

Realisierung eines autonomen Messsystems zur Schwingungsanalyse

Beitrag zur Embedded Intelligence 2001

Autoren: Jens Kolupa, Phil Alder

Kontakt: DSPeSpecialists GmbH, Rotherstraße 22, 10245 Berlin

T: 030/467805-0, F: 030/467805-99, E: Jens.Kolupa@DSPeSpecialists.de

1. Einleitung

Der vorliegende Beitrag beschreibt das Konzept und die Realisierung eines autonomen Messsystems zur Schwingungsanalyse. Einsatzgebiet ist die permanente Langzeit-Diagnose der Fahrzeugflotte des Hochgeschwindigkeits-Schienenverkehrs. Mit den durch das System gewonnenen Daten soll primär eine Verbesserung der Instandhaltungsabläufe erfolgen (off-line-Auswertung). Zusätzlich ermöglicht das System auch eine on-line-Überwachung von Fahrwerkskomponenten.

Im folgenden wird ausgehend von den Systemanforderungen die Umsetzung als kombiniertes DSP-PC-System beschrieben. Dabei wird sowohl auf die Hardware-Struktur wie auch auf die implementierte Software eingegangen, die eine Aufteilung der Verarbeitungsvorgänge unter Ausnutzung der jeweiligen Stärken der Sub-Systeme ermöglicht. Ein wichtiger Aspekt ist dabei eine effiziente Host-Kommunikation zwischen DSP und PC, die speziell für das System entwickelt wurde.

2. Anforderungen

Die Anforderungen ergaben sich zum einem aus dem Einsatzbereich des Systems und zum anderem aus den zu messenden Größen.

Die Messaufgabe besteht in der Aufnahme und Verarbeitung von 20 Beschleunigungssignalen, 8 Temperatursignalen und 8 weiteren Signalen. Aufgrund der Frequenzen der Signale liegt die benötigte Abtastrate bei 32 kSamples/s. Die Verarbeitung und Analyse der Signale im Zeit- und Frequenzbereich muss in Echtzeit erfolgen, wobei die Vorgänge sowohl zeit- wie auch ereignisgesteuert sein können. Diese hohen Anforderungen an die Rechenleistung sind am besten mit einem DSP-System zu erfüllen.

Andererseits soll das System autonom, d.h. ohne Benutzereingriff bis zu 2 Jahre laufen und dabei die Messdaten zur späteren off-line-Auswertung sichern. Über eine Standard-Schnittstelle kann eine Diagnose erfolgen, und es können Konfigurationsparameter geändert werden. Hierbei sprechen insbesondere die anfallenden Datenmengen und die Notwendigkeit eines standardisierten Dateisystems für ein PC-System.

Der Einsatz im mobilen Bereich stellt auch erhöhte Anforderungen an Robustheit, Stabilität und Energieeffizienz. Alle diese Anforderungen lassen sich durch die Verwendung eines kooperativen Multiprozessorsystems bestehend aus einem Embedded-PC und einem DSP-System erfüllen. Der Rückgriff auf einen großen Anteil von off-the-shelf-Komponenten ermöglicht zusätzlich eine schnelle Entwicklung des Gesamtsystems.

3. Systemaufbau

Das gesamte System ist modular in einem 19"-Rack aufgebaut. Die einzelnen Subsysteme, Netzteil, PC, Festplatte und DSP inkl. Signalaufnahme sind als Einschubkassetten ausgeführt. Dies ermöglicht eine separate Entwicklung und Inbetriebnahme wie auch einen schnellen Austausch einzelner Komponenten.

