Anbindung von SD-Karten an ARM- und 8-Bit-MCUs

Controller spielt Karten

Flash-Speicherkarten sind durch den Markterfolg von Digitalkameras inzwischen so günstig, dass sie zunehmend auch für den Einsatz in Embedded Systemen attraktiv werden. Durch die jüngsten Veröffentlichungen von Teilspezifikationen zu SD- und MMC-Kartentypen können Entwickler jetzt auch diese besonders einfach anzusteuernden Massenspeicher nutzen. Der vorliegende Beitrag stellt die Karten mit ihren technischen Details vor, beschreibt die Schnittstellen und Protokolle und zeigt in beispielhaften Implementierungen den Anschluss der Karten an Prozessoren auf ARM-Basis mit integriertem Karten-Host-Controller ebenso wie an einfache 8-Bit-Mikrocontroller.

Mirco Fuchs
ist wissenschaftlicher Mitarbeiter am FTZ Leipzig,
Carsten Kögler
arbeitet als Entwickler am FTZ Leipzig,
Nico Mäding
ist System Application Engineer bei Renesas,
Tobias Wengemuth
arbeitet als Field Application Engineer bei Spoerle, alle haben Elektrotechnik an der HTWK Leipzig studiert

Bei der Entwicklung von Embedded Systemen sind große und zugleich portable Datenspeicher zunehmend gefragt. So sollen zum Beispiel mit Monitoring-Mechanismen verschiedenste Daten über einen langen Zeitraum aufgezeichnet werden, oder der Anwender soll auf sehr einfache Weise Firmware-Upgrades durchführen oder anderweitig Parameter einer Applikation verändern können.

Als portable Datenspeicher haben sich in den letzten Jahren vor allem MMC- und SD-Speicherkarten auf dem Markt etabliert. Beide Kartentypen basieren auf Flash-Speicher, meist in NAND-Technik, da diese für Massenspeicher gegenüber NOR-Flash den Vorteil wesentlich niedrigerer Preise bietet.

Dieser Beitrag stellt zunächst die Grundlagen der betrachteten SD- und MMC-Karten sowie die in diesem Zusammenhang benutzte Testumgebung vor (siehe Kasten »Die Testumgebung im Detail«). Danach wird gezeigt, wie sich diese Karten an verschiedene Mikrocontroller-Typen einfach anbinden lassen. Da in vielen Embedded-Applikationen immer häufiger leistungsfähige Controller zusammen mit einem Betriebssystem eingesetzt werden, stellt dieser Beitrag zusätzlich auch die Anbindung an einen ARM9-basierten Prozessor mithilfe von Embedded-Linux vor.

Consumer-Anwendungen wie Digitalkameras, Handys und MP3-Player bestimmen den Markt für Speicherkarten. Hier hat sich eine breite Vielfalt an Kartenformaten etabliert: Neben der inzwischen etwas in die Jahre gekommenen »CompactFlash«-Karte (CF) vor allem »Memory-Stick« (MS), »Secure-Digital«-Memory-Card (SD), »MultiMedia-Card« (MMC) und »xD-Picture-Card« [11]. Jedes Kartenformat hat wiederum verschiedene Derivate in unterschiedlichen Größen und für andere Anwendungsgebiete hervorgebracht. Den Embedded-Bereich dominierte zunächst lange Zeit die CompactFlash-Karte, sie wird heute aber auf breiter Front von der MultiMedia-Card und der SD-Card abgelöst.

Die »MultiMediaCard«

1997 entwickelte die Siemens-Tochter Ingentix zusammen mit SanDisk die »MultiMedia-Card« (MMC) und damit die Vorgängertechnik der »SD-Card«. Die MMC besitzt einen integrierten Controller, der die Ansteuerung der Karte übernimmt und somit, im Gegen-

satz zur Smart-Media-Card, ein einfacheres Interface nach außen zulässt. Ihre Speicherkapazität reicht von 2 MByte bis 4 GByte, die Datenübertragungsgeschwindigkeit liegt real bei bis zu 2 MByte/s [3]. Die »MultiMediaCard Association« ([2]) entwickelt den Standard weiter, womit die MMC in Konkurrenz zur SD-Card steht. Eine Verbreiterung des Busses bei der »MMCplus« soll auch den Nachteil der im Vergleich geringeren Geschwindigkeit ausgleichen.

Die »SD-Card«

Im Januar 2000 gründeten die Firmen Matsushita (besser bekannt durch die Marke Panasonic), SanDisk und Toshiba die »Secure Digital Card Association«. Ziel der Organisation war die Weiterentwicklung der MultiMediaCard. 2001 kamen die ersten SD-Karten auf den Markt. Sie unterstützten im Gegensatz zum MMC-Standard von Anfang an Digital-Rights-Management (DRM) und besaßen ein schnelleres Interface. Die Spezifikation stand bis letztes Jahr lediglich den Lizenznehmern der SD-Card-Association zur Verfügungen. Anfang 2006 veröffentlichte die SD-Card-Association schließlich eine vereinfachte Spezifikation. Ein wichtiger Schritt, der diesen Standard nun für einen deutlich größeren Anwenderkreis interessant macht und letztlich auch diesen Beitrag motivierte. Tabelle 1 zeigt die Versionsübersicht der SD-Card-Spezifikation mit den wichtigsten Neuerungen.

Bild 1: SDHC-Geschwindigkeitsklassen [1]

Beide Standards wurden über die Jahre weiterentwickelt, und neue Kartenformate sind hinzugekommen. Eine Übersicht gibt Tabelle 2. Zu beiden Karten kamen größenreduzierte Versionen auf den Markt, zudem stiegen die Datenraten durch Verbreiterung der Busbreiten und Taktfrequenzen. Mit der »secureMMC« ist auch eine MMC mit DRM-Funktion verfügbar.

