

MOBA Mobile Automation AG

**Systemtest***SPI-Interrupt*

Version 2.000

|  |  |
| --- | --- |
| **Produkt** | **MRW 4-20mA**  (**M**omenten unabhängige **R**edundante **W**ägezelle) |
| **Auftraggeber** | **MOBA Mobile Automation AG**  Kapellenstraße 15  65555 Limburg  Germany |
| **Auftragnehmer** | **MOBA Mobile Automation AG**  Kapellenstraße 15  65555 Limburg  Germany |

|  |  |  |
| --- | --- | --- |
| Dokument erstellt von  M.Offenbach | Datum  28.04.2022 | Unterschrift |

Diese Dokumentation des Unittests basiert auf einem Vordruck der MOBA AG.

Der Inhalt darf ausschließlich den am Projekt beteiligten Personen zugängig gemacht werden.

Insbesondere die Weitergabe an Dritte ist ohne ausdrückliche schriftliche Erlaubnis der MOBA AG nicht erlaubt.

Außerhalb des gemeinsamen Projektes darf kein Teil dieser Unterlagen für irgendwelche Zwecke vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln dies geschieht.

Die hier getroffenen Festlegungen schließen nicht aus, dass in einer gesonderten Geheim­haltungsvereinbarung weiterreichende oder abweichende Vereinbarungen zur Wahrung der Ver­traulichkeit getroffen und festgeschrieben werden.

Copyright by

MOBAMobile Automation AG

Kapellenstr. 15

D-65555 Limburg

Internet: [www.moba.de](http://www.moba.de)

**Inhaltsverzeichnis**

1 Einführung 4

1.1 Vorwort 4

1.2 Änderungshistorie 4

1.3 Ansprechpartner 5

1.4 Anhänge 5

1.5 Glossar 5

2 Systemtest ‚SPI-Interrupt‘ 6

2.1 Interrupt-Funktion ‚ADuC836\_SPIInterrupt()‘ 6

2.1.1 Funktionsbeschreibung 6

2.1.2 Testbeschreibung 6

2.1.3 Testmittel 7

2.1.3.1 Firmware ‚VisionMRW 420 – Testing‘ – zu testende Firmware in der Debugversion mit zusätzlichen Testroutinen zum SPI-Interrupt 7

2.1.3.2 Entwicklungsumgebung ‚Keil V2‘ 7

2.1.3.3 Terminalsoftware ‚Docklight Scripting V2.3‘mit Projekt ‚MRW420\_V1.200 - Testing - IRQ.ptp‘ 7

2.1.3.4 Entwicklungsumgebung ‚MRW420‘ 8

2.1.3.5 MRW-Kommunikationsleitung 8

2.1.3.6 Adapter DB9 auf USB 8

2.1.4 Testablauf 8

2.1.5 Testergebnisse 10

2.1.6 Resultierendes Testergebnis 11

3 Kommentare 12

4 Anhang 13

# Einführung

## Vorwort

Die MOBA AG versteht sich als Partner für die Entwicklung und Lieferung kundenspezifischer Elektronikkomponenten und daraus zusammengestellter Steuerungssysteme, die für den Einsatz an mobilen Maschinen konzipiert sind.

Der hier vorliegend beschriebene Systemtest überprüft das exakte Verhalten der Funktionionaität der SPI-Interrupt, welche aufgrund von Kompatibilitätsgründen mit alten Firmware-Varianten von Nöten ist.

Dokumentiert ist zunächst das erwartete Verhalten der Firmware in Bezug auf die Eeprom-Reorganisation, gefolgt von der Auflistung der benötigten Testmittel und der Beschreibung des Testablaufs. Im anschließenden Teil finden sich die Testergebnisse in Bezug auf das geforderte Verhalten wieder.

## Änderungshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Kapitel** | **Änderung / Ergänzung** |
| 1.0 | 28.04.2022 | alle | Erstellung |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Ansprechpartner

**MOBA Mobile Automation AG**

Kapellenstraße 15

65555 Limburg

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Position** | **Telefonnummer** | **E-Mail** |
| Boris Zils | Produktmanager | +49(0)6431-9577-123 | [b.zils@moba.de](mailto:b.zils@moba.de) |
| Sebastian Schlesies | Vertrieb | +49(0)6431-9577-267 | [s.schlesies@moba.de](mailto:s.schlesies@moba.de) |
| Jürgen Stiller | Entwicklungsleiter | +49(0)6431-9577-282 | [j.stiller@moba.de](mailto:j.stiller@moba.de) |
| Norbert Lipowski | Entwicklung | +49(0)6431-9577-137 | [n.lipowski@moba.de](mailto:n.lipowski@moba.de) |

## Anhänge

|  |  |
| --- | --- |
| **Dokumentname** | **Beschreibung** |
|  |  |
|  |  |

## Glossar

|  |  |
| --- | --- |
| **Abkürzung / Fachbegriff** | **Beschreibung / Definition** |
| MRW | Momenten unabhängige Redundante Wägezelle |
| DMS | Dehnungsmessstreifen |

# Systemtest ‚SPI-Interrupt‘

## Interrupt-Funktion ‚ADuC836\_SPIInterrupt()‘

### Funktionsbeschreibung

Die Bibliotheksfunktion *ADuC836\_SPIInterrupt()* der ADuC836-Bibliothek dient dem Empfang von in *STX* und *ETX* eingerahmten Frames über Interrupt.  
Dabei muss zunächst ein *STX* empfangen worden sein, bevor weitere Daten Einzug in den Empfangspuffer nehmen können. Bleibt ein *ETX* aus, beginnt mit dem nächsten *STX* die Aufzeichnung. Frames, welche über den Empfangspuffer hinausreichen würden, werden nur bis zur Puffergrenze empfangen und dann verworfen.

### Testbeschreibung

Zu testen sind die Spezifikationen:

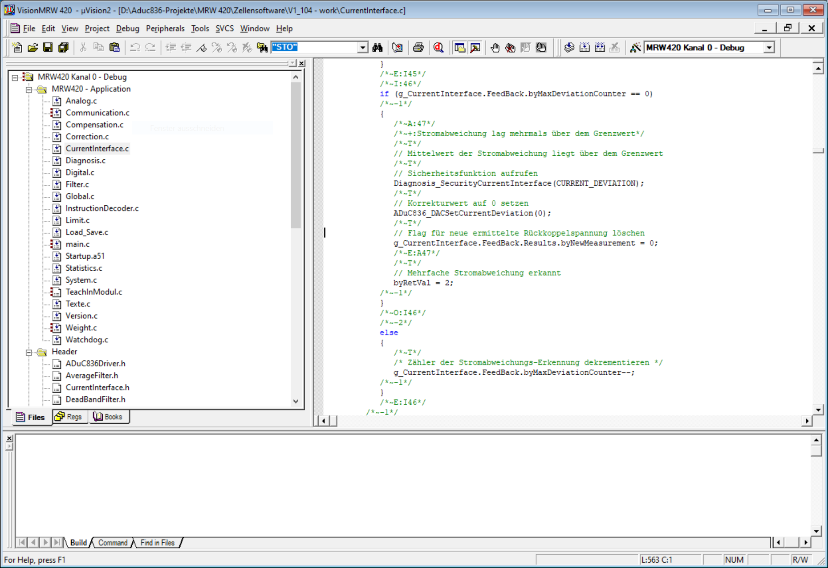
|  |  |  |
| --- | --- | --- |
| **Spec.** | **Verhalten** | **Bemerkung** |
| ST21.2.0 | Empfängt die Firmware ein in STX und ETX eingerahmtes Frame, welches nicht länger als der Empfangspuffer abzüglich einem Byte ist, so ist hinter den Empfangsdaten eine Null zur Endeerkennung des Empfangsstrings zu setzen.  Das *Flag SPI.chNewCommandReceived* muss gesetzt sein. | Empfangsframe passt in den Empfangspuffer |
| ST21.2.1 | Empfängt die Firmware ein in STX und ETX eingerahmtes Frame, welches länger als der Empfangspuffer abzüglich einem Bytes für das Stringendezeichen ist, so darf der Speicherbereich unmittelbar hinter dem Puffer nicht überschrieben werden. Auch ist das Frame zu verwerfen. Das *Flag SPI.chNewCommandReceived* darf nicht gesetzt sein. | Kein freier Empfangspuffer |

