

Verteiltes Rechnen auf Voyager und Galileo

Seminar: Rechnertechnik in der Raumfahrt

Fabian Braun
ge69tas@mytum.de

Lukas Schröder
lukas.schroeder@tum.de

1. GESCHICHTLICHES

1.1 Geschichtlicher Kontext

Nach dem Apollo Programm und einigen Sondenmissionen wie Viking und Pioneer, sollten die äußeren Planeten des Sonnensystems genauer untersucht werden. Der erste Mann auf dem Mond war ein Amerikaner, somit war der Wettlauf ins All, zumindest was die zivile Eroberung des Weltalls betraf, vorerst zu Ende. Die NASA litt infolgedessen die nächsten Jahre unter heftigen Budgetkürzungen. Diese hatten neben dem generellen Design der in den 70er und 80er Jahren gestarteten Sonden Voyager und Galileo auch Auswirkungen auf deren Rechnerarchitektur.

Dennoch schaffte es die NASA, uns die bis heute hochaufgelötesten Bilder von Jupiter, Saturn, Uranus und Neptun anzufertigen. Auch das berühmte "Family Portrait" der Voyager 1 Sonde, in welchem man fast das komplette Sonnensystem sieht, wurde so ermöglicht. Letztendlich schafften es beide Sonden sogar als einzige menschengemachten Objekte, unser Sonnensystem zu verlassen.

Neben der Rechnerarchitektur wird in dieser Arbeit auch der geschichtliche Kontext der Sonden, sowie die Ziele ihrer Mission beleuchtet. Ebenfalls wird auf die Unterschiede zur Viking Sonde eingegangen, die erheblichen Einfluss auf die Entwicklung von Voyager und Galileo hatte. Im Anschluss wird das verteilte Rechnen auf Voyager und Galileo, ein wichtiger Beitrag zum Erfolg der Mission, genauer betrachtet. Als Abschluss wird das verteilte Rechnen auf beiden Sonden verglichen sowie in Bezug mit den Nachfolgemissionen gesetzt.

1.2 Die Voyager Mission

1.2.1 Hintergrund

Die NASA hatte bereits in den späten 60er Jahren Erfolg bei der Erkundung der Planeten Venus und Mars. Die Voyager Sonden sollten zunächst auch nur eine weitere Nummer innerhalb des Mariner Programms sein. Zunächst war nur geplant, nach den ersten Erforschungen von Jupiter und Saturn durch die Pioneer Sonden, diese Planeten genauer zu untersuchen. Erst später kamen als weitere Ziele auch Uranus und Neptun in Betracht.¹²³


¹Vgl. [34] unbekannt. *Mission and spacecraft library Mariner*.

²Vgl. [33] unbekannt. *Mariner*. 2017.

³Vgl. [35] unbekannt. *Mission and spacecraft library Pio-*

1.2.2 Ziele

Die Hauptziele der Voyager Mission war die Erforschung der Gasplaneten Jupiter und Saturn. Die erste der beiden Sonden, Voyager 2, sollte später noch die entfernten Planeten Uranus und Neptun untersuchen. Die zweite Sonde, Voyager 1, konnte aufgrund einer anderen Bahn diese Planeten nicht erreichen. Durch den geänderten Kurs sollte diese Sonde den besonders großen Saturn Mond Titan genauer erforschen.

Abgesehen von wissenschaftlichen Zielen, sollte mit der "Golden Record" auch enstein der Menschheit gefeiert werden. Auf dieser vergoldeten Kupferscheibe waren verschiedenste kulturelle als auch wissenschaftliche Erkenntnisse unserer Zivilisation verewigt. Auf ihr befanden sich unter anderem Begrüßungen auf 50 unterschiedlichen Sprachen, Musik aus allen Kontinenten der Erde sowie Naturaufnahmen.⁴⁵

1.2.3 Planung

Im Sommer 1965 zeigten Berechnungen, dass bis 1980 die einmalige Gelegenheit bestand, alle 4 äußersten Planeten des Sonnensystems zu besuchen. Hierzu sollten sogenannte Swing-By Manöver genutzt werden, bei welchen ein Raumfahrzeug seinen Kurs durch die Gravitation eines Himmelskörpers ändert. So konnte an Treibstoff gespart werden. Die nächste Konstellation, bei welcher dies möglich ist, liegt erst im 22. Jahrhundert vor.

Der Vorbeiflug an den Planeten Uranus und Neptun war zunächst nicht geplant. Hierbei sollte auf den Erkenntnissen der vorangegangenen Missionen bei den Planeten Venus und Mars aufgebaut werden. Das zunächst noch als Marinaer Jupiter/Saturn 1977 betitelte Projekt nahm mit dem ersten wissenschaftlichen Treffen 1972 an Fahrt auf. Im Jahr 1977 schließlich wurde das Projekt in den heutigen Namen Voyager umbenannt.⁶⁷

1.2.4 Ablauf

Voyager 2 startete am 20. August 1977 von Cape Canaveral, USA. Knapp zwei Wochen später, am 5. September 1977 startete die baugleiche Voyager 1 Sonde. Bereits anderthalb Jahre später erreichte Voyager 1 im März 1979

neer.

⁴Vgl. [21] Nelson. *Voyager Mission Overview*.

⁵Vgl. [20] Nelson. *Voyager Fast Facts*.

⁶Vgl. [22] Nelson. *Voyager Mission Timeline*.

⁷Vgl. [21] Nelson. *Voyager Mission Overview*.

⁸37.

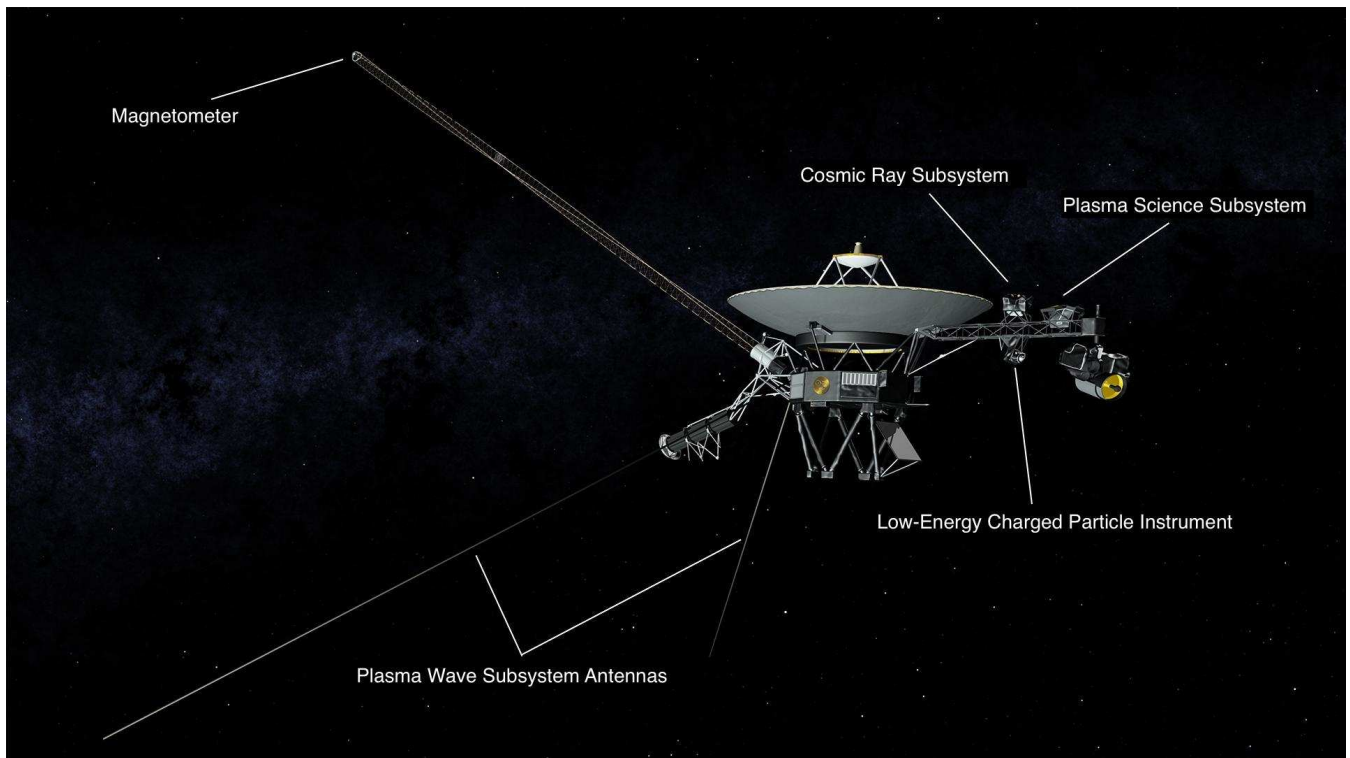


Figure 1: Voyager Übersicht ⁸

den Jupiter, Voyager 2 folgte im Juli. Der sechste Planeten unseres Sonnensystems, Saturn, wurde im Jahr 1980 von Voyager 1 und im Jahr 1981 von Voyager 2 besucht. Die 2 äußersten Planeten wurden nur noch von Voyager 2 erreicht. Da die Distanzen zwischen den äußersten Planeten deutlich zunahmen, erreichte die Sonde Uranus erst knapp 5 Jahre später. Die kürzeste Annäherung an den türkisen Planeten erfolgte am 24 Januar 1986. Den letzten Planeten, Neptun, erreichte Voyager 2 am 25. August 1989.

Im Jahre 1990 nahm Voyager 1 Sonde das berühmte "Pale Blue Dot" Foto auf. Hierzu wurde die Sonde zur Sonne ausgerichtet und nahm ein Bild des gesamten Sonnensystems auf. Die Erde kann man mit einer Größe von 0,12 Pixeln kaum erkennen. 1998 überholte Voyager 1 schließlich die Pioneer 10 Sonde und wurde damit das am weitesten entfernte, menschengemachte Gerät.

Schließlich verließen Voyager 1 im Jahr 2012 und Voyager 2 im Jahr 2018 unser Sonnensystem. Beide Sonden bewegen sich seitdem mit etwa 3,5 AE pro Jahr in den interstellaren Raum. Bei beiden Sonden funktioniert nur noch etwa die Hälfte der wissenschaftlichen Instrumente. Dennoch können noch Funksignale beider Voyager Raumfahrzeuge empfangen werden. Dies wird sich allerdings in den nächsten Jahren ändern, da die Radionukleidbatterien zur Neige gehen. Bis heute ist Voyager 2 die einzige Sonde, welche die äußersten Planeten besuchte.

⁹¹⁰

1.3 Die Galileo Mission

⁹Vgl. [22] Nelson. *Voyager Mission Timeline*.

¹⁰Vgl. [21] Nelson. *Voyager Mission Overview*.

1.3.1 Hintergrund

Nachdem in den 1970er Jahren, mit den Raumsonden Pioneer 10 & 11 und Voyager 1 & 2, bereits mehrere Missionen der NASA den Jupiter kurzzeitig besucht hatten oder es noch tun sollten, sollte eine längerfristige Mission das Jupiter-System genauer erkunden. Diese Mission war das Jupiter-Orbiter-Probe Projekt der NASA, welches später in Galileo umbenannt wurde.¹²¹³¹⁴

1.3.2 Ziele

Das primäre Ziel der Galileo Mission war die detaillierte Erkundung des Gasplaneten Jupiter und seiner vier größten Monde: Io, Europa, Ganymed und Kallisto. Außerdem wurde eine Atmosphärensonde zum Jupiter gebracht, die in die Atmosphäre eintauchen sollte, um möglichst viele Daten über diese zu sammeln. Neben der Atmosphärensonde befanden sich noch neun weitere wissenschaftliche Experimente an Bord des Galileo Orbiters. Auch auf dem Weg zu seinem Hauptziel wurden mehrere Möglichkeiten ergriffen, um noch andere Ziele zu beobachten. Dazu gehören ein Vorbeiflug an dem Planeten Venus und den Asteroiden Gaspra und Ida. Und auch der direkte Einschlag des Kometen Shoemaker-Levy 9 in den Jupiter konnte noch aufgenommen

¹¹⁸.

¹²Vgl. [36] unbekannt. *Solar System Exploration Galileo*. 2018.

¹³Vgl. [4] Dick. *Mission to Jupiter Why We Explore*. 2007.

¹⁴Vgl. [17] Meltzer. *Mission to Jupiter A History of the Galileo Project*. 2007.

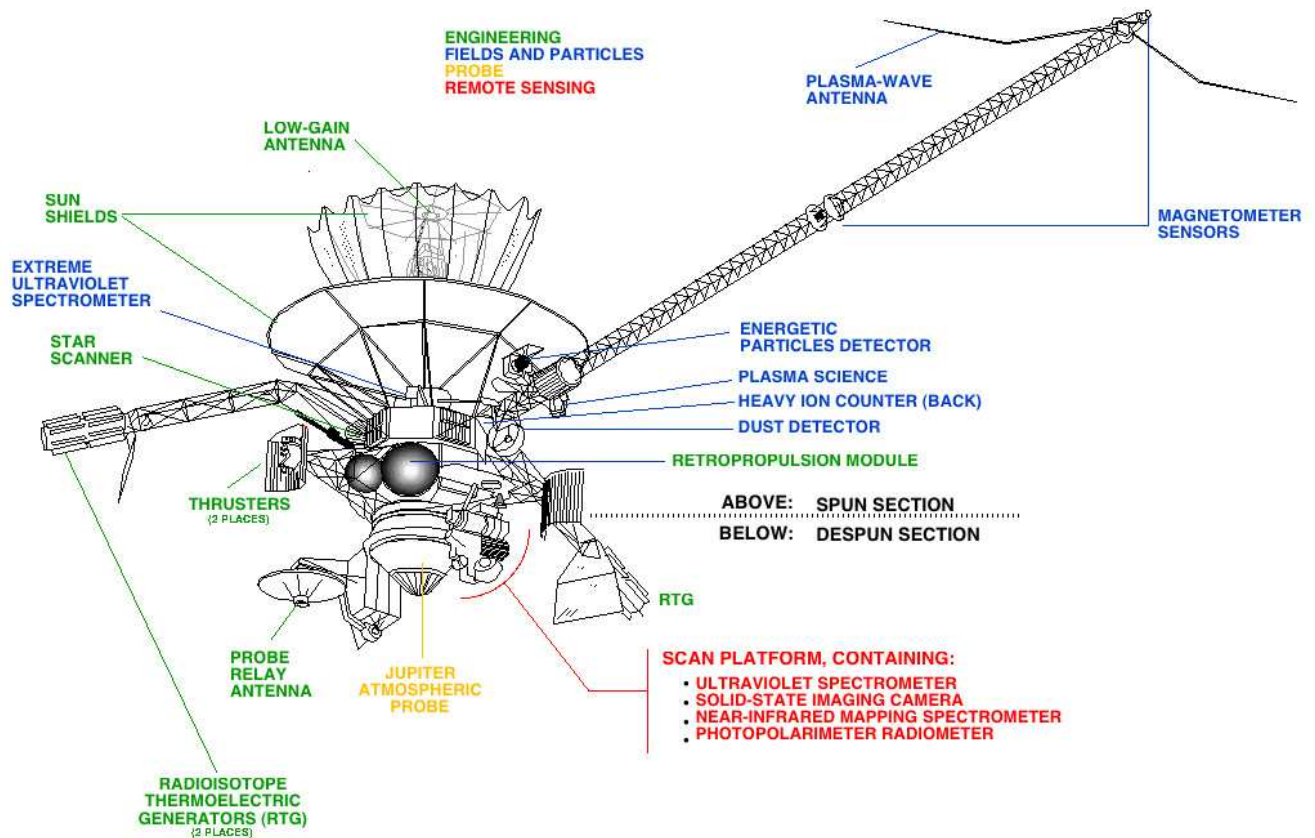


Figure 2: Galileo Orbiter ¹¹

men werden.¹⁵¹⁶¹⁷

1.3.3 Planung

Nachdem das Projekt 1977 genehmigt wurde, dauerte es noch 12 Jahre, bis die Raumsonde Galileo starten konnte. Grund dafür waren sowohl Verzögerungen bei der Entwicklung des Space Shuttles, des geplanten Startvehikels und veränderte Budgetkürzungen. Außerdem mussten nach der Challenger Katastrophe, der Explosion des Shuttles Challenger kurz nach dem Start 1986, einige Änderungen an der Mission vorgenommen werden, da es danach als zu gefährlich galt, flüssigen Treibstoff für die Raketenstufe zum Jupiter zu benutzen. Durch die Anpassungen an der oberen Raketenstufe und dem resultierenden Treibstoffverlust musste auch die Flugbahn massiv verändert werden. Statt einem direkten Weg zum Jupiter musste nun mit einem Swing-by Manöver an der Venus und zwei weiteren solcher Manöver an der Erde genug Geschwindigkeit gesammelt werden, um die Reise zum Jupiter zu schaffen. Durch die neue Flugbahn verzögerte sich die geplante Ankunft am Gasplaneten

