



Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Hardwarenahe Softwareentwicklung

ARM, Cortex-M7, STM32H743, Leguan

V5.1, ©2023 roger.weber@bfh.ch

Lernziele

Sie sind in der Lage:

- ▶ Die Architekturen von ARM, Cortex-Mx und STM32H743 zu erklären.
- ▶ Register und Betriebsmodi eines Cortex-Mx zu erklären.
- ▶ Den Schaltplan des Leguan-Boards mit einem STM32H743 zu analysieren.



Inhaltsverzeichnis

1. ARM Übersicht und Architektur
2. Cortex-Mx Übersicht und Architektur
3. STM32H743
4. Core Register, Program Status Register, Statusbits
5. Datenformate, Speichermodell
6. Leguan-Board

ARM Übersicht und Architektur

Vorteile der ARM-Architektur

Alles, was sich Entwickler*innen wünschen:

- ▶ geringer Leistungsaufnahme für hohe Rechenleistung
- ▶ preiswert
- ▶ viele Silizium-Hersteller
- ▶ in der Industrie weit verbreitet
- ▶ viele und gute Tools
- ▶ gute Dokumentation
- ▶ guter Support

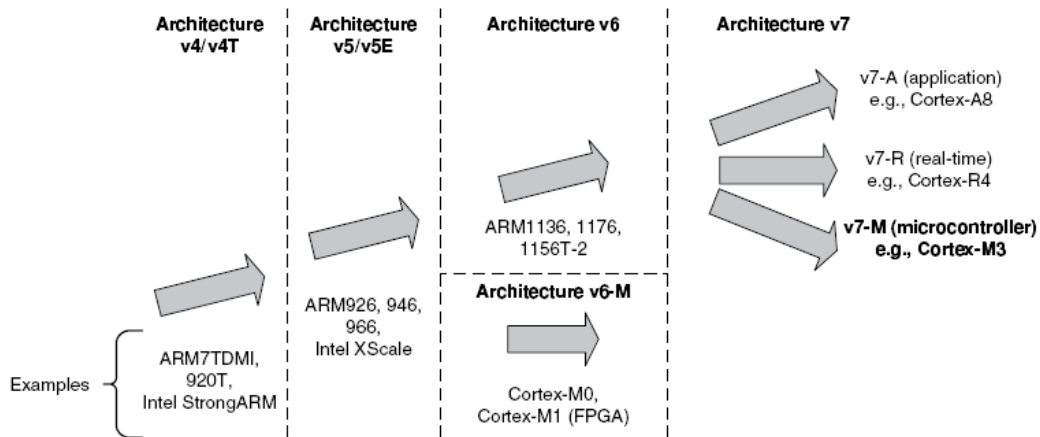
-keine Wirklichen Nachteile

Eigenschaften der ARM-Architektur

Reduced Instruction set

- ▶ Typische RISC Architektur:
 - ▶ Grosse Registerbank
 - ▶ Load/Store Architektur
 - ▶ Orthogonaler Befehlssatz
- ▶ ARM-spezifische Eigenschaften
 - ▶ ALU + Barrel Shifter in 1 Instruktion
 - ▶ Bedingte Befehlsausführung
 - ▶ 32-Bit ARM Instruktionen sowie 16-Bit Thumb(-2) Instruktionen
- ▶ Effizientes Interrupt-Subsystem

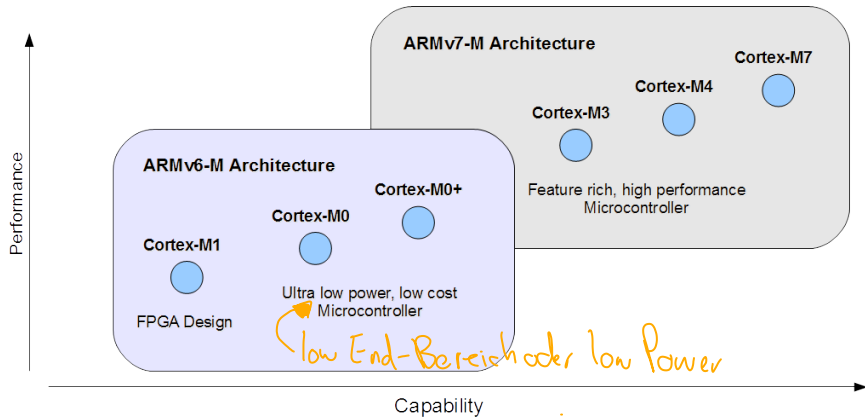
Versionen der ARM-Architektur



The Evolution of ARM Processor Architecture.