So genannte SD-High-Capacity-Karten (SDHC) mit Speicherkapazitäten über 2 GByte wurden mit der neuen SD-Spezifikation 2.0 eingeführt. Sie definiert drei Klassen hinsichtlich Geschwindigkeit und Leistung, welche die Mindestdatenrate für Schreibzugriffe festlegen (Bild 1). Bei Karten der Klasse 2 sind Schreibvorgänge mit 2 MByte/s möglich, bei Klasse 4 4 MByte/s und bei Klasse 6 entsprechend 6 MByte/s. Dies bedeutet, dass eine SDHC-Karte, die mit »6« gekennzeichnet ist, Daten mit mindestens 6 MByte/s schreiben können muss. Die bisherigen Spezifikationen sahen solche für alle Hersteller allgemeinverbindliche Standards für Transferraten nicht vor. SD- und SDHC-Karten sind zwar hinsichtlich Größe und Form identisch, jedoch sind die neuen SDHC-Karten nur in SDHC-kompatiblen Geräten einsetzbar.

Erwähnt werden sollten auch die so genannten »SD Input/Output-Cards« (SDIO). Diese sind keine Speicherkar-

Nr.	Jahr	Kurzbeschreibung
1.0	2000	Aufbauend auf MMC. Um DRM-Funktionen erweitert. Die Taktrate ist auf 12,5 MHz begrenzt
1.01	2001	Nur kleinere Änderungen. In diesem Jahr wurde die SD-Card offiziell vorgestellt.
1.1	2004	Spezifiziert Karten ab 2 GByte und eine höhere Taktrate von 50 MHz. Somit sind theoretisch Datenraten von 25 MByte/s, (133x) möglich
2.0	2006	Einführung von SDHC. Spezifikation nicht öffentlich verfügbar.

Tabelle 1: Entwicklung der SD-Card-Spezifikation [1]

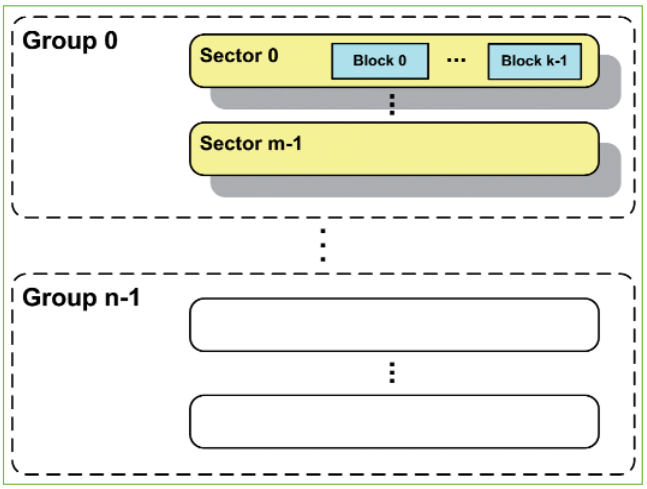


Bild 2: Speicherpartition einer SD-Card

ten, sondern bieten Funktionen wie WLAN, Bluethooth oder USB. Um diese Karten nutzen zu können, muss der Host diese aber explizit unterstützen. Zusätzlich gibt es so genannte Combo-Cards, die eine Speicherkarte mit einer SDIO-Karte kombinieren.

Der Speicheraufbau

Die Verwaltung von SD-Karten erfolgt über interne Register, auf die mit unterschiedlichen, im Verlauf des Artikels beschriebenen Methoden zugegriffen werden kann. Mithilfe dieser Register lassen sich einige Parameter der Karte konfigurieren, die meisten sind nur lesbar. Der Speicher einer SD-Karte ist aufgrund der verwendeten Flash-Technik blockweise aufgebaut (Bild 2). Ein Block bildet die kleinste beschreibbare Speichereinheit, seine maximale Länge ist als Konstante im »CSD«-Register der SD-Karte gespeichert. Ein Sektor umfasst die Anzahl von Blöcken, die als zusammenhängende Einheit gelöscht werden kann. Bei einigen SD-Karten besteht die Möglichkeit, mehrere, als Gruppen zusammengefasste Sektoren mithilfe eines Kommandos als »schreibgeschützt« zu deklarieren.
Zusätzlich zu diesem Datenbereich ist bei der SD-Karte ein separater Speicherbereich für die DRM-Funktionalität der Karte (Digital Rights Management) reserviert. Dieser, als »Protected-Area« bezeichnete Speicherbereich ist ebenfalls in Sektoren und Blöcken organisiert und nur nach einer Authentifizierung zwischen Karte und Host verfügbar. Sämtliche Spezifikationen hierzu sind nicht öffentlich. Der nicht durch diese Funktion geschützte Speicherbereich der Karte wird im weiteren Verlauf als »User-Bereich« bezeichnet.
Zur Bezeichnung der Karte werden normalerweise SI-Vorsilben verwendet. Eine »1-GB«-Karte ist zum Beispiel aus 2 004 480 Blöcken zu je 512 Byte aufgebaut, sie hat also eine Kapazität von 979 MByte. Bei der SD-Card entfällt etwa 1% dieses Speichers für den geschützten Speicherbereich, damit verbleiben noch 969 MByte für den User-Bereich. Die genaue Blockanzahl kann von Hersteller zu Hersteller variieren. Der User-Speicherbereich ist normalerweise wie bei einer Festplatte mithilfe einer Partitionstabelle organisiert. In der Regel gibt es nur eine Partition. Dies ist meist eine »FAT16«-Partition (FAT = File Allocation Table), wobei hier die Grenze bei 2 GByte liegt. Darüber kommt meist »FAT32« zum Einsatz. Selbstverständlich lassen sich auch andere Dateisysteme verwenden.

