

# **2 EASY**

## **Technische Informatik 2**

**Projekt:  
embedded processor**

**bei Dr. Sommer**

**Ahmet Emre Açar  
174387**

**Jan Koernicke  
174336**

**Martin Apel  
165290**

## 2 EASY INDEX

|                                      |           |
|--------------------------------------|-----------|
| <b><u>2 EASY INDEX</u></b>           | <b>2</b>  |
| <b><u>Vorwort</u></b>                | <b>3</b>  |
| <b><u>Idee &amp; Konzept</u></b>     | <b>3</b>  |
| <u>Funktionsweise</u>                | 4         |
| <b><u>Spezifikationen</u></b>        | <b>4</b>  |
| <u>Eine Erklärung zum Kapitel</u>    | 4         |
| <u>Allgemeines</u>                   | 4         |
| <u>Der “2 Easy” – Design</u>         | 5         |
| <u>Der Speicher</u>                  | 5         |
| <u>Das Display</u>                   | 5         |
| <u>Sende/ Empfangs – Einrichtung</u> | 6         |
| <u>Die PC-Schnittstelle</u>          | 7         |
| <b><u>Implementation</u></b>         | <b>8</b>  |
| <b><u>BYO (build your own)</u></b>   | <b>9</b>  |
| <b><u>Anwendungsszenarien</u></b>    | <b>9</b>  |
| <b><u>Datenformate</u></b>           | <b>10</b> |
| <b><u>Instruction Format</u></b>     | <b>10</b> |
| <b><u>Speicherorganisation</u></b>   | <b>12</b> |
| <b><u>Struktur</u></b>               | <b>13</b> |
| <u>Erläuterungen zur Struktur</u>    | 14        |
| <u>Bauweise</u>                      | 14        |
| <b><u>Flußdiagramm</u></b>           | <b>15</b> |
| <b><u>Assembler</u></b>              | <b>16</b> |
| <b><u>Microprogram</u></b>           | <b>21</b> |
| <b><u>CU microprogrammed</u></b>     | <b>23</b> |
| <b><u>Ausblick</u></b>               | <b>23</b> |
| <b><u>Fazit</u></b>                  | <b>23</b> |

### Vorwort

Der vorliegende Entwurf des “2 Easy” – Gerätes ist das Ergebnis einer Projektarbeit im Rahmen des Informatikstudiums, Teilbereich Technische Informatik. Beteiligt waren Martin Apel, Ahmet Acar und Jan Koernicke. Betreut wurde das Praktikum von Dr. Siegmund Sommer. Die Aufgabenstellung lautete, ein Produkt im Bereich mobile, bzw. embedded Computing zu entwickeln und vorzustellen. Mit dieser Arbeit glauben wir, der Aufgabe gerecht zu werden.

### Idee & Konzept

Die Idee der persönlichen Visitenkarte “2 Easy” wurde aus einem ganz praktischen Problem heraus geboren:

Zum einen ist da das Problem, welches sich immer wieder ergibt, wenn man sich von einem Mitmenschen ‚Kontaktmöglichkeiten‘ notieren will. In der Regel ist es doch so, dass man gerade nichts zur Hand hat, um sich die Telefonnummer, eMail-Adresse oder was auch immer zu notieren. Ist dann doch etwas gefunden, so gestaltet sich der Vorgang unnötig aufwendig und kompliziert (Stift raussuchen, Organizer aktivieren, wildes auf dem Handy Tippen, etc.).

Ein anderes Problem ist die Verwaltung von Adressdaten, v.a. Telefonnummern. In einer Zeit wo Handy und womöglich auch ein Organizer schon Einzug in das persönliche Chaos-Management gefunden haben, der papierene Kalender aber noch nicht endgültig ausgedient hat und die Zettel, wo man sich mal eben was notiert, sowieso immer präsent sind, passiert es nicht zu selten, dass sich die Suche nach den gewünschten (und irgendwo auch notierten) Informationen zumindest als umständlich, wenn nicht sogar erfolglos erweist. Wer schafft es schon, eine strikte Ordnung und Disziplin in seine persönlichen ‚Datenbanken‘ zu bringen?

Dieses Problem löst der “2 Easy”. Telefonnummern, eMailadresse & Anschrift können schnell und problemlos ausgetauscht und jederzeit ebenso schnell wieder abgerufen werden. Da das Gerät klein, handlich und unkompliziert ist, kann (sollte) sich sein Design so gestalten lassen, dass man es immer schnell zur Hand hat.

Darüber hinaus ist der “2 Easy” auch eine ausgesprochen preiswerte Lösung, die die Preisklasse etwa eines PDA weit überdimensioniert erscheinen läßt. Dies ist auch eine wesentliche Voraussetzung für eine weite Verbreitung des Gerätes – und ein ganz wesentlicher Punkt für die Frage des Erfolgs eines neuen technischen Gerätes. Zu guter Letzt vereint der “2 Easy” auch Einfachheit mit Flexibilität: Im täglichen Gebrauch ist es intuitiv zu bedienen und erfordert keinerlei technisches Hintergrundwissen; durch seine hohe Spezialisierung ist es für die gestellte Aufgabe hervorragend geeignet und gleichzeitig die günstigste Alternative. Und eine besondere Möglichkeit zur Erweiterung des Funktionsumfanges bietet die PC-Schnittstelle. Hiermit kann der Benutzer den “2 Easy” seinen individuellen Bedürfnissen anpassen und ist nicht einer einseitigen Konzeption ‚ausgeliefert‘.

### Funktionsweise

Eine kurze Andeutung der Funktionsweise des Gerätes:

Im Speicher des “2 Easy” stehen an einer festen Adresse die eigenen Daten (also die des Besitzers), welche per PC-Schnittstelle nach Belieben editiert werden können. Auf Knopfdruck tauschen dann zwei der mobilen Geräte per Funk ihre ‚eigenen‘ Daten miteinander aus und speichern die empfangenen ‚fremden‘ Daten. Ein Datensatz besteht maximal aus Name, Vorname, zwei Telefonnummern und eMail-Adresse. Des weiteren ist eine Such- und Anzeigefunktion von gespeicherten Daten im mobilen Gerät implementiert, so dass man immer auf diese Zugriff hat – die Ausgabe erfolgt über ein kleines Display am Gerät..

Über die PC-Schnittstelle können dann sowohl der eigene, als auch die gespeicherten Datensätze geschrieben, gelesen und bearbeitet werden. (Genauere Erläuterungen zu den Funktionsweisen finden sich in den einzelnen Kapiteln der Projektarbeit.)

## 2 EASY

### Spezifikationen

#### Eine Erklärung zum Kapitel

Die Spezifikationen werden teilweise aufgeteilt in “Allgemeine Betrachtungen” und “Technische Referenz”.

Unter ersterem finden sich die grundlegenden Überlegungen zum Gerät, in welchen verschiedene Alternativen diskutiert werden – sie zeigen die technischen Anforderungen, Überlegungen und Möglichkeiten auf relativ breitem Raum auf.

Die “Technische Referenz” bietet einen konkreten technischen Ansatz. Dieser ist nicht immer unbedingt der beste, allerdings bietet er ein ganz konkretes Beispiel heutiger Technik, welches den Bedürfnissen in etwa entsprechen könnte. Dass sich bei näherer und ernsthafterer Beschäftigung mit der Problematik eine bessere Möglichkeit ergeben könnte, ist keineswegs ausgeschlossen.

#### Allgemeines

Das Gerät als ganzes besteht im wesentlichen aus einem Display, einer Sende / Empfangs – Einrichtung, einer Stromquelle, zwei Knöpfen zur Bedienung, dem Speicher und natürlich einer CPU. Außerdem gehört noch eine Dockingstation für PCs zum Gerät – diese ist aber natürlich ein extra Gerät.

#### Eingabe

Es gibt 2 Tasten zur Kontrolle der Funktionen. 2 bit reichen aus, um die Zustände der Tasten abzurufen: A aktiv, B aktiv, beide aktiv, beide inaktiv.