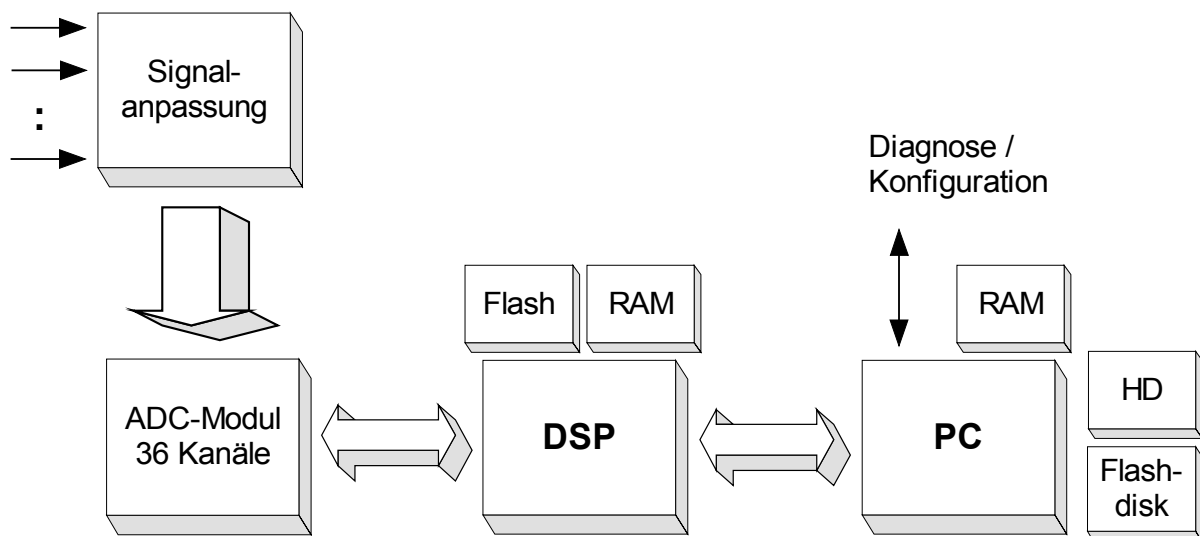


Abbildung 1: Systemaufbau

Als PC findet ein handelsüblicher Einplatinen-Embedded-PC der Pentium-Klasse mit den üblichen Schnittstellen Verwendung. Externe Erweiterungskarten lassen sich über die AT96-Backplane anschließen. Für die temporäre Speicherung der Messdaten ist eine Flash-Disk angeschlossen, die dauerhafte Archivierung erfolgt auf einer Festplatte. Hier wird aus Gründen der Robustheit und des Energieverbrauchs auf eine Laptop-Platte zurückgegriffen. Mit dieser wurden schon frühzeitig erfolgreiche Testfahrten in einem ICE-Zug vorgenommen, da die verlässliche Abspeicherung der Daten eine wichtige Eigenschaft des Systems ist.

Auf dem DSP-Modul kommt ein Texas Instruments TMS320C6701 Prozessor zum Einsatz. Er ist das bisher leistungsstärkste Mitglied (1300MIPS / 1 GFLOPS) der 32 bit Floating Point Familie C67x und verfügt über 6 parallele FP-Ausführungseinheiten. Auf dem Modul kann der Prozessor auf 256 kByte SRAM, 512 kB Flash und 32 MB externes SDRAM zugreifen.

Die Verbindung zum PC erfolgt über die Anbindung des Host-Port-Interfaces (HPI) des DSPs an den AT96-Bus des PCs. Dies ermöglicht eine hohe Transferrate bei der Verwendung eines robusten, einfachen Interfaces. Hierauf wird ausführlich in dem Abschnitt 5 eingegangen.

Die Signale der Sensoren werden über eine Signalkonditionierung an die Eingangsbereiche der verwendeten Analog-Digital-Umsetzer angepasst. Neben einer Pegelanpassung von Gleichspannungssignalen findet hier auch eine Anpassung an die ICP-Eingänge der Beschleunigungsaufnehmer und an PT1000-Temperaturaufnehmer statt. Die Wandlerkarte wurde unter den Gesichtspunkten der erforderlichen Kanalanzahl, der Umsetzrate von 32 kSamples/s und einer einfachen Anbindung an den DSP konzipiert und realisiert. Als AD-Umsetzer kommt der AD73360 von Analog Devices zum Einsatz. Er enthält 6 synchrone Kanäle und kann durch sein serielles Daten-Interface mit weiteren Wandlern in Reihe geschaltet werden (kaskadiert), um die Anzahl von Leitungen zum Prozessor möglichst gering zu halten. Aufgrund der maximalen Transfer-Rate des seriellen, synchronen Interfaces des DSPs (McBSP) von 50 Mbit/s, lassen sich bis zu 3 Bausteine in Reihe bei der erforderlichen Abtastrate von 32 kSamples/s anschließen. So wurde ein Modul mit 18 synchronen Kanälen entwickelt, das an einen seriellen McBSP-Port des DSPs angeschlossen wird. Durch die Verwendung zweier Module ergibt sich eine maximale Kanalanzahl von 36.