Hardwareschnittstellen und Protokolle

Grundsätzlich lassen sich zwei unterschiedliche Hardwareschnittstellen für die Kommunikation mit der SD-Karte verwenden. Zum einen ist dies der bidirektionale »SD-Bus«, der entweder im 4-Bit-Modus oder im 1-Bit-Modus betrieben werden kann (Tabelle 3). Im 1-Bit-Modus ist der SD-Bus weitestgehend kompatibel mit dem MMC-Bus.
Die zweite Möglichkeit ist die Verwendung einer SPI-Schnittstelle (Tabelle 4), die für Mikrocontroller ohne integriertes SD-Card-Hostinterface die erste Wahl darstellt. Im Gegensatz zu MMC-Karten verfügen SD-Karten über einen Schreibschutzschalter, dessen Position allerdings der Host auswerten muss. Es besteht kein Kontakt zum Kartencontroller. Der Zustand des Schalters kann über einen Controllerpin eingelesen und durch Software ausgewertet werden. Ein weiterer Kontakt im Steckverbinder dient der Erkennung einer eingeschobenen Karte.
Soll das Embedded System sowohl MMC- als auch SD-Karten unterstützen, muss die Taktrate für die Datenübertragung variabel sein. Bei MMC-Karten geht die CMD-Leitung nach dem Einschalten in einen Open-Drain-Modus und muss während der Initialisierung mit einer Frequenz von 400 kHz angesteuert werden. Anschließend erfolgt über ein Kommando die Umstellung auf eine höhere Frequenz und Push-Pull-Betrieb. Auf das Bus-Protokoll im SD- bzw. MMC-Bus-Modus soll hier nur kurz eingegangen werden, da beide Modi für Mikrocontroller

	MMC	RS-MMC	MMC plus	MMC mobile	secure MMC	SD	miniSD	microSD (TransFlash)
Pins	7	7	13	13	7	9	11	8
Größe B/L/H	24/32/1,4	24/18/1,4	24/32/1,4	24/18/1,4	24/32/1,4	24/32/2,1	20/21,5/1,4	11/15/1
SPI mode	ja	ja	ja	ja	ja	ja	ja	ja
1-Bit-mode	ja	ja	ja	ja	ja	ja	ja	ja
4-Bit mode	nein	nein	ja	ja	nein	ja	ja	ja
8-Bit mode	nein	nein	ja	ja	nein	nein	nein	nein
Takt	bis 20 MHz	bis 20 MHz	bis 52 MHz	bis 52 MHz	bis 20 MHz	bis 50 MHz	bis 50 MHz	bis 50 MHz
DRM	nein	nein	nein	nein	ja	ja	ja	ja

Tabelle 2: Derivate der »MultiMediaCard« (MMC) und der »SecureDigital Card« (SDC)

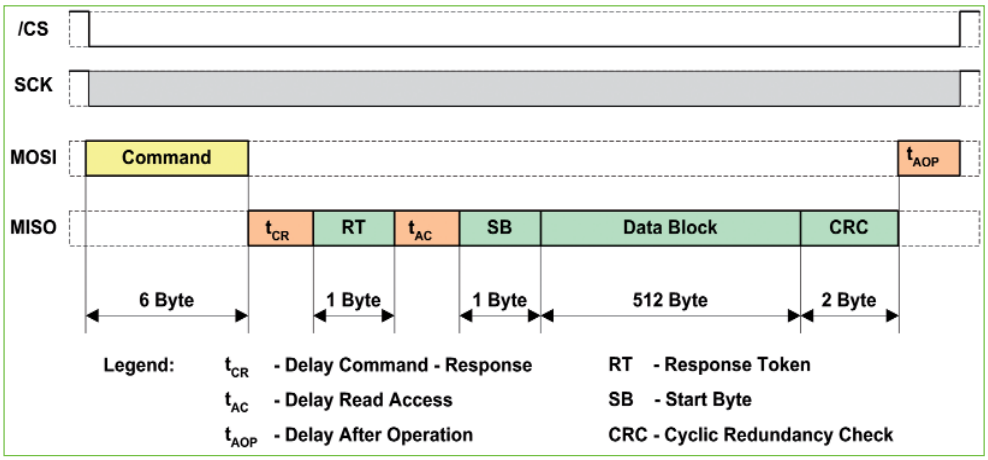


Bild 3: Ablauf eines Lesezugriffs über das SPI-Interface im Single-Block-Modus

ohne Hostcontroller nicht in Frage kommen.
Beide Bussysteme ermöglichen die Ansteuerung mehrerer Karten an einem Host. Beim SD-Bus teilen sich dabei bis zu 10 Karten eine Taktleitung. Im Falle des MMC-Bus teilen sich die maximal 30 Karten zusätzlich die CMD- und Datenleitungen. Während der Initialisierung weist der Host jeder Karte anhand der eindeutigen Nummer im CID-Register (Card Identification Register, siehe Tabelle 6) eine »Session-Address« zu. Über die CMD-Leitung laufen Kommandos vom Host zur Karte sowie die Antworten (Response) der Karte. Eine CRC-Prüfsumme dient der Übertragungssicherung von Kommandos und Daten.
Von diesem Protokoll unterscheidet sich der Ablauf einer SPI-Übertragung dahingehend, dass die Datenleitungen (MISO, MOSI) unidirektional sind (Tabelle 4). Kommandos gehen also über die »DataIn«-Leitung vom Host zur Karte, die Antwort der Karte an den Host erfolgt über die »DataOut«-Leitung. Jeder Datenblock wird dabei mit einem 16-Bit-breiten CRC-Wert versehen.
Nahezu unabhängig vom verwendeten Interface erfolgt die Ansteuerung der Karten mit demselben Befehlssatz. Einige Kommandos und Registerbits sind im SPI-Modus nicht verfügbar. Ebenso unterscheiden sich die Kommandos und Registerbits geringfügig zwischen MMC- und SD-Karten.
Die SD-Card-Spezifikation definiert 41 Kommandos. Diese sind entsprechend ihrer Funktion in Klassen sortiert. So definiert die Klasse »Basic« beispielsweise grundlegende Kommandos wie das Zurücksetzen (CMD0) und das Auslesen des CSD-Registers (CMD9). Welche Kommandoklassen von einer Karte unterstützt werden, ist im CSD-Register hinterlegt. Für die Datenübertragung zwischen Host und Karte ist zwischen Single-Block- und Multi-Block-Modus zu unterscheiden. Beim Multi-Block-Modus werden die Daten solange gesendet, bis der Host

Pin	Name	Modus	Beschreibung
1	CD/ DAT[3][2]	Bidirektional, PP	Card Detect / Datenbit 3
2	CMD[3]	Bidirektional, OC	Command und Response
3	GND		Masse
4	VDD		Versorgungsspannung
5	CLK	Input	Takt
6	GND		Masse
7	DAT[0]	Bidirektional, PP	Datenbit 0
8	DAT[1][1,2]	Bidirektional, PP	Datenbit 1
9	DAT[2][1,2]	Bidirektional, PP	Datenbit 2

[1] Kontakt bei MMC nicht vorhanden.
[2] Diese Signale sind nach Reset der Karte Eingänge und sollten mit einem Pull-Up beschaltet werden.
[3] Sowohl MMC als auch SD-Spezifikation empfehlen einen Pull-Up-Widerstand für diese Leitung. Ein Wert von 1 kΩ hat sich bewährt.

