# 32-bit-Mikrocontroller ATMEL AT32UC3C CAN

Bei dem **AVR32** handelt es sich um eine proprietäre 32-bit-RISC-Architektur des Herstellers ATMEL. Das AT32UC3C-EK ist ein Evaluierungs- und Entwicklungswerkzeug für den AVR32-UC3C-Mikrocontroller.
In dem ToolSet enthalten sind das Entwicklungsboard AT32UC3C-EK von Atmel, eine auf 32 KByte Code limitierte Version der **IAR Embedded Workbench** für AVR32 und eine **JTAG In-Circuit Emulator mk-II** Debug-Probe von Atmel.
In der Entwicklungsumgebung IAR Embedded Workbench sind alle notwendigen Programmentwicklungs-Tools (C/C++ Compiler, Assembler, Linker, Library Builder, Editor, Projektmanager, Debugger und Simulator) integriert.

## 1. Versuchsziel

Es sollen die Funktion und die Assembler-Programmierung des Mikrocontrollers AT32UC3C am Beispiel der seriellen Kommunikation über den **CAN-BUS** demonstriert werden. Zur richtigen Ansteuerung der vier LEDs auf dem Enwicklungsboard **AT32UC3C-EK** ist daher die entsprechende Konfiguration des **CANIF**, des Interruptsystems und des GPIO-Controllers notwendig.

## 2. Grundlagen

Der AVR32-RISC-Prozessors **AT32UC3C** ist ein Mitglied der AVR32-UC3-Mikrocontrollerfamilie der Firma ATMEL. Er verfügt über 512 KB internen Flash und 64 KB internen Arbeitsspeicher. Es handelt sich um einen 32 bit Prozessor mit einer Taktfrequenz von bis zu 66 MHz. Auf dem Board **AT32UC3C-EK** wird ein 16 MHz Oszillator verwendent (PBA Clock=16 MHz). Aufgrund einer umfangreichen Integration von Standard-Peripherie, des flexiblen Interruptsystems, der hohen Rechengeschwindigkeit und der verschiedenen Betriebsmodi bietet sich ein breites Einsatzspektrum in eingebetteten Systemen.

Die **AVR32-CPU** besitzt einen orthogonalen Befehlssatz, daher können alle Register des allgemeinen Register-Satzes (Allzweckregister R0-R15) als Quelle oder Ziel bei Registeroperationen verwendet werden. Der Befehlssatz enthält kompakte (Länge 16 bit) und erweiterte (Länge 32 bit) Instruktionen. Der Stack Pointer (SP) ist in Register R13 des Register-Files gemappt. Ebenso belegen das Link Register (LR) und der Programm Counter (PC) die Register R14 und R15. Das LR-Register enthält die Rücksprungadresse eines Unterprogramms. Das PC-Register enthält die Adresse der aktuell ausgeführten Instruktion. Neben den Allzweckregistern enthält der AVR32UC noch zahlreiche Systemregister (Status- und Konfigurationsregister), die nur über spezielle Assemblerbefehle (MTSR, MFSR) gelesen oder geschrieben werden können.

Der **GPIO-Controller** (General Purpose Input Output Controller) dient dem Management der Ein/Ausgabeleitungen des Mikrocontrollers. Jeder als Eingang oder Ausgang konfigurierte E/A-Pin kann wahlweise vom PIO-Controller oder den internen Peripheriemodulen angesteuert werden.

Der integrierte CAN-Feldbus-Controller (**CANIF** - Controller-Area-Network-Interface) stellt eine asynchrone serielle Multi-Master-Schnittstelle für das CAN-Protokoll zur Verfügung. Duch die automatische Steuerung der CANIF-Hardware wird die Ansteuerung von 2 Kanälen mit je 16 Messageobjekten möglich. Das CAN-Interface unterstützt drei Channel-Modes: Normal Mode, Listening Mode und Loopback Mode. Auf dem Entwicklungsboard **AT32UC3C-EK** sind die beiden CAN-Schnittstellen **Channel0** und **Channel1** über die genormten DB9-Steckverbinder zugänglich.