weiter.¹⁸¹⁹²⁰

1.3.4 Ablauf

Die Galileo Raumsonde wurde am 18. Oktober 1989 an Bord des Space Shuttle Atlantis vom Kennedy-Space Center gestartet. Der Vorbeiflug an der Venus 1990 wurde genutzt, um die Instrumente der Raumsonde zu testen und einige Aufnahmen der Venus zu machen. Danach durchflog Galileo nach jedem der beiden Erdbesuche den Asteroidengürtel und konnte so beim ersten Mal detaillierte Aufnahmen des Asteroiden Gaspra machen. Beim zweiten Durchflug entdeckte man bei dem Asteroiden Ida einen kleinen Mond. 1994 konnte die Raumsonde den direkten Einschlag der Fragmente des Kometen Shoemaker-Levy 9 in den Jupiter, wenn auch aus 238 Millionen Entfernung, statt wie gedacht aus nächster Nähe, beobachten. Etwa 5 Monate vor der Ankunft am Jupiter wurde die Atmosphärensonde von der Raumsonde getrennt und flog unabhängig weiter bis sie im Dezember 1995 mit etwa 47 km/s in die Jupiteratmosphäre eintrat. Während des Abstiegs durch diese sendete die Sonde knapp eine Stunde Wetterdaten an die Raumsonde im Jupiteror-

¹⁵Vgl. [36] unbekannt. *Solar System Exploration Galileo*. 2018.

¹⁶Vgl. [4] Dick. *Mission to Jupiter Why We Explore*. 2007.

¹⁷Vgl. [17] Meltzer. *Mission to Jupiter A History of the Galileo Project*. 2007.

¹⁸Vgl. [36] unbekannt. *Solar System Exploration Galileo*. 2018.

¹⁹Vgl. [4] Dick. *Mission to Jupiter Why We Explore*. 2007.

²⁰Vgl. [17] Meltzer. *Mission to Jupiter A History of the Galileo Project*. 2007.

bit. Die Hauptsonde begann ihre Hauptmission: Elf stark elliptische Jupiterorbits, die es ermöglichten, möglichst nah an den großen Jupitermonden vorbeizufiegen. Nach dem problemlosen Abschließen dieser geplanten Ziele 1997, konnte die Galileo Raumsonde noch viele weitere Jahre ihre Mission fortsetzen und genauere Daten über Eis und Wasser auf dem Jupitermond Europa, Vulkane auf dem Mond Io und viele Erkenntnisse über die Auswirkungen von starker Strahlung auf die Hardware der Raumsonde sammeln. Nach insgesamt 34 Orbits wurde die Galileo-sonde am 21. September 2003 mit seinen letzten Treibstoffreserven in den Jupiter gelenkt, um dort in der Atmosphäre zu verglühen.^{21 22 23}

2. HARDWARE EINER AUF SICH ALLEIN GESTELLTEN SONDE

2.1 Besonderheiten der Hardwareanforderungen an Voyager hinsichtlich anderer Raumsonden

Rechnertechnik für Raumsonden zu entwickeln ist schon seit jeher eine große Herausforderung. Durch den großen Energieaufwand Geräte ins All zu bringen, spielt das Gewicht eine große Rolle. Aufgrund der begrenzten Größen von Raketen darf die Größe der Sonde auch nicht zu groß ausfallen. Durch diese Limitation muss die gesamte Hardware sehr energieeffizient sein. Ein Problem stellt die extreme Kälte im Weltall dar. Um die Rechner funktionstüchtig zu halten, müssen diese stets beheizt werden, um in einem funktionstüchtigen Zustand zu sein. Ein weiteres Problem stellt die Weltraumstrahlung dar, welche für Computer zu einem ernsthaften Problem werden können. Durch die Strahlung können Berechnungen verfälscht oder sogar der gesamte Rechner unbrauchbar werden. Neben der normalen Weltraumstrahlung waren die Voyager und Galileo Sonden auf ihrer Reise neben dem Jupiter zusätzlich dessen Strahlung ausgesetzt. Diesen Problemen war die NASA bereits auf früheren Missionen ausgesetzt und hatte entsprechende Lösungen erarbeitet. So wurden damals oft statt komplexer General-Purpose Computer noch häufig fest verdrahtete Schaltungen eingesetzt. Um die Ausfallsicherheit zu erhöhen, wurde oft nicht die neueste Technik sondern nur bewährte Computer eingesetzt. Um das Problem mit der Stromversorgung zu lösen, wurden Photovoltaikzellen verbaut, um stets Energie durch die Sonne zu gewinnen.

Die Voyager und Galileo Missionen stellte die NASA-Ingenieure dennoch vor viele neue Herausforderungen. Neben der großen Strahlung Jupiters wurde das Problem der Energieversorgung wieder kritischer. Die Energie, die auf einem Quadratmeter durch Sonnenlicht erzeugt werden kann, nimmt nicht linear ab. Diese sogenannte Solarkonstante beträgt auf der Erde 1.4 kW, 0.6 kW auf dem Mars und beim Jupiter nur 50 Watt. Spätestens bei den späteren Zielen Jupiters hätte die Energie beim Neptun nicht mehr ausgereicht (1,5 W).

Ein weiteres Problem, dem sich die NASA konfrontiert

²¹Vgl. [36] unbekannt. *Solar System Exploration Galileo*. 2018.

²²Vgl. [4] Dick. *Mission to Jupiter Why We Explore*. 2007.

²³Vgl. [17] Meltzer. *Mission to Jupiter A History of the Galileo Project*. 2007.

sah, waren große Signallaufzeiten zwischen Sonde und Bodenstation. Zwischen dem Mars und der Erde vergehen zwischen dem Senden eines Kommandos und der Antwort der Sonde in der günstigsten Konstellation knapp 20 Minuten. Die Distanz zwischen Erde und Jupiter ist nochmals deutlich größer, im besten Fall braucht das Signal für beide Richtungen eine Stunde. Die Sonde musste somit viele Sicherheitsvorkehrungen besitzen, um kritische Situationen selbstständig lösen zu können. Diese sollten im Verlaufe der Mission noch mehrmals auf die Probe gestellt werden. Auch viele Operationen wie das Fotografieren oder das Einschwenken in eine Umlaufbahn musste die Sonde autonom bewerkstelligen können.²⁴

2.2 Lösungsansätze der NASA

Die früheren Voyager Sonden verfügten, wie auch die Galileo Sonde über keine Solarpaneele. Aufgrund der im Verlauf der Reise immer größeren Distanz zur Sonne hätten diese nicht genügend Leistung zum Betrieb der Sonde geliefert. Deshalb wurden bei beiden Sonden Radeonukleidbatterien verbaut, welche Strom aus dem Zerfall von Atomen gewinnt.

Zudem konnte Energie in beiden Sonden eingespart werden, da statt einem zentralen Großrechner viele kleinere spezialisierte Computer verwendet wurden. Dadurch konnten die Aufgaben deutlich schneller und effizienter abgearbeitet werden.

Für die Ausfallsicherheit war jedes Subsystem redundant vorhanden. So gab es bei den Voyager Sonden jeden Computer doppelt. Gab es mit einem Rechner Probleme konnte einfach auf den anderen Rechner umgeschaltet werden. Dies geschah bei beiden Missionen durch spezielle Routinen automatisch. Durch den Ansatz des verteilten Rechnens wurde zusätzliche Sicherheit geschaffen. Ein Zentralrechner ist aufgrund der Möglichkeit eines Ausfalls eine große Schwachstelle. Bei vielen verteilten Computern kann der Ausfall eines Systems durch Updates gegebenenfalls kompensiert werden.

3. RECHNERARCHITEKTUR DER VOYAGER SONDEN UND GALILEO

3.1 Rechnertechnik der Voyager Sonden


Die Voyager Sonde bestand aus 3 Computersystemen, welche aus Redundanz jeweils doppelt ausgeführt waren. Die 3 Subsysteme bestanden aus dem "Flight Data System", dem "CCS Memory Subsystem" und dem "Attitude Articulation and Control System". Das CSS-Memory System ist bis auf wenige Änderungen identisch zu dem der vorangegangenen Viking Sonden. Es war für grundlegende Funktionen, die den Sondenbetrieb aufrechterhalten, verantwortlich. Das Flight Data System verarbeitete Telemetriedaten und war zudem für den Funkkontakt zur Bodenstation zuständig. Für die Lagekontrolle und Ausrichtung war schließlich das letzte System, das Attitude Articulation and Control System, zuständig. Aus Haltbarkeitsgründen, wurde stets unbenötigte Hardware abgeschaltet. Das CCS System musste, um die zum Betrieb notwendigen Dienste zur Verfügung zu stellen, stets laufen. Die anderen zwei Systeme wurden nur

²⁴Vgl. [1] Angabe. *Formelsammlung Naturwissenschaften*. 2013.



bei Lageänderungen oder Funkkontakt eingeschaltet. Meist war von den beiden redundanten Prozessoren nur einer aktiv, konnte aber dennoch auch auf den Speicher des jeweils inaktiven Prozessors zugreifen. Hierauf soll allerdings erst im Kapitel "Verteiltes Rechnen" genauer eingegangen werden.


Als Massenspeicher diente ein 328 Meter langes Magnetband, welches 536MBit umfasste. Dieser wurde auch "Data Storage Subsystem" genannt. Der jeweils 4K Wörter umfassende Speicher der Subsysteme hätte noch nicht mal ein Foto aufnehmen können. Somit wurden Fotos und andere Messdaten bei fehlender Funkverbindung auf dem Band zwischengespeichert. Bestand eine Funkverbindung, wurden die Daten allerdings meist direkt gesendet, obwohl das Magnetband eine knapp 12 mal höhere Datenrate (schreibend 115KBits/s) als die Funkverbindung bieten konnte.^{25,26}

3.1.1 Der CCS Computer

Auf dem  sieht man ein Blockdiagramm des CCS der Viking Sonde. Bis auf wenige Veränderungen war das CCS-System der Voyager Sonden identisch mit diesem. Aus dem Diagramm kann man gut die Redundanz des Systems erkennen. Jedes Element kommt zweimal vor und ist mit dem Backup-System verbunden.

Der Prozessor arbeitete seriell 18-Bit Wörter ab und unterstützte 64 Instruktionen. Diese Limitierung lag an der Interpretation des 18-Bit Wortes. Die 6 letzten signifikanten Bits kodierten die Befehle, die 12 signifikantesten Bits die Adressen. Die somit 4K adressierbaren Speicherzellen und die 64 Befehle wurden von der NASA komplett ausgeschöpft. Dabei wurden zum Speichern von Ganzzahlen das Zweierkomplement verwendet, sodass Zahlen im Bereich -131,072 bis 131,071 dargestellt werden konnten. Der 4K Wörter fassende Speicher war in 4, mit jeweils 1K Wörter gleich große Abschnitte eingeteilt. Bei den ersten 3 Blöcke konnten die Zugriffsrechte individuell mit "read only", "write protected" und "read/write" versehen werden. Nur der 4. Abschnitt war stets auf read/write gesetzt.


 Taktzyklus dauerte 88µs. Im Prozessor waren 13 Register implementiert, wie ein 18-Bit Akkumulator, ein 12-Bit Programm Counter und ein  Link Register. Zudem gab es ein 4-Bit Status Register, um overflow, minus, parity und nonzero als Flag zu implementieren. Es gab einen Timer, der alle 60 Minuten, jede Sekunde und alle 10ms ein Signal an den Interrupt Prozessor abgab. Nachdem die Pulse von dem Interrupt Prozessor geprüft wurden, leitete er sie an den Hauptprozessor weiter. Diesem standen 32 Interrupt Level zur Verfügung, welche nach dem "Highest priority first" Prinzip abgearbeitet wurden. Die gleiche Technik der Interrupts wurde auch in den Space-Shuttles und den Apollo Missionen verwendet, dort allerdings in Software implementiert, statt in Hardware.

Die wichtigste Änderung des CCS hinsichtlich  Voyager Sonde waren die neuen Ein-/Ausgabe Schnittstellen zum Flight Data System und dem bauähnlichem Attitude and Control System. Um der Funktion als Kommandocomputer gerecht zu werden, wurden alle Kommandoänderungen und Speicherzugriffe durch das CCS geleitet. Außerdem




musste der "Herzschlag" der anderen Systeme überwacht werden. Dazu wurden die 2 neuen Funktionen, MEM-LOAD und AAC SIN, implementiert. Dass eine so komplexe Sonde wie die 2 Voyagers, mit einem Computer der gleichen Speichergröße gesteuert werden konnte, lag an dem wiederbeschreibbaren Programmspeicher. So wurden bei Voyager 1 allein bei der Ankunft beim Jupiter 18 Programmänderungen von der Bodenstation empfangen.

