Interrupt handling in real-time control systems

R. BAUMANN
Technical University of Munich, W. Germany

Introduction

Typical Third Generation Computing Systems are time sharing systems, teleprocessing systems, information and service systems, interactive systems and computer networks, as well as real-time control systems. They are designed to provide the user with a wide range of problem-solving facilities on a low-cost basis through sharing resources and information. The programmer especially wants to find an efficient software environment for development, debugging and execution of his programs.

Owing to the complexity of these systems, general software must be designed very carefully, following sound engineering principles similar to those of hardware design. The structure of the systems considered is determined primarily by functional components independently of the realisation which can actually be given either by hardware or software units.

Since properties like concurrency of programs, automatic resource allocation, sharing of resources, multi-programming, multi-tasking and multi-processing are common to third generation computing systems, general software for these systems will also have much in common. We may therefore ask for components typical of real-time control systems only, which will not be present in other computing systems.

A real-time control system is determined by its ability to synchronise with a technical process. It must therefore contain facilities to:

1. Accept data from, and deliver data to, the technical process at given real-time intervals.
2. Act immediately in response to foreseen or unforeseen events in the technical process.

From the point of view of the computing system, this implies that there must be a functional unit, which can (at least temporarily) interrupt processor activities and alter the programmed sequence of processes due to external events. Emphasis is on the adjective 'external', since interrupts caused by internal events occur in every computing system which realises concurrency of programs, sharing of resources or even some modest fail-soft mechanism.

We therefore conclude that mainly external interrupt handling facilities distinguish real-time control systems from other third generation computing systems. A typical component of a real-time control system is one that is dedicated to handling external interrupt signals. Nevertheless, for practical reasons, external interrupt handling is usually embedded in general interrupt handling facilities.

Following this guideline, a working group of Unterausschuss 'Programmierung' of VDE /VDI Fachausschuss 'Prozessrechner zum Messen, Steuern, Regeln' began to describe hardware and software features of an interrupt input device (Appendix). This effort can also be considered as preparatory work for future standardisation within the scope of Arbeitskreis C of Fachnormenausschuss Informationsverarbeitung (FNI) im Deutschen Normenausschuss (DNA).

1. Classification of interrupts

Owing to their urgency, interrupts can be classified according to their origin as follows:

1. Emergencies (e.g. power supply failure).
2. Traps (e.g. every kind of error occurring in the central processing unit).
3. Peripheral interrupts (messages from peripheral devices).
4. External interrupts (messages from outside).

Emergency interrupts are always served instantly in order to save as much information as possible in case of a breakdown. Traps should be checked before any other interrupt is served (except for emergencies) in order to ensure correct functioning of the computing system itself. Hence interrupts of class (1) and (2) may have immediate access to the central processing unit, while those of class (3) and (4) are handled by a distinct interrupt device. These should only compete with other activities of the computing system on the same level of urgency.

2. Interrupt device

2.1 The interrupt device has to perform the following functions:

a. Accept the interrupt signals.

b. Attach and evaluate interrupt priorities.
c. Assign start addresses of interrupt response programs.

Interrupt signals enter the interrupt device by interrupt entries, which can consist of three flip-flops, the outer mask element, the interrupt location (holding the signal) and the inner mask element (see Fig. 1, Appendix).

2.2 The set of interrupt entries can be structured in different ways by forming subsets according to common properties such as:

a. Equal interrupt priority.
b. Common mask elements (groupwise locking).
c. Common read/write instructions for the mask elements.
d. Common response program.

The subsets defined in this way are usually called groups, except for the case of equal interrupt priority, where the term 'level' is used. Obviously different structures can be realised simultaneously in a given interrupt device (Figs 3 and 4, Appendix). An interrupt entry can be:

a. Armed or disarmed by setting the outer mask element.
b. Enabled or disabled by setting the inner mask element.

These functions can also be performed groupwise.

2.3 Since it must be assumed that, in reality, several interrupt signals will compete for execution simultaneously, an interrupt priority scheme is needed to determine the order of execution.