Die Programmentwicklung und Testung auf dem Mikrocontroller AT32UC3C0512C wird durch ein leistungsfähiges On-Chip Debug System (OCD) unterstützt. Dieses Debug System basiert auf dem Standard NEXUS 2.0 claas 2+ und kann über ein externes Debug Tool (JTAGICE mk-II) am JTAG-Port des **AT32UC3C-EK** für den Download und Trace des Programms benutzt werden.
Die Programmentwicklung erfolgt in Assembler in der integrierten Entwicklungsumgebung **IAR Embedded Workbench** unter dem MS/Windows-Betriebssystem.

### 2.1 Der General Purpose Input Output Controller

Der **GPIO-Controller** (General Purpose Input Output Controller) dient dem Management der Ein/Ausgabeleitungen des Mikrocontrollers. Jeder als Ein- oder Ausgang konfigurierte E/A-Pin kann wahlweise vom PIO-Controller oder den internen Peripheriemodulen angesteuert werden. Alle Konfigurationsregister sind innerhalb von 4 Ports organisiert. Das Register GPIO_OVR (output value register) fungiert als Output-Register. Über das Input-Register GPIO_PVR (pin value register) können digitale Eingangssignale von den Controller-Pins eingelesen werden. Bei entsprechender Konfiguration des Interrupt Controllers (INTC) kann der GPIO-Controller Interrupt Requests (Gruppe 18, Line 0-15) bei der CPU anmelden, die durch externe Ereignisse an den PINs (Flanken- oder Pegeländerungen) ausgelöst werden. Eine detaillierte Funktionsbeschreibung des GPIO-Controller liefert ATMEL im Kapitel 23 (GPIO-Controller) des AT32UC3C-Manuals, sowie in der Applikation Note AVR32111.

### 2.2 CANIF - Controller-Area-Network-Interface

Das **CANIF-Modul** stellt eine multimasterfähige asynchrone serielle Feldbus-Schnittstelle zur Verfügung. Über das CAN Configuration Register CANCFG kann in Abhängigkeit vom Peripherietakt HSB und PB (CAN-clock) die Datenübertragungsgeschwindigkeit konfiguriert werden. Die Peripherietaktsignale sind nach einem RESET für die zwei Kanäle im CANIF-Module aktiviert, können aber über das Clock MASK Register im User-Interface des Power Manager PM bei Bedarf deaktiviert werden. Das CANIF-Modul besitzt ein Konfigurations- und ein Steuerregister (CANCFG und CANCTRL) zur Konfiguration der Chanel Modes.
Jeder der beiden CAN-Kanäle bietet die folgenden Funktionen:

- Message framing - unterstützt CAN-Protokoll-Spezifikation 2.0A und 2.0B
- Message filtering - Identifier mit 11 oder 29 Bits
- Message and status handling - 16 Message Objekte pro Kanal
- Message validation and acknowledgement
- Transfer rate and timing - maximale Bitrate von 1 Mbit/s
- Bus arbitration - CSMA/CA
- Fault Confinement - interne Sender- und Empfänger-Error-Counter
- Error Detection and Signaling - CAN-Error-Frames

Das CANIF-Module ist in das Interruptsystem des AVR32 eingebunden und der Gruppe 9 zugeordnet. Auf dem Entwicklungsboard **AT32UC3C-EK** sind die Signale der beiden seriellen Kanäle Channel0 und Channel1 über die GPIO-Funktionsgruppe B auf Port B und C zugänglich. Eine detaillierte Funktionsbeschreibung des CAN-Interfaces liefert ATMEL im Kapitel 29 (CANIF) des AT32UC3C-Manuals, sowie in der Applikation Note AVR32129.

