Hochschule Bremerhaven

Embedded Systems

Software Test Documentation

Description of the plan and specifications to verify and validate the software and the results.

Autoren: Jan Löwenstrom (34937) und Johann Hoffer (34461)
begleitet von
Prof. Dr. Lipskoch

1. Februar 2019

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Introduction

1.1 System Overview

1.2 Test Approach

Um die Logik der Software unabhängig vom Mikrokontroller zu testen, wurde eine Projektstruktur gewählt, bei der der Hardware-Code, also Code der mittels des avr-gcc-Compilers compiliert werden muss und hardwarespezifische Teile aufweist wie z.B. Konstanten für PORTS, Interruptvektoren und Hardwarefunktionen (sleep_cpu(), _delay_us(), etc.), strikt vom restlichen Code abgekapselt. Mit diesem Aufbau lassen sich ohne Probleme einzelne Teile und Funktionen des Systems testen und letztendlich alle Unit-Tests mit dem einfachen gcc-Compiler compilieren und lokal auf jeder Maschine ausführen. Jeglicher Code der sich nicht im „/src/hardware“ befindet, ist auf diese Art testbar.

Dieses Dokument beinhaltet zwei Arten von Tests. Die einen, Softwaretests, werden alle automatisiert ausgeführt, näheres im Kapitel Testing Tools and Environment. Auf der anderen Seite wird auch ein Plan erstellt, um die verwendete Hardware auf mögliche Fehler zu überprüfen und gegebenenfalls auszutauschen.

2 Test Plan

2.1 Features to be Tested

Sämtliche Softwarekomponenten, die eine Rolle bei der Verarbeitung des DCF-Signals und dessen Umwandlung in ein valides Datum mit Uhrzeit spielen, werden mit Hilfe von Unit-Tests getestet. Es ist anzumerken, dass die meisten Komponenten testgetrieben erstellt worden sind. Die Komponenten, die Teil zur Umsetzung der Anforderungen REQ-001, REQ-002, REQ-003 und REQ-004 sind und getestet werden sind:

- Das Extrahieren und Konvertieren der DCF-Reihenfolge in Informationen zu Datum und Uhrzeit
- Die Überprüfung zur Echtheit eines Datums
- Der Paritätscheck zur Validierung der DCF-Reihenfolge
- Das Hochzählen der internen Zeit mit korrektem Übergang zwischen Stunden, Tagen, Jahren, etc.
- Das Bereitstellen von Daten, die an das Display gesendet werden sollen

Zudem wird auch die Hardware, die zum Einsatz kommt mit Test Cases abgedeckt. Wichtig ist hierbei die ungestörte Kommunikation zwischen dem Mikrocontroller und dem Display bzw. dem DCF-Empfangsmodul. Der Fokus liegt also auf der Überprüfung der fehlerfreien Leiterbahnen und den Schnittstellen zwischen den den Hardwarekomponenten. Dabei werden folgende Features indirekt getestet:

- Das Übertragen der Displayanweisungen vom Eva-Board zum Display
- Der Zugang zu dem abgegebenen, modularisiertem Datensignal des DCF-Empfängermoduls
- Das Aus- und Einschalten des Display mit Hilfe des Tasters auf dem Eva-Board
- Die ungestörte Anzeige auf dem Telefondisplay

2.2 Features not to be Tested

Alle Features die zu den Anforderungen REQ-005 bis REQ-011 gehören, werden nicht getestet, da diese nicht verwirklicht wurden.

Es existieren keine „Integration-Tests“ in dem Sinne, dass die Schnittstelle zur Übertragung von Daten und Anweisungen an das Display nicht getestet werden. Getestet wird lediglich, ob Daten überhaupt gesendet werden sollen, aber nicht, ob gewisse Anweisungen zu bestimmten Zeitpunkten gesendet werden und dabei das vorgegebene Timing, welches das Display vorgibt, eingehalten wird. Indirekt kann man dies jedoch über die, vom Simulator ausgegebene, .vcd-Datei manuell überprüfen.

