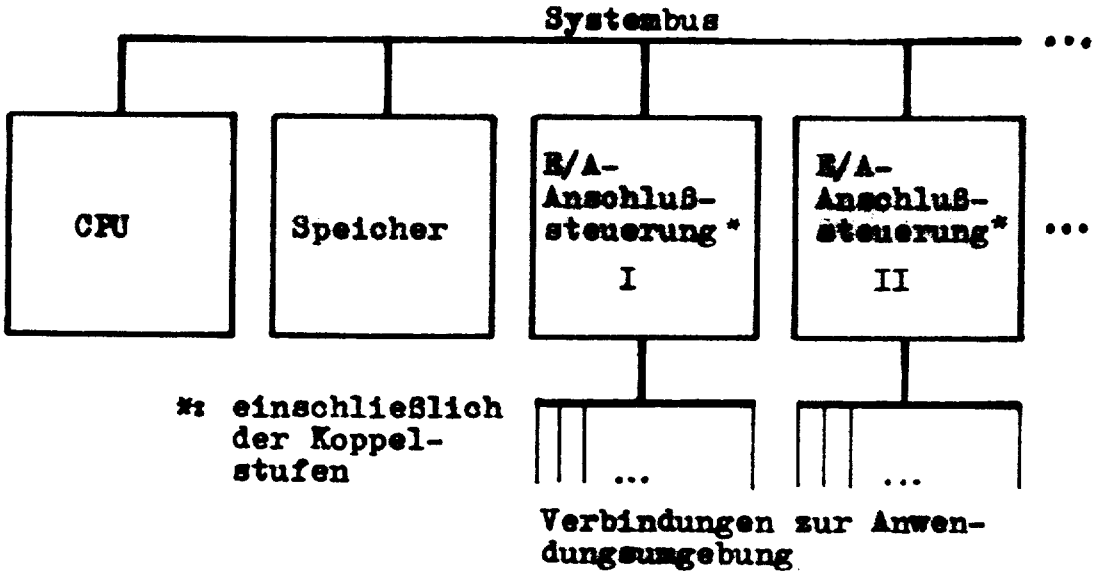
# System.015: eine Universalrechnerarchitektur für komplexe industrielle Steuerungssysteme

Wolfgang Matthes

## 1. Anwendungsziele

Das System.015 (im folgenden: S.015) repräsentiert ein einheitliches Architekturkonzept für eine Familie von Universalrechnern im Leistungsbereich vom Mikrocontroller bis zum Hochleistungsrechner. Wesentliche Anwendungsgebiete sind Steuerungen aller Art, Zusatzeinrichtungen (Acceleratoren) für herkömmliche Systeme sowie komplexe Informationsverarbeitungssysteme, vorzugsweise solche der Prozeßsteuerung. Der Einsatzbereich umfaßt dabei sowohl die unmittelbare Prozeßkopplung, d. h. die schnelle und flexible Steuerung vielfältiger prozeßspezifischer Interfaces, als auch übergeordnete verarbeitungsintensive Koordinierungs-, Entscheidungs-, Optimierungs- und Präsentationsaufgaben.
Die Schaltungsstrukturen des S.015 sollen in ASIC-Zellenbibliotheken bereitgestellt werden, so daß alle Hardwarekomplexe gemäß dem jeweiligen Einsatzfall über reguläre Interfaces zusammengeschaltet werden können.
Bild 1 veranschaulicht das wichtigste Anwendungsziel des S.015 aus der Sicht des Praktikers: herkömmliche rechnergestützte Steuerungssysteme des oberen Leistungsbereichs sind aus mehreren Leiterplatten zusammengesetzt, die über Bussysteme un-

