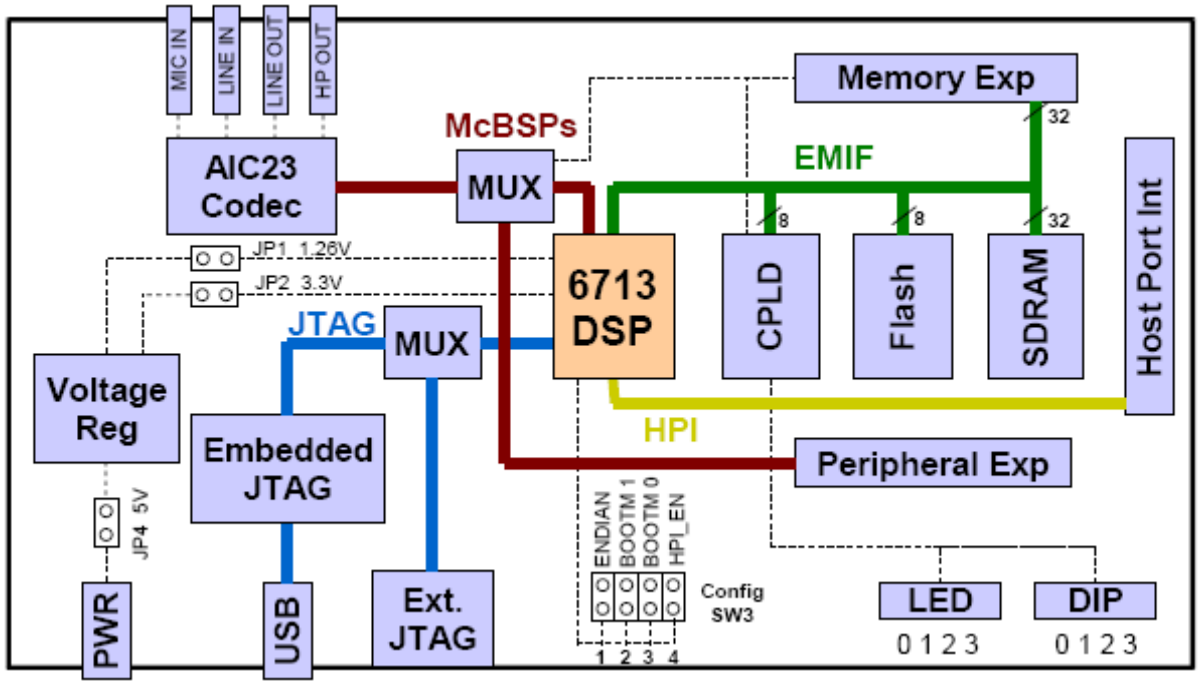
Infoblatt

DSP Starter Kit DSK6713

Das DSK6713 beinhaltet eine PC-gestützte Entwicklungsumgebung und eine auf dem digitalen Signalprozessor TMS320C6713 basierende Demonstrationsplatine von Texas Instruments bzw. Spectrum Digital zur Entwicklung bzw. zum Test von Software für diesen DSP.

Die Entwicklungsumgebung Code Composer Studio, kurz CCS, kommuniziert über einen USB-Port des PC‘s mit der Demonstrationsplatine.

Die Platine enthält neben dem Signalprozessor weitere Peripherie, wie z.B. flüchtigen und nicht-flüchtigen Speicher (SDRAM und Flash-ROM), einen Stereo-Codec (AD-/DA-Wandler-Baustein), 4 LEDs, 4 Schalter usw.



DSP TMS320C6713

Ein DSP (Digitaler Signalprozessor) unterscheidet sich von herkömmlichen Mikroprozessoren durch seinen internen Aufbau.
Die Architektur eines DSP basiert meist auf der sogenannten 'Harvard-Architektur' bzw. einer 'modifizierten Harvard-Architektur' (vgl. Vorlesung). Sie ist gekennzeichnet durch die Trennung von Daten- und Programmspeicher. (1944 – Harvard Mark 1 Computer)

