Softwareentwicklung für eingebettete Systeme

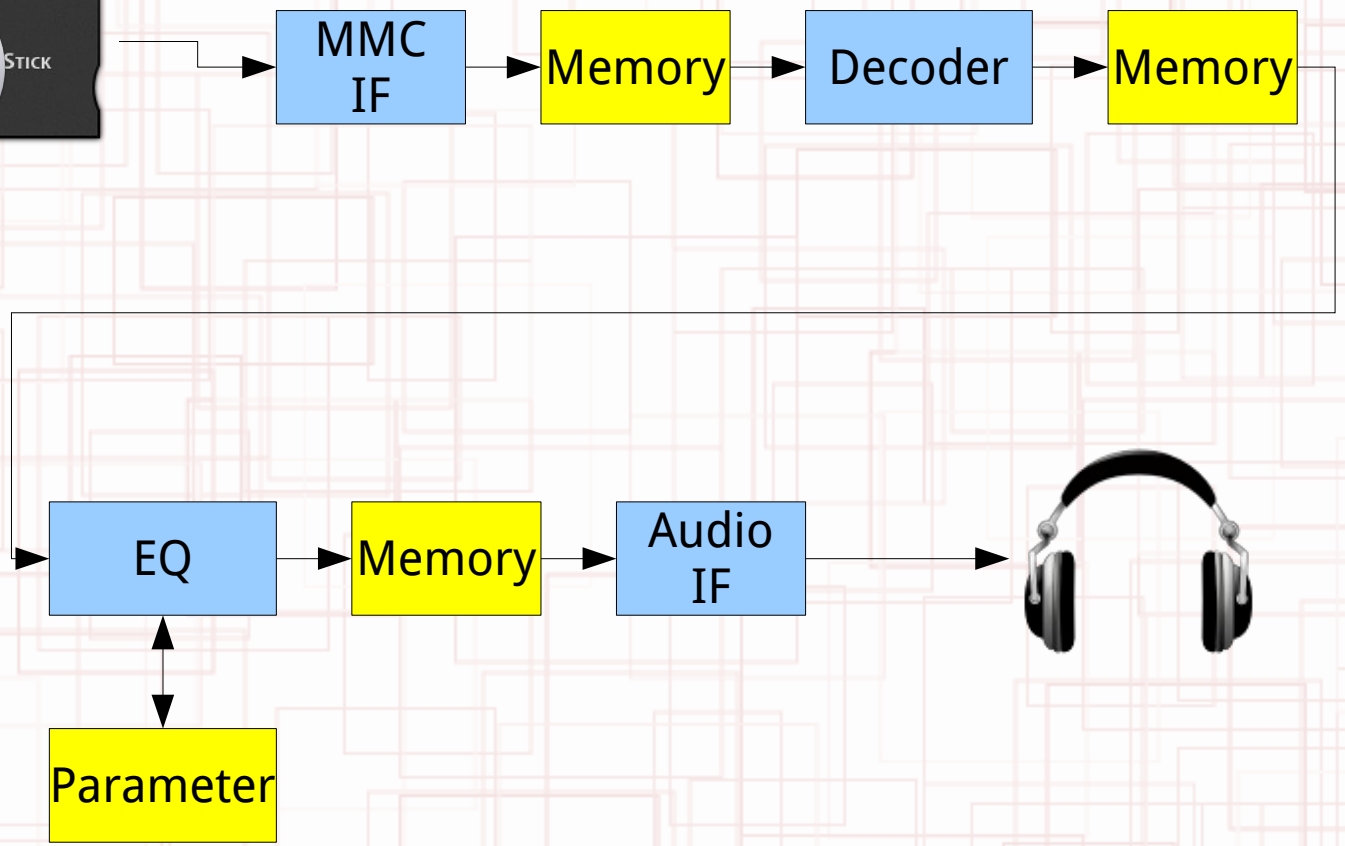
Bachelorstudiengang Technische Informatik
Hochschule Pforzheim