### 2.3 Der Interruptcontroller

Der maskierbare **Interruptcontroller** sammelt die möglichen peripheren Unterbrechungs-Anforderungen und leitet einen Interrupt-Request und einen Interrupt-Vektor an die CPU weiter. Jede Peripheriekomponente belegt einen Eingang des Interrupt-Controllers und ist damit fest einer Interrupt-Gruppe und einem Interrupt-Prioritäts-Register (IPR0-46) zugeordnet. So lassen sich die Interrupt-Quellen aktivieren und priorisieren. Es gibt 4 Prioritätsgruppen. In den IPR-Registern werden auch die Interrupt-Vektoren festgelegt. Die Anfangsadresse der Event/Interrupt-Vektor-Tabelle enthält das Register EVBA innerhalb der AVR32-Systemregister. Innerhalb des Interrupt-Controllers (INTC) gibt es 64 Interrupt-Prioritäts-Register (IPR0-63), 64 Interrupt-Request-Register (IRR0-63) und 4 Interrrupt-Cause-Register (ICR0-3). Die Interrupt-Bits GM (Global Interrupt Mask) und EM (Exception Mask) befinden sich im Status Register der AVR32-CPU. Das Register ECR (Exception Cause Register) innerhalb der AVR32-Systemregister enthält den Adressen-Offset des letzten aufgetretenen Events.
Eine detaillierte Funktionsbeschreibung der Interrupt Controller liefert ATMEL in den Kapiteln 11 und 12 (INTC und EIC) des AT32UC3C-Manuals, sowie in der Applikation Note AVR32101.

## 3. Studienfragen

- 3.1 Charakterisieren Sie kurz den Frame-Aufbau im CAN-Protokoll.
- 3.2 Erläutern Sie das User-Interface vom CANIF-Module. Welche Konfigurationsmöglichkeiten bieten die Control- und Mode-Register?
- 3.3 Wie erfolgt die Datenübertragung in dem jeweiligen seriellen Sende- und Empfangskanal?
- 3.4 Welche internen Testmöglichkeiten bieten die CANIF-Kanäle des AVR32-Controllers?
- 3.5 Warum muss die Bitübertragungsgeschwindigkeit im Sender und Empfänger gleich sein?
- 3.6 Erläutern Sie die Einbindung der CANIF-Module in das Interruptsystem des AVR32-Controllers.
- 3.7 Erläutern Sie das CRC-Verfahren zur Fehlererkennung bei der seriellen Datenübertragung.
- 3.8 Wie erfolgt im CANIF-Modul des AVR32 die Bildung und Übertragung der CRC-Bits?
- 3.9 Wie lassen sich Message-Objekte anlegen? Erläutern Sie deren Datenstruktur.
- 3.10 Wie erfolgt die zerstörungsfreie Busarbitrierung mit Hilfe der Nachrichtenidentifier?

## 4. Aufgaben

- 4.1 Erzeugen Sie in Ihrem Heimatverzeichnis einen eigenen IAR-Workspace anhand der Praktikumsvorlage.
- 4.2 Durch Betätigen der Schalter PB0 und PB1 auf dem AVR32-Board sollen die LEDs angesteuert werden.
- 4.3 Erweitern Sie Ihr Assemblerprogramm für die Konfiguration der Peripheriemodule CANIF, INTC und GPIO.
- 4.4 Schreiben Sie ein Programm zur Duplexkommunikation zwischen zwei AT32UC3C-EK-Boards über die serielle Schnittstelle CAN_0. Durch Betätigen der Schalter PB0 und PB1 auf dem Senderboard sollen auf dem Empfängerboard die LEDs angesteuert werden.
- 4.5 Konfigurieren Sie das Interruptsystem zur Steuerung der Unterprogramme für den Sende- und Empfangsbetrieb. Welche Funktion erfüllt dabei die Interrupt Masken- und Status-Register CANIMR und CANISR des CAN User Interfaces?

