Der Digitale Signalprozessor TMS320C50

Robert Dörfel

05.06.2003

Inhaltsverzeichnis

1 Allgemeine Eigenschaften

Der 1992 erschienene DSP TMS320C50 von Texas Instruments soll als praktisches Beispiel für die Signalprozessoren dienen. Er besitzt folgende allgemeine Eigenschaften:

- *Fixpunkt-DSP*
  Der TMS320C50 ist ein Fixpunkt-DSP, der als Zahldarstellung die 2-er Komplementdarstellung mit einer Breite von 16 Bit benutzt.

- *Harvard-Architektur*
  Er ist nach der Harvard-Architektur gebaut, was bedeutet, dass es zwei Busse gibt - einen für das Programm und den anderen für die Daten. Allerdings werden nach außen hin nur 16 Pins für die Adresse und 16 Pins für die Daten gelegt, was Kosten bei der Produktion erspart. Diese Zusammenlegung der beiden Busse nach außen hin, nennt Texas Instruments die „modifizierte Harvard Architektur“.

- *4-Level-Pipelining auf dem Datenpfad*
  Dies bedeutet, dass er in einem Takt den Befehl aus dem Programm-Speicher holt und im nächsten Takt dekodiert, und zusätzlich schon den nächsten Befehl einliest. Im darauffolgenden Takt holt er den Operanden um im folgenden Takt die Operation auszuführen, während er schon den nächsten Befehl dekodiert und den übernächsten Befehl aus dem Speicher holt. Das gesamte Schema ist in Abbildung 1 dargestellt.

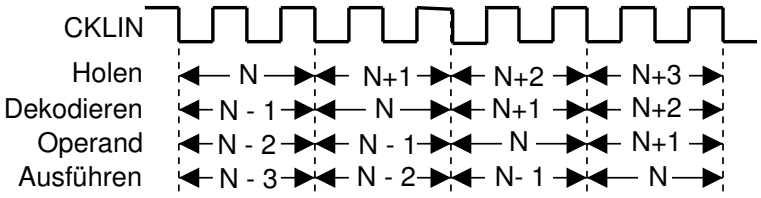


Abbildung 1: Pipelining

2 Architektur

Der DSP TMS320C50 besitzt auf dem Chip folgende zwei Hauptbestandteile: den On-Chip Speicher und die Central Processing Unit (CPU), die in den nächsten Abschnitten weiter erklärt werden.

2.1 On-Chip Speicher

Der TMS320C50 besitzt im Chip selbst Speicher, auf den er in einem Takt zugreifen kann, und somit einen schnellen Zugriff auf die Daten und Programme hat. Der Speicher teilt sich in folgende drei Teile auf:

- 1056 × 16 Bit dual-access Daten-RAM
  Auf diesen Speicherteil kann der DSP in einem Takt gleichzeitig lesend und schreibend zugreifen. Deshalb ist er besonders für Daten geeignet.

- 9K × 16 Bit single-access Daten- oder Programm-RAM
  Dieser Speicherteil dient für die Programme und bietet zusätzlich noch Platz für größere Daten.

- 2K × 16 Bit Programm-ROM
  Das ROM ist für Programme gedacht. Da der Speicher nicht flüchtig ist, kann der DSP sich mit diesem Speicherteil initialisieren. Allerdings müssen die fertigen Programme Texas Instruments geschickt werden, die sie anschließend auf das ROM brennen.

2.2 Central Processing Unit (CPU)

Die CPU läßt sich wiederum in drei Teile unterteilen: die Systemkontrolle, die Central Arithmetic Logic Unit (CALU) und die Parallel Logic Unit (PLU).

2.2.1 Systemkontrolle

Die Systemkontrolle sorgt dafür, dass der Befehl aus dem Speicher geholt wird, um anschließend dekodiert und ausgeführt zu werden. Ebenfalls kümmert sie sich um Programmverzweigungen und Unterprogrammaufrufe.