Cortex-Mx Übersicht und Architektur

Die ARM Cortex-Mx Microcontroller-Familie



Quelle: Joseph Yiu

Vorteile

- ▶ Technische Vorteile:
 - ▶ **Low-Power**: Energieeffizientes Core-Design
 - ▶ Einfaches, gut konfigurierbares **Interrupt-System**
 - ▶ Einfach **skalierbar**
 - ▶ Optimiert für die **Programmiersprache C**
- ▶ Organisatorische Vorteile:
 - ▶ Angebot an **Chips und Tools** auf dem Markt
 - ▶ Viele **OS und Libraries** einsetzbar
- ▶ Finanzielle Vorteile:
 - ▶ Hohe **Rechenleistung** für tiefen Preis
 - ▶ **Verbreitet**, Wissen vorhanden

Architekturen

M0 noch Weniger als M3

▶ Cortex-M3:

- ▶ 3-stufige Pipeline
- ▶ Harvard-Architektur on-chip
- ▶ Linearer Adressraum, 32-Bit Adressen (4GB Memory Space)
- ▶ NVIC Interrupt Controller

▶ Cortex-M4 unterstützen zusätzlich folgende Funktionen:

- ▶ SIMD Single Instruction Multiple Data 1 Instruktion 4x 8bit wird selten verwendet
- ▶ MAC (Multiply-Accumulate)
- ▶ Single Precision FPU optional
- ▶ MPU optional ca. 200 MHz

▶ Cortex-M7 unterstützen zusätzlich folgende Funktionen:

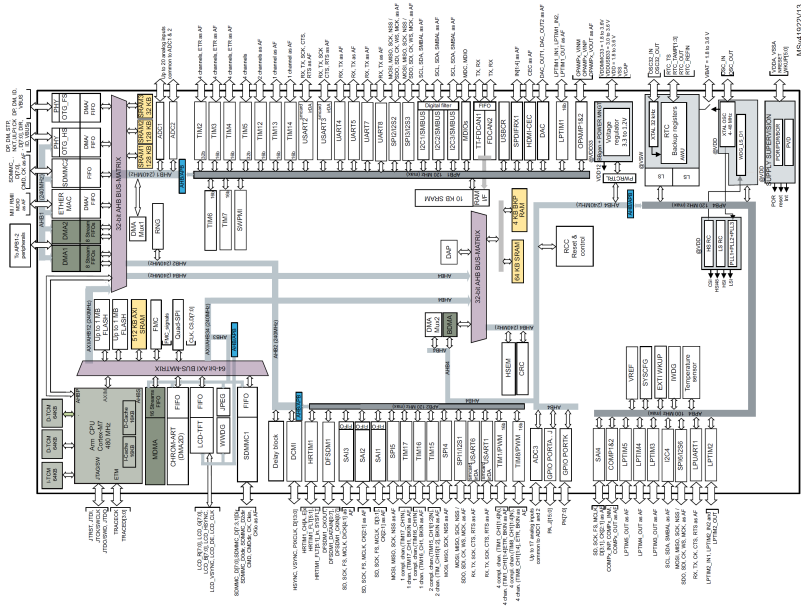
- ▶ L1 Cache
- ▶ Single Precision FPU, optional double precision
- ▶ 6-stufige Pipeline
- ▶ MPU optional
- ▶ Taktfrequenz bis 1 GHz