Vorlesung 2
Sommersemester 2014

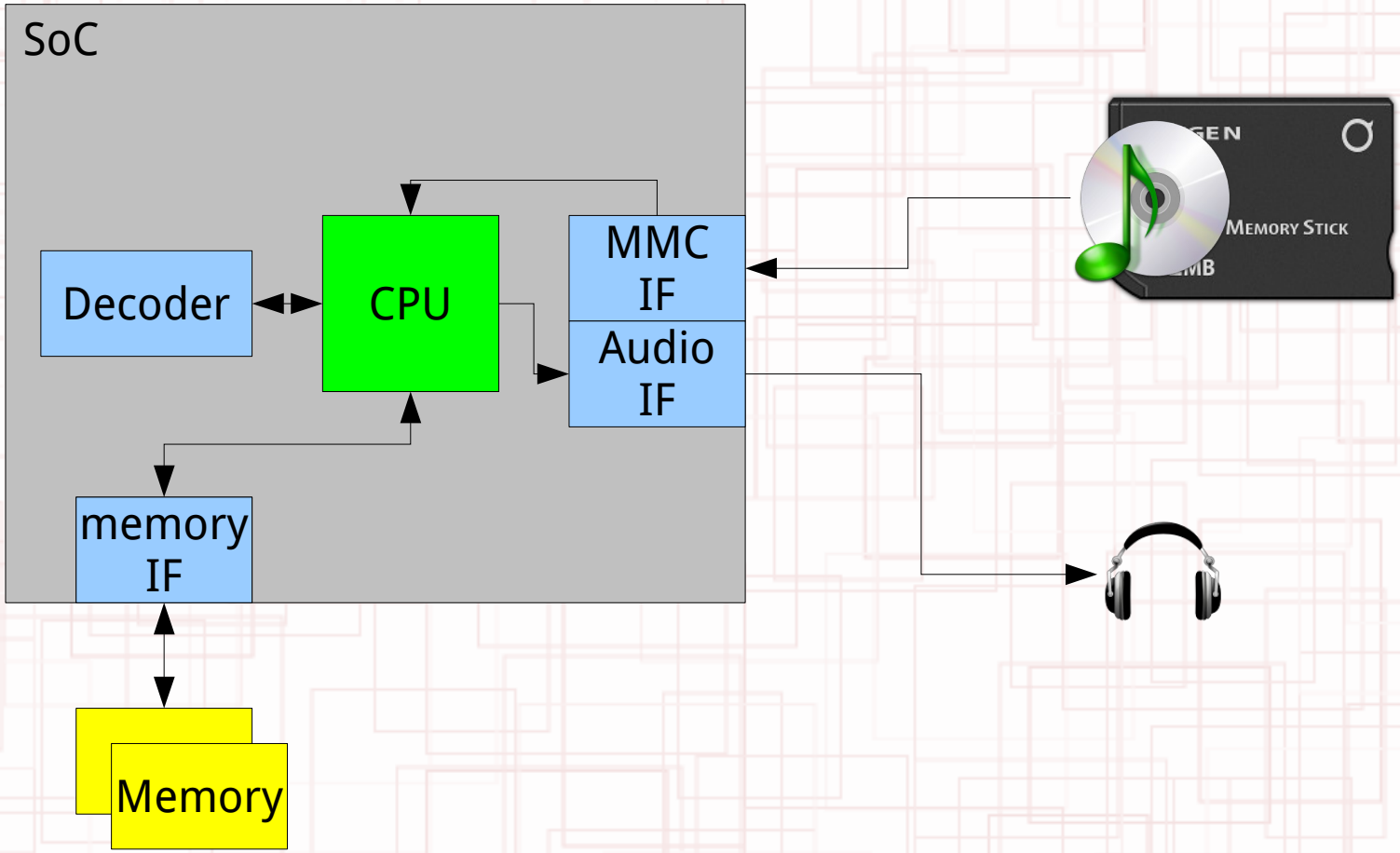
Dipl.-Ing.(FH) Marc Jüttner

MP3-Player

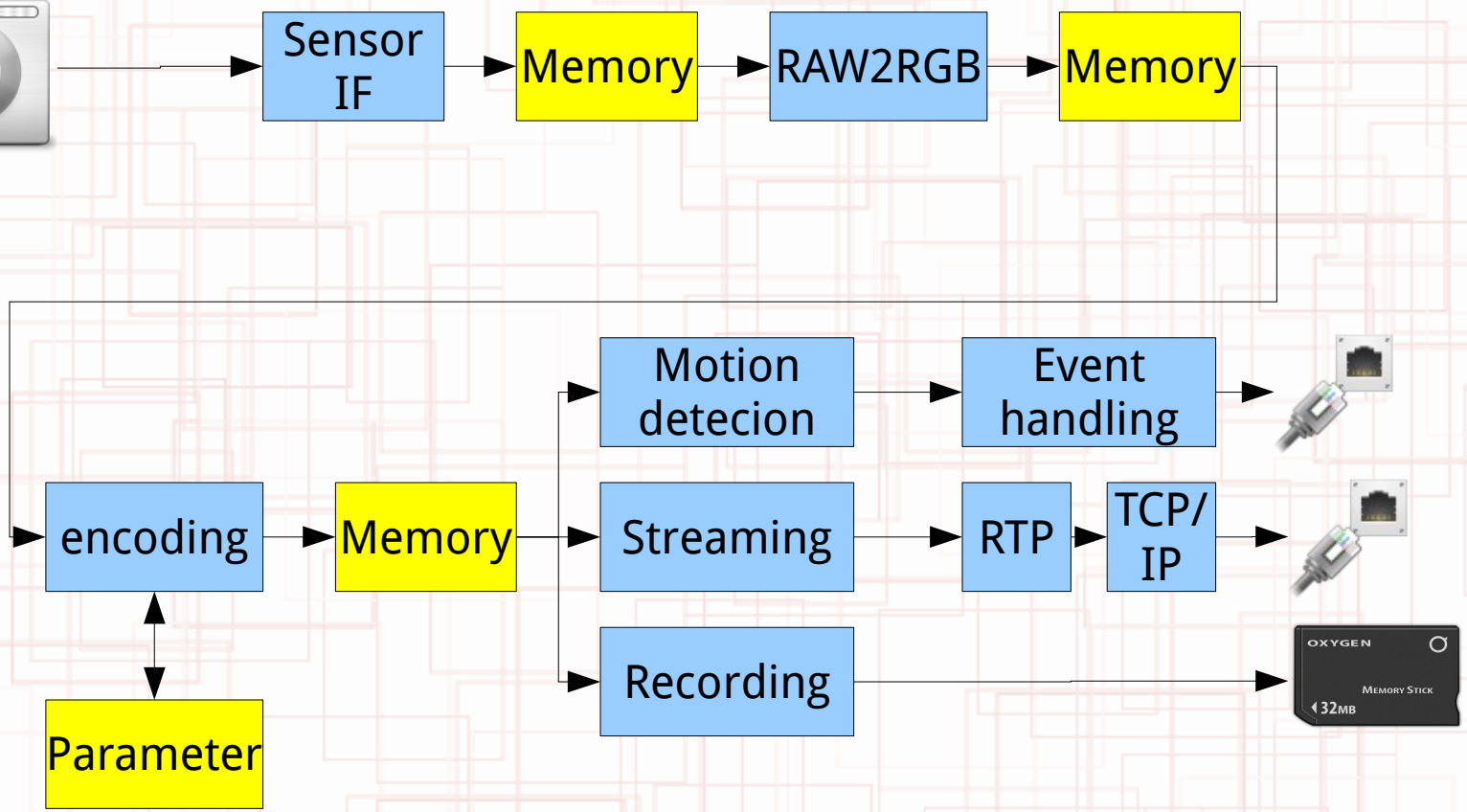

OXYGEN
STICK


MMC IF
Memory
Decoder
Memory
EQ
Memory
Audio IF
Parameter

MP3-Player


SoC
Decoder
CPU
MMC IF
Audio IF
memory IF
Memory
Memory Stick

IP-Überwachungskamera



Funktionsweise

- Lesen von Daten vom Datenträger
  - Data stream, regelmäßige Datenpakete
- Kopieren in den Arbeitsspeicher
- Übergabe an den Decoder
  - MP3
- Kopieren in den Arbeitsspeicher
- Ausgabe auf Lautsprecher/Kopfhörer

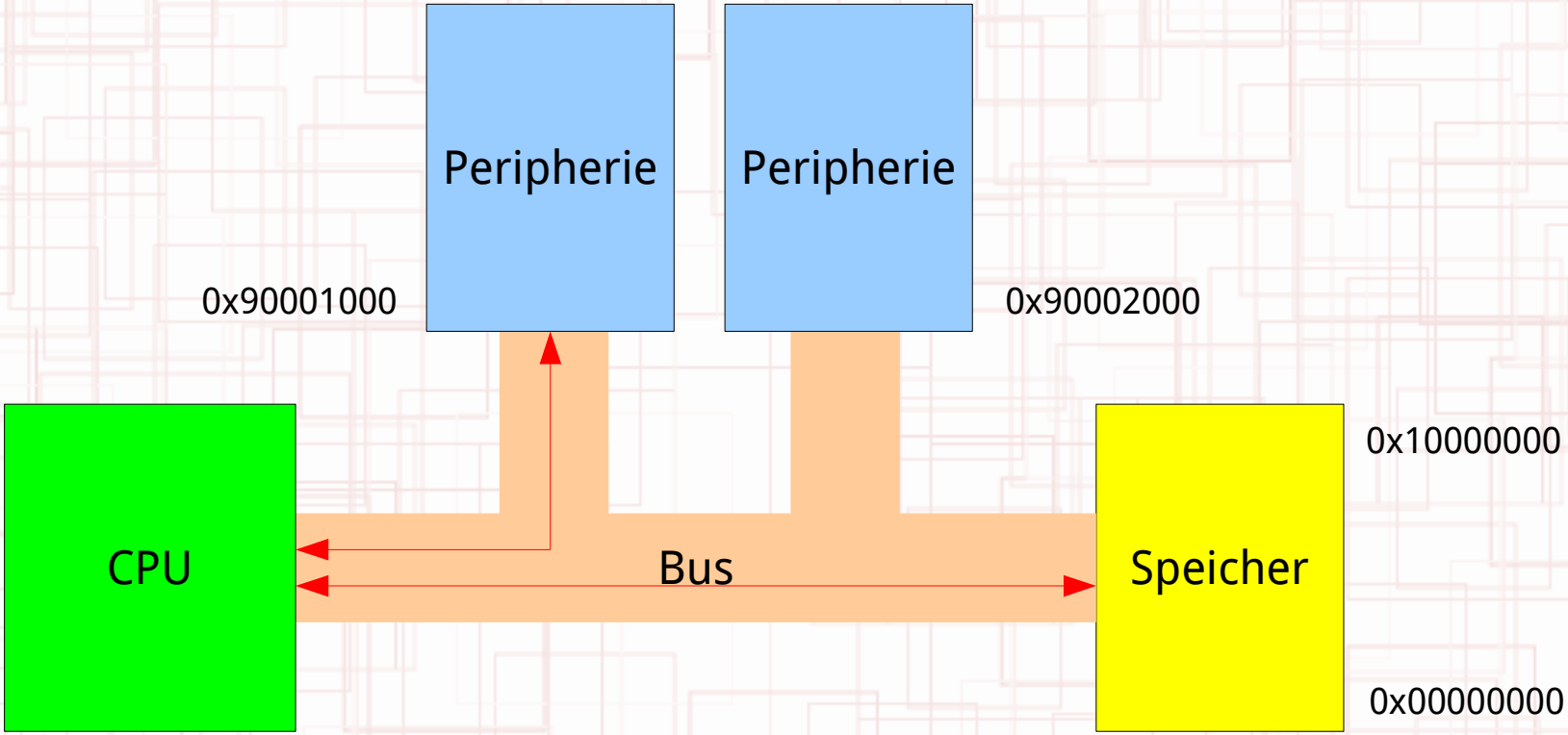
Kopieren von Daten…

- Peripherie ↔ Speicher
  - Lesen und Schreiben mittels Prozessorbefehlen
- Speicher ↔ Decoder
  - Lesen und Schreiben mittels Prozessorbefehlen
- Load und Store benötigen Prozessorzeit
  - Ungünstig bei großen Datenmengen

Gesamtsystem

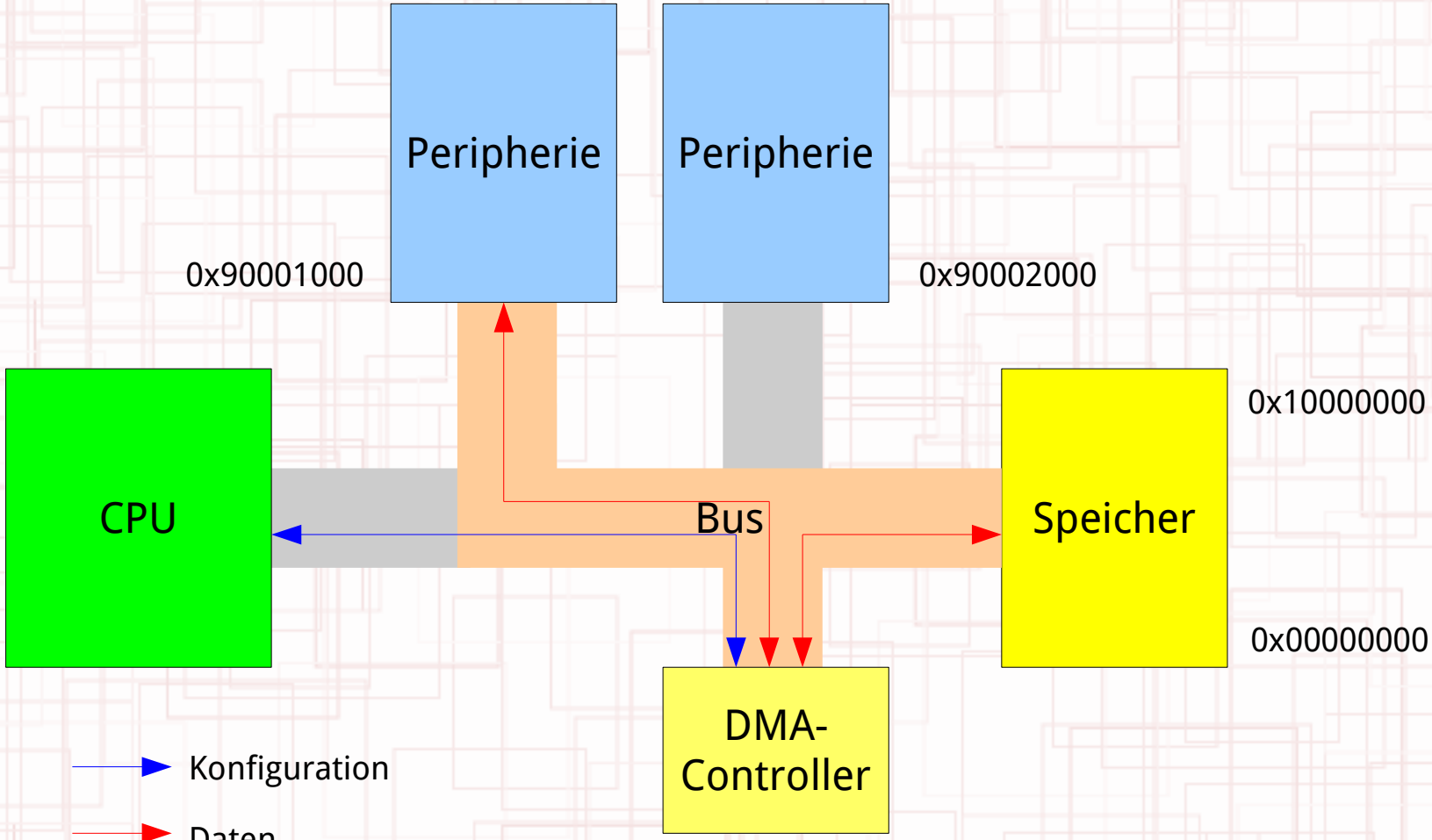
- Gesamtsystem hat weitere Aufgaben
  - Verwaltung Dateisystem
  - Akkuverwaltung
  - GUI/HMI
  - EQ/Lautstärkeregelung
- Latenz bei HMI-Ereignissen ist höher
- Echtzeitanforderung bei Audiodaten!