2.2.2 Central Arithmetic Logic Unit (CALU)

Die CALU ist die Recheneinheit, in der die wichtigsten arithmetischen Funktionen ausgeführt werden. Die superskalare Architektur ist hier besonders auffällig. Denn die CALU unterteilt sich in die die drei unterschiedlichen Einheiten: die Schieber, der Multiplizierer und die Arithmetic/Logic Unit (ALU).

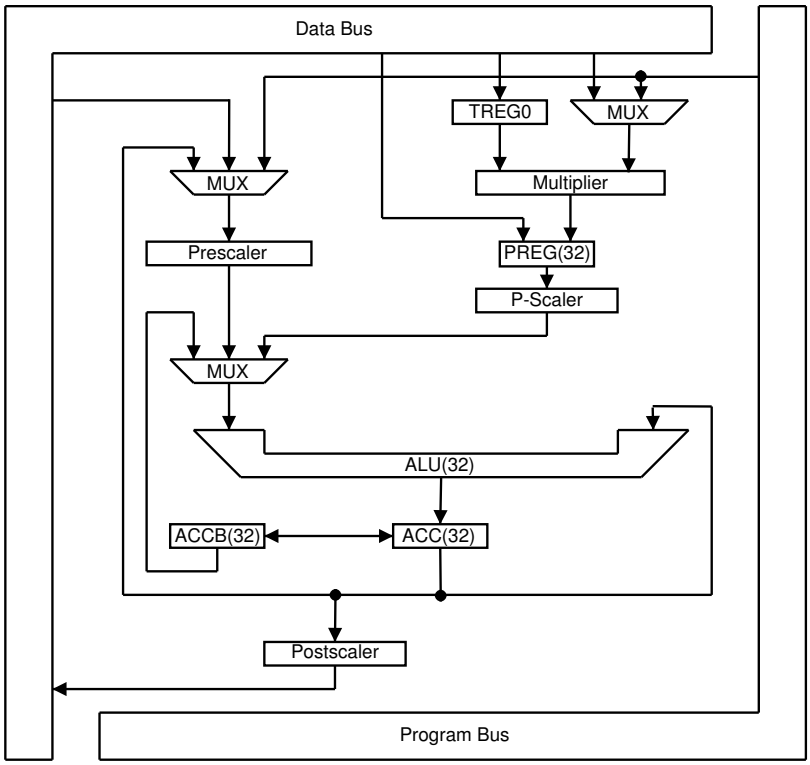


Abbildung 2: Central Arithmetic Logic Unit (CALU)

Schiebe-Einheiten Die Schiebe-Einheiten sind dafür zuständig, dass die Operanden, bevor sie von der ALU verarbeitet werden, nach links oder rechts (ohne die ALU zu benutzen) geschoben werden können. Wie aus der Abbildung 2 ersichtlich befinden sich eine der Schiebe-Einheiten vor der ALU. Diese sorgt dafür, dass der erste Operand, der aus dem Speicher kommt oder der Akkumulator selbst ist, um gewisse Bits nach links oder rechts geschoben werden. Ebenfalls kann das Produkt-Register, bevor es auf den Akkumulator aufaddiert wird, geschoben werden. Wenn der Akkumulator zurück in den Speicher geschrieben wird, besteht ebenfalls die Möglichkeit, das Ergebnis nach links oder rechts zu verschieben.

Multiplizierer Der Multiplizierer führt eine 16 Bit $\times$ 16 Bit Multiplikation in einem Takt aus und speichert das Ergebnis in dem 32 Bit breiten Product Register (PREG). Der erste Faktor ist immer das 16 Bit breite Temporary Register 0 (TREG0). Der zweite Operand kommt entweder vom Daten Bus oder vom Programm Bus.

Arithmetic/Logic Unit (ALU) Die 32 Bit ALU ist für die verschiedenste Operationen, von denen sie die meisten in einem Takt ausführt, zuständig. Sie addiert, subtrahiert, führt logische Operationen wie die UND-/ODER-Operation, Schiebe- und Rotier-Operationen aus. Der erste Operand ist immer der 32 Bit breite Akkumulator. Der zweite Operand ist entweder der Akkumulator-Puffer, das Produkt Register oder ein Operand vom Daten oder vom Programm Bus.