4. Software-Struktur

Die Aufgabenverteilung auf die Prozessoren und deren Umsetzung in Software erfolgt unter Ausnutzung der jeweiligen Stärken der Subsysteme. Auf dem DSP wurde u.a. die Datenaufnahme mit Hilfe der ADCs und die aufwendige Signalverarbeitung im Zeit- und Frequenzbereich implementiert. Die DSP-Ablaufsteuerung wird durch den PC getriggert und besitzt einen Sub-Timer zur Koordinierung der internen DSP-Funktionen. Die Datenübergabe an den PC erfolgt asynchron über auf dem C6x-HPI aufgesetzte Kommunikationskanäle.

Sämtliche Prozesse auf dem DSP sind deterministisch, das heißt, es sind sowohl die Abfolge wie auch die Ausführungszeiten der Prozesse bekannt und konstant. Die zeitliche Abfolge der Prozesse hängt von dem DSP-internen Timing und von dem jeweiligen Systemkommando ab, welches vom PC an den DSP über das HPI gesendet wurde. DSP-Kommandos und Prozesse werden dabei in einer State-Machine (Super-Loop-Technik) verwaltet, so daß auf einen aufwendigen Scheduler verzichtet werden konnte. Synchronisiert werden die meisten DSP-internen Abläufe von dem Einleseprozess der ADCs. Die serielle Schnittstelle des DSPs wird dabei als Slave von dem Mastertakt der ADCs getrieben. Der Datentransfer von der seriellen Schnittstelle in einzelne, kanalabhängige Sortierpuffer im DSP-Speicher erfolgt per DMA. Haben alle Sortierpuffer einen gewissen Füllstand erreicht, übernimmt ein weiterer DMA-

Kanal das Umkopieren in die eigentlichen Dateneingangspuffer. Dieses zweistufige DMA-Verfahren ist notwendig, da der Adressraum des ersten DMAs nicht ausreicht, um die notwendigen Datenmengen für die Signalverarbeitung bereitzustellen. Durch die Verwendung von DMA-Kanälen für das mehrkanalige Einlesen und Sortieren der Eingangssamples wird nahezu keine CPU-Zeit benötigt, wohl aber andere DSP-Ressourcen wie z.B. Interrupts, der DMA-Bus sowie das externe Memory-Interface. Stehen nach jeweils einer Sekunde in allen Kanälen die Daten in den Eingangspuffern bereit und liegt ein Meßkommando vom PC vor, wird in der State-Machine die eigentliche Signalverarbeitung angestoßen. Ein einzelner Meßzyklus des Systems dauert 30 Sekunden. Daher wurden die Implementierung der Algorithmen und die Ablaufsteuerung in der DSP-State-Machine auf 30 dieser 1-Sekunden-Zyklen ausgelegt. Das Triggern, d.h. das Setzen der Nullmarke übernimmt der PC als Master durch Senden eines neuen Meßkommandos. Der DSP muß jedoch seinen "30 Sekunden-Zyklus" etwas eher als der PC beenden, um für einen neuen Meßzyklus bereit zu sein. Da für beide Systeme quartz stabile Impulse als Taktgeber zum Einsatz kommen, ist die DSP-Abtastzeit bzw. der Mastertakt des ADCs so gewählt, dass diese Bedingung eingehalten wird.

Je nach Signalart kommen unterschiedliche DSP-Algorithmen zur Anwendung. Die Verarbeitung im Zeitbereich läuft sowohl für die Temperatursignale wie auch für die Beschleunigungssignale durchgängig, das heißt jede Sekunde gleich ab. Auf den Beschleunigungskanälen wird zusätzlich eine Verarbeitung im Frequenzbereich vorgenommen. Dazu wird über eine gewisse Zeit (< 30 Sekunden) eine FFT-Frequenzanalyse der Signale vorgenommen, aus deren Ergebnissen anschließend statistische Kenngrößen berechnet werden. Sämtliche Ergebnisse aus dem Zeit- und Frequenzbereich werden ständig über das HPI-Interface an den PC abgesetzt.

