Testspezifikation Sofware-Watchdog

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	18.01.2011
Quelle	Google Code → Dokumente → 04_Test → 04.01_Testspezifikation
Autoren	Norman Nieß
	Kai Dziembala
Version	1.0
Status	freigegeben

1 Historie

Version	Datum	Autor	Bemerkung
0.0	20.05.2010	Kai Dziembala Norman Nieß	Initialisierung der Testspezifikation
0.1	14.06.2010	Kai Dziembala Norman Nieß	Anpassungen an Review: • 'helloModul(module_id, status)' im gesamten Dokument durch 'helloModul(byte module_id, byte status)' ersetzt • Umformulierung in Kapitel 9.3 • neuer Dokumentenstatus: freigegeben
1.0	17.06.2010	Kai Dziembala	Freigabe der Testspezifikation 'SW-Watchdog'

2 Inhaltsverzeichnis

3 Testziele

Der Test des Software-Moduls 'SW-Watchdog' soll sicherstellen, dass Zeitüberschreitungen bei Modulabarbeitung oder fehlerhafte Modulstatusmeldungen zur Auslösung eines Not-Aus-Signals führen. Des weiteren wird geprüft, ob sich der SW-Timer zurücksetzten und auch deaktivieren lässt. Dies dient dem Gesamtziel, die Fahraufgabe gemäß Pflichtenheft (Kapitel 6) auszuführen.

4 Testfall 1 „Zurücksetzen und Stoppen“

4.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)
- SoftwareWatchdogStop.h (Version 1.6, Repository 181)

4.2 Test-Identifikation

Testname: Test_Zurücksetzten-Stoppen

Verzeichnisse

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

4.3 Testfallbeschreibung

Es wird getestet, ob der Timer durch die Funktion 'hello()' zurückgesetzt wird. Desweiteren wird überprüft, ob sich der Timer durch die Funktion 'stopSW()' abschalten lässt.

4.4 Testskript

Zunächst wird das Modul 'SW-Watchdog' initialisiert. Anschließend wird in einer for-Schleife mit drei Durchläufen die folgende Testsequenz, zur wiederholten Überprüfung des Testfalls, durchgeführt. Zuerst wird der Timer '0' gestartet und die Funktion 'hello()' aufgerufen. Über den zweiten Timer '1' wird nun 15ms gewartet und dann erneut 'hello()' aufgerufen. Im Anschluss wird 10ms gewartet, wobei kein Aufruf der Funktion 'emergency_off()' erfolgen darf. Anschließend bewirkt der Funktionsaufruf 'stopSW()', dass der Timer '0' gestoppt wird und somit in einer erneuten Wartezeit von 25ms kein 'emergency_off' Aufruf erfolgt darf. Nach Beendigung der for-Schleife wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog

4.5 Testreferenz

Die Funktion 'emergency_off()' darf während des Testfalls zu keiner Zeit aufgerufen werden.

4.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

5 Testfall 2 „Fehlerfreie Statusmeldungen“

5.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)
- SoftwareWatchdogStop.h (Version 1.6, Repository 181)
- SoftwareWatchdogHelloModul.h (Version 1.6, Repository 181)

5.2 Test-Identifikation

Testname:	Test_fehlerfreierAufruf_'helloModul'
Verzeichnisse	
Testskripts:	Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog
Testprotokolle:	Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

5.3 Testfallbeschreibung

Es wird getestet, dass die Schnittstelle 'helloModul(byte module_id, byte status)', bei korrekter Verwendung (definierte Modul-ID und sich nicht wiederholende Statusmeldungen), kein Not-Aus-Signal auslöst.

5.4 Testskript

Zunächst wird das Modul 'SW-Watchdog' initialisiert. Anschließend wird in einer for-Schleife mit drei Durchläufen die folgende Testsequenz, zur wiederholten Überprüfung des Testfalls, durchgeführt. Zuerst wird der Timer '0' des SW-Watchdogs mittels der Funktion 'stopSW()' ausgeschaltet. Anschließend wird in einer weiteren for-Schleife mehrfach die Funktion 'helloModul(byte module_id, byte status)' mit fester Modul-ID (=1) und dem Schleifenindex als Statusmeldung aufgerufen. Es darf kein Not-Aus-Signal ausgelöst werden. Nach Beendigung der for-Schleife wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog

5.5 Testreferenz

Während des Testdurchlaufs darf die Funktion 'emergency_off()' nicht ausgelöst werden.

5.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

6 Testfall 3 „fehlerhafte Modul-ID“

6.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)
- SoftwareWatchdogStop.h (Version 1.6, Repository 181)
- SoftwareWatchdogHelloModul.h (Version 1.6, Repository 181)

6.2 Test-Identifikation

Testname: Test_fehlerhafteModul-ID

Verzeichnisse

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

6.3 Testfallbeschreibung

Die Funktion 'helloModul(byte module_id, byte status)' wird mehrmals mit einer fehlerhaften Modul-ID aufgerufen, wodurch jeweils ein Not-Aus-Signal ausgelöst werden muss.

6.4 Testskript

Der Hauptteil besteht aus einer for-Schleife mit drei Durchläufen, in der die folgende Testsequenz zur wiederholten Überprüfung des Testfalls durchgeführt wird. Das Modul 'SW-Watchdog' wird initialisiert und im Anschluss bewirkt die Funktion 'stopSW()', dass der Timer '0' ausgeschaltet wird. Danach erfolgt der Funktionsaufruf 'helloModul(byte module_id, byte status)' mit der Modul-ID 'Schleifenindex + 10' und dem Status '1'. In jedem Schleifendurchlauf muss dieser Funktionsaufruf aufgrund der undefinierten Modul-ID ein Not-Aus-Signal auslösen (Aufruf der 'emergency_off()'-Funktion). Nachdem die drei Schleifendurchläufe beendet sind, wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog'

6.5 Testreferenz

Die Funktion 'emergency_off()' muss während jedem Schleifendurchlauf aufgerufen werden.