a) Herkömmliche Systemstruktur




b) Systemstruktur mit S.015

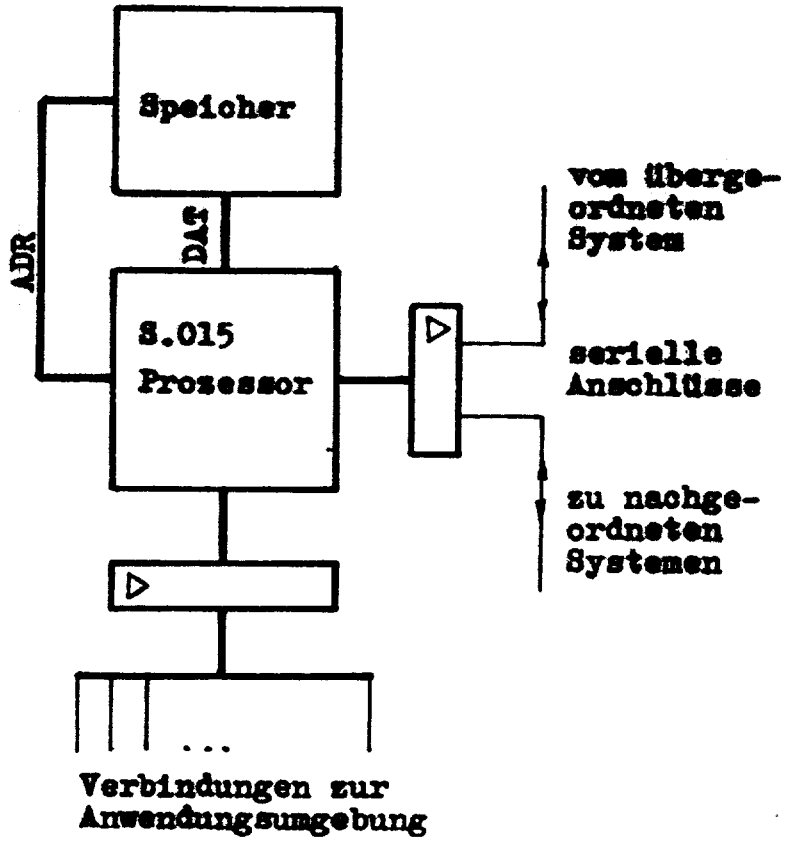



Bild 1 Anwendungsziel des S.015

tereinander verbunden sind. Solche Konfigurationen sollen durch einzelne S.015-Prozessoren ersetzt werden. In den meisten Einsatzfällen sollen anwendungsspezifische Verknüpfungsschaltungen völlig entbehrlich sein. Vielmehr sollen neben dem programmierbaren S.015-ASIC und seiner Speicherausstattung nur noch die unumgänglich notwendigen Schaltmittel zur Anpassung an die Anwendungsumgebung erforderlich sein (wie z. B. Pegelwandler, Optokoppler usw.). Zum Aufbau komplexer Steuerungssysteme sind serielle Verbindungen zwischen den einzelnen Baugruppen vorgesehen. In den übergeordneten Ebenen können sowohl herkömmliche Rechner als auch solche auf Grundlage der S.015-Architektur vorgesehen werden. Um Systeme dieses Leistungsbereichs implementieren zu können, enthält die S.015-Architektur Vorkehrungen für den Betrieb "dicht gekoppelter" Multiprozessorkonfigurationen und für die Parallelausführung mehrerer Befehle im Einzelprozessor.

## 2. Entwicklungsziele

S.015-Komplexe müssen technologisch vergleichbaren Systemen, die auf herkömmlichen Architekturen beruhen, im absoluten Leistungsvermögen bzw. im Preis/Leistungs- Verhältnis deutlich überlegen sein. Dabei müssen sie wichtigen anwendungspraktischen Forderungen gerecht werden (z. B.: praxisgerechte Belegung der Schaltkreisanschlüsse, Nutzbarkeit preiswerter Speicherschaltkreise, sparsame Belegung von Siliziumfläche, Vorkehrungen für Überwachung, Diagnose, Fehlerbehandlung usw.). Die Vorteile des S.015 müssen also im wesentlichen aus seinen Strukturen und Wirkprinzipien hervorgehen. Die S.015-Architektur muß eine ausgewogene Kombination innovativer und bewährter Lösungen darstellen. Im besonderen muß sie mit Fremdsystemen problemlos koppelbar sein und auf Grundlage industrieüblichen Fachwissens genutzt werden können.