2.2.3 Parallel Logic Unit (PLU)

Die Parallel Logic Unit ist eine zur ALU parallele Logik-Einheit, die Und-, Oder- und Exklusiv-Oder-Operationen neben der ALU ausführen kann. Der Vorteil dabei ist, dass der Akkumulator den aktuellen Wert beibehalten kann, während die Logik-Operation in der PLU ausgeführt wird. Der erste Operand ist ein Wert aus einer beliebigen Speicherzelle, die entweder direkt oder indirekt[1] adressiert wird. Der zweite Operand ist entweder das DBMR-Register oder ein Immediate Operand[1].

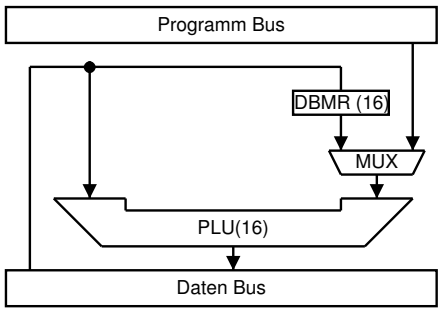


Abbildung 3: Parallel Logic Unit

3 Adressierung

Der Adressraum geht von der Adresse 0h bis FFFFh (16 Bit breite Adressen). Allerdings hat der TMS320C50 keine 64K Wörter Speicher. Deshalb ergibt sich folgende Belegung des Speichers, die in Abbildung 4 dargestellt ist. Dabei ist zu beachten, dass RAM, CNF und OVL Statusbits sind, die während

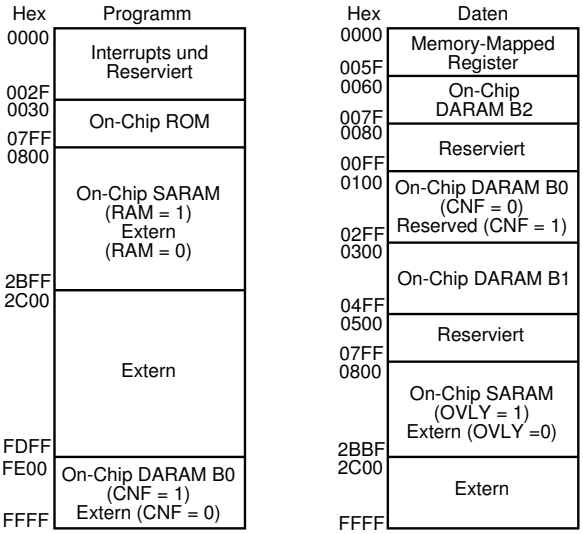


Abbildung 4: Adressraum

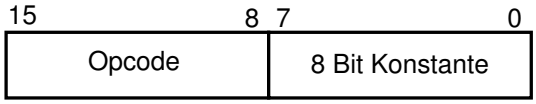
[1]Die Adressierungsarten sind in Kaptitel 3 erklärt

der Laufzeit in den Statusregistern PMST und ST1 gesetzt werden können und sich dann entsprechend der Adressraum ändert.

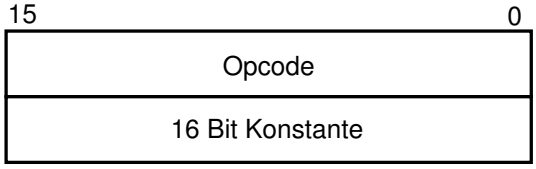
3.1 Short/Long Immediate Adressierung

Bei der Immediate Adressierung steht der Operand direkt hinter dem Befehl im Programmspeicher. In dem Assemblercode zeigt das Symbol # vor dem Operand an, dass es sich um einen Immediate-Operanden handelt. Dabei gibt es zwei Arten von der Immediate Adressierung:

- *Short Immediate Adressierung*
  Hier handelt es sich um ein 8 Bit Wert, der in den unteren 8 Bit des Befehlswortes steht.