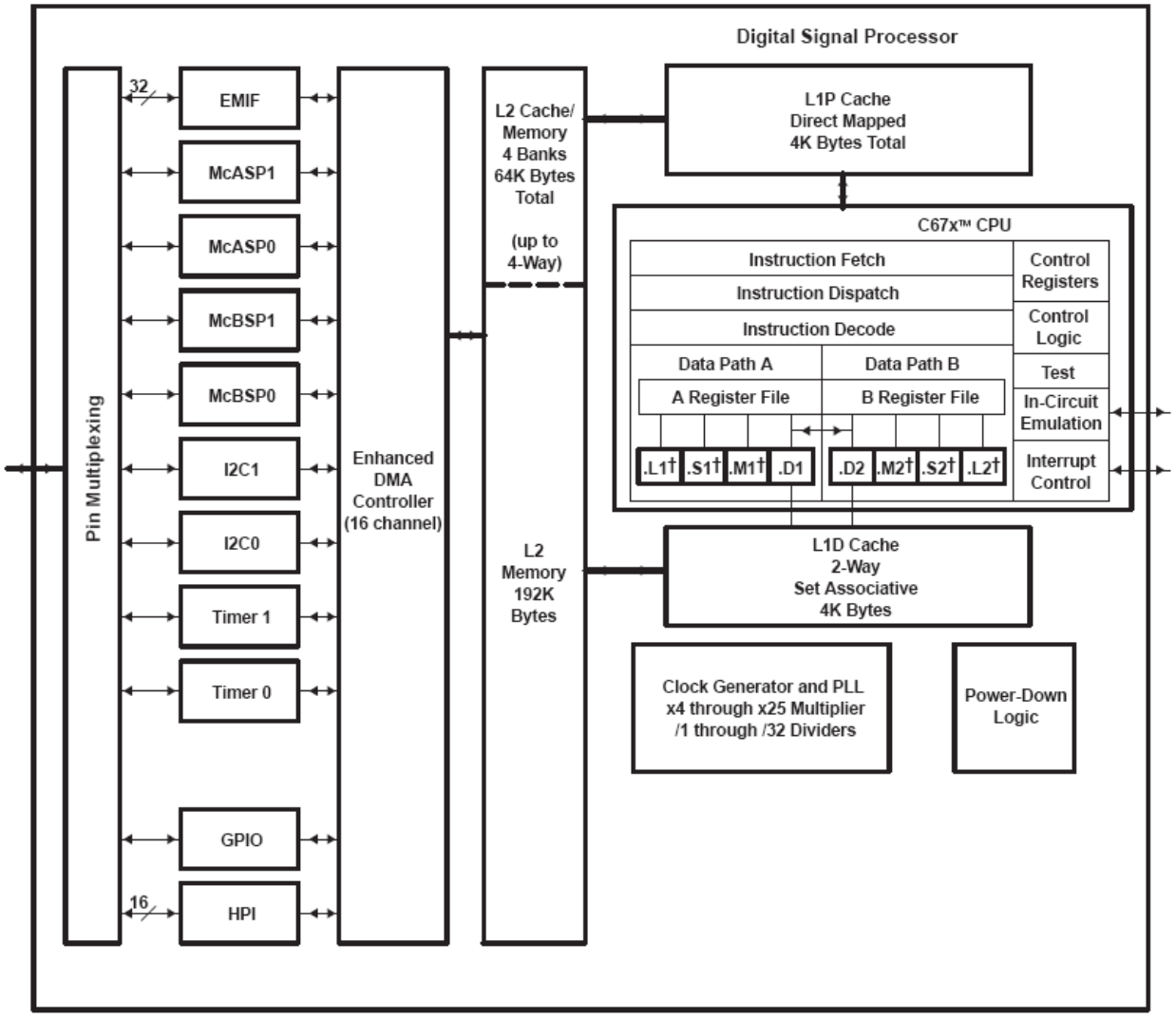
Herkömmliche Mikroprozessoren haben einen gemeinsamen Speicher für Daten und Programmcode, auch 'Von-Neumann-Architektur' genannt. Somit müssen Befehle und Daten nacheinander geladen werden. (1946 veröffentlicht von Burks, Goldstine und von Neumann)
Bei der Signalverarbeitung ist aber ein hoher Datendurchsatz i.d.R. unerlässlich. Erst durch die Trennung von Befehls- und Datenspeicher wird ein paralleles Lesen von Befehlen und Daten ermöglicht und somit ein hoher Datendurchsatz erreicht. Durch die getrennten Speicher benötigt man allerdings auch separate Busse (Daten- und Programmbus). Die entsprechenden Adressregister bzw. Adressrechenwerke müssen getrennt aufgebaut sein.

Der TMS320C6713 ist ein leistungsfähiger Gleitkomma-DSP (Floating-Point Digital Signal Processor) der C6000-Familie von Texas Instruments.

Er basiert auf einem VLIW (Very Long Instruction Word) DSP-Kern (VelociTI). D.h. durch Laden eines VLIW (256 Bit Instructionword = 1 FP Fetch-Packet) werden eigentlich acht 32-Bit-Befehle auf einmal geladen – und durch acht parallel arbeitende Ausführungseinheiten (6 ALUs und 2 Multiplizierer) wird dann eben eine entsprechend hohe Performance erreicht.

Weiterhin bietet der C6713 eine zweistufige Speicherarchitektur mit Programm- und Daten-Cache (32 KBit L1P, 32 KBit L1D) sowie OnChip-RAM (512 KBit L2 RAM/Cache).

C6713 Blockdiagramm



† In addition to fixed-point instructions, these functional units execute floating-point instructions.

EMIF interfaces to:
–SDRAM
–SBSRAM
–SRAM,
–ROM/Flash, and
–I/O devices

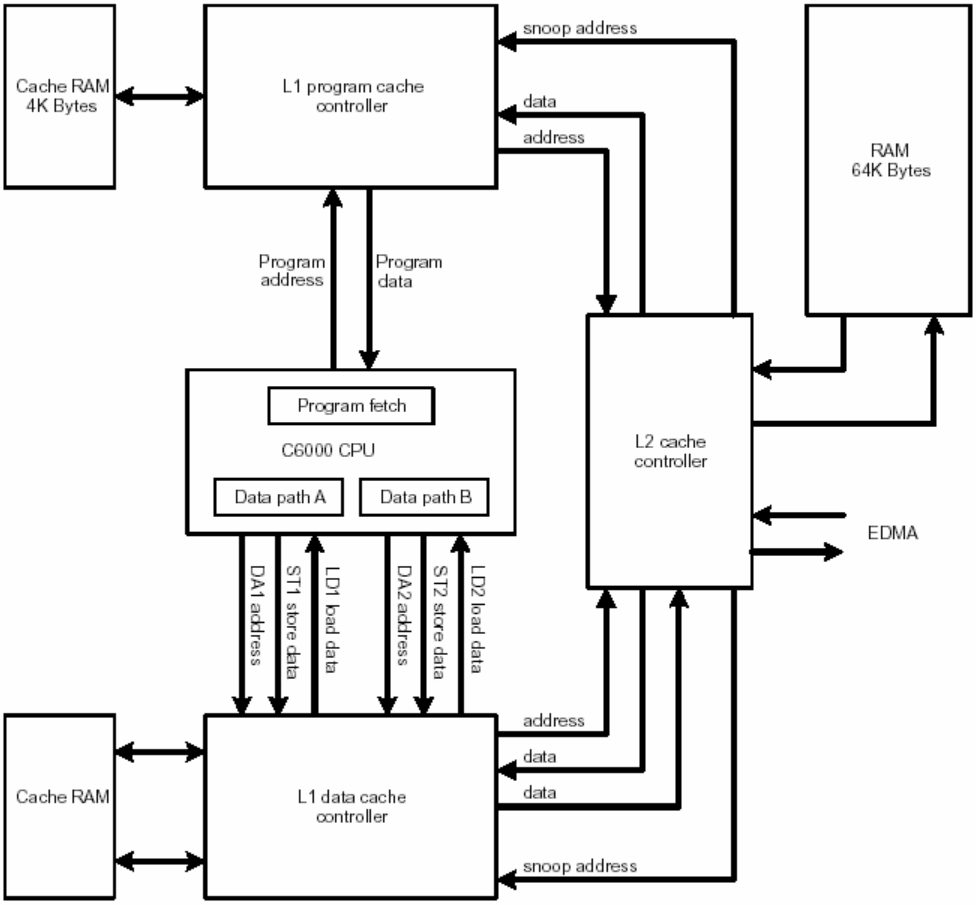
McBSPs interface to:
–SPI Control Port
–High-Speed TDM Codecs
–AC97 Codecs
–Serial EEPROM