Der PC wird unter Linux betrieben, um die gute Unterstützung dieses Betriebssystems bei der Prozesssteuerung, dem File-Handling und bei der Systemsteuerung einfach nutzen zu können. Außerdem übernimmt der PC als Master die Aufgaben der Steuerung des Mess- und Analyseablaufes (zeit- und ergebnisgesteuert). Die vom DSP gelieferten Ergebnisse werden auf dem PC ausgewertet und einer Entscheidungslogik zugeführt. Die Resultate dieser Logik stellen die eigentlichen Diagnosedaten des Systems dar. Sie werden abhängig von ihrer Bedeutung weiterverarbeitet (z.B. Ausdünnung oder Generierung einer Meldung) und auf der Festplatte abgespeichert. Mittels einer seriellen- oder Ethernet-Verbindung kann man sich auf dem Linux-System einloggen und Änderungen an der Konfiguration vornehmen oder Daten entnehmen.

5. DSP-Host-Kommunikation

Der Host-Kommunikation kommt eine zentrale Bedeutung innerhalb des Systems zu. Sie dient einerseits der Steuerung und Konfiguration des DSP-Programms, andererseits dem bidirektionalen Transfers mehrerer unabhängiger Datenströme. Der PC dient hierzu als Schnittstellen-Master, wobei beidseitig eine Interrupt-gesteuerte Kommunikation möglich ist. Ent-

sprechende Treiber wurden sowohl für den DSP als auch für den PC (hier Linux und DOS) entwickelt. In einer Erweiterung ist auch das Booten des DSP über den PC möglich.

Die DSP-Typen C6201, C6211, C6701 und C6711 aus der Texas Instruments TMS320C6x Familie unterstützen einen effizienten Datenaustausch mit einem Host-Prozessor über ein spezielles Host-Port-Interface (HPI) plus einen dedizierten Direct-Memory-Access (DMA) Kanal bzw. interner Adressgenerierungslogik. Diese von der DSP-CPU unabhängig agierenden C6x-Peripherieelemente ermöglichen es einem Host-Prozessor, auf DSP internen und externen Speicherraum zuzugreifen, ohne die DSP-CPU in ihrer Echtzeitverarbeitung zu unterbrechen bzw. zu blockieren. Da die C6x-CPU hierbei als Slave keine direkte Kontrolle über das HPI hat, bzw. eine im Hintergrund ablaufende Datenübertragung auch nicht beeinflussen kann, muß durch spezielle Steuer- und Kontrollmechanismen ein bidirektionaler Datenaustausch realisiert werden.

Ausgehend von diesen Anforderungen sind in dem System ein C6x-HPI zu PC-ISA-Bus Hardware-Interface und verschiedene Software-Layer implementiert, die nachfolgend einzeln beschrieben werden.

5.1. C6x-Host-Port-Interface (HPI)

Das C6x-HPI ist ein 16 bit breites paralleles Interface, über das ein Host-Prozessor auf DSP-interne und -externe Speicherinhalte lesend oder schreibend zugreifen kann. Das HPI stellt hierzu eine interne Hardwarelogik, 16 explizite, bidirektionale Datenleitungen und einige Kontrollpins zur Verfügung, um Datenübertragungen unabhängig vom C6x-External-Memory-Interface-Bus bzw. ohne Unterbrechung der C6x-CPU durchführen zu können. Der Host-Prozessor ist dabei immer Master, d.h. alle Datenübertragungen müssen durch den Host angefordert und gesteuert werden. Das C6x-HPI stellt dann als Slave mit Unterstützung des DMA-Controllers die angeforderten Daten bereit bzw. speichert übergebene Daten in vorher zu definierende DSP-Speicherbereiche. Die C6x-CPU hat keinen direkten Zugriff auf die HPI-interne Logik, kann jedoch den Host durch ein low-aktives Interrupt-Request-Pin (HINT) um Aufmerksamkeit bitten. Der Host kann ebenfalls einen Interrupt auf dem DSP anfordern, indem er das DSPINT-Bit in dem HPI-Controll-Register setzt.