- *Long Immediate Adressierung*
  Dabei ist der Immediate-Operand ein 16 Bit Wert der hinter dem Befehlswort in der nächsten Speicherzelle im Programmspeicher steht.



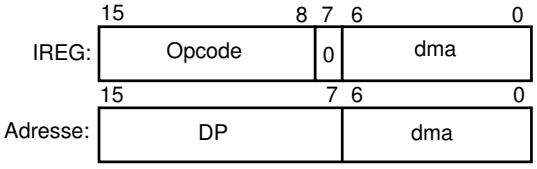
Welche der beiden Adressierungen verwendet wird, entscheidet der Assembler während der Assemblierung: Ist der Immediate-Operand ein Wert der sich durch eine 8 Bit Zahl darstellen läßt, dann wird die Short Immediate Adressierung verwendet, ansonsten die Long Immediate Adressierung.
Beispiele:

- `ADD #32h`
  Hier wird der Wert 32h auf den Akkumulator addiert, wobei die Short Immediate Adressierung verwendet wird, weil der Wert 32h noch mit 8 Bit dargestellt werden kann.

- `ADD #FFFh`
  Hier wird der Wert FFFh auf den Akkumulator addiert. Allerdings wird die Long Immediate Adressierung verwendet, da der Wert FFFh nicht mit 8 Bit dargestellt werden kann.

3.2 Direkte Adressierung

Der gesamte Adressraum ist in 512 Seiten, die jeweils 128 Wörter haben, eingeteilt. Mit dem 9 Bit breiten Data Memory Page Pointer (DP) werden die Seiten angesteuert. Die Offset-Adresse steht in den 7 niedrigsten Bits des Befehlwortes. Diese Bits werden auch Data Memory Address (dma) genannt. Somit ergibt sich für die direkte Adressierung folgendes Bild:



Beispiele:
`ADD 56h; DP = 30h`
Die Adresse lässt sich wie folgt errechnen: $30h \cdot 80h + 56h = 0f56h$. Somit wird der Wert mit Adresse 0f56h auf den Akkumulator aufaddiert.

3.3 Memory-Mapped Adressierung

Wie in der Abbildung 4 über den Adressraum zu erkennen ist, werden im Datenraum in den ersten Wörtern die Register des DSP eingeblendet. Auf diesem Weg können den Registern Werte zugewiesen werden, indem der Data Page Pointer auf 0 gesetzt wird, und mit der entsprechenden Offset-Adresse das Register angesprochen wird. Für diesen Zugriff auf die Register bietet der DSP die Befehle `LAMM` und `SAMM` an, mit denen die Register gelesen beziehungsweise den Registern Werte zugewiesen werden können, ohne dass man den DP auf null setzen muss.

3.4 Indirekte Adressierung

Die indirekte Adressierung geschieht über die so genannten acht Auxiliary Register (AR0 - AR7), die jeweils 16 Bit breite haben, und eine Adresse enthalten. Zusätzlich gibt es noch den 3 Bit breiten Auxiliary Register Pointer (ARP), der angibt, welche der acht AR's gerade verwendet wird. Für die Manipulation dieser Register gibt es zusätzlich die Auxiliary Register Arithmetic Unit (ARAU), die der ALU wiederum Arbeit abnimmt, indem sie die Manipulation der Auxiliary Register übernimmt. Die ARAU kann folgende Operationen ausführen:

- Den ARP auf einen neuen Wert setzen.
- Das aktuelle AR inkrementieren oder dekrementieren.
- Das aktuelle AR um das Index Register (INDX) erhöhen oder erniedrigen.
- Das aktuelle AR um das Index Register (INDX) mit einer umgekehrten Übertragsrichtung erhöhen oder erniedrigen.