When the decision as to whether or not to interrupt the processor's activity is to be influenced by the priority scheme, priorities are to be attached not only to interrupt entries but also to the central processor, according to the piece of program occupying it. In that case the priority scheme is much stronger, since not only the single selection of an interrupt request, but also the interruptibility of the affiliated response program is within its scope. Here is the chance to weld together external interrupt handling with other activities of the computing system, combining the priority-allocation scheme for internal system activities and external interrupt handling.

Concerning interrupt priority we have to differentiate between:

a. Fixed priority (attached to every interrupt entry or interrupt level by hardware).
b. Variable priority (attached by software).
c. Subpriority (due to a fixed sequence of serving interrupt entries of a given interrupt level).

The interrupt device has to put in order the interrupt signals standing at enabled entries, according to their priority, and has to offer the interrupt request of highest priority for execution.

3. Interrupt execution

3.1 The interrupt is actually effected if the following conditions are fulfilled on the processor side:

a. The state of main scheduling or supervisory program controlling the processor's activity must permit the interrupt (e.g. according to a comparison between processor priority and interrupt priority).
b. The piece of program occupying the processor must be in an interruptible state.
c. The instruction the processor is working on must be interruptible.

In multi-processor systems, interrupt requests can be dispatched to all equal processors on the basis of a comparison between highest interrupt priority and lowest processor priority. Such a schedule makes the system more flexible than dedicating a single processor to interrupt handling.

3.2 In the case of normal competition [interrupts belonging to class (3) or (4), see Section 1], it is desirable that the interrupted program can later re-enter the processor without avoidable loss of time and information. This implies that, before releasing the processor, the contents of all registers and other locations relevant for continuation are saved by putting them in store or by just switching to another register block. Admittedly, in the latter case, the processor has to provide for as many register blocks as different programs are to be served simultaneously by it.

In several process control systems there is a one-to-one correspondence of interrupt levels to register blocks. In other systems the register blocks are scheduled by the supervisory program.

4. Characteristic time intervals

In order to judge the capacitance of a real-time control system and to compare the efficiency of its performance, the user and the programmer need information on time spans consumed in interrupt handling. For reasons of correct comparison, time intervals and related activities should be defined clearly.

When tracing an interrupt request from the origin of the interrupt signal to the completion of the responding action, the following instants are of interest (Fig. 5, Appendix):

P_0 – the interrupt signal enters the interrupt device,

P_1 – the interrupt request is announced to the processor,
P_2 – the interrupt request is accepted by the processor for execution,
P_3 – the processor starts to execute the first instruction of the specific response program,
P_4 – the processor finishes the last instruction of the specific response program,
P_5 – the processor starts to execute the first instruction of the next scheduled program.

For the intervals T_i defined by

T_i = time between P_{i-1} and P_i

the following denotations are recommended:

T_1 – transition time
T_2 – latency time
T_3 – recognition time
T_4 – execution time
T_5 – return time.

Additional recommendations are:

$T_1 + T_2$	– waiting time
$T_1 + T_2 + T_3$	– reaction time
$T_3 + T_4 + T_5$	– interrupt time
$T_1 + T_2 + T_3 + T_4$	– response time
$T_3 + T_5$	– organisational time.

It must be admitted that a considerable amount of philosophy is hidden behind these denotations. Nevertheless, they may be helpful for comparing the interrupt handling facilities of different real-time control systems.

Conclusion

Real-time control systems can be expected to provide for interrupt handling facilities on different stages. Hard interrupts will have immediate processor access while soft interrupt requests are handled via the operating system, competing with other activities on a priority basis.

APPENDIX

Editors note: This appendix has not been translated since it is concerned with precise terminology.

Entwurf für eine einheitliche Beschreibung von Unterbrechungsvorgängen

1. Übersicht
2. Unterbrechungseingänge
3. Zustandsbezeichnungen von Unterbrechungsgängen
4. Unterbrechungsgruppen
5. Auswertung der Priorität der Unterbrechungssignale
6. Definition charakteristischer Zeitintervalle

ausgearbeitet von der Arbeitsgruppe "Interruptwerke" des Unterausschusses "Programmiertechnik" des VDI/VDE-Ausschusses "Prozessrechner zum Messen, Steuern, Regeln".