Der Host kann nicht direkt auf den DSP Speicher zugreifen, sondern muß hierzu die entsprechenden HPI-Register beschreiben bzw. konfigurieren. Der eigentliche, HPI-interne Transfer vom HPI-Datenregister in den DSP-Speicher erfolgt über einen dedizierten Auxiliary-DMA-Kanal der C6201 / C6701 CPU bzw. über eine interne Anbindung des HPI-Interfaces an die Address-Generation-Units im Falle der C6211 / C6711 CPUs. Ein Host-Zugriff auf den DSP-Speicher unterbricht somit nicht die Programmausführung der DSP-CPU, benötigt aber trotzdem interne Ressourcen, wie z.B. den Auxiliary-DMA-Kanal oder das External-Memory-Interface (falls die Daten im externen Speicher liegen). Kommt es dabei zu einem Zugriffskonflikt zwischen dem HPI/DMA und der CPU bzgl. einer bestimmten Adresse oder

der zu belastenden Bussysteme, werden die Anforderungen entsprechend einer konfigurierbaren Priorität bedient.

Die Hardware-Anbindung des C6x-HPI an PC-ISA-Bus (hier AT96) läßt sich mit wenigen externen Bauelementen relativ einfach realisieren, da die meisten Signale zur Steuerung des C6x-HPI vom ISA-Bus bereits direkt bereitgestellt werden. Das HPI kann dabei über eine Adressdekodierlogik entweder in den Speicher- oder in den IO-Adressbereich des PCs gemappt werden. Durch Verwendung eines zweiten Adressdekoders kann der DSP vom PC aus Software-gesteuert in den RESET-Zustand versetzt und dort auch gehalten werden.

5.2. Low-Level-Kommunikationsmethoden

Ein Kommunikationsvorgang zwischen PC und DSP über den C6x-HPI muß vom PC als Master gesteuert bzw. initiiert werden. Der PC muß hierzu folgende Zugriffssequenz auf die HPI-Register ausführen:

- Initialisierung des HPI-Controll-Registers (HPIC)
- Initialisierung des HPI-Address-Registers (HPIA)
- Datenzugriff über das HPI-Data-Register (HPID)

Jeder Schreib- oder Lesezugriff auf HPID löst einen C6x-Auxiliary-DMA-Zyklus aus, welcher die benötigten Daten bereitstellt bzw. in den C6x-Speicher schreibt. Basierend auf der HPI-Zugriffssequenz wurden Low-Level-Kommunikations-Routinen für einen wortweisen (32 bit) bzw. blockweisen Speicherzugriff in C auf dem PC implementiert. Die Blockzugriffe nutzen die HPI-Autoincrement-Funktion, welche mehrere, sequentielle Zugriffe nach einer einmaligen Initialisierung des HPI-Address-Registers unterstützt. In diesem Fall wird das HPIA intern nach dem Datenzugriff erhöht bzw. die HPI-Hardware führt einen automatischen Pre-fetch auf das nächste 32-bit-Datum zur Erhöhung der Datenübertragungsrate aus. Bei einzelnen Wort-Zugriffen muss das HPIA jedoch für jeden Zugriff neu initialisiert werden.

Die C6x-CPU hat als Slave keine direkte Kontrolle über das HPI bzw. über den Ablauf der Host-Routinen auf dem PC. Außerdem kann der C6x eine im Hintergrund ablaufende HPI-Datenübertragung nicht direkt erkennen oder beeinflussen. Hierzu muss ein bidirektionales Handshaking bzw. eine Interrupt-Signalisierung zwischen DSP und Host implementiert werden:

- Der DSP kann durch das Setzen des HINT-Bits im HPIC einen Interrupt auf dem Host anmelden und damit z.B. eine erforderliche Datenübertragung vom DSP zum Host anfordern. Der PC sollte nach Abarbeitung einer Service-Routine den Interrupt-Request des DSP bestätigen, indem er das HINT-Bit nochmals beschreibt und damit wieder löscht.
- Der PC kann durch das Setzen des DSPINT-Bits im HPIC einen Interrupt auf dem DSP anfordern. Der DSP sollte nach Abarbeitung einer Service-Routine die Interrupt-Anforde-

zung des Hosts bestätigen, indem er das DSPINT-Bit in HPIC nochmals beschreibt und damit wieder löscht.