#### ALU

Die ALU besteht nur aus einem einfachen Addierer. Subtraktionen werden durch die Nutzung eines Subtraktionsregisters (SR) durchgeführt. Für Jump-Operationen kann ein Zero Flag gesetzt werden.

### Der “2 Easy” – Design

Das Volumen des “2 Easy” sollte das einer Streichholzsachtel nicht übersteigen, womöglich ist es auch in der halben Größe realisierbar. Das endgültige Layout kann und soll nicht festgelegt sein. Im Gegenteil soll das Produkt in den verschiedensten Ausführungen zur Verfügung stehen. Denkbar wären z.B. eine Integration in Armbanduhren, ein Design als Schmuckstück, eine Ausführung in Form einer flachen Karte oder gar eine Integration in Kleidungsstücke oder andere Gebrauchsgegenstände, z.B. als Schlüsselanhänger. Die sehr unterschiedlichen Layout-Entwürfe des Gerätes sind eine Referenz an den Kunden, welcher dann nach eigenen Vorlieben und Bedürfnissen aus einer breiten Palette wählen kann. Auch kann durch immer neue Entwürfe die Attraktivität des Gerätes über längere Zeit erhalten werden.

Für den Benutzer sichtbar und allen Geräten (in welcher Form auch immer) gemeinsam sind nur die zwei Knöpfe zur Bedienung und das Display. Die restlichen Bestandteile sind von Außen nicht zu identifizieren (von einem möglichen “Transparency-Design” abgesehen ;)) und im einzelnen für den Benutzer uninteressant.

### Der Speicher

#### Allgemeine Betrachtungen

Die benötigte Speicherkapazität des Geräts ist fast unwesentlich klein – die maximal adressierbare Speichermenge liegt bei 16 Kbyte (siehe VI.). Wahrscheinlich wird jedes Gerät einen 128 Kbyte- Chip besitzen. Dieser ist zwar um ein Vielfaches größer als der benötigte Speicher, allerdings sind 128 Kbyte- Chips aus der industriellen Massenfertigung billiger als kleinere Modelle, welche womöglich extra gefertigt werden müßten. Als Speichermedium kommen natürlich nur elektronische Speicher-Chips in Frage; diese sind den Erfordernissen entsprechend problemlos und billig verfügbar.

#### Technische Referenz

Der Speicher besteht aus einem wiederbeschreibbaren, von der Stromzufuhr unabhängigen Baustein. Die Speicherstellen sind segmentiert in Einträge und Positionen. Ein Eintrag umfasst 64 Positionen, eine Position ist 2 Byte groß. Die Anfangspositionen des Speichers sind für das Programm reserviert (Programmspeicher), die restlichen Positionen für die Einträge (Datenspeicher). Der Gesamtspeicher umfasst 16Kbyte.

### Das Display

#### Allgemeine Betrachtungen

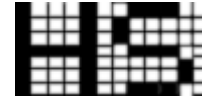
Idealerweise sind auf dem Display bei einer Breite von nicht mehr als zwei bis drei Zentimetern 20 oder mehr Zeichen darstellbar. Dies entspricht in etwa der möglichen Auflösung von neueren Handy-Modellen. Allerdings sollten hier auch finanzielle Überlegungen eine Rolle spielen, so dass die 20 Zeichen keine dringende Notwendigkeit sind; weniger ist grundsätzlich möglich und relativ problemlos realisierbar. Auf eine Hintergrundbeleuchtung der Anzeige muß verzichtet werden. Dies ist zum einen durch den möglichst geringen Stromverbrauch und v.a. der Größe des Geräts bedingt. Durch eine Hintergrundbeleuchtung würde die Dicke des Displays um ein Vielfaches zunehmen und einen großen Teil des gesamten Volumens ausmachen.

Eine absolute Notwendigkeit des Displays ist allerdings ein internen Speicher von min. 128 Byte. Dieser dient zur Pufferung des aktuellen Eintrages, der gerade angezeigt wird und ist nach dem technischen Entwurf der CPU unverzichtbar.

### Technische Referenz

Zeichen werden im Unicode in einen Zeichengenerator (ROM) eingegeben; von dort aus gelangen sie zum Signalgenerator (und dann zum Display selbst). Das Displayinterface hat einen  $2^7$  Byte großen Zwischenspeicher. Das Display selbst besteht aus einer Punktmatrix mit  $5 \times 7$  Rasterung für ein Zeichen. Die Gesamtbreite des Displays beträgt 66 Punkte (55 für Zeichen, je 11 Punkte Zwischenraum).

Das Display eine Höhe von  $\frac{1}{2}$  cm und eine Breite von 2 cm.



### Sende/ Empfangs – Einrichtung

#### Allgemeine Betrachtungen

Hier lautet die Gretchenfrage: Infrarot oder Funk? Infrarot bietet Vorteile, was den Platzbedarf, den Preis und auch den Stromverbrauch bietet. Eine technische Lösung wäre mit der IrDA- Technik<sup>1</sup> zur Hand – diese Schnittstelle ist auch schon weit verbreitet, was die Konzeption der PC-Schnittstelle des “2 Easy” erleichtern würde. Allerdings gibt es auch gute Gründe, die gegen IrDA sprechen: So gibt es kein einheitliches Datenprotokoll. Diese sind im Gegenteil sehr verschieden und mitunter ziemlich umständlich<sup>2</sup>. Darüber hinaus hat Infrarot natürlich den ganz erheblichen Nachteil, dass eine Sichtverbindung zwischen den einzelnen Geräten bestehen muß und es nur einen bestimmten Toleranzwinkel beim Austausch von Daten gibt. Und auch die Zukuntorientierung sieht bei Infrarot nicht sehr gut aus; diese Art der Übertragungstechnik hat ihre Hochzeit hinter sich, in Zukunft wird die Datenübertragung per Funk immer mehr zur Normalität werden.

Rein theoretisch wäre Bluetooth<sup>3</sup> die ideale Lösung für die Übertragung. Hier gibt es ein ordentliches Protokoll, in Zukunft eine Vielzahl von Schnittstellen an allen möglichen Geräten und quasi keinerlei Beschränkungen beim praktischen Gebrauch. Allerdings ist Bluetooth eine Dimension zu groß gedacht – sowohl was die räumliche als auch was die finanzielle Größe betrifft, ist diese Technik im Moment wohl eher eine Utopie für den “2 Easy”. Dies mag in ein oder zwei Jahren ganz anders aussehen – immerhin läuft die Entwicklung von Bluetooth, einschließlich der technischen Gerätschaften auf Hochtouren – zur Zeit allerdings müßte sich ein potentieller Hersteller des Gerätes wohl nach anderen Lösungen umschauen.

Nun ist das Feld an Funk-Lösungen schon heute ein sehr weites, so dass sich bei ordentlicher Suche wohl durchaus eine akzeptable Funklösung finden lassen würde. Die Größenordnung von Computer-Funkmäusen ist vielleicht eine (wenn auch noch etwas überdimensionierte) Orientierung.

Die beste Lösung ist im Moment also ein bescheidenes Funk – Protokoll. Es ist anzunehmen, dass sich eine derartige Lösung mit etwas Aufwand auf dem Markt finden läßt. (Ein Erfordernis der Software an dieses Protokoll wäre zumindest, dass es die Kommunikation zwischen zwei mobilen Geräten und zwischen mobilem Gerät/ PC-Schnittstelle unterscheiden kann. Dies sollte jedoch z.B. durch eine einfache Kennungen am Anfang jeder Übertragung zu realisieren sein.)

---

<sup>1</sup> <http://www.irda.org/>

<sup>2</sup> [http://www.vishay.com/docs/fmod\\_data\\_formats.pdf](http://www.vishay.com/docs/fmod_data_formats.pdf)

<sup>3</sup> <http://www.bluetooth.com>

### Technische Referenz

Daten gelangen zuerst in einen  $2^7$  Byte großen Zwischenspeicher und werden von dort aus an die Funk-Einheit weitergegeben. Interface und Funkprotokoll sind kompatibel zu seriellen Schnittstellen (nach DIN 66020). Somit können Funkmäuse und -Tastaturen angesprochen und die Daten an den Rechner übertragen werden. Reichweite beträgt etwa 1m.