McASPs interface to:
–I2S Multichannel ADC, DAC, Codec, DIR
–DIT: Multiple Outputs

Zusätzliche OnChip-Peripherie des C6713:

- EMIF (External Memory Interface)
- EDMA (Enhanced Direct-Memory-Access)
- 2 McASPs (Multichannel Audio Serial Ports)
- 2 McBSPs (Multichannel Buffered Serial Ports)
- 2 I²C (Philips I²C serial ports for control purposes)
- 2 Timer
- HPI (Host-Port Interface)
- GPIO (General Purpose I/O-Pins)

Blockdiagramm der C6000-CPU:



Speicher-Belegungstabelle (Memory Map) des C6713

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	192K	0000 0000 – 0002 FFFF
Internal RAM/Cache	64K	0003 0000 – 0003 FFFF
Reserved	24M – 256K	0004 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	128K	0184 0000 – 0185 FFFF
Reserved	128K	0186 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers	4	019C 0200 – 019C 0203
Reserved	256K – 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers	16K	01B0 0000 – 01B0 3FFF
Reserved	240K	01B0 4000 – 01B3 FFFF
I2C0 Registers	16K	01B4 0000 – 01B4 3FFF
I2C1 Registers	16K	01B4 4000 – 01B4 7FFF
Reserved	16K	01B4 8000 – 01B4 BFFF
McASP0 Registers	16K	01B4 C000 – 01B4 FFFF
McASP1 Registers	16K	01B5 0000 – 01B5 3FFF
Reserved	160K	01B5 4000 – 01B7 BFFF
PLL Registers	8K	01B7 C000 – 01B7 DFFF
Reserved	264K	01B7 E000 – 01BB FFFF
Emulation Registers	256K	01BC 0000 – 01BF FFFF
Reserved	4M	01C0 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	16M – 52	0200 0034 – 02FF FFFF
Reserved	720M	0300 0000 – 2FFF FFFF
McBSP0 Data Port	64M	3000 0000 – 33FF FFFF
McBSP1 Data Port	64M	3400 0000 – 37FF FFFF
Reserved	64M	3800 0000 – 3BFF FFFF
McASP0 Data Port	1M	3C00 0000 – 3C0F FFFF
McASP1 Data Port	1M	3C10 0000 – 3C1F FFFF
Reserved	1G + 62M	3C20 0000 – 7FFF FFFF
EMIF CE0†	256M	8000 0000 – 8FFF FFFF
EMIF CE1†	256M	9000 0000 – 9FFF FFFF
EMIF CE2†	256M	A000 0000 – AFFF FFFF
EMIF CE3†	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space.

Interner Speicher des TMS320C6713

Der DSP hat 2048 kBit (entspricht 256 kByte = 64 kWorte (32 Bit)) internen Speicher (L2 / Cache).

Startadresse:	`0x0000 0000 hex`	
Länge in Byte:	`0x0004 0000 hex`	(= 256 kByte = 262 144 Byte)
↳ Länge in 32-Bit-Worten:	`0x0001 0000 hex`	(= 64 kWorte = 65 536 32-Bit-Worte)
Endeadresse:	`0x0003 ffff hex`	

(Siehe Linker-Datei: `dsk6713.cmd`)

Externer Speicher auf dem DSK6713

Auf dem Starter-Kit ist ein SDRAM (Synchronous DRAM) vom Typ MT48LC4M32B2 mit 128 MBit (entspricht insgesamt 16 MByte = 4 MWorte (32 Bit)) über das EMIF an den DSP angebunden.

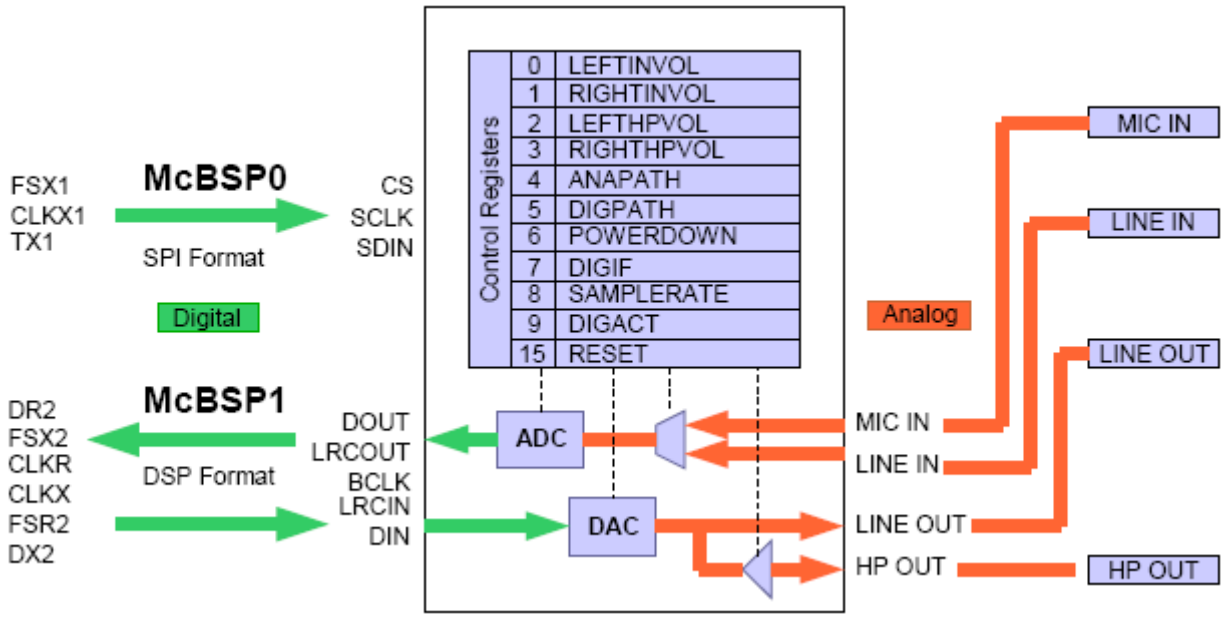
Startadresse:	`0x8000 0000 hex`	
Länge in Byte:	`0x0100 0000 hex`	(= 16 MByte = 16 777 216 Byte)
↳ Länge in 32-Bit-Worten:	`0x0040 0000 hex`	(= 4 MWorte = 4 194 304 32-Bit-Worte)
Endeadresse:	`0x80ff ffff hex`	