Tabelle 3: Pinbelegung des SD/MMC-Bus

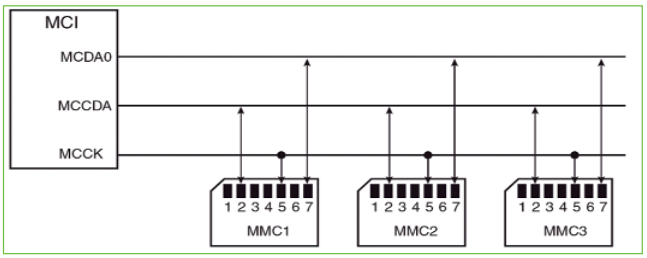


Bild 4: MCI-Slot mit MMC-Bus [5]

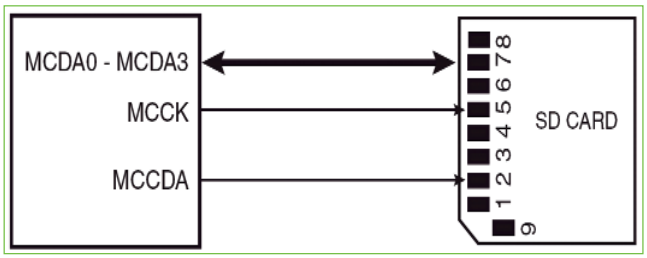


Bild 5: MCI-Slot mit SD-Card [5]

die Übertragung mit einem »Stop«-Kommando beendet. Auf diese Weise lässt sich der Overhead auf ein Minimum reduzieren. Bild 3 zeigt den Ablauf eines Lesezugriffs über das SPI-Interface im Single-Block-Modus.

Als Angabe für die Datentransferrate ist häufig die Burst-Datenrate zu finden. Sie gibt die maximal mögliche Datenrate vom Host in den karteninternen Pufferspeicher an und ergibt sich direkt aus der maximalen Busfrequenz (Tabelle 5). In der Praxis müssen aber die Zeiten für das Schreiben der Daten vom Puffer in den Flash-Speicher berücksichtigt werden.

Um Informationen zur Konfiguration einer Host-Verbindung zu gewinnen, stehen interne kartenspezifische Register zum Auslesen bereit. Tabelle 6 zeigt eine Auflistung mit der Bedeutung der wichtigsten Register [4].

Viele leistungsstarke 32-Bit-Controller besitzen einen integrierten Host-Controller zur Ansteuerung von SD-Karten. Durch die große Ähnlichkeit der Kartenstandards ist meist auch die Ansteuerung von MMC-Karten möglich. Im folgenden Abschnitt soll anhand eines Controllers vom Typ »AT91RM9200« exemplarisch das »MultiMediaCard-Interface« (MCI) von Atmel vorgestellt werden, welches in Controllern der »AT91«-Baureihe auf ARM7- und ARM9-Basis implementiert ist. Ein in einem Controller integriertes MCI-Modul kann mehrere MCI-Slots enthalten, wobei an jedem Slot eine Karte angeschlossen sein kann.

Atmel-Controller mit MCI-Interface

Das MultiMediaCard-Interface (MCI) des AT91RM9200 unterstützt die MultiMediaCard-Spezifikation 2.2 (MMC) und die SD-Card-Spezifikation 1.0 und enthält zwei MCI-Slots. Es ermöglicht die Ansteuerung von SD-Karten über eine 9-Pin-Schnittstelle und von MMC-Karten über eine 7-Bit-Schnittstelle. Am MCI-Bus können entsprechend der MMC-Spezifikation bis zu 30 Karten angeschlossen werden. Diese teilen sich die Signale Clock (MCCK), Command (MCCDA) und Data (MCDA0). Dabei ist die Command-Leitung als Open-Collector ausgeführt und bedarf eines externen Pull-up-Widerstandes von ca. 1 kΩ.

Die Aufgaben des MCI sind der Kommando- und Datentransport von und zur Karte sowie die CRC-Prüfung. Mithilfe des integrierten Direct-Memory-Access-Controllers (DMA) lassen sich Daten sehr effektiv ohne die Beteiligung der CPU auf einer Karte speichern und von einer Karte lesen, somit ist der in den Spezifikationen definierte »Multiple-Block«-Modus zur Datenübertragung realisierbar. Ohne DMA-Übertragung wird der »Single-Block«-Modus verwendet. Die Auswertung der Leitungen, die den Zustand des Schreibschutzschalters und eine eingelegte Karte signalisieren, erfolgt nicht automatisch durch das MCI. Die Anbindung von MMC-Karten beziehungsweise SD-Karten im 1-Bit-Modus sowie SD-Karten im 4-Bit-Modus, ist in Bild 4 und Bild 5 dargestellt. Grundsätzlich können die SD- und MMC-Karten auch über ein SPI-Interface angeschlossen werden, dies soll aber aufgrund der Analogie zur ebenfalls beschriebenen Mikrocontroller-basierten AVR-Lösung hier nicht weiter betrachtet werden.

Zur Übertragung von Kommandos an die SD- oder MMC-Karte stellt das MCI die Register MCI_CMDR (Kommando) und MCI_ARGR (Argument, optional) zur Verfügung. Das Ende der Übertragung kann entweder per Polling abgefragt oder mittels eines Interrupts signalisiert werden. Die Antwort der Karte und ggf. ein Fehlercode werden automatisch in dafür vorgesehene Registern gespeichert.