Serielle Interfaces haben eine Übertragungsrate von min. 1200 Bit/s. Dies entspricht 150 Byte/s, d.h. ein Eintrag wird innerhalb einer Sekunde übertragen. Die Mindestanforderungen, die an den Funk gestellt sind, werden jedoch durch die Tatsache übertroffen, daß heutige serielle Schnittstellen (insbesondere Funkmäuse und -Tastaturen) eine Rate von 9600 Bit/s haben. Die Wahl der Übertragungsrate hängt somit von Energieverwaltung und die damit verbundene Kostenfrage ab.

Das Funkinterface verfügt über ein Datenregister und eine Vergleichsfunktion zur Überprüfung der Gültigkeit von Empfangenen Daten. Über ein Statusregister werden die drei Zustände (Empfangen, Senden, Inaktiv) angesprochen. Empfang- und Sendemodus wechseln sich jedoch in Intervallen ab (es wird also gleichzeitig gesendet und empfangen). Die Kommunikation mit der Gegenstelle ist z.B. durch ein handshake denkbar.

### Die PC-Schnittstelle

#### Allgemeine Betrachtungen

In diesem Entwurf des "2 Easy" findet die PC-Schnittstelle vorerst nicht sehr viel Beachtung, da sie zum einen nicht zum eigentlichen Gerät gehört und zweitens hauptsächlich aus ,normaler' Software-Programmierung besteht. Die technische Seite besteht aus einer simplen Sende / Empfangs-Einrichtung analog zu der im mobilen Gerät; sie muß von dieser Seite keinen besonderen Ansprüchen genügen. Diese Einrichtung wird dann über eine Schnittstelle (seriell, USB, etc.) an der PC angeschlossen und der Rest ist reine Software – Programmierung.

Aufgabe dieser Software ist das Lesen und Schreiben des Speichers der mobilen Einheit. Der Benutzer hat über seinen PC Zugriff auf seine eigenen gespeicherten Daten (die, welche er an andere weitergeben will) und auf die empfangenen Daten. Auf den Programmspeicher des Gerätes hat der Benutzer keinen Zugriff! <sup>4</sup>

Nur per PC können einzelne Datensätze im Gerät gelöscht oder verändert werden. Auch sollte die Software in der Lage sein, die Datensätze an andere gängige Programme (z.B. Outlook) zu exportieren und selbständig Tabellen zu erstellen, welche gespeichert und ausgedruckt werden können.

---

<sup>4</sup> Dieser Zugriff auf den Programmspeicher könnte evtl. in späteren Entwürfen realisiert werden, wenn eine Art Entwicklungsumgebung für die Firmware des Gerätes herausgegeben wird. Benutzer könnten die internen Funktionsweisen des "2 Easy" dann neu ,programmieren'. Allerdings ist dies ein Feature, welches auf Grund seiner Komplexität und ,Sicherheitsproblematik' vorher ausführlich diskutiert werden sollte.

## Implementation

Unsere Anwendungsszenarien stellen folgende Forderungen an die Implementation:

1. niedrige Herstellungskosten
2. hohe Zuverlässigkeit und Kompatibilität
3. geringer Stromverbrauch
4. biegsame Verknüpfung der Komponente

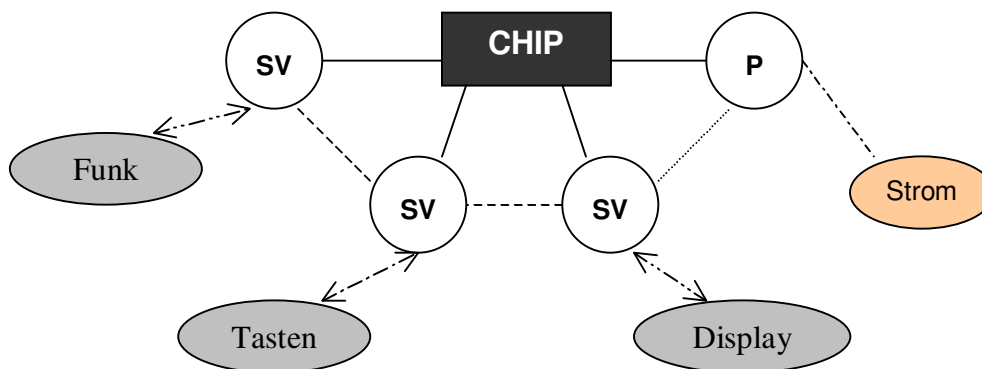
Das System würde möglicherweise als 1-chip System auf einer Karte implementiert werden. Der Aufwand an Hardware wird minimiert, um die Kosten niedrig zu halten. Man benötigt eine hohe Systemzuverlässigkeit. Die gesamte Hardware kann auf einer einzigen Karte untergebracht werden, die Funktionseinheiten (Prozessor, Programm- und Datenspeicher, Ein/Ausgabe Interfaces) sind über die Signalleitungen des Prozessors (Prozessorbus) miteinander verbunden. Die Adressierung von Systemkomponenten erfolgt speicherbezogen.

### Vorteile dieser Vorgehensweise:

Niedrige Herstellungskosten. Geringer Stromverbrauch. Elastische Verknüpfungen möglich.

### Nachteile dieser Vorgehensweise:

Prozessor-Struktur kann im nachhinein nicht geändert werden.



Die Signalverstärker zwischen den „Endgeräten“ und dem Prozessor nehmen die Last vom Bus, damit dieser kapazitiv nicht überlastet wird. Die Stromquelle ist an die Peripherie angeschlossen.

|                             |
|-----------------------------|
| SV = Signalverstärker       |
| P = Anschluss an Peripherie |

Da das Gerät tragbar sein soll, muss die Stromquelle leicht sein und effizient ausgenutzt werden. Neben Batteriebetrieb ist eine Wiederaufladung durch Solarenergie, kinetische Energie und Wärmeenergie (z.B. Körperwärme) denkbar. Statt einer Batterie könnte auch ein selbst aufladender Zinkakku (lädt sich mit Sauerstoff aus der Luft auf) verwendet werden. Die Wahl der Energiequelle hängt jedoch stark von der finanziellen Tragbarkeit ab.

## BYO (build your own)

Das Konzept sieht vor, daß Karte und I/O Module flexibel benutzt werden können. Das Gerät kann sowohl in ein eigenes Gehäuse als auch in Kleidung oder Accessoires eingebaut werden. Der ursprüngliche Gedanke sah die Verwendung in Form einer Visitenkarte vor.

Wichtig ist, die anfallenden Kosten möglichst gering zu halten, damit das Gerät Zuspruch findet. Die genauen Kosten können von uns nicht kalkuliert werden, aber ein grober Aufbau würde folgende Unkosten veranschlagen:

|  |                   |
|--|-------------------|
| Speicher (EP-ROM, benutzt, 128Kbyte).....      | DM 1,50.-         |
| Funkmodul (hochfrequent, inkl Antenne).....    | DM 12,90.-        |
| Display (kl. Kristalldisplay).....             | DM 5,00.-         |
| Tasten (2 Stück).....                          | DM 0,50.-         |
| Hauptkarte (Lötplatte, Transistoren etc.)..... | ~DM 8,00.-        |
| <b>Insgesamt.....</b>                          | <b>DM 27,90.-</b> |

In dieser Beispielrechnung wurden die Komponente aus dem Einzelhandel herausgesucht. Statt einer Komponenten-Bauweise könnte man alles auf einen einzigen Chip pressen. Die einzigen weiteren Komponente wären dann die Signalverstärker, das Display und die Funkantenne. In beiden Fällen kommen noch die Kosten für ein Gehäuse und die Stromversorgung dazu. Diese können jedoch je nach Präferenz sehr variieren. Zuletzt sollte vermerkt werden, daß bei einer Massenproduktion der Preis noch zusätzlich sinkt.