Kopieren durch CPU




```
memcpy(0x0002000, 0x90001000, 0x0000200)
memcpy(0x0012000, 0x90002000, 0x0000100)
...
memcpy(0x9001000, 0x90001000, 0x0000200)
...
```

Kopieren mittels DMA



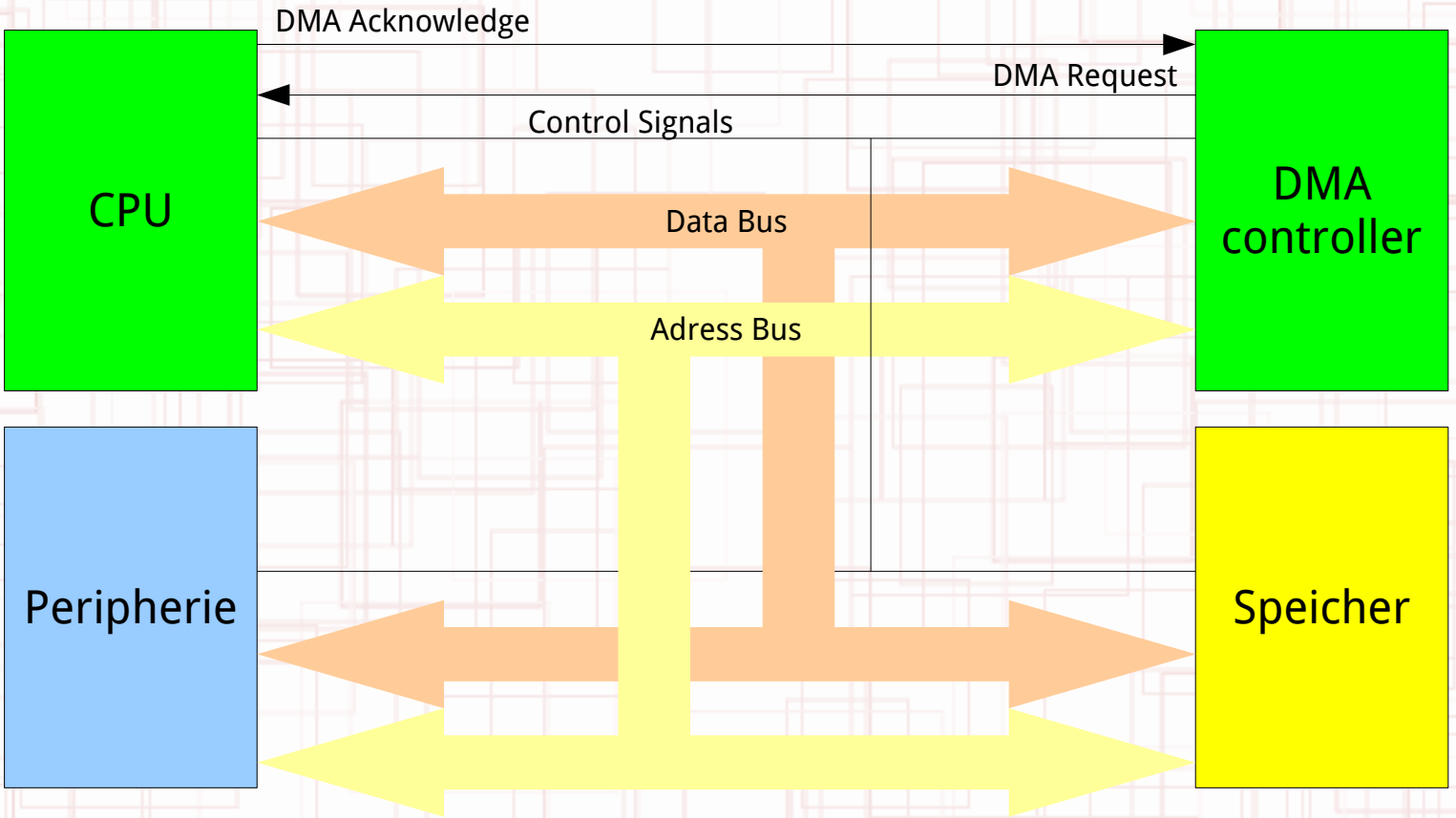
Problemstellungen

- Multi-Master-Fähigkeit des Busses
- Kollisionsmöglichkeit auf dem Bus
- Mehr als ein DMA-Controller auf modernen SoCs
  - DSPs haben oft eigenen DMA-Controller
  - Hardware mit hohem Datenaufkommen: Video-Ports, GigE, Displaycontroller, ...

Lösung

- PC: Leitung für DMA request/DMA grant
  - CPU gibt „freiwillig“ die Buskontrolle ab
- Prioritätsvergabe zwischen CPU und DMA-Kanal auf Busebene
- Network on Chip für internen Bus
  - Dennoch Prioritätenvergabe erforderlich!

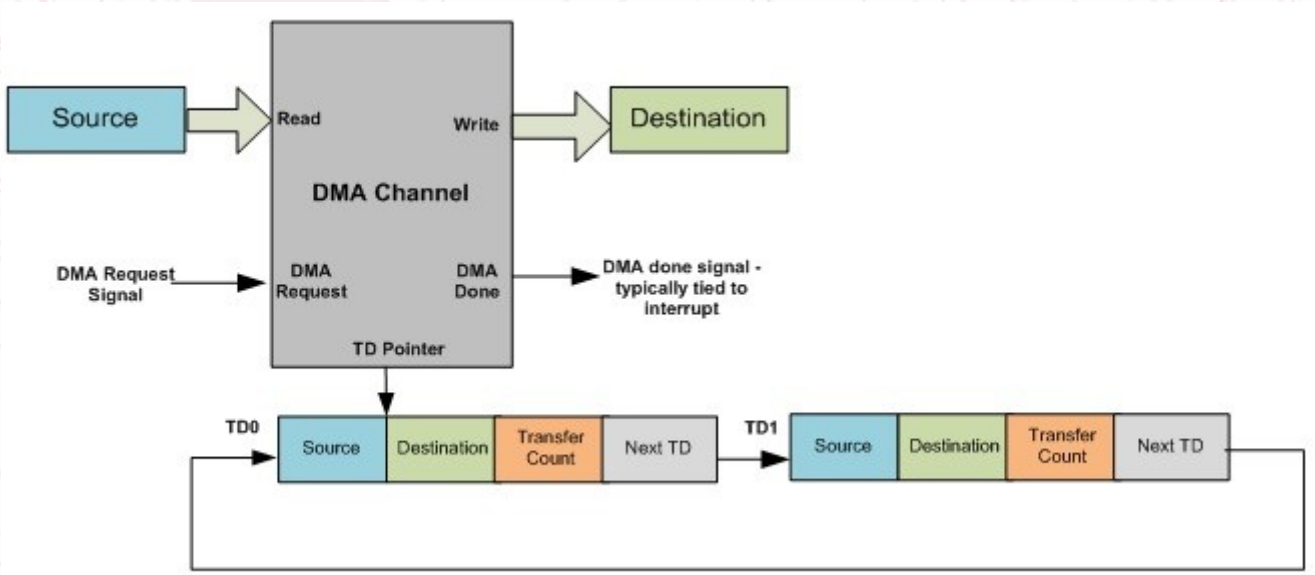
DMA auf PC


DMA Acknowledge
DMA Request
Control Signals
CPU
Data Bus
Adress Bus
DMA controller
Peripherie
Speicher

DMA-Transferarten

- Peripheral to memory
- Memory to peripheral
- Memory to memory
- Peripheral to peripheral