MMC-Stack unter Linux

Für die Ansteuerung von MMC- und SD-Karten ist unter Linux das MMC-Subsystem zuständig. Bild 6 zeigt seinen Aufbau. Seit der Kernel-Version 2.6.14 werden auch SD-Karten unterstützt. Für die im Folgenden durchgeführten Tests wurde die Version 2.6.17.8 verwendet. Damit SD- und/oder MMC-Karten vom Linux-Kernel angesteuert werden können, muss ein Treiber für das entsprechende Interface des jeweiligen Prozessors vorhanden sein. Die Implementierung entspricht der dargestellten MMC-Host-Schicht und kann entweder auf einer MCI- oder SPI-Schnittstelle basieren, wobei Letz-

Pin	Name	Modus	Beschreibung
1	CS	Input	SPI-Chip-Select
2	DataIn	Input	Daten und Kommandos vom Host zur Karte
3	GND		Masse
4	VDD		Versorgungsspannung
5	CLK	Input	Takt
6	GND		Masse
7	DataOut	Output	Daten und Status von der Karte zum Host
8	-	Floating Input	Beschaltung mit Pull-Up-Widerstand
9	-	Floating Input	Beschaltung mit Pull-Up-Widerstand

Tabelle 4: Pinbelegung im SPI-Modus

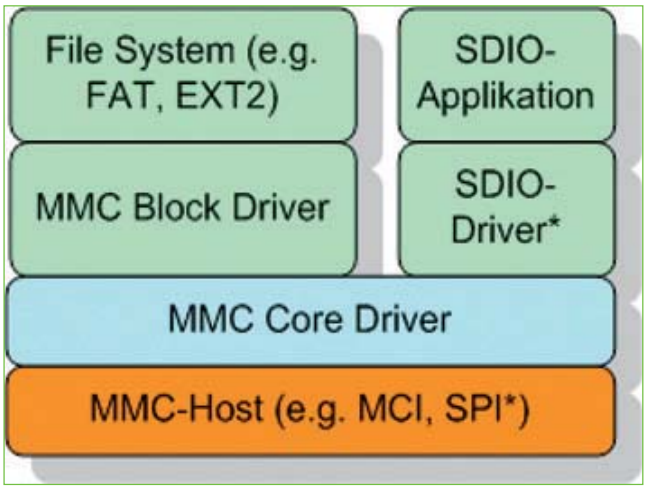


Bild 6 MMC-Stack für Linux (* in Entwicklung)

tere sich gegenwärtig noch in der Entwicklung befindet. Im Falle des AT91RM9200 ist ein Treiber für das MCI-Modul vorhanden. Die »höheren« Schichten sind unabhängig von der verwendeten Hardware.

Der Standard-Linux-Kernel unterstützt bereits einige MMC-Hosts. Beispiele dafür sind das beschriebene MCI von Atmel (`at91_mci.c`), Schnittstellen der Firma Winbond (`wbsd.c`), Intel-XScale-Prozessoren (`pxamci.c`), OMAP-Plattformen von Texas Instruments (`omap.c`), ARM PrimeCell (`mmci.c`) und Au1xxx-Prozessoren von AMD (`imxmmc.c`) sowie das SD-Host-Controller-Interface (SDHCI) der SD-Card-Association (`sdhci.c`). Das SDHCI ist eine standardisierte Host-Spezifikation, die inzwischen von vielen Herstellern verwendet wird. In der Version 2.6.18 wurde der Treiber stark überarbeitet, nachdem die SD-Card-Association Teile der Spezifikation veröffentlicht hatte. Bei Problemen und der Recherche nach noch in Entwicklung befindlichen Treibern empfiehlt sich die Suche in den einschlägigen Mailinglisten [6], [7].

Unter Verwendung der in der MMC-Host-Schicht bereitgestellten Funktionen übernimmt die MMC-Core-Driver-Schicht die Initialisierung und die Kommunikation mit der Karte. Aus Sicht des Anwenders erfolgt die Kommunikation entweder über ein Dateisystem, das einen blockbasierten Zugriff über den MMC-Block-Device-Treiber voraussetzt, oder mithilfe einer SDIO-Applikation über den zugrunde liegenden SDIO-Driver. Letzteres befindet sich allerdings gegenwärtig noch in der Entwicklungsphase. Eine einfache Dokumentation der MMC-Core-Treiber in Form eines Wikis findet sich in [8].

Am Beispiel des AT91RM9200 wurde das

Produkt	Maximaler Takt	Burst-Datenrate
MMC, SPI	20 MHz	2,5 MByte/s
MMC, 1-Bit	20 MHz	2,5 MByte/s
SDC, SPI	25 MHz	3,125 MByte/s
SDC, 1-Bit	25 MHz	3,125 MByte/s
SDC, 4-Bit	25 MHz	12,5 MByte/s

Tabelle 5: Maximale Datenrate MMC/SDC [9]

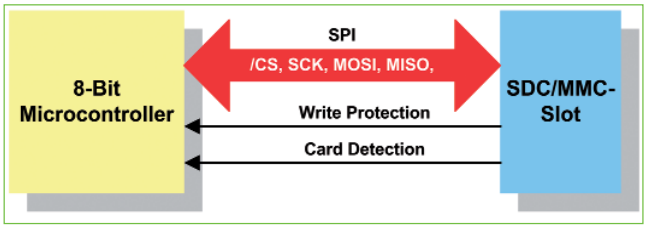


Bild 7: SDC/MMC-Anbindung über SPI

MMC-Subsystem in Betrieb genommen und getestet. In der Kernel-Konfiguration sind für die MMC-Unterstützung verschiedene Makros zu aktivieren. Eine ausführliche Anleitung für die Konfiguration des Kernels zur Unterstützung von MMC- und SD-Karten und des Dateisystems FAT16 sowie für weitere notwendige Arbeitsschritte ist auf der »easyTo-Web«-Homepage verfügbar [10]. Im MCI-Treiber des AT91RM9200 wird gegenwärtig nur der 1-Bit-Modus unterstützt, der 4-Bit-Modus wurde aufgrund von Problemen in der aktuellen Kernel-Version deaktiviert.