## Anwendungsszenarien

### Messeinsatz

Einarbeitung des 2 Easy in die Messeausweise der Aussteller und Fachbesucher. Somit haben sowohl Aussteller als auch Fachbesucher die Möglichkeit ihre Daten schnell und unkompliziert auszutauschen. Gegenbenfalls könnten die Datensätze erweitert werden, um zusätzliche Informationen (Standinfo, Firmeninfo etc) zu beinhalten.

### Partyeinsatz

Dieses Szenario entspringt dem ursprünglichen Problem der „kleinen Zettelchen“ und Bierabsetzer, auf denen verschmierte Nummern kleben und die dann in der Wohnung verteilt herumliegen. Hier würden Telefonnummer und Name als Einträge genügen- das Gerät müßte in dem Fall robust untergebracht werden und „stylish“ aussehen (etwa als Schmuck getarnt).

### Konzerteinsatz

Es wäre denkbar, 2 Easy soweit zu verkleinern, daß sie auf Konzertkarten Platz finden. Man könnte das Gerät auch in Leuchtstangen verpacken und sie mit den Karten ausgeben (damit verbrennt man sich bei den langsamen Stücken wenigstens nicht die Finger).

### Weiteres

Generell wäre 2 Easy ein nettes Werbegeschenk. Im Handel könnte man verschiedene Gehäuse für das Gerät anbieten und die Dockingstation (für Leute ohne Funkmaus) würde natürlich auch was kosten.

## Datenformate

Es gibt **2** verschiedene Datenformate.

### 1. Unsigned Integer (16Bit)

Der Wert wird im Zweierkomplement abgelegt und hat demnach einen Wertebereich von 0-65535

### 2. Unicode-Charakter (16Bit)

besteht im Prinzip aus einem Unsigned Integer, allerdings ist jedem Wert ein Buchstabe zugeordnet

Daraus werden **2** weitere Datentypen abgeleitet:

#### 1. String

Mehrere aufeinander folgende Unicode-Character, eventuell durch ein 0 Wort terminiert

#### 2. DER Datensatz (64 Worte – 1024 Bit)

String name //maximal 10 Zeichen

String vorname //maximal 10 Zeichen

String Telefon1 //maximal 11 Zeichen

String Telefon2 //maximal 11 Zeichen

String Email //maximal 22 Zeichen

## Instruction Format

Eine Instruction umfaßt genau 32Bit:

|             |             |             |                  |                   |
|-------------|-------------|-------------|------------------|-------------------|
| 2Bit Opcode | 2Bit Mode 1 | 2Bit Mode 2 | 13Bit 1. Operand | 13 Bit 2. Operand |
|-------------|-------------|-------------|------------------|-------------------|

Es gibt genau  $2^2$  Addressierungsmodi:

1.Immediate

2.Speicher direkt

3.Register direkt

4.Register indirekt

Wird der Modus Immediate verwendet, so bedeutet dies natürlich, dass nur ein enger Wertebereich darstellbar ist (0-8191). Dies stellt für unsere Anwendung kein Problem dar.

Zu codieren sind folgende **Zwei-Address**befehle:

| Befehl | Opcode | 1. Operand               | 2. Operand | Beschreibung   |
|--------|--------|--------------------------|------------|--|
| mov    | 00     | Ziel                     | Quelle     | Bewegt einen Wert von der Quelle zum Ziel. Für den 1. Operanden ist kein Immediate Wert erlaubt.   |
| add    | 01     | 1. Summand und Summe     | 2. Summand | Addiert die beiden Werte und schreibt das Ergebnis wieder in den ersten Operanden. Auch hier ist für den 1. Operanden kein Immediate erlaubt.    |
| sub    | 10     | Subtrahend und Differenz | Minuend    | Subtrahiert die beiden Werte und schreibt das Ergebnis in den 1. Operanden. Auch hier ist natürlich kein Immediate für den 1. Operanden erlaubt. |

Ist der Opcode 11, so bedeutet dies, dass der Befehl ein 1 Addressbefehl ist und dass das Addressfeld des 1. Operanden verwendet wird um den Befehl genauer zu spezifizieren. In diesem Fall ist der Addressmodi für den 1. Operanden 00 also Immediate.

Es werden die folgenden Ein-Addressbefehle verwendet:

| Befehl | Opcode | 1. Operand | 2. Operand       | Beschreibung  |
|--------|--------|------------|------------------|---|
| jmpz   | 11     | 000        | Sprungziel       | Springe wenn Zero-Flag gesetzt  |
| jmpu   | 11     | 001        | Sprungziel       | Springe wenn Underflow-Flag gesetzt   |
| jmp    | 11     | 010        | Sprungziel       | Springe   |
| outf   | 11     | 011        | Datensatzadresse | Datensatz zum Funk schicken   |
| inf    | 11     | 100        | Datensatzadresse | Datensatz vom Funk zur Adresse schreiben  |
| outd   | 11     | 101        | Datensatzadresse | Datensatz zum Display schicken  |
| andf   | 11     | 110        | Operand          | Verknüpft das Flagregister mit dem 2. Operanden und schreibt das Ergebnis auch dort hin(daher kein Immediate) |

## Speicherorganisation

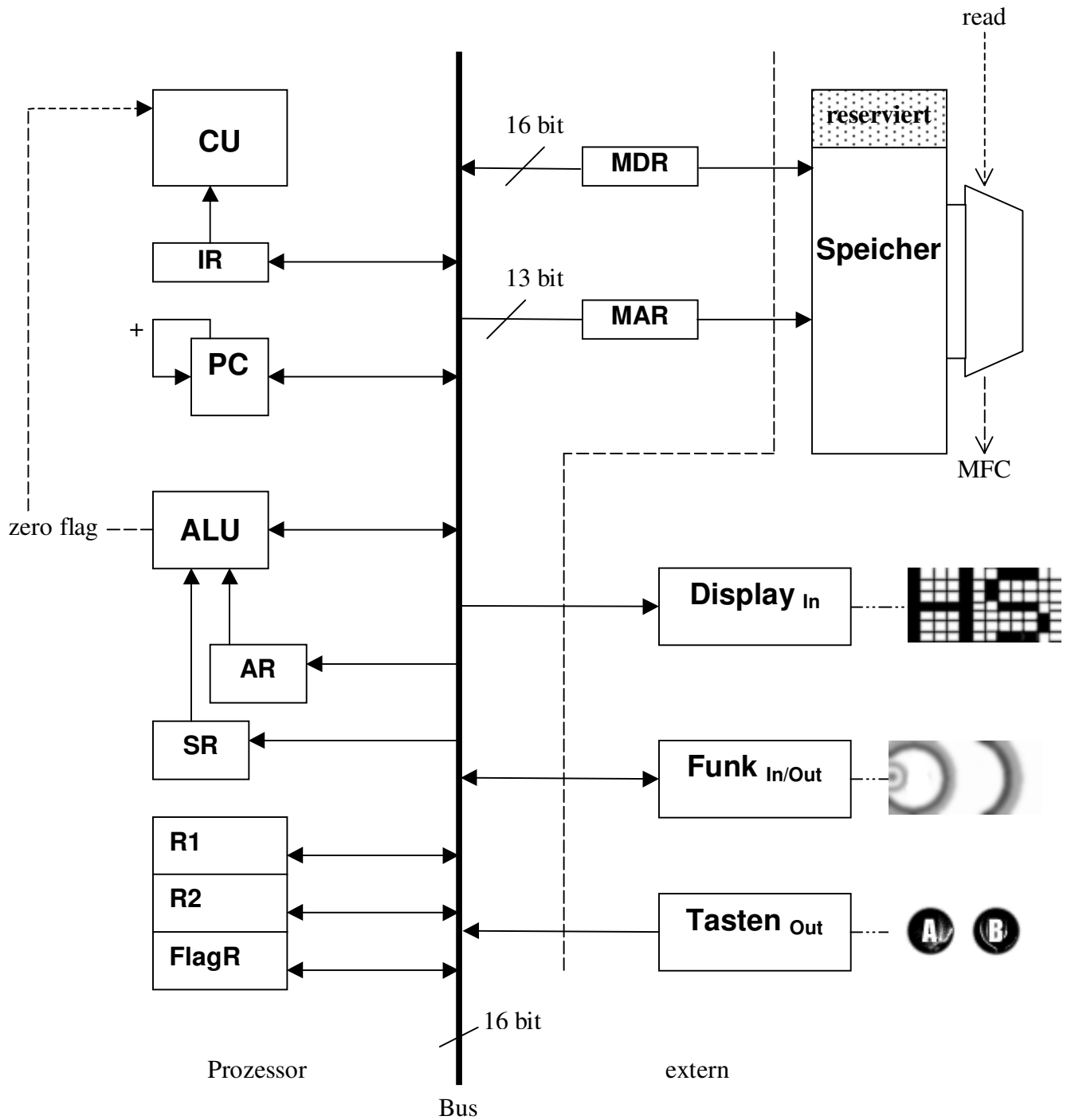
Der Speicher ist Wortweise adressierbar, da der kleinste Datentyp bereits 16Bit umfasst. Da der Prozessor eine Addressbusbreite von 13Bit hat, können 8192 Worte adressiert werden, also genau 16284 Byte.