Des Weiteren wird die Genauigkeit der Interrupts und Timer nicht kontrolliert.

Es wird zudem vorausgesetzt, dass das DCF-Empfängermodul und das Telefondisplay keine internen Fehlerquellen besitzen. Eine separate Prüfung der externen Hardwaremodule und des ATMega32 findet nicht statt. Lediglich die Schnittstellen können auf Fehler überprüft werden.

2.3 Testing Tools and Environment

Für das Testing wurde reiner C-Code verwendet ohne der Verwendung eines Frameworks. Es wurde ein kleines eigenes Helper-Framework erstellt, welches die Ausgabe erleichter.
Wie schon angesprochen zählen auch die Programme simavr und GTK-Wave zu den Testing-Tools, da sie es ermöglichen ohne den eigentlichen Mikrokontroller zu entwickeln.

3 Test Cases

3.1 Case-1: DateExtractor

Purpose

Dieser Test prüft die Funktionalität des DateExtractors. Damit am Ende die richtige Zeit und korrekten Daten angezeigt werden können, muss das Dekodieren der DCF-Folge vollkommen fehlerlos ablaufen.

Inputs

0-**********-00100-1-1001101-0-110001-1-101010-100-00001-00011000-1

Expected Outputs and Pass/Fail criteria

Minuten: 59
Stunden: 23
Tag: 15
Wochentag: 1
Monat: 10
Jahr: 18

Der Test ist erfolgreich, falls die aufgeführten Werte aus der DCF-Folge herausgelesen werden. Sobald ein Wert nicht dem erwarteten Ergebnis entspricht, schlägt der Test fehl.

Test Procedure

Der Test läuft so ab, dass Werte für Minuten, Stunden, Tag, Monat, Jahr und Wochentag in die DCF-Folge geschrieben und anschließend ausgewertet werden.

3.2 Case-2: Paritätschecker

Purpose

Der Paritätschecker ist von hoher Bedeutung, da über falsche Paritätsbits eine fehlerhafte Übertragung und dementsprechend auch eine fehlerhafte DCF-Folge identifiziert werden kann. Falls mindestens ein inkorrektes Paritätsbit ermittelt wird, kann die entsprechende DCF-Folge verworfen werden. Im Endeffekt wird dadurch verhindert, dass die interne Zeit mit verkehrten Daten synchronisiert wird.

Inputs

0-**********-00100-1-1001101-0-110001-0-101010-100-00001-00011000-1

Expected Outputs and Pass/Fail criteria

Parität Minute: korrekt
Parität Stunde: falsch
Parität Datum: korrekt

Der Test ist erfolgreich, falls die Korrektheit aller Paritäten richtig festgestellt worden sind und diese DCF-Folge verworfen wird, da die Parität der Stunde falsch ist.

Test Procedure

Der Test läuft so ab, dass das rawDCF-Array auf den festgelegten Input gesetzt wird. Der Paritätschecker rechnet dann die jeweiligen Paritäten aus und vergleicht diese mit den Paritätsbits. Falls sich die errechneten Werte von den Paritätsbits unterscheidet, muss der Paritätschecker dies als Fehler erkennen.

3.3 Case-3: Datenvalidierung

Purpose

Die Datenvalidierung wird für zwei Funktionalitäten benötigt. Die erste davon findet nach einem erfolgreichen Paritätscheck statt. Eine DCF-Folge kann diesen Test bestehen, auch wenn sie inkorrekt ist. Deshalb muss sie zusätzlich einen weiteren Test bestehen, der prüft, ob Datum und Zeit valide sind.

Zudem wird die Validierung bei jedem Hochzählen der internen Zeit benötigt. Werden beispielsweise die Sekunden auf 60 gesetzt, schlägt die Validierung fehl. Ist dies der Fall werden die Sekunden auf 0 gesetzt, die Minuten einmal inkrementiert und es findet erneut eine Validierung statt. Ist diese erfolgreich, ist das Weiterzählen der internen Zeit abgeschlossen. Wenn dies jedoch nicht der Fall ist, muss dieses Verfahren für die nächst größere Zeiteinheit angewendet werden bis schließlich das Jahr inkrementiert wird.

