Fachbereich Computertechnik

I B M - P S / 2

TastaturTreiber

SRS

Pascal Näf, Fabian Heusser
Dozent: Alex Klapproth

HTA LUZERN
Technikumstrasse 21
CH—6048 Horw

www.w3p.ch/ps2/; www.planetnaef.ch
Projekt-Ref #: 11

History

Nr.	Datum	Name	Beschreibung
1	2002-04-23	hef	Initial Version C-Requirements
2	2002-06-18	hef	Überarbeitung des ganzen Dokuments

Status

approved final

Inhalt

1. Einführung 3
1.1 Absicht 3
1.2 Umfang 3
1.3 Definitionen, Begriffe und Abkürzungen 4
1.4 Verweise 4
1.5 Überblick 4
2. Allgemeine Beschreibung (C-Requirements) 4
2.1 Produkt Perspektive 4
2.2 Schnittstellen 5
2.2.1 System Schnittstellen 5
2.2.2 Benutzer Schnittstellen 5
2.2.3 Hardware Schnittstellen 5
2.2.4 Software Schnittstellen 5
2.2.5 Kommunikations- Schnittstellen 6
2.2.6 Vorausgesetzte Ressourcen 6
2.2.7 Adaptions-Anforderungen vor Ort 6
2.3 Benutzer Charakteristiken 6
2.4 Vorraussetzungen 6
2.5 Verteilung von Anforderungen 6
3. Konkrete Anforderungen (D-Requirements) 6
3.1 Äussere Schnittstellen 6
3.1.1 Benutzer Schnittstelle 6
3.1.2 Hardware Schnittstelle 6
3.1.3 Software Schnittstelle 7
3.2 Interne Schnittstellen 8
3.2.1 Kommunikationsschnittstelle 8
3.2.2 Datenschnittstelle 8
3.3 Performance Anforderungen 8
3.4 Vorraussetzungen Design 8
3.5 Software System Eigenschaften 8
3.5.1 Zuverlässigkeit 8
3.5.2 Sicherheit 8
3.5.3 Wartbarkeit 8
3.5.4 Portierbarkeit 8

1. Einführung

1.1 Absicht

Dieses Dokument beinhaltet alle Anforderungen für den IBM-PS/2 Tastatur Treiber. Die Kapitel 1 und 2 sind besonders für den Kunden interessant (C-Requirements). Kapitel 3 ist hauptsächlich für den Software Ingenieur (D-Requirements).

1.2 Umfang

Dieses Dokument behandelt die Anforderungen für das Release 1.0 des IBM-PS/2 Tastatur Treiber für den 68HC11. Das SRS soll während der Entwicklung eine Hilfe für den Kunden, wie auch für den Software Entwickler sein.

1.3 Definitionen, Begriffe und Abkürzungen

Begriff oder Term	Beschreibung
C-Requirement	Kunden- Anforderung (Customer): Anforderung an die Anwendung, ausgedrückt in einer präzisen Form, welche dem Kunden verständlich ist.
D-Requirement	Entwickler-Anforderung (Developer): Anforderung an die Anwendung, ausgedrückt in einer präzisen Form, welche genügend detailliert ist, um den Entwicklern für das Design und die Implementation behilflich zu sein.
PS/2 – Device	Ein Gerät das an der PS/2-Tastatur Buchse angeschlossen werden kann. Zum Beispiele Tastatur, Barcodereader.
Scan-Codes	Die Tastatur liefert typischer weise nur Byte – Werte, welche mit einer Fortlaufenden Nummer bezeichnen welche Taste gedrückt wurde. Diese nennt an Scan-Codes
Code-Page	Um Scan-Codes in ASCII Zeichen umzuwandeln werden sogenannte Codeseiten Benötigt. Diese sind Tabellen, welche die Scan-Codes zu ASCII Zeichen übersetzten, und je nach Tastatur und Land verschieden sind.
Host	68HC11 mit Tastaturtreiber

1.4 Verweise

- Software Configuration Management Plan (SCMP)
- Software Project Management Plan (SPMP)
- Software Design Document (SDD)