Hierbei sind einige Adressen reserviert, diese Adressen werden für die Übergabe von Datensätzen an den Funk und den eigenen Datensatz verwendet.

Zuerst kommt der eigene Datensatz (also der eigene Name etc.) und dann der Funkeingang. Nach diesen 128 Worten folgen die Variablen die das Programm benötigt, dann das Programm selbst und schliesslich stehen am Ende des Speichers noch die gesammelten Einträge. Es können maximal 100 Einträge gespeichert werden.

| Adresse | Länge<br>(in Wörtern) | Beschreibung  |
|---------|-----------------------|---|
| 0       | 64                    | Datensatz der die eigenen Daten enthält                             |
| 64      | 64                    | Buffer für den Funk - hier landen empfangene Datensätze             |
| 128     | 100                   | Tabelle mit den sortierten Adressen der Datensätze                  |
| 228     | 1                     | Länge der obigen Tabelle  |
| 229     | 14                    | Hier steht der Schlüssel nach dem gesucht werden soll               |
| 243     | 1                     | Zeiger auf die Position an der der nächste Datensatz eingefügt wird |
| 244     | 1                     | Einfach nur ein Zwischenspeicher                                    |
| 245     | 1                     | Pointer auf den 1. String (vergleichen)                             |
| 246     | 1                     | Pointer auf den 2. String (vergleichen)                             |
| 247     | 1                     | Variable n (in comp verwendet) - Position des aktuellen Buchstabens |
| 248     | 1                     | Rückgabewert von Comp (ungleich)                                    |
| 249     | 1                     | Hier wird die Rücksprungadresse von Comp gespeichert                |
| 300     | 1492                  | Das eigentliche Programm  |
| 1792    | 6400                  | Die gespeicherten Datensätze  |

## Struktur



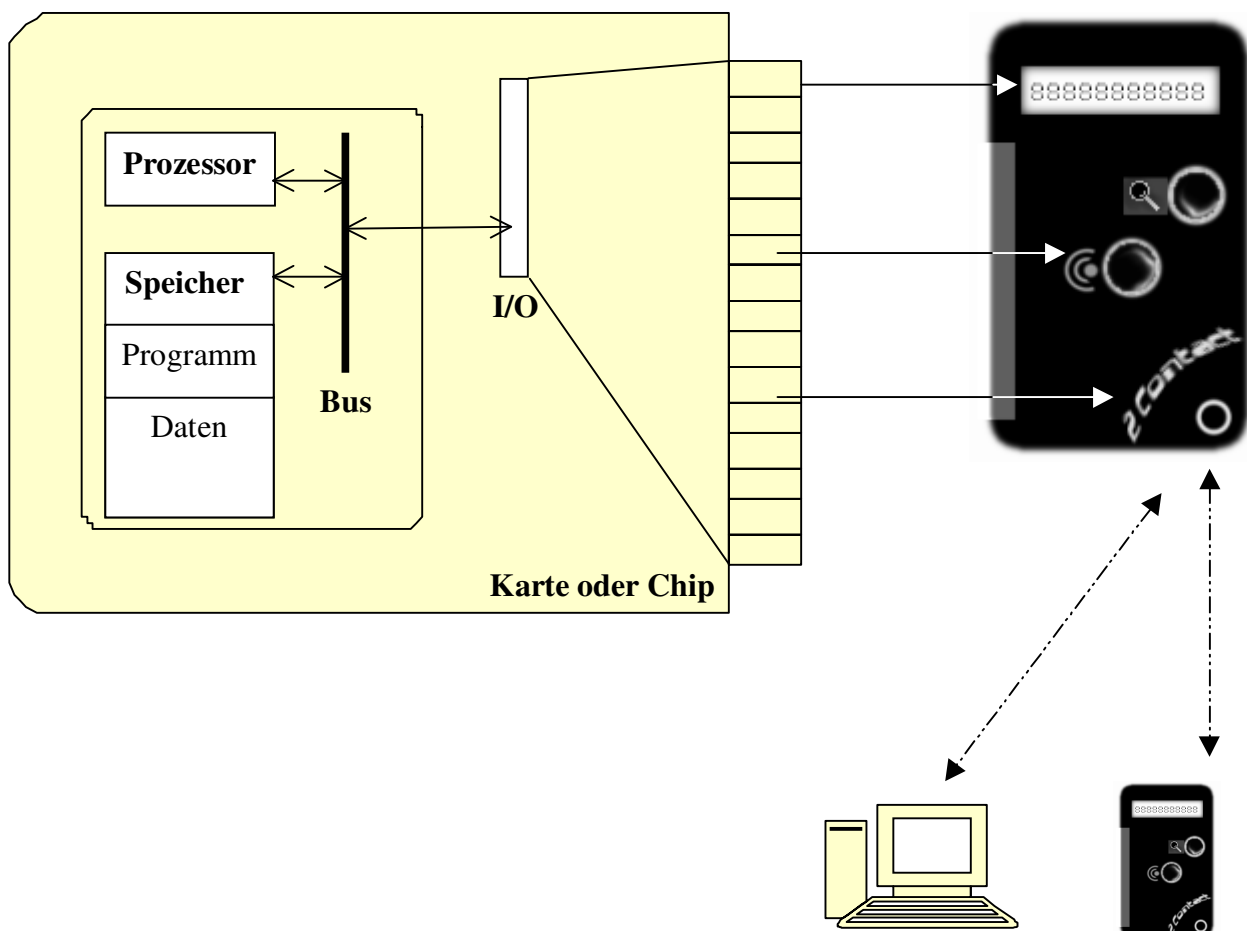
## Erläuterungen zur Struktur:

CU: Control Unit  
 IR: Instruction Register, 32 bit  
 PC: Program Counter, Postincrement, 13 bit  
 ALU: 16 bit Addierer  
 AR: 16 bit Additionsregister (Zwischenspeicher für Addition)  
 SR: 16 bit Subtraktionsregister (Zwischenspeicher für Subtraktion)  
 R1: 16 bit Register  
 R2: 16 bit Register  
 FlagR: 16 bit Flag Register (für Zustandsüberprüfung bei Tasten, Funk und Display)  
 MDR: 16 bit Memory Data Register  
 MAR: 13 bit Memory Address Register

Speicher: reservierter Bereich wird im Normalfall nicht beschrieben (siehe Anwendung),  
 insgesamt  $2^{13}$  Positionen im Speicher adressierbar, 1 Position = 2 byte,  
 insgesamt ~ 16 Kbyte Speicher.

