MOBA Mobile Automation AG

Spezifikation

ADuC836_SPISendPtr()

aus der ADuC836-Bibliothek

Version 1.016

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	Datum	Unterschrift
M.Offenbach	12.05.2022	

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

MOBA Mobile Automation AG
Kapellenstr. 15
D-65555 Limburg
Internet: www.moba.de

Inhaltsverzeichnis

1 Einführung

1.1 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.

Die hier vorliegende Spezifikation beschreibt das exakte Verhalten der Bibliotheksfunktion *ADuC836_SPISendPtr()* der Datei *ADuC836_SPISendPtr.c*
Dies beginnt mit der Angabe der Übergabeparameter sowie dem Rückgabewert der Funktion. Es folgen dann die Beschreibungen des Verhaltens der Funktion
Jede Beschreibung wird indiziert festgehalten. Somit ist in weiteren Dokumenten leicht Bezug auf die Spezifikation zu nehmen.

1.2 Änderungshistorie

Version	Datum	Kapitel	Änderung / Ergänzung
1.0	12.05.2022	alle	Erstellung

1.3 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
Sebastian Schlesies	Vertrieb	+49(0)6431-9577-267	s.schlesies@moba.de
Jürgen Stiller	Entwicklungsleiter	+49(0)6431-9577-282	j.stiller@moba.de
Norbert Lipowski	Entwicklung	+49(0)6431-9577-137	n.lipowski@moba.de

1.4 Anhänge

Dokumentname	Beschreibung

1.5 Glossar

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

2 ADuC836_SPISendPtr()

2.1 Beschreibung

ADuC836_SPISendPtr() sendet zeichenweise die Zeichen aus einem Speicherbereich aus. Hierzu bedient man sich der Funktion , ADuC836_SPISendChar()‘ zum Senden eines einzelnen Zeichens.
Aufgrund von Ergänzungsmaßnahmen in der SPI-Interruptroutine ergibt sich für diese eine längere Laufzeit. Dies wiederum hat zur Folge, dass bereits vor Abarbeitung des Interrupts, das nächste Zeichen gesendet wird. Dies kann demzufolge nicht empfangen werden.
Um das Problem zu beheben, ist nun in der zentralen Sendefunktion zur SPI ‚ADuC836_SPISendPtr()‘ nach dem Versand des Zeichens eine Pause von etwa 200us einzulegen.
Hierzu ist die Definition ‚ADUC836_SPI_GLOBALS_SEND_WITH_BYTEPAUSE‘ in der Datei SPI_Global.h zu treffen und in der gleichen Datei den Wert der Definition , ADUC836_SPI_GLOBALS_BYTEPAUSE‘ auf 20 zu setzen.

2.2 Spezifikation

Alle Spezifikationen sind in aufsteigender Reihenfolge zu erfüllen!

ADuC836_SPISendPtr()		
Index	**Parameter**	**Datentyp**
22.2.0.0	Zeiger auf den Sendepuffer	unsigned char*
	Anzahl der zu sendenden Zeichen (im Falle eines zu sendenden Strings darf dieser Wert auch 0 sein)	unsigned int
	Rückgabe	**Datentyp**
22.2.1.0	Sende-Status 0: Puffer erfolgreich gesendet 1: Fehler ‚Timeout'	char
	Verhalten	**Bemerkung**
22.2.2.0	Der Rückgabewert ‚byRetVal' ist zu Beginn der Funktionsausführung mit 0 zu initialisieren.	Initialisierung
22.2.2.1	Ist die Anzahl der zu sendenden Zeichen = 0, ist diese aus der Länge der Zeichenkette im Sendepuffer zu ermitteln.	Anzahl der zu sendenden Zeichen ermitteln.
22.2.2.2	Es sind Vorbereitungen für den anschließenden Empfang zu treffen. Dazu erfolgt der Aufruf der Bibliotheksfunktion ‚ADuC836_SPIReleaseReception()'	Vorbereitungen für den anschließenden Empfang treffen
22.2.2.3	SPI-Mastermodus einschalten. Dies geschieht durch Aufruf der Funktion , ADuC836_SPISwitch2Mode(ADUC836_SPI_MASTER_MODE)'	Bürde abgeklemmt – Abweichungszähler setzen ADUC836_SPI_MASTER_MODE = 0x10
22.2.2.4	Nacheinander sind nun die Zeichen aus dem Sendepuffer zu übertragen. Hierzu bediene man sich der Funktion ‚ADuC836_SPISendChar()' und legt deren Rückgabe in der Variablen ‚byRetVal' ab.	Sendepuffer ausgeben
22.2.2.5	Nach dem Senden eines Zeichens ist eine Pause von etwa 200us einzulegen. Dies geschieht über das ‚ADUC836_SPI_GLOBALS_BYTEPAUSE'-fache Inkrementieren einer zuvor mit 0 initialisierten Laufvariablen	Sendepause nach dem Senden eines Zeichens einlegen ADUC836_SPI_GLOBALS_WITH_BYTEPAUSE ist definiert ADUC836_SPI_GLOBALS_BYTEPAUSE = 20
22.2.2.6	Wurde beim Senden des Zeichens ein Fehler entdeckt, ist die Sendeschleife nach der Sendepause zu verlassen.	Sendefehler erkannt
22.2.2.7	SPI-Slavemodus einschalten. Dies geschieht durch Aufruf der Funktion ‚ADuC836_SPISwitch2Mode(ADUC836_SPI_SLAVE_MODE)'	Sendevorgang abschließen ADUC836_SPI_SLAVE_MODE = 0x00
22.2.2.8	Rückgabe der Variablen ‚byRetVal'.	Rückgabe

3 Kommentare

4 Anhang