(Siehe Linker-Datei: `dsk6713.cmd`)

Schnittstelle des DSK zur analogen Welt:

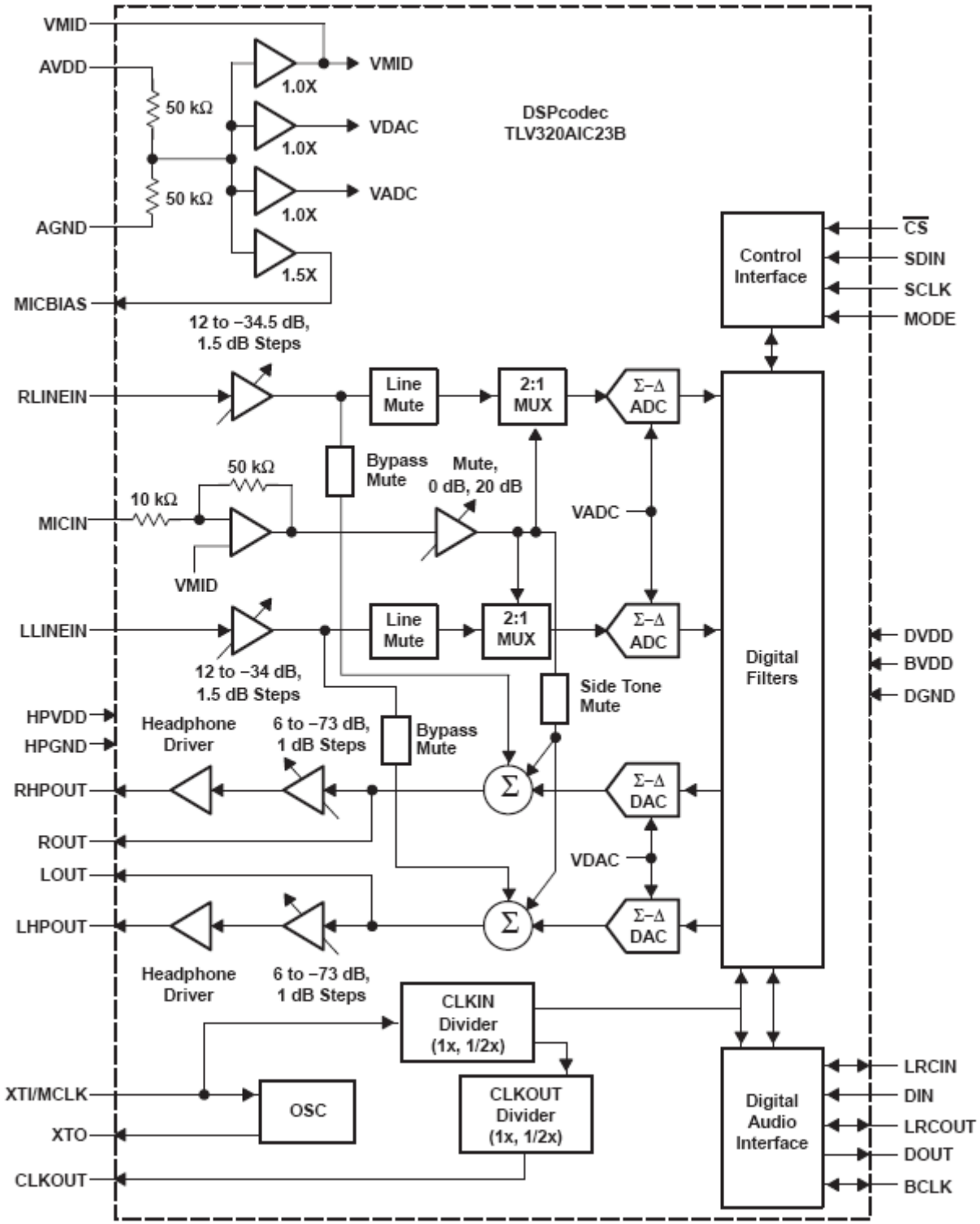
Stereo-Codec TLV320AIC23B / Multichannel Buffered Serial Port (McBSPx)

Als Schnittstelle zur analogen Welt befindet sich auf dem DSK6713 ein Stereo-Codec (Multifunktions – A/D - D/A – Wandlerbaustein: TLV320AIC23B).
Dieser Codec (das Wort setzt sich zusammen aus: **co**der/**dec**oder) bietet ausser den im Labor genutzten Line-Ein- und Ausgängen (Stereo – A/D - D/A – Wandlung für Sprach-/Musiksignale mit üblichem Line-Pegel, hier max. 1 V_{RMS}) auch einen empfindlichen Eingang mit einstellbarer Verstärkung für ein Elektretmikrofon und einen ebenfalls in der Lautstärke einstellbaren Stereo-Kopfhörerausgang.