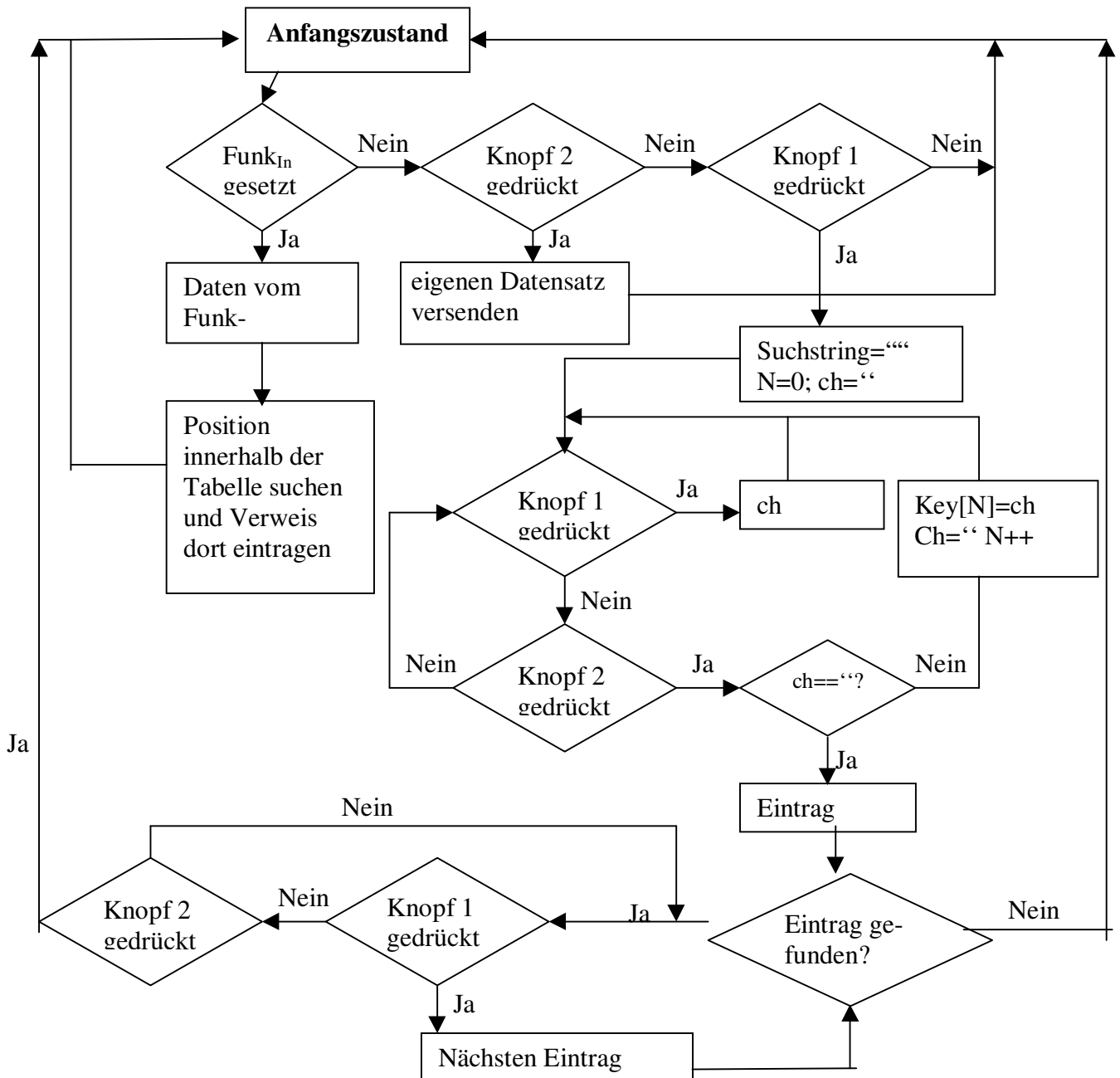
Bus: 16 bit Prozessorbus (umfaßt Datenbus, Adreßbus und Steuerbus)

## Bauweise



## Flußdiagramm

Das Programm nimmt die Knopfdrücke des Benutzers oder die Funkeingaben entgegen und verarbeitet diese entsprechend dem folgenden Diagramm.



## Assembler

Die verwendeten Variablen wurden noch nicht in Speicheradressen aufgelöst, sind aber alle in der Übersicht zur Speicherorganisation zu finden. Auch die Sprungadressen wurden noch nicht in Speicheradressen umgesetzt, allerdings ist auch dies ohne größere Probleme möglich.

```

:0    //Hauptschleife
mov r1, 4
andf r1    //Wenn Funkeingang gesetzt
sub r1, 4
jmpz 200    //Dann einfügen
mov r1, 1
andf r1    //Mit Flags verknüpfen
sub r1, 1    //Knopf 1 gedrückt?
jmpz 30    //Knopf 1 gedrückt!
mov r1, 2
andf r1    //Mit Flags verknüpfen
sub r1, 2    //Knopf 2 gedrückt?
jmpz 20    //Knopf 2 gedrückt!
jmp 0
:20
//Senden - hier wird nur gesendet, empfangen wird in der Hauptschleife
sub flagr, 2 //Flag für Knopf 2 löschen
outf 0 //Datensatz in Funkbuffer übertragen
add flagr, 8 //FunkOut Flag setzen
//Zurück zur Hauptschleife
:30
sub flagr, 1 //Flag zurücksetzen
mov r1, 1 //n=1
mov c, 0 //c=0
:31
mov r2, flagr
sub r2, 2 //Knopf 2 gedrückt?
jmpz 32
jmp 33
32:
sub flagr, 2 //Flag löschen
sub c, 0
jmpz 100 //Wenn c==0 dann suchen gehen
//Ansonsten
add r1, 1 //n erhöhen (die Position)
mov c, 0 //c (Charakter) wieder zurücksetzen
jmp 31
:33
mov r2, 2
andf r2 //Flags mit r2 verknüpfen
sub r2, 2 //k2 gesetzt

```

```

jmpz 34          //Wenn k2 gesetzt
sub k1, 1       //Flag löschen
add c, 1        //c1 erhöhen
jmp 31

:100 //suchen
//Verwendete Variablen:
//tab_length(länge der Tabelle), key(schlüssel nach dem gesucht wird),
//Konstante: TAB_START, TAB_END
//Werte zurücksetzen
mov r1, TAB_START //Adresse des 1. Elements
mov r2, 0
:101
mov r2, r1 //Kopie des Indexes machen
sub r2, TAB_END //Mal gucken ob schon am Ende angekommen
jmpz 0 //Wenn am Ende, dann zurück zur Anfangsschleife(ev. noch
//Meldung)
//Ansonsten
mov r2, [r1] //Adresse des ersten Datensatzes holen
//Comp aufrufen
//Übergabewerte speichern
mov string1, key
mov string2, r2
mov rücksprung, 110 //Ruecksprungadresse speichern
jmp 300 //jmp comp
110: //Ruecksprung für Comp
sub ungleich, 0
jmpz 150 //Wenn ungleich == 0 dann zu 150
//Ansonsten
add r1, 1 //Index erhöhen
jmp 101 //und Suche wiederholen
:150
//r2 enthält die Adresse
//r1 enthält den Index
outd r2 //Ausgabe des Datensatzes an der Adresse die r2 enthält
:151 //Eingabe auswerten
mov r2, 1 //r2 wird mal kurz zum vergleichen verwendet
andf r2 //r2 mit den Flags verknüpfen
sub r2, 1 //Knopf 1 gedrückt?
jmpz 152 //Knopf 1 gedrückt !
mov r2, 2
andf r2 //r2 mit den Flags verknüpfen
sub r2, 2 //Knopf 2 gedrückt?
jmpz 153 //Knopf 2 gedrückt!
jmp 151 //Wenn keine Knöpfe gedrückt, einfach warten
:152
sub flagr, 1
add r1, 1 //Auf nächsten Datensatz zeigen