28,29,30

3.1.2 Das Lagekontrollsystem

Das Attitude Articulation and Control System steuerte die 3-Achsen Ausrichtung der Sonde an. In diesem System  en aber nicht nur die dafür benötigten Berechnungen angefertigt, sondern auch die Ausrichtung der Kamera und weiterer Sensoren übernommen. Die Architektur dieses Systems hatte sich bereits innerhalb der letzten Missionen stark verändert. Noch in den ersten Mariner Sonden wurden fest verdrahtete analoge Computer benutzt. Erst im Verlaufe des Programms wurden diese durch digitale Schaltkreise ersetzt. Auch im Vorbild Voyagers, der Viking Sonde, war dieses System digital. Das Lagekontrollsystem der Voyager nutzte einen hybriden Ansatz, indem sowohl ein digitaler, programmierbarer Computer, als auch analoge Schaltkreise verwendet wurden. Der Prozessor war letztendlich eine modifizierte Version des CCS, welcher von der Viking Sonde übernommen wurde.

Dieses als HYPACE (Hybrid Programmable Attitude Control Electronics) bezeichnete System bestand aus einem auf ca. 36kHz getakteten Prozessor (28µs pro Zyklus).

 diese Geschwindigkeit zu erreichen wurden TTL Schaltkreise verwendet. Anbei standen dem Prozessor 4K Hauptspeicher, welcher mit einer Breite von 18-Bit angebunden  Die TTL-Bausteine waren auf 4-Bit Operationen angelegt, sodass gegenüber einer bit-seriellen Architektur  taktzyklen zum Laden bzw. Speichern eines 18-Bit Wortes gespart werden konnten.

Die unterschiedlichen Aufgaben arbeitete der HYPACE mittels 4 verschiedener Taktzyklen ab. Aktive Komponenten wie wie Motoren oder Antriebe wurden in 10ms Intervallen angesprochen. Die Berechnungen zur Ansteuerung der Schubdüsen wurden alle 20ms abgewickelt. Grundlegende Arbeiten, wie das Interpretieren von Kommandos oder Timer wurden alle 240ms angestoßen. Da das Ansprechen der Steuerröhren sehr zeitkritisch war, wurden diese Routinen mittels Interrupt ausgeführt. Dies konnte dazu führen, dass unter Umständen die weniger wichtigen Berechnungen bis zu 110ms später als zugeteilt ausgeführt wurden. Das verteilte Rechnen auf der Voyager-Sonde benötigte Schnittstellen zwischen den einzelnen Komponenten, um Daten und Befehle auszutauschen. Alle Computer der Sonde mussten in festen Abständen (2s) einen sogenannten "Herzschlag" senden, um die Betriebsbereitschaft dieser Komponente mitzuteilen. Erfolgte 10 Sekunden lang kein Herzschlag, wurde auf den Backupprozessor umgeschaltet.

²⁸Vgl. [13] Leitenberger. *Voyagers: Die Sonde*. 2017.

²⁹Vgl. [30] unbekannt. *Distributed Computing On Board Voyager and Galileo*.

³⁰Vgl. [25] Unbekannt. *From Sequencers to Computers: Exploring the Moon and the Inner Planets*.

²⁵Vgl. [30] unbekannt. *Distributed Computing On Board Voyager and Galileo*.

²⁶Vgl. [13] Leitenberger. *Voyagers: Die Sonde*. 2017.

²⁷27.

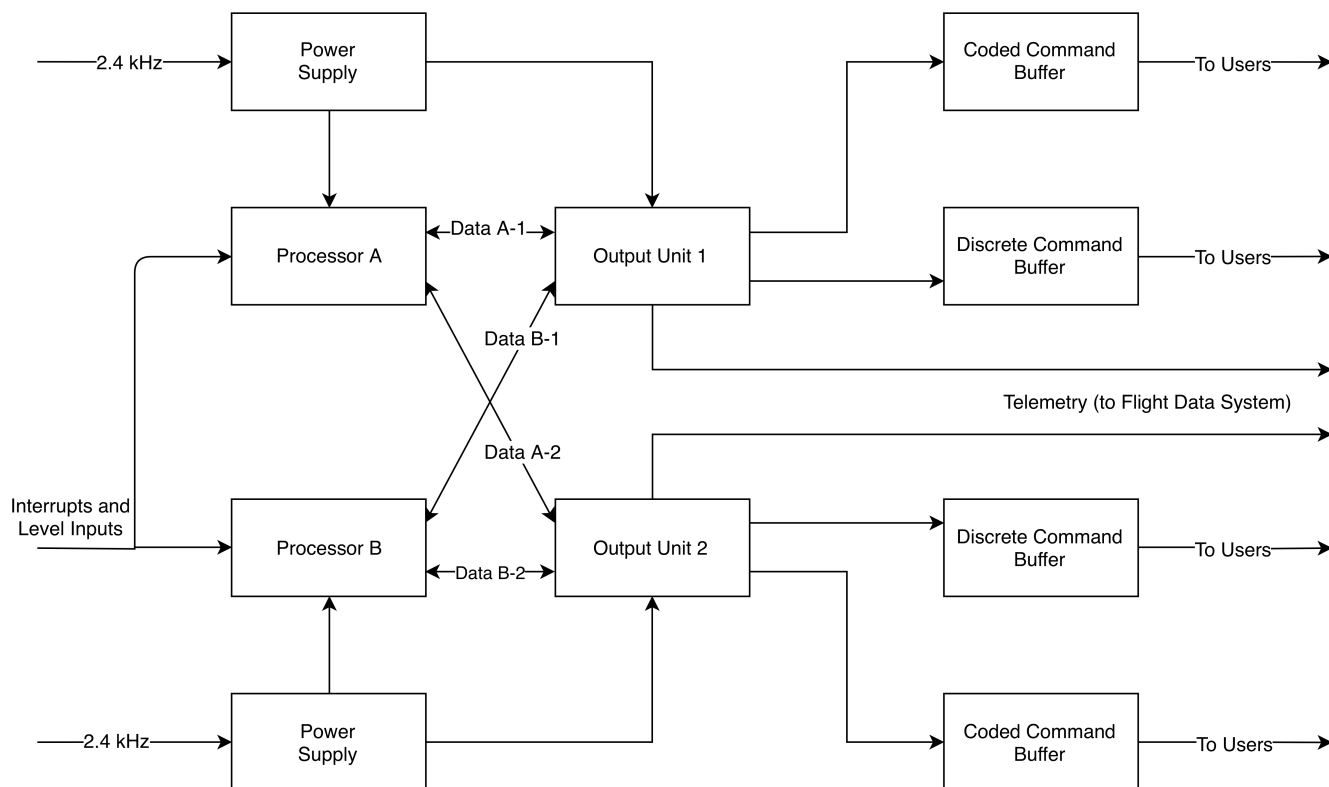


Figure 3: Übersicht des CCS der Viking Sonde ²⁷

3.1.3 Der Flight Data System Computer

Der Flight Data System Computer war für das Verarbeiten und Speichern der anfallenden wissenschaftlichen Daten zuständig. Wurden Daten von der Bodenstation an die Sonde geschickt, leitete der Computer die Daten an ein Magnetband weiter, welches diese dann aufzeichnete. Aufgrund eines Zuwachses an Messinstrumenten und Messgenauigkeit waren (für damalige Verhältnisse) hohe Datenraten erforderlich. Dieser Umstand führte dazu, dass beim CCS von einem festverdrahtetem System auf einen digitalen Ansatz gewechselt wurde.

Bereits die Vikinger Sonden verfolgten einen solchen Ansatz. Auf einem digitalen Speicher wurden kleinere Subroutinen gespeichert, welche die Verarbeitung der Daten unter unterschiedlicher Datenraten steuerte. Bei der Voyager Sonde bewilligte die NASA jedoch eine komplette Neuentwicklung des Computers. Bei der Entwicklung durchlief der Computer mehrere Phasen mit Hardwareänderungen. Als die Entwicklung begann, standen dem Prozessor 4K Speicher mit 18-Bit Worten zur Verfügung. Die Bauart war ähnlich dem des Magnetkernspeichers, konnte aber maschinell hergestellt werden, was ihn günstiger machte. Zudem wurden 120 der gleichen TTL Bausteine wie im CCS System verbaut. Die Ausführungszeit eines Befehls dauerte zwischen $12\mu s$ und $24\mu s$.

Da der Prozessor zu langsam war, wurde als erste Modifikation ein direkter Speicherzugriff realisiert. Daten konnten direkt an den Speicher geleitet werden, ohne den Hauptprozessor durchlaufen zu müssen. Die gängige Methode, das zu realisieren, wäre einen Interrupt zu benutzen. Bei dieser Lösung würde durch den Interrupt den aktuell laufenden

Instruktionen ein als "geklaut" werden. Aufgrund der hohen Datenraten wäre das allerdings sogar eine Verschlechterung gewesen, da in zeitkritischen Situationen die Ausführung von Befehlen gewährleistet gewesen Die NASA löste dies, indem jede Instruktion, auch welche ohne Speicherzugriff, einen Zyklus mehr brauchten, um so eine exakte Laufzeitberechnung zu ermöglichen.

Bei der zweiten Modifikation wurden zum ersten Mal in der Raumfahrt CMOS ICs verwendet. Im Vergleich zum Magnetkernspeicher war er schneller und hatte weitere Vorteile wie eine hohe Spannungstoleranz und einen geringen Energieverbrauch. Trotz der guten Eigenschaften war damals der Einsatz des Speichers nicht unumstritten. Zum einen war die Technologie noch relativ neu, zum anderen war der Speicher flüchtig. Hätte es in kritischen Situationen einen Stromausfall gegeben, wäre der Speicherinhalt verloren gewesen.

Die fertige Version des Computers unterschied sich stark von anderen programmierbaren Computern. Es gab mit 128 Registern deutlich mehr als in anderen Prozessoren. Die hohe Anzahl wurde dadurch erreicht, dass die Register im Speicher realisiert wurden. Aufgrund des speziellen Einsatzes zum Verarbeiten von Daten waren arithmetische Operationen weniger wichtig und deren Performance wurde vernachlässigt. Dafür wurde ein spezielles Augenmerk auf die Shift- und I/O-Operationen gelegt. Neben den üblichen Operationen "ADD", "SUB", "AND" und "XOR" gab es noch viele In- und Dekrement Operationen. Wie auch die anderen beiden Subsysteme war auch der Flight Data Computer in doppelter Ausführung vorhanden. Allerdings

konnte dieser nicht im Notfall selbstständig umschalten, sondern musste manuell von der Bodenstation aktiviert werden. Letztendlich hatte der Computer einen Datenfluss von bis zu 115K Bits/s, was für damalige Flight Data Systeme einen Meilenstein setzte.

3.2 Rechnertechnik der Galileo Raumsonde

Bei ersten Planungen des Galileo Projekts sollte noch der National Standard Spacecraft Computer (NSSC-1) aus vorangehenden Missionen wie den Voyager Sonden, als Komponente für alle Hauptsysteme benutzt werden. Allerdings wurde sich in allen Fällen, aus Kostengründen und der fehlenden Möglichkeit Fließkomma-Operationen zu berechnen, dagegen entschieden. Stattdessen entschied man sich für ein System aus mehreren Mikroprozessoren ähnlich dem Unified Data System (UDS) zu konstruieren. Das Konzept des UDS aus den 1970er war, mehrere Computer über ein Bussystem zu einem flexiblen, zuverlässigen System zu verbinden.³²

Eine große Herausforderung bei der Konstruktion des Computersystems war die Unterteilung der Raumsonde in zwei Bereiche. Die Supn section und die Despun section. Diese Einteilung war nötig, da der Orbiter um seine eigene Achse rotieren sollte. Um also gute Bilder machen zu können, wurde ein Teil der Sonde, die Despun section gegen die Rotationsachse gedreht. Dadurch war diese Sektion zwar unbewegt im Vergleich zum Weltraum, aber die Steuerung der Systemkomponenten auf dieser verkomplizierte sich. Für die Datenverbindung des statischen und des rotierenden Bereichs wurde ein Rotationstransformer genutzt, sodass eine Übertragung zwischen den beiden Bereichen bis zu 400ms gedauert hätte.³³

Letztendlich bestand das Computersystem auf dem Galileo Orbiter aus einem ganzen Netzwerk von insgesamt 19 Mikroprozessoren mit zusammen 320KB Random Access Memory (RAM) und 41KB nur lesbaren Hauptspeichers (ROM). Aufgeteilt bildeten die Mikroprozessoren 10 Subsysteme, von denen das Command and Data Subsystem (CDS) und das Attitude and Articulation Control Computer System (AACS) die beiden größten und wichtigsten waren, da sie die zentralen Funktionen des Galileo Orbiters steuerten. Siehe Figure 4

Die Hauptaufgaben des CDS waren die Verwaltung des Bussystems und der Fehlerschutz (Fault protection) der meisten Subsysteme. Außerdem war das CDS für die Kommunikation mit der Erde, also das Dekodieren von empfangenen Kommandos und deren Weiterleitung an das entsprechende Subsystem und das Kodieren der Telemetriedaten der verschiedenen Subsysteme, um diese an die Erde zu senden, zuständig. Zusätzlich zählte ein Magnetbandspeicher zum CDS. Auf diesem Data Memory Subsystem (DMS) konnten mit $9 \cdot 10^8$ bits größere Datenmengen der Experimente gespeichert werden, bis sie zur Erde gesendet werden konnten.

³¹2.

³²Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space.*

³³Vgl. [7] Hamilton. *Galileo Project.*

Die Aufgaben des anderen wichtigen Systems, des AACS, bestanden vor allem in Positionsbestimmung, Triebwerksteuerung und der Steuerung der Rotationsplattform des Galileo Orbiters. Das System wurde aber auch benutzt, um die Raumsonde auszurichten und Schwanken zu erkennen und verhindern. Das AACS war das einzige System an Bord der Raumsonde, das einen anderen Typ Mikroprozessor benutzte, um die bestmögliche Reaktionszeit sicherzustellen. Auch acht der neun Experimente an Bord des Orbiters hatten ihren eigenen Prozessor. Verbunden waren diese Systeme über 806kHz Datenbusse. Bis auf die Prozessoren des AACS waren alle eingesetzten Mikroprozessoren des Typen CDP1802 der RCA Corporation. Jeder dieser Prozessoren hatte seinen eigenen RAM. Die Mikroprozessoren der einzelnen Experimente waren dabei speziell für ihre Aufgabe und mit Ausnahme eines Systems in Assembler programmiert. Der Spezialfall unter den Experimenten war der Prozessor des Magnetometers mit jeweils 2KB RAM und ROM. Dieser war in Forth programmiert.³⁴³⁵³⁶

3.2.1 Das Command and Data Subsystem (CDS)

Der Kern des CDS bestand aus sechs der CDP1802 Mikroprozessoren der RCA Corporation und insgesamt 176.000 Byte-Wörtern RAM.