Ausgehend von diesen Überlegungen wurden neben den Low-Level Kommunikationsroutinen noch einige C-Funktionen für das Setzen und Testen der Handshaking-Bits implementiert.

5.3. High-Level Kommunikationsmethoden

Die Low-Level-Kommunikationsmethoden begründen die Basis für jede Datenübertragung zwischen PC und DSP. Auf der Slave-Seite (DSP) ist für den eigentlichen Datenaustausch kein Code erforderlich, da alle Zugriffe durch den Master (PC) initiiert und kontrolliert werden und dann unabhängig von der C6x-CPU-Aktivität ablaufen.

Der DSP kann jedoch durch Setzen des HINT-Bits in HPIC den PC auffordern, vom C6x bereitgestellte Daten (einzelne 32 bit-Wörter oder größere, zusammenhängende Datenblöcke) unter Kontrolle des PC-Hostes über das HPI abzuholen. Der DSP definiert (oder besser erzeugt) hierzu ein sogenanntes DSP_Exchange_Packet (DEP) an einer dem PC bekannten (oder über einen vom C6x verwalteten PEP-Zeiger flexibel adressierbaren) C6x-Speicherstelle:

ID	C6xADDRESS	C6xLENGTH	FLAG	CHECKSUM
----	------------	-----------	------	----------

Abbildung 2: Aufbau eines DEP

Die Checksumme kann als Quersumme, Parity oder CRC erzeugt werden und sichert die Gültigkeit eines übertragenen DEPs. Der DSP erzeugt ein DSP_Exchange_Packet in seinem Adressraum durch Aufruf einer C-Funktion und fordert dann ggf. einen HINT-Interrupt beim Host-PC an. Die ISR des PCs leitet diesen Interrupt an die PC-Applikation oder einen speziellen Datenübertragungskernel weiter. Es ist dann Aufgabe der PC-Applikation bzw. des Kernels, auf eine Datenübertragungsanforderung vom DSP zu reagieren, indem sie:

1. Das DEP über die HPI-Primitive von einer dem PC bekannten DEP-Startadresse ausliest
2. Die Gültigkeit des DEPs bzw. seiner Elemente über die Checksum überprüft
3. Anhand der ID den Datentyp bzw. die auszuführende Operation erkennt
4. Entsprechend der C6xADDRESS bzw. der LENGTH den eigentlichen Datenblock aus dem DSP-Speicher über das HPI abholt
5. Die Anforderung bestätigt, indem sie das HINT-Bit in HPIC und die ID im DEP löscht.

Die über das HPI eingelesenen Daten werden anschließend abhängig von ihrem Inhalt in Filestrukturen geschrieben, auf welche die eigentliche Linux-Applikation (Datenauswertung) zugreift.

Ein ähnlicher Mechanismus kann benutzt werden, um vom PC aus Informationen, d.h. Datenblöcke, in den C6x-Speicher zu übertragen und dann in einem zweiten Schritt den DSP über

die bereits erfolgte Datenübertragung durch ein PEP (PC_Exchange_Packet) zu informieren. Der PC erzeugt und überträgt hierzu ein PEP via HPI an eine dem DSP bekannte Speicherstelle im DSP-Adressraum (fest, oder über einen vom PC verwalteten, flexibel adressierbaren PEP-Zeiger). Dann fordert der PC einen Interrupt beim DSP über das HPI an. Die DSP-ISR kann dann entsprechend der PEPID, der C6xADDRESS bzw. der C6xLENGTH den übertragenen Datenblock identifizieren und ggf. weiterverarbeiten.