```

```

mov r2, [r1] //Adresse des Datensatzes holen
//Comp aufrufen
//Übergabewerte speichern
mov string1, key
mov string2, r2
mov rücksprung, 155 //Rücksprungadresse speichern
jmp 300 //jmp comp
:155 //Rücksprungadresse
sub ungleich, 0
jmpz 151 //Wenn gleich - Datensatz anzeigen
//Ansonsten
mov r1, 0 //Werte zurücksetzen
mov r2, 0
jmp 0 //Zurück zur Hauptschleife
:153
sub flagr, 2 //k2-Flag löschen
mov r1, 0 //Werte zurücksetzen
mov r2, 0
jmp 0 //Zurück zur Hauptschleife

:200 //einfügen
//Benötigte Variablen
//add_pos(hier werden neue Elemente eingefügt),
//zwi(zwischenspeicher)
//Konstante: TAB_START, TAB_END

sub flagr, 4 //FunkFlag löschen
//Datensatz ans Ende anfügen
inf 64 //Eintrag vom Funk holen
mov n, 64 //Zähler initialisieren
mov r2, add_pos //Dest. initialisieren
mov r1, 64 //Source initialisieren(auf Eingangsbuffer)
:210
mov [r2],[r1]
add r2, 1
add r1, 1
sub n, 1
jmpz 220 //Wenn 64mal wiederholt, dann stoppen
jmp 210 //Alles nochmal
:220
//Datensatz in der Tabelle speichern
mov r2, TAB_START //Adresse des ersten Datensatzes holen
//Comp aufrufen
//Übergabewerte speichern
mov string1, add_pos //Name des Datensatzes laden
mov string2, [r2] //Adresse des Tabellendatensatzes nach string2
mov rücksprung, 230 //Ruecksprungadresse speichern
jmp 300 //jmp comp

```

```

230:                //Rücksprungadresse von comp
sub ungleich, 2    //String1>String2 ?
jmpz 235          //String1>String2 !
//Ansonsten
add r2, 1         //Index erhöhen
jmp 220          //Und weitermachen
235:
//Elemente verschieben und einfügen
//Stelle ist in r2
mov r1, [r2]     //Inhalt der Speicherstelle in r2 nach r1
mov [r2], add_pos //Einfügen des Elementes
add tab_length, 1 //Tabellenlänge vergrößern
:240
//Den ganzen Schund nach unten verschieben
//in r2 steht der Index, in r1 das einzufügende Element
mov zwi, [r2]
mov [r2], r1
mov r1, zwi
add r2, 1       //Erhöhen von r2 - also des Indexes
mov zwi, r2     //Holen des Indexes
sub zwi, tab_length //Vergleichen der beiden Elemente
jmpu 240        //Wenn zwi<tab_length(also der Index<tab_length)
add add_pos, 64 //add_pos aktualisieren
jmp 0 //Und nach Hause (Hauptschleife)

:300 //comp
//Verwendet die folgenden Variablen
//n, ungleich (als rückgabe mit 0: string1=string2, 1: string1<string2,
//2: string1>string2), *string1, *string2, reg1, reg2
//strings sind entweder durch 0 terminiert oder 14 Zeichen lang
//außerdem: rücksprungadresse, hierhin springt das Programm nach
//der Ausführung
//sichern der Register
mov reg1, r1
mov reg2, r2
//Initialisierung
mov n, 15 //n=15
mov r1, string1 //Zeiger auf 1. String
mov r2, string2 //Zeiger auf 2. String
:301
sub n, 1 //n--
jmpz 340 //Wenn bereits 14 Zeichen verglichen
mov string1, [r1] //Holen eines Buchstabens aus dem 1. String nach string1
mov string2, [r2] //Holen eines Buchstabens aus dem 2. String nach string2
//Test auf Ende der Zeichenfolgen
sub string1, 0 //Wenn String1 zu Ende
jmpz 340 //Dann zum Aufrufer mit Gleichheit
sub string2, 0 //Wenn String2 zu Ende

```

```
jmpz 340    //Dann zum Aufrufer mit Gleichheit
//Erhöhen auf die nächste Position
add r1, 1
add r2, 1
sub string2, string1 //Differenz zwischen den beiden Strings(vor dem inc)
jmpz 301    //Wenn gleich dann weiterprobieren
jmpu 310    //r1>r2, da Unterlauf
mov ungleich, 1 //Ansonsten Wert für r1<r2 speichern
jmp 350     //Zurück zum Aufrufer
:310
mov ungleich, 2 //Wert für r1>r2 speichern
jmp 350     //Zurück zum Aufrufer
:340
mov ungleich, 0 //Dann als gleich werten
jmp 350     //Zurück zum Aufrufer
:350 //Verlassen der Comp-Routine
//Restaurieren der Register und Rücksprung
mov r1, reg1
mov r2, reg2
jmp rücksprung
```

## Microprogram

Da der Bus eine Breite von 16 bit hat, kann ein Befehl erst in zwei Anläufen komplett ins 32 bit Befehlsregister geholt werden. Im folgenden wird nur eine Auswahl an Befehlen codiert.

i = immediate

\* = speicherdirekt

R<sup>x</sup> = Angabe des Registers, allgemeine Signale werden mit spezifischen des Befehls Verknüpft. Je nach Adressierungsart sieht das Microprogramm etwas anders aus.

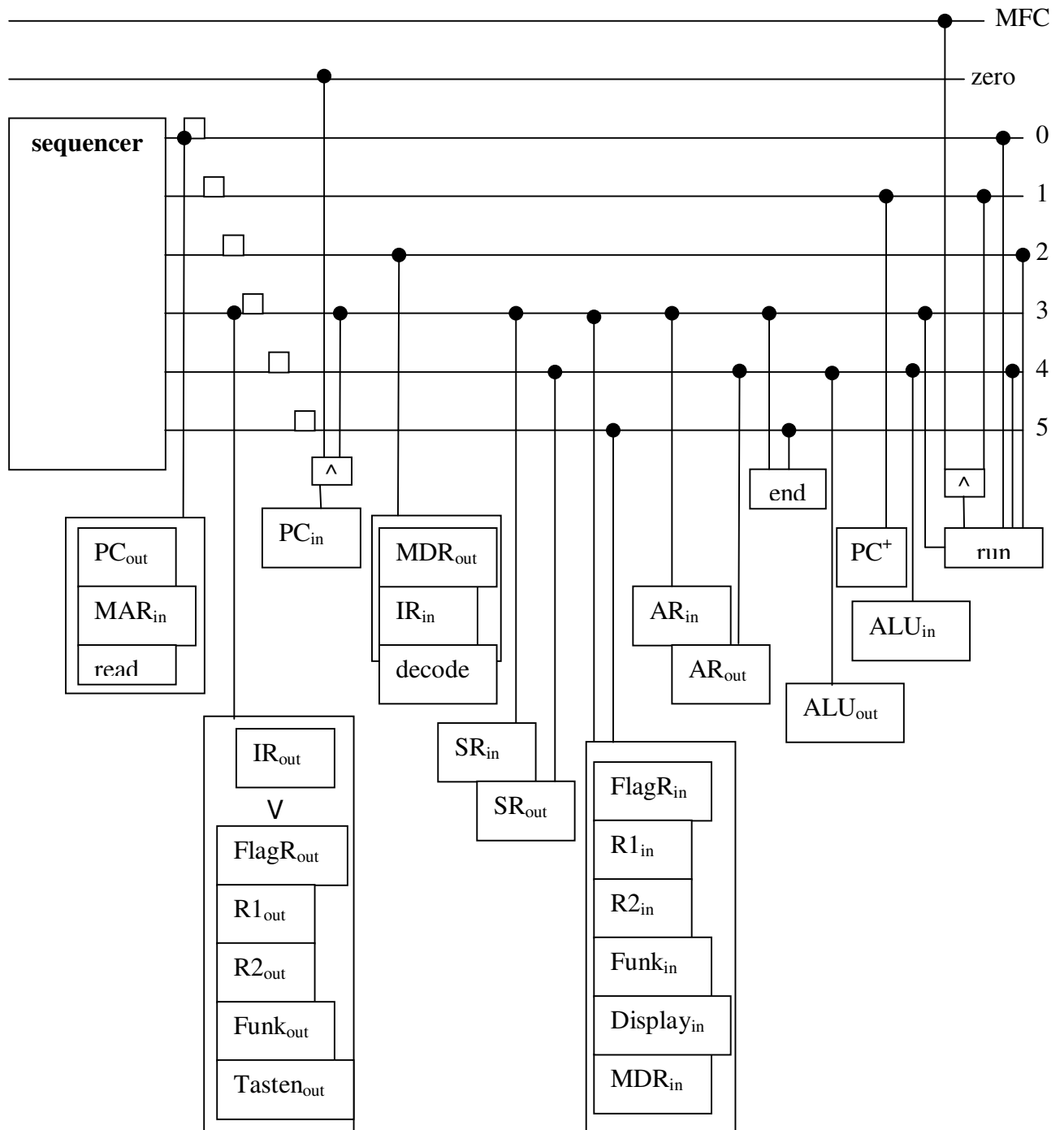
Die fetch Phase geht allen anderen voraus und wird daher in der Tabelle nicht wiederholt.