Bei den eingesetzten CDP1802 Prozessoren handelte es sich um 8-bit Halbleiter-Prozessoren, die auf etwa 1,6MHz getaktet waren. Diese hatten den Vorteil eines geringen Stromverbrauchs von ca. 30mW und konnten auch bei schwankenden Spannungen betrieben werden. Außerdem war es möglich die Chips durch eine Silizium-auf-Saphir Bauweise robuster gegenüber der starken Strahlung des Jupiters zu machen. Der große Nachteil der CDP1802 Chips war allerdings ihre Langsamkeit, die mit einer Taktzeit von 5µs und durchschnittlich zwei Takten zum Durchführen der meisten Instruktionen große Probleme in der Softwareentwicklung machte. Grund für diese Einschränkung war die sehr umständliche Unterstützung für 16Bit Operationen, da, trotz der 16 16Bit Universalregister, für Speicherzugriffe und somit für die meisten Operationen der nur 8Bit große Akkumulator genutzt werden musste. Dadurch wurden viele Operationen in mehrere Stücke aufgeteilt und so mehrere Takte benötigt. Die 16Bit Register konnten auch als 32 8Bit Register angesprochen werden und deren Funktion z.B. als Programmzähler war frei wählbar. Jeder Befehl des CDP1802 war 8Bit lang, davon waren die ersten 4Bit der Opcode und die zweiten 4Bit gaben das Register an. Folglich konnten nur 16 Instruktionen unterschieden werden. Außerdem gab es keine direkte Unterstützung Subroutinen über CALL und RET, **keine bedingten Sprünge** oder ein Stack. Auch wenn der CDP1802 keine native Unterstützung von Subroutinen hatte, funktionierte die Virtual Machine Language, in der ein Großteil der CDS Software geschrieben war und welche eigens für das Galileo Projekt entwickelt worden war, auf eine ähnliche

³⁴Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space.*

³⁵Vgl. [10] Leitenberger. *Der Galileo Orbiter.* 2017.

³⁶Vgl. [2] Daniel P. Siewiorek. *Reliable Computer Systems Design and Evaluation.* 1992.

³⁷29.

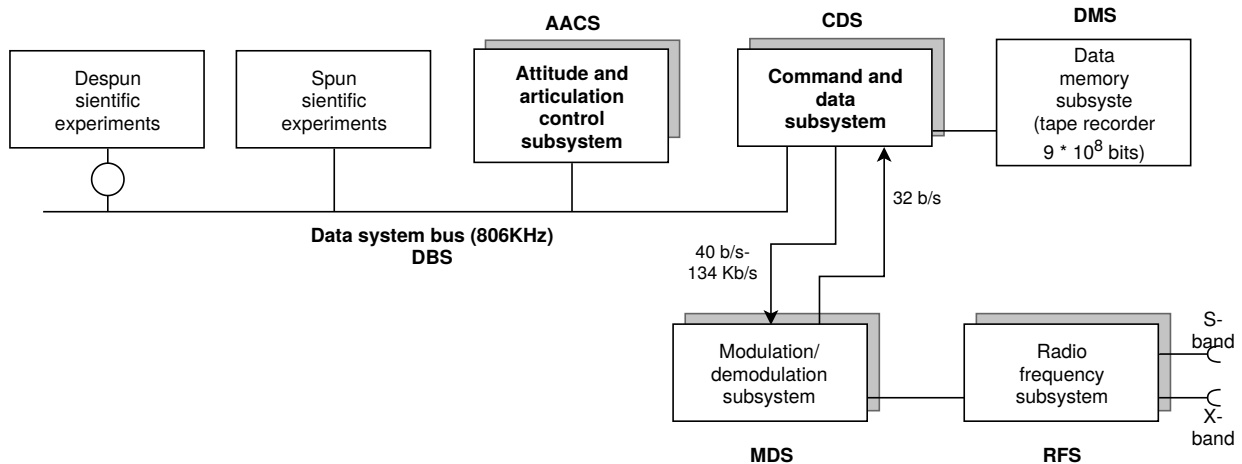


Figure 4: Galileo Orbiter System Block Diagram ³¹

Weise. In dieser Sprache waren verschiedene, einfache Statements wie IF, ELSE, DO und ASSIGN definiert, die vorgefertigten Blocks an Assemblerbefehlen ähnlich einer Subroutine aufrufen. Auf ähnliche Weise konnte auch ein Stack emuliert werden.³⁸³⁹⁴⁰⁴¹⁴²

Wie im Vorbild UDS wurden hier zwei Level von Mikroprozessoren unterschieden, die High-Level Modules (HLM) mit 32KB dediziertem RAM und die Low-Level Modules (LLM) mit 16KB dediziertem RAM. Von den sechs Prozessoren waren zwei HLMs und vier LLMs. Allerdings war ein siebter High Level Prozessor unabhängig von den restlichen Prozessoren als Reserve vorhanden.

Als wichtigstes System der Raumsonde war das Command and Data Subsystem aktiv redundant vorhanden. Dies bedeutete, dass zwei sogenannte Strings parallel durchgehend angeschaltet waren. Während Missionskritischer Abschnitte wurden sogar alle Aufgaben gleichzeitig von beiden Strings ausgeführt. Siehe Figure 5 Beide Strings waren dabei gleich aufgebaut. Jedem String war ein HLM mit dazugehörigem Bus-Controller und Datenbus zugeteilt. Außerdem gab es zwei LLMs, ein Massenspeicher (BUM) und ein Data Management Massenspeicher (DBUM) pro String. Weiterhin gibt es für jeden String einen Golay-Decoder (GC) und zwei Critical Controller (CRC). Außerdem gab es zwei redundante Hardware Command

Decoder (HCD), die an das Modulation/Demodulation System (MDS) angeschlossen waren. Diese leiteten die von der Erde kommenden, dekodierten Kommandos direkt an beide HLMs und alle Critical Controller weiter.

Die High Level Modules waren zur Verwaltung des Systems gedacht und waren so dafür zuständig, die von der Erde kommenden Kommandos auszuführen, die Raumsonden-Clock für den Basistakt zu erhalten, Daten über den Systembus zu bewegen, gespeicherte Programme auszuführen, Telemetrie zu steuern und Fehlerüberwachung und Recovery zu betreiben. Von den 32KB RAM der HLMs waren jeweils 12KB nur lesbar (ROM) eingestellt.

Die Low Level Modules bildeten sozusagen den Input/Output zwischen CDS und den restlichen Systemen. Sie sammelten und formatierten Hardwaredaten und Experimentdaten der Subsysteme und speicherten diese im Massenspeicher, der auch als Puffer vor der Übertragung an die Bodenstation genutzt wurde. Weiterhin konnten die LLMs kodierte, **komplexe Befehle** an die Nutzer ausgeben und Statusangaben innerhalb des Toleranzbereiches erkennen. Auch manche fehlerüberwachende Funktionen wurden von den LLMs übernommen.

Die Massenspeicher (BUM) bestehen aus jeweils 16KB RAM und die Data Management Massenspeicher (DBUM) aus 8KB RAM. Beide werden vor allem als Pufferspeicher für Telemetriedaten und Daten, die über den Bus transportiert werden müssen, benutzt. Außerdem stellen die DBUMs die Verbindung zum Data Management System dar, der Magnetbandspeicher, auf dem die Experimentdaten auf längere Zeit gespeichert wurden. Die Hauptaufgabe der Critical Controller bestand darin, kritische Events, wie das Ausklinken der Atmosphärensonde, zu initialisieren. Aber auch wichtige, schreibgeschützte Speicherbereiche werden von den CRCs überwacht. Mit den Golay Codern gab es eine Hardwaremöglichkeit, um die Experimentdaten mit einem Golay (24,12) fehlerkorrigierenden Code zu kodieren.

³⁸Vgl. [3] Deese. *APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS*.

³⁹Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁴⁰Vgl. [9] Leitenberger. *Computer in der Raumfahrt Teil 2*. 2001.

⁴¹Vgl. [10] Leitenberger. *Der Galileo Orbiter*. 2017.

⁴²Vgl. [32] unbekannt. *Galileo Engineering*. 2006.

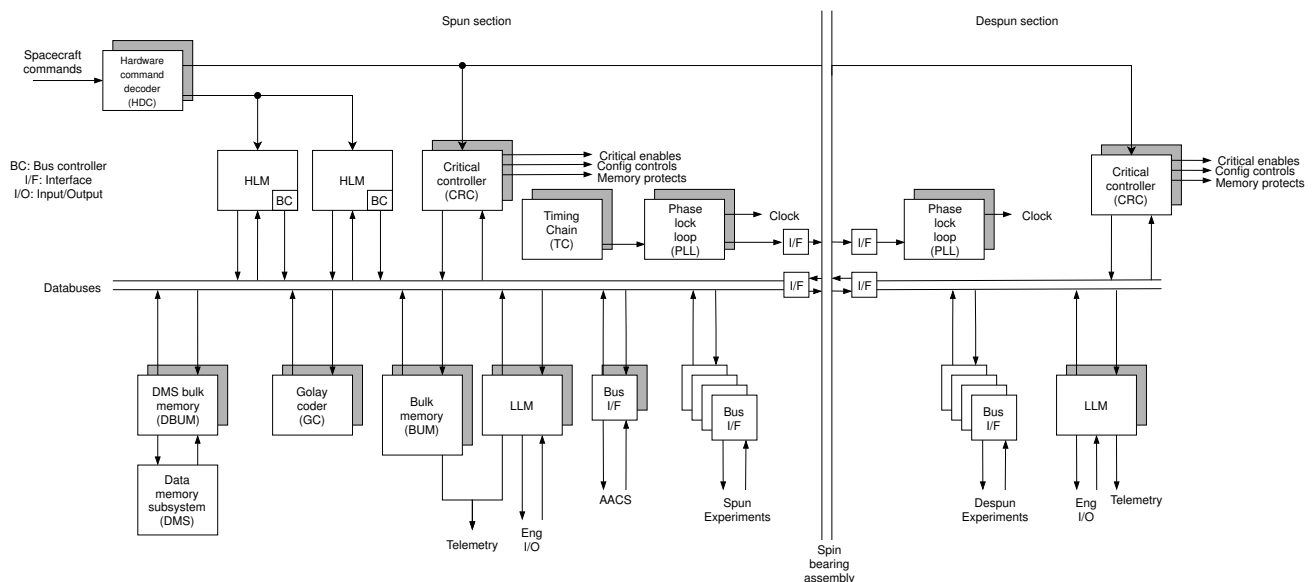


Figure 5: Galileo CDS simplified Block Diagram ³⁷

So wurde bei allen Daten außer Fotos sichergestellt, dass möglichst keine Übertragungsfehler entstanden. Ein weiterer Teil des CDS waren der Timing Chain (TC) und der Phase lock Loop (PLL), welche die Echtzeituhr (Clock) des Orbiters betrieben und so 15-mal pro Sekunde ein Interrupt an alle Systeme der Galileo Raumsonde sendeten. An diesem „Herzschlag“ orientierte sich also der vordergründige Takt aller Systeme.

Aber nicht nur hardwaretechnisch, sondern auch räumlich war das CDS aufgeteilt. So befand sich ein LLM und ein CRC pro String auf dem rotierenden Bereich des Galileo Orbiters, um auch diesen ohne größere Signalverzögerungen zu kontrollieren.⁴³⁴⁴⁴⁵

3.2.2 Das Attitude and Articulation Control Computer System (AACS)

Da das Attitude and Articulation Control Computer System (AACS) unter anderem auch Aufgaben wie die Steuerung und Kontrolle der Triebwerke und die Positionsbestimmung übernahm, welche viel schnellere Reaktionen benötigten als die meisten anderen Systeme, kam hier der langsame CDP1802 Mikroprozessor nicht in Frage. Stattdessen entschied man sich für den sogenannten Advanced Technology Airborne Computer (ATAC). Dieser Prozessor von Ittek bestand aus einem Verbund von 4 AM2901 Prozessoren, 4Bit Prozessoren, die auf 9,5MHz getaktet waren. Mit 123 Instruktionen konnten so alle Rechenoperationen einschließlich Fließkommarechnungen auf 4Bit Basis ausgeführt werden. Durch den Verbund von vier dieser Prozessoren konnte also ein 16Bit

Prozessor mit vierfacher Geschwindigkeit geschaffen werden. Dadurch erreichte der ATAC eine Taktzeit von 250ns. Da jedoch ein Speicherzyklus $2\mu s$ dauerte, war die reelle Geschwindigkeit auf 143.000 Instruktionen/sek reduziert. Neben der Geschwindigkeit war ein weiterer Vorteil des ATAC, dass durch Mikrocode eigene Instruktionen generiert werden konnten. Durch 4 neue Instruktionen war es so möglich über 1500 Wörter an Code zu sparen.⁴⁷⁴⁸⁴⁹⁵⁰

Auch die Hardware des AACS war auf den statischen Teil und die Rotierende Plattform der Galileo Sonde aufgeteilt. Auf dem festen Bereich bestand diese aus den Attitude Control Electronics (ACE), den Acquisition Sun Sensors (AS), dem Star Scanner (SS), den Propulsion Drive Electronics (PDE) und den linear Boom Actuators (LBA). Sensoren (AS) und Scanner (SS) wurden dabei zur Positionsbestimmung anhand der Sonne und anderer Sternpositionen benutzt. Außerdem konnten so ungewollte Schwankungen der Raumsonde erkannt werden, welche über den LBA gesteuerten, beweglichen Ausleger ausgeglichen werden konnten. Die PDE war die Elektronik, um die Triebwerke zu kontrollieren. Der zentrale Teil des AACS waren allerdings die ACE, welche die eigentlichen Funktionen des AACS steuerten und überwachten, die Sensordaten verarbeiteten, Kommandos ausführten und die Fehler-schutzfunktionen des AACS übernahmen. Aufgebaut waren die ACE aus Input/Output, einem der ATAC Prozessoren und 32K 16Bit Wörtern an Hauptspeicher (RAM) von dem 1K Wörter als nur lesbar (ROM) gekennzeichnet waren.

⁴⁷Vgl. [10] Leitenberger. *Der Galileo Orbiter*. 2017.

⁴⁸Vgl. [3] Deese. *APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS*.

⁴⁹Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁵⁰Vgl. [9] Leitenberger. *Computer in der Raumfahrt Teil 2*. 2001.

⁴³Vgl. [10] Leitenberger. *Der Galileo Orbiter*. 2017.

⁴⁴Vgl. [2] Daniel P. Siewiorek. *Reliable Computer Systems Design and Evaluation*. 1992.

⁴⁵Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁴⁶2.

