110 Das letzte Zeichen wird vom Master empfangen und NICHT bestätigt

I²C-Timing

Die Pulslänge eines Clock-Signals auf der SCL-Leitung liegt bei 15,4µs (High =7,5µs, Werte gemessen). Die Zeit von der ersten steigenden Flanke bis zu letzten entspricht rund 123,2µs. Eine Ausgabe eines Bytes mit Start- und Stop-Condition beträgt rund 162,4µs. Das Timing ist somit für normale I²C-IC's völlig unkritisch und liegt bei ca. der halben Frequenz, die als obere Grenze bei den in C5 eingesetzten IC's spezifiziert ist.

I²C-Bausteine im C5

Im C5 sind 3 Bausteine mit über den I²C-Bus an den EP200 angeschlossen:

- die PLL („TBB200“ od. „Zarlink NJ88C33“)
- der kombinierte AD/DA-Wandler PCF8591 von Philips
- ein E²Prom 24C02A mit 256Byte Speicher

Mit der PLL und dem AD/DA-Wandler klappt die Kommunikation hervorragend. Mit dem E²Prom ist noch nicht so viel anzufangen, hier muss noch einmal das s.g. „Acknowledge-Polling“ überprüft werden – erste Test waren lieferten recht merkwürdige Ergebnisse (nur 4 statt max. 16Byte beim PageWrite geschafft etc.).

PLL

Die PLL arbeitet im double Modulus Betrieb mit einem Vorteiler-IC mit Frequenz/64 und Frequenz/65. Die Referenzfrequenzquelle liegt im Empfänger des C5, liefert 14,85MHz und ist über den PCF8591 justierbar. Um ein Raster von 12,5kHz zu bekommen, wird der Referenzteiler (RCounter) auf 1188 gesetzt. Der Gesamtteiler des Systems muss jetzt so gesetzt werden, das die gewünschte Oszillatorfrequenz / Gesamtteiler = 12,5kHz ergibt. Dieser Teiler (Counter) muss nun so „zerlegt“ werden, das A- und N-Counter entstehen. Es gibt mehrere Möglichkeiten dafür. die einfachste: Der N-Counter = Counter DIV 64, falls Counter MOD 64 > 0, sonst N-Counter = Counter DIV 64 – 1. Der A-Counter = Counter MOD 64, falls Counter MOD 64 > 0, sonst A-Counter = 64.

Natürlich muss die PLL auch initialisiert werden (Status, im C5 = 7Eh). Einfach Datenblatt einsehen.

AD/DA-Wandler

Der Wandler ist weiter nichts Besonderes: Ein Lesezugriff befördert ein Byte aus dem internen Datenregister des Wandlers nach Draußen. Mit Schreibzugriffe können der Wandlungsmodus, die Kanalverschaltungen, der als nächstes zu wandelnde Kanal oder der Ausgabewert verändert werden. Einfach Datenblatt einsehen.

EEProm 24C02A

Das EEPROM im C5 erhält eine Sonderbehandlung. Während die Schreiboperation mit max. 2 Bytes wie im Datenblatt beschrieben arbeitet, funktioniert das Sequenzielle Lesen im C5 offensichtlich nicht. Daher wird für jedes Byte eine Random-Read-Operation ausgeführt. Dies ist auch der Grund, warum extra API-Funktionen zum Zugriff bereitstehen. Der Programmierer braucht sich keine Gedanken um die o.g. Unzulänglichkeiten zu machen.

Motorola DSP 56001 / 56002

Der Motorola DSP 56001 ist ein inzwischen veralteter „klassischer“ digitaler Signalprozessor mit 24bit-Festkomma-Arithmetik. Er verfügt über 512Wörter Programmspeicher und 2 mal 256Wörter Datenspeicher (X, Y). Auch hat er eine 256Wörter große Sinustabelle mit an Board. (Jedes Word hat natürlich 24Bit.)