DMA-Konzept

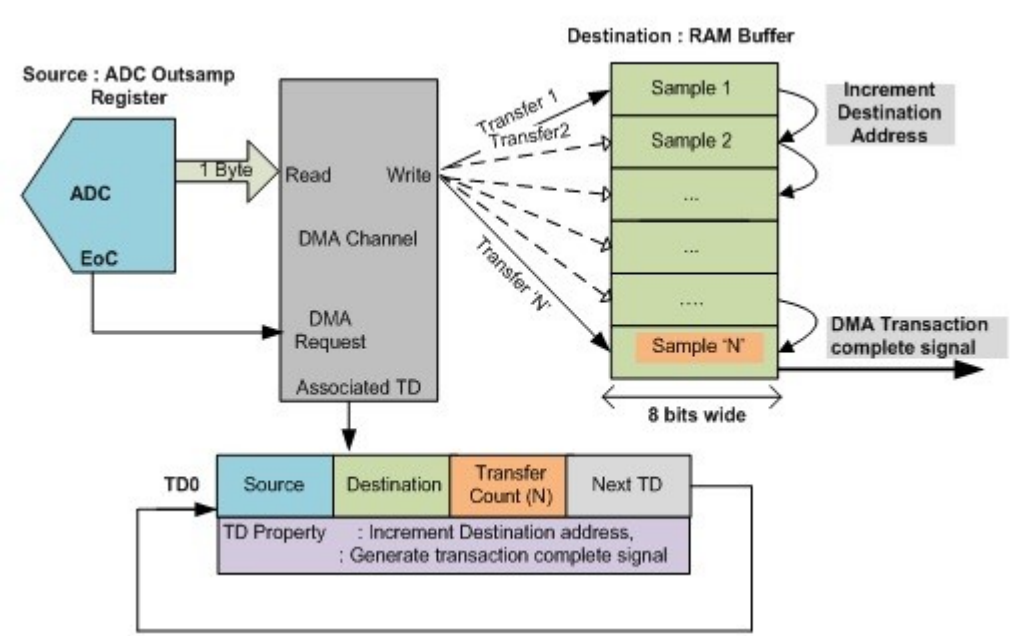


Quelle: cypress.com

Steuerung

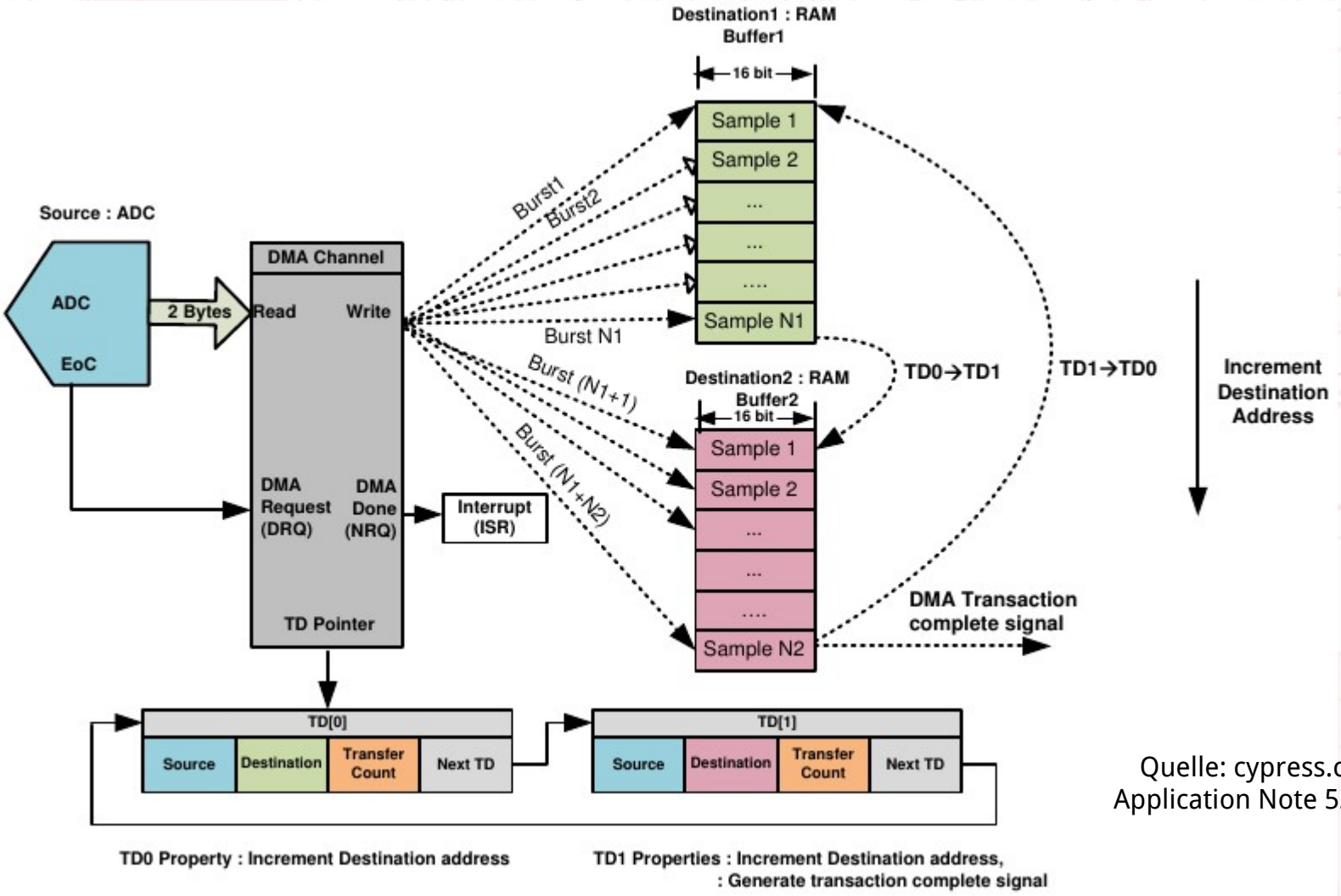
- Start eines Transfers durch Startereignis
  - Manuelles Setzen eines Registerbits
  - Verknüpfung mit Peripherieinterrupt
- Meldung des erledigten Transfers
  - DMA complete event
  - Kann mit weiteren Stufen verknüpft werden

