DIEHL
Avionik Systeme

Pub. No. **D0501310**
Issue **1**

VERBUNDFÖRDERVORHABEN PROHMS ABSCHLUSSBERICHT ZUM DAV TEILVORHABEN 20F0001F

Classification		
Diary No.	D0501310.doc	
Number of Sheets	37	Consisting of
		Pages CONFIDENTIAL
		Pages RESTRICTED
	37	Pages UNCLASSIFIED
		Pages blank
Copy No.		

Manufacturer

Prepared by _____
(R. Schumacher)

Checked by _____
(J. Nielsen)

Approved by
(B. Petersen)

DIEHL
Avionik Systeme

Issue 1
Pub. No. D0501310

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 BESCHREIBUNG DES VORHABENS

Eine Hauptaufgabe von ProHMS war, die AIRBUS Systemtechnik mit einem Rechnersystem auszustatten, welches die neuen Anforderungen erfüllt, aber im Wesentlichen auf EFCS Technologie basiert. Das Rechnersystem erfuhr eine „umfangreiche“ Modifikation. Im Zuge dieser Arbeiten wurde auch die Flexibilität des EFCS Rechnersystems unter Beweis gestellt .

Insbesondere wurde eine Adaption in folgenden Bereichen durchgeführt

- Interface
- Backwall
- Similarisierung

Im Rahmen des Vorhabens wurden im Rahmen der ersten Stufe (ProHMS) ein Funktionsnachweis des Gesamtsystems gefordert.
Um Kosten zu sparen und das geforderte Ziel im Rahmen des Budgets realisierbar zu machen wurden entsprechende Module similarisiert.

Ein weiteres Ziel von ProHMS war die Weiterentwicklung des EFCS OS, so daß es

- Konfigurierbar ist,
- aus der Konfigurationsdatenbasis autom. Source Code generieren kann und
- eine Konzeption eines effizienten Entwicklungsprozesses basierend auf SCADE, für die Applikationsentwicklung erstellt wird.

1.1 Aufgabenstellung

Das Teilvorhaben der Diehl Avionik Systeme GmbH (DAv) „Rechnersystem der sekundären Flugsteuerung“ im Rahmen des nationalen Forschungs- und Technologievorhabens ProHMS (**Pro**zesskette **H**ochauftrieb mit **m**ultifunktionalen **S**teuerflächen) befasste sich mit der Konzeption und Entwicklung eines Rechnersystem zur Steuerung eines Flügels mit multifunktionalem Klappensystem. Der Hochauftriebsrechner dieses Systems hat die hohen Sicherheitsanforderungen zu erfüllen, wie sie im Bereich der Primärsteuerung von zivilen Transportflugzeugen bereits üblich sind. Die Wahrscheinlichkeit für den Totalverlust der Systemfunktionen mit katastrophalen Folgen muss unter 10^{-9} pro Flugstunde liegen.
Neben der eigentlichen Rechnerentwicklung ist die Effizienzsteigerung des Entwicklungsprozesses ein weiteres Aufgabenfeld im Rahmen dieses Vorhabens. Hierzu ist es erforderlich eine Straffung des Entwicklungsprozesses zu erarbeiten.

1.2 Voraussetzungen / Technische Ausgangsbasis

Voraussetzung und technische Ausgangsbasis für die Durchführung dieses Vorhabens war der erfolgreiche Abschluss des FuT-Vorhabens „Elektronische Flugsteuerung“, DAv-Teilvorhaben „Rechnersystem der primären Flugsteuerung“ FKZ LFT 9403B (Definitionsphase) und 20F9501L (Realisierungsphase). So wurde die Hardware- und Software Architektur des Hochauftriebsrechners vom Rechnersystem zur primären Flugsteuerung (EFCS) abgeleitet.

1.3 Planung und Ablauf der Vorhabens

Für die Realisierung des ProHMS Vorhabens wurde eine Dauer von drei Jahren geplant und zunächst ein Leistungszeitraum vom 01.02.2000 bis zum 31. 12.2003 bewilligt.
Am 19.4. 2002 wurde in Absprache mit der federführenden Firma und dem Projektträger Luftfahrtforschung eine Aufstockung der Mittel und Verlängerung des Vorhabens bis zum 31.12. 2003 beantragt, die im wesentlichen einer Weiterführung und Vertiefung der geplanten Arbeiten zum Gegenstand hatten. Dieser Antrag wurde mit Bescheid vom 3. 7. 2002 bewilligt.
ProHMS
Als Abschluss der Realisierungsphase wurde die Demonstration des ProHMS Rechners am Hochauftriebs-Testrigg bei Airbus Deutschland in Bremen geplant.

1.4 Zusammenarbeit mit Verbundpartnern / anderen Stellen

1.4.1 Parallel laufende Fortschritte bei anderen Stellen

In Deutschland gibt es kein Vorhaben mit einer ähnlichen Zielsetzung.

1.4.2 Zusammenarbeit mit den Verbundpartnern

Entsprechend dem Charakter eines Verbundfördervorhabens sind die Arbeiten der Partner eng miteinander verzahnt und bauen

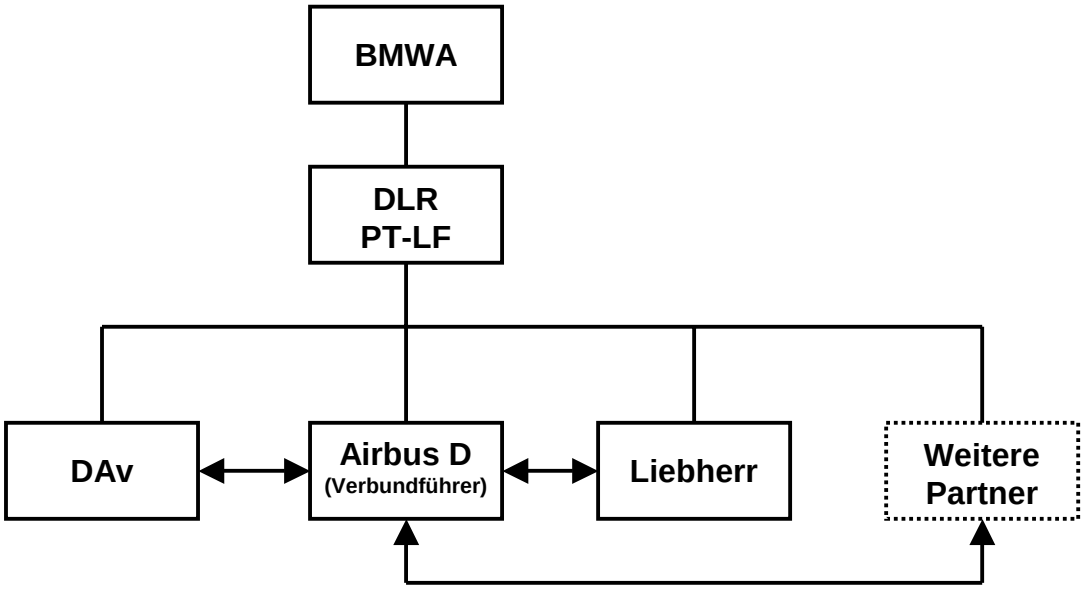


Bild 1 Verbundpartner des ProHMS Projekts

aufeinander auf. Die Systemanforderungen sowie die Spezifikation der Schnittstellen wurden in enger Zusammenarbeit zwischen Airbus Deutschland und DAv sowie Liebherr Lindenberg (LLI) erarbeitet und abgestimmt.

2 ERGEBNISSE DES VORHABENS

Ein wesentliches Ergebnis des Vorhabens war für die Pro-HMS Systemarchitektur die Ansteuerung aller SMART Aktuatoren über einen Rechner mit zentralem Redundanzmanagement. Die Einführung einer Zwischenebene mit Klappen-Paar Controllern war somit nicht erforderlich.

Eine besondere Herausforderung stellte die schnelle Synchronisation der linken und rechten Seite eines Klappenpaares dar.

2.1 Allgemeine Übersicht über das ProHMS Rechnersystem

2.1.1 ProHMS Übersicht

Die im Rahmen des ProHMS Vorhabens konzipierte Hochauftriebskonfiguration enthält ein neuartiges Hochauftriebssystem mit unabhängigen ansteuerbaren Zusatzklappen. Diese zusätzlich angebrachten Klappen mit hoher Dynamik erlauben die Implementierung leistungssteigender Eigenschaften, wie

- Verbesserung der Start- und Steigflugleistungen
- Widerstandsminimierung im Reiseflug
- Steiler oder langsamer Landeanflug zur Lärmreduktion
- Böenlastminderung
- Manöverlastminderung
- etc...

Bei dieser Konfiguration wird das Querruder durch eine äußere Landeklappe ersetzt. Somit reicht das Landeklappensystem über die gesamte Spannweite. An der Hinterkante der Landeklappen sind vier zusätzliche Stellflächen (Tabs) und im Flügelaußenbereich ein Spoiler angebracht. Jede dieser Stellflächen kann von zwei Rechnermodulen (SFCU = **S**lat **F**lap **C**ontrol **U**nit) angesteuert werden. Bei den bisherigen Landeklappensystemen wird im allgemeinen eine durchgehende Transmissionswelle zur Leistungsübertragung zwischen zentralem Antriebsmotor und den Antriebsstationen der Flügelvorderkantenklappen (Slats) und Hinterkantenklappen (Flaps) verwendet. Bei der Systemlösung des ProHMS Vorhabens wird jede Klappe des Landeklappensystems einzeln von dem ProHMS Rechnersystems angesteuert, so dass sowohl ein spannweitig differenzierter Ausschlag als auch ein zwischen linken und rechtem Tragflügel unterschiedlicher Ausschlag möglich ist.

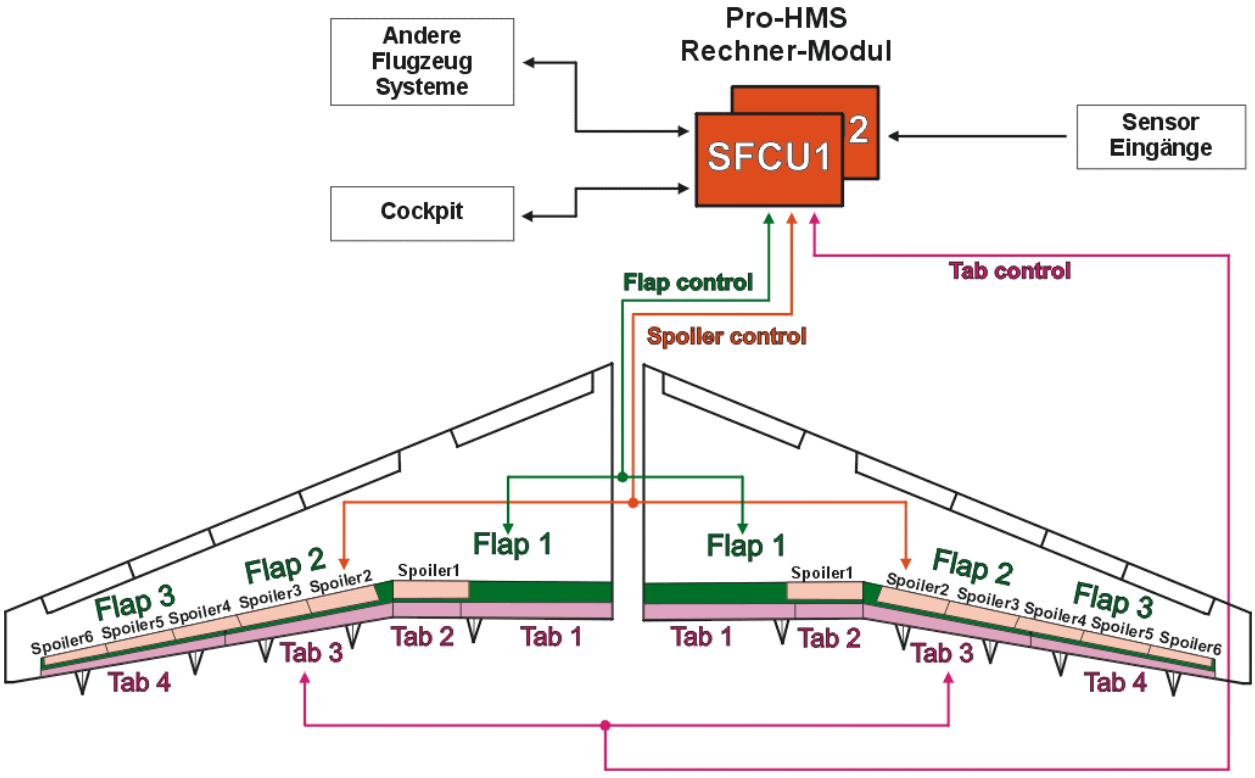


Bild 2 Ansteuerung des ProHMS Klappensystems mit den SFCUs

2.1.1.1 Aufbau der SFCU

Die grundlegende Rechnerarchitektur wurde aus den Arbeiten und Ergebnissen zum Forschungsvorhaben EFCS abgeleitet (siehe Abschlussbericht zum DAv Teilvorhaben 20F9501 L). Auf einen dissimilaren Aufbau der SFCU in Hardware und Software wurde aus Performance- und Kosten-Gründen verzichtet. ProHMS