Im Rahmen der Software-Prüfung soll getestet werden, ob beim Eintreffen eines Frames über die SPI-Schnittstelle mit einer Länge größer der Puffergröße, der Adressbereich hinter dem Empfangspuffer nicht überschrieben wird und die empfangenen Daten verworfen werden.

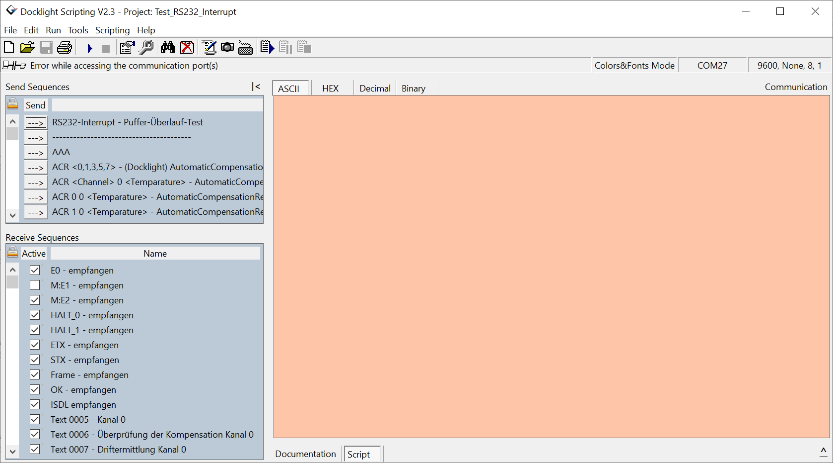
### **Testmittel**

#### Firmware ‚VisionMRW 420 – Testing‘ – zu testende Firmware in der Debugversion mit zusätzlichen Testroutinen zum SPI-Interrupt

#### Entwicklungsumgebung ‚Keil V2‘



#### Terminalsoftware ‚Docklight Scripting V2.3‘mit Projekt ‚MRW420\_V1.200 - Testing - IRQ.ptp‘



#### Entwicklungsumgebung ‚MRW420‘



#### MRW-Kommunikationsleitung

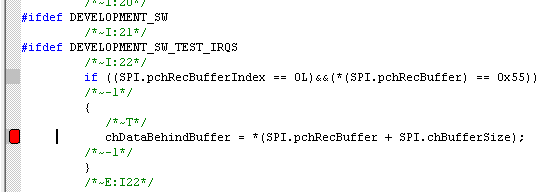


#### Adapter DB9 auf USB



### Testablauf

Alle Tests werden stellvertretend über den ersten Kanal abgewickelt, da im Bereich der zu testenden Interrupt-Funktion die Versionen für den jeweiligen Kanal identisch sind.  
Als Vorbereitung des Tests ist der zweite Kanal mit der passenden Test-Firmware zu bestücken. Dazu das uVision-Projekt ‚VisionMRW 420 - Testing.Uv2‘ in der Keil-Entwicklungsumgebung öffnen und das Target ‚MRW420 - Kanal 1 – Testing‘ wählen. Nach der Kompilierung ist diese Firmware auf den zweiten Kanal zu programmieren und zu starten.  
Alle weiteren Schritte finden ab jetzt ausschließlich auf dem ersten Kanal statt.  
Wechseln sie in das Target ‚MRW420 - Kanal 0 – Testing‘ und erstellen sie die Firmware. Nun die Firmware auf den ersten Kanal herunterladen aber noch nicht starten.  
An folgender Stelle in der Interruptfunktion der ADuC836-Bibliothek ‚SPI\_Interrupt()‘ einen Breakpunkt setzen:



Firmware starten.  
Es folgen zwei Tests:

1. Empfang eines Frames, welches exakt in Puffer passt.
2. Empfang des gleichen Frames wie unter 1., jedoch ein Byte länger.