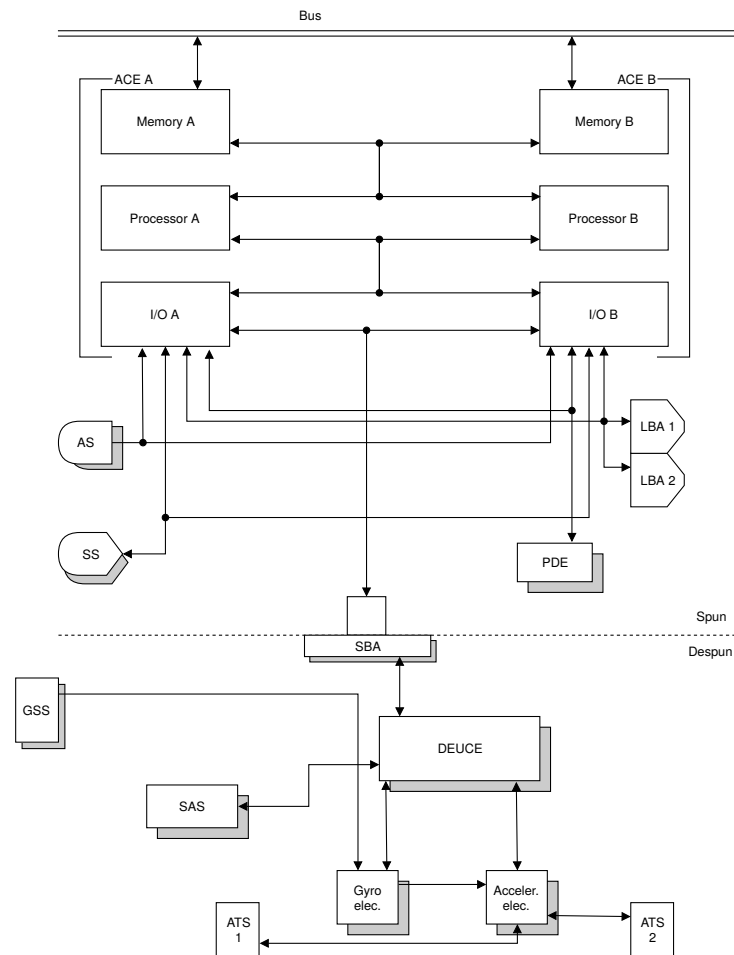


Figure 6: Galileo AACS simplified Block Diagram ⁴⁶

Auf der rotierenden Seite des Galileo Orbiters setzte sich das AACS aus den Despun Controls Electronics (DEUCE), dem Spin Bearing Assembly (SBA), Gyroskopen (GSS) und Beschleunigungssensoren (ATS) inklusive der direkt dazugehörigen Steuerungselektronik. GSS und ATS wurden dazu verwendet Ausrichtung und Geschwindigkeit der Raumsonde zu kontrollieren, während die Aufgabe der DEUCE darin bestand die Hardwaredaten der rotierenden Seite über die Transformatoren der SBA an die Attitude Control Electronics (ACE) weiterzuleiten. Siehe Figure 6 Auch die Komponenten des AACS waren alle redundant vorhanden. Im Gegensatz zum CDS war aber immer nur ein Teil aktiv, während der andere Teil, nicht mit Strom versorgt, sich im Standby befand. Einzige Ausnahme waren die Hauptspeicher der ACE-Blöcke. Diese waren dauerhaft mit Strom versorgt, um sie verlustfrei zu halten. Um das Attitude and Articulation Control Computer System (AACS) zu betreiben wurde das Galileo Real-time AACS Operating System (GRACOS) entwickelt, ein Echtzeitbetriebssystem, welches es ermöglichte auf dem AACS in HAL/S programmierte Programme

auszuführen.⁵¹⁵²⁵³⁵⁴

3.3 Gemeinsamkeiten und Unterschiede

Die Galileo Sonde startete 12 Jahre nach den Voyager-Sonden. Fast alle Mikroprozessoren waren schneller getaktet und hatten deutlich mehr Speicher als in der Voyager Mission. Zudem gab es mit 19 Mikroprozessoren in 10 Subsystemen deutlich mehr Unterteilungen als noch in den Voyager-Sonden mit nur 3 Subsystemen. Trotz der großen technologischen Unterschiede gab es auch viele Gemeinsamkeiten. Bei Galileo, wie auch bei den Voyager Sonden wurden statt einem zentralen Großrechner viele kleinere Computer verwendet. Hierzu wurden die Sonden in mehrere kleinere Subsysteme aufgeteilt, in denen dann speziell auf die jeweiligen Aufgaben angepasste Mikroprozessoren die Arbeit verrichteten. Die Ausfallsicherheit wurde wieder durch redundante Systeme sichergestellt. Auch die speziellen Subsys-

⁵¹Vgl. [10] Leitenberger. *Der Galileo Orbiter*. 2017.

⁵²Vgl. [2] Daniel P. Siewiorek. *Reliable Computer Systems Design and Evaluation*. 1992.

⁵³Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁵⁴Vgl. [32] unbekannt. *Galileo Engineering*. 2006.

teme wie ein Kommandosystem und ein Lagekontrollsystem hatten sich in den Voyager Sonden bewährt und hielten somit auch in der Galileo Sonde Einzug. Ebenfalls gab es einen Magnetbandspeicher. Die Galileo Sonde hatte zwar deutlich mehr Speicher, für eine große Anzahl an Fotos war dieser dennoch zu gering, sodass für eine Zwischenspeicher das Band verwendet wurde. Der einzige Prozessor der leistungsmäßig in die Liga Voyagers einzuordnen wäre, ist der Prozessor des Magnetometers. Mit jeweils 2KB an Rom und Ram lässt sich dieser mit den Rechnern der Voyager Raumsonden mit 4K 18-Bit Wort Speicher vergleichen. Ein großer Umstieg war der Wechsel der Speicherarchitektur von 18-Bit Wortbreite auf Vielfache von 8, so hatte die Galileo Sonde den Speicher in Byte-Blöcke unterteilt.

Durch den Wegfall der 18-Bit Worte unterstützten manche Prozessoren der Galileo Sonde sogar weniger Instruktionen. So interpretierte das "Command and Data Subsystem" nur 16-Bit Worte von denen nur 4-Bit für die Instruktion verwendet wurden. Verglichen mit den 6-Bit der Voyager-Sonden konnte diese 64 Instruktionen, statt nur 16 bei der Galileo Sonde, unterscheiden. Beim Lagekontrollsystem der Galileo-Sonde hingegen konnten in einem Zusammenschluss von 4 Computern über 123 Instruktionen genutzt werden.

Die Entwickler der NASA hatten bereits bei Voyager Erfahrung bei der Mikroprogrammierung gesammelt. Diese Expertise wurde nun bei der Voyager Sonde genutzt. Durch geschickte Neuentwicklung von wichtigen Instruktionen durch Mikrocode wurde die Codegröße um 1500 Wörter verringert.

3.4 Vergleich zur Viking Sonde

Wie bereits angesprochen, basierte ein großer Teil der der Voyager Sonden auf der vorangegangenen Viking Mission. Bereits die Viking Sonde war deutlich leistungsfähiger als **Mariner Missionen**. Dies lag daran, dass geplant war, die Viking Sonde auf dem Mars landen zu lassen. Da die Voyager Sonden aufgrund der vielen wissenschaftlichen Geräten und der komplexen Navigation durch mehrere Swing-by Manöver ebenfalls viel Leistung benötigte, entschied man sich, die Computer der Viking-Sonde zu übernehmen.

Auch die Viking-Sonde hatte schon redundante digitale Rechner verbaut. Damals noch ein absolutes Novum, Computer wurden bis dahin nur bei der bemannten Raumfahrt benutzt. **Falls** die Backup-Computer waren neu, selbst die Apollo-Missionen kamen ohne Backup aus. Da die Viking-Sonde auf dem Mars landen sollte, war sie in 2 Stufen aufgeteilt, den Lander und den Orbiter. Der Orbiter besaß das CSS, nahezu identisch dem der Voyager Sonde. Der Lander wurde mit dem GCSC (Guidance, Control, and Sequencing Computer) gesteuert. Es war das erste mal, dass das JPL-Team der NASA einen autonomen Flugcomputer programmieren musste. Dadurch, dass die Viking-Sonde, wie auch die Voyager-Sonden aus verteilten Rechnern bestand, wuchs das Softwareprojekt immer mehr. In kurzer Zeit wurde die NASA eine der führenden Softwareherstellern weltweit, auf Augenhöhe mit militärischen Konzerne wie Boeing Military Airplane Company und der TRW Corporation. Das JPL lernte recht schnell professionelle Softwareentwicklungstechniken anzuwenden. Gute Dokumentation und ausführliche Tests, sind auch noch heute nicht veraltet. Die hier gesammelte Erfahrung kam den Nachfolgeprojekten, insbesondere dem direkten Nachfolger Voyager zugute.

Ebenfalls neu, war die Herangehensweise zuerst die Software-Architektur zu erstellen und anschließend passende Hardware zu finden ("software first"). Bei dieser Art der Entwicklung, konnte für das Programm die passende Hardware ausgesucht werden und **Ärger** vermieden werden. Sonst wurde zuerst die Hardware entwickelt und die Programmierer mussten sich mit dieser arrangieren und die Software so schreiben, dass sie auf dieser lief. Das endete häufig darin, dass viel zu wenig Speicher zur Verfügung stand und die Programme lange optimiert werden mussten. Auch das andere Extrema, dass viele Ressourcen ungenutzt blieben, war vertreten. Selbst dann verzögerte sich die Entwicklung, da zusätzliche Features implementiert wurden, um die Hardware voll auszureizen. Dies verzögerte nicht nur die Projekte merklich, sondern sprengte häufig auch den Budgetrahmen.

Weitere Entwicklungen, welche die Nachfolger der Viking-Sonde prägten, war die Erstellung zusätzlicher Instruktionen durch eigene Mikroprogrammierung. So konnten spezielle Befehle selbst auf **belieblicher Hardware** implementiert werden. Die Mikroprogrammierung wurde unter anderem besonders bei der Galileo-Sonde genutzt.

Der Aufbau des CCS ist nahezu identisch zu dem der Voyager Sonde. Voyager hatte nur ein paar zusätzliche Funktionen eingebaut, um die zusätzlichen Instrumente ansprechen zu können.

Das GCSC sollte zunächst aus zwei einzelnen Subsystemen bestehen. Ein Rechner sollte die Landung, der andere die Berechnungen im gelandeten Zustand übernehmen. Aus zeitlichen Gründen entschied man sich aber nur einen Computer, dafür redundant, zu verwenden. Das GCSC bestand aus zwei Honeywell HDC 402 Prozessoren, mit jeweils 18000 Wörter umfassenden Ringkernspeicher. Die Prozessoren hatten eine Wortbreite von 24-Bit und unterstützten 47 Instruktionen. Verglichen mit den in den Voyager Sonden eingesetzten Prozessoren, hatte dieser nicht nur knapp das dreifache an Speicher, sondern war auch deutlich schneller. Es gab 8 Interrupt-Level und softwareseitig **ein Echtzeitsystem simuliert**. Jeder Prozess dachte, er könnte alle Ressourcen exklusiv nutzen. Dieses wurde allerdings nur während der Landung benutzt und war durch mehrere Routinen, welche in 20ms Intervallen ausgeführt wurden implementiert. Die Idee einer "virtuellen Maschine" wurde bei der Galileo-Sonde wieder aufgenommen. Die Entwicklung war noch weiter vorangeschritten, sodass für die Virtualisierung eigens mehrere Mikroprozessoren, wie in heutiger Computertechnik, zuständig waren.

Abschließend lässt sich sagen, dass die Nachfolger Galileo und besonders Voyager stark durch die Viking-Mission geprägt wurden. Ob die Voyager Mission wie wir sie heute kennen ohne die Viking-Mission ein solcher Erfolg geworden wäre, ist aufgrund der starken Ähnlichkeit beider Sonden fraglich.⁵⁵

4. VERTEILTES RECHNEN IN VOYAGER UND GALILEO

4.1 Einführung in verteiltes Rechnen

⁵⁵Vgl. [12] Leitenberger. *Viking Sonden*.

Allgemein lässt sich ein verteiltes System beschreiben als ein System, in dem mehrere Computer zu einem Netzwerk verbunden werden, um kombiniert an einem Ziel zu arbeiten. Dabei können einzelne, gleiche Komponenten oder Komponenten gleicher Art mehrfach vorhanden sein. Die Konfiguration der Einzelteile ist dabei nicht entscheidend, es kann sowohl eine Kombination aus Server und Mikroprozessor sein als auch zwei Workstations und so weiter. Außerdem können Software- und Hardwarekomponenten verteilt sein. Weiterhin ist es nicht wichtig, ob alle Teile des Systems physisch nah beieinander sind oder über mehrere Orte verteilt sind. Die Verbindung einzelner Teile lokaler Systeme findet meist über Bussysteme statt, genauso können aber weit verteilte Systeme auch drahtlos und zum Beispiel über das Internet verbunden werden.

Gegenüber klassischen zentralen Systemen bieten verteilte Systeme vor allem Vorteile wie Redundanz, bei der Systemkomponenten in gleicher Ausführung vorhanden sind und so eine große Ausfallsicherheit erreicht werden kann. Ein zweiter großer Vorteil besteht in der einfachen Erweiterbarkeit, da bei verteilten Systemen ohne großen Aufwand weitere Komponenten hinzugefügt werden können, um die Leistung des Netzwerks zu erhöhen. Auf diese Weise kann kostengünstig aus vielen, kleinen Komponenten ein Leistungstarkes System gebildet werden.⁵⁶

4.2 Verteiltes Rechnen in Voyager

Zunächst verfolgte die NASA noch den Ansatz einen großen Zentralrechner zu benutzen. In der damaligen Zeit kamen damit, sofern man Gewicht und Stromverbrauch beachtete, IBMs 4Pi Serie sowie deren Saturn Launch Vehicle Computer in Betracht. Gegenüber einem verteilten System ließen sich bei einem Zentralrechner Gewichts- und Stromverbrauchteinsparungen erzielen. Eine Studie des Elektronikkonzerns General Electric befürwortete letztendlich jedoch verteilte Computer, da in der Raumfahrt Zuverlässigkeit deutliche Priorität hat und die Einsparungen somit vernachlässigbar seien. Während der Entwicklung gab es seitens der NASA Ingenieure verschiedene Ansätze verteiltes Rechnen auf der Voyager zu realisieren.⁵⁷

4.2.1 Motivation für verteiltes Rechnen




Ein Ansatz sah den Einsatz der bereits in den Viking Missionen erprobten Computer vor. Durch den Einsatz auf mehreren Sonden ließe sich die Arbeitseffizienz deutlich steigern, da Tools wie Assembler oder Compiler der vorherigen Missionen genutzt und nicht neu entwickelt werden mussten. Ein solcher Rechner hätte für die umfangreicheren Aufgaben der Voyager Sonde nicht ausgereicht. Dieses Problem sollte dadurch behoben werden, dass mehrere Computer, für jedes System einer, eingesetzt werden sollten. Ein weiterer Einwand beschrieb den Kampf der Entwickler um Ressourcen auf dem Rechner. Die Voyager Sonde hatte durch die neuen Messinstrumente etwa 10 mal mehr zu berechnen als ihr Vorgänger, die Viking Sonde. Bei einem Zentralrechner hätten sich die Entwickler oft absprechen müssen und in kritischen Situationen gegebenenfalls Kompromisse eingehen müssen. Durch den Einsatz mehrerer Rechner hatte jedes größere Team einen eigenen Computer

und die Hoheit über dessen Leistung.