Geschwindigkeitstest mit MCI-Interface

Die alleinige Integration der notwendigen Treiber in den Kernel reicht nicht aus, um eine Karte erfolgreich verwenden zu können. Aus Sicht einer Anwendung wird auf die Karte über einen so genannten Geräteknoten zugegriffen. Auch dieser muss erstellt werden, sofern er nicht bereits vorhanden ist. Erst anschließend ist es möglich, die Karte über den Geräteknoten zu partitionieren und zu formatieren. Am Ende dieser Arbeitsschritte lässt sich die Partition in ein Verzeichnis des Root-Dateisystems einhängen (mounten).

Zur Bestimmung der Datenrate wurde eine Applikation geschrieben, die zunächst 50 MByte auf die Partition der Karte schreibt und anschließend wieder liest. Interessant dabei war die Tatsache, dass zunächst betriebssysteminterne Puffermechanismen die Messungen verfälschten. Erst durch die Verwendung der Funktion `fsync()` am Ende eines Schreibzyklus, welche die Übertragung der im Systempuffer gespeicherten Daten erzwingt, wurden die Daten tatsächlich auf der Karte gespeichert. Die Datenrate beim Schreiben betrug zirka 200 KByte/s. Beim Lesen wurden 1,1 MByte/s erreicht. Die niedrige Datenrate für den schreibenden Zugriff ist unter anderem darin begründet, dass der Treiber für den Controller kein »Multi-Block-Write« unterstützt.

SD-Karte am 8-Bit-Controller

Eine Lösung zur Anbindung einer SD-Karte an einen 8-Bit-Mikrocontroller stellt der

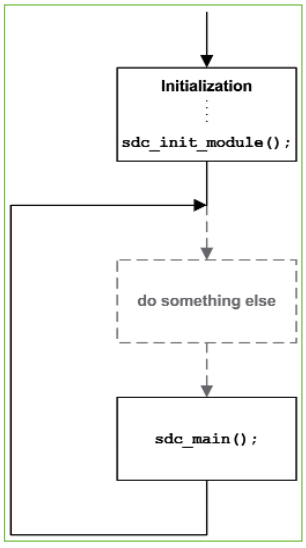


Bild 8: Integration des SDC-Softwaremoduls in die Applikation

Name	Größe in Bits	Beschreibung
Operating Condition Register (OCR)	32	definiert den möglichen Betriebsspannungsbereich zwischen 1,6 V und 3,6 V
Card Identification Register (CID)	128	eindeutige Identifikationsnummer der Karte, Produktname Herstellungsdatum, weitere Angaben
Relative Card Address (RCA)	16	Adresse der Karte am SD-Card-Bus, im SPI-Modus nicht verfügbar
Card Specific Data (CSD)	128	alle relevanten Parameter für die Datenübertragung, Informationen über die Partition, maximale Speicherkapazität
SD Configuration Register (SCR)	64	Informationen über SD-spezifische Erweiterungen (z.B. die unterstütze Busbreite)

Tabelle 6: Über diese Register lassen sich die Karten ansprechen

nächste Abschnitt im Detail vor. Die passende Treibersoftware in C ist als API erhältlich und aufgrund der modularen Struktur auf unterschiedliche Plattformen portierbar. In der hier vorgestellten Lösung erfolgt die Ansteuerung der Karte über die SPI-Schnittstelle.

Diese SPI-Anbindung lässt sich relativ einfach realisieren, da nahezu jeder kleine »8-Bitter« eine taktsynchrone serielle Schnittstelle vorweisen kann. Bild 7 zeigt schematisch die Verbindung zwischen einem Controller und einem SDC/MMC-Slot. Insgesamt werden vier Steuerleitungen und zwei Kontrollleitungen benötigt. Eine detaillierte Beschreibung aller I/O-Pins, einschließlich der Spannungsversorgung, ist in Tabelle 4 zusammengestellt. Alle ungenutzten Pins im SPI-Mode sollten mit Pull-up-Widerständen abgeschlossen sein.

Die Karten arbeiten mit einer Betriebsspannung von bis zu 3,3 V, deshalb muss der Entwickler eines 5-V-Systems entsprechende Vorkehrungen zur Pegelanpassung treffen. Um die Position des Schreibschutzschalters zu ermitteln, wird ein normaler Controllereingang verwendet. Die Einschuberkennung einer Karte gestaltet sich meist identisch.

Software für den »ATmega128«

Als Entwicklungsbasis für das hier beschriebene API diente der »ATmega128«, ein Vertreter der 8-Bit-AVR-Familie der Firma Atmel. Tabelle 7 zeigt die Trennung der einzelnen Softwaremodule bezüglich ihrer Aufgabe und Funktion in einer Zusammenfassung. So stellt das API beispielsweise Standardroutinen bereit, um Daten lesen, schreiben oder löschen zu können. Die Debug-Schnittstelle erlaubt über eine Kommandostruktur, bestimmte Funktionen im Vorfeld zu testen oder im laufenden Prozess zusätzliche Debug-Information auf einem PC-Terminalprogramm darzustellen. In den Files `project.h` und `hardware.c` werden projektspezifische Makros und Funktionen für die Behandlung der Ein- bzw. Ausgänge und der SPI-Schnittstelle definiert.

Dateiname	Modulbeschreibung
sdcard.c / sdcard.h sdcard_debug.h	API zur Ansteuerung einer MMC/SDC
flashcard_debug.c	Debug-Schnittstelle für Funktionstests
crc.c / crc.h	CRC-Berechnungsfunktionen
project.h	Projektspezifische Makrodefinitionen
hardware.c	Controllerspezifische Hardwarefunktionen

Tabelle 7: Überschicht der Softwaremodule

Eine mögliche Implementierung des Softwaremoduls stellt der Ablauf in Bild 8 dar. Zu Beginn initialisiert die Funktion `sdc_init_module()` alle notwendigen I/O-Pins. Der Aufruf der Funktion `sdc_main()` sollte innerhalb einer `while`-Schleife im Hauptprogramm erfolgen, um eine Einschuberkennung sicherzustellen.