## 3. Überblick über die Architekturprinzipien

### Ressourcen-Paradigma

Die Hardware-Ressourcen, also die Verarbeitungsschaltungen, Speichermittel, Datenwege usw. bestimmen - entsprechend ihrer Gestaltung und Anzahl - in erster Linie die Verarbeitungsleistung. Befehle sind codierte Steuerangaben für die zeitsequentielle Nutzung der Ressourcen. Sie müssen so gestaltet sein, daß die Ressourcen soweit wie möglich mit nützlicher, d. h. zur Lösung der Anwendungsaufgaben unmittelbar beitragender Arbeit beschäftigt werden können. Um die gewünschte Überlegenheit zu erreichen, muß die Ressourcen-Ausstattung von S.015-Prozessoren des oberen Leistungsbereiches über herkömmliche Mikroprozessoren hinausgehen. Das bedeutet z. B. mehrere Verarbeitungswerke sowie die schaltungsseitige Implementierung komplexer leistungsbestimmender Operationen. In den unteren Leistungsbereichen liegt die Ressourcen-Ausstattung im wesentlichen fest; sie ist durch die Aufwandsbeschränkung nicht beliebig erweiterbar. Hier soll ein besseres Preis/Leistungs-Verhältnis durch neuartige Formen der Codierung von Steuer- und Zugriffsangaben sowie durch Verfeinerungen innerhalb der einzelnen Ressourcen erreicht werden.
Das S.015 kann somit nicht einer der üblichen Auffassungen von Rechnerarchitekturen, die durch Begriffe wie CISC, RISC, VLIW usw. gekennzeichnet sind, zugeordnet werden.

### Verpackungsprinzip

Alle Informationsstrukturen können mit möglichst wenigen Bits codiert und dicht gepackt gespeichert werden. Aus technischen Gründen fest vorgegebene Strukturen, wie z. B. Maschinenworte, sind die Verpackungseinheiten für die codierte Information. Um derart codierte Angaben für die Verarbeitung auswählen und Resultate entsprechend abspeichern zu können, sind alle Datenstrukturen bis aufs Bit adressierbar.
Die grundlegenden Informationsstrukturen der S.015-Architektur sind fest formatiert, wobei in manchen Fällen eine gewisse Auswahl unterschiedlicher Formate vorgesehen ist.

Parallelverarbeitung

Es sind folgende Formen der Parallelverarbeitung vorgesehen bzw. bei Forderung nach höchstem Leistungsvermögen implementierbar:

a) parallele bzw. überlappte Ausführung verschiedener Phasen des Befehlsablaufs
b) parallele bzw. überlappte Ausführung verketteter Operationen. Wichtige arithmetische und logische Operationsverkettungen (z. B. "MULTIPLY-ADD") sind als Befehle vorgesehen, so daß entsprechende Anordnungen aus mehreren Verarbeitungswerken direkt gesteuert werden können.
c) Parallelausführung von maximal 5 Befehlen in einem Prozessor (z. B. 4 Operandenverknüpfungen und 1 Verzweigung)
d) dicht gekoppelte Multiprozessoranordnungen aus maximal 4 Prozessoren, deren Universalregister und E/A- Schaltmittel untereinander freizügig zugänglich sind
e) gleichzeitiges Abarbeiten von Programmen in Master-Slave- bzw. seriell gekoppelten Multiprozessorkonfigurationen.

Es ist Angelegenheit des Programmierers bzw. des Compilers, Gelegenheiten zur Parallelausführung von Befehlen in einem Prozessor, wahrscheinliche Verzweigungsrichtungen usw. zu erkennen. In den Befehlen sind Codepositionen vorgesehen, die solche Angaben aufnehmen können.