Die Gründe, die letztlich dazu führten, dass die NASA auf das verteilte System setzte, lassen sich kurz zusammenfassen. Mussten mehrere zeitkritische Routinen berechnet werden, hätte es auf einem Zentralrechner Probleme geben können. Wurden stattdessen diese Routinen auf mehrere Subsysteme aufgeteilt, konnten diese echt parallel abgearbeitet werden. Waren die zu berechnenden Aufgaben zu unterschiedlich müsste der Zentralcomputer erweitert werden. Dies würde den Gewichts- und Größenvorteil wieder zunichte machen. Nach Plänen der NASA sollte es die Subsysteme "Guidance and Control", "Telemetry", "Command", "Data Automation Equipment", "Computer and Sequencer" geben. Jedes dieser Subsysteme benötigte spezielle Algorithmen. So unterschieden sich beispielsweise stark die Routinen der Datenverarbeitung mit denen der Sondensteuerung. Durch den Ansatz mehrere verteilte Systeme zu nutzen, konnte so jedes Subsystem einen jeweils auf sich abgestimmten Rechner erhalten. So wurde schon im Kapitel "Rechnertechnik der Voyager Sonden", der spezielle Befehlssatz des Prozessors im System Datenverarbeitung erwähnt. Zuletzt wäre durch den Ausfall des Zentralrechners die gesamte Mission gescheitert gewesen. Den Ausfall eines kleineren Submoduls hätte man hingegen kompensieren können.

Die Pläne der NASA konnten realisiert werden. Als die Entwicklung der Voyager Sonden abgeschlossen war, wurden für zwei der drei Subsysteme Computer der Viking Sonde verwendet.

4.2.2 Ablauf des verteilten Rechnens

Im Kapitel "Rechnerarchitektur Voyager" wurde bereits der Ansatz des verteilten Rechnens angerissen. Es gab 3 große Subsysteme, welche jeweils von zwei komplett redundanten Rechnern gesteuert wurden. Hierbei nahm das "Command and Control System" (CCS) die Rolle der Überwachung der anderen Systeme, da diese sich mit dem sogenannten "Herzschlag" alle 2 Sekunden melden mussten. So war davon auszugehen, dass diese korrekt funktionierten. Die meiste Zeit rechnete einer der Rechner isoliert, während der andere aus Sicherheitsgründen ausgeschaltet war. Die einzelnen Computer der Subsysteme waren über Schnittstellen mit den anderen Systemen vernetzt. Der CCS Computer koordinierte die Ausführung von Befehlen und Abläufe auch auf anderen Rechnern auf der Sonde. Bekam er neue Befehle über Funk von der Erde so wurden auch diese ausgeführt.  dem Diagramm 7 kann man diese Abläufe gut erkennen. Ebenso wurden wichtige Telemetriedaten über die Subsystemengrenzen ausgetauscht. Wissenschaftliche Daten wurden über das Flight Data System gesammelt. Besonders wichtige und prestige-trächtige Daten wie die ersten Fotos von Uranus und Neptun wurden in Echtzeit gesendet. Andere Daten wurden auf das Magnetband der Sonde gespeichert und später gesendet.  enders im Funkschatten hinter anderen Planeten war diese Funktion besonders wichtig, da der Speicher nicht ausgereicht hätte. Die zu sendenden Daten wurden vorher aufbereitet, hierzu gehörten unter anderem eine CRC Kodierung. Im Verlaufe Mission sendete die NASA ein Update, sodass Bilder auch komprimiert verschickt werden  ten. So konnten etwaige Übertragungsfehler korrigiert werden. Erst nach dieser Verarbeitung wurden die Daten mit einer Bandbreite von bis zu 115KBits/s an das CCS geleitet, welche diese schließlich zur Erde schickte.

⁵⁶Vgl. [38] unbekannt. *What is distributed computing.*

⁵⁷Vgl. [31] unbekannt. *Distributed computing on board Voyager and Galileo.*

Sollten Fotos von den Monden Saturns und Neptun geschossen werden, war neben dem CCS und dem Flight Data System auch das AACS beteiligt. Die hohen Geschwindigkeiten von bis zu 20km/s dieser Himmelskörper machten ein eigenes System zur Berechnung der Kameraausrichtung zwingend notwendig. So waren die anderen Systeme bereits gut ausgelastet, aber selbst als eigenes System wurden die Ressourcen des AACS voll ausgeschöpft. Bereits zum Start war der Speicher bis auf 2 Worte komplett belegt.⁵⁸

In Ausnahmefällen, etwa bei Begegnungen mit anderen Planeten, musste sehr viel berechnet werden und Speicher war knapp. In diesen Fällen wurden teilweise beide Computer des CCS gleichzeitig genutzt. In diesem sogenannten Tandem-Modus berechneten beide CPUs unabhängig voneinander das Gleiche, benutzten dafür aber den gemeinsamen Speicher. Somit konnte praktisch der Speicher verdoppelt werden. Das CCS unterstützte, neben dem normalen Betrieb (nur ein Rechner aktiv) und diesem Tandem Betrieb noch zwei weitere Modi. Im Parallel-Betrieb führten beide Prozessoren zur gleichen Zeit, die gleichen Instruktionen aus. Im Individual-Betrieb, rechneten beide Computer, aber an komplett unterschiedlichen Aufgaben.⁵⁹


4.2.3 Neue Funktionen und Updates

Noch weit bevor durch Smartphones der Begriff "Update over the air" populär wurde, beherrschte die NASA diese Technik. Das CCS und das Flight Data System konnten von der Erde aus mit Updates versorgt werden. Das Installieren des Updates gestaltete sich nicht sehr einfach. Auch heute ist der Prozess noch teilweise sehr kompliziert und es besteht die Gefahr, dass das Gerät unbrauchbar wird. Das war natürlich für eine Sonde keine Option und die NASA ließ sich einen alternativen Weg einfallen. Die ursprüngliche Software der Voyager Sonden war in Fortran 5 geschrieben. Mit einem späteren Update wurde Software, welche in Fortran 7 geschrieben war als Update gesendet. Dies lag daran, dass durch die andere Programmiersprache, die Programmgröße abnahm. Hinsichtlich der immer längeren Singallaufzeiten, wurde dies benötigt. Zuletzt portierte die NASA sogar Programme in C, um die Speichergröße weiter zu verringern.⁶¹

4.2.4 Sicherheit durch verteiltes Rechnen

Im Viking Programm wurde ein neu entwickelter Computer namens "Star" (self test and repair) erprobt. Mithilfe dieses Systems sollte eine Sonde 10 Jahre autonom gesteuert werden können. Ursprünglich wollte die NASA diesen auch in den Voyager Sonden verwenden, allerdings mussten sie aus Budgetgründen eine Alternative finden. Die Entwicklung wurde später teilweise in der Galileo Sonde eingesetzt. Stattdessen war die NASA der Auffassung, dass sich Fallsicherheit mit Hilfe eines verteilten Systems ebenfalls bewerkstelligen lasse. Hierzu wurden die einzelnen Systeme jeweils redundant ausgelegt und mit zusätzlichen Sicher-

heitsschaltungen versehen. Diese konnten Fehler im aktiven Computer erkennen und dann auf den inaktiven Rechner umschalten.

Insgesamt gab es 7 Sicherheitsroutinen, welche autonom Fehler erkennen konnten. Diese brauchten  nur Sekunden oder wenige Minuten. Das war insbesondere im Verlauf der Mission wichtig, als die Laufzeiten der Steuerkommandos von der Erde mehrere Stunden brauchten.^{62,63,64,65}

Davon besaß das CCS allein 5 Fehlerschutzalgorithmen (FPAs), welche kritische Fehler erkennen und mit entsprechenden Maßnahmen, wie dem Umschalten auf den Backup Prozessor reagieren konnten. Eine Routine überprüfte, ob innerhalb eines bestimmten Intervalls ein Kommando empfangen wurde. Blieb dieser aus, wurde auf das Backup System umgeschaltet. Eine andere Routine überprüfte das Sende-/Empfangssystem welche für das S- und X-Band zuständig war. Der "Power Check" überprüfte die Spannung. Lag diese zu niedrig oder gab es Probleme innerhalb eines bestimmten Schaltkreises, wurde dieser isoliert und auf die redundante Hardware umgeschaltet. Lag ein schwerwiegender Fehler im gesamten Computersystem des CCS vor, wurden alle ausgeführten Programme gestoppt und in einen Notfallmodus umgeschaltet. Anschließend wurde auf weitere Befehle der Bodenstation gewartet. Diese konnte dann ein Update aufspielen und hatte damit eine Chance, den Fehler softwareseitig zu beheben.

Konnte durch Umschalten auf die Backup-Systeme eine verlorengegangene Verbindung zur Bodenstation nicht wieder hergestellt werden, gab es einen besonderen Softwarezustand. Dieser wurde durch die Command Loss Routine durch den abgelaufenen Timer angesteuert. Blieb das Durchschalten auf andere Systeme viermal erfolglos, schaltete die Sonde in den sogenannten "Backup Mission Load" Modus. Diese Sequenz beschrieb die wichtigsten Funktionalitäten, die die Sonde weiterhin ausüben musste. Dazu gehörten das weitere selbstständige Aufzeichnen von wissenschaftlichen Daten auf das Magnetband. Weiterhin mussten Daten über die Flugbahn sowie Telemetrie gesammelt werden. In regelmäßigen Abständen wurden diese an die Erde geschickt. Dabei wurde für den Notfallmodus nicht die volle Funktionalität des Sendesystems genutzt. Für Basisdaten wie Telemetrie wurden nur 160Bit/s verwendet, bei wissenschaftlichen Daten bis zu 1400Bit/s. Diese Daten konnten dann auch nicht von allen Bodenstationen der NASA empfangen werden, sondern nur von solchen, die das 34m Band unterstützen.

Abschließend lässt sich sagen, dass sich das redundante System bewährt hat. Nach über 40 Jahren im Einsatz, sind von 12 Systemen der Voyager Sonde noch 11 funktionstüchtig. Nur ein Prozessor des CCS wurde durch einen Fehler groß beschädigt. Von den wissenschaftlichen Systemen funktionieren noch alle, die Hälfte davon ist aber, um Energie zu sparen, ausgeschaltet. Erst in den letzten Jahren waren beide Voyager Sonde nochmals in den Schlagzeilen, als sie das Verlassen des Sonnensystems dokumentierten. Heute sendet die NASA nur noch selten Kommandos an die Sonde. Wirklich steuern lässt sie sich aufgrund der

⁵⁸Vgl. [9] Leitenberger. *Computer in der Raumfahrt Teil 2*. 2001.

⁵⁹Vgl. [16] Lövskog. *Firmware update over the vacuum*.

⁶⁰11, Quelle:

⁶¹Vgl. [16] Lövskog. *Firmware update over the vacuum*.

⁶²Vgl. [26] Unbekannt. *The Star Computer*.

⁶³Vgl. [19] Nelson. *Voyager: Did you know?*

⁶⁴Vgl. [30] unbekannt. *Distributed Computing On Board Voyager and Galileo*.

⁶⁵Vgl. [14] Ludwig and Taylor. *Voyager Telecommunications*.

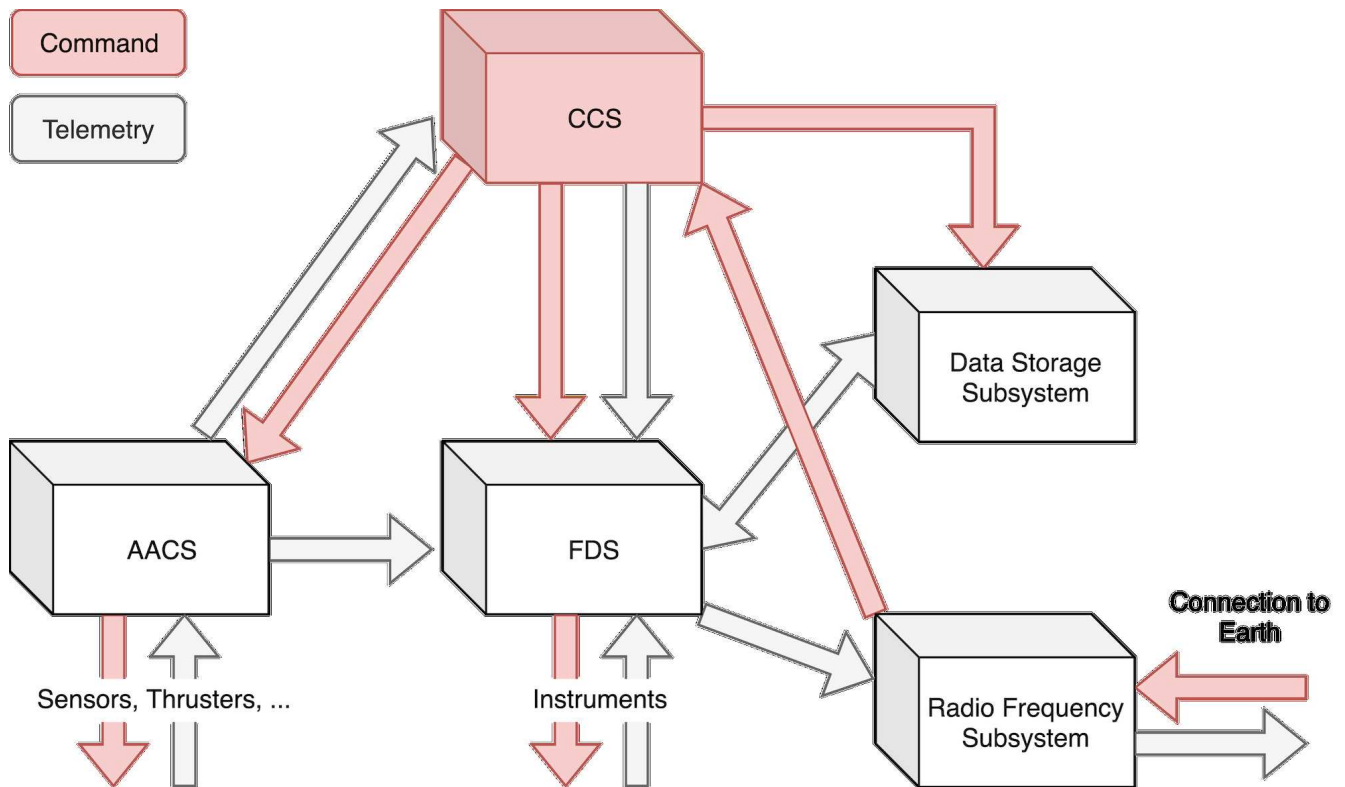


Figure 7: Übersicht über das verteilte Rechnen der Voyager Sonde ⁶⁰

fast tagelangen Signallaufzeit ohnehin nicht mehr. Zudem hat die Weltraumbehörde kaum Personal, welches mehr die spezielle Programmiersprache der Sonde beherrschen. Erst vor kurzem stellte die NASA deshalb solche Experten wieder ein.⁶⁶⁶⁷

4.2.5 Missionskritische Fehler

Ein Fehler passierte direkt beim Start der Voyager 2 Sonde am 11. August 1977. Auf die starken Beschleunigungen und Vibrationen war das Steuersystem der Voyager-Sonde nicht vorbereitet. Der Weg ins All dauerte mehrere Minuten, im Lagekontrollsystem wurde die Fehlerbehandlung aktiv. Für die Sonde sah es so aus, als ob es einen Defekt in ihren Computersystemen gab. Zuerst wurde wieder auf den Backup-Rechner umgeschaltet. Da das natürlich keinen Effekt erzielte, da die Sonde während des Starts nur als Nutzlast mitflog, wurde mehrfach zwischen den Backup-Systemen umgeschaltet. Als der Start vorbei und die Sonde in der Erdumlaufbahn flog, versuchte die NASA wieder Kontakt zur Sonde aufzunehmen. Die Sonde war in einem unbekannten Status, der es nicht ermöglichte, auf Kommandos von der Erde aus zu reagieren. Einem NASA Ingenieur, der an der Programmierung der Sonde beteiligt war, gelang kurz darauf wieder der Kontakt zur Sonde. Umgehend wurde ein Update an die Sonde geschickt. Zudem wurde auch die Firmware der Voyager 1 Sonde, welche erst kurz darauf startete aktualisiert. Trotz des Updates sollte noch ein weiterer schwerwiegender Fehler

bei der Voyager 2 Sonde auftreten.