Der Analog-Teil lässt sich für verschiedene Anforderungen anpassen und programmieren (siehe Datenblatt tlv320aic23b.pdf bzw. Blockschaltbild und Codec-Initialisierung weiter unten).

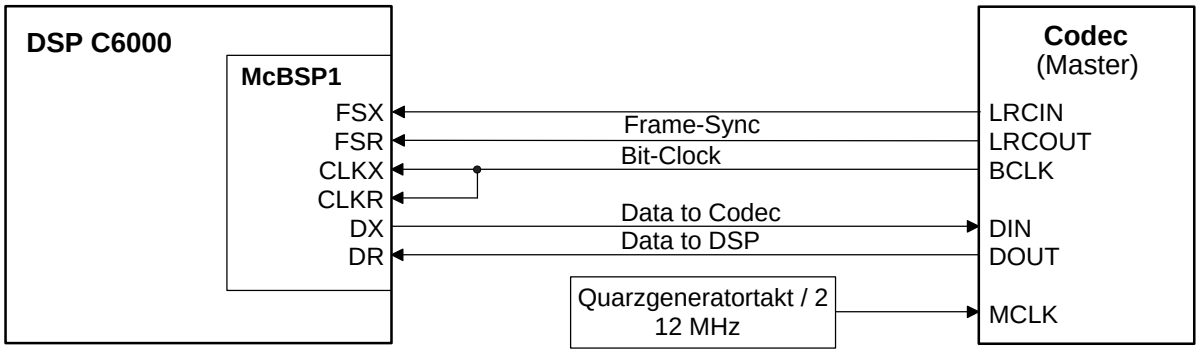
Auf dem DSK6713 ist die 3,5 mm Stereo-Klinkenbuchse LINE IN mit den Eingängen LLINEIN und RLINEIN des Codecs verbunden (vgl. Blockschaltbild unten) und die 3,5 mm Stereo-Klinkenbuchse LINE OUT ist mit den Ausgängen LOUT und ROUT des Codecs verbunden.



Codec Blockschaltbild

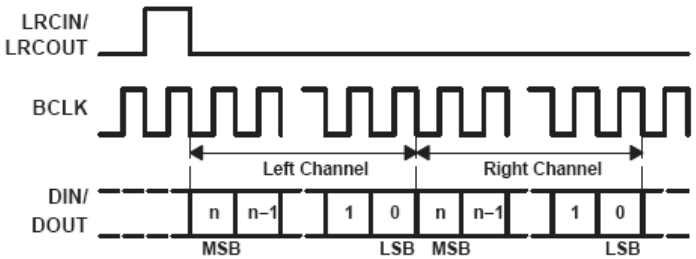
Der Codec bietet eine variable Abtastrate zwischen 8 kHz und 96 kHz. Bei den Wandlern (ADC und DAC) handelt es sich um Sigma-Delta-Typen mit Oversampling und Noiseshaping, die eine mit klassischen 16 Bit Wandlern vergleichbare Audioqualität liefern. Dem ADC ist ein Dezimationsfilter nachgeschaltet und dem DAC ein Interpolationsfilter vorgeschaltet.

Der Codec ist über zwei serielle Interfaces an den DSP, genauer gesagt an die Multichannel Buffered Serial Ports des DSP's angebunden (McBSP0 zur Konfiguration und McBSP1 für den Audiodatentransfer).
Für den DSP stellt ein McBSP ein (on-chip) Peripherie-Device dar, das vor dem Datentransfer entsprechend der Anforderungen der angeschlossenen externen Peripherie (hier also dem Codec) initialisiert / konfiguriert werden muss (siehe McBSP-Initialisierung weiter unten).



Bei der digitalen **Audiodatenübertragung** auf dem DSK6713 fungiert der **Codec als Takt-Master**. D.h. Bit-Clock (BCLK) und Frame-Sync (LRCIN/LRCOUT) des seriellen Interfaces zwischen Codec und McBSP1 werden vom Codec zur Verfügung gestellt (abgeleitet von einem 24 MHz Quarzgenerator). Der McBSP1 läuft also taktmäßig als Slave des Codecs.

Der Bit-Clock entspricht hier dem zugeführten Master-Clock (MCLK) und damit dem halben Quarzgeneratortakt (BCLK = MCLK = 12 MHz).
Der Codec wird im so genannten DSP-Mode betrieben (vgl. Datenblatt tlv320aic23b.pdf). Standardmäßig erzeugt er alle 250 Bit-Clocks einen Frame-Sync-Impuls für die Dauer einer Bit-Clock-Periode und leitet so alle 20,83 µs (entspricht einer Abtastfrequenz von 48 kHz) die **Übertragung eines Datenwortes, bestehend aus zwei Samples**, ein.
Das MSB des ersten (linken) Samples wird in diesem Mode mit der zweiten steigenden Bit-Clock-Flanke nach der steigenden Frame-Sync-Flanke vom Empfänger übernommen. Je nach betrachteter Richtung der Datenübertragung also entweder beim McBSP1 im DSP (DR) oder beim Codec (DIN).



Wegen der 16 Bit äquivalenten Audioqualität des Codec's und der Standardwortbreite von 32 Bit des DSP's bietet es sich an, für die **Wortbreite der Samples 16 Bit** zu wählen und Codec und McBSP1 entsprechend zu konfigurieren.
Der Empfangsteil im Codec erwartet dann also, dass ihm **32 Bit breite Datenwörter** übergeben werden (zwei 16 Bit Samples direkt hintereinander).

Da für die Versuche im Labor eine geringere **Abtastfrequenz** ausreichend ist, wird der Codec für **32 kHz** konfiguriert. Der Frame-Sync-Impuls wird demnach nur alle 375 Bit-Clocks generiert (12 MHz / 32 kHz).

Die 32 Datenbits werden mit den ersten 32 Bit-Clocks nach einem Frame-Sync-Impuls eingetaktet. Während der restlichen 343 Bit-Clocks werden keine weiteren Datenbits eingetaktet.