Der vereinfachte Ablaufgraph in Bild 9 zeigt, dass die Erkennung einer Karte den Initialisierungsprozess einleitet. Nach dem Einschalten der Betriebsspannung befindet sich die Karte zunächst im »Idle«-Status und muss in den »SPI«-Modus überführt werden. Zuvor ist jedoch eine Verzögerung von 74 mal 8 Taktperioden nötig, um eine korrekte Funktionsweise zu garantieren. Ein typischer Wert von 74 Bytezyklen scheint für manche Karten sehr knapp bemessen zu sein und sollte sicherheitshalber etwas großzügiger gewählt werden.

Anschließend erfolgt das Senden des Reset-Kommandos `CMD0`. Mit der Bestätigung dieses Kommandos durchläuft die Speicherkarte eine interne Initialisierungsprozedur. Durch das zyklisch Pollen des Kommandos `CMD1` wird detektiert, ob die Initialisierungsphase abgeschlossen ist.

Die folgende Softwareerkennung dient der notwendigen Unterscheidung des Kartentyps, die mit der Auswertung des »OCR«- und »CSD«-Registers endet und alle notwendigen Parameter temporär speichert. Ab diesem Zeitpunkt steht einem Zugriff

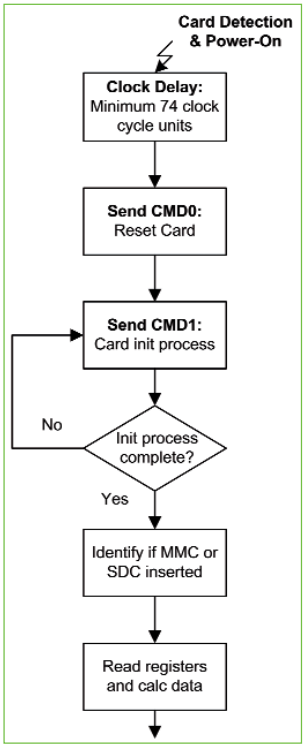
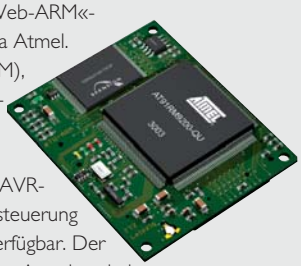


Bild 9: Ablauf der Initialisierung einer Speicherkarte

Die Testumgebung im Detail

Für die Ansteuerung über ein integriertes Host-Interface wurde das »easyToWeb-ARM«-Modul verwendet. Es basiert auf dem AT91RM9200 mit ARM9-Kern der Firma Atmel. Das Board ist mit statischem (NOR-Flash) und dynamischem Speicher (SDRAM), einem Ethernet-Switch, Echtzeituhr und EEPROM ausgestattet. Ein Embedded-Linux-System mit Linux-Entwicklungsumgebung (VMware-Image) wird mitgeliefert, ebenso eine Lösung zum initialen Programmieren des Flash-Speichers. Für die Ansteuerung über einen 8-Bit-Mikrocontroller wurde ein easyToWeb-AVR-Modul mit einem ATmega128 der Firma Atmel verwendet. Der Code zur Ansteuerung der SD-Karten unterliegt der LGPL-Lizenz und ist als Subversion-Repository verfügbar. Der Code ist für den CodevisionAVR-C-Compiler und die AVR-Mikrocontroller von Atmel optimiert. Mehr Informationen sowie ein Forum zur Software und zum Artikel gibt es unter **www.easytoweb.net**



auf das Speichermedium nichts mehr im Wege. Aufgrund der eingeschränkten Taktkontrolle im Anlaufmoment einer MMC-Karte kann die Frequenz jetzt auf den Maximalwert erhöht werden. Die Kommunikation zwischen einem Host und einer Karte im SPI-Mode basiert auf einem Master-Slave-Prinzip. Jedes empfangene Kommando wird entsprechend bewertet und durch eine Antwort quittiert. Das Format eines Kommandos, bestehend aus 6 Byte, wird von einem Start- bzw. Stoppbit umrahmt und beinhaltet das Kommando selbst, ein Argument für Parameterübergaben und ein CRC-Feld. Je nach Funktion sind die festgelegten Kommandos in unterschiedliche Klassen unterteilt. Bild 10 stellt die Abläufe eines Schreib- und Lesevorgangs im Multi-Block-Modus in Diagrammen gegenüber. Zur Vereinfachung verzichteten die Autoren auf die Darstellung von Kontrollmechanismen und detaillierten Fehlerbehandlungen.

Lesen und Schreiben von Daten

Im Falle einer Leseoperation kann die Blocklänge üblicherweise zwischen einem und 512 Byte variieren. Die Grundeinstellung, die für das Erreichen eines maximalen Datendurchsatzes zu empfehlen ist, beträgt 512 Byte. Mit dem Kommando `CMD18` teilt die Applikation der Karte die Startadresse für das Lesen eines Blocks mit. Durch Einlesen des Startbytes synchronisiert sich die Software auf den Datenstrom. Nach der Übertragung eines Blocks wird ein zusätzlicher Wartezyklus eingeschoben, um die Operation zu beenden. Im Allgemeinen entspricht ein Wartezyklus der Übertragungsdauer eines Bytes, also jeweils acht Taktperioden.
Bezogen auf eine Schreiboperation beträgt die Blocklänge stets 512 Byte, soweit im »CSD«-Register nichts anderes spezifiziert ist. Unmittelbar vor der Ausführung wird der Karte die Anzahl der vorher zu lö-