Am 6. April 1978, noch bevor Voyager 2 einen einzigen Planeten besichtigt hatte, trat ein Fehler im Empfangsmodul auf. Die Schutzmechanismen reagierten aber zunächst wie erwartet und schalteten vom primären auf den zweiten Empfänger um. Dabei stellte sich heraus, dass dieser bereits von einem unbekannten früheren Ereignis beschädigt war, wenn auch nicht so stark wie der erste Empfänger.

Dadurch wurde nach einiger Zeit durch die Fehlererkennungs-Routinen automatisch auf den ersten Empfänger umgeschaltet. Kurz danach fiel dieser Empfänger komplett aus, sodass die Sonde nicht mehr durch die Bodenstation gesteuert werden konnte. Dieser Zustand dauerte 7 Tage an, bis die Sonde autonom wieder auf den Backup-Empfänger umschaltete. Der war zwar auch nicht voll funktionstüchtig, war aber nicht wie der erste komplett ausgefallen. Nachdem die NASA einige spezielle Kommandos an die Sonde geschickt hatte, hatte sie wieder die volle Kontrolle über das Raumfahrzeug.⁶⁸⁶⁹

4.3 Verteiltes Rechnen auf Galileo

Das Verteilte Netzwerk der Galileo Raumsonde war und ist bis heute eines der größten Computernetzwerke auf einer unbemannten Raumsonde. Anders als bei verteilten Systeme-

⁶⁶Vgl. [13] Leitenberger. *Voyagers: Die Sonde*. 2017.


⁶⁷Vgl. [14] Ludwig and Taylor. *Voyager Telecommunications*.

⁶⁸Vgl. [14] Ludwig and Taylor. *Voyager Telecommunications*.

⁶⁹Vgl. [18] Mosher. *NASA's famous Voyager probes nearly failed during launch — here's what went terribly wrong*. 2017.

men vergleichbarer Missionen wie den Voyager Sonden, wurden im Galileo Orbiter neben großen, die Raumsonde verwaltenden und steuernden Subsystemen auch mehrere kleine Subsysteme eingesetzt, die speziell für bestimmte Experimente zuständig waren. Das Netzwerk aus all diesen Subsystemen war durch über einen Datenbus verbundene Mikroprozessoren realisiert. Dabei trat das Prinzip des verteilten Rechnens in zwei verschiedenen Ausführungen auf. Zum einen gab es verteiltes Rechnen in Form von Aufteilung innerhalb eines Systems wie dem Command and Data Subsystem, welches als größtes System gleich mehrere Prozessoren enthielt, auf die Aufgaben verteilt wurden. Zum anderen gab es die Aufteilung von Aufgaben auf größtenteils voneinander unabhängige Subsysteme. Das gesamte Netzwerk und somit alle Interaktionen zwischen den verschiedenen Systemen war durch das CDS verwaltet. Dadurch konnte ein gewisser Grad an Organisation und ein gemeinsamer Basistakt über alle Subsysteme erreicht werden. Eine weitere wichtige Art des verteilten Systems, welche auf der Galileo Sonde Anwendung fand, war Redundanz. Alle wichtigen Subsysteme, waren in ihren Hardwarekomponenten doppelt also redundant vorhanden, sodass wenn ein System ausfiel, keine Funktion verlorenging.⁷⁰

4.3.1 Warum verteiltes Rechnen auf Galileo?

Beim Design des Galileo Orbiters entschied man sich aus verschiedenen Gründen für eine verteiltes Netzwerk in diesem Ausmaß. Wie bereits in 3.2 erwähnt, wurde aus Kostengründen und der Notwendigkeit von Fließkommaberechnungen relativ früh eine Kombination aus NSSC-1 und einem kleineren Mikroprozessorsystem, was einem System wie auf den Voyager Sonden sehr ähnlich gekommen wäre, ausgeschlossen. Stattdessen sollte es als Verwaltungssystem ein verteiltes System basierend auf dem Konzept des UDS geben. Zur Designzeit war der einzige Mikroprozessor,  als für die Raumfahrt geeignet angesehen wurde, der bereits beschriebene CDP1802. Obwohl dieser selbst für damalige Verhältnisse ziemlich langsam war, überzeugten die Vorteile des niedrigen Stromverbrauchs und der extremen Spannungs- und Temperaturtoleranz. Um die Langsamkeit zu überwinden, wurden also mehrere Prozessoren verwendet, die über Software kombiniert wurden. Durch die Aufteilung des Systems und die Unterscheidung zwischen Verwaltungsprozessoren (HLM) und Terminalprozessoren (LLM) nach dem Konzept des UDS entstanden viele Vorteile, wie die Möglichkeit bei steigenden Anforderungen ohne großen Aufwand mehr LLMs hinzuzufügen. Auch Software konnte verteilt und speziell für einzelne Module gestaltet werden, was sowohl die Flexibilität als auch die Fehlertoleranz erheblich steigerte. Durch weitere Aufteilungen zwischen einzelnen Experimentssystemen und dem Verwaltungssystem (UDS), gewann man den großen Vorteil von weiterer Unabhängigkeit. So war jedes System größtenteils für sich selbst verantwortlich, was Interfaces zur Experimenthardware und Softwaredesign erheblich vereinfachte. Zum Beispiel konnten verschiedene Systeme in unterschiedlichen Programmiersprachen programmiert werden. Die einzigen Interaktionen, die zwischen den einzelnen Systemen stattfinden mussten, waren einfache Kommandos, Datentransfers und Synchronisationssignale.

⁷⁰Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space.*

Durch die Kombination all dieser Techniken konnte eine sehr große Ausfallsicherheit hergestellt werden, was im Fall von Galileo ein mehr als ausschlaggebender Grund war, da es in Gegensatz zu den meisten Interplanetarischen Missionen wie Voyager nur eine Raumsonde und somit nur eine Chance auf Erfolg gab.

Ein weiterer guter Grund für verteilte Systeme war der Aufbau des Orbiters. Dieser hatte auch eine Experimentplattform, die gegen die Rotationsachse der Sonde gedreht wurde, um dadurch statisch im Vergleich zum Raum zu erscheinen und damit zum Beispiel bessere Bilder zu erhalten. Durch die räumliche Verteilung der Steuerungssysteme CDS und AACCS auf beide Sektionen, wurde die ansonsten entstandene Verzögerung von Signalen zwischen den zwei Raumsonden-Teilen verhindert.⁷¹

4.3.2 Verteiltes Rechnen innerhalb eines einzelnen Systems (CDS)

Eine große Herausforderung aus dem Bereich des verteilten Rechnens bildete die Software des Command and Data Subsystems, da diese ein System, welches über mehrere Mikroprozessoren, Speichermodule und ein Bussystem verteilt war, zusammenführen und betreiben musste. Eine der wichtigsten Designentscheidungen, um eine gute Software möglich zu machen, war die bereits angesprochene Unterteilung der verschiedenen Prozessoren nach dem Prinzip des Unified Data System (UDS). Hier wurden die Mikroprozessoren, wie bereits in 3.2.1 beschrieben, in zwei Stufen unterteilt, von denen eine, die HLM, ausschließlich für Verwaltungsaufgaben innerhalb des Systems zuständig war, und die andere, die Terminalmodule (LLM) für zugewiesene Datenverarbeitungsaufgaben und Input/Output übernahm. Nur durch die Aufteilung der Prozessoren des CDS auf diese Stufen war es möglich, auch die Software zu unterteilen und das System zu organisieren. Nachdem also die Software der LLMs sehr speziell angepasst wurde, war vor allem die Software der High Level Module entscheidend, um das CDS funktionsfähig zu halten.

Diese war unterteilt in Vorder- und Hintergrundprozesse. Die Vordergrundprozesse waren angepasst an den Raumsonden weiten, Interrupt gesteuerten Basistakt und liefen so 15-mal pro Sekunde. Zu den vordergründigen Funktionen zählte ein Test auf die eigene Funktionsfähigkeit, das Betreiben einer Systemuhr und das Überprüfen und Steuern des System Buses. Sollten also Daten am Bus anliegen, werden diese von einem HLM zum entsprechenden Speicher bewegt.

Die Hintergrundprozesse waren weiter aufgeteilt auf sechs verschiedene Virtuelle Maschinen, von denen die Hälfte als privilegiert galt. Dies bedeutete, dass ihre Funktionen nicht unterbrochen werden konnten, auch nicht nach Ablauf des 2/3 Sekunden Zyklus der Hintergrundprozesse. Die privilegierten Maschinen hatten feste Funktionen zum Ausführen von Kommandos und größtenteils zur Fehlerüberprüfung der anderen Subsysteme und dem Reagieren auf fehlerhafte Abläufe und deren Wiederherstellung. Um diese Funktionen immer gewährleisten zu können, war der Code dieser in schreibgeschützten Teil des Hauptspeichers der HLMs gesichert.

Die nichtprivilegierten Virtuellen Maschinen waren hingen-

⁷¹Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space.*

gen zum Ausführen von Standardaufgaben, später geplanten Aufgaben und gespeicherten Sequenzen. Diese Funktionen konnten noch im Flug upgedated und verändert werden. Dadurch konnten häufiger Fehler in diesen Funktionen auftreten, weshalb diese nicht zum Not-Recovery Modus gehörten. Die Ausführung dieser Aufgaben wurden außerdem am Ende jedes Hintergrundzyklus unterbrochen, sollte sie noch nicht abgeschlossen sein.⁷²⁷³

4.3.3 Verteiltes Rechnen über Subsysteme hinweg

Über das gesamte Netzwerk des Galileo Orbiters gab es zwei große Aufgabe, welche vor allem durch verteiltes Rechnen übernommen wurden. Zum einen wurde die Datenverarbeitung der einzelnen Experimente auf das Command and Data Subsystem ausgelagert. Die Mikroprozessoren der einzelnen Experiment Subsysteme übernahmen also nur Steuerungsaufgaben, während gesammelte Daten und Fotos direkt über den Datenbus an das CDS weitergeleitet wurden, um dort von den LLMs weiterverarbeitet oder zwischengespeichert zu werden. Der zweite große, verteilte Funktionsbereich waren Fehlerüberwachung und Fehlerschutzmechanismen (Fault protection und Recovery). Unabhängig von Datentransporten beschränkten sich die Interaktionen zwischen den Subsystemen meistens auf einfache Befehle wie On/Off.

4.3.4 Fehlerschutz durch verteiltes Rechnen

Als verwaltendes System war es das Command and Data Subsystem, das die meisten Fault protection Funktionen übernahmen. Dabei wurden nicht nur CDS interne, sondern auch Systemweite Fehler erkannt. Sobald ein Fehler erkannt wurde, wurde dieser vom CDS nach Schwere und Auswirkungen kategorisiert, um eine entsprechende Reaktion auszuführen. Siehe Figure 8 Dabei unterschied man: Privilegierte Fehler, wie Verlust der Stromversorgung, die direkt essentielle Funktionen des CDS, wie Telemetrie, das Ausführen von Kommandos und den Fehlerschutz selbst beeinträchtigten.

Nichtprivilegierte Fehler, die externen Funktionen des Systems und der Raumsonde, wie die Experimente, beeinträchtigten.

Message-only-errors: Fehler die keine der Systemfunktionen störten.

Während der Verarbeitung eines aufgetretenen Fehlers wurde die Funktion des CDS angehalten, um die Raumsonde vor weiteren Fehlern zu schützen. Einzige Ausnahme waren kritische Missionsabschnitte, während denen mindestens einer der beiden redundanten Strings weiterarbeiten musste. Bei der Fehlerverarbeitung wurde je nach Schwere und System, von dem der Fehler kam, eine Reaktion aktiviert. Zum Beispiel gab es den Radio frequency loss also der Verlust der Verbindung zur Erde. Wurde ein solcher Fehler erkannt, versuchte das CDS diese über einen der redundanten Verbindungskomponenten wiederherzustellen und den Fehlerhaften vorerst abzuschalten. Gelang es nicht, eine neue Verbindung herzustellen, versetzte das CDS die Raum-

sonde in den Safe-Mode. Dieser Modus wurde bei einem nicht bearbeitbaren Fehler eingeschaltet, um die Raumsonde zu retten. Dabei wurden alle Funktionen und Manöver angehalten, möglichst viele Komponenten abgeschaltet, um Strom zu sparen und versucht ein Notsignal zu senden. Außerdem wurde die Temperatur der Experimente reguliert, damit diese nicht beschädigt wurden. Ein weiteres Beispiel sind die Science alarms, bei denen ein Fehler in einem der Experimente festgestellt wurde. Dieses Experiment wurde dann abgeschaltet und beheizt, um mögliche Beschädigungen zu verhindern.

Auch das AACS wurde, obwohl es seine eigene Fault protection hatte, vom CDS unterstützt. Dazu sendete das AACS alert codes and das CDS die über den momentanen Status des AACS informierten, sodass das CDS entsprechend reagieren konnte. War zum Beispiel kein zu überprüfender Status vorhanden, wurde der Alert code 0 oder NO-OP also kein Status geschickt. Wurde jedoch ein alert code wie 4 oder Command Error gesendet, so wusste das CDS, dass es die Raumsonde in den Safe-Mode versetzen musste. Zusätzlich zu den alert codes sendete das AACS in regelmäßigen Abständen von 666ms einen sogenannten Herzschlag, war dieser zu oft nicht gültig (etwa 5s lang), so wechselte das CDS die aktive ACE Einheit des AACS, bis wieder ein gültiger Herzschlag gesendet wurde.⁷⁵⁷⁶⁷⁷⁷⁸

4.3.5 Spätere Änderungen am System

Auch beim Galileo Orbiter war die Möglichkeit gegeben, während des Fluges Software bestimmter Systeme zu verändern. So wurde zum Beispiel die Software des Magnetometers aktualisiert. Am wichtigsten wurde diese Funktion allerdings nach dem Auftreten einiger, teilweise Missionskritischer Fehler, wie einem defekten Anteil des Magnetbandes. Hier wurde per Softwareupdate verhindert, dass dieser Teil weiter benutzt wurde.