2.1.1.2 Integration von Funktionen

ProHMSDie SFCU Rechnerarchitektur basiert auf zusammenhängende Funktionsgruppen. Es werden hier Funktionen betrachtet, deren Sicherheitsanforderungen eine Realisierung in folgender Form erzwingt:

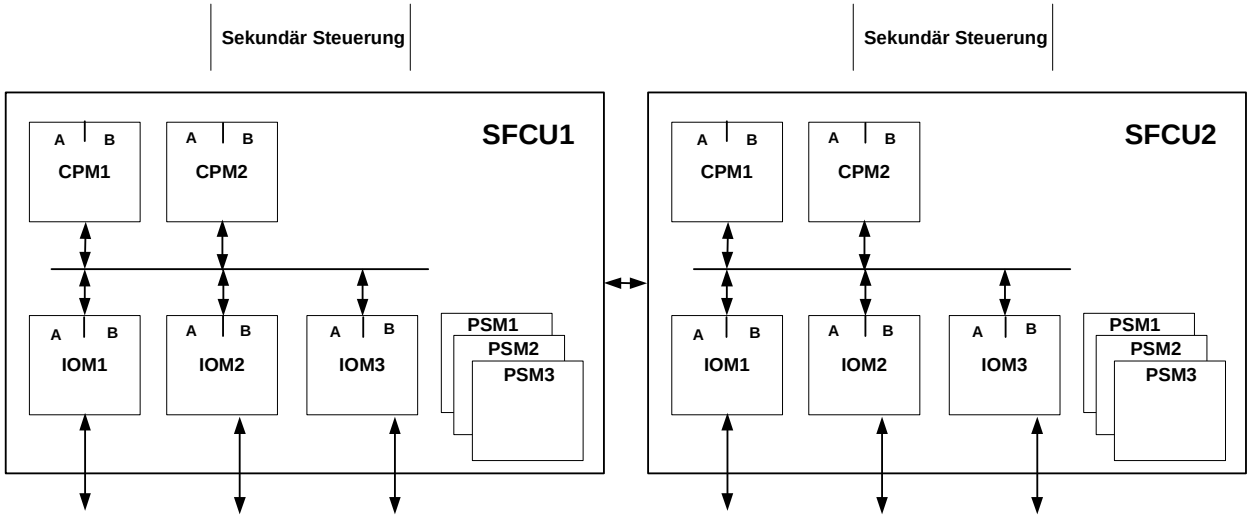
1. die Realisierung der Funktion muss Fehler absolut sicher selbst erkennen und passivieren können,
2. die Realisierung der Funktion muss fehlertolerant arbeiten und alle fehlerbedingten Rekonfigurationen selbst durchführen.

- Eine in diese Kategorie fallenden Funktionen ist die sicherheitskritische Funktion sekundäre Flugsteuerung.

Diese Funktionen werden in zwei gleiche, parallel laufende Rechner (SFCUs) integriert, wobei jeder Rechner in der Lage ist, die komplette Funktionalität durchzuführen

2.1.1.3 Rechnerstruktur und Funktionsaufteilung

Die modulare Rechnerarchitektur der SFCUs wurde über verschiedene Funktionsmodule aufgebaut, die in Duplex-Architektur realisiert wurden. Jedes Funktionsmodul besteht aus zwei Einheiten (Lane A und Lane B), die gleiche Software enthalten und es somit ermöglichen, dass sich die beiden Lanes innerhalb eines Funktionsmoduls gegenseitig überwachen (siehe Bild 3). Hierbei sind folgende Funktionsmodule zu unterscheiden: Computing Module (CPM), Input/Output Module (IOM), Maintenance Module (MAM) und Power Supply Module (PSM). Diese wurden in einer Kabinett-Lösung (siehe Bild 12) zum SFCU integriert.



CPM	Computing Module
IOM	Input/Output Module
PSM	Power Supply Module
A	Lane A
B	Lane B

Bild 3 Integration von Funktionen in zwei parallel laufenden Rechnern

Die Aufgaben der einzelnen Module sind wie folgt:

- CPM1
  CPM1 bearbeitet die Regelgesetze für die sekundäre Flugsteuerung und den Kern des Fehlermanagements für das Rechnerkabinett wie auch für das komplette Hochauftriebssystem, dem sogenannten **S**lat **F**lap **C**ontrol **S**ystem (SFCS). Bei beiden Funktionen handelt es sich um sicherheitskritische Funktionen. Beide zusammen werden in ihrer Gesamtheit vereinfachend Flight Control-Prime (FC-Prime) Funktion genannt. Solange ein CPM die FC-Prime Funktion bearbeitet (im FC-Prime Mode ist), kommandiert es seine Aktuatoren.
- CPM2
  Solange CPM1 intakt ist, ist CPM2 im FC-Second Mode. Das heißt, CPM2 bearbeitet die gleichen Funktionen wie CPM1, kommandiert aber nicht seine Aktuatoren. Fällt CPM1 aus, wechselt CPM2 in den FC-Prime Mode.
- IOM1, IOM2 und IOM3
  Diese Module führen das Einlesen und die Ausgabe von Daten sowie das dazugehörige Fehlermonitoring durch. Zusätzlich positionieren und überwachen sie die Aktuatoren.

2.1.1.4 Strategie des Fehlermanagements

Naturgemäß ist die Arbeitsweise des Fehlermanagements in so einem System sehr mannigfaltig. Deshalb soll hier der Kernpunkt des Fehlermanagements, nämlich die Rekonfiguration zwischen FC-Prime und FC-Second Mode, sowie die Ansteuerungen von Stellflächen an einem Beispiel erläutert werden.

Bild 4 zeigt die Ansteuerung eines Flaps. Jede Flap-Steuerfläche wird über zwei Aktuatoren angesteuert, wobei Aktuator 1 (ACT1) dem Rechner SFCU1 bzw. Aktuator 2 (ACT2) dem Rechner SFCU2 zugeordnet ist. Im fehlerfreien Fall wird die Flap-Stellfläche von SFCU1 über seinen Stellmotor ACT1 und von SFCU2 über den Stellmotor ACT2 angesteuert. Zu diesem Zweck kommandieren SFCU1 und SFCU2 den jeweiligen Aktuator in den „Active Mode“. D.h., beide SFCUs steuern aktiv über die jeweiligen Aktuatoren die Stellfläche an. Für die tatsächliche Ansteuerung von ACT1 und ACT2 steht der SFCU1 und der SFCU2 jeweils das Modul IOM2 zur Verfügung.
Tritt nun zum Beispiel in CPM1 von SFCU2 ein Fehler auf, so wechselt CPM2 von SFCU2 von FC-Second in den FC-Prime Mode und der entsprechend zugeordnete Aktuator von der SFCU2 wird weiterhin angesteuert.
Somit bearbeitet SFCU2 auch nach einem ersten Fehler im CPM noch die FC-Prime Funktion.
In gleicher Weise wiederholt sich die Rekonfiguration bei weiteren Fehlern (siehe Bild 4).

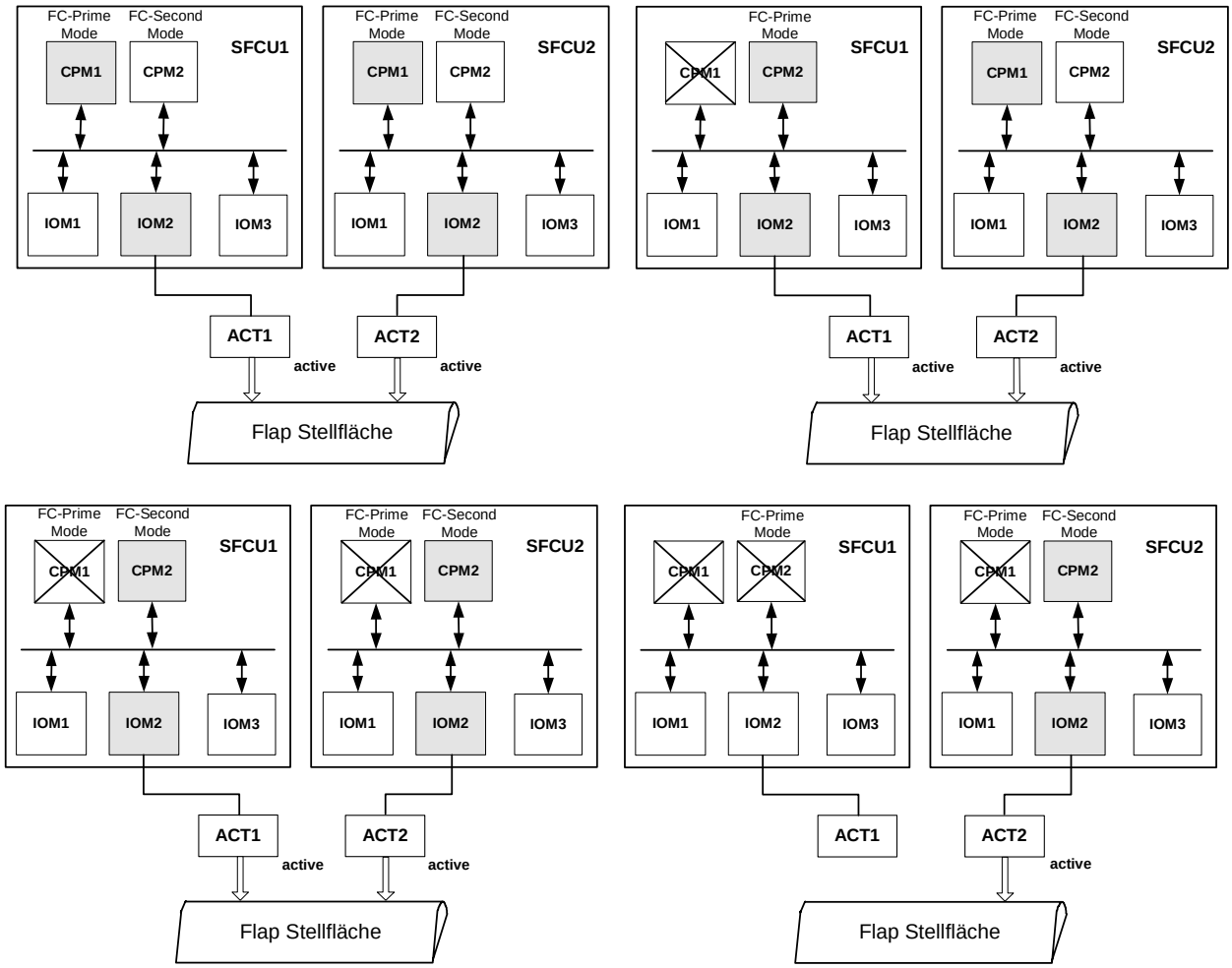


Bild 4 Arbeitsweise des Systems bei Modulfehlern

In Anbetracht der Tatsache, dass für dieses Forschungsvorhaben keine Flugerprobung vorgesehen war, wurde aus Kosten- und Termingründen auf eine Integration der CPM2 verzichtet. Zum Zwecke der Bodenerprobung am Testrig bei Airbus Deutschland (Bremen) hatte dies keinen Einfluss auf die Funktionalität des ProHMS Rechnersystems. Die Integration eines CPM2 im Redundanzmanagement eines solchen Systems wurde im Programm EFCS erprobt.

2.1.2 Prinzipieller Aufbau der Software

Die Struktur der Software in jeder Lane eines Moduls ist schalenförmig gestaltet. Wir unterscheiden hierbei zwischen dem Kernbereich des Betriebssystems, der sogenannten inneren Schale, dem Redundanzbereich (mittlere Schale) und dem Applikationsbereich der äußeren Schale.

Im innersten Bereich der Software (Kernbereich des Betriebsystems) zum Beispiel einer CPM-Lane befinden sich:
- die Basisfunktionen des Betriebsystems,
- das Kommunikationsnetzwerk.

In der mittleren Schale (Redundanzbereich des Betriebsystems) befinden sich
- alle Konsolidierungsmechanismen,
- alle Synchronisierungsmechanismen,
- alle Mechanismen des BIT,
- alle Mechanismen für die lokalen Maintenance-Aufgaben.

