MOBA Mobile Automation AG

Spezifikation

ADuC836_RS232Interrupt()

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_RS232Interrupt()* der Datei *ADuC836_RS232Interrupt.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_RS232Interrupt()

2.1 Beschreibung

ADuC836_RS232Interrupt ist die Interruptroutine zum Senden und Empfangen von über die RS232-Schnittstelle gesendete Zeichen. Diese Frames sind in die Framebegrenzungszeichen ‚STX' (0x02) und ‚ETX' (0x03) gebettet.
Der Empfang und damit der Eintrag der Zeichen an die erste Stelle im Empfangspuffer startet mit dem ersten Datum nach dem empfangenen ‚STX' und endet mit dem letzten Zeichen vor dem Empfang von ‚ETX'. Mit ‚ETX' muss noch ein Stringendezeichen (0x00) dem Puffer angefügt werden, um das Ende der Empfangszeichen erkennbar zu machen.
Während des Datenempfangs ist auf einen etwaigen Pufferüberlaufs zu prüfen. In diesem Fall werden die Daten verworfen und mit dem nächsten ‚STX' startet der Empfang erneut.
Beim Senden werden die Daten des Sendepuffer zeichenweise ausgegeben. Sobald im Sendepuffer eine 0x00 liegt, wird der Sendvorgang abgebrochen.

2.2 Spezifikation

Alle Spezifikationen sind in aufsteigender Reihenfolge zu erfüllen!

ADuC836_RS232Interrupt()		
Index	**Parameter**	**Datentyp**
20.2.0.0	./.	void
	Rückgabe	**Datentyp**
20.2.1.0	./.	void
	Verhalten	**Bemerkung**
20.2.2.0	Zur Ermittlung der Datenrichtung das RI-Flag (Receive Interrupt) auswerten. Ist das Flag gesetzt liegt ein Datum im Empfangsregister.	Datenrichtung ermitteln
	Datenempfang	
20.2.2.1	Zunächst ist zu prüfen, ob noch ein nicht verarbeitetes(r) Frame/Befehl im Empfangspuffer liegt (RS232.chNewCommandReceived = 1). Ist dies der Fall, ist die Interruptroutine nach dem Löschen des Receive Interrupt Flags (RI = 0) sofort zu verlassen	Überprüfung auf ein noch nicht verarbeitetes Frame
20.2.2.2	<u>Empfangspuffer ist leer:</u> Das Empfangszeichen ist aus dem RS232-Datenregister (SBUF) auszulesen und zwischen zu speichern	SPI-Datenregister auslesen
20.2.2.3	<u>Empfangspuffer ist leer:</u> Handelt es sich beim Empfangszeichen um ‚STX', ist ein Empfangspointer (‚RS232.pchRecBufferIndex') auf die Startadresse des Empfangspuffers (‚RS232.pchRecBuffer') zu legen. Zu statistischen Zwecken nun noch den STX-Zähler ‚RS232.Statistics.ulSTXCount' inkrementieren. Nach dem Löschen des Receive Interrupt Flags (RI = 0) wird die Funktion verlassen.	‚STX' empfangen RS232_WITH_STATISTICS ist definiert
20.2.2.4	Wurde kein ‚STX' empfangen muss untersucht werden, ob dies zuvor stattgefunden hat und damit der Empfangsprozess eingeleitet wurde. Hierzu bediene man sich des Empfangspointers ‚RS232.pchRecBufferIndex'. Ist dieser 0, wurde zuvor kein ‚STX' empfangen – der Empfangsprozess läuft nicht. Nach dem Löschen des Receive Interrupt Flags (RI = 0) wird die Funktion verlassen.	Kein ‚STX' empfangen. Prüfung auf eingeleiteten Empfangsprozess
20.2.2.5	<u>Empfangsprozess läuft:</u> Es ist nun zu untersuchen, ob das Empfangszeichen dem Frameabschlusszeichen ‚ETX' entspricht.	Empfangsprozess läuft – Auswertung des nächsten Zeichens
20.2.2.6	<u>Das aktuelle Empfangszeichen ist kein ‚ETX':</u> Um die Daten nicht fälschlich hinter den Empfangspuffer abzulegen, ist prüfen, ob der Empfangspointer ‚RS232.pchRecBufferIndex' innerhalb des Puffers liegt. Dies erfolgt mittels der bekannten Puffergröße (‚RS232.chBufferSize'):	Kein ‚ETX' empfangen Überprüfung auf Pufferüberlauf