Es ist jedoch auch möglich, einen Datenübertragung PC zu DSP ohne PEP, d.h. ohne eine Benachrichtigung der C6x-CPU über das HPI auszuführen, um z.B. einen Bootload oder eine Neukonfiguration des DSP-Systems vorzunehmen. Eine klare Unterscheidung zwischen benachrichtigter und „blinden“ Übertragung obliegt dem Master, d.h. der PC-Applikation. Daher sollten möglichst immer applikationsspezifische PC-Funktionen auf den Low-Level Kommunikationsmethoden aufgesetzt werden, um spezielle Funktionen zu realisieren.

Bei sehr kurzen Datenblöcken (wie z.B. bei einfachen Konfigurations- und Synchronisationskommandos) kann die PEP- bzw. DEP-Elemente C6xADDRESS, C6xLENGTH und CHECKSUM direkt für Daten oder Flags verwendet werden, anstatt Verweise auf einen Datenblock anzulegen. So existieren z.B. einige „kurze“ Kommandos zum Übertragen bzw. zur Abfrage von Statusinformationen oder zum Setzen und Löschen von Steuerflags, welche die DSP-Applikation in ihrer Abarbeitung beeinflussen.

5.4. Anwendung der Kommunikationsmethoden im System

Wie im Kapitel Software-Struktur beschrieben durchläuft der DSP nach dem Erhalt eines Synchronisations-Kommandos vom PC als Master einen festen Zyklus:

1. Nach Erhalt des Synchronisationskommandos vom PC beginnt die mehrkanalige Messwertaufnahme und Analyse der Meßdaten in Echtzeit.
2. Stehen die ersten Ergebnisse zur Verfügung, erzeugt der DSP ein DEP mit einer dem Datentyp zugeordneten ID und Verweisen auf die C6xADDRESS bzw. die C6xLENGTH im DSP-Speicher.
3. Das neue DEP wird in einem DEP-Ringpuffer eingetragen und dann wird dem PC per HINT-Interrupt signalisiert, dass neue Daten zur Abholung bereitstehen.
4. Der DSP wartet jedoch nicht, ob / bis der PC diese Daten abgeholt hat, sondern führt weitere Echtzeitanalysen durch und setzt dann weitere Datenpakete / DEPs zum PC hin ab, bis der aktuelle Meßzyklus beendet ist.

Diese Vorgehensweise stellt sicher, dass der DSP nicht in seiner Echtzeitverarbeitung durch den PC bzw. die Datenübertragungen auf dem ISA-Bus blockiert wird. Die Datengenerierung durch den DSP und die Datenübertragung durch den PC erfolgen daher in diesem System zeitversetzt und asynchron zueinander, da dem DSP auch keine Information über die Auslastung / Status des PC zur Verfügung stehen.

Es sollte jedoch immer sichergestellt werden, daß der PC die Daten bereits abgeholt hat, bevor der DSP den jeweiligen Datenpuffer im nächsten Zyklus wieder allokiert. Der PC löscht hierzu das erste Datenfeld bzw. die ID nach einer erfolgten Datenübertragung. Trifft der DSP dann doch einmal auf einen Datenblock bzw. ein PEP, welches noch nicht übertragen wurde, werden Doppelpufferungs- bzw. Datenverwerfungs-Strategien eingesetzt.

6. Zusammenfassung / Ausblick

Der vorliegende Beitrag beschreibt die Komponenten und den Aufbau eines autonomen Messsystems zur Schwingungsanalyse. Neben der Signalverarbeitung in Echtzeit ist dabei auch die Langzeit-Diagnose im beschriebenen Anwendungsgebiet von großer Bedeutung. Diese Anforderungen wurden durch die Verwendung eines Multiprozessorsystems bestehend aus DSP und PC erfüllt. Dabei ist eine effektive Kommunikation beider Prozessoren sehr wichtig.

Das System wurde im Auftrag des Instituts für Sicherheitstechnologie (ISTec) GmbH entwickelt; Endkunde und Nutzer ist die Deutsche Bahn AG. Nach Voruntersuchungen bezüglich der Robustheit von einzelnen Komponenten ist ein Erprobungsmuster des Systems seit Ende 2000 in einem ICE2-Mittelwagen installiert.