Auf der äußeren Schale (Applikationsbereich) befinden sich die anwendungsbezogenen Funktionen wie:
- der Kern des Fehlermanagements (CoCo) für das Gesamtsystem und
- die Flugregelgesetze

Durch diese strikte Trennung der einzelnen Funktionen wurden folgende Ziele erreicht:
- Den Applikationen wird vom Betriebssystem eine klare Schnittstelle angeboten, die APEX genannt wird.
- Alle Konsolidierungsmaßnahmen für die Applikationen (Regelgesetze) wie auch alle Redundanzmaßnahmen hinsichtlich Sensorik, Stellmotorik und des Rechners selbst werden vom Betriebssystem außerhalb der Applikationen durchgeführt. Damit wurden die Applikationen klar von der Redundanzverwaltung getrennt.
- APEX stellt der Software von Anwenderfunktionen eine Quasi-Simplexschnittstelle an Daten zur Verfügung.
- Es konnte eine saubere Datenkonsistenz innerhalb des redundanten Rechnersystems aufgebaut werden; das heißt es ist garantiert, dass redundante Einheiten innerhalb des Rechnersystems im gleichen Zyklus immer auch zu gleichen Ergebnissen und somit gleichen Entscheidungen kommen. Das byzantinische Generalsproblem, das Fundamentalproblem aller redundanten Systeme, kann somit exakt gelöst werden.

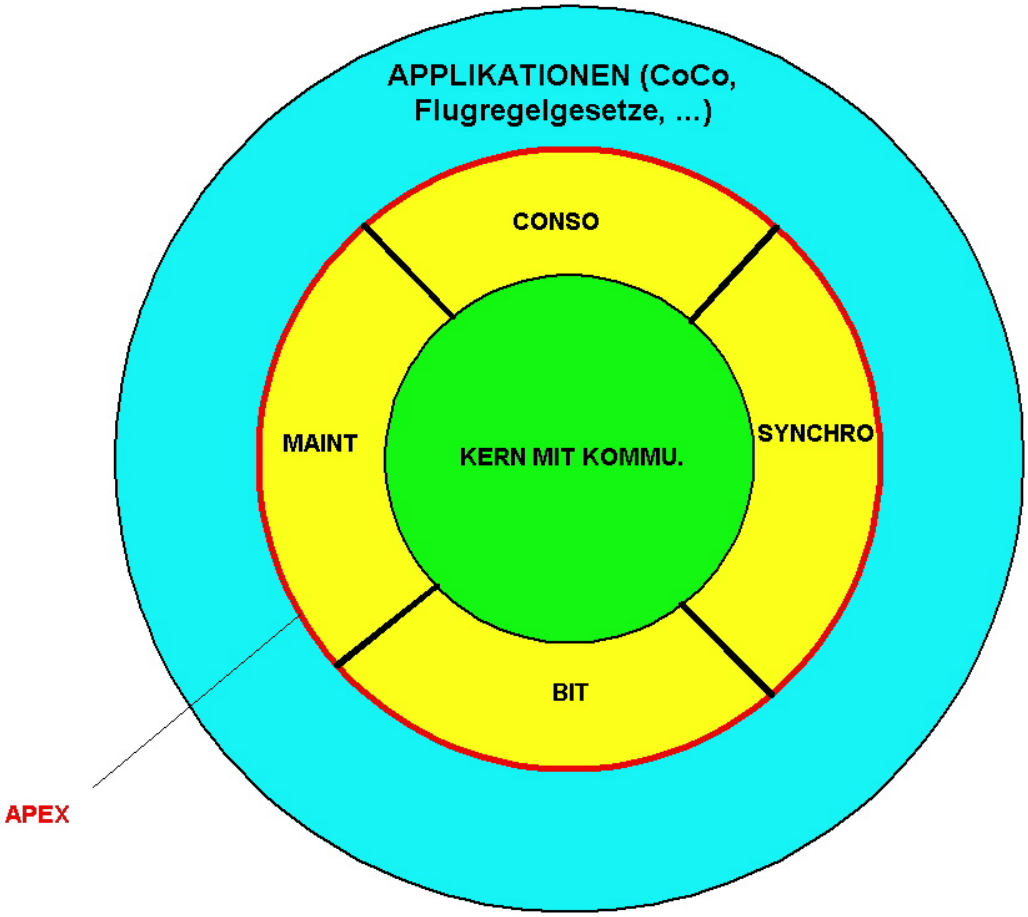


Bild 5 Schalenstruktur der Software im CPM

CONSO	...Konsolidierung	SYNCHRO	...Synchronisation	CoCo	...Configuration Control
BIT	…Built in Test	KOMMU	...Kommunikationsnetzwerk		
MAINT	...Maintenance	APEX	...Schnittstelle zwischen Applikation und Betriebssystem		

Im Gegensatz zu den APEX-Vorstellungen bei ARINC 653 setzt dieses APEX funktionell betrachtet wesentlich höher an. Es überlässt der Applikation nicht all den Aufwand der Fehlerverwaltung und Konsolidierung, sondern nimmt den Applikationen diese aufwendige Aufgabe ab. Es erlaubt so eine einfache, quasi Simplex-Implementation selbst von Software, deren Funktionalität letztlich hoch redundant abläuft.

Die Software einer IOM-Lane ist analog dazu organisiert (siehe Bild 6). Lediglich die Anwenderfunktionen unterscheiden sich von denen in den CPM. Im IOM sind es

- das Sensormanagement,
- das Aktuatormanagement,
- das Outputmanagement

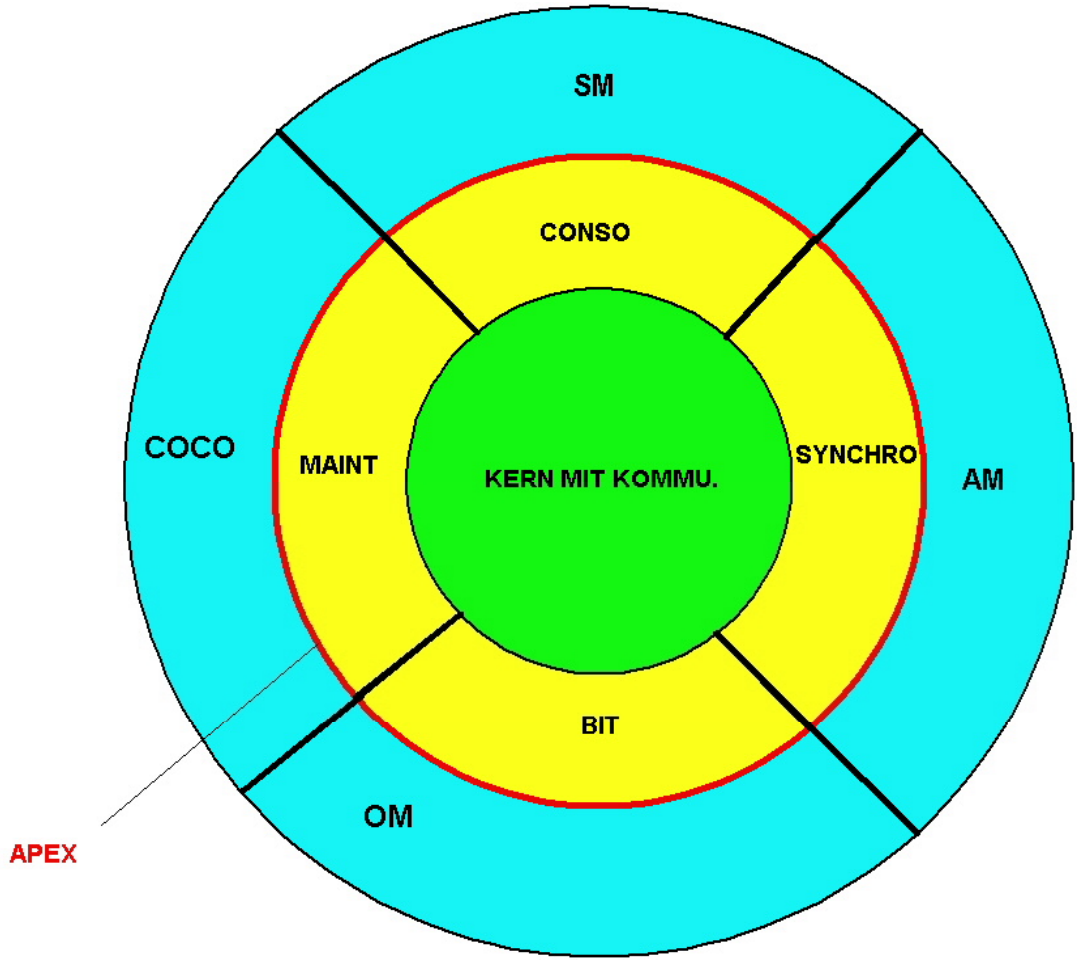


Bild 6 Schalenstruktur der Software im IOM

	SM ...Sensormanagement		AM ...Aktuatormanagement	OM	...Outputmanagement
	CONSO ...Konsolidierung		SYNCHRO ...Synchronisation	CoCo	...Configuration Control
	BIT …Built in Test		KOMMU ...Kommunikationsnetzwerk		
MAINT	...Maintenance	APEX	...Schnittstelle zwischen Applikation und Betriebssystem		

2.2 Aufbau der konfigurierbaren System- und Basissoftware

2.2.1 Entwicklung eines konfigurierbaren Betriebssystems für fehlertolerante Rechnersysteme (CORSYS)

Bei der Entwicklung eines konfigurierbaren Betriebssystems für fehlertolerante Rechnersysteme standen folgende Ziele im Fordergrund:

- Wiederverwendbarer Rechnerkern für redundante Rechnersysteme
- Skalierbar im Anwendungsraum (Redundanzebene, Ressourcenebene)
- Prinzipieller Aufbau eines Konfigurations-Tools in dem das System „Know-How“ verpackt ist
- Quell-Code muss zertifizierbar (DO-178B) sein

- Reduktion der Software-Erstellungskosten bei zukünftigen Anwendungen

2.2.1.1 Analyse, Definition und Implementierung von CORSYS

Die in Bild 7 dargestellten Funktionsgruppen
- Software-Kern mit Kommunikationsnetzwerk (KERN mit KOMMU.)
- Synchronisationsfunktionen (SYNCHRO)
- Konsolidierungsfunktionen (CONSO)

bilden die Basis für ein konfigurierbare Betriebssystem für fehlertolerante Rechnersysteme (CORSYS).

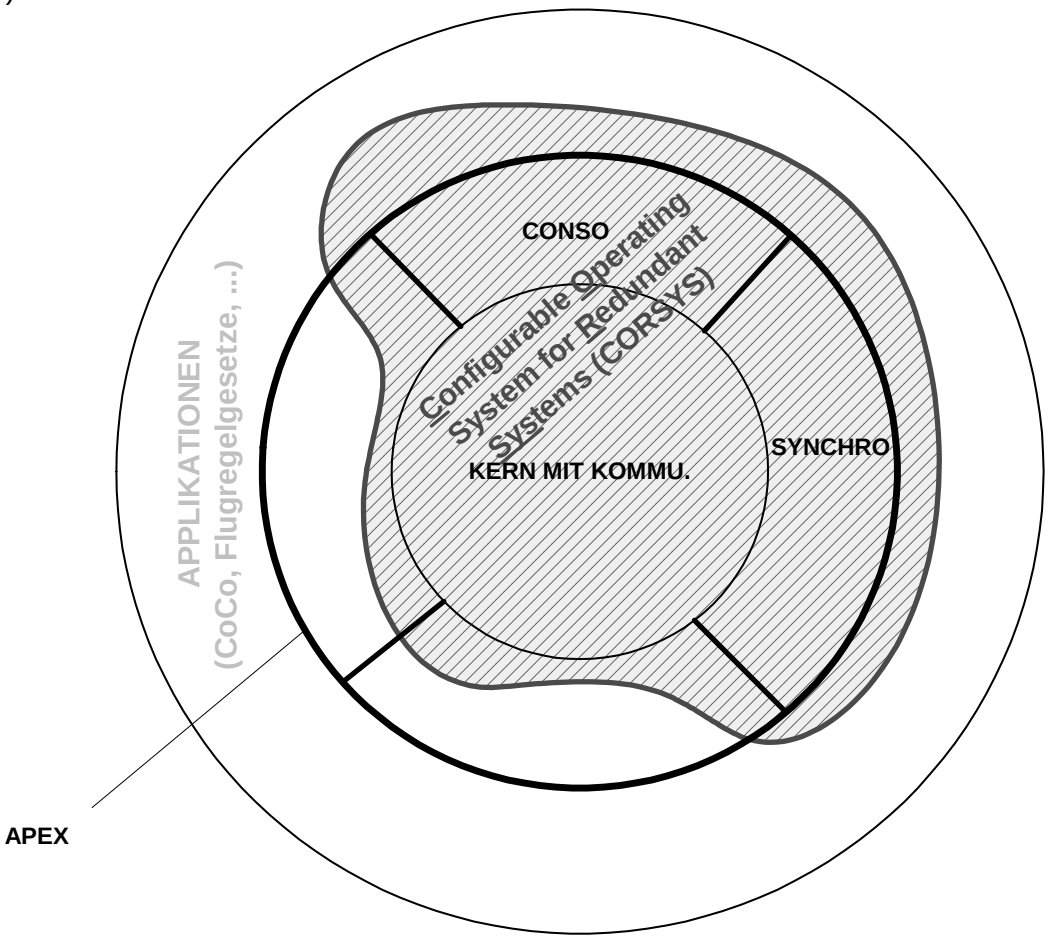


Bild 7 Funktionsgruppen von CORSYS

Zur Funktionalität der Funktionsgruppen im einzelnen:

1. Software-Kern mit Kommunikationsnetzwerk
   Um eine saubere Datenkonsistenz innerhalb eines Rechnersystems gewährleisten zu können, hat CORSYS folgende Eigenschaften
   i. Der Softwarekern enthält ein job-orientiertes Betriebssystem
   ii. Ein Rechnersystem mit zwei Lanes pro Module ist streng synchron

Job-orientiert bedeutet: jede Lane eines Module bearbeitet einen Job mit definierten Kommandierungs-Sequenzen innerhalb einer fest definierten Zeitdauer.