STM32H743

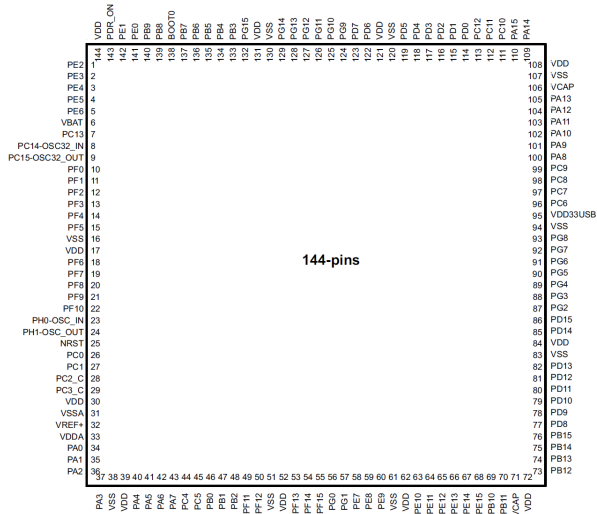
Eigenschaften

- ▶ ARM Cortex-M7 32-bit CPU und MPU
- ▶ Clock bis 480 MHz, 1027 DMIPS/ 2.14 DMIPS/MHz
- ▶ Bis 2 MByte Flash
- ▶ Bis 1 MByte SRAM
- ▶ Double Precision FPU
- ▶ L1 Cache mit 16Kb data und 16kB instruction cache
- ▶ Memory Controller für externe Speicher: SRAM, SDRAM, NOR/NAND Flash
- ▶ 3x16-bit, 3,6 MSPS A/D-Wandler
- ▶ 2x12-bit D/A-Wandler (1 MHz)
- ▶ 4 DMA Controller
- ▶ Bis 22 Timers: 16-bit und 32-bit, Watchdog, RTC
- ▶ Debugging: Serial Wire Debug (SWD) & JTAG, 4-Kbyte Embedded Trace Buffer
- ▶ Bis zu 168 I/O Ports (GPIO) *General Purpose Input and Output*
- ▶ 4 x I2C, 4 USART / 4 UART, 6 SPI, 2 x CAN, USB 2.0, 10/100 Mbit/s Ethernet

Blockdiagramm



Pinbelegung



Quelle: STM32H743 Datasheet

Core Register, Program Status Register,
Statusbits

Registermodell

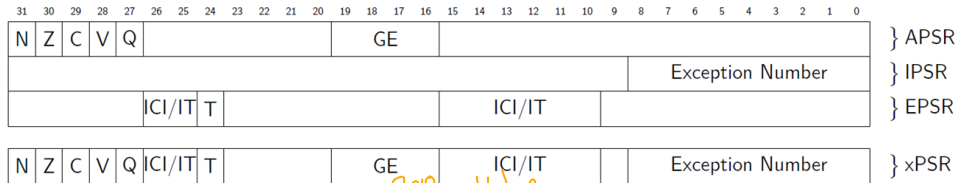
- ▶ Register bilden die Schnittstelle zwischen der Hardware (CPU, Hardware-Peripherie) und der Software.
- ▶ Registerbank bei der CPU: 16 Core Register (r0 bis r15) + xPSR + Exception-Masks. Zugriff nur über Assembler möglich.
- ▶ SFR (Special Function Registers) Memory-Mapped für den Zugriff auf die Hardware-Peripherie.

Core Registers, gemäss ARM Architecture Procedure Call Standard

Register	Beschreibung
r0 bis r12	Applikation, gemäss AAPCS
SP (r13)	Stack Pointer (SP_main, SP_process)
LR (r14)	Link Register <i>Arm Spezifisch</i>
PC (r15)	Program Counter
APSR	Application Program Status Register
IPSR	Interrupt Program Status Register
EPSR	Execution Program Status Register
PRIMASK	Exception Mask Register
FAULTMASK	Fault Mask
BASEPRI	Base Priority Mask
CONTROL	special-purpose CONTROL register

Program Status Register (PSR)

► Darstellung des aktuellen Zustands des Prozessors



- Statusbits (Condition Flags): N, Z, C, V, Q
- Greater-Than or Equal Flags (GE)
- If-Then Instruction Status bits (ICI/IT)
- Thumb-State (T) (immer 1)
- Exception Number
- $xPSR = APSR \mid IPSR \mid EPSR$

Statusbits (Condition Flags)

