Mikrocontroller in der Signalverarbeitung

Stephan Kutzner

2018

Inhaltsverzeichnis

1 Einführung

Im folgenden wird der Einsatz von Mikroprozessoren zur Signalerfassung- und Verarbeitung behandelt. Hierzu wird in Kapitel 2 der Aufbau von Mikroprozessoren erläutert. Dabei wird auf Besonderheiten eingegangen, welche für die Funktionalität von Mikroprozessoren in der Signalverarbeitung besonders tragende Rollen spielen.
In dem folgenden Kapitel 3 wird das Thema 'Abtastung' behandelt. Nach der Erörterung der theoretischen Grundlagen wird das Nyquist-Theorem bewiesen, um schließlich in der praktischen Anwendung betrachtet zu werden. Wichtig sind hierbei die Themen 'Input/Output' sowie 'A/D-Wandler'.
Das nächste Kapitel 4 befasst sich mit dem Thema 'D/A-Wandler'. Hierzu werden zunächst die Grundlagen dieses Konzeptes behandelt. Anschließend wird das Verfahren der D/A-Wandlung selber erläutert. Danach folgt ein Vergleich von Mikroprozessoren und Mikrocontrollern, welcher Grundlage für das Verständnis eines 'Digital Signal Processors (DSP)' liefert.

2 Mikroprozessoren

Für das Verständnis eines DSPs ist das Verständnis der zu Grunde liegenden Hardware unumgänglich. Hierzu wird zunächst in Kapitel 2.1 der Begriff 'Echtzeit' erklärt. In der praktischen Anwendung ist die Signalverarbeitung ein Echtzeit-Problem, weswegen dieser Begriff eine wichtige Grundlage bildet. Des weiteren führt dieses Kapitel noch andere ebenfalls wichtige Begriffe ein, beispielsweise 'harte Echtzeit' oder auch 'Deadlines'.
Kapitel 2.2 befasst sich mit der Bauweise von Speicherelementen. Dazu zählen 'Random-Access-Memory' (RAM), 'Read-Only-Memory' (ROM), 'Flash'-Speicher (EEPROM) sowie Caches. Insbesondere wird auf Vor- und Nachteile dieser Optionen eingegangen.
Bestandteile des Kapitels 2.3 sind verschiedene Architekturen, wobei Harvard sowie Von-Neumann eine tragende Rolle spielen. Im Folgenden befasst sich Kapitel 2.4 mit weitergehender Architektur, welche entscheidende Vorteile für die Verarbeitung von Signalen bietet. Hierzu zählen Pipelines und Multicore Architekturen.
Für die Verarbeitung von Daten sind grundlegende Befehle notwendig. In welcher Form und in welchem Umfang diese elementaren Befehle vorliegen wird in Kapitel 2.5 betrachtet. Besondere Aufmerksamkeit wird der praktischen Anwendung in der Signalverarbeitung gewidmet.
Kapitel 2.6 behandelt das Thema 'Scheduling', also der zeitliche Ablauf verschiedener Prozesse. Hierbei werden Verfahren wie 'Round-Robin', 'First-Come-First-Serve' sowie 'Earliest Deadline First' analysiert.
Schließlich werden weitere Optionen wie 'Mailboxes' sowie Gefahren, beispielsweise 'Deadlocks' in Kapitel 2.7 aufgezählt. Besonders wird auf Themen eingegangen, die einen Mikroprozessor in der Signalverarbeitung stören könnten.

2.1 Grundlegende Begriffe

Um den Begriff der Echtzeit zu verstehen, ist es notwendig, zunächst die Bezeichnung 'System' zu definieren:

Definition 1. *System*
Ein System beschreibt die Zuordnung von Eingabevektoren auf Ausgabevektoren. [?, p. 3]

Dabei müssen weder alle denkbaren Eingaben angenommen, noch alle denkbaren Ausgaben ausgegeben werden können. Beispielsweise besteht kein Grund dazu, dass ein Geldautomat auf Audiosignale reagiert. Die Tastatur und die Karteneingabe beschränkt somit den Eingaberaum auf eine endlich Anzahl an Eingaben.
Andere Systeme besitzen lediglich Sensoren als Eingabegeräte, beispielsweise einen Wärmesensor. Bei diesem besteht der Eingaberaum aus allen möglichen Ausgabewerten des Sensors.