Tabelle 1veranschaulicht das prinzipielle Design von einem Job:

Job_Identifier	
time 1	action 1
time 2	action 2
•	•
•	•
•	•
time n	action n

Tabelle 1 Prinzipiell Design von einem Job

2. Synchronisationsfunktionen
   Die Synchronisationsanforderungen eines Rechnersystems mit einer Duplex Architektur wird hier am Beispiel einer Lane-Lane Synchronisation veranschaulicht:

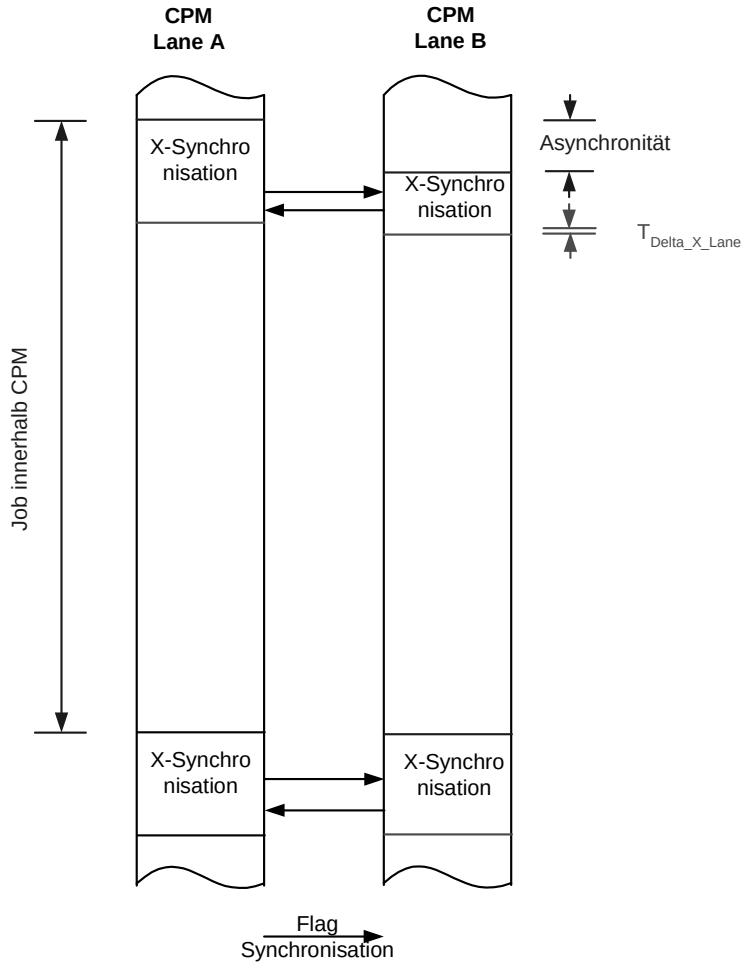


Bild 8 Lane-Lane Synchronisation

Das in Bild 8 dargestellte CPM-Module mit zwei Lanes bezeichnet man als synchron, wenn beide Lanes identische Jobs und dessen Funktionsaufrufe (actions) innerhalb einer Zeit $T_{Delta\_X\_Lane}$ durchführen.

3. Konsolidierungsfunktionen
   Damit Daten in einem Rechnersystem mit mehr als einer Lane als konsistent gelten, müssen zusätzlich zu den oben beschriebenen Maßnahmen noch Konsolidierungsmechanismen eingeführt werden. Diese Konsolidierungsfunktionen gewährleisten bei einem synchronen System, dass Applikationen und der Kern des Betriebssystems (z.B. ein CPM mit zwei Lanes) mit identische Eingangsdaten arbeiten.

Die Konfigurierbarkeit der oben aufgeführten Funktionsgruppen wurde folgendermaßen erreicht:

Jede Funktionsgruppe enthält Basisfunktionen, die abhängig von der Rechnerarchitektur aufgerufen werden. Die Systemanforderungen sind nicht in der Funktionsgruppe oder in den Basisfunktionen direkt implementiert, sondern in Konfigurationstabellen enthalten. Diese Konfigurationstabellen sind an Funktionsgruppen gekoppelt und bestimmen deren spezifiziertes Verhalten.

Die Datenkommunikation der Funktionsblöcke innerhalb des eigenen, als auch zu benachbarten externen Prozessoren erfolgt über ein Kommunikationsnetzwerk, das wiederum über Tabellen konfigurierbar ist. Das Kommunikationsnetzwerk ist nicht nur für die drei wesentlichen Funktionsblöcke von CORSYS (Kern des Betriebssystem, Synchronisation- und Konsolidierungsmechanismen) gedacht, sondern konfiguriert die Datenkommunikation des Gesamtsystems. D.h., das Kommunikationsnetzwerk bestimmt auch den Datenverkehr des Redundanzmanagement zum Beispiel zur APEX-Schnittstelle der Applikationen.

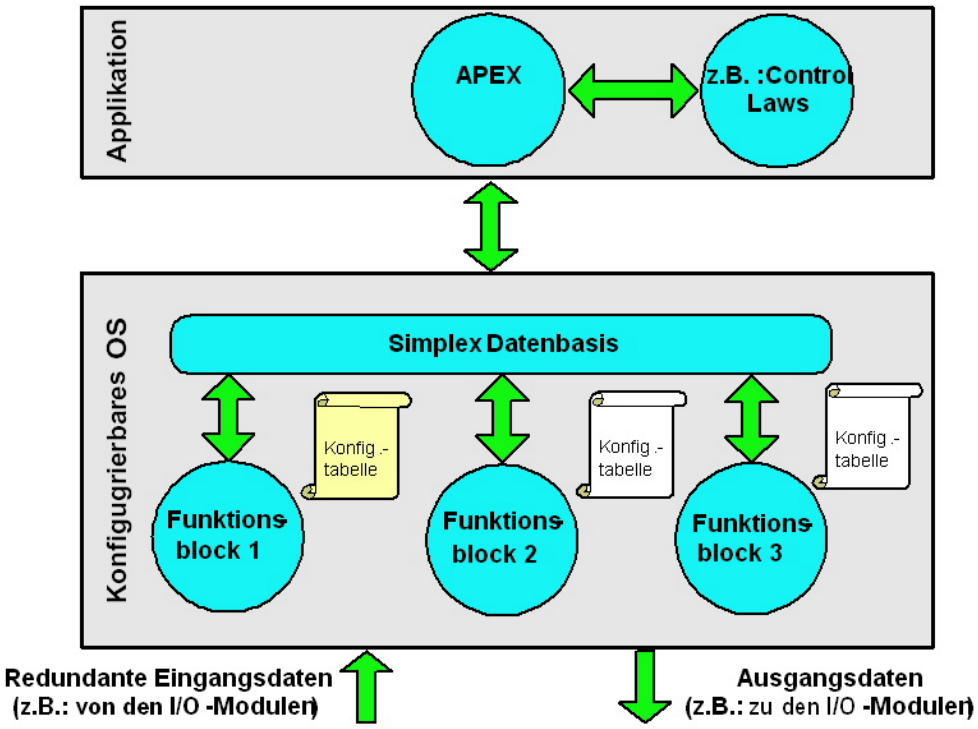


Bild 9 Schematische Darstellung von CORSYS

Ein wesentliches Charakteristikum der Trennung zwischen Redundanzmanagement- und Applikations-Prozessen ist das sogenannte „Simplex Minded Control Law Design“. Die Idee ist, über die APEX-Schnittstelle die Flugregelgesetze und das Redundanz-Management zu trennen. Die vom Redundanz-Management erzeugten Simplex Daten werden von der APEX Schnittstelle in ein für die Flugregelgesetze lesbares Format konvertiert. Der Designer der Flugregelgesetze kann somit die Regelgesetze so entwickeln, als würde es sich um eine Simplex-Rechnerarchitektur handeln. Die mit einem Auto-coding Tool generierte Applikationssoftware kann direkt in das Rechnersystem geladen werden.

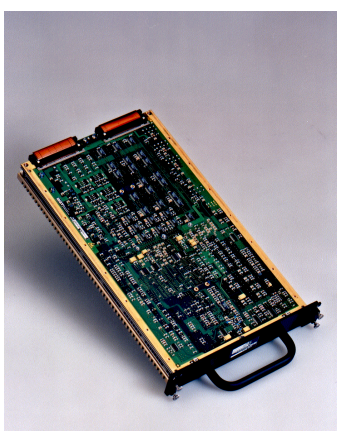
CORSYS wurde so spezifiziert und umgesetzt, dass es folgende Rechnerarchitekturen unterstützt:

- Simplex
- Duplex

Der Datenfluss für Triplex- und Quad.- Rechnerarchitekturen ist in CORSYS zwar schon berücksichtigt, deren Funktionsgruppen, Synchronisation und Konsolidierung aber nicht erweitert worden. Die Erweiterung der o.g. Funktionsgruppen war nicht Gegenstand des Forschungsvorhabens ProHMS ProHMS

2.2.1.2 Inbetriebnahme von CORSYS auf einer Duplex-Rechnerstruktur

Die Inbetriebnahme des konfigurierbaren Betriebssystems für fehlertolerante Rechnersysteme (CORSYS) erfolgte auf einer Duplex Rechnerarchitektur. Als Hardwareplattform diente das im Forschungsvorhaben EFCS entwickelte Primay Flight Module (PFM)



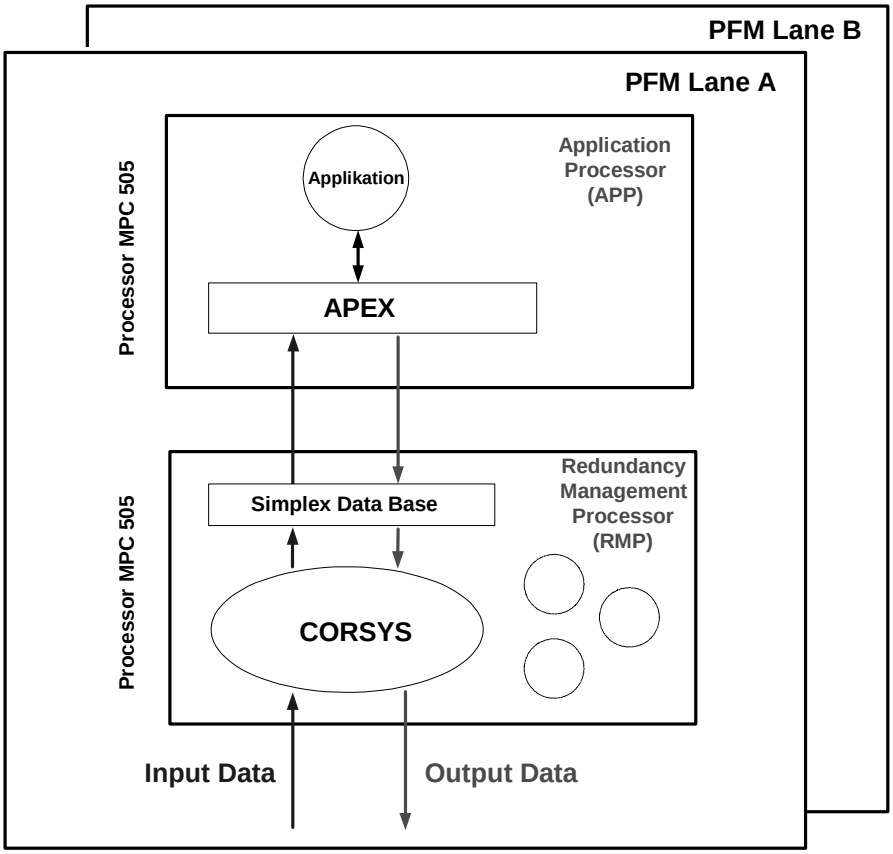


Bild 10 Inbetriebnahme von CORSYS

Bei der Inbetriebnahme von CORSYS ging es weniger um die Funktionalität der Applikation im Application Processor (APP), sondern vielmehr darum, die Funktionsgruppen inklusive der Basisfunktionen von CORSYS eingehend zu testen und den Datenfluss zwischen Redundanzmanagement und Applikation zu überprüfen.