Das größte Problem während der Mission stellte jedoch das Nichtausfahren der High Gain Antenne dar. Diese war durch den Transport der Raumsonde verklemmt und es konnte, nachdem dies beim ersten Erdvorbeiflug festgestellt wurde, kein Weg gefunden werden diese zu befreien. Dennoch konnte die Low Gain Antenne benutzt werden, um Daten an die Erde zu senden. Das Problem hierbei war allerdings, dass die Datenrate wesentlich geringer war, weswegen die Daten um einiges mehr komprimiert werden mussten. Die letztendliche Lösung der Ingenieure war deswegen, die Software des CDS und des AACS komplett umzuschreiben, sodass große Teile dieser Systeme zur Datenkompression verwen-

⁷⁵Vgl. [2] Daniel P. Siewiorek. *Reliable Computer Systems Design and Evaluation*. 1992.

⁷⁶Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁷⁷Vgl. [23] Siewiorek and Narasimhan. *APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS*.

⁷⁸Vgl. [24] Turner. *COMMAND AND TELEMETRY IN AUTONOMOUS SPACECRAFT DESIGN*.

⁷²Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*.

⁷³Vgl. [15] Lutz. *Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems*.

⁷⁴24.

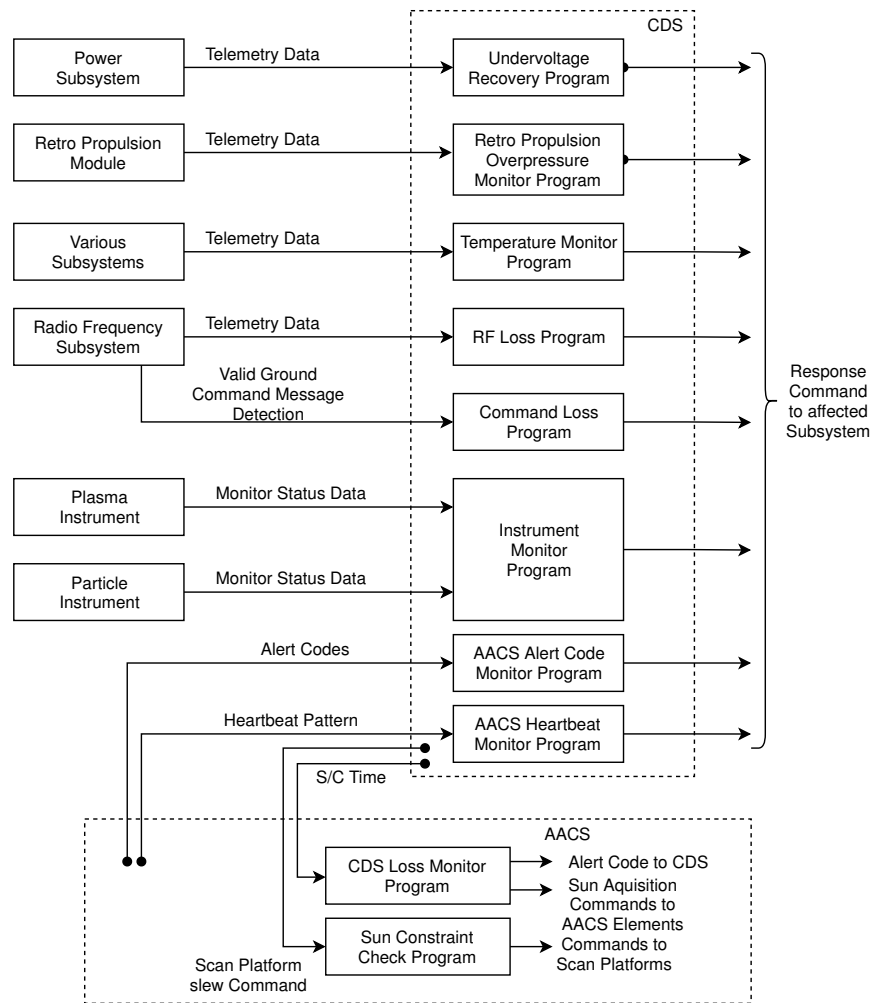


Figure 8: Galileo Fault protection Diagram⁷⁴

det werden konnten.⁷⁹⁸⁰⁸¹⁸²

5. BEWERTUNG DES VERTEILTEN RECHNENS AUF VOYAGER UND GALILEO

5.1 Vergleich zwischen Voyager und Galileo hinsichtlich der Viking Sonde

Auch auf der Viking Sonde war jedes Subsystem redundant vorhanden. Diese waren so miteinander verschaltet, dass falls ein Teil eines Rechners ausfiel, dieser auf das Gegenstück im Backupcomputer zugreifen konnte. Die


⁷⁹Vgl. [29] unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space.*

⁸⁰Vgl. [6] Garret. *Lisping at JPL 1992-1993 - Miscellaneous stories.* 2002.

⁸¹Vgl. [10] Leitenberger. *Der Galileo Orbiter.* 2017.

⁸²Vgl. [8] Jansma. *Open! Open! Open! Galileo High Gain Antenna Anomaly Workarounds.*


Sonde bestand aus einem Lander und einem Orbiter, welche vor der Landung noch miteinander verbunden waren. So mussten bereits diese zwei Systeme (GCSC und CCS, siehe 3.4) miteinander über Interfaces kommunizieren können. Verglichen mit der Voyager Sonde und später der Galileo Sonde wurde hier allerdings nicht wirklich verteiltes Rechnen angewandt. Beide Teile der Sonde (Lander und Orbiter), hatten jeweils ein eigenen großen Zentralcomputer. Nach der Trennung waren diese bis auf die Weiterleitung der Daten des Landers durch den Orbiter, auf sich alleine gestellt.

Bei den Voyager Sonden und der Galileo Sonde gab es deutlich mehr Subsysteme, welche auch viele enger miteinander verzahnt waren. Dies sollte auch der Ausfallsicherheit dienen,  der Ausfall eines Subsystems unter Umständen kompensiert werden konnte.

5.2 Bewertung der verteilten Systeme und ihrer Auswirkungen

Während die Voyager Sonden im Vergleich zu ihrem

Vorgänger Viking, bis auf die Erweiterung um das Lagekontrollsystem, nur wenig Veränderungen am verteilten System erfuhren, gab es bei dem Galileo Projekt einige größere Innovationen in Form des ausschließlichen Gebrauchs von Mikroprozessoren und der direkten Anbindung von Prozessoren an die Experimente. Beide Projekte brachten große Fortschritte in der funktionalen Verteilung von Systemen, wie Steuerungs- und Positionssystem neben Verwaltungssystem. Der Galileo Orbiter machte außerdem erste Erfolge in verteilter VM-Software innerhalb eines einzelnen Systems. In beiden Systemen gab es Schwachstellen. Primär kamen diese durch fehlende Erfahrung bei Auswirkungen der Strahlung des Jupiters auf die Computerhardware, die öfters zu Problemen, wie Speicherstörungen, führte, zustande. Ansonsten waren beide Projekte natürlich durch die Hardware ihrer Zeit limitiert. So waren Prozessoren zu langsam und Speicher zu klein, um große Datenmengen zu verarbeiten. Auch die schlechte Übertragungsmöglichkeit von Daten, welche oft starke Kompressionen, niedrige Qualität oder fehlerhafte Übertragungen zur Folge hatte, war eines der größten Probleme der Missionen.

In den Folgejahren nach den Starts der beiden Missionen gab es viele, große Entwicklungen im Bereich der Mikroprozessoren und Speicher, die wesentlich leistungsfähigere, verteilte Systeme ermöglichten. So wurden recht schnell weitere Raumsonden mit verteilten, modularen Systemen geplant, deren geschätzte Kosten sich gegenüber der Galileo Sonde  **erhöhen** werden konnten. Auch im Bereich der Software wurde weiter an der Unterstützung höherer Programmiersprachen, wie sie teilweise schon in den beiden Projekten vorhanden war, festgehalten.

Insgesamt lassen sich also beide Projekte, egal ob bereits abgeschlossen oder nicht als Erfolge verzeichnen, die große Fortschritte sowohl in der Wissenschaft als auch in der Raumsondenentwicklung brachten.⁸³⁸⁴

bibliography

References

- [1] Keine Angabe. *Formelsammlung Naturwissenschaften*. Cornelsen, 2013.
- [2] Robert S. Swarz Daniel P. Siewiorek. *Reliable Computer Systems Design and Evaluation*. Digital Press, 1992, p. 671.
- [3] Samuel G. Deese. *APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS*.
- [4] Steven J. Dick. *Mission to Jupiter Why We Explore*. 2007. URL: https://www.nasa.gov/exploration/whyweexplore/Why_We_26.html.
- [5] John A. Zoutendyk Donald K. Nichols Lawrence S. Smith. *SINGLE EVENT UPSET IMMUNE INTEGRATED CIRCUITS FOR PROJECT GALILEO*.
- [6] Ron Garret. *Lisping at JPL 1992-1993 - Miscellaneous stories*. 2002. URL: <http://www.flownet.com/gat/jpl-lisp.html>.
- [7] Calvin J. Hamilton. *Galileo Project*. URL: <http://www.iki.rssi.ru/solar/eng/galifs.htm#spacecraft>.
- [8] P. A. "Trisha" Jansma. *Open! Open! Open! Galileo High Gain Antenna Anomaly Workarounds*.
- [9] Bernd Leitenberger. *Computer in der Raumfahrt Teil 2*. 2001. URL: <https://www.bernd-leitenberger.de/computer-raumfahrt2.shtml>.
- [10] Bernd Leitenberger. *Der Galileo Orbiter*. 2017. URL: <https://www.bernd-leitenberger.de/galileo-orbiter.shtml>.
- [11] Bernd Leitenberger. *Verteiltes Rechnen Voyager*. 2017. URL: <https://www.bernd-leitenberger.de/img/voyager-computer.gif>.
- [12] Bernd Leitenberger. *Viking Sonden*. URL: <https://www.bernd-leitenberger.de/viking.shtml>.
- [13] Bernd Leitenberger. *Voyagers: Die Sonde*. 2017. URL: <https://www.bernd-leitenberger.de/voyager-sonde.shtml>.
- [14] Roger Ludwig and Jim Taylor. *Voyager Telecommunications*. URL: <https://voyager.gsfc.nasa.gov/Library/DeepCommo.Chapter3--141029.pdf>.
- [15] Robyn R. Lutz. *Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems*.
- [16] Thomas Lövskog. *Firmware update over the vacuum*. URL: <https://www.additude.se/bloggar/thomas-lovskog/firmware-update-over-the-vacuum/>.
- [17] Michael Meltzer. *Mission to Jupiter A History of the Galileo Project*. 2007.
- [18] Dave Mosher. *NASA's famous Voyager probes nearly failed during launch — here's what went terribly wrong*. 2017. URL: <https://www.businessinsider.de/nasa-voyager-probes-rocket-leak-computer-problems-2017-12?r=US&IR=T>.
- [19] Jon Nelson. *Voyager: Did you know?* URL: <https://voyager.jpl.nasa.gov/mission/did-you-know/>.
- [20] Jon Nelson. *Voyager Fast Facts*. URL: <https://voyager.jpl.nasa.gov/frequently-asked-questions/fast-facts/>.
- [21] Jon Nelson. *Voyager Mission Overview*. URL: <https://voyager.jpl.nasa.gov/mission/>.
- [22] Jon Nelson. *Voyager Mission Timeline*.
- [23] Daniel P. Siewiorek and Priya Narasimhan. *APPLICATION OF MICROPROCESSORS TO INTERPLANETARY SPACECRAFT DATA SYSTEMS*.
- [24] Philip R. Turner. *COMMAND AND TELEMETRY IN AUTONOMOUS SPACECRAFT DESIGN*.
- [25] Unbekannt. *From Sequencers to Computers: Exploring the Moon and the Inner Planets*. URL: <https://history.nasa.gov/computers/Ch5-6.html>.
- [26] Unbekannt. *The Star Computer*. URL: <https://history.nasa.gov/computers/Ch5-5.html>.
- [27] Autor unbekannt. *Block Diagramm des Viking Orbiter CCS*. URL: <https://history.nasa.gov/computers/p158.htm>.
- [28] Autor unbekannt. *Distributed Computing On Board Voyager and Galileo Future unmanned spacecraft computers*. URL: <https://history.nasa.gov/computers/Ch6-4.html>.

⁸³Vgl. [28] unbekannt. *Distributed Computing On Board Voyager and Galileo Future unmanned spacecraft computers*.

⁸⁴Vgl. [5] Donald K. Nichols. *SINGLE EVENT UPSET IMMUNE INTEGRATED CIRCUITS FOR PROJECT GALILEO*.

- [29] Autor unbekannt. *Distributed Computing On Board Voyager and Galileo Galileo - True distributed computing in space*. URL: <https://history.nasa.gov/computers/Ch6-3.html>.
- [30] Autor unbekannt. *Distributed Computing On Board Voyager and Galileo*. URL: <https://history.nasa.gov/computers/Ch6-2.html>.
- [31] Autor unbekannt. *Distributed computing on board Voyager and Galileo*. URL: <https://history.nasa.gov/computers/Ch6-1.html>.
- [32] Autor unbekannt. *Galileo Engineering*. 2006. URL: <http://web.archive.org/web/20061006065048/http://www.wcresa.k12.mi.us/nasa/engineer.htm>.
- [33] Autor unbekannt. *Mariner*. 2017. URL: https://www.nasa.gov/mission_pages/mariner.
- [34] Autor unbekannt. *Mission and spacecraft library Mariner*. URL: <https://space.jpl.nasa.gov/msl/Programs/mariner.html>.
- [35] Autor unbekannt. *Mission and spacecraft library Pioneer*. URL: <https://space.jpl.nasa.gov/msl/Programs/pioneer.html>.
- [36] Autor unbekannt. *Solar System Exploration Galileo*. 2018. URL: <https://solarsystem.nasa.gov/missions/galileo/in-depth/>.
- [37] Autor unbekannt. *Voyager 2 Spacecraft Instruments*. URL: <https://www.jpl.nasa.gov/spaceimages/details.php?id=PIA22915>.
- [38] Autor unbekannt. *What is distributed computing*. URL: https://www.ibm.com/support/knowledgecenter/en/SSAL2T_8.2.0/com.ibm.cics.tx.doc/concepts/c_wht_is_distd_comptg.html.