6.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

7 Testfall 4 „fehlerhafter Status“

7.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)
- SoftwareWatchdogStop.h (Version 1.6, Repository 181)
- SoftwareWatchdogHelloModul.h (Version 1.6, Repository 181)

7.2 Test-Identifikation

Testname: Test_fehlerhafterStatus

Verzeichnisse

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

7.3 Testfallbeschreibung

Die Funktion 'helloModul(byte module_id, byte status)' wird mehrmals mit dem Status '0xFF' aufgerufen, wodurch jeweils ein Not-Aus-Signal ausgelöst werden muss.

7.4 Testskript

Der Hauptteil besteht aus einer for-Schleife mit drei Durchläufen, in der die folgende Testsequenz, zur wiederholten Überprüfung des Testfalls, durchgeführt wird. Das Modul 'SW-Watchdog' wird initialisiert und im Anschluss bewirkt die Fuktion 'stopSW()', dass der Timer '0' ausgeschaltet wird. Danach erfolgt der Funktionsaufruf 'helloModul(byte module_id, byte status)' mit dem Schleifenindex als Modul-ID. Der Status hat den Wert 0xFF, wodurch die Funktion 'emergency_off()' ausgelöst werden muss. Nach dreimaligem Durchführen des Testfalls wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog'

7.5 Testreferenz

Die Funktion 'emergency_off()' muss während jedem Schleifendurchlauf aufgerufen werden.

7.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

8 Testfall 5 „ Wiederholter Status“

8.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)
- SoftwareWatchdogStop.h (Version 1.6, Repository 181)
- SoftwareWatchdogHelloModul.h (Version 1.6, Repository 181)

8.2 Test-Identifikation

Testname: Test_Statuswiederholung

Verzeichnisse

Testskripts: Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog

Testprotokolle: Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

8.3 Testfallbeschreibung

Es wird getestet, ob das Senden von aufeinanderfolgenden identischen Statusmeldungen bei einer gleichen Modul-ID ein Not-Aus-Signal auslöst.

8.4 Testskript

Zunächst wird das Modul 'SW-Watchdog' initialisiert und der Timer '0' durch den Aufruf der Funktion 'stopSW()' ausgeschaltet. Anschließend wird dem SW-Watchdog über die Funktion 'helloModul(byte module_id, byte status)' eine Modulstatusmeldung mit der Modul-ID „1“ und dem Status „2“ übermittelt. Danach wird zur Mehrfachüberprüfung in jedem Schleifendurchlauf, einer dreistufigen for-Schleife, die Funktion 'helloModul(byte module_id, byte status)' mit der Modul-ID „1“ und dem Status „2“ aufgerufen. Dies muss jeweils zum Aufruf der Funktion 'emergency_off()' führen. Nach Beendigung der for-Schleife wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog

8.5 Testreferenz

Die Funktion 'emergency_off()' muss während jedem Schleifendurchlauf aufgerufen werden.

8.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

9 Testfall 6 „Zeitüberschreitung“

9.1 Identifikation des Testobjektes

Es wird der Programmcode zum Modul 'Software-Watchdog' getestet:

- SoftwareWatchdog.c (Version 2.0, Repository 181)
- SoftwareWatchdog.h (Version 1.6, Repository 181)

9.2 Test-Identifikation

Testname:	Test_Zeitüberschreitung
Verzeichnisse	
Testskripts:	Google Code → Dokumente → 04_Tests → 04.02_Testskript → 04.02.10_SW-Watchdog
Testprotokolle:	Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog

9.3 Testfallbeschreibung

Es wird getestet, ob eine Zeitüberschreitung (ausbleibendes Rücksetzten des SW-Watchdog-Timers) ein Not-Aus-Signal auslöst.

9.4 Testskript

Zunächst wird das Modul 'SW-Watchdog' initialisiert. Anschließend wird in einer for-Schleife mit drei Durchläufen die folgende Testsequenz, zur wiederholten Überprüfung des Testfalls, abgearbeitet. Ein Aufruf der Funktion 'hello()' setzt den Timer '0' des SW-Watchdogs definiert zurück. Dann wird mittels des zweiten Timers '1' 25ms gewartet. Vor Ablauf dieser Zeit muss ein Not-Aus-Signal ausgelöst wurden sein. Nachdem die Testfälle dreimal durchlaufen sind, wird das Testergebnis in der Konsole ausgegeben.

Dies wird mit folgendem Test-Skript realisiert:

siehe 'Google Code → 04_Test → 04.02_Testskripts → 04.02.10_SW-Watchdog

9.5 Testreferenz

Die Funktion 'emergency_off()' muss während jedem Schleifendurchlauf nach der Wartezeit aufgerufen wurden sein.

9.6 Test-Protokoll

Das Konsolen-Ergebnis wird in das Dokument 'Protokoll_Test_SW-Watchdog' kopiert und diese Datei im Ordner 'Google Code → Dokumente → 04_Tests → 04.03_Testprotokolle → 04.03.10_SW-Watchdog' abgelegt.

10 Auswertung

wird nach Testdurchführung erstellt