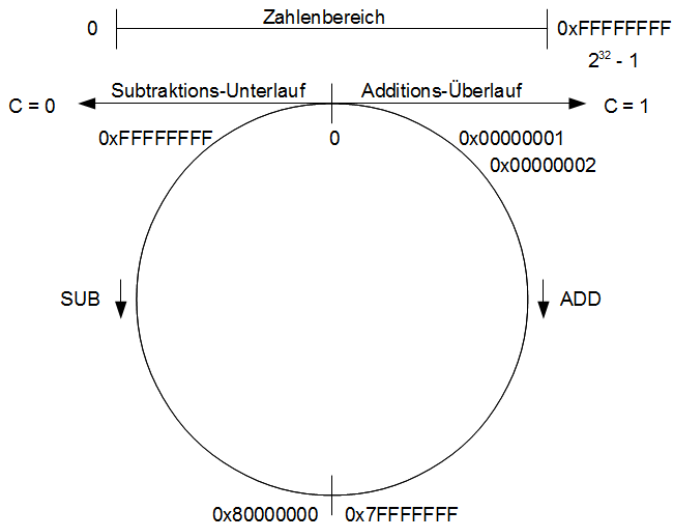
- ▶ Werden bei Vergleichen oder Operationen durch die ALU aktualisiert (S-Suffix).
- ▶ Bedingte Sprungbefehle werden immer abhängig von den Statusbits durchgeführt.
- ▶ Beispiele:

CMP	r0,#0	@ vergleicht r0 mit 0
BEQ	label1	@ bedingter Sprung falls r0 == 0
MOVS	r0,r1	@ r1 → r0, Statusbits setzen
BEQ	label2	@ bedingter Sprung falls r0 == 0

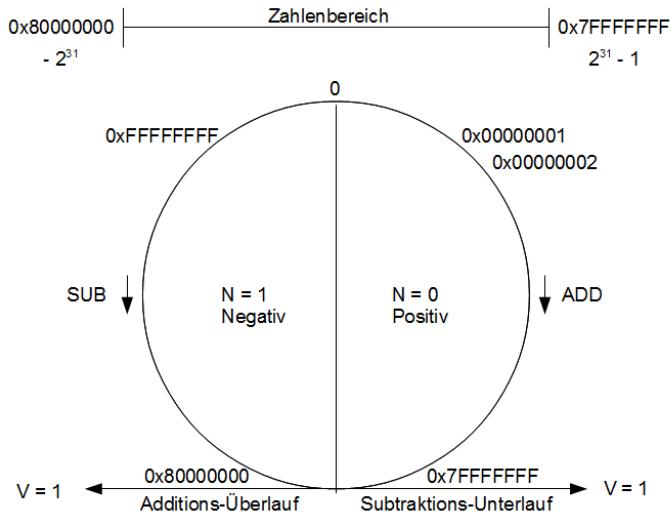
Statusbits (Condition Flags)

Flag	Flag Name	Beschreibung
N	Negative	Vorzeichenbehaftete Zahlen (signed): N-Bit als Vorzeichen interpretiert. negative Zahlen → N-Bit gesetzt (1) positive Zahlen → N-Bit gelöscht (0).
Z	Zero	Z-Bit gesetzt a) Resultat einer ALU-Operation war Null b) Vergleichen von zwei identischen Werten
C	Carry	C-Bit Additionen: bei Überlauf gesetzt Subtraktion: bei Unterlauf gelöscht Shift-Operationen: letztes Bit der Operation
V	Overflow	V-Bit gesetzt: Überlauf von vorzeichenbehafteten Zahlen
Q	Saturation	Q-Bit gesetzt: Überlaufbegrenzung (Saturation)

Zahlenkreis vorzeichenloser Dualzahlen



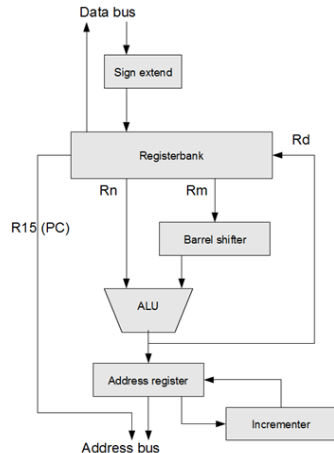
Zahlenkreis vorzeichenbehafteter Dualzahlen



Datenformate, Speichermodell

Datenformate

- ▶ Daten werden im “little-endian Format“ abgelegt (tieferwertige Bytes auf tieferen Adressen).
- ▶ ARM-Prozessoren ab Architektur Version v4 unterstützen folgende Datenformate:
 - ▶ signed und unsigned Byte (8 Bit, char)
 - ▶ signed und unsigned Halfword (16 Bit, short int)
 - ▶ signed und unsigned Word (32 Bit, long int)
- ▶ ALU-Operationen verarbeiten immer 32-Bit.
- ▶ Datentransfer-Befehle (Load und Store) mit Byte, Halfword und Word → “Sign extend“.