Externen Speicher oder ROM wurde im C5 nicht angeschlossen. Programme und Daten bekommt der DSP über das Host-Interface (HI), einem parallelem 8bit breiten High-Speed-Port, der mit dem EP200 verbunden ist. Der NEC kann den HI über eine Speichereinblendung erreichen und erhält zusätzlich noch ein INT-Signal vom DSP. Seine eigentlichen Daten bekommt der DSP über seine beiden SPI-Schnittstellen an denen je ein Audio-PCM-Codec angeschlossen ist. Die Taktung des SPI-Busses und der Codecs übernimmt dabei der EP200, im DSP sind nur die beiden RX-Ints zu behandeln.

Host-Interface

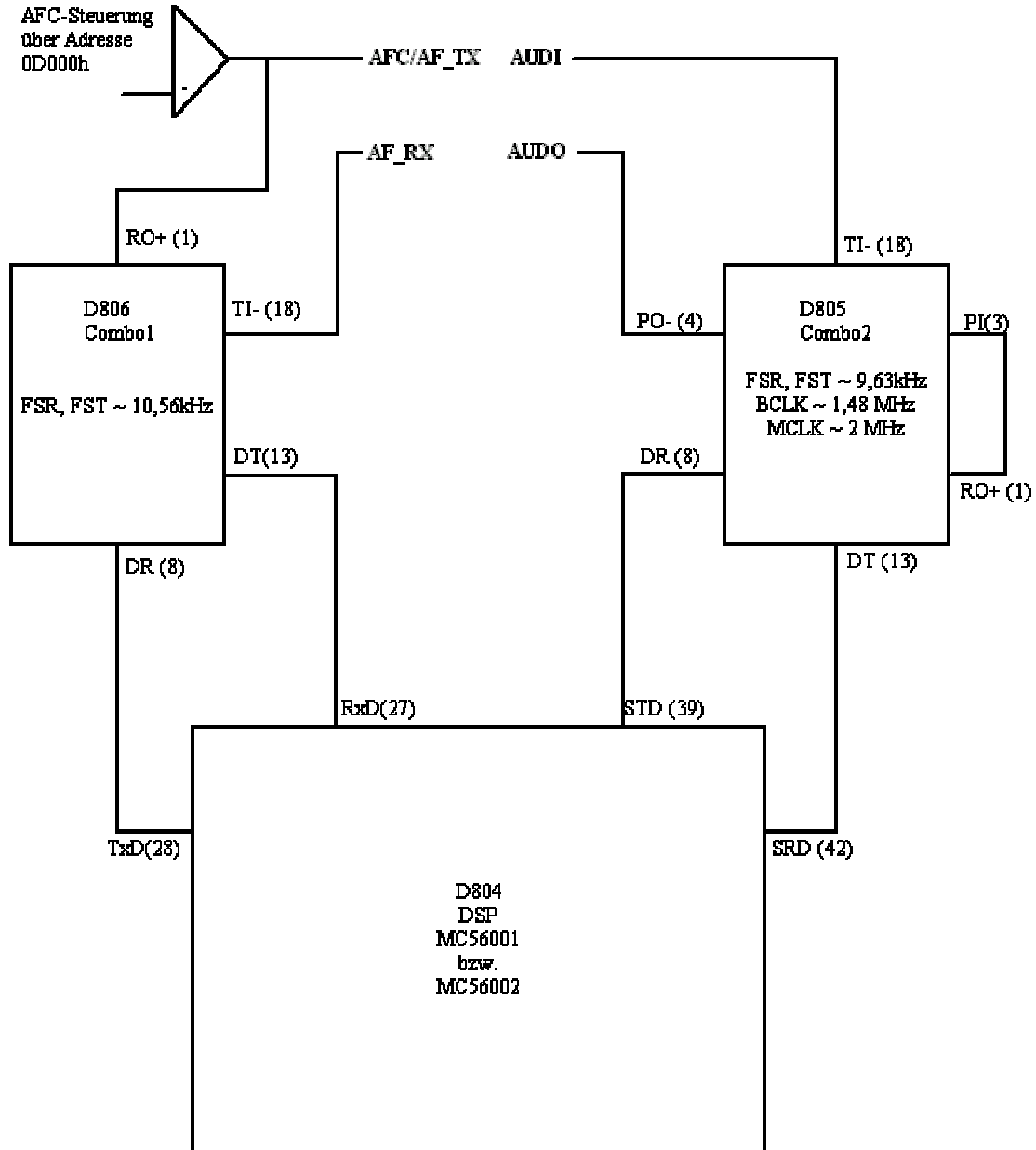
in Arbeit.