Der Ausgaberaum besteht aus jeder möglichen Repräsentation der Ausgabe des Systems. Im Falle des Geldautomaten wäre es das Display, die Rückgabe der Karte sowie die Ausgabe der Geldscheine.
Doch die Ausgabe muss von außen nicht direkt sichtbar sein. Ebenso können Daten den Ausgaberaum beschreiben. Im Beispiel des Wärmesensors könnte das System mit einer Heizung verbunden sein, die mithilfe der von dem System ausgegebenen Werte die Temperatur reguliert.

'Echtzeit' begrenzt den Zeitraum, in dem die Ausgabe auf eine bestimmte Eingabe folgen muss.

Definition 2. *Echtzeit*
Ein Echtzeit-System ist ein Computer-System, welches bestimmte Antwortzeiten('Deadlines') einhalten muss. Andernfalls ist mit gravierenden Folgen, darunter auch einem Ausfall zu rechnen. [?, p. 5]

Zur Verdeutlichung werden die vorangegangenen Beispiele erneut genutzt.
Man stelle sich vor, der Geldautomat würde das Geld erst nach einigen Minuten ausgeben. Dies wäre nicht gefährlich, jedoch unangenehm und würde die Zuverlässigkeit erheblich einschränken.
Anders sieht es im Falle des Temperaturreglers aus. Sensor und Heizung seien nun mit einem Aquarium verbunden. Die Ausgabe des Systems, die Heizung auszuschalten verzögert sich nun um einige Stunden. Bis die eigentliche Regulierung erfolgt ist das Wasser wahrscheinlich schon bei weitem zu heiß. Die Auswirkungen sind hier weitaus drastischer als bei dem Geldautomaten.

Definition 3. *Deadline*
Als Deadline wird eine einer Aufgabe zugeordneter Zeitspanne bezeichnet, welche die größtmögliche Zeit zum Beenden dieser Aufgabe angibt.

Es ist offensichtlich, dass nicht alle denkbaren Systeme in Echtzeit agieren müssen. Bei vielen Anwendungen ist diese Bedingung jedoch unumgänglich, wie auch bei der Signalverarbeitung. Sollte die Verarbeitung einer Zeitreihe länger als die Dauer der Zeitreihe benötigen, so kann dies nicht als Echtzeitsystem bezeichnet werden.

Beweis 1. *Deadline*
Sei t_i die Dauer einer Zeitreihe i. Entsprend sei $p_i = t_i + d_i$ die Verarbeitungszeit der Zeitreihe, bestehend aus der Länge sowie einem Delay d_i.
Sei t_l eine fixe Zeit, die die maximal zugelassene Verzögerung einer Zeitreihe angibt, also die Deadline.
Werden nun mehrere Eingaben nacheinander berechnet, so summiert sich der Delay auf:
$d = \sum_{i=1}^{N} d_i$
Angenommen, der Delay sei konstant. So wird nach $\frac{t_l}{d_i}$ Eingaben der Delay die Deadline überschreiten. Dies gilt für jede endliche Deadline.

Jedoch muss die Deadline nicht der Länge der Zeitreihe entsprechen. Hört man Musik, so möchte man nicht, dass die Ausgabe Sekunden nach dem eigentlichen Abspielen erfolgt, sondern ohne merkliche Verzögerung. Auch von der Eingabe mittels einer Tastatur bis zur Visualisierung des Zeichens auf dem Bildschirm sollte nicht mehr als wenige Zehntel einer Sekunde vergehen. Die tiefere Bedeutung des Begriffes 'Echtzeit' wird deutlich, wenn verschiedene Unterarten definiert werden. Im Folgenden werden drei Kategorien vorgestellt: Weich, hart sowie fest.

Definition 4. *Weiche Echtzeit*
Ein weiches Echtzeitsystem zeichnet sich dadurch aus, dass auch viele verpasste Deadlines nicht zu einem Versagen des Systems führen können, sondern lediglich die Performance beeinträchtigen können.

Im Gegensatz hierzu stehen die harten Echtzeitsysteme:

Definition 5. *Harte Echtzeit*
Ein hartes Echtzeitsystem zeichnet sich dadurch aus, dass bereits eine verpasste Deadline zu einem kompletten Systemversagen führen kann.

Das wohl beste Beispiel ist die Vorstellung, dass Sensoren die Leistung eines Atomkraftwerkes überwachen sollen. Steht nun eine Überlastung bevor, so muss das System binnen weniger Augenblicke reagieren - sollte diese Deadline nicht eingehalten werden, so kann dies eine Katastrophe zur Folge haben. Auf der anderen Seite steht der Geldautomat, dessen verpasste Deadlines auch in großer Zahl weder Systemversagen- noch Katastrophen mit sich führen, jedoch massive Performanceverluste bedeuten.