Zunächst soll der Empfang eines Frames, exakt in den Emfangspuffer passend, untersucht werden. Es ist zu prüfen, dass kein Zeichen hinter den Puffer geschrieben wird, das letzte Zeichen im Puffer das Stringendezeichen ‚0x00‘ ist und der Merker zur Signalisierung eines erfolgreich empfangenen Frames (*SPI.chNewCommandReceived*) gesetzt ist. Zur besseren Sichtbarkeit des Pufferendes wird an dieser Stelle ein 0xAA eingetragen. Da das Frame in den Empfangspuffer exakt hineinpasst, muss dieses überschrieben worden sein.  
Dieser Test wird durch das Senden des Befehls *iTST* mittels Docklight-Scripting ausgeführt und anschließend nach dem Erreichen des Breakpunkts gemäß obigen Kriterien ausgewertet.  
In gleicher Weise erfolgt die Untersuchung eines zu langen Empfangsstrings. Die Aktivierung des Tests erfolgt nun über den Befehl   
*[iTST - Frame zu groß für Puffer]*. Die Kriterien für diesen Test sind, dass am Ende des Puffers das letzte empfangene Zeichen des Frames (*0x55)* eingetragen und der Merker für ein empfangenes Frame auf 0 steht. Zur besseren Sichtbarkeit des Pufferendes wird an dieser Stelle ein 0xAA eingetragen. Da das Frame nicht in den Empfangspuffer hineinpasst, muss dies noch sichtbar sein.  
Das wichtigste Kriterium jedoch ist, dass kein Zeichen über das Empfangspufferende hinaus in den Speicher eingetragen wurde.

### Testergebnisse

|  |  |  |
| --- | --- | --- |
| **Spec.** | **Verhalten** | **Ergebnis** |
| ST21.2.0 | Empfängt die Firmware ein in STX und ETX eingerahmtes Frame, welches nicht länger als der Empfangspuffer abzüglich einem Byte ist, so ist hinter den Empfangsdaten eine Null zur Endeerkennung des Empfangsstrings zu setzen.  Das *Flag SPI.chNewCommandReceived* muss gesetzt und die Adresse *SPI.pchRecBufferIndex* zur weiteren Freigabe des Empfangs nach Eingang eines *STX* muss 0 sein. Zur besseren Sichtbarkeit des Pufferendes wird an diese Stelle ein 0xAA eingetragen. Da das Frame in den Empfangspuffer exakt hineinpasst, muss dies überschrieben worden sein. | OK |
|  |
| ST21.2.1 | Empfängt die Firmware ein in STX und ETX eingerahmtes Frame, welches länger als der Empfangspuffer abzüglich einem Bytes für das Stringendezeichen ist, so darf der Speicherbereich unmittelbar hinter dem Puffer nicht überschrieben werden. Auch ist das Frame zu verwerfen. Das *Flag SPI.chNewCommandReceived* darf nicht gesetzt und die Adresse *SPI.pchRecBufferIndex* zur weiteren Freigabe des Empfangs nach Eingang eines *STX* muss 0 sein. Zur besseren Sichtbarkeit des Pufferendes wird an diese Stelle ein 0xAA eingetragen. Da das Frame nicht in den Empfangspuffer hineinpasst, muss dies noch sichtbar sein. | OK |
|  |

### Resultierendes Testergebnis

Test bestanden

# Kommentare

# Anhang