Mitarbeiter:

Prof. Dr R. Baumann, Mathematisches Institut der Technischen Universität München
Dipl-Phys. J. Brandes, Institut für Informatik, Universität Karlsruhe
Dr G. Heller, BASF Ludwigshafen
Dipl-Ing. G. Hepke, IDT der GfK Karlsruhe
Dr P. Höhne, Siemens ZL Munchen
Dipl-Phys. E. Huber, MBB Ottobrunn
Dr M. Prasser, AEG-Telefunken Konstanz
Dipl-Phys. W. Rüb, Mathematisches Institut der Technischen Universität München.

1. Übersicht

Die Verarbeitung von Unterbrechungssignalen in einem Rechnersystem wird durch eine Unterbrechungseinheit veranlasst.

Der Unterbrechungseinheit fallen dabei folgende Hauptfunktionen zu:

1. Annahme der Unterbrechungssignale
2. Zuordnung und Auswertung der Unterbrechungspriorität, und
3. Zuordnung von Anfangsadressen der Antwortprogramme.

Dabei bleibt offen, ob die einzelnen Funktionen ganz oder teilweise durch Hardware oder Software realisiert sind.

Es wird davon ausgegangen, dass die Unterbrechnungsanforderung durch ein binäre Unterbrechungssignal gestellt wird. Die Übertragung zusätzlicher Information über die Unterbrechungsursache auf anderen Datenwegen wird hier nicht betrachtet.

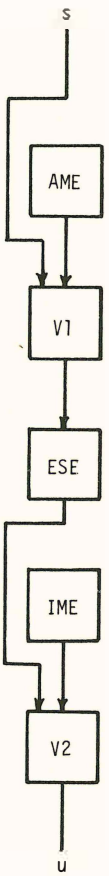
Zur Beurteilung des Verhaltens von Rechnersystemen bei Unterbrechungsanforderungen muss auch der zeitliche Ablauf bei der Bearbeitung angegeben werden.

2. Unterbrechungseingänge

Ein Unterbrechungseingang hat die Aufgabe, ein ankommendes Unterbrechungssignal wahrzunehmen, zu speichern und gegebenenfalls gegenüber dem Rechner abzusperren.

Ein Unterbrechungseingang kann aus Aussenmaskenelement, Eingangsspeicherelement sowie

Innenmaskenelement bestehen (s. Bild 1). Jedes dieser Elemente kann den Wert L oder O speichern.



Abkürzungen:

AME Aussenmasken-Element
ESE Eingangsspeicher-Element
IME Innenmasken-Element
s,u Unterbrechungs-Signale
V1 Verknüpfungsglied 1
V2 Verknüpfungsglied 2

Bild 1 Unterbrechungseingang

Aussenmaskenelement und Eingangsspeicherelement sowie Eingangsspeicherelement und Innenmaskenelement sind je durch ein Verknüpfungsglied verbunden. Bei geeigneter Besetzung des Aussenmaskenelements verhindert das Verknüpfungsglied V1 den Durchgang eines ankommenden Signals zum Eingangsspeicherelement, bei geeigneter Besetzung des Innenmaskenelements verhindert das Verknüpfungsglied V2 die Weitergabe des im Eingangsspeicherelement angekommenen Signals.

Aussenmasken- und Innenmaskenelement werden nur vom Rechner (DIN 44300) her gesetzt und gelöscht. Das Eingangsspeicherelement kann von einer externen Signalquelle (bei geeigneter Besetzung des Aussenmaskenelements) oder vom Rechner her gesetzt, jedoch im allgemeinen nur vom Rechner her gelöscht werden.

3. Zustandsbezeichnungen von Unterbrechungseingängen

Für die Zustände der Elemente eines Unterbrechungseingangs werden folgende Bezeichnungen vorgeschlagen:

TABLE 1

	Bezeichnung
Aussenmaskenelement	aussensperrend
	nicht aussensperrend
Innenmaskenelement	innensperrend
	nicht innensperrend
Eingangspeicher-element	gesetzt
	nicht gesetzt

TABLE 2

Aussenmasken-element	Eingangs-speicherelement	Innenmasken-element	Zustand des Unterbrechungs-eingangs
aussensperrend	*	*	aussengesperrt
*	*	innensperrend	innengesperrt
nicht aussensperrend	nicht gesetzt	*	bereit
*	gesetzt	*	belegt

*bedeutet eine beliebige Besetzung des betreffenden Elements

Daruber hinaus werden für die 8 verschiedenen Besetzungsmöglichkeiten eines Unterbrechungseingangs 4 Zustandsbezeichnungen angegeben. Alle weiteren Zustände lassen sich als Kombination dieser Bezeichnungen beschreiben.