DMA-Mehrfachtransfers



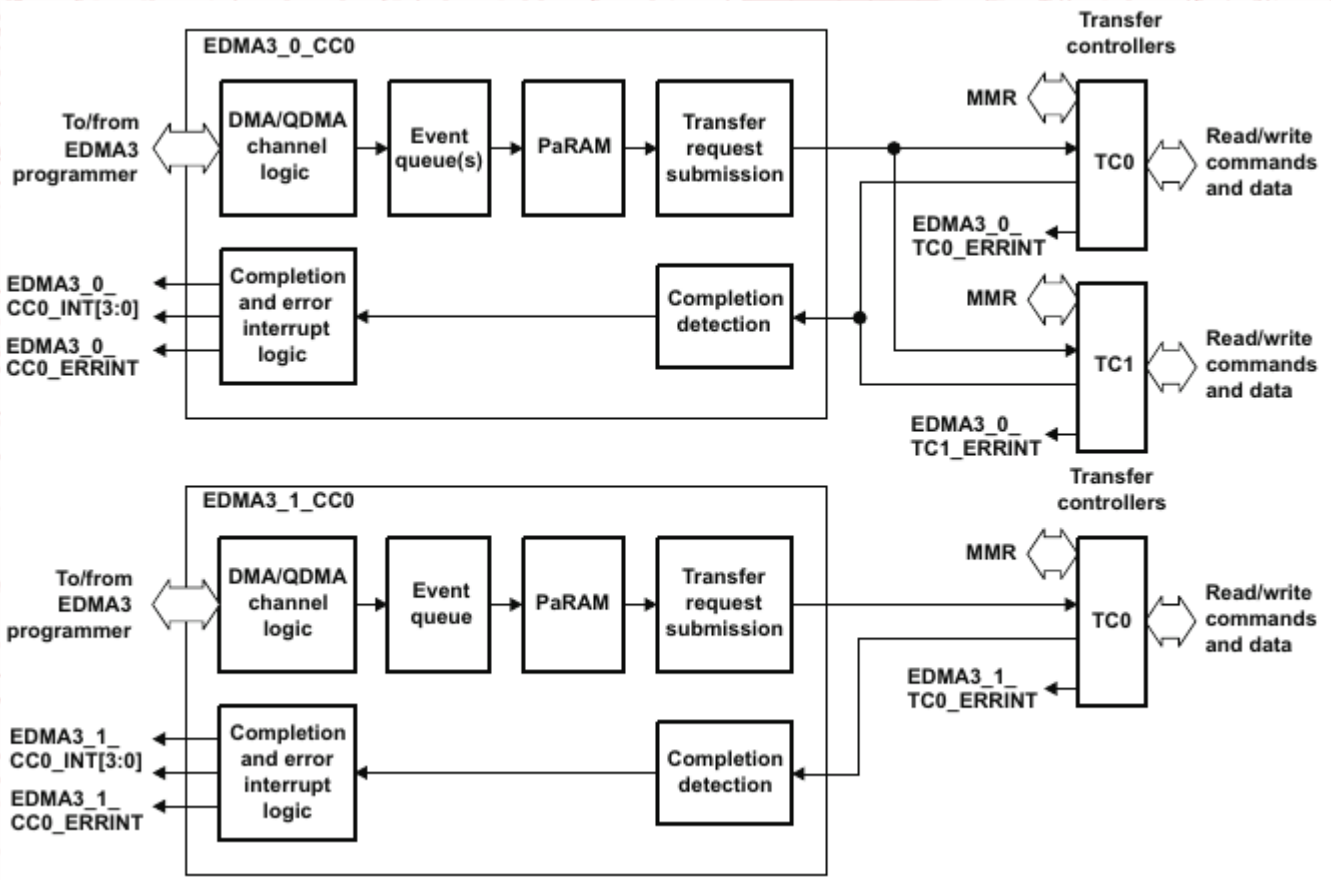
Quelle: cypress.com

TD-Verkettung: „Chaining“



Quelle: cypress.com,
Application Note 52705

DMA OMAP-L138

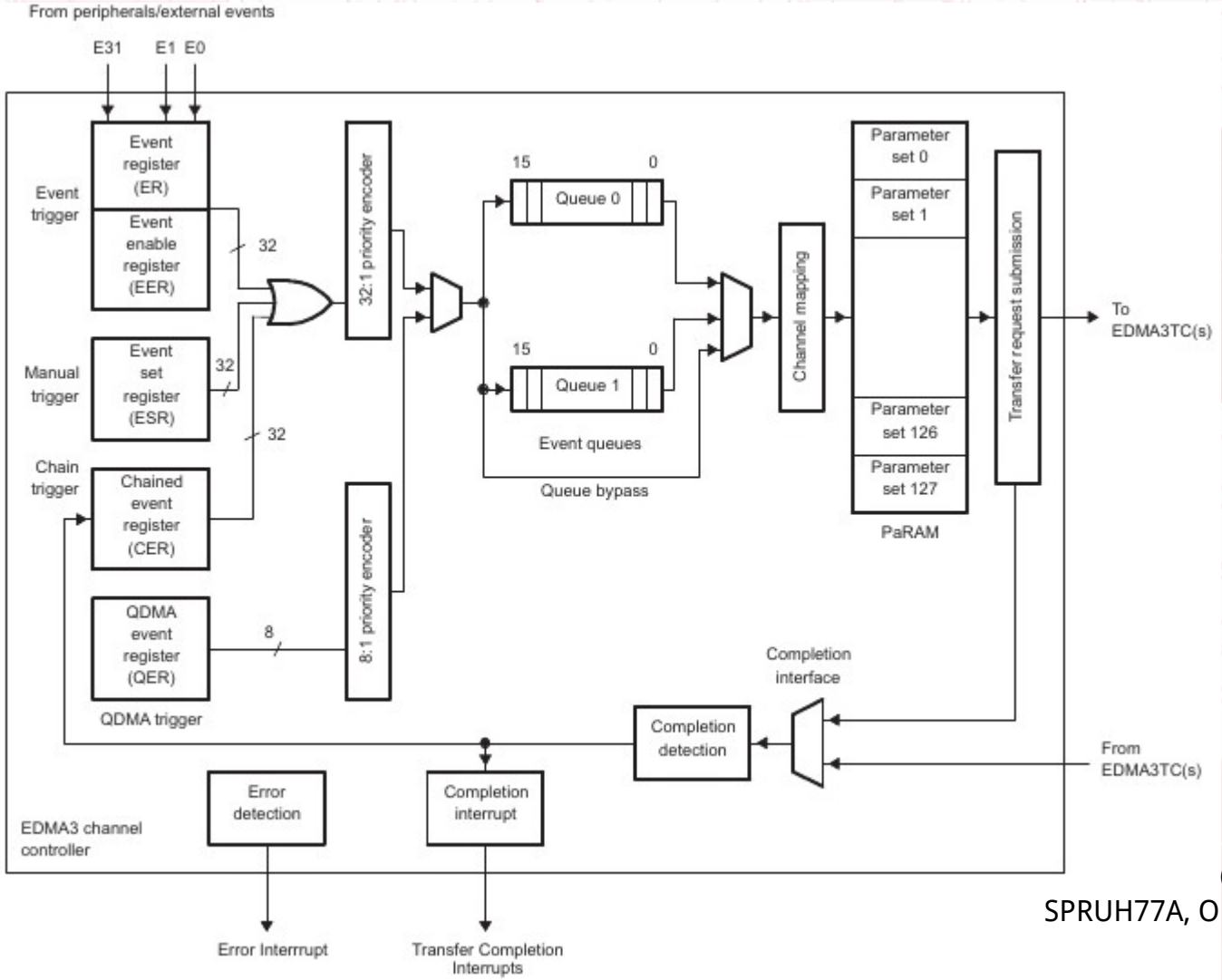


Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

Aufbau

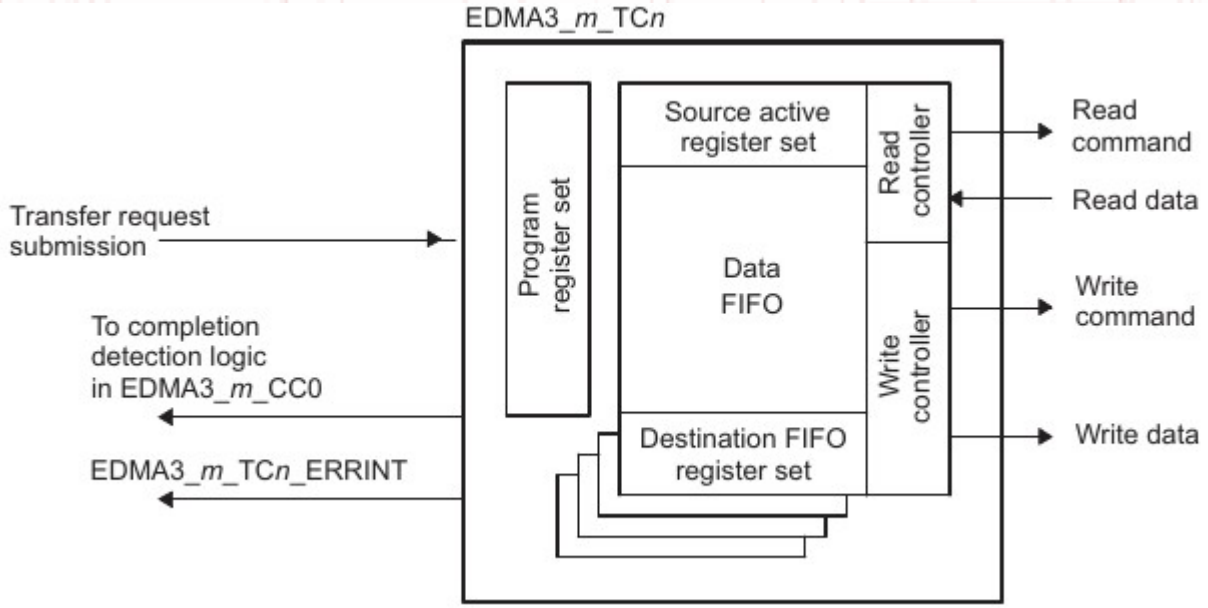
- Channel controller
  - DMA-Userinterface
  - Priorisierung
  - Steuerung
- Transfer Controller
  - Durchführung der Kopieroperation

DMA channel controller



Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

DMA transfer controller



Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

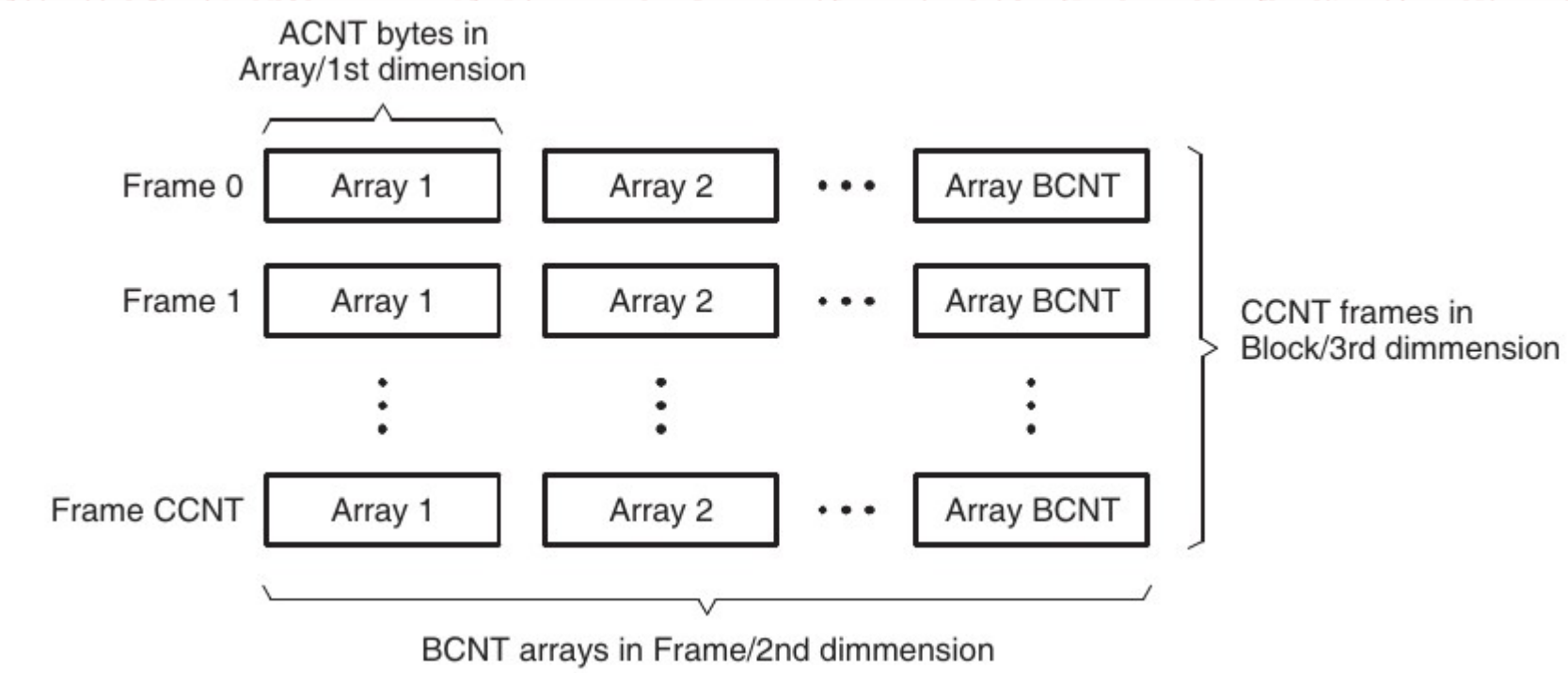
Funktionen

- Eindimensionale Kopien
  - Kopieren zusammenhängender Datenbereiche
- Zweidimensionale Kopien
  - Kopieren mehrerer zusammenhängender Datenbereiche
- Dreidimensionale Kopien
  - Verkettung von 2D-Kopien