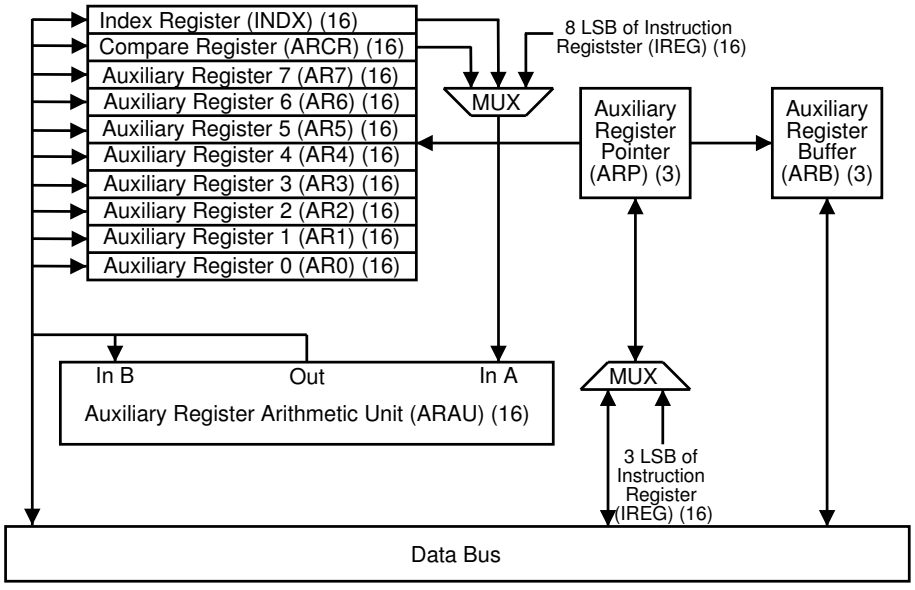


Abbildung 5: Die Auxiliary Register

Im Assemblercode bedeutet * als Operand, dass der Operand indirekt adressiert wird. Somit wird mit dem Wert an der Adresse, auf die das aktuelle AR zeigt, gerechnet. Ein Zusätzliches + oder - bedeutet, das das aktuelle AR nach der Operation inkrementiert oder dekrementiert wird. Falls hinter dem * ein 0+ oder 0- steht, wird das Indexregister nach der Operation aufaddiert oder abgezogen. Ein BR vor der 0 bedeutet, dass das Indexregister mit einer umgekehrten bertragsrichtung aufaddiert beziehungsweise subtrahiert wird. Nach dieser Kombination der indirekten Adressierung kann noch abgegrenzt durch ein

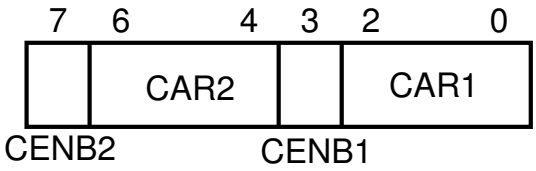
Komma der neue Wert des ARP übergeben werden.
Beispiele:

- `ADD *`
  Addiert den Speicherinhalt, auf den das aktuelle Auxiliary Register zeigt, auf den Akkumulator.
- `ADD *+, AR5`
  Addiert den Speicherinhalt, auf den das aktuelle Auxiliary Regsiter zeigt, auf den Akkumulator und erhöht den Wert vom aktuellen AR um eins und setzt den ARP auf 5.
- `ADD *0-`
  Addiert den Speicherinhalt, auf den das aktuelle Auxiliary Register zeigt, auf den Akkumulator und erniedrigt das aktuelle AR um den Wert des Index-Register.
- `ADD *BR0+`
  Addiert den entsprechenden Speicherinhalt auf den Akkumulator und erhöht das aktuelle AR um den Wert des Indexregister mit umgekehrten Übertragsrichtung.