Byte-Speichermodell

Adresse

n	Byte 0
n + 1	Byte 1
n + 2	Byte 2
n + 3	Byte 3

Halfword-Speichermodell

Adresse

n	Halfword 0	low Byte Halfword 0
		high Byte Halfword 0
n + 2	Halfword 1	low Byte Halfword 1
		high Byte Halfword 1

Word-Speichermodell

Adresse

n

Word 0

lowest Byte Word 0

$n + 4$

Word 1

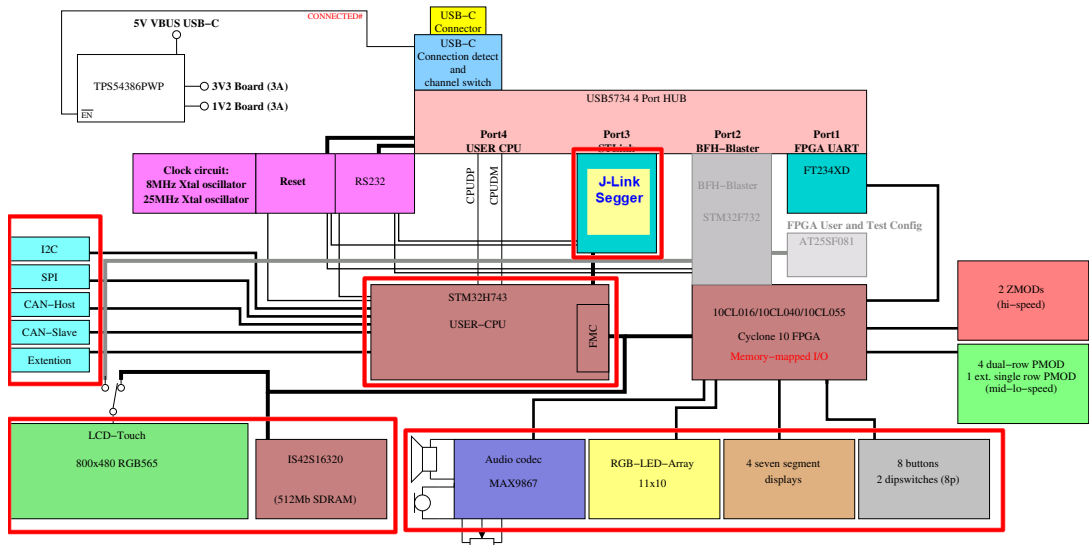
highest Byte Word 0

lowest Byte Word 1

Highest Byte Word

Leguan-Board

Blockschaltbild Leguan



Memory-Map Cortex-Mx

Gesamter Adressbereich		Peripherie	
0xFFFF FFFF	512 Mbyte Block 7 Cortex-M4's internal peripherals	0x5FFF FFFF	Reserved
0xE000 0000		0x5006 0C00	
0xDFFF FFFF	512 Mbyte Block 6 Not used	0x5006 0BFF	AHB2
0xC000 0000		0x5000 0000	
0xBFFF FFFF	512 Mbyte Block 5 FSMC registers	0x4FFF FFFF	Reserved
0xA000 0000		0x4008 0000	
0x9FFF FFFF	512 Mbyte Block 4 FSMC bank 3 & bank 4	0x4007 FFFF	AHB1
0x8000 0000		0x4002 0000	
0x7FFF FFFF	512 Mbyte Block 3 FSMC bank 1 & bank 2	0x4001 FFFF	Reserved
0x6000 0000		0x4001 6C00	
0x5FFF FFFF	512 Mbyte Block 2 Peripherals	0x4001 6BFF	APB2
0x4000 0000		0x4001 0000	
0x3FFF FFFF	512 Mbyte Block 1 SRAM	0x4000 FFFF	Reserved
0x2000 0000		0x4000 8000	
0x1FFF FFFF	512 Mbyte Block 0 Code	0x4000 7FFF	APB1
0x0000 0000		0x4000 0000	