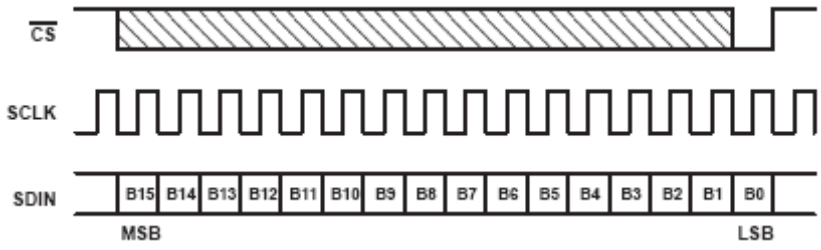
Die 16 Bits jedes so übergebenen Samples interpretiert der Codec dann als 16 Bit K2-Festkommazahl, die vom jeweiligen D/A-Wandler analog gewandelt und ausgegeben wird.
Als Folge davon muss man bei der Datenübertragung vom DSP zum Codec vor der Ausgabe der Datenworte über den McBSP1 dafür sorgen, dass die analog zu wandelnden **Samples im 16 Bit K2 Format** vorliegen und die oberen 16 Bit des 32 Bit Datenwortes das linke Sample und die unteren 16 Bit das rechte Sample enthalten.

Codec-Initialisierung

Der Codec enthält insgesamt 11 Control-Register:

ADDRESS	REGISTER
0000000	Left line input channel volume control
0000001	Right line input channel volume control
0000010	Left channel headphone volume control
0000011	Right channel headphone volume control
0000100	Analog audio path control
0000101	Digital audio path control
0000110	Power down control
0000111	Digital audio interface format
0001000	Sample rate control
0001001	Digital interface activation
0001111	Reset register

Die Steuerdaten zur Konfiguration des Codecs werden über den McBSP0 im SPI-Mode vom DSP zum Codec übertragen. Die Steuerwörter sind 16 Bit breit, wobei die oberen 7 Bit die Adresse des zu beschreibenden Control-Registers darstellen und die unteren 9 Bit die Steuer- bzw. Registerdaten (vgl. Datenblatt tlv320aic23b.pdf).



Eine **Grundinitialisierung** beginnt mit dem Auslösen eines Resets:

Register 15: `0001111 000000000 bin = 1E00 hex` – software reset

Anschließend wird für alle Funktionseinheiten bis auf den Mikrofoneingang (dieser wird im Labor nicht benötigt) der Power-Down-Mode aufgehoben:

Register 6: `0000110 000000010 bin = 0C02 hex` – no power down

Danach werden die Register für die Konfiguration des digitalen Audio-Interfaces initialisiert und mit dem Register Digital-Interface-Activation der Betrieb freigegeben:

Register 8: `0001000 000011001 bin = 1019 hex` – sample rate control:
32 kHz sampling rate
USB clock mode

Register 7: `0000111 001010011 bin = 0E53 hex` – digital audio interface format:
master mode
DSP format
16 Bit

Register 9: `0001001 000000001 bin = 1201 hex` – digital audio interface activation

Zum Schluss werden die Register für die Kontrolle des Audiopfads im Codec nach Wunsch beschreiben. Hier sind spätere Änderungen während des laufenden Betriebs denkbar.

Register 4: `0000100 000010010 bin = 0812 hex` – analog audio path control:
no side tone
DAC selected
no bypass
ADC input: Line
mic mute, no mic boost

Register 5: `0000101 000000010 bin = 0A00 hex` – digital audio path control:
no DAC soft mute
no de-emphasis
ADC input: Line
no ADC high-pass filter

Register 0: `0000000 000010111 bin = 0017 hex` – left line in volume control:
no l/r simultaneous update
no left line in mute
left line in volume: 0 dB

Register 1: `0000001 000010111 bin = 0217 hex` – right line in volume control:
no r/l simultaneous update
no right line in mute
right line in volume: 0 dB

Register 2: `0000010 011110011 bin = 04F3 hex` – left headphone volume control:
no l/r simultaneous update
left channel zero cross detect
left headphone volume: -6 dB

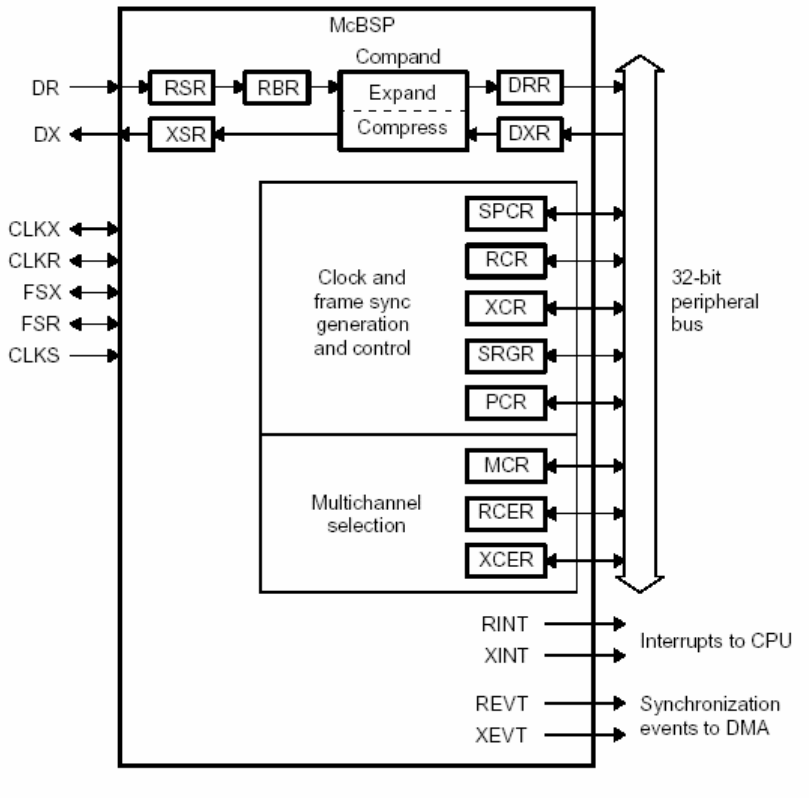
Register 3: `0000011 011110011 bin = 06F3 hex` – right headphone volume control:
no r/l simultaneous update
right channel zero cross detect
right headphone volume: -6 dB

Ganz am Ende wird zur Übernahme des vorangegangenen (letzten) Steuerwortes noch ein weiteres Steuerwort gesendet (erzeugen eines abschließenden Chip-Select-Impulses).
Das kann eine Wiederholung des letzten Steuerwortes sein oder ein Zugriff auf das Reset-Register ohne aber einen Reset auszulösen.