1.5 Überblick

Das SRS ist gemäss dem IEEE Standart organisiert. (Siehe Inhaltsverzeichnis)

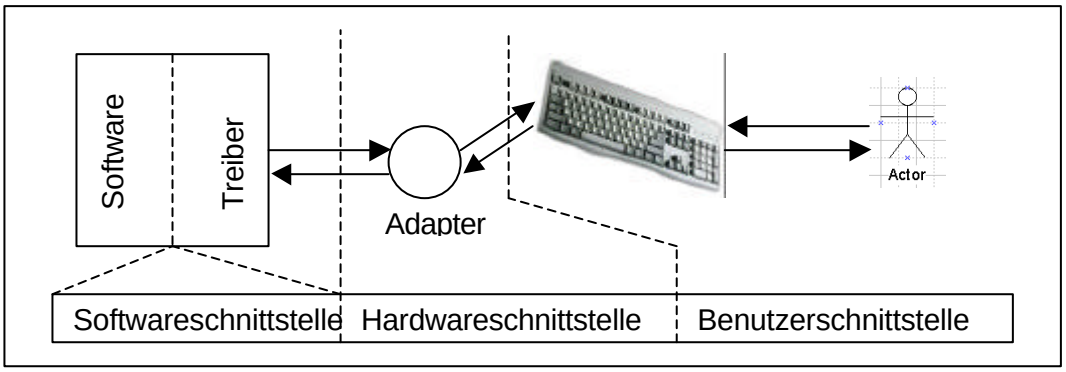
2. Allgemeine Beschreibung (C-Requirements)

Unser Treiber soll die Tastatur Schnittstelle IBM-PS/2 für den 68HC11 implementieren und Anwendungen einen einfachen API zur Verfügung stellen. Es sollen Funktionen zur Verfügung stehen welche ASCII-Zeichen zurückliefern, aber auch die direkte Kommunikation mit dem PS/2-Device erlauben. Weiter sollen Funktionen für das setzten der Tastatur-LEDs, das Zurücksetzten und das Testen der Tastatur vorhanden sein.

2.1 Produkt Perspektive

Unser Treiber richtet sich an Entwickler welche mit dem Motorola Mikrocontroller 68HC11 arbeiten und für ihr Projekt ein komfortables, alpha-numerisches Eingabegerät benötigen. Es sind keine Kenntnisse über die Kommunikation mit der Tastatur vonnöten um die grundlegenden Funktionen der Tastatur zu nutzen. Mit tieferen Kenntnissen des Übertragungsprotokolls ist es möglich der volle Umfang der Tastatur zu nutzen.
Es ist ein einfaches Beispiel vorhanden, dass die Funktionen demonstriert, die Einbindung des Treibers aufzeigt, und die Funktion der Angeschlossenen Tastatur verifiziert.
Der Treiber ist in C geschrieben und für die Metrowerks CodeWarrior Umgebung konzipiert.

2.2 Schnittstellen



2.2.1 System Schnittstellen
Unser Treiber beihaltet keine Systemschnittstellen.

2.2.2 Benutzer Schnittstellen
Die Benutzerschnittstelle gestaltet sich aus einer Handelsüblichen 101-104 Tasten Tastatur.
Diese erlaubt es dem Benutzer eine Taste zu drücken und den Zustand der Tastatur über 3 LEDs abzulesen.

2.2.3 Hardware Schnittstellen
Unser Produkt gestaltet die Anbindung der Tastatur möglichste einfach. So stellt es einen weiblichen Mini Din 6 Pol zur Verfügung an welchen eine Tastatur angeschlossen werden kann. Dieser befindet auf einem Modul das einfach auf dem 68HC11 Entwicklungssystem Meduse Trainer eingesetzt werden kann. Das Modul wird mittels Kabelverbindungen an den Mikrocontroller, Vcc und GND verbunden.