Datenstrukturen

Die allgemeinste Informationsstruktur ist der Binärvektor mit einer Länge von 1 bit bis 65 536 bits. Kurze Binärvektoren von 1 bit...32 bit werden bevorzugt unterstützt. Solche Binärvektoren heißen Bitfelder. Bitfelder konstanter Länge, die zur Aufnahme von "kurzen" Informationsstrukturen, wie z. B. Zeichencodes, eingerichtet sind, heißen Beutel. Die S.015-Architektur kennt keine Bytes. Das Byte ist vielmehr ein Beutel von 8 bit Länge und wird als solcher von der allgemeinen Bitfeldverarbeitung unterstützt.

Die elementare Informationsstruktur aus technischer Sicht ist

das Maschinenwort von 32 bit, im folgenden kurz als Wort bezeichnet.
Binärvektoren bzw. Bitfelder können als natürliche oder ganze Binärzahlen interpretiert werden. Des weiteren sind Gleitkommazahlen von 32, 64 und 96 bit Länge vorgesehen.

Befehlsstrukturen
Befehle werden grundsätzlich in Blöcken von 6 32-bit-Worten verpackt (Bild 2). Ein solcher Block wird als Eimer bezeichnet. Das erste Wort des Eimers enthält das sogenannte Strukturcodon. Dieses beschreibt die Struktur der nachfolgenden Worte. Je Befehl sind 6 Strukturbits vorgesehen (anders ausgedrückt, enthält ein Eimer 5 38-bit-Befehle). In einem alternativen Modus werden 190 bit des Eimers als ein extrem langer Befehl (bzw. als "horizontaler" Mikrobefehl) aufgefaßt. Die beiden höchstwertigen Bits des Strukturcodons sind zur Moduswahl und dazu vorgesehen, das voreilende Holen des jeweils folgenden Eimers zu steuern. Bild 3 gibt einen Überblick über die wichtigsten Befehlsformate.

Adressierung
Im S.015 sind mehrere Adressenräume vorgesehen, darunter ein homogener interner Speicheradressenraum, der maximal $2^{31}-1$ Worte umfaßt. Adressenangaben beziehen sich grundsätzlich auf Worte. Wortadressen können durch Bitfeldselektoren ergänzt werden, so daß grundsätzlich jedes Bit unmittelbar programmseitig zugänglich ist.

Multitasking
Ein S.015-Prozessor kann hardwareseitig 2...16 unabhängige Tasks unterstützen. Dafür sind entsprechende Registersätze sowie Umschalthardware vorgesehen. Darauf aufbauend kann softwareseitig ein grundsätzlich unbeschränktes Multitasking organisiert werden.

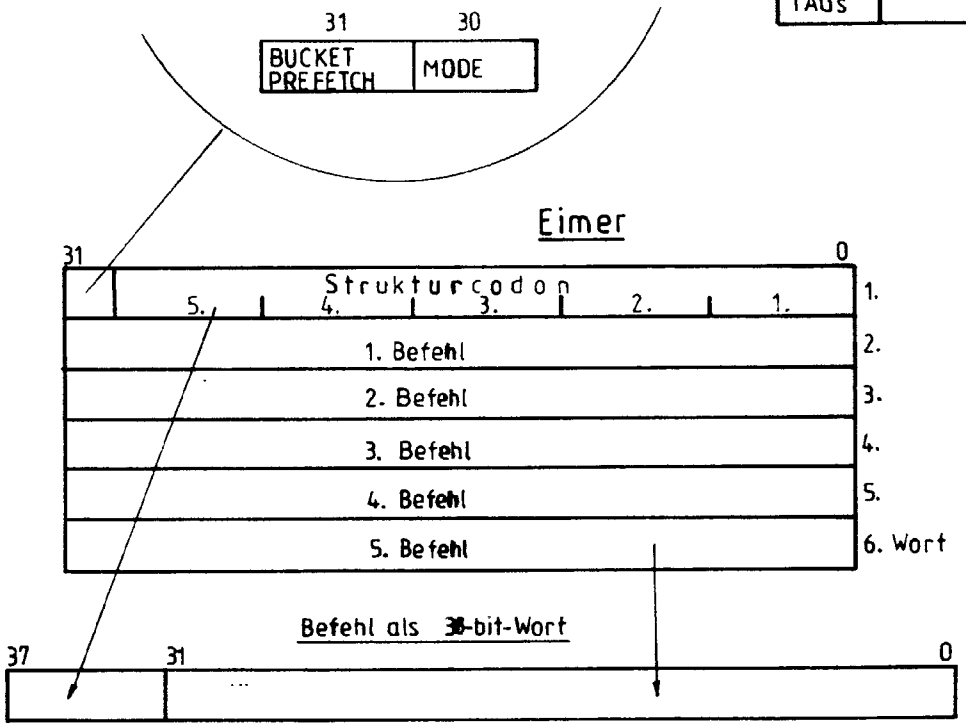
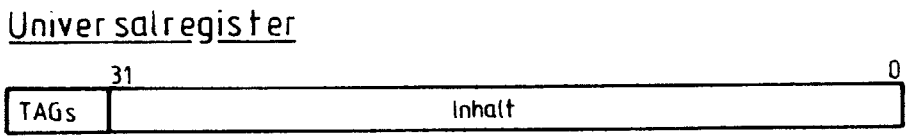