Die Validierung muss vollkommen korrekt ablaufen, da das Hochzählen immer richtig funktionieren muss.

Inputs

Folgendes Datum und folgende Uhrzeit werden geprüft:

Minute: 54
Stunde: 12
Tag: 29
Monat: 2
Jahr: 19

Expected Outputs and Pass/Fail criteria

nicht valide, da Monat = 2, kein Schaltjahr und Tag = 29

Der Test ist erfolgreich, falls die Korrektheit des Datums richtig erkannt wird. In diesem Fall müsste ein Fehler erkannt werden.

Test Procedure

Der Test läuft so ab, dass den Validierungsmethoden die Zeit und das Datum übergeben wird. Diese Methoden prüfen, ob die Zeit eine gültige Uhrzeit ist und die einzelnen Werte des Datums zueinander passen. Die Zeit ist immer valide, solange die Stunden kleiner als 24 und die Minutn kleiner als 60 sind. Beim Datum hängt die meist vom Monat ab, je nachdem ob dieser 30 oder 31 Tage hat. Beim Februar können es 28 oder 29 Tage sein, je nachdem ob das getestete Jahr ein Schaltjahr ist. Dies muss dann zusätzlich ausgerechnet werden.

3.4 Case-4: Weiterzählen der internen Zeit

Purpose

Dieser Test prüft die Funktionalität des Weiterzählens der internen Zeit. Dies wird jede Sekunde ausgeführt und ohne das Weiterzählen könnte keine sekündlich korrekte Zeit angezeigt werden.

Inputs

Im Struct der internen Zeit werden folgende Werte gesetzt.
seconds = 53
minutes = 59
hours = 23
year-tens = 19
year-hundreds = 20
weekday = 4
month = 1

day = 31
Danach wird die Zeit 20-Mal hochgezählt.

Expected Outputs and Pass/Fail criteria

seconds = 13
minutes = 0
hours = 0
year-tens = 19
year-hundreds = 20
weekday = 5
month = 2
day = 1

Der Test ist erfolgreich, wenn nach n Inkrementierungsvorgängen die Zeit um n Sekunden weitergezählt wurde. Es muss also der Input+n Sekunden als Output ausgegeben werden, in diesem Fall wären n = 20.

Test Procedure

Zunächst wird die interne Zeit auf bestimmte Werte gesetzt. Dann werden beliebig viele Inkrementierungen durchgeführt und geprüft, ob die einzelnen Werte der internen Zeit gleich den erwarteten Werten sind.

3.5 Case-5: Display Instructions

Purpose

Dieser Test prüft die Display Instructions nach deren Länge. Die Instructions senden die Daten zum Display und sind somit die Grundlage dafür, dass die Anzeige korrekt funktionieren kann. Es wurde vorher festgelegt, welche Daten in welcher Zeile auf dem Display zu finden sein sollen. Ist die Länge richtig, kann davon ausgegangen werden, dass die Daten an den passenden Ort gelangen.

Inputs

Im Struct der internen Zeit werden folgende Werte gesetzt.
seconds = 06
minutes = 45
hours = 12
year-tens = 19

year-hundreds = 20
weekday = 1
month = 1
days = 07

Expected Outputs and Pass/Fail criteria

Die Anzahl der zu sendenden Spalten in einer Zeile des Displays muss der Anzahl entsprechen, die vorher festgelegt wurde. Für jedes Zeichen werden 6 Spalten benötigt. In Zeile 0 müssen es 48 sein, da die Uhrzeit aus sechs Ziffern und zwei Doppelpunkten besteht. In Zeile 1 sind es 60, da das Datum 8 Ziffern und 2 Punkte hat. In Zeile 2 steht der Wochentag als Abkürzung in Form von zwei Buchstaben, sodass 12 Pixelspalten benutzt werden müssen.