2.2.2 Konzeption einer Toolkette für CORSYS

Um dem Systemdesigner eine zugeschnittene Bedienoberfläche (Toolumgebung/Toolkette) anzubieten, die den Umgang mit dem eigentlichen Werkzeug CORSYS sicher und effizient gestaltet, wurden verschiedene Ansatzpunkte identifiziert. Infolgedessen konnte eine entsprechende Toolkette für CORSYS mit folgenden Einzeltools konzipiert werden.

- Konfiguration des Kommunikationsnetzwerk eines Moduls und des Gesamtsystems
- Die Definition der Rechner-Modes (z.B. Passive Mode, Active Mode, etc...) und deren Transitionen

- Die Definition der Sequenzen von Funktionsaufrufen innerhalb eines Jobs
- Die Definition der Konsolidierungsvorschriften und -mechanismen für diskrete und analoge Daten
- Die Bestimmung des Synchronisationsverhaltens

Zur leichteren Handhabung der Toolkette ist eine Bedienoberfläche mit kontextbezogener Hilfestellung vorgesehen.

Die Konfiguration eines Systems erfolgt folgendermaßen:

i. Der Systemingenieur definiert das Systemverhalten in Form von Konfigurationsdaten und gibt sie in MS Excel Tabellen ein
ii. Diese Daten werden in ein ASCII lesbares Format übersetzt und in einer Datenbank verwaltet.
iii. Für jede Funktionsgruppe existieren Code-Templates in denen festgelegt wird, welche Informationen aus der Datenbank in den Source-Code einfließen.
iv. Mit den Konfigurationsdaten aus der Datenbank und den Code Templates erzeugt ein sogenannter Template Compiler den CORSYS Source-Code.
v. Dieser Source Code wird von dem prozessorabhängigen Target Compiler übersetzt.

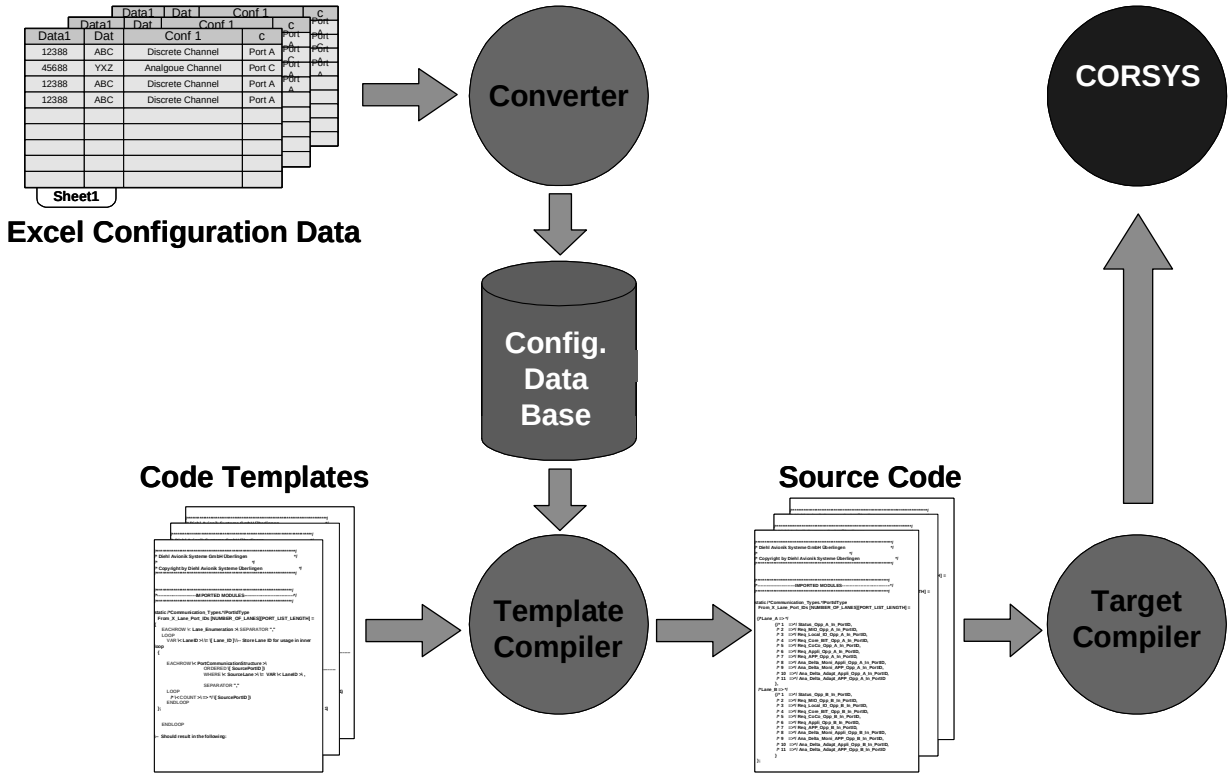


Bild 11 Prinzipieller Aufbau einer Toolkette

2.2.3 Machbarkeitsstudie zur Nutzung von SCADE für das CORSYS – System

Durch den Einsatz von Entwicklungswerkzeugen, die automatische Code-Generierung ermöglichen, können der Softwareentwicklungs- und verifikationsaufwand erheblich reduziert und spätere Änderungen sehr schnell realisiert werden, vorausgesetzt der Code Generator selbst wurde entsprechend dem Standard DO178B entwickelt. Nur in diesem Fall kann der durch das Tool generierte Code für den Steuerungsrechner mit minimalem Verifikationsaufwand direkt verwendet werden. Derzeit existiert auf dem Markt nur ein Tool, welches diese Anforderungen erfüllt. Dieses Tool wurde ursprünglich von Aerospatiale für den internen Gebrauch entwickelt, es wird mittlerweile aber kommerziell unter dem Namen SCADE vertrieben.
Das Tool SCADE erhält hierdurch eine zentrale Rolle bei der Anwendung in der Entwicklung von Applikationen (z.B. Control Laws) und dem schnittstellenübergreifenden Einsatz im Entwicklungsprozess bei Systemhersteller und Supplier.

Für die Evaluierung von SCADE wurden folgende Untersuchungen durchgeführt:

- Erprobung einzelner Prozessschritte aus der Prozessdefinition des optimierten Entwicklungsprozesses unter Verwendung der in SCADE spezifizierten Flap Load Relief Funktion
- Definition der Compiler Einstellungen für den qualifizierbaren Codegenerator
- Source Code Kompatibilitätschecks mit dem bei DAv bevorzugten Target Compiler
- Inspektionen des Source Codes auf Auffälligkeiten
- Laufzeitmessung der Flap Load Relief Funktion
- Laufzeitmessung zur manuell codierten Funktion aus dem kommerziellen SFCC Redesign Programm als Referenz
- Laufzeitvergleich und Bewertung
- Untersuchung „Copy Mem“ Funktion
- Anforderungen (Schnittstellenbeschreibung) an externe Library-Funktionen
- Änderungsanforderungen an die in EFCS erstellen Libraries, zur Einbindung in SCADE Library Funktionen
- Toolhandling (Erfahrung mit dem Tool bei der Evaluierung)
- Robustheit des Tools

Die oben aufgeführten Untersuchungen haben gezeigt, dass – von der Rechnerherstellerseite aus gesehen – der Einsatz von SCADE durchaus positiv zu bewerten ist. Die Integration des vom SCADE Code Generator erzeugten Source Codes verlief unproblematisch.
Lediglich der Vergleich der Laufzeitmessungen hat gezeigt, dass der manuell erzeugte Code deutlich effektiver als der von SCADE erzeugte Code ist.
Es ist zu vermerken, dass der durch SCADE automatisch erzeugte Source Code schwer lesbar ist.

2.3 Konfiguration und Aufbau des ProHMS Rechnersystems

2.3.1 Aufbau der Hardware und Inbetriebnahme

2.3.1.1 Kabinett

Das SFCU-Gehäuse besteht aus einem mechanischen strukturellen Rahmen (dem sogenannten Kabinett), welches auch schon in dem Forschungsvorhaben EFCS eingesetzt wurde. Durch das modulare Aufbaukonzept der SFCU befindet sich die Elektronik nicht in einer kompletten Elektronik-Einheit, sondern in mehreren sogenannten Line-Replaceable Modules (LRMs) und einer Backplane-Einheit. Jedes LRM besitz einen Rahmen, der zwei Platinen aufnehmen kann (Bild 14 zeigt ein LRM, die CPM Platine).

Der Rahmen der Module ist mit einem Schutzblech versehen. An der Rückseite der LRMs befinden sich die oberflächenkontaktierten Modulstecker.
Dieser Steckertyp hat folgende Vorteil:

- Er ermöglicht auf engem Raum viele Kontakte.
- Er besitzt geringe Übergangswiderstände.
- Beim Einstecken wie auch bei der Entnahme eines LRM entstehen keine hohen Kräfte.

Eine weitere Steckerebene innerhalb eines LRMs gibt es nicht.

Ein Kabinett enthält zwölf LRMs, die durch Führungsschienen in dem Kabinett fixiert und durch vier Schrauben verriegelt werden (siehe Bild 12).

An der Rückseite des Kabinetts ist die Backwall-Einheit inklusive Verteilerkarte untergebracht.
Die Verteilerkarte gewährleistet die Verbindung zwischen den Modulen (LRMs) und verteilt die ein- und ausgehenden Signale auf die verschiedenen Module.

Die Backwall-Einheit ist die zentrale Einheit zur Aufnahme von zwei Steckerebenen:

- Backplanestecker
- Modulstecker.

Das Kabinettgehäuse mit den eingesetzten LRMs ist so aufgebaut, dass die Luft durch den Kabinettrahmen und die Module fließen kann; d.h. die Module (LRMs) führen ihre Wärme sowohl über die Struktur an den Kabinettrahmen, als auch direkt an die sie umfließende Luft ab. Das Kabinettgehäuse ist an der Ober- und Unterseite durchlässig für Kühlluft.