Bild 2 Verpackung der Befehle




TAG – Belegung

| | | |
|---|---|---|
| | 0 | Datenwort |
| | 1 | Adresse |
| | 2 | Datenbereich |
| | 3 | Adressenbereich |
| | 4 | Verkettung |
| | 5 | Zugriffssteuerwort (ACW) |
| | 6 | E/A Steuerwort (IOCW) |
| | 7 | Teilstruktur |
| ab | 8 | direkte Ausgabe |

Bild 4

Registerstruktur und
TAG-Funktionen

## Operationsbefehle

| P | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 0 | COND | OP | RS1 | RS2 | RS3 | | |
| | 0 1 | COND | OP | R1 | R2 | R3 | 0 0 | |
| | | | OP | RS | IMMEDIATE | | CTL | 0 1 |
| | | | OP | RS1 | RS2 | | CTL | 1 0 |
| | | | OP | RS/IMMEDIATE | | | CTL | 1 1 |

## Zugriffs- und Steuerbefehle

| P | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 0 | 0 0 0 | BASE | DESTINATION | DISPLACEMENT | | | | | |
| | | 0 0 1 | BASE | SOURCE | DISPLACEMENT | | | | | |
| | | 0 1 0 | BASE | INDEX | MASK | | | | CTL | 0 0 |
| | | | RS1 | | RS2 | | | | CTL | 0 1 |
| | | | R1 | BIT | R2 | BIT | VAL 1 | VAL 2 | CTL | 0 0 |
| | | | R1 | BIT | R2 | BIT | LEN | | CTL | 1 1 |
| | | 0 1 1 | R1 | BIT | R2 | BIT | LEN1 | | LEN2 | |
| | | 1 0 0 | IMMEDIATE | | | | | | | |
| | | 1 0 1 | BOOLEAN INSTRUCTION 1 | | | BOOLEAN INSTRUCTION 2 | | | | |
| | | 1 1 0 | DIAGNOSE, CONTROL, I/O FORMATS | | | | | | | |
| | | 1 1 1 | reserved | | | | | | | |

## Verzweigungsbefehle

| P | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 1 | 0 0 | G | R | BIT | CND | SKP | BRANCH TARGET | |
| | | 0 1 | G | BASE | CND | SKP | BRANCH TARGET | | |
| | | 1 0 0 | BUCKET ADRS (BRANCH) | | | | | | |
| | | 1 0 1 | BUCKET ADRS (CALL) | | | | | | |
| | | 1 1 0 | CTL | CONDS 1 \| 2 | BRANCH TARGETS 1 \| 2 \| 3 \| 4 | | | | |
| | | 1 1 1 | CTL | TABLE SELECT | 1st ORDINAL | | | | |