Test Procedure

Für jeden Subtest wird die interne Zeit erneut frisch erzeugt. Danach werden jeweils die „setInstructionsForRow(X)“ aufgerufen. Diese Funktion lädt alle Daten-Arrays, welche für einzelne Buchstaben stehen, in das globale Array „display_data“. Aufgrund des Inhalts dieses Arrays, können anschließend Rückschlüsse auf die Korrektheit der gesetzten Bits gezogen werden.
Ein Subtest simuliert zudem den Output, der auf dem Display zu sehen wird durch Iteration durch das globale Array. Stößt die Schleife auf eine 1, so printet sie ein 'X' aus, im Fall von 0 ein Leerzeichen. Das Ausgabeergebnis und die Beurteilung, ob das Angezeigte korrekt ist, kann man durch einen Blick auf den Test-Output beurteilen.

3.6 Case-6: Display

Purpose

Dieser Test prüft, ob auf dem Display die richtige und das korrekte Datum angezeigt wird.

Inputs

Im Struct der internen Zeit werden am Anfang des Programmablaufs folgende Werte gesetzt:
seconds = 06
minutes = 45
hours = 12
year-tens = 19

year-hundreds = 20
weekday = 1
month = 1
days = 07

Expected Outputs and Pass/Fail criteria

Das Anzeigen auf dem Display funktioniert, wenn zunächst die oben gennanten Werte angezeigt werden, und diese jede Sekunde korrekt weitergezählt werden.

Test Procedure

Es wird kein besonderer Testablauf gestartet, sondern die normale Software, wie sie bei Fertigstellung des Systems zum Einsatz kommt. Lediglich zu Beginn wird der gewünschte Input im Struct gespeichert.

3.7 Case-7 Evaluationsboard prüfen

Purpose

Um beim Fehlerfall auszuschließen, dass das Evaluationsboard einen Defekt hat, ist es angebracht, die Pins auf dem Wannenstecker zu kontrollieren. Des Weiteren sollte man kontrollieren, ob der Jumper JP1 entfernt wurde, da dieser sonst mit einem intern genutzten PIN im PORTD interferiert.

Inputs

Die gemessene Spannung auf jedem Pin des Wannensteckers.
Ein Blick auf das Jumper-Setup.

Expected Outputs and Pass/Fail criteria

Jeder Pin, der auf der Abbildung unten mit einem roten Punkt markiert ist, sollte eine Spannung von 5V aufweisen. Der Test schlägt fehl, sobald mindestens ein Pin nicht 5V aufbringt. Zudem ist der JP1 nicht aktiv.

Test Procedure

Zunächst einmal spielt man das Pin-Testprogramm auf den ATMega32. Dies geschieht mit den beiden Kommandos „make pintest“und „make firmtest“. Anschließend sollte auf jedem Pin, der in der Software gebraucht wird eine permanente Spannung von 5V liegen. Mit Hilfe eines Voltmeter geht man nun jeden einzelnen Pin durch, der auf der

Abbildung mit einem rotem Kreis markiert ist (GND liegt auf dem PIN, der mit dem schwarzen Kreis markiert ist):