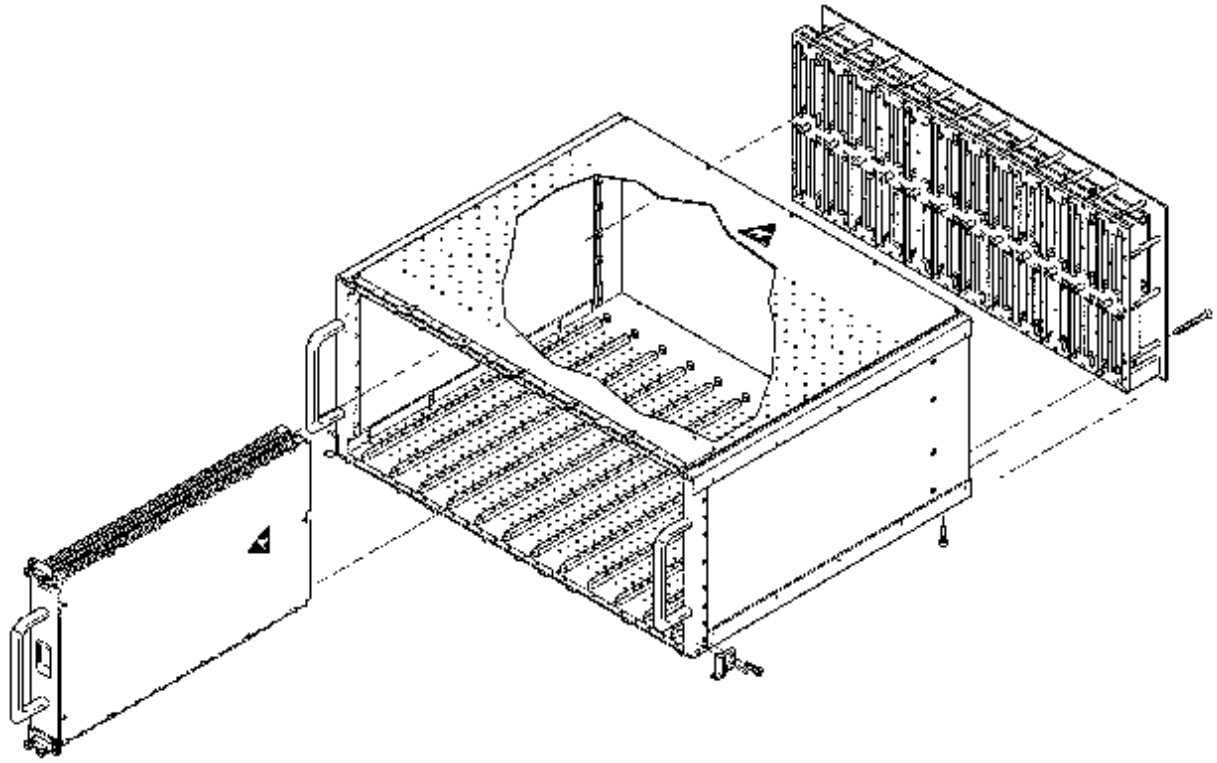


Bild 12 Schematischer Aufbau eines Kabinetts mit Backwall-Einheit und LRM

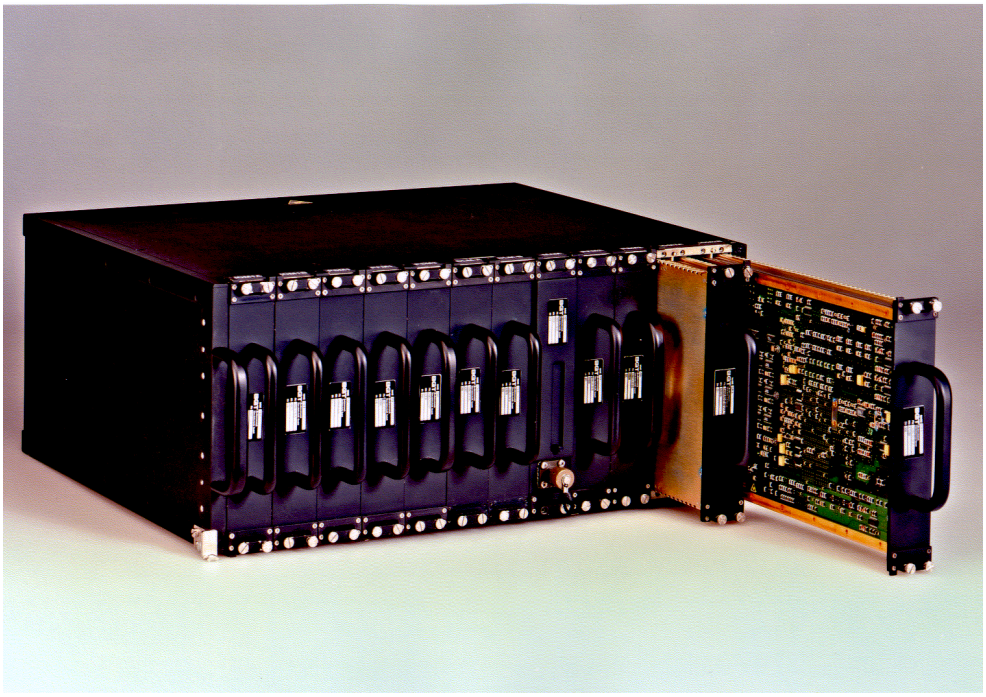


Bild 13 Ansicht einer SFCU von der Frontseite mit herausgezogenen LRMs

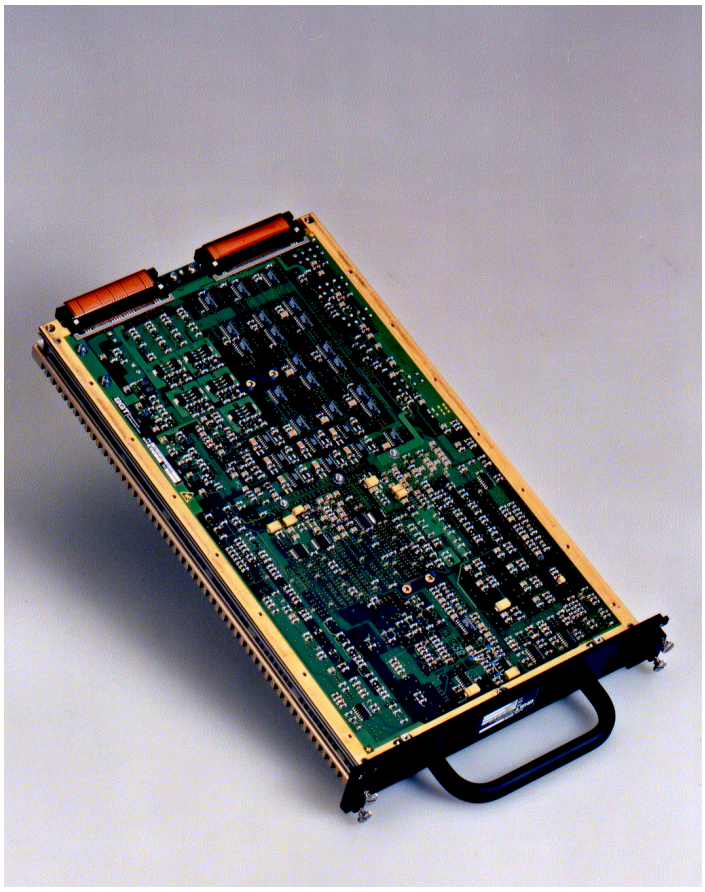


Bild 14 Ansicht einer Computing Module (CPM) Platine

Ein Kabinett enthält folgende LRMs:

- die beiden Computing Module: CPM1, CPM2
- die drei Input/Output Modules: IOM1 Lane-A, IOM2 Lane-A, IOM3 Lane A
- die drei Input/Output Modules: IOM1 Lane-B, IOM2 Lane-B, IOM3 Lane B
- dem Maintenance Module (MAM)
- und den drei baugleichen Power Supply Modulen: PSM1, PSM2, PSM3

Tabelle 2 zeigt die Anzahl der Platinen pro LRM

LRM	Lane	Anzahl Platinen
CPM1	A	1
	B	1
IOM1	A	2
	B	2
IOM2	A	2
	B	2
IOM3	A	2
	B	2
MAM		2
PSM1	A	1
	B	1
PSM2	A	1
	B	1
PSM3	A	1
	B	1

Tabelle 2 Anzahl der Platinen pro LRM

2.3.1.2 Modulbeschreibung

2.3.1.2.1 Computing Modul (CPM)

Das in EFCS entwickelte Computing Modul setzt sich aus zwei CPM-Lanes (unabhängige „Kanäle“) zusammen: Lane-A und Lane-B.
Jede CPM-Lane besteht aus zwei autonomen 32-Bit Computing-Einheiten,

1. Application Proccesing Einheit (APP)
2. Redundancy-Management Processing Einheit (RMP)

die über einen gemeinsamen Speicherbaustein (Dual-Port-RAM) miteinander kommunizieren. Die erste Computing-Einheit wird zur Berechnung der Hight-Lift-Control-Laws benutzt. Die zweite Computing-Einheit wird zur Redundanzverwaltung und zur Kommunikation mit der CPM Lane B, den I/O-Modulen und MAM eingesetzt.

2.3.1.2.2 Input/Output Module

Das in der SFCU benutzte Input/ Output Funktionsmodul setzt sich aus zwei LRM zusammen (z.B. IOM1 Lane- A und IOM1 Lane- B).Jedes dieser LRMs setzt sich ebenfalls aus zwei Platinen zusammen. Das LRM für die Lane- A besteht aus einer IOA- Karte und einer IO-Karteanaloges gilt für das Lane-B LRMDie IOA- bzw. Die IOB-Karte enthält als wesentliche

Einheiten den Prozessor, ARINC- Schnittstelle, SFCU interne Kommunikationsschnittstelle und das Cross- Interface zur Ansteuerung der IO-Karte. Auf der IO-Karte sind die Input- und Output-Funktionen zur SFCU-Peripherie untergebracht. Diese in dem Forschungsvorhaben EFCS entwickelten Karten wurden für die SFCU modifiziert bzw. neuentwickelt.
Die wesentliche Änderung dieser Karten bestand in der Modifikation des „Programmable Array Logic „ Bausteins.

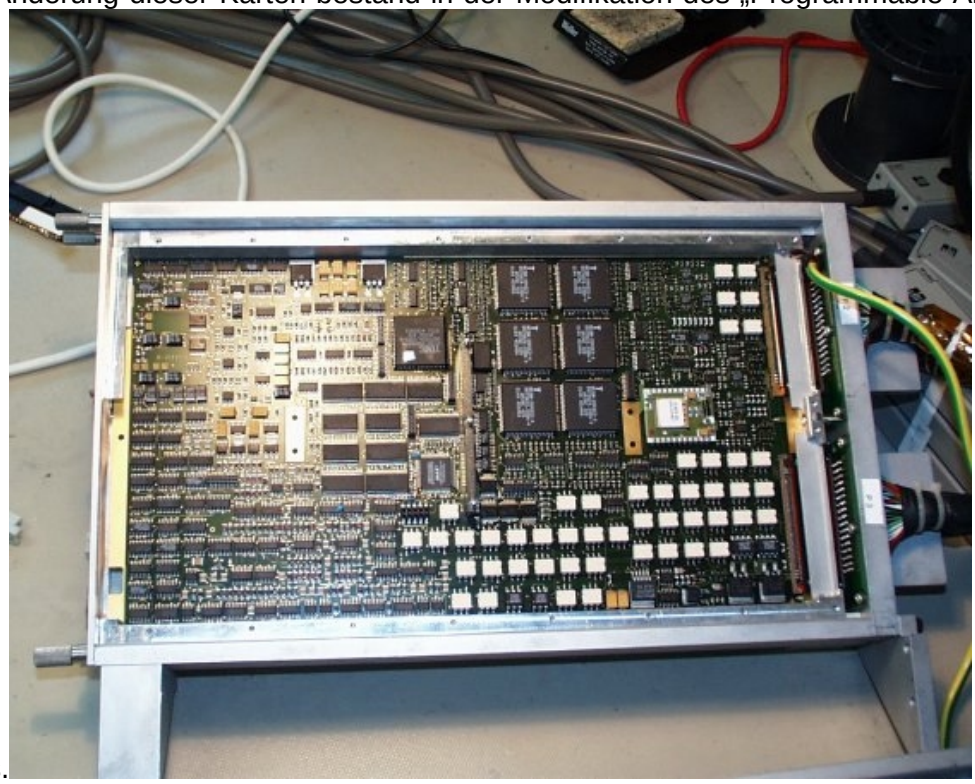


Bild 15 ProHMS IO-Karte

Für die Ansteuerung der in Bild 2 dargestellten Stellflächen und das Einlesen der Sensorwerte war eine Neuentwicklung der IO-Karte notwendig. Aufgrund der hohen Anforderungen bezüglich Actuator-, Sensor- und Output-Management ergab sich eine Funktionsdichte dieser Karte mit

- über 8000 Pins
- über 6000 Verbindungen

und über 2400 Bauteile

2.3.1.2.3 Maintenance Module (MAM)

Das MAM verbindet das Testsystem mit der SFCU. Es ermöglicht damit den schreibenden und lesenden Zugriff auf SFCU interne Daten.
Der HW-Aufbau des Maintenance Modules entspricht weitgehend der RMP Einheit des CPM.

2.3.1.2.4 Power Supply Module

Die aus dem Forschungsvorhaben EFCS übernommene Power Supply Modul (PSM) setzt sich aus zwei identischen Power Supply Units (PSU) zusammensetzt.
Jede PSU enthält primärseitig zwei Konsolidierungsschaltungen für jeweils zwei 28V-DC Versorgungen. Durch diese Aufteilung werden die Maximalströme pro Konsolidierungszweig reduziert.
Eine Konsolidierungsschaltung versorgt zum Beispiel den DC/DC- Wandler der PSU
Eine Monitorschaltung überwacht die Primärspannungen. Bei Unterspannung wird eine Kondensatorstützung zugeschaltet, die den DC/DC- Wandler über einen definierten Zeitraum (Tranparency Time) versorgt. Die externen Lasten werden nicht gestützt.
Sekundärseitig versorgt eine PSU je eine Lane eines IOM und eines CPM. Diese Ausgänge sind gegen Überlastung geschützt.
Eine weitere Monitorschaltung überwacht die Sekundärseite des DC/DC- Wandlers. Bei einem Power-Interrupt werden in Abhängigkeit der Zeitdauer der Spannungsunterbrechung Statussignale gesetzt.

2.3.1.2.5 Backwall-Einheit

Die für die SFCU neu entwickelte Backwall Einheit besteht aus einer Verteilerkarte, die in Multilayer-Technologie aufbebaut ist. Diese hat ausschließlich die Aufgabe, die notwendigen elektrischen Verbindungen zwischen den Modulen untereinander herzustellen. Die Karte enthält keinerlei Bauteile. Die Verbindungen zu den Modulen erfolgt über oberflächenkontaktierende Stecker.
Bild 16 zeigt die in der SFCU benutzte Verteilerkarte.