SCI/SSI-Ports

in Arbeit.

Der Codec-Signalweg

Aufgrund der kombinierten analogen Sprach- und digitalen Datenübertragung ergeben sich Unterschiede zwischen dem AF-Codec-Teil (an Modulator / Empfänger angeschlossen) und dem NF-Codec-Teil. Die beiden an sich gleichen MC145480-PCM-Codes werden mit unterschiedlichen Samplingraten betrieben (siehe Bild). Damit wurde die Sprache einfach durch Pufferung im DSP komprimiert. Weiterhin hat der DSP die Sprachverschleierung (Invertierung) vorgenommen und sicher auch DTMF-Töne erzeugt.



Zeichensatz des Standard-Hörers

Der Hörer des C5 verfügt nicht über einen kompletten ASCII- oder erweiterten ASCII-Zeichensatz. Es können daher nur folgende Zeichen (20h bis 91h) dargestellt werden:

| | 2xh | 3xh | 4xh | 5xh | 6xh | 7xh | 8xh | 9xh |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| x0h | | 0 | @ | P | ` | p | Ä | ↵ |
| x1h | ! | 1 | A | Q | a | q | Ö | .. |
| x2h | " | 2 | B | R | b | r | Ü | |
| x3h | # | 3 | C | S | c | s | ä | |
| x4h | \$ | 4 | D | T | d | t | ö | |
| x5h | % | 5 | E | U | e | u | ü | |
| x5h | & | 6 | F | V | f | v | ß | |
| x7h | ' | 7 | G | W | g | w | → | |
| x8h | (| 8 | H | X | h | x | ← | |
| x9h |) | 9 | I | Y | i | y | ↑ | |
| xAh | * | : | J | Z | j | z | ↓ | |
| xBh | + | ; | K | [| k | { | ↻ | |
| xCh | , | < | L | \ | l | | ▶ | |
| xDh | - | = | M |] | m | } | ◀ | |
| xEh | . | > | N | ^ | n | ~ | ▲ | |
| xFh | / | ? | O | _ | o | ■ | ▼ | |

In wieweit eigene Zeichendefinitionen in den Hörer geladen werden können steht nicht fest. Wahrscheinlich geht so etwas nicht. Zeichen mit Nummern unter 20h (0 bis 31) werden vom Hörer ignoriert – der Rest des Textes rückt einfach nach. Alle Zeichen mit Nummer größer 91h sind ebenfalls ‘. .’ (horizontaler Doppelpunkt). Das Zeichen 8Bh stellt das Netzstecker-Symbol dar. das Zeichen 90h ist ein solcher Return, der Pfeil beginnt jedoch von unten (an x-Achse gespiegelt).

Tastaturcode des Standard-Hörers

| Taste | Code | ASCII | Taste | Code | ASCII |
|-------|------|-------|---------|------|-------|
| 0 | 30h | '0' | * | 2Ah | '*' |
| 1 | 31h | '1' | # | 23h | '#' |
| 2 | 32h | '2' | C | 3Ah | ',' |
| 3 | 33h | '3' | . | 3Bh | ',' |
| 4 | 34h | '4' | .. | 3Ch | '<' |
| 5 | 35h | '5' | ◀ | 4Bh | 'K' |
| 6 | 36h | '6' | ▶ | 21h | '!' |
| 7 | 37h | '7' | ▲ | 3Fh | '?' |
| 8 | 38h | '8' | ▼ | 3Dh | '=' |
| 9 | 39h | '9' | { | 3Eh | '>' |
| | | | Kontakt | 48h | 'H' |

SC5BOS erzeugt für die ●-Taste (On/Off) zusätzlich noch den Tastencode 27h beim Loslassen dieser Taste (natürlich nur wenn sie nicht zu lange gedrückt wurde und das Telefon abschaltet). Jedoch gibt es kein BFBUS-Paket dafür, der Tastencode wird nur dem Tastaturhandler übergeben.