Register 15: `0001111 111111111 bin = 1FFF hex` – doesn't trigger software reset

Damit ist die Initialisierung des Codec's abgeschlossen.

McBSP Pins, Register und Initialisierung



McBSP Blockdiagramm

Pins :

DR	Received serial data
DX	Transmitted serial data
CLKX	Transmit clock
CLKR	Receive clock
FSX	Transmit frame synchronization
FSR	Receive frame synchronization
CLKS	External clock

Register :

DRR	Data receive register
DXR	Data transmit register
RBR	Receive buffer register
RSR	Receive shift register
XSR	Transmit shift register
SPCR	Serial port control register
RCR	Receive control register
XCR	Transmit control register
SRGR	Sample rate generator register
PCR	Pin control register
MCR	Multichannel control register
RCER	Receive channel enable register
XCER	Transmit channel enable register

Initialisierung der für die Konfiguration und den anschließenden Betrieb des Codecs TMS320AIC23B wichtigen McBSP-Register

Schrittweise Initialisierung der McBSP0-Register für die Konfiguration des Codecs im SPI Format:

Eine **Grundinitialisierung** beginnt mit dem Reset des McBSP0 via Serial Port Control Register:

```
McBSP0_SPCR      = 0x0000 0000;
```

Bringt die folgenden McBSP0-Funktionseinheiten in den Reset:

- Frame Sync Generator
- Sample Rate Generator
- Transmitter
- Receiver

Sample Rate Generator Register:

```
McBSP0_SRGR      = 0x2000 1999;
```

Bewirkt die folgenden Einstellungen:

- GSYNC: FREE
- CLKSP: RISING
- CLKSM: INTERNAL, SRG-Clock derived from CPU-Clock
- FSGM: DXR2XSR, Transmit-Frame-Sync-Signal on every DXR-to-XSR copy
- FPER: `0x000` hex
- FWID: `0x19` hex
- CLKGDV: `0x99` hex

Serial Port Control Register:

```
McBSP0_SPCR      = 0x0000 1000;
```

Bewirkt u.a. die folgenden Einstellungen:

- FRST: YES, Frame Sync Generator in Reset
- GRST: YES, Sample Rate Generator in Reset
- XINTM: XRDY, Transmit Interrupt XINT0 is driven by XRDY-Bit (Bit 17 in `McBSP0_SPCR`)
- XRST: YES, Transmitter is disabled (in Reset)
- **CLKSTP**: NODELAY, Clock starts with falling edge without delay
- RRST: YES, Receiver is disabled (in Reset)

Pin Control Register:

```
McBSP0_PCR       = 0x0000 0A02;
```

Bewirkt u.a. die folgenden Einstellungen:

- XIOEN: SP, Standard Serial Port Operation (not GPIO)
- **FSXM**: INTERNAL, Internal Frame Synchronisation for TX
- **CLKXM**: OUTPUT, McBSP is Master (generates CLKX)
- **FSXP**: ACTIVEHIGH, Frame Synchronisation Pulse is active-high for TX
- **CLKXP**: FALLING, TX Data driven on falling TX-clock-edge

Transmit Control Register:

```
McBSP0_XCR      = 0x0001 0040;
```

Bewirkt u.a. die folgenden Einstellungen:

- **XPHASE**: SINGLE, Single Phase Frame
- XCOMPAND: MSB, no Companding, data transfer starts with MSB first
- **XDATDLY**: 1BIT, 1 Bit Data Delay (data begins after frame synchronisation puls)
- **XFRLEN1**: 1 Word per Frame
- **XWDLEN1**: 16BIT, 16 Bit TX Word Length

Serial Port Control Register, zweiter Zugriff:

```
McBSP0_SPCR     = 0x0040 1000;
```

Bewirkt die folgende Änderung:

- **GRST**: NO, Sample Rate Generator is taken out of Reset

Serial Port Control Register, abschließender Zugriff:

```
McBSP0_SPCR     = 0x0041 1000;
```

Bewirkt die folgende Änderung:

- **XRST**: NO, Transmitter is enabled (not in Reset)

Schrittweise Initialisierung der McBSP1-Register für die Audiodatenübertragung zwischen DSP und Codec:

Eine **Grundinitialisierung** beginnt mit dem Reset des McBSP1 via Serial Port Control Register:

```
McBSP1_SPCR     = 0x0000 0000;
```

Bringt die folgenden McBSP1-Funktionseinheiten in den Reset:

- Frame Sync Generator
- Sample Rate Generator
- Transmitter
- Receiver

Sample Rate Generator Register:

```
McBSP1_SRGR     = 0x0000 0000;
```

Der Sample Rate Generator wird nicht benötigt und deshalb auch nicht initialisiert.