2.2.4 Software Schnittstellen
Um den Tastatur Treiber in einem Programm zu Nutzen stellen wir ein Header-File ‚ps2drv.h' zur Verfügung.
Dies Definiert die folgenden Funktionen, welche High-Level und 2 Low-Level Funktionen beinhalten:

High-Level Funktionen sind:

- getch - Wartet bis eine Taste gedrückt wurde und liefert dessen ASCII-Code
- getScanCode - Wartet bis eine Taste gedrückt wurde und liefert dessen Scancode
- setLED - Setzt die LEDs der Tastatur
- testKB - Testet die Kommunikation mit der Tastatur
- reset - Setzt die Tastatur zurück

Low-Level Funktionen sind:

- read - Liest ein Zeichen von der Tastatur
- write - Sendet der Tastatur ein Zeichen

Blockierend/nicht Blockierend
‚getch', ‚getScanCode' und ‚read' sind Blockierend, d.h. es wird gewartet bis ein Zeichen im Buffer der Tastatur ist und übertragen wird.
Alle Funktionen blockieren, wenn keine Tastatur angeschlossen ist.

2.2.5 Kommunikations- Schnittstellen
Gegen aussen sind keine Kommunikationsschnittstellen verfügbar.

2.2.6 Vorausgesetzte Ressourcen
Es wird keine besondere Anforderung an den Ressourcenverbrauch gestellt.

2.2.7 Adaptions-Anforderungen vor Ort
Das Hardwareinterface sollte mit möglichst wenig Aufwand an den 68HC11, speziell den Medusa-Trainer angeschlossen werden können.

2.3 Benutzer Charakteristiken
Das C-Benutzerinterface ist einfach zu Implementieren, es müssen nicht alle Funktionen über High-Level Funktionen zur Verfügung gestellt werden. Bei speziellen Anwendungen ist es jedoch möglich, der volle Funktionsumfang des angeschlossenen Gerätes zu nutzen. Dies benötigt tieferes Verständnis über die Kommunikation mit der Tastatur.

2.4 Vorraussetzungen
Das System ist für den Medusa Trainer konzipiert, für andere Entwicklungsumgebungen sind Anpassungen nötig.
Weiter werden 2 freie Port-Pins, und eine Speisung von 5V 300mA vorausgesetzt.
Der Keyboard Driver funktioniert nur mit AT und PS/2 Tastaturen. XT- und USB-Tastaturen werden nicht unterstützt.

2.5 Verteilung von Anforderungen
Sämtliche Anforderungen werden in Version 1.0 gefordert. Weiter Versionen sind nicht vorgesehen.

3. Konkrete Anforderungen (D-Requirements)
Dieser Teil des SRS sollte alle Software Anforderungen enthalten. Dies muss in einem genügend detaillierten Grad sein, um den Designern zu ermöglichen ein System zu entwerfen, welches diesen Anforderungen entspricht, und damit die Tester das System entsprechend dieser Anforderungen testen können.

3.1 Äussere Schnittstellen

3.1.1 Benutzer Schnittstelle
Die Benutzerschnittstelle muss sich aus einer MF-II kompatibler Tastatur mit Amerikanischem Tastaturlayout zusammensetzen. Letzteres ist für das korrekte Funktionieren der getch Funktion nötig, da sonst z.b. y und z vertauscht sind. Dies kann jedoch leicht im Treiber geändert werden.

3.1.2 Hardware Schnittstelle
Die Anbindung an den 68HC gestaltet sich nach folgendem Schema:

3.1.3 Software Schnittstelle

Vom Keyboard lesen

unsigned char getch ();
Diese Funktion liest mittels ‚getScanCode' ein Zeichen ein und übersetzt es anschliessend in ein ASCII Zeichen. Die Übersetzung erfolgt mittels des Arrays ‚scancodes' dieses ist unter 3.2.1 genauer beschrieben. Diese Funktion ist Blockierend.