| Befehl           | Zeit | Prozesse  |
|------------------|------|---|
| fetch            | 0    | PC <sub>out</sub> , MAR <sub>in</sub> , read, run                           |
|                  | 1    | PC <sup>+</sup> , run if MFC  |
|                  | 2    | MDR <sub>out</sub> , IR <sub>in</sub> , decode, run                         |
|                  |      |   |
| mov <sup>i</sup> | 3    | IR <sub>out</sub> , R <sup>x</sup> <sub>in</sub> , end                      |
|                  |      |   |
| add <sup>i</sup> | 3    | IR <sub>out</sub> , AR <sub>in</sub> , run                                  |
|                  | 4    | R <sup>x</sup> <sub>out</sub> , AR <sub>out</sub> , ALU <sub>in</sub> , run |
|                  | 5    | ALU <sub>out</sub> , R <sup>x</sup> <sub>in</sub> , end                     |
|                  |      |   |
| sub <sup>i</sup> | 3    | IR <sub>out</sub> , SR <sub>in</sub> , run                                  |
|                  | 4    | R <sup>x</sup> <sub>out</sub> , SR <sub>out</sub> , ALU <sub>in</sub> , run |
|                  | 5    | ALU <sub>out</sub> , R <sup>x</sup> <sub>in</sub> , end                     |
|                  |      |   |
| jz <sup>i</sup>  | 3    | IR <sub>out</sub> , PC <sub>in</sub> if zero, end                           |
|                  |      |   |
| mov*             | 3    | R <sup>y</sup> <sub>out</sub> , R <sup>x</sup> <sub>in</sub> , end          |
|                  |      |   |
| add*             | 3    | R <sup>x</sup> <sub>out</sub> , AR <sub>in</sub> , run                      |
|                  | 4    | R <sup>y</sup> <sub>out</sub> , AR <sub>out</sub> , ALU <sub>in</sub> , run |
|                  | 5    | ALU <sub>out</sub> , R <sup>y</sup> <sub>in</sub> , end                     |
|                  |      |   |
| sub*             | 3    | R <sup>x</sup> <sub>out</sub> , SR <sub>in</sub> , run                      |
|                  | 4    | R <sup>y</sup> <sub>out</sub> , SR <sub>out</sub> , ALU <sub>in</sub> , run |
|                  | 5    | ALU <sub>out</sub> , R <sup>y</sup> <sub>in</sub> , end                     |
|                  |      |   |
| jz*              | 3    | R <sup>x</sup> <sub>out</sub> , PC <sub>in</sub> if zero, end               |
|                  |      |   |

In das Flagregister und in die Interface Register (Display, Funk, Tasten) kann ebenfalls eingeschrieben bzw. ausgelesen werden. Dies wurde im Microprogramm nicht dargestellt, kommt jedoch aus dem Assemblercode deutlich heraus.

## CU hardwired

Da der Prozessor in der Praxis auf ein chip gepreßt werden würde und nicht sehr komplex ist, nehmen wir an, daß die Implementierung hardwired erfolgen würde. Daher ist hier eine hardwired - Umsetzung ansatzweise veranschaulicht.



Die in einem Blockkasten zusammengestellten Teile wurden nur der Übersicht halber gruppiert.

## CU microprogrammed

Die Aufgaben des Prozessors umfassen die Kontrolle der Peripherie und die Sortierung der empfangenen Daten. Der Sortieralgorithmus wird in Assembler dargestellt, die Manipulation durch den Rechner ist nur für die Daten des Benutzers und die empfangenen Daten vorgesehen. Die Bit-weise Codierung der CU wird hier daher nicht weiter ausgeführt.

Wir gehen davon aus, daß die I/O Interfaces über entsprechende Mechanismen verfügen, um über die CU angesprochen werden und ihre eigenen Aufgaben erledigen zu können. Zu den Aufgaben der I/O gehören etwa die Überprüfung der Datenvalidität im Funkmodul und die automatische Generierung der Zeichen im Displaymodul.

## Ausblick

Ein kurzer Blick auf eine hypothetische Zukunft des Gerätes sei erlaubt: Am grundlegenden Design und an den Funktionsweisen des "2 Easy" wird sich in Zukunft wohl kaum noch etwas verändern. Alle grundlegenden Veränderungen, wie z.B. eine Implementierung von weiteren Funktionen, würden dem Design, welches in erster Linie auf einfach Funktionalität ausgelegt ist, nicht entsprechen.

Entwicklungen können und sollten jedoch v.a. auf der technischen Seite geschehen: bessere Übertragungsart, hochauflösendes Display, flexible Leitungen (etwa aus Stoff) zum Einbau in Kleidung u.ä. sind durchaus wünschenswerte Entwicklungen für die Zukunft. Auch bietet eine fortschreitende Miniaturisierung Möglichkeiten für Layout und Steuerung; denkbar wären Sprach-Ansteuerung oder ein "2 Easy" in Form eines Ringes. Auch die Stromversorgung des Gerätes könnte in den nächsten Jahren spürbare Fortschritte machen – die Batterie, auf die man zur Zeit noch angewiesen ist, könnte evtl. durch eine Stromgewinnung aus Körperwärme oder Luftsauerstoff ersetzt werden; Forschung wird auf diesen Gebieten<sup>5</sup> bereits betrieben.

## Fazit

Insgesamt kann gesagt werden, dass unsere Ansprüche an das Gerät erfüllt werden konnten. Natürlich gab es Punkte, an denen wir unsere anfänglichen Vorhaben zurückstecken mußten – etwa das Bearbeiten von Datensätzen direkt am Gerät. Allerdings stellt die PC-Schnittstelle als Alternative einen akzeptablen Kompromiß dar.

Einige Fragen konnten im Entwurf auch nicht endgültig geklärt werden, u.a. die Diskussion der Sende/ Empfangs – Einrichtung oder des Displays seien hier genannt. Allerdings sind wir der Ansicht, die Machbarkeit hinreichend aufgezeigt zu haben, auch wenn dieses Aufzeigen im Detail noch Lücken beinhaltet (dessen sind wir uns durchaus bewußt).

Auch müssen wir einräumen, dass der eigentlich Prozessor weniger einen Prozessor im „klassischen“ Sinne darstellt, wirkliche Berechnungen werden kaum vorgenommen – eher funktioniert das Gerät als Datenbank oder Sortierwerk im Kleinstformat. Dies ist allerdings ein Resultat der Funktionen, die es zu implementieren galt. Bei der Entwicklung stand der Gedanke an ein möglichst sinnvolles und zweckmäßiges Gerät im Vordergrund, es stand für uns nicht zur Debatte einen „Rechner um des Rechnens willen“ zu implementieren – dies hätte dem Sinn des Projekts nicht entsprochen.

---

<sup>5</sup> <http://www.technologyreview.com>

## 2 EASY

Abschließend können wir wohl sagen, dass wir mit dem entwickelten Projekt zumindest soweit zufrieden sind, als das wir es in der Praxis – also im täglichen Leben – benutzen (und also auch erwerben) würden. Wenn wir mit unserer Idee und der Konzeption auch andere überzeugen können, haben wir jeden Grund, mit dieser Arbeit zufrieden zu sein...

Ahmet Emre Açar  
Jan Koernicke  
Martin Apel