3.5 Ringpuffer

Der TMS320C50 besitzt zwei Ringpuffer, die mit Hilfe der indireketen Adressierung und folgenden Registern gelöst sind:

- *CBSR1* enthält die Startadresse des ersten Ringpuffers.
- *CBER1* enthält die Endadresse des ersten Ringpuffers.
- *CBSR2* enthält die Startadresse des zweiten Ringpuffers.
- *CBER2* enthält die Endadresse des zweiten Ringpuffers.
- *CBCR* (Ringpuffer-Statusregister) hat folgenden Aufbau:



  - *CAR1* gibt an, mit welchem Auxiliary Register der erste Ringpuffer verbunden ist
  - *CENB1* zeigt an, ob der erste Ringpuffer aktiviert (=1) oder deaktiviert (=0) ist
  - *CAR2* gibt an, mit welchem Auxiliary Register der zweite Ringpuffer verbunden ist
  - *CENB2* zeigt an, ob der zweite Ringpuffer aktiviert(=1) oder deaktiviert (=1) ist

Falls einer der zwei Ringpuffer aktiviert ist, so überprüft die ARAU nach einer indirekten Adressierung, ob das in CAR1 beziehungsweise CAR2 angegebene Auxiliary Register immer noch zwischen der Start- und Endadresse des Ringpuffers ist. Falls die Adresse über der Endadresse ist, so wird das Auxiliary Register auf die Startadresse gesetzt. Ebenso wird es auf die Endadresse gesetzt, wenn es die Startadresse durch Dekrementieren unterläuft.

4 Befehlssatz

Der folgende Auszug aus dem Befehlssatz des TMS320C50 ist nur eine kleine Auswahl und dient bloß der Übersicht über die Möglichkeiten des Signalprozessors. Sie ist keinesfalls vollständig. Allerdings sieht man, wie die Befehle für die digitale Signalverarbeitung vor allem für die Filterberechnung zugeschnitten sind.

4.1 Befehlssatz für die ALU

Bei den Befehlen für die ALU ist der erste Operand immer der Akkumulator. Der zweite wird mit den vorher genannten Adressierungsarten angesprochen oder als Immediate-Operand übergeben.

Befehl	*Beschreibung*
ADD	Addiert einen Wert auf den Akkumulator
ADDB	Addiert den Akkumulator-Puffer auf den Akkumulator
CMPL	Komplementiert den Akkumulator
AND	Und-Operation
OR	Oder-Operation
ROL	Rotiert den Akkumulator nach links
ROR	Rotiert den Akkumulator nach rechts
SFL	Schiebt den Akkumulator nach links
SFR	Schiebt den Akkumulator nach rechts

4.2 Befehlssatz für den Multiplizierer

Bei Befehlen für den Multiplizierer wird zum Teil auch die ALU angesprochen: Wenn ein Multiplikation mit einer Addition ausgeführt wird, so wird das TREG0 mit dem Operanden multipliziert und das letzte Ergebnis der Multiplikation von der ALU auf den Akkumulator aufaddiert. Diese Befehle werden vor allem bei der Filterberechnung gebraucht.

Befehl	*Beschreibung*
LT	Lädt das TREG0-Register mit einem Wert
LTA	Lädt das TREG0-Register mit einem Wert und addiert das vorhergehende Produkt
MPY	Multipliziert einen Wert mit dem TREG0-Register
MPYA	Mulitpliziert wie MPY und addiert das vorhergehende Produkt auf den Akkumulator
MAC	Multipliziert und addiert Werte
MACD	Multipliziert und addiert Werte mit einem zusätzlichen Verschieben der Daten
SQRA	Quadriert einen Wert und addiert das vorhergehende Produkt

4.3 Befehlssatz für die PLU

Die PLU ist eine zu der ALU parallele Logik-Einheit, die folgende Befehle ausführen kann:

Befehl	*Beschreibung*
APL	Und-Operation
OPL	Oder-Operation
SPLK	Speichert einen long immediate Wert in den Daten-Speicher
XPL	Exklusiv-Oder-Operation

4.4 Befehlssatz für die Systemkontrolle

Hier sind neben bedingten und unbedingten Programmverzweigungen und Unterprogrammaufrufe vor allem die Wiederholungsanweisungen auffällig. Sie spielen in Kombination mit den Befehlen der ALU und des Multiplizerers eine wichtige Rolle bei der Filterbrechnung.