## 5. Literaturverweise


- ATMEL AVR32 Homepage
- CAN in Automation (CiA) CAN knowledge
- AVR32 Forum Homepage: www.avrfreaks.net
- ATMEL AVR32-Tools: AT32UC3C-Evaluation Kit
- ATMEL AVR32-Devices: AT32UC3C0512C
- ATMEL Manual AVR32 Microcontroller AT32UC3C
- ATMEL Manual AT32UC3C: 4.4 Programmiermodell
- ATMEL Manual AT32UC3C: 11. Interrupt Controller
- ATMEL Manual AT32UC3C: 12. External Interrupt Controller
- ATMEL Manual AT32UC3C: 23. GPIO-Controller
- ATMEL Manual AT32UC3C: 29. CAN-Interface
- Application Note AVR32101: Configuring the AVR32 Interrupt Controller
- Application Note AVR32919: AT32UC3C-EK User Guide
- Application Note AVR32129: Using the AVR32 UC3C CAN-Interface
- ATMEL AVR32 Architecture Document: 2. AVR32 Programmiermodell
- ATMEL AVR32 Architecture Document: 8. AVR32 RISC Instruction Set
- ATMEL AVR32 IAR Assembler Reference Guide
- ATMEL AVR JTAG In-Circuit Emulator mk-II: Description
- ATMEL AVR JTAG In-Circuit Emulator mk-II: User Guide
- ATMEL AVR JTAG In-Circuit Emulator mk-II: User Guide - LEDs
- ATMEL AVR JTAG In-Circuit Emulator mk-II: Quick Start Guide
- ATMEL AVR32 UC3C-EK Documentation: Software Framework
- ATMEL AVR32 UC3C-EK Documentation: CAN Stack Example 1
- ATMEL AVR32 UC3C-EK Documentation: CAN Stack Example 2
- ATMEL AVR32 UC3C-EK Documentation: CAN and LIN Loopback Application
- ATMEL AVR32 UC3C-EK Documentation: Schematics
- D. Tavangarian, D. Versick: Basiswissen Rechnerstrukturen & Betriebssyteme, Verlag W3I GmbH, 2008, ISBN 3937137289, 9783937137285: 5.6. Prozessorarchitekturen
- Vorlesung IEF/MD Technische Grundlagen der Rechnerkommunikation


## 6. Anhang

### 6.1 AT32UC3C-EK - Belegung der Ein- und Ausgabeports

Alle Konfigurationsregister des GPIO-Controllers sind innerhalb von 4 Ports angeordnet.

- Port0: 0xFFFF2000
- Port1: 0xFFFF2200
- Port2: 0xFFFF2400
- Port3: 0xFFFF2600

| GPIO-Port | GPIO-Adresse | GPIO-Register | Peripherie |
|---|---|---|---|
| Port0 | FFFF2000 | GPER | PA[0 - 29] |
| Port0 | FFFF2004 | GPERS | PA[0 - 29] |
| Port0 | FFFF2008 | GPERC | PA[0 - 29] |
| Port0 | FFFF200C | GPERT | PA[0 - 29] |
| Port0 | … | | |
| Port0 | FFFF2050 | OVR - output value register | PA[0 - 29] |
| Port0 | FFFF2054 | OVRS - ovr-set | PA[0 - 29] |
| Port0 | FFFF2058 | OVRC - ovr-clear | PA[0 - 29] |
| Port0 | FFFF205C | OVRT - ovr-toggle | PA[0 - 29] |
| Port0 | FFFF2060 | PVR - pin value register | PA[0 - 29] |
| | Port0[8] | | LED0 |
| | Port0[14] | | Taste PB0, SW4 |
| | Port0[29] | | Taste PB1, SW5 |
| | … | | |
| Port1 | FFFF2200 | GPER | PB[0 - 31] |
| Port1 | FFFF2204 | GPERS | PB[0 - 31] |
| Port1 | FFFF2208 | GPERC | PB[0 - 31] |
| Port1 | FFFF220C | GPERT | PB[0 - 31] |
| Port1 | FFFF2210 | PMR0 - peripherie mux register0 | PB[0 - 31] |
| Port1 | FFFF2220 | PMR1 - peripherie mux register1 | PB[0 - 31] |
| Port1 | FFFF2230 | PMR0 - peripherie mux register2 | PB[0 - 31] |
| Port1 | … | | |
| | Port1[4] | | RxCAN_0 |
| | Port1[5] | | TxCAN_0 |
| | … | | |
| Port2 | FFFF2400 | GPER | PC[0 - 31] |
| Port2 | FFFF2404 | GPERS | PC[0 - 31] |
| Port2 | FFFF2408 | GPERC | PC[0 - 31] |
| Port2 | FFFF240C | GPERT | PC[0 - 31] |
| Port2 | FFFF2410 | PMR0 - peripherie mux register0 | PC[0 - 31] |
| Port2 | FFFF2420 | PMR1 - peripherie mux register1 | PC[0 - 31] |
| Port2 | FFFF2430 | PMR0 - peripherie mux register2 | PC[0 - 31] |
| Port2 | … | | |
| Port2 | FFFF2450 | OVR - output value register | PC[0 - 31] |
| | Port2[11] | | RxCAN_1 |
| | Port2[12] | | TxCAN_1 |
| | Port2[13] | | LED2 |
| | … | | |
| Port3 | FFFF2600 | GPER | PD[0 - 30] |
| Port3 | … | | |
| Port3 | FFFF2650 | OVR - output value register | PD[0 - 30] |
| | Port3[22] | | LED3 |
| | Port3[23] | | LED1 |