DMA-Initiierung

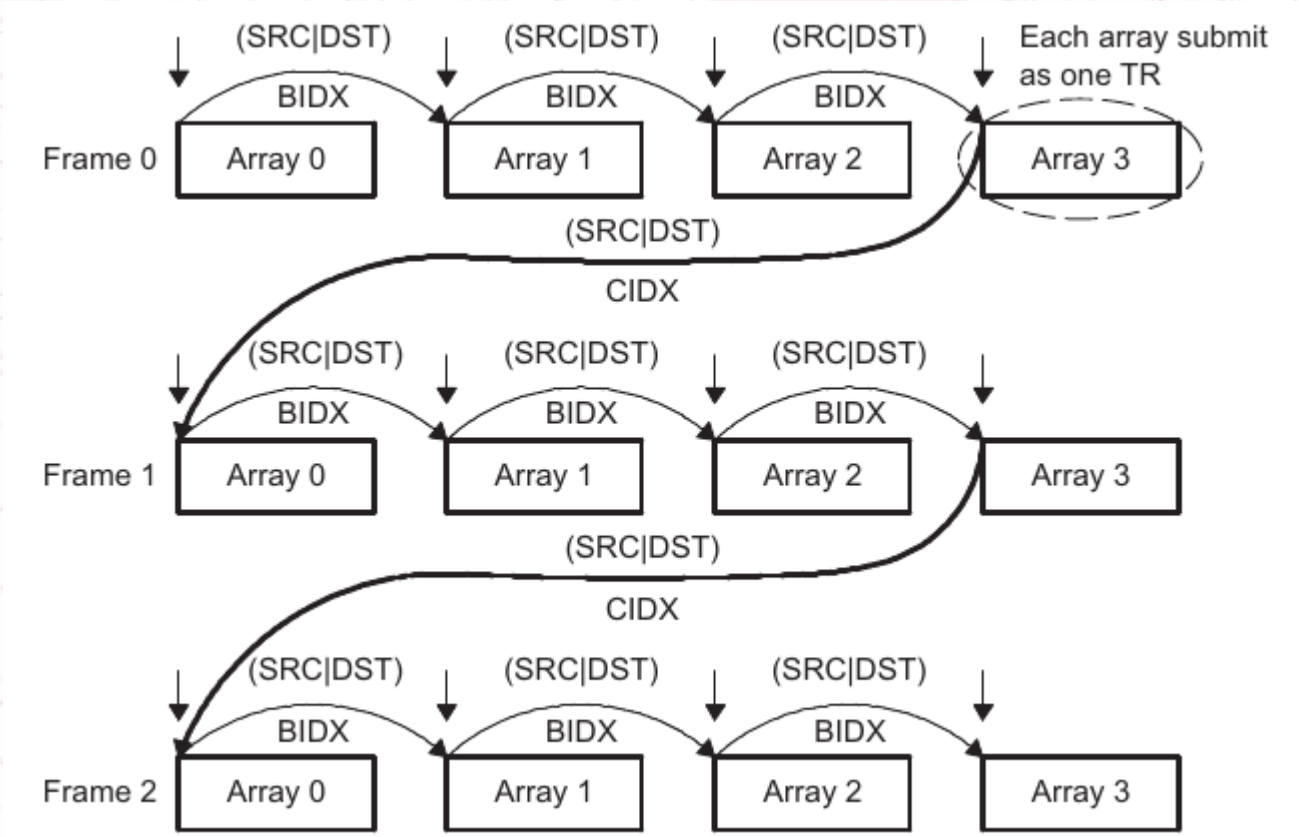
- DMA kann initiiert werden durch
  - Event-triggered: Peripherie, System, externes Ereignis
  - Manually-triggered: Setzen eines Bit im event set register (ESR)
  - Chain-triggered: Bei Signalisierung eines anderen Transfers

DMA-Transfers



Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

Eindimensionale Transfers

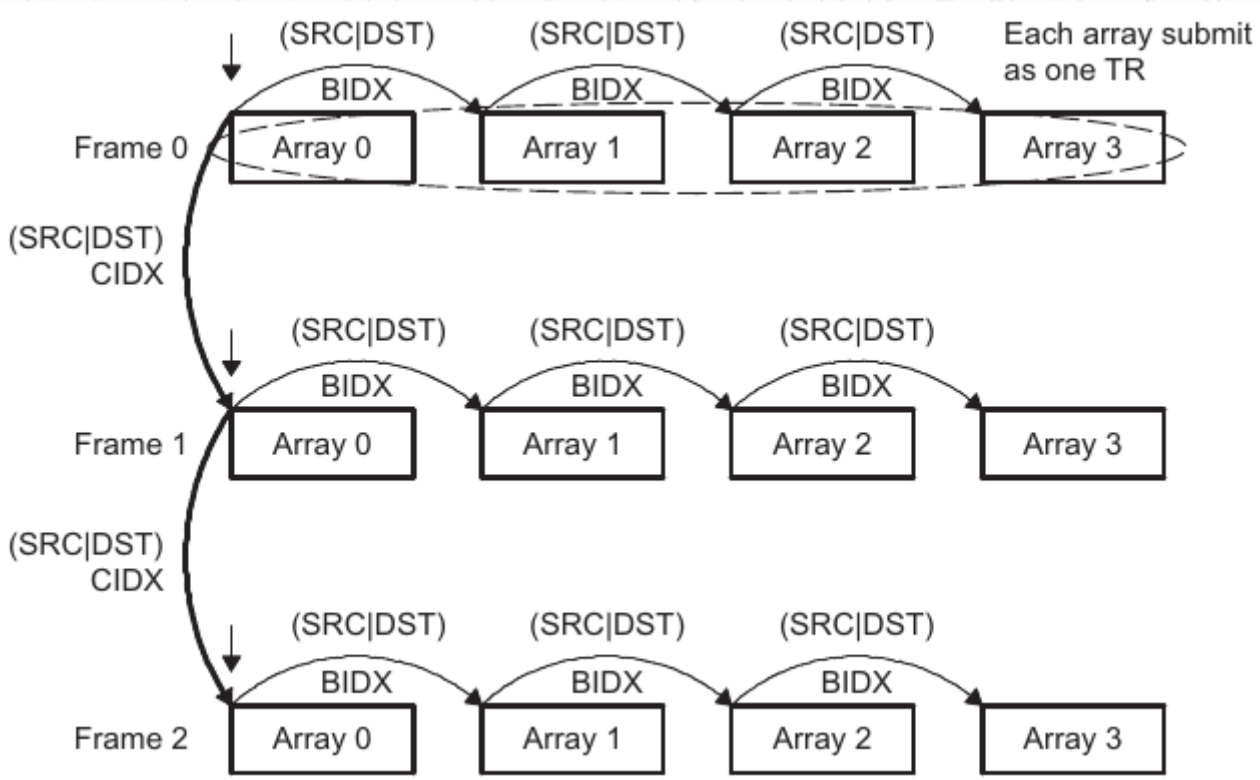


Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

Eindimensionale Transfers

- DMA-Event nach jedem Array
  - Je ein Event pro Array
- Viele Events
  - Zahl der Events = BCNT x CCNT

Zweidimensionale Transfers



Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

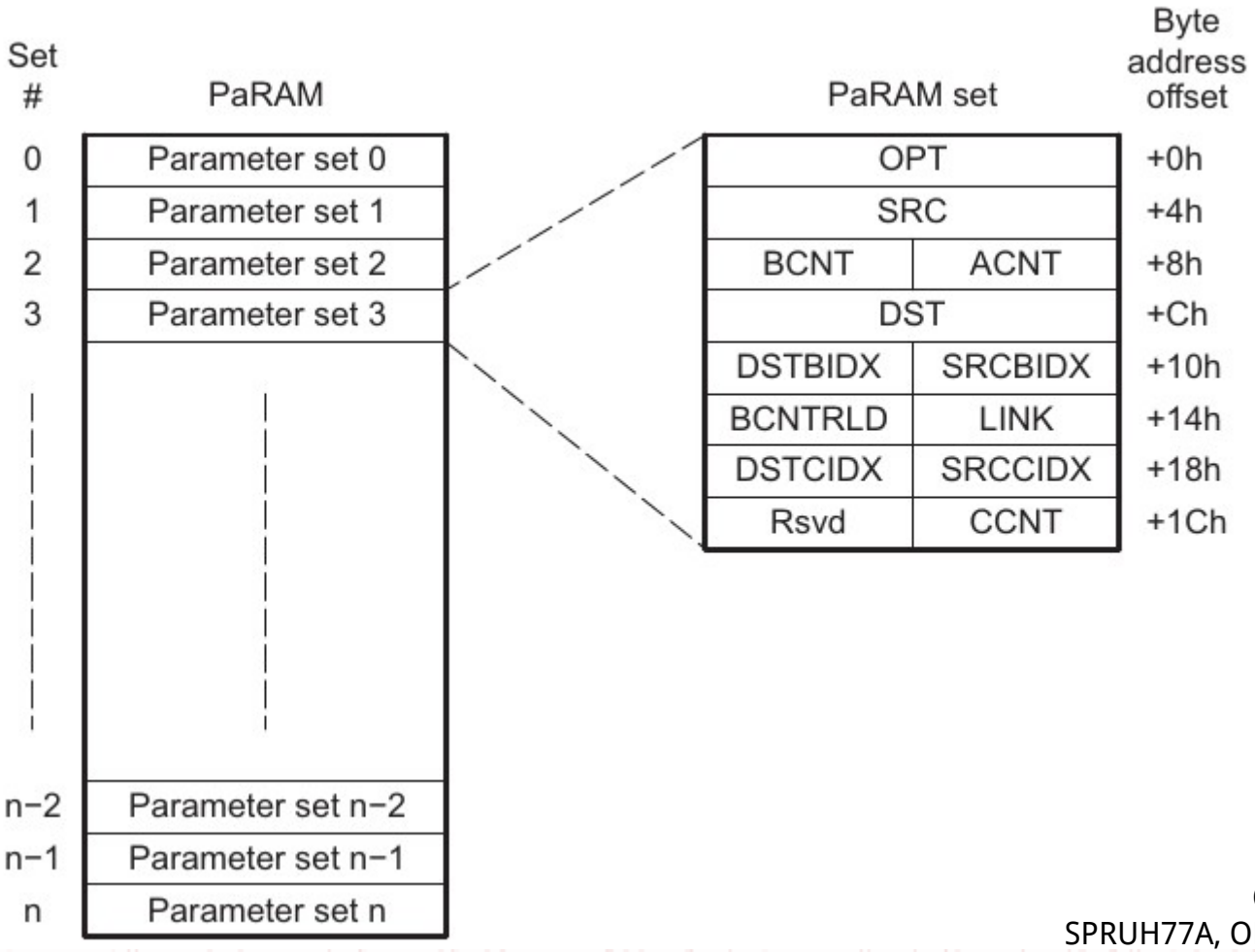
Zweidimensionale Transfers

- DMA-Event nach jeder Arraygruppe
  - Je ein Event pro Arraygruppe
- Weniger Events als bei 1D-Transfer
  - Zahl der Events = CCNT
- 3D-Transfers durch „Chaining“ mehrerer 2D-Transfers