Befehl	*Beschreibung*
B	unbedingter Sprung
BCND	bedingter Sprung
CALL	Unterprogramm-Aufruf
CC	bedingter Unterprogramm-Aufruf
RET	Unterprogramm-Rücksprung
RETC	bedingter Unterprogramm-Rücksprung
RPT	Wiederholung der nächsten Anweisung
RPTB	Wiederholung eines bestimmten Blocks

5 Das DSP Starter Kit für den TMS320C50

Das DSP Starter Kit (DSK) von Texas Instruments besteht zum einen aus dem DSK-Board, das eine Platine mit dem DSP TMS320C50 ist. Zum anderen enthält das Paktet MS-DOS Programme, die zur Erstellung und zur Ausführung von DSP-Programmen auf der Testplatine gedacht sind. Dieses Kit wird von Texas Instruments angeboten, um Entwickler die Möglichkeit zu geben, DSP-Programme in einer Testumgebung zu entwickeln.

5.1 Software

Die Software besteht erstens aus den drei Hauptprogrammen:

- Der Assembler „dsk5a“
  Der Assembler erzeugt aus einem in Maschinensprache geschriebenem DSP-Programm eine so genannte DSK-Datei, die dann mit Hilfe des Loaders oder des Debuggers auf den DSP gespielt wird.
- Der Debugger „dsk5d“
  Der Debugger bietet die Möglichkeit, ein DSP-Programm auf den DSP zu spielen und es anschließend zu debuggen, um entsprechend Fehler zu suchen oder das Programm nachzuverfolgen.
- Der Loader „dsk5l“
  Der Loader spielt über die serielle Schnittstelle eine DSK-Datei auf den DSP und startet die Ausführung des Programms auf dem DSP.

Zusätzlich sind auch viele Beispiel DSP-Programme enthalten, die einerseits die Leistungsfähigkeit des DSP zeigen und andererseits den Entwicklern als Anschaungsmaterial dienen sollen.

5.2 Das DSK-Board

Das DSK-Board ist wie folgt aufgebaut und besitzt folgende Komponenten:

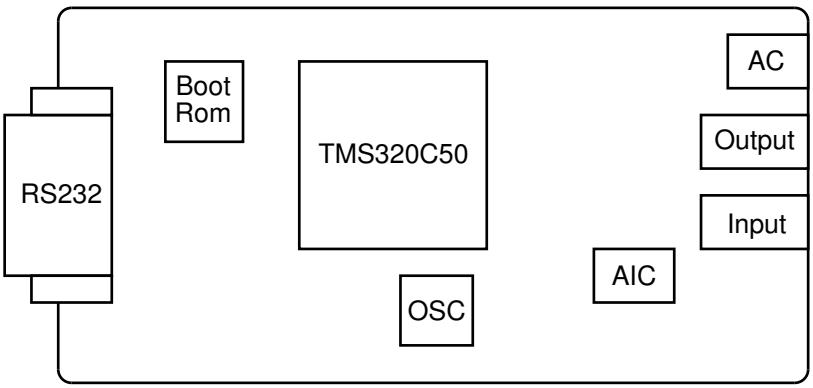


Abbildung 6: Das DSK-Board

DSP TMS320C50 ist der DSP des Boards.

Boot Rom ist ein 32K × 16 Bit ROM, das das Initialisierungsprogramm enthält. Da der Speicher des DSP bis auf das interne ROM flüchtig ist, ist es nötig den DSP nach dem Einschalten zu initialisieren. Dies geschieht mit Hilfe des Boot Rom, das entsprechende Programme für die serielle Schnittstelle enthält, damit der DSP anschließend neue Programme von dem Computer erhalten kann. Nach dem Initialisieren wird das Boot Rom wieder aus dem Adressraum genommen, und anschließend besteht kein Zugriff mehr auf das ROM.

OIC ist der Taktgeber für den DSP. Er läuft mit einer Frequenz von 40 MHz.