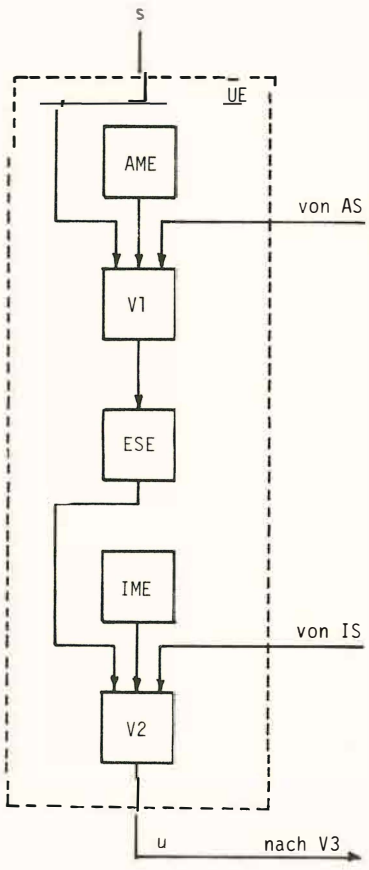
4. Unterbrechungsgruppen

Eine Unterbrechungsgruppe ist ein Satz von Unterbrechungseingängen, die nach einer gemeinsamen Eigenschaft zusammengefasst sind. Solche Eigenschaften können z.B. sein:

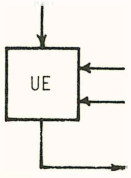
a) Gleiche Unterbrechungspriorität der Eingänge

Jedem Unterbrechungseingang ist eine Unterbrechungspriorität p zugeordnet (vergleiche 5).

a) Schaltung:



b) Schaltsymbol:

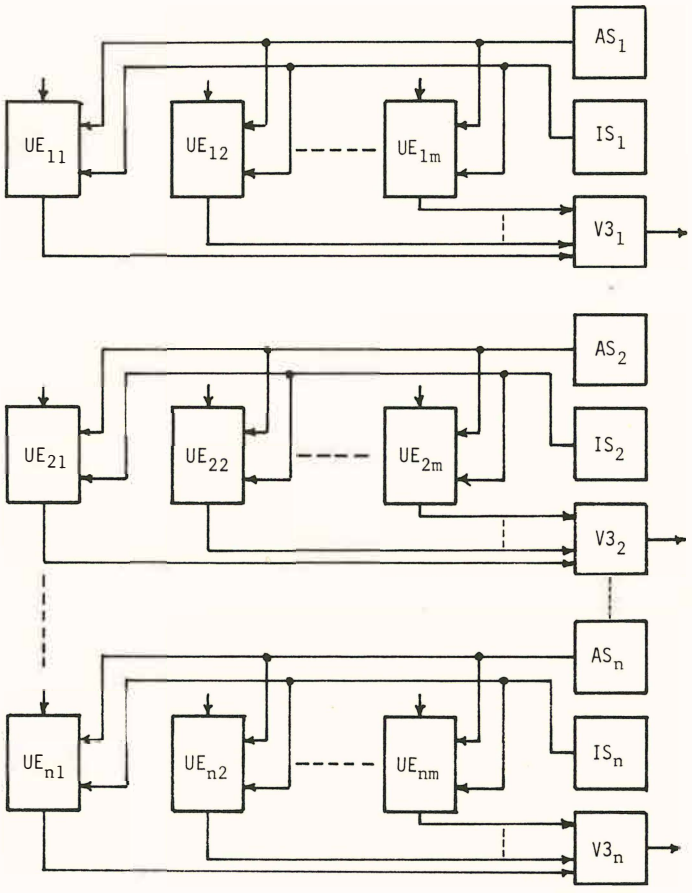


Abkürzungen:

AME	Aussenmasken-Element
AS	gruppenweise Aussensperre
ESE	Eingangsspeicher-Element
IME	Innenmasken-Element
IS	gruppenweise Innensperre
s,u	Unterbrechungs-Signale
UE	Unterbrechungs-Eingang
V1	Verknüpfungsglied 1
V2	Verknüpfungsglied 2
V3	Verknüpfungsglied 3