20.2.2.7	Das aktuelle Empfangszeichen ist kein ‚ETX' und es findet kein Pufferüberlauf statt: Empfangszeichen an die Adresse des Empfangspointers (‚RS232.pchRecBufferIndex') ablegen und den Empfangspointer inkrementieren. Nach dem Löschen des Receive Interrupt Flags (RI = 0) wird die Funktion verlassen.	Kein ‚ETX' empfangen und kein Pufferüberlauf – Zeichen ablegen
20.2.2.8	Das aktuelle Empfangszeichen ist kein ‚ETX' aber es findet ein Pufferüberlauf statt: Empfangsprozess durch Setzen des Empfangspointers (‚RS232.pchRecBufferIndex') auf 0 anhalten. Damit wird erneut auf ein eingehendes ‚STX' gewartet. Nach dem Löschen des Receive Interrupt Flags (RI = 0) wird die Funktion verlassen.	Kein ‚ETX' empfangen aber ein Pufferüberlauf – Empfangsprozess anhalten
20.2.2.9	Das aktuelle Empfangszeichen ist ein ‚ETX': An die Adresse des Empfangspointer ‚RS232.pchRecBufferIndex' ist zur Erkennung des Frameendes eine 0 zu schreiben. Durch das Setzen des Empfangspointers auf 0 wird der Empfangsprozess angehalten.	‚ETX' empfangen Empfangsprozess anhalten und Flag ‚Neuer Befehl' setzen
	Im weiteren Ablauf muss untersucht werden, ob das erste Zeichen im Empfangspuffer einem ‚ESC' (0x1B) entspricht.	Abfrage auf ‚ESC'-Frame
	Erstes Zeichen im Empfangspuffer ist nicht ‚ESC': Damit andere Funktionen darüber informiert werden, dass ein neues Kommando vorhanden ist, nun noch das Flag ‚Neuer Befehl' (‚RS232.chNewCommandReceived') auf 1 setzen.	Kein ‚ESC'-Frame empfangen
	Erstes Zeichen im Empfangspuffer ist ‚ESC': In diesem Fall das Flag RS232.bESC auf 1 setzen.	‚ESC'-Frame empfangen
	Zu statistischen Zwecken nun noch den STX-Zähler ‚RS232.Statistics.ulETXCount' inkrementieren.	RS232_WITH_STATISTICS ist definiert
20.2.2.10	Sofern die Funktion bisher noch nicht verlassen wurde, das Receive Interrupt Flag nun löschen (RI = 0) und die Funktion abschließen.	Abschluss
Senden von Daten		
20.2.2.11	Zu sendendes Datum aus dem Sendepuffer auslesen und zwischenspeichern. Die Adresse des Sendepuffers ergibt sich aus der Pufferstartadresse (‚RS232.pchTransBuffer') und dem Pufferzeiger (‚RS232.nTransPointer').	Sendedatum auslesen
20.2.2.12	An die Stelle des Zeichens im Sendepuffer eine 0 eintragen.	Stelle im Sendepuffer löschen
20.2.2.13	Ist das aus dem Sendepuffer ausgelesene Zeichen = 0, den Pufferzeiger löschen und das Flag für eine beendete Übertragung (‚RS232.bEOT') auf 1 setzen.	Ausgelesenes Zeichen gleich 0

	Abschließend muss die TXD-Leitung auf 1 und das Transmit Interrupt Flag (TI) auf 0 gesetzt werden. Die Funktion ist zu verlassen	
20.2.2.14	Das aus dem Sendepuffer ausgelesene Zeichen != 0: Den Pufferzeiger inkrementieren und das zu sendende Zeichen in das RS232-Senderegister ‚SBUF' schreiben.	Ausgelesenes Zeichen ungleich 0
20.2.2.15	Das aus dem Sendepuffer ausgelesene Zeichen != 0: Ist die RS232-Timeoutzeit ungleich -1 (RS232.ulTimeoutRS232Release), diese auf die aktuelle Zeit (‚Timer.ulOperatingTime') plus 50ms setzen	Ggf. Timeout setzen
20.2.2.16	Nach dem Löschen des Transmit Interrupt Flags (TI) wird die Funktion verlassen.	Abschluss

3 Kommentare

4 Anhang