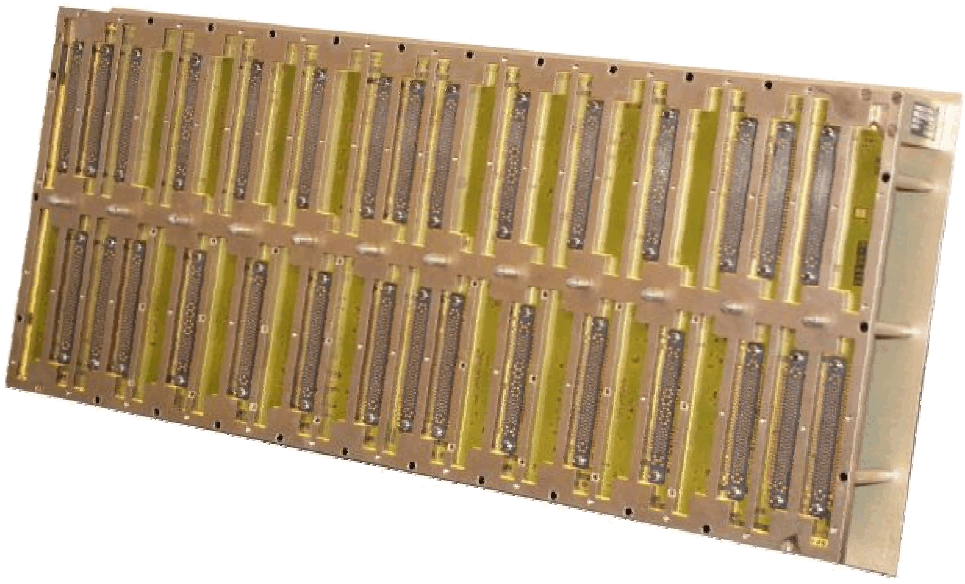


Bild 16 SFCU Verteilerkarte (Backwall-Einheit)

2.3.1.2.6 Inbetriebnahme der Hardware

Für die Inbetriebnahme der neu entwickelten Platinen wurden eine Testvorrichtung und eine Prüfspezifikation erstellt.
Die Ergebnisse der Erstinbetriebnahme eines jeden Platinentyps wurden in einem Prüfprotokoll festgehalten.
Die Tabelle 3 zeigt einen Auszug aus der Prüfspezifikation der IO-Karte:

Interface	Bemerkung	Seite in	Schalter	Funktion LED	Überstrom-Abschaltwelle Sollwert in A	Überstrom-Abschaltwelle In A	OK?
DO1N_1	Überstrom: 7Ohm var. + 10 Ohm in Serie Schalter auf KS	55	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,44	ok
DO1N_2	Überstrom: 7Ohm var. + 10 Ohm in Serie Schalter auf KS	56	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,44	ok
DO3LS_1	Überstrom: 14 Ohm var. in Serie Schalter auf KS	63	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,48	ok
DO3LS_2	Überstrom: 14 Ohm var. in Serie Schalter auf KS	64	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,46	ok
DO3LS_3	Überstrom: 14 Ohm var. in Serie Schalter auf KS	65	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,47	ok
DO3LS_4	Überstrom: 14 Ohm var. in Serie Schalter auf KS	66	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,46	ok
DO3LS_5	Überstrom: 14 Ohm var. in Serie Schalter auf KS	67	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,49	ok
DO3LS_6	Überstrom: 14 Ohm var. in Serie Schalter auf KS	68	Normal Überstrom Overcurrent Reset Reset	An Aus An Aus	0,441...0,539	0,47	ok

Tabelle 3 Auszug aus der IO-Karten Prüfspezifikation

Die Integration der HW zu einem Kabinett und der HW Test dieses Kabinetts wurden in der HW-SW Integration des Kabinetts durchgeführt.

2.3.2 Implementierung und Inbetriebnahme der Software

Der prinzipielle Aufbau der CPM- und IOM-Software ist in Kapitel 2.1.2 dargestellt. Um CORSYS für das Redundanzmanagement einzusetzen, war eine Erweiterung der Konsolidierungsmechanismen notwendig. Somit war es möglich, die Vorteile von CORSYS bei der Software Entwicklung der SFCU direkt zu nutzen.
Die High-Lift Regelgesetze wurden von Airbus Deutschland mit dem Autocoding-Tool SCADE spezifiziert und von DAv in die SFCU integriert.
Die Spezifikation und Implementierung der hardwarenahen Softwarefunktionen „Sensormanagement“, „Aktuatormanagement“ und „Outputmanagement“ erfolgte ohne Tools.

Um die Funktionalität der SFCU zu überprüfen wurden intensive Engineering-Tests bezüglich HW-SW-Integration und Systemintegration durchgeführt.

Hardware-Software-Integration:
Die Hardware-Software-Integration dient zum Nachweis, dass die auf die Ziel-Hardware heruntergeladene Software korrekt funktioniert. Die Engineering Tests bezüglich Hardware-Software-Integration wurden am DAv Teststand (siehe Bild 17 und Bild 18) durchgeführt.

Systemintegration:
Die Systemtests weisen die korrekte Funktion des Gesamtsystems gegenüber den vorgegebenen Anforderungen nach. Die Engineering Tests bezüglich Systemintegration wurden am DAv Teststand (siehe Bild 17 und Bild 18) durchgeführt.

Für die oben aufgeführten Tests war ein Redesign des Testsystems und Inbetriebnahme der Testsystem Hardware notwendig.

Bild 17 ProHMS Testsystem

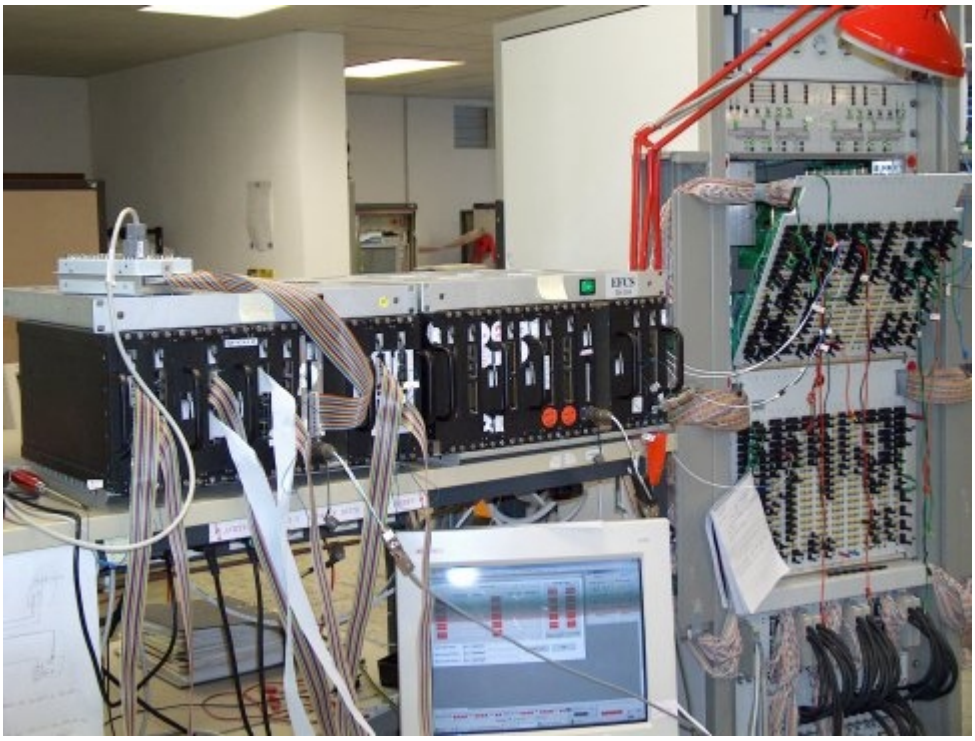


Bild 18 SFCUs am Testsystem bei DAv

2.3.3 Integration und Test der Regelgesetze

Die von Airbus Deutschland mit SCADE spezifizierten High-Lift Regelgesetze wurden von DAvk Systeme in die SFCU integriert.

Die Integration der Regelgesetzte erfolgte wie folgt:
Die Softwarestruktur der High-Lift-Regelgesetze ist streng hierarchisch und modular aufgebaut. Um die Funktionalität dieser Regelgesetze zu testen, wurden von Airbus Deutschland Testvektoren erstellt. Diese Testvektoren enthalten Angaben über Eingangs- und Ausgangswerte für die Regelgesetze. Jeder Satz von Eingangs- und Ausgangsdaten wird als Testvektor klassifiziert und ist in einem Datenfile enthalten. Die Testvektoren sind so aufgebaut, dass möglichst jeder Pfad der Software überprüft wird.
Die High-Lift Regelgesetze in der SFCU werden mit den Soll-Eingangsdaten der Testvektoren gespeist. Die Ergebnisse oder Ist-Ausgangswerte wurden mit den Soll-Ausgangswerten der Testvektoren verglichen.

2.4 Erprobungsassistenz

Nach Fertigstellung des ProHMS Rechnersystems erfolgte die Auslieferung der SFCUs an Airbus Deutschland / Bremen.

Die Erprobungsassistenz hatte folgenden Umfang:

- Support bei der Integration des Rechnersystems (SFCUs) in das Testrig bei Airbus Deutschland / Bremen.
- Support bei der Visualisierung von SFCU internen Ein- und Ausgangsdaten.

2.5 Konzeption eines Smart Lever

Stand der Technik in der A330/340 ist eine Command Sensing Unit, die die Stellung des Bedienhebels der Vorflügel und Klappen mit optischen Mitteln (Lichtschranke) erfasst und an einen externen Rechner, den Slat Flap Control Computer (SFCC) zur Signalverarbeitung weitergibt. Ziel dieses Arbeitspaketes war es, ein Konzept für einen Bedienhebel mit integrierten Abgriffen und Signalaufbereitung zu erarbeiten.

2.5.1 Vorgehensweise

Das Konzept sieht eine Verlagerung der Signalauswertung der CSU-Sensoren vom Slat Flap Control Computers (SFCC) in die Command Sensor Unit (CSU) vor. Die Signalverarbeitung, die bisher im SFCC Rechner erfolgte, wird in die im Bild 19 dargestellten Prozessoren (µP) verlagert (d.h., die CSU ist smart). Die dazu erforderlichen Sensorsignale zur Messung der Slat/Flap-Hebelpositionen erfolgt über entsprechende Sensoren. Für die Anbindung der CSU Prozessoren an SFCC1 und SFCC2 ist ein Bussystem vorgesehen.

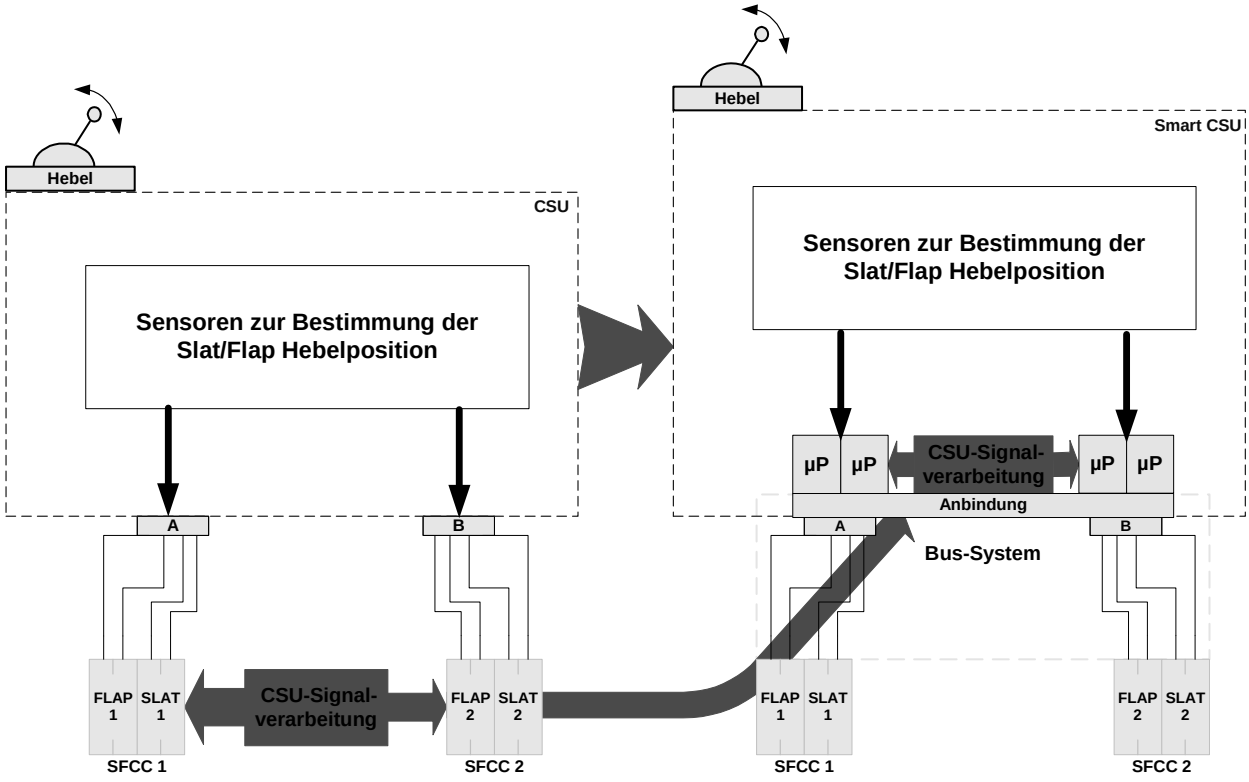


Bild 19 Von der konventionellen CSU zur „smarten“ CSU

2.5.2 Analyse

Bei der Analyse zur Konzeption eines „smart“ Levers standen folgende Lösungsaspekte im Vordergrund:

- Bussystem
  Durch die Verwendung von Mikroprozessoren in der CSU bietet sich ein Bus-System für die Anbindung der CSU an den SFCC an. Es eigenen sich dafür neben dem in der Luftfahrt einführten ARINC429-Bus der im Automobilbereich weitverbreitete CAN-Bus.
- Sensor- und Prozessorarchitektur
  Um die geforderten Zuverlässigkeits- und Verfügbarkeitsanforderungen zu erfüllen ist eine Analyse bezüglich der Anzahl der Sensoren und deren Anbindung an ein Prozessornetzwerk in der CSU durchzuführen.