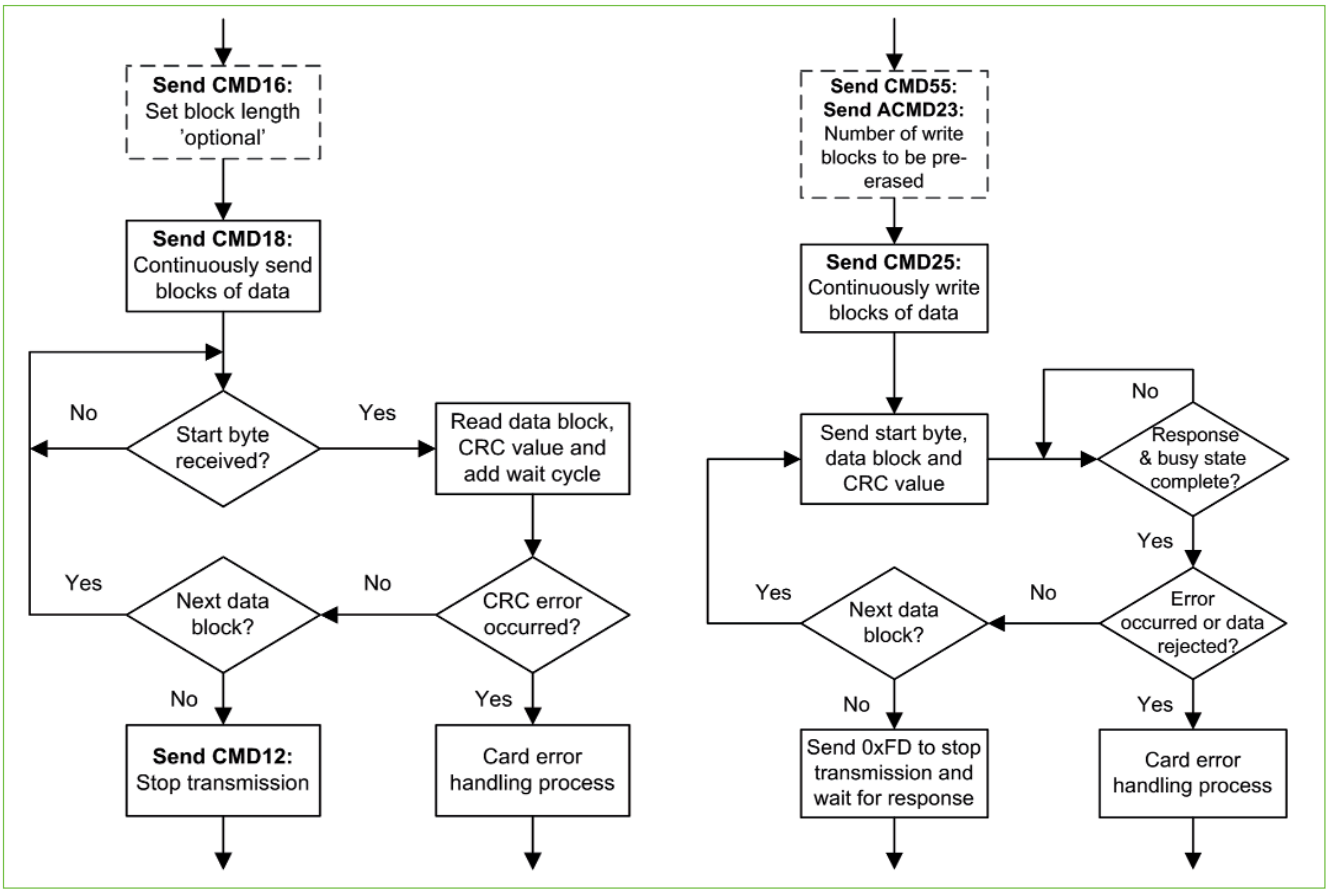


Bild 10: Ablaufdiagramm von »Daten von der Karte lesen« (links) und von »Daten auf die Karte schreiben« (rechts)

schenden Blöcke übermittelt. Die eigentliche Operation beginnt mit der Bekanntgabe der Blockstartadresse durch das Kommando `CMD25`. Nachdem alle relevanten Daten eines Blockes übertragen und geprüft wurden, bestätigt die Karte den Empfang und treibt anschließend die MISO-Leitung auf Low-Pegel, bis der Prozess beendet ist. Im Falle eines Fehlers besteht die Möglichkeit, über ein applikationsspezifisches SDC-Kommado die Anzahl der bis dahin erfolgreich geschrieben Blöcke zurückzulesen.

Quellen

[01] SD Card Association; www.sdcard.org/
[02] MultiMediaCard Association: www.mmca.org/
[03] SD Cards und MultiMediaCards: c't 06/2005.
[04] SanDisk Cooperation: SD Card Product Manual Version 2.2
[05] Atmel Corporation: www.atmel.com/
[06] ARM-Linux-Project: http://www.arm.linux.org.uk/
[07] Linux Kernel Mailing List: http://www.lkml.org/
[08] MultiMediaCard OpenSource Flash-Storage: http://mmc.drzeus.cx/wiki/
[09] SanDisk Cooperation: Host Design Considerations: NAND MMC and SD-based Products
[10] easyToWeb-Projekt: www.easytoweb.de
[11] All Memory Cards: http://www.allmemorycards.com/

Geschwindigkeitstest mit SPI-Interface

Beide Operationen, sowohl das Lesen als auch das Schreiben im Multi-Block-Modus, müssen mithilfe einer Stoppbedingung abgeschlossen werden. Ist das Prinzip des Pollens nicht die erste Wahl, erlauben die Parameter im CSD-Register, zusammen mit der verwendeten SPI-Taktfrequenz, die Berechnung der maximalen Schreib- oder Lesezeit eines Blocks. Erfahrungen, gerade mit älteren Kartenmodellen, haben gezeigt, dass bei eventuellen Problemen zusätzliche Wartezyklen berücksichtigt werden sollten. Meist liefert der Hersteller die entsprechenden Angaben im Datenblatt mit.
Um die Übertragungsrate zu messen, wurden 1000 Blöcke zu je 512 KByte von der Karte gelesen beziehungsweise auf die Karte geschrieben. Dadurch wurde sichergestellt, dass auch wirklich die Dauertransferrate gemessen wird und der karteninterne Buffer die Messung nicht verfälscht. Lesend und schreibend erreichte der Testaufbau 85 KByte/s. Hier wirkt sich die Beschränkung der SPI-Geschwindigkeit auf 8 MHz und die Berechnung der CRC-Werte in Software limitierend aus. (cg)

FTZ Leipzig
Telefon 03 41/30 76 11 36
www.easytoweb.de

Spoerle
Telefon 03 61/78 93 56 15
www.spoerle.de