DMA-Programmierung

- Transferparameter im RAM
- Übergabe durch Konfiguration eines Adressregisters
- PaRAM: Parameter RAM
  - 8x4 Byte

PaRAM sets



Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

PaRAM sets

Offset	Acronym	Parameter
0h	OPT	Channel Options
4h	SRC	Channel Source Address
8h	A_B_CNT	A Count/B Count
Ch	DST	Channel Destination Address
10h	SRC_DST_BIDX	Source B Index/Destination B Index
14h	LINK_BCNTRLD	Link Address/B Count Reload
18h	SRC_DST_CIDX	Source C Index/Destination C Index
1Ch	CCNT	C Count

Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

Update der PaRAM sets

- Beim Absenden eines transfer request muss das nächste PaRAM set vorgeladen werden
  - Vor dem nächsten DMA event
  - Inkrementieren von Adressen und Zählern
- Beim abschließenden DMA event muss ein verlinktes PaRAM set geladen werden

Beispiel: Blockkopie

4000 0000h

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	...	...	
	...	...	244	245	246	247	248
249	250	251	252	253	254	255	256

1180 0000h

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	...	...	
	...	...	244	245	246	247	248
249	250	251	252	253	254	255	256

Parameter Contents

0010 0008h	
4000 0000h	
0001h	0100h
1180 0000h	
0000h	0000h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter

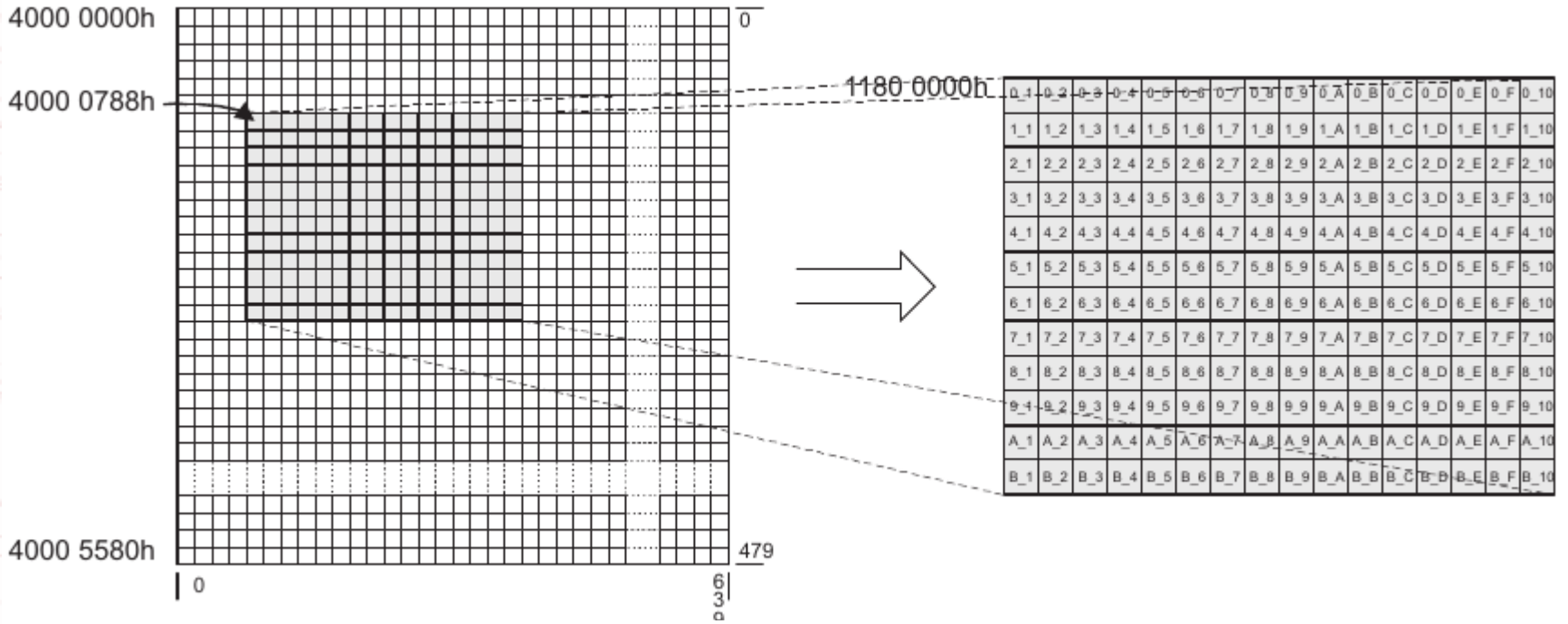
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

Quelle: ti.com,
SPRUH77A, OMAP-L138 TRM

Beispiel: 2D-Kopie

- Ausschneiden eines Bereichs aus Bilddaten im Arbeitsspeicher
  - 640x480, 16bit pro pixel
  - Bereich: 16x12 pixel
  - Ziel: Internes SRAM

Beispiel: 2D-Kopie



Parameter Contents

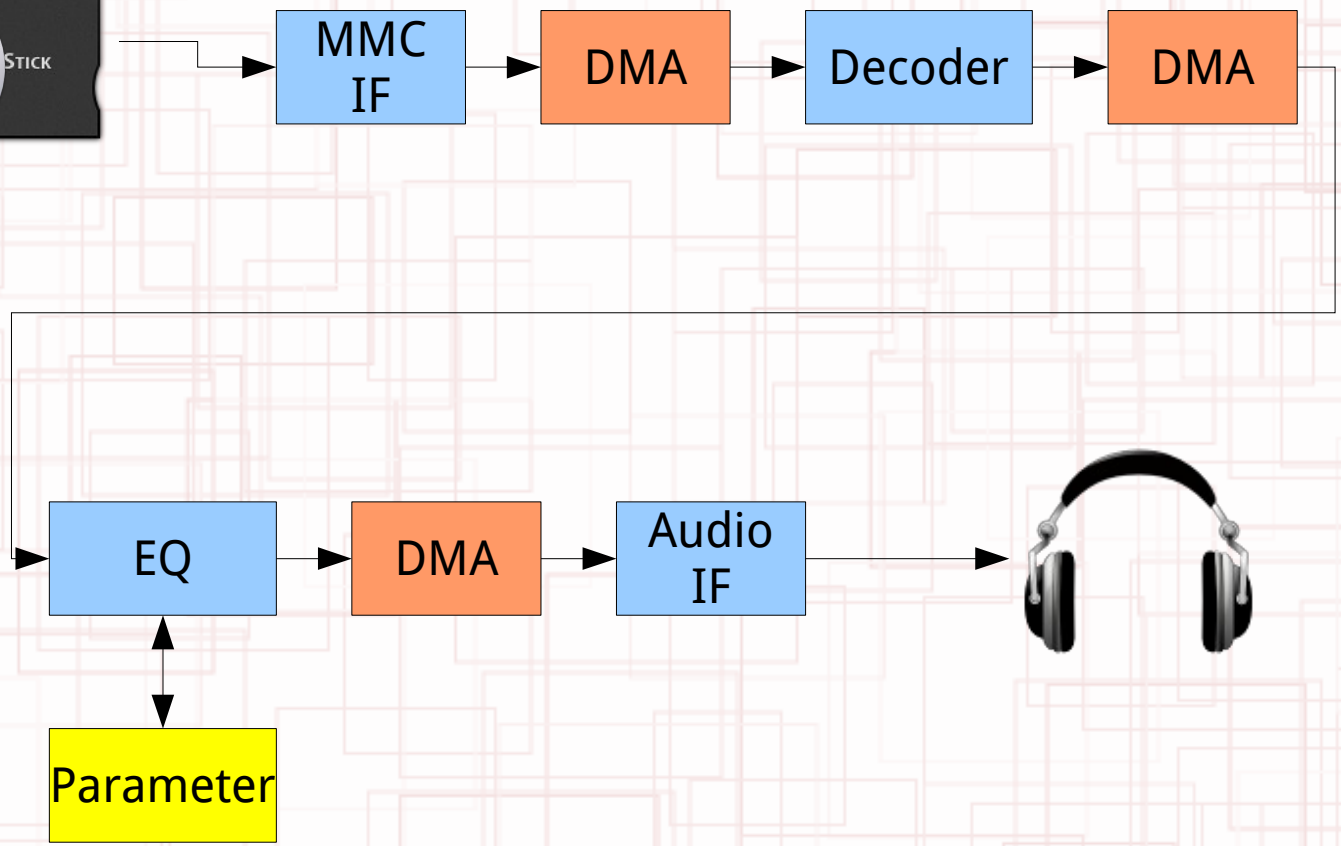
0010 000Ch	
4000 0788h	
000Ch	0020h
1180 0000h	
0020h	0500h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter

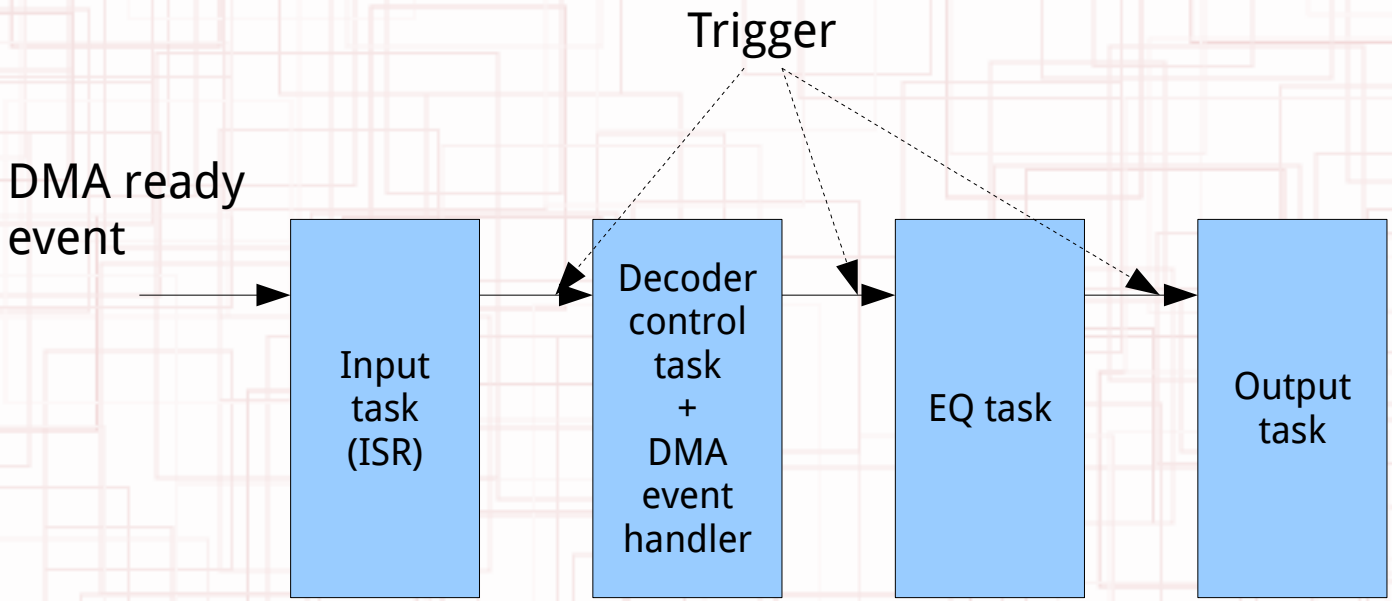
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

Quelle: ti.com, SPRUH77A, OMAP-L138 TRM

MP3-Player mit DMA



Softwaredesign



Direkte DMA-Programmierung

- Initialisierung des DMA-Controllers
- DMA channel programmieren
- Transfer starten
- Warten auf Fertigstellung

Definitionen

```
/* EDMA register address and definitions */

#define EDMA_CC_BASE   (0x02A00000) /* DM648. Check address for other devices. */
#define DCHMAP0     *((volatile unsigned int *)(EDMA_CC_BASE + 0x0100))
#define DMAQNUM0    *((volatile unsigned int *)(EDMA_CC_BASE + 0x0240))
#define QUEPRI      *((volatile unsigned int *)(EDMA_CC_BASE + 0x0284))
#define EMCR        *((volatile unsigned int *)(EDMA_CC_BASE + 0x0308))
#define EMCRH       *((volatile unsigned int *)(EDMA_CC_BASE + 0x030C))
#define QEMCR       *((volatile unsigned int *)(EDMA_CC_BASE + 0x0314))
#define CCERRCLR    *((volatile unsigned int *)(EDMA_CC_BASE + 0x031C))
#define QWMTHRA     *((volatile unsigned int *)(EDMA_CC_BASE + 0x0620))
#define ESR         *((volatile unsigned int *)(EDMA_CC_BASE + 0x1010))
#define IPR         *((volatile unsigned int *)(EDMA_CC_BASE + 0x1068))
#define ICR         *((volatile unsigned int *)(EDMA_CC_BASE + 0x1070))

#define PARAMENTRY0     (0x02A04000) /* DM648. Check address for other devices. */
#define OPT             *((volatile unsigned int *)(PARAMENTRY0 + 0x00))
#define SRC             *((volatile unsigned int *)(PARAMENTRY0 + 0x04))
#define A_B_CNT         *((volatile unsigned int *)(PARAMENTRY0 + 0x08))
#define DST             *((volatile unsigned int *)(PARAMENTRY0 + 0x0C))
#define SRC_DST_BIDX    *((volatile unsigned int *)(PARAMENTRY0 + 0x10))
#define LINK_BCNTRLD    *((volatile unsigned int *)(PARAMENTRY0 + 0x14))
#define SRC_DST_CIDX    *((volatile unsigned int *)(PARAMENTRY0 + 0x18))
#define CCNT            *((volatile unsigned int *)(PARAMENTRY0 + 0x1C))

/* Allocate srcBuff and dstBuff. Do a cache flush and cache invalidate,if required. */
static signed char srcBuff[512];
static signed char dstBuff[512];
```

Konfigurieren und starten

```
/* Step 1: EDMA initialization */
QUEPRI=0x10;
QWMTHRA =(16<<8u)|(16 & 0xFF);
EMCR =  0xFFFFFFFF;
CCERRCLR = 0xFFFFFFFF;

/* Step 2: Programming DMA Channel (and Param set) */
DCHMAP0=0x0;
DMAQNUM0=0x0;
OPT = 0x00100000;  /* only TCINTEN is set */
SRC = (unsigned int)srcBuff;
A_B_CNT = ((1 << 16u) | (512 & 0xFFFFu));    /* ACNT = 512, BCNT = 1 */
DST = (unsigned int)dstBuff;
SRC_DST_BIDX = (512 << 16u) | (512 & 0xFFFFu);    /* SRC_BIDX = 512, DST_BIDX = 512 */
LINK_BCNTRLD = (1 << 16u) | 0xFFFFu;    /* LINK = 0xFFFF, BCNTRLD = 1 */
SRC_DST_CIDX = 0;
CCNT = 1;

/* Step 3: Triggering the Transfer and Waiting for Transfer Completion */
ESR = 0x1;
while(((IPR) & 0x1) == 0);

/* Transfer has completed, clear the status register. */
ICR=0x01;

/* Transfer is complete. Compare the srcBuff and dstBuff */
```

Initialisierung...

- Event Queue Priority
  - QUEPRI
- Queue Watermark Threshold Level
  - QWMTHRA, Debug-Funktionalität
- Fehlerregister löschen
  - Event miss, Channel controller error

Konfiguration

- Mapping von DMA channel zu PaRAM set
- DMA channel und event queue verbinden
- PaRAM set initialisieren
  - Transfer completion interrupt muss aktiviert werden

Ausführen...

- Manueller Start
  - Event Set Register (ESR) = 1
- Abfrage des Interrupt Polling Register
  - IPR
- Rücksetzen des Interrupt Polling Register
  - Interrupt Clear Register, ICR

Präprozessiert...

```
static signed char srcBuff[512];
static signed char dstBuff[512];


*((volatile unsigned int *)((0x02A00000) + 0x0284))=0x10;
*((volatile unsigned int *)((0x02A00000) + 0x0620)) =(16<<8u)|(16 & 0xFF);
*((volatile unsigned int *)((0x02A00000) + 0x0308)) = 0xFFFFFFFF;
*((volatile unsigned int *)((0x02A00000) + 0x031C)) = 0xFFFFFFFF;


*((volatile unsigned int *)((0x02A00000) + 0x0100))=0x0;
*((volatile unsigned int *)((0x02A00000) + 0x0240))=0x0;
*((volatile unsigned int *)((0x02A04000) + 0x00)) = 0x00100000;
*((volatile unsigned int *)((0x02A04000) + 0x04)) = (unsigned int)srcBuff;
*((volatile unsigned int *)((0x02A04000) + 0x08)) = ((1 << 16u) | (512 & 0xFFFFu));
*((volatile unsigned int *)((0x02A04000) + 0x0C)) = (unsigned int)dstBuff;
*((volatile unsigned int *)((0x02A04000) + 0x10)) = (512 << 16u) | (512 & 0xFFFFu);
*((volatile unsigned int *)((0x02A04000) + 0x14)) = (1 << 16u) | 0xFFFFu;
*((volatile unsigned int *)((0x02A04000) + 0x18)) = 0;
*((volatile unsigned int *)((0x02A04000) + 0x1C)) = 1;


*((volatile unsigned int *)((0x02A00000) + 0x1010)) = 0x1;
while(((*((volatile unsigned int *)((0x02A00000) + 0x1068))) & 0x1) == 0);


*((volatile unsigned int *)((0x02A00000) + 0x1070))=0x01;
```

DMA unter Linux

- SoC-DMA-Controller wird durch einen Treiber unterstützt
- Treiber-API bietet Zugriff auf die DMA-Funktionen
  - Channel allocation
  - Channel configuration
  - Interrupt/Event linking

DMA-Beispiele

- Beispiele für TI OMAP-L138
  - Basierend auf lowlevel-Treiber für den EDMA-Controller
- Memory-to-memory EDMA-Transfer
- UART-Datentransfer mit DMA