Memory-Map Leguan und STM32H743

0xFFFF FFFF	Reserviert	0xDFFF FFFF	Reserviert
0xE000 0000		0xD000 0000	
0xDFFF FFFF	Leguan Peripherie	0xCFFF FFFF	Reserviert
0xC000 0000		0xC800 0800	
0xBFFF FFFF	Reserviert	0xC800 07FF	RGB Array, pixel based control
0xA000 0000		0xC800 0600	RGB Array, line based control
0x9FFF FFFF	Reserviert	0xC800 05FF	
0x8000 0000		0xC800 0400	
0x7FFF FFFF	Reserviert	0xC800 03FF	Reserviert
0x6400 0000		0xC800 0040	
0x601F FFFF	Leguan Ext. SDRAM, 32MB (AS4C16M16SA-6TAN)	0xC800 003F	
0x6000 0000		0xC800 0020	Button, Joystick, Dipswitch
0x37FF FFFF	Reserviert	0xC800 001F	7-Segment Anzeige
0x3408 0000		0xC800 0010	
0x3800 FFFF	SRAM1, 64kB (STM32H743)	0xC800 000F	Reserviert
0x3800 0000		0xC800 0002	
0x37FF FFFF	Reserviert	0xC800 0001	FPGA Reset
0x3408 0000		0xC800 0000	
0x3004 7FFF	SRAM1, 288kB (STM32H743)	0xC7FF FFFF	Reserviert
0x3000 0000		0xC000 0004	
0x2FFF FFFF	Reserviert	0xC000 0003	LCD Interface
0x2408 0000		0xC000 0000	
0x2407 FFFF	SRAM, 512kB (STM32H743) AXI RAM		
0x2400 0000			
0x2003 FFFF	Reserviert		
0x2002 0000			
0x2001 FFFF	SRAM, 128kB (STM32H743) DTCM (Data Tightly Coupled Memory)		
0x2000 0000			
0x1FFF FFFF	Reserviert		
0x0020 0000			
0x081F FFFF	Flash, 1MB (STM32H743)		
0x0800 0000			
0x07FF FFFF	Reserviert		
0x0001 0000			
0x0000 FFFF	SRAM 64 kB (STM32H743) ITCM (Instruction Tightly Coupled Memory)		
0x0000 0000			

Memory-Map Leguan Peripherie

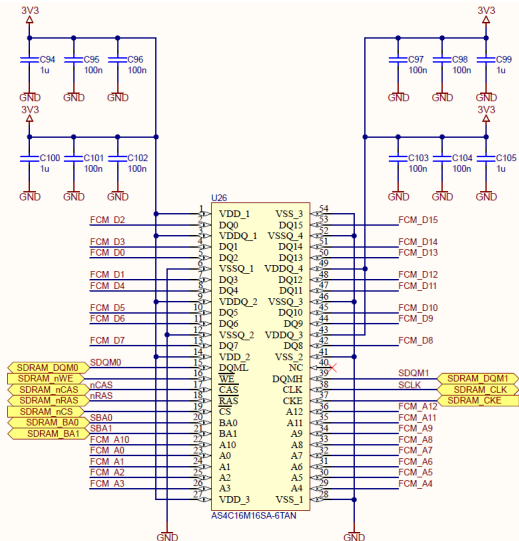
Adressbereiche Peripherie:

- ▶ 0xC000 0000 bis 0xC000 0003: LCD Interface
- ▶ 0xC800 0000 bis 0xC800 0001: FPGA Reset
- ▶ 0xC800 0010 bis 0xC800 001F: 7-Segment Anzeige
- ▶ 0xC800 0020 bis 0xC800 003F: Button, Joystick, Dipswitch
- ▶ 0xC800 0400 bis 0xC800 05FF: RGB Array, line based
- ▶ 0xC800 0600 bis 0xC800 07FF: RGB Array, pixel based

Zudem über GPIO angeschlossen:

- ▶ GPIO-In: Button ganz rechts: PD11
- ▶ GPIO-Out: Dezimalpunkt 7-Segment ganz rechts: PB15

Externes SDRAM



Signal	Kurzbeschreibung
/CS	Chip-Select (Input): Aktiv „low“ um den SDRAM-Baustein zu aktivieren.
/WE	Write Enable (Input): Aktiv „low“ wenn Daten ins SDRAM geschrieben werden.
/RAS	Row Address Strobe (Input), auf dem Adressbus befindet sich die Row-Address.
/CAS	Column Address Strobe (Input), auf dem Adressbus befindet sich die Column-Address.
CLK	Clock (Input)
CKE	Clock Enable (Input)
DQx	Datenleitungen (Input / Output)
Ax	Adressleitungen (Input)
BAx	Bank Select (Input)
DQML	Lower Byte, Input/Output Mask
DQMH	Upper Byte, Input/Output Mask