Pin Control Register:

```
McBSP1_PCR      = 0x0000 0003;
```

Bewirkt die folgenden Einstellungen:

- XIOEN: SP, Standard Serial Port Operation (not GPIO)
- **FSXM**: EXTERNAL, External Frame Synchronisation for TX
- **FSRM**: EXTERNAL, External Frame Synchronisation for RX
- **CLKXM**: INPUT, External TX-Clock
- **CLKRM**: INPUT, External RX-Clock
- **FSXP**: ACTIVEHIGH, Frame Synchronisation Pulse is active-high for TX
- **FSRP**: ACTIVEHIGH, Frame Synchronisation Pulse is active-high for RX
- **CLKXP**: FALLING, TX Data driven on falling TX-clock-edge
- **CLKRP**: RISING, RX Data sampled on rising RX-clock-edge

Receive Control Register:

```
McBSP1_RCR      = 0x0001 00A0;
```

Bewirkt u.a. die folgenden Einstellungen:

- **RPHASE**: SINGLE, Single Phase Frame
- RCOMPAND: MSB, no Companding, data transfer starts with MSB first
- RFIG: NO, RX Frame Synchronisation Pulse restarts transfer
- **RDATDLY**: 1BIT, 1 Bit Data Delay (data begins after frame synchronisation puls)
- **RFRLEN1**: 1 Word per Frame
- **RWDLEN1**: 32BIT, 32 Bit RX Word Length

Transmit Control Register:

```
McBSP1_XCR      = 0x0001 00A0;
```

Bewirkt u.a. die folgenden Einstellungen:

- **XPHASE**: SINGLE, Single Phase Frame
- XCOMPAND: MSB, no Companding, data transfer starts with MSB first
- XFIG: NO, TX Frame Synchronisation Pulse restarts transfer
- **XDATDLY**: 1BIT, 1 Bit Data Delay (data begins after frame synchronisation puls)
- **XFRLEN1**: 1 Word per Frame
- **XWDLEN1**: 32BIT, 32 Bit TX Word Length

An dieser Stelle erfolgt sinnvoller weise ein Schreibzugriff auf das Data-Transmit-Register DXR, um möglichen Inhalt zu löschen (siehe dazu beispielsweise Funktion `void mcbsp1_init()` in `sinetone1.c` im ersten Labor-Projekt). Anschließend wird der Betrieb des McBSP1 via Serial Port Control Register aktiviert.

Serial Port Control Register:

```
McBSP1_SPCR     = 0x0041 0001;
```

Bewirkt u.a. die folgenden Einstellungen:

- FRST: YES, Frame Sync Generator in Reset
- GRST: YES, Sample Rate Generator in Reset
- **XINTM**: XRDY, Transmit Interrupt XINT1 is driven by XRDY-Bit (Bit 17 in `McBSP1_SPCR`)
- **XRST**: NO, Transmitter is enabled (not in Reset)
- RJUST: RZF, Right-justify and zero-fill MSBs in DRR
- CLKSTP: DISABLE, no Clock Stop mode (normal clocking)
- **RINTM**: RRDY, Receive Interrupt RINT1 is driven by RRDY-Bit (Bit 1 in `McBSP1_SPCR`)
- **RRST**: NO, Receiver is enabled (not in Reset)

LEDs und Schalter auf dem DSK6713:

Dem Anwender stehen auf dem Starter-Kit vier USER-LEDs und vier USER-SWITCHES zur freien Verfügung. Mit diesen lässt sich z.B. der Programmablauf visualisieren bzw. beeinflussen.

Der Zugriff erfolgt über die Adresse `0x9008 0000` bzw. den in der Datei `c6713dsk.h` definierten Alias `IO_PORT` wie folgt:

```
*(unsigned volatile int *)0x90080000
```

bzw.:

```
*(unsigned volatile int *)IO_PORT
```

(Erläuterungen siehe Infoblatt ‚Code Composer Studio...', Kapitel ‚Direkter Zugriff auf adressierbare Prozessor- und Peripherie-Control-Register')

Die vier LEDs werden über die Bits 1...4 des 32 Bit Datenwortes an dieser Adresse eins-aktiv angesteuert.

		Bit 32 ... 17 16 ... 1
LED0 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1 bin
LED1 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx1x bin
LED2 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx x1xx bin
LED3 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx 1xxx bin
LEDs on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx 1111 bin
LEDs off	=	xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 bin

Die Zustände der vier User-Switches werden über die Bits 5...8 des 32 Bit Datenwortes an dieser Adresse eingelesen. Die Kodierung ist hier low-aktiv.

		Bit 32 ... 17 16 ... 5 ... 1
SW0 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xxx0 xxxx bin
SW1 on	=	xxxx xxxx xxxx xxxx xxxx xxxx xx0x xxxx bin
SW2 on	=	xxxx xxxx xxxx xxxx xxxx xxxx x0xx xxxx bin
SW3 on	=	xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx bin

Um die LEDs an verschiedenen Stellen des Programms einzeln ein- und wieder ausschalten zu können, muss man sich den aktuellen Status in einer globalen Variablen merken. Die gewünschte Änderung erfolgt dann durch bitweises 'verodern' bzw. 'verunden' einer Änderungsmaske mit dem bisherigen Status.

Vor dem ersten Zugriff muss der Status initialisiert werden, beispielsweise durch:

```
led_status = (*(unsigned volatile int *)IO_PORT = 0x00000000)
```

danach lässt sich z.B. die LED3 wie folgt einschalten:

```
led_status = (*(unsigned volatile int *)IO_PORT = 0x00000008 | led_status)
```

LED3 wieder ausschalten:

```
led_status = (*(unsigned volatile int *)IO_PORT = 0x00000007 & led_status)
```

Das Auslesen der Stellung eines einzelnen Schalters erfolgt ebenfalls durch ‘verunden’ des gelesenen Datenwortes mit einer Maske und einem anschließenden Vergleich.

z.B. USER_SW3 gedrückt, Ausdruck wahr:

```
((*(unsigned volatile int *)IO_PORT>>4) & 0x08) != 8
```

Folgende **#defines** solcher Ausdrücke stehen Ihnen in den Labor-Projekten zur Verfügung (beim ersten in `sinetone1.c`):

```
LED0_on, LED1_on, LED2_on, LED3_on, LEDs_on
LED0_off, LED1_off, LED2_off, LED3_off, LEDs_off
USER_SW0, USER_SW1, USER_SW2, USER_SW3
```