Definition 6. *Feste Echtzeit*
Ein festes Echtzeitsystem zeichnet sich dadurch aus, dass wenige verpasste Deadlines lediglich die Performance beeinträchtigen, eine größere Zahl jedoch zu Systemversagen führen könnte.

Es existiert keine eindeutige Bestimmungsvorschrift für die Art eines Systems. Die Grenzen sind nicht fest definiert und so kann ein System je nach Betrachter als 'fest' oder 'hart' bezeichnet werden.
Jedoch ist diese dritte Klasse sinnig, um nicht nur zwischen einem sehr robusten System und einem, dessen Deadlines um jeden Preis eingehalten werden müssen unterscheiden zu können.
Ein weiterer wichtiger Begriff, welcher vor allem zur Abschätzung der Zuverlässigkeit eines Systems sowie des Verhaltens von bestimmten Aufgaben genutzt wird lautet 'Determinismus'.

Definition 7. *Deterministisches System*
Ein deterministisches System zeichnet sich dadurch aus, dass für jeden möglichen Zustand und jeden möglichen Eingabevektor der Folgezustand sowie Ausgaben bestimmt werden können.

Ist dies für ein System der Fall, so lässt sich sein gesamtes Verhalten bestimmen und jede mögliche Verletzung von Deadlines ist im Vorfeld bestimmbar. Im Folgenden wird jedoch gezeigt, dass viele Optimierungsmethoden Nichtdeterminismus mit sich ziehen.

2.2 Speicher

Es lassen sich verschiedene Arten von Speicher angeben. Dabei spielen nicht nur Schreib- und Lesezeiten, Kosten und die Flächennutzung eine Rolle, sondern auch Energieverbrauch und die Möglichkeit des Schreibens neuer Daten.

Hierzu werden im Folgenden auf die Arten RAM, ROM sowie Flash-Speicher eingegangen. Ebenfalls wird der Unterschied zwischen dynamischem und statischem RAM deutlich gemacht.

2.2.1 Random Access Memory

Unter Random Access Memory (RAM) versteht man eine unbeständige Art der Datenspeicherung. Diese Art von Speicher lässt sich in der Regel beliebig oft neu beschreiben, löschen und auslesen. All diese Vorgänge sind schneller als bei anderen Arten von Speicher, weshalb jede Form eines modernen Rechners eine Form von RAM besitzt, meist in der Regel eines Caches, der im folgenden noch beschrieben wird.
Durch die unbeständige Art der Datenspeicherung eignet sich RAM-Speicher jedoch nicht für persistente Speicherung. Sobald die Stromversorgung unterbrochen wird, werden die Speicherzellen geleert und die Daten somit verworfen.

2.2.2 Read-Only Memory

Im Gegensatz dazu steht Read-Only Memory, auch ROM. Wie der Name bereits andeutet, eignet sich dieser nicht zur Änderung der gespeicherten Daten. Es ist möglich, den Speicher zu leeren und zu beschreiben, jedoch ist die maximale Anzahl dieser Vorgänge limitiert und um einiges langsamer als beim RAM. Dies ist jedoch nicht bei jeder Form von ROM möglich. Mask ROM beispielsweile lässt sich nur bei der Herstellung beschreiben. Ebenso ist das Auslesen der Daten langsamer.
Die Speicherung ist persistenter Natur, weswegen ROM in der Regel fest verdrahtet Firmware für ein Gerät bereit stellt, welche nicht alternierbar ist. Ein weiterer Vorteil ist der weitaus geringere Preis.

2.2.3 Flash

Flash-Speicher, auch Flash-EEPROM (Electrically Erasable Programmable Read-only Memory) ist eine erweiterte Form des normalen ROMs. Im Gegensatz zu EEPROM können einzelne Bytes nicht überschrieben werden.
Dank der Natur des ROM-Speichers sind Daten jedoch persistent gespeichert. Ebenso ist der Energieverbrauch verhältnismäßig gering, und die Kosten sind ebenfalls minimal. Somit bietet der Flash-Speicher eine robuste, kosteneffiziente Lösung für eingebettete Systeme.

3 Abstatung

3.1 Theorie

3.2 Beweis

3.3 Bedeutung

3.4 Periphere Schnittstellen

3.5 Analoger und digitaler In-/Output

4 DA Wandler

4.1 Theorie

4.2 Beweis

4.3 Bedeutung

4.4 Mikroprozessoren - Mikrocontroller

4.5 Digital Signal Processor (DSP)