Sensorarchitektur:
Die Analyse ergab, dass es im wesentlichen zwei Varianten für die Anbindung der Sensoren an die Prozessoren in der CSU gibt.

1. Die erste Variante geht von einer Anbindung von zwei CSU-Sensoren pro Prozessor aus. Diese Möglichkeit toleriert aber keine fehlerhaften CSU-Sensoren ohne Einbuße der Verfügbarkeit.
2. Bei einer Sensoranbindung von mehr als drei Sensoren pro Prozessor ist zu beachten, dass Voting Mechanismen eingesetzt werden müssen, um einen defekten Sensor zu ermitteln. Aus Symmetriegründen ist aber immer nur eine gerade Anzahl von Sensoren pro Prozessor zulässig. Da in dem hier angesprochenen Fall aber zwei Signale miteinander verglichen werden, darf bei vier Sensoren (je zwei von verschiedenen Einheiten stammende Positionssignale) kein Sensor ausfallen. Damit ergibt sich eine minimale Anzahl von sechs Sensoren pro Prozessor. Dieser Ansatz scheitert aber daran, das es keinen bekannten Mikrocontroller mit der dafür benötigten Anzahl von A/D-Wandler gibt.

Prozessorarchitektur:
Bei der Prozessorarchitektur geht die Konzeptanalyse von einer Acht- und einer Vier-Prozessor-Architektur aus. Eine Zwei-Prozessorlösung fällt aus Zuverlässigkeits- und Verfügbarkeitsgründen aus. Eine Drei- bzw. Sechs-Prozessorlösung entfällt aus Symmetriegründen, da jeder SFCC-Channel immer die gleiche Anzahl von Prozessoren benötigt.
Die Acht-Prozessor-Architektur bietet den Vorteil, dass bei einem Prozessorausfall nur ein SFCC-Channel betroffen ist. Der Ausfall eines Sensors betrifft zwei CSU-Prozessoren und führt damit zum Verlust von zwei SFCC-Channels.

Die Vier-Prozessor-Architektur spart Kosten durch weniger Hardware. Im Gegensatz zur Acht-Prozessorarchitektur ist bei einem CSU-Sensorausfall auch nur ein Prozessor in der CSU betroffen. Aber dieses Szenario führt zum Ausfall von zwei SFCC-Channels, einem Slat- und einem Flap-Channel.

Die Konzeptanalyse ergab zwar, dass es prinzipiell möglich ist, die Signalverarbeitung vom SFCC in die CSU zu verlagern, aber ein wirklicher Vorteil gegenüber der bisherigen Lösung war nicht erkennbar. Die mit vertretbarem Aufwand in der CSU realisierbare Vier-Prozessor-Lösung erreicht nicht die geforderte Verfügbarkeit. Bei einem CSU-Prozessorausfall gehen immer zwei SFCC-Channels gleichzeitig verloren. Beim bisherigen Konzept (ohne „smarte“ CSUs) hingegen geht beim Ausfall eines CSU-Sensors nur ein SFCC-Channel verloren.

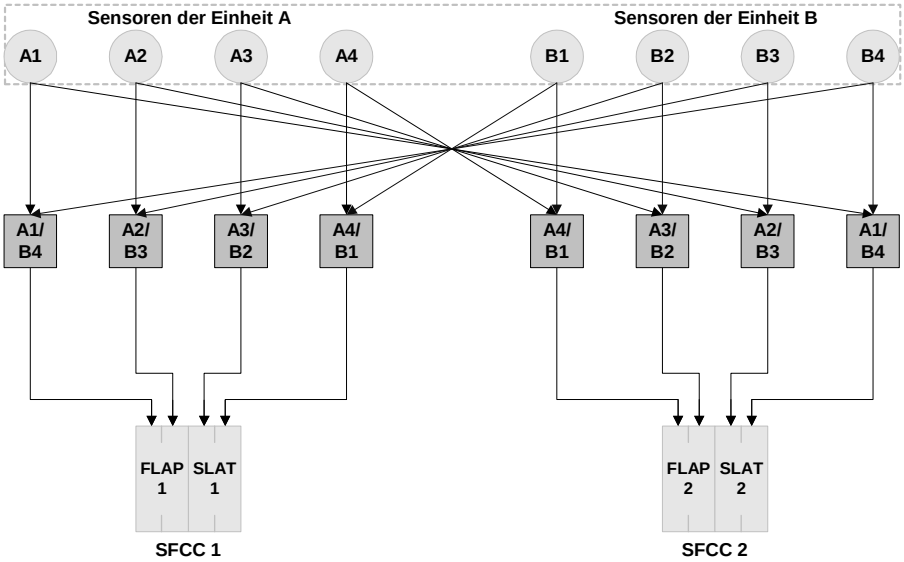


Bild 20 Acht Prozessor zwei Sensoren CSU Architektur

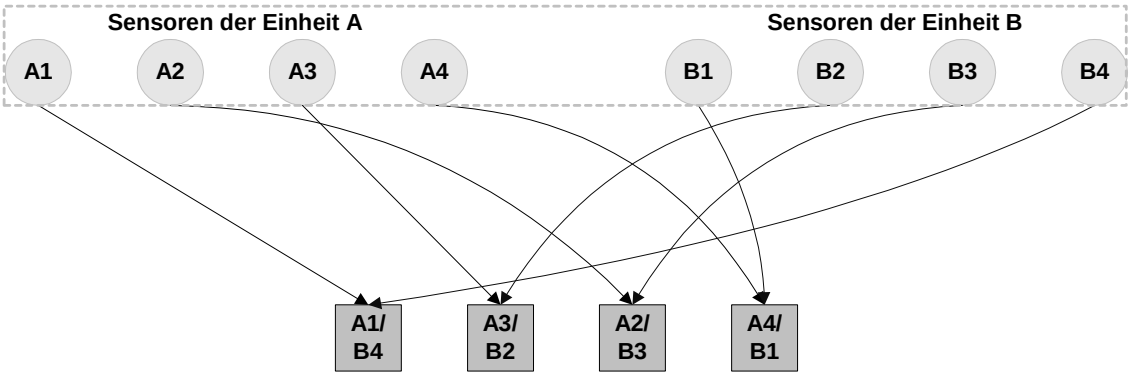


Bild 21 Vier Prozessor zwei Sensoren CSU Architektur

2.6 Voraussichtliche Nutzung der Ergebnisse

Die erarbeiteten erweiterten Funktionsmöglichkeiten für den Anwender unterstützen einen wirtschaftlichen Einsatz in zukünftigen operativen Airbus Projekten.

2.7 Erreichte Vorhabensziele

Die im Antrag angestrebten Ziele dieses Vorhabens wurden erreicht. Die einzelnen Arbeitspakete sind in der u.a. Tabelle dargelegt

	Arbeitspakete	Stand zum Abschluss des Vorhabens
AP 1	Definition der Rechnerarchitektur	• abgeschlossen (erfolgreich)
AP 2	Entwicklung der konfigurierbaren System- und Basissoftware	
AP 2.1	Entwicklung von CORSYS	• abgeschlossen (erfolgreich)
AP 2.2	Konzeption einer Toolkette von CORSYS	• abgeschlossen (erfolgreich)
AP 2.3	Machbarkeitstudie zur Nutzung von SCADE	• abgeschlossen (erfolgreich)
AP 3	Konfiguration und Aufbau der SFCU	
AP 3.1	Analyse, Konfiguration und Aufbau des SFCU Rechnersystems	• abgeschlossen (erfolgreich)
AP 3.2	Modifikation vorhandener Testsysteme	• abgeschlossen (erfolgreich)
AP 4	Konzeption eines Smart Lever	• abgeschlossen (erfolgreich)
AP 5	Erprobungsassistenz	
AP 5.1	Integration in die Bodenprüfeinrichtung	• abgeschlossen (erfolgreich)
AP 5.2	Assistenz bei den Tests	• entfallen

Tabelle 4 Vorhabensziele

2.8 Fortschritte auf dem Gebiet des Vorhabens bei anderen Stellen

Es sind uns keine bekannt.

2.9 Veröffentlichungen

Entwurf eines Hochauftriebssystems zur Ansteuerung multifunktionaler Stellflächen
Bremen, 2002

3 ANLAGEN

3.1 Zusammenfassung der verwendeten Fachliteratur

keine

3.2 Verzeichnis der Veröffentlichungen, in denen Ergebnisse aus dem Projekt verwendet wurden

/ 1 / RTCA / DO-178B / ED-12B
Software Considerations in Airborne Systems and Equipment Certification

/ 2 / Software Management And Development Plan (SMDP)
DAv Doc.No. D0500017

/ 3 / Software Verification Plan (SVP)
DAv Doc.No. D0500015

/ 4 / SFCC-R Project Handbook (PHB)
DAv Doc.No. D0500020

/ 5 / Software Configuration Management Plan (SCMP)
DAv Doc.No. D0500019

/ 5a / Software Configuration Management Standards (SCMS)
DAv Doc.No. D0500018

/ 6 / Software Quality Assurance Plan (SQAP)
DAv Doc.No. D0500021

/ 7 / A330/A340 Certification Review Item CRI S-1019, Issue 2 (Draft), 10.2.2000

/ 8 / ABD 100.2.4 Issue C
Airbus Directives Equipment-Design-General Requirements For Suppliers

/ 9 / „A330/340 Slat Flap Electrical System 2750M1S 020100 Issue 9“ Rev.1,
Date: 23.07.1999

/ 10 / „A330/340 Slat Flap Electrical/Mechanical Interface Data, 2750 M1I 020100 Issue 9“,
Date: 5.03.1999

/ 11 / „A330/340 Slat Flap Electrical System, Interface To CMS“, 2750 M1S 020100 Issue 9
Appendix 1/2“,
Date: 29.01.1999

/ 12 / ABD 100.1.10 Issue C
Airbus Directives Equipment-Design-General Requirements
For Suppliers

/ 13 / SFCC Specification
2750 M 1S 020 100, Issue 9 Rev. 2 from 26.01.2000

/ 14/ Software Accomplishment Summary
DAv Doc.No. D0500146

/ 15/ Safety Analysis Report
DI-074, Issue 4

4 ABKÜRZUNGEN

A/D	Analog/Digital
APEX	Schnittstelle zwischen Applikation und Betriebssystem
AM	Actuator Management
APP	Application Processor
ARINC	Bussystem
BIT	Build In Test
BMWA	Bundesministerium für Wirtschaft und Arbeit
CAN	Bussystem
COCO	Configuration Control
CONSO	Consolidation
CORSYS	Configurable Operating System for Redundant Systems
CPM	Computing Module
CSU	Command Sensor Unit
DAv	Diehl Avionik Systeme
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DO-178B	Software Considerations in Airborne Systems and Equipment Certification
EFCS	Electronic Flight Control System
FC	Flight Control
HW	Hardware
IO	Input Output
IOM	Input Output Module
KOMMU.	Kommunikationsnetzwerk
LLI	Liebherr Lindenberg Aerospace
LRM	Line Replaceable Module
MAINT	Maintenance
MAM	Maintenance Module
MS	Microsoft
OM	Output Management
PFM	Primary Flight Module
ProHMS	Prozesskette Hochauftrieb mit multifunktionalen Steuerflächen
PSM	Power Supply Module
PSU	Power Supply Unit
RMP	Redundancy Management Processor
SCADE	Safety Critical Applications Development Enviroment
SFCC	Slat Falp Control Computer
SFCU	Slat Flap Control Unit
SFCS	Slat Flap Control System
SM	Sensor Management
SW	Software
SYNCHRO	Synchronisierung