### 6.2 AT32UC3C - Interrupt Request Signal Map

Innerhalb des Interrupt-Controllers (INTC) gibt es Interrupt-Prioritäts-Register (IPR), Interrupt-Request-Register (IRR) und Interrrupt-Cause-Register (ICR). Das Register EVBA (Exception Vector Base Address) innerhalb der AVR32-Systemregister enthält die Anfangsadresse der Interrupt-Vektor-Tabelle. Die Interrupt-Bits GM (Global Interrupt Mask) und EM (Exception Mask) befinden sich im Status Register der AVR32-CPU. Das Register ECR (Exception Cause Register) innerhalb der AVR32-Systemregister enthält den Adressen-Offset des letzten aufgetretenen Events.

| Adresse | Register USART0 | Name |
|---|---|---|
| FFFF2800 | Control Register | CR |
| FFFF2804 | Mode Register | MR |
| FFFF2808 | Interrupt Enable Register | IER |
| FFFF280C | Interrupt Disable Register | IDR |
| FFFF2810 | Interrupt Mask Register | IMR |
| FFFF2814 | Channel Status Register | CSR |
| FFFF2818 | Receiver Holding Register | RHR |
| FFFF281C | Transmitter Holding Register | THR |
| FFFF2820 | Baud Rate Generator Register | BRGR |
| FFFF2824 | Receiver Time-out Register | RTOR |
| FFFF2828 | Transmitter Timeguard Register | TTGR |
| FFFF2840 | FI DI Ratio Register | FIDI |
| FFFF2844 | Number of Errors Register | NER |
| FFFF284C | IrDA Filter Register | IFR |
| FFFF2850 | Manchester Encoder Decoder Register | MAN |
| FFFF28FC | Version Register | VERSION |

### 6.3 AT32UC3C - CANIF Register Memory Map

Innerhalb des CAN-Controllers (CANIF) gibt es Konfigurations-, Control- und Status-Register (CANCFG, CANCTRL, CANSR). Das Konfigurationsregister CANRAMB der beiden Kanäle des CANIF-Controllers enthält die Basis-Adresse des RAM-Bereiches für die möglichen 16 MOBs pro Kanal.

| Adresse Kanal 0 | Register | Name |
|---|---|---|
| FFFD1C00 | Version Register | VERSION |
| FFFD1C04 | Parameter Register | PARAMETER |
| FFFD1C08 | RAM Base Address Register | CANRAMB |
| FFFD1C0C | Configuration Register | CANCFG |
| FFFD1C10 | Control Register | CANCTRL |
| FFFD1C14 | Status Register | CANSR |
| FFFD1C18 | Fault Confinement Register | CANFC |
| FFFD1C1C | Interrupt Enable Register | CANIER |
| FFFD1C20 | Interrupt Disable Register | CANIDR |
| FFFD1C24 | Interrupt Mask Register | CANIMR |
| FFFD1C28 | Interrupt Status Clear Register | CANISCR |
| FFFD1C2C | Interrupt Status Register | CANISR |
| FFFD1C30 | MOb Search Register | MOBSCH |
| | … | |
| FFFD1C5C+(n*0xC) | MOb Control Register (0<=n<=15) | MOBCTRLn |
| FFFD1C60+(n*0xC) | MOb Status Clear Register | MOBSCRn |
| FFFD1C64+(n*0xC) | MOb Status Register | MOBSRn |

Laboringenieur
Dipl.-Ing. Th. Wegner
Raum: W1314
Tel.: 498 7267

Schnelleinstieg
Publikationen
Anfahrt
Kontakt
Laborpraktikum
Lehrangebot
Highlights
Projekte