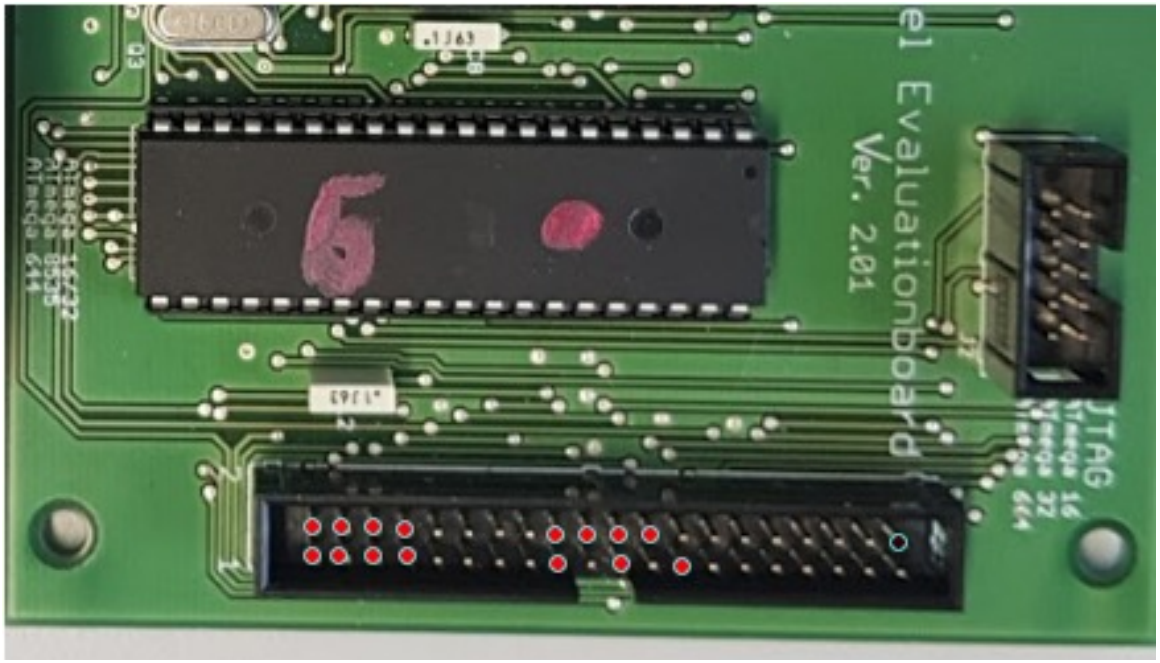

Abbildung 1: Prüfstellen auf dem Evaluationsboard

Ist man damit fertig, so muss noch sichergestellt werden, dass der JP1 nicht angeschlossen ist.

Abbildung 2: Jumpersetting auf dem Evaluationsboard

3.8 Case-8 Löt-Board prüfen

Purpose

Um beim Fehlerfall auszuschließen, dass das Löt-Board einen Defekt hat, ist es notwendig dieses zu testen. Dabei sollten alle Leiterbahnen überprüft werden und vor allem die Display-Pins. Das Löt-Board ist selbst zusammengelötet worden und „offen“, wodurch davon auszugehen ist, dass es schnell abnutzt.

Inputs

Die gemessene Spannung auf jedem Display-Pin. Die gemessenen Widerstände zwischen zwei Punkten.

Expected Outputs and Pass/Fail criteria

Jeder Display-Pin sollte eine Spannung von 5V aufweisen, solange das „pinTest-Programm“ läuft. Der Test failed, sobald mindestens ein Pin nicht 5V aufbringen oder halten kann. Es dürfen keine Kurzschlüsse auf dem Löt-Board zu identifizieren sein.

Test Procedure

Zunächst einmal spielt man das Pin-Testprogramm auf den ATMega32. Dies geschieht mit den beiden Kommandos „make pintest“ und „make firmtest“. Anschließend sollte auf jedem Pin, der in der Software gebraucht wird, eine permanente Spannung von 5V liegen. Mit Hilfe eines Voltmeters geht man nun jeden einzelnen Display-Pin auf dem Löt-Board durch und misst die anliegende Spannung.

Um das Löt-Board auf Kurzschlüsse zu überprüfen, testet man jeweils zwei nebeneinander liegende Kupferleiterbahnen. Diese dürfen beim Multimeter keinen Ausschlag oder Ton erzeugen. Zudem sollte überprüft werden, ob alle Widerstände korrekt genutzt werden oder ob diese, wie auch immer, überbrückt werden.

4 Test Plan

Test	Uhrzeit	Zeit in Min	Prüfer	Ergebnis	Anmerkung
Automatisierte Tests ausführen (make test)					
Evaluationsboard prüfen (Pins und Jumper)					
Löt-Board prüfen (Pins und Leiterbahnen)					