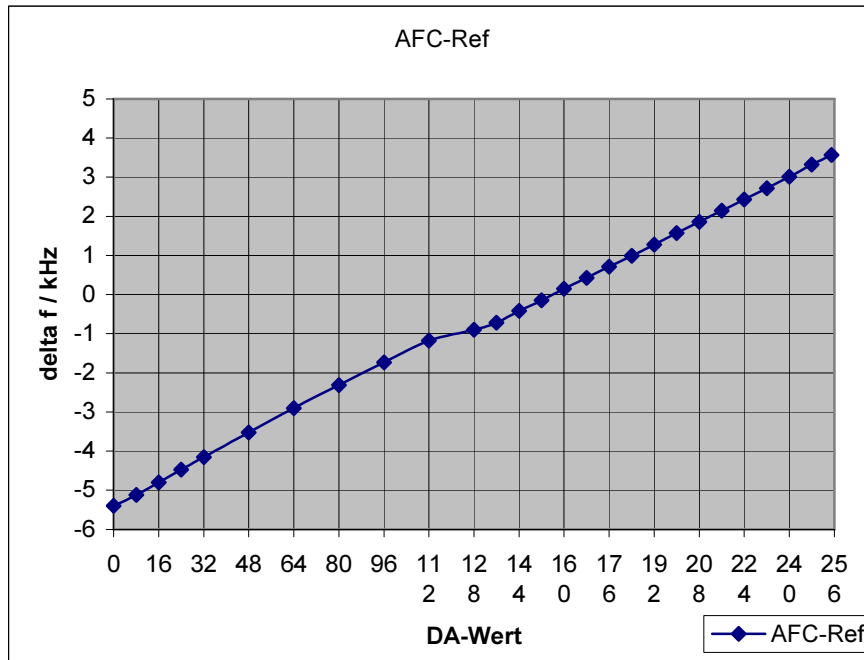
Der Empfänger

Der Empfänger besitzt leider keine so ausgezeichnete Empfindlichkeit wie moderne 70cm-Geräte. Direkt (am Eingang, ohne Duplexer oder Koppelschaltung) gemessen liegt sie bei -114,5dbm (20db SINAD). Die Werksangabe von Siemens/ABB gibt -113dbm an.

Die AFC Steuerung

Die PLL bekommt ihre Referenzfrequenz aus dem Empfänger. Hier sitzt ein silberner Block, der eine Frequenz von 14,85 MHz produziert. Dieser ist über eine Verstellerschraube abgleichbar. Neben dieser Möglichkeit das Gerät abzugleichen, gibt es noch 2 per Software veränderliche Größen, als AFC_Ref und AFC bezeichnet. Der

erste Wert wird über die I²C-AD/DA-Wandler eingestellt und stimmt die Referenzquelle (14,85MHz) fein ab. Der Regelbereich liegt dabei bei etwa $\pm 4,5$ kHz der Oszillatorfrequenz. Pro Digit wird die Frequenz also um rund 35Hz verändert (pro 16 Schritte ~ 560 Hz).



Die andere Einstell-Möglichkeit ist AFC per parallelem DA-Wandler D807. AFC gleicht die evtl. auftretenden Versatz der Sendefrequenz zur Empfangsfrequenz aus, indem dem NF-Signal (AF_TX) ein Gleichspannungsoffset überlagert wird. Der Einstellbereich entspricht ca. $\pm 6,6$ kHz der Sendefrequenz. Pro Digit wird die Sendefrequenz um ca. 52 Hz verstellt (ca. 833Hz pro 16Schritten). Eine Erhöhung des Wertes bewirkt jedoch eine Verringerung der Sendefrequenz.