P: executable in parallel ; RS: reference specifier ; R: register
CTL: control bits ; G: branch direction guess

**Bild 3** S.015 Befehlsformate

Universalregister

Jeder Task sind 16 Universalregister fest zugeordnet. Jedes Universalregister hat 32 Informationsbits und ist darüber hinaus durch 4 Zusatzbits (TAG-Bits) erweitert, die durch besondere Befehle geladen werden können. In den üblichen Befehlen ist angebbar (YIELD-Bits), ob unmittelbar zum Registerinhalt zugegriffen werden soll oder ob gemäß der TAG-Belegung zusätzliche Wirkungen (indirekte Adressierung, Bereichskontrollen, E/A usw.) ausgeübt werden sollen. Bild 4 gibt einen Überblick über die TAG- Funktionen.
Neben den 16 fest zugeordneten Registern sind von den Befehlen aus weitere 16 Register erreichbar. Dabei ist programmseitig wählbar, ob dieser Registersatz im eigentlichen Registerspeicher oder im Stack der jeweiligen Task lokalisiert ist. Der eigentliche Registerspeicher eines S.015-Komplexes kann bis zu 2048 Register umfassen. Werden Registerblöcke im Stack angeordnet, so läßt sich praktisch unbeschränkt jedem Unterprogrammaufruf ein eigener Registersatz zuweisen (für den schnellen Parameteraustausch sind besondere Befehle vorgesehen).

Tabellenadressierung

Dies ist eine elementare Form der objektorientierten Zugriffsorganisation: Datenstrukturen werden nicht direkt adressiert, sondern durch Angabe von Ordinalzahlen, die Einträge in Zugriffstabellen auswählen. Diese Einträge beschreiben dann die jeweilige Datenstruktur. Weiterhin werden solche Tabellen für Programmverzweigungen bzw. Unterprogrammaufrufe genutzt.

Kurzadressen

In den Befehlen sind kurze Auswahlangaben mit 6- und 12-bit-Ordinalzahlen vorgesehen. Diese können wahlweise zur Registeradressierung, zur Tabellenadressierung oder als Displacementangaben (bezogen auf eine Basisadresse in einem implizit genutzten Register) verwendet werden.

Ereignissteuerung

Ein "Ereignis" kennzeichnet das Auftreten einer bestimmten Bedingung. Ereignisse können ausgelöst werden durch
- Erregung von Eingangsleitungen
- interne Bedingungen (Programmausnahmen, Maschinenfehler)
- programmseitige Aktivitäten (CAUSE- Funktionen).

Die Auslösung eines Ereignisses kann folgende Wirkungen haben:
- gar keine
- Aktivieren einer Task
- Unterbrechungsbehandlung in einer Task
- Verlassen eines Wartezustandes
- Signalisierung zwecks programmseitiger Abfrage.

Das Ereigniskonzept ist eine Erweiterung des bekannten Unterbrechungsprinzips, wobei weitere leistungsbestimmende Funktionen der Hardware übertragen werden. Zur Steuerung dieser Hardware sind programmseitig ladbare Ereignissteuervektoren (ECV) vorgesehen.

Getrennte Adressenräume

Im S.015 gibt es unabhängige Adressenräume für Register, den systeminternen Speicher, die Ein- und Ausgabe, die schaltkreisinternen Speichermittel (z. B. Caches, TLBs) sowie für einen externen homogenen Speicherbereich. Die letztgenannte Vorkehrung gestattet es, S.015-Prozessoren unproblematisch in Fremdsysteme einzusetzen (z. B. an die Konventionen einer VME-Bus- Konfiguration anzupassen).

Flexible Operationscodes

Die Operationsbefehle haben Operationscodes von 8 bzw. 12 bit Länge. Nur ein Teil dieser Codes wird unmittelbar von der Hardware interpretiert. Mit den verbleibenden Codes lassen sich über Zugriffstabellen Unterprogrammrufe organisieren (solche Operationsbefehle sind also praktisch extrem schnelle und komprimiert codierte Unterprogrammrufe).