unsigned char getScanCode();
Die Funktion liest mittels ‚read' das nächste Zeichen im Tastaturpuffer. Das Zeichen wird im Tastaturpuffer gelöscht. Liefert ‚read' 0x00, d.h. einen Fehler sendet die Tastatur einen Befehl dass die Tastatur das letzte Zeichen nochmals sendet (0xFE). Wird 10mal einen Fehler detektiert bricht die Funktion ab und gibt den letzten Wert von ‚read' zurück.

unsigned char read();
Low-Level Funktion welche das nächste Zeichen im Tastaturpuffer zurück gibt. Zuerst wird der Host auf bereit gesetzt mit Clock = 1, dann das Byte Empfangen. Am Ende wird die Parity (odd) überprüft und der Host wieder auf ‚nicht bereit' gesetzt mit Clock = 0. So puffert die Tastatur zu sendende Bytes.

Zum Keyboard schreiben

void setLED(varchar code);
Diese Funktion setzt die LED auf der Tastatur. Der Code zum Setzen der LED sieht folgendermassen aus:

MSb LSb

Always 0	Always 0	Always 0	Always 0	Always 0	Caps Lock	Num Lock	Scroll Lock

void write(unsigned char);
Diese Low-Level Funktion sendet ein Byte zur Tastatur. Zuerst wird signalisiert dass der Host senden will, dann das Byte übertragen und am Ende der Host auf nicht bereit gesetzt

Test- und Initialisierungsfunktionen

unsigned char testKB();
Testet die Funktion der Tastatur mit dem ‚Echo' (0xEE) Befehl. Zuerst wird ein Echo gesendet kommt dieses zurück liefert die Funktion 0x00 zurück sonst einen Wert ungleich 0x00.

void reset();
Setzt die Tastatur mittels Reset Befehl (0xFF) zurück. Man sollte das kurze aufleuchten der Dioden bemerken.

void init();
Initialisiert den Tastatur Treiber in dem der Host auf ‚nicht bereit' (Clock = 0) gesetzt wird. Dies bewirkt, dass die Tastatur alle Zeichen Buffert.

unsigned char extededInit();
Initialisiert den Tastaturtreiber und testet die Tastatur mit ‚testKB'. Wird 0x00 zurückgegeben war die Initialisierung erfolgreich, sonst trat ein Fehler auf.

3.2 Interne Schnittstellen

3.2.1 Kommunikationsschnittstelle
Die Kommunikationsschnittstelle zwischen Mikrocontroller und Tastatur ist ein Serielles Übertragungsprotokoll. Es ist Bidirektional und besitzt 2 Leitungen. Eine Clock und eine Datenleitung. Der Clock wird immer vom PS/2 Device generiert. Näheres zum Protokoll findet man in den Ressource die im Hauptdokument und auf unser Homepage unter Ressourcen vermerkt sind.

3.2.2 Datenschnittstelle
Um die Umwandlung vom ScanCode in den ASCII-Wert zu Realisieren ist ein Array mit 256 Byte Implementiert. Jede Position repräsentiert ein ScanCode, welcher das ASCII Zeichen oder 0 enthält.

3.3 Performance Anforderungen
Es wird keine besondere Anforderung an den Ressourcenverbrauch gestellt.

3.4 Vorraussetzungen Design
Das Design soll aus Zeitgründen möglichst einfach gehalten werden.

3.5 Software System Eigenschaften

3.5.1 Zuverlässigkeit
Der Erstellte Treiber hat bis zum Abgabe Termin Alpha Status erreicht. Die Funktionalität muss gewährleistet sein, ohne jedoch eine Zuverlässigkeit in allen Fällen garantieren zu können. Weiter kann nicht die Stabilität des Systems nicht garantiert werden.

3.5.2 Sicherheit
Der Treiber implementiert keine Mechanismen um das Überschreiben von Speicherbereichen oder ähnlichen Fehlern, welche die Stabilität oder Sicherheit der Anwendung Garantieren.

3.5.3 Wartbarkeit
Der Treiber ist einfach zu Warten, Erweitern und Bugfixen sein.

3.5.4 Portierbarkeit
Der Treiber ist in C geschrieben, somit ist eine Portierung auf andere Systeme leicht möglich aber nicht Garantiert.