AIC ist der Analog/Digital und Digital/Analog Wandler, der benötigt wird, um die analogen Eingangssignale in digitale Werte umzuwandeln und die vom DSP gerechneten digitalen Werte wieder in analoge Ausgangssignale umzuwandeln. Dieser AD/DA-Wandler arbeitet mit einem Takt von 19kHz.

RS232 ist eine serielle Schnittstelle, mit der der DSP an einen Computer angeschlossen werden kann. Über sie können dann die DSP-Programme vom Computer dem DSP übertragen werden. Aber es können auch während der Laufzeit Daten zwischen DSP und Computer ausgetauscht werden.

Input ist der analoger Chinch-Eingang, von dem der DSP seine Signale erhält.

Output ist der analoge Chinch-Ausgang, an dem die vom DSP gerechneten Signale ausgegeben werden.

AC ist die 9 Volt Stromversorgung.

6 Beispielprogramme

Das erste Beispielprogramm verdeutlicht die Mächtigkeit eines MACD-Befehls (**M**ultiply and **Ac**cumulate with **D**ata move) in Kombination mit einer RPT-Anweisung. Dabei wird mit der Befehlsfolge

```
RPT #3
MACD #h0, *-
```

jeweils vier mal zwei verschiedene Werte miteinander multipliziert und die Produkte auf dem Akkumulator aufaddiert. Und zwar ist der erste Operand ab der Adresse h0 zu finden und der zweite ab der Adresse des aktuellen Auxiliary Register. Zusätzlich wird nach jeder Multiplikation die Adresse des ersten Operanden um eins erhöht und die des zweiten um eins erniedrigt. Außerdem werden die Daten, die indirekt adressiert werden, jeweils in die nächst höhere Speicherzelle kopiert.
Für die nächsten Beispiele werden auf dem Computer per MP3-Player Musikstücke abgespielt, die über

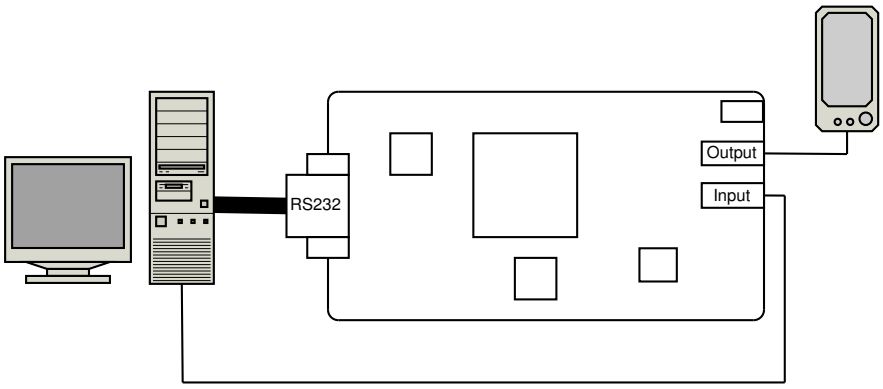


Abbildung 7: Dsk-Board-Konfiguration für die Beispielprogramme

den Line-Out des Rechners in den analogen Eingang des DSK-Boards gelangen. Das DSK-Board verarbeitet diese Signale, und gibt die veränderten Musikdaten über den analogen Ausgang wieder aus. Der Ausgang ist mit einem Lautsprecher verbunden, der die verarbeiteten Musikstücke hörbar macht. Als Beispielfilter werden ein Tiefpass und ein Hochpass verwendet, der entsprechend die hohen beziehungsweise tiefen Töne herausfiltert. Ebenso verleiht eine Hallfilter dem Musikstück ein Echo.

Literatur

[1] Texas Instruments: *TMS320C5x DSK - Applications Guide;* Texas Instruments, 1997

[2] Texas Instruments: *TMS320C5x DSP Starter Kit - User's Guide;* Texas Instruments, 1994

[3] Texas Instruments: *TMS320C5x - User's Guide;* Texas Instruments, 1993