Bild 2 Element einer Unterbrechungsgruppe

Die Unterbrechungsprioritäten regeln den Vorrang der Unterbrechungseingänge. Die Bearbeitung einer Unterbrechungs-Anforderung der Priorität p kann nicht durch eine Unterbrechungs-Anforderung gleicher oder niedrigerer Priorität unterbrochen werden. Die Unterbrechungsgruppe mit Eingängen gleicher Priorität p heisst Unterbrechungsebene. Innerhalb einer Unterbrechungsebene wird die Auswahl der zu bearbeitenden Anforderung durch eine Subpriorität festgelegt.



Abkürzungen:

AS	Aussensperre
IS	Innensperre
UE	Unterbrechungs-Eingang
V3	Verknüpfungsglied 3 (für Eingänge gleicher Unterbrechungspriorität)

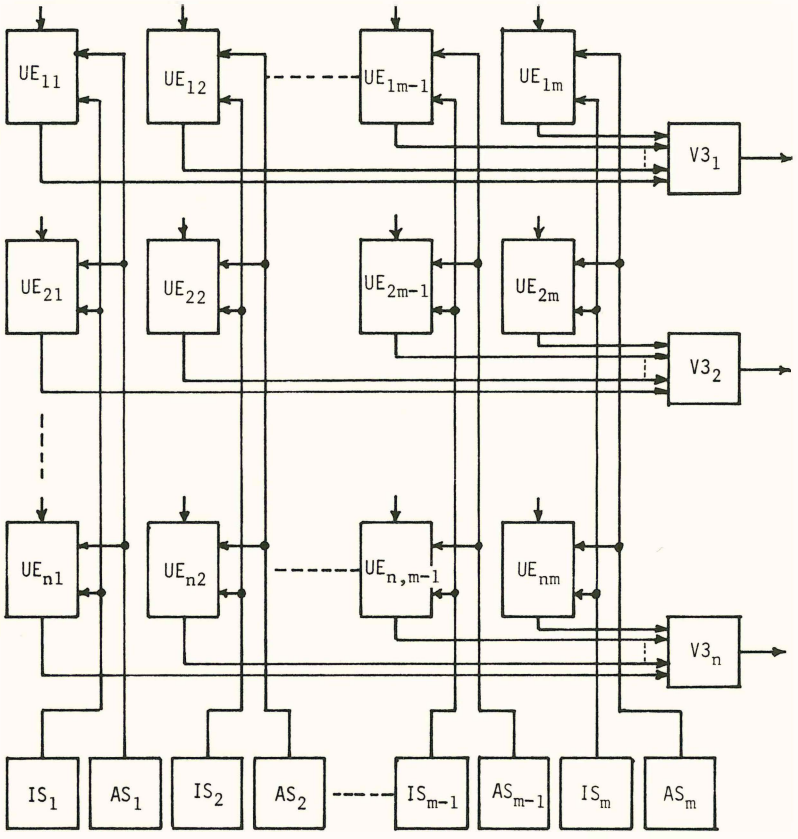
Bild 3 Beispiel zur Gruppierung der Unterbrechungseingänge, bei dem Unterbrechungsebenen und Gruppen mit gemeinsamer Aussen- und Innensperre zusammenfallen

b) Gemeinsame Lese- und Schreibbefehle

Bei einer solchen Gruppe können sämtliche Aussen-, Innenmasken- sowie Eingangsspeicherelemente gemeinsam gelesen und beschrieben werden. Die Zusammenfassung der Aussen- bzw. Innenmaskenelemente dieser Gruppe heisst Aussenmaske bzw. Innenmaske.

c) Gemeinsames Sperrelement der Eingänge

Diese Gruppe besitzt ein gemeinsames Sperrelement, das die Werte L oder O speichern kann. Dieses Element heisst Aussensperre, wenn es bei geeigneter Besetzung für alle Eingänge der Gruppe aussensperrende Wirkung hat. Es heisst Innensperre, wenn es innensperrende Wirkung besitzt. Aussen- und Innensperre konnen vom Rechner her gelesen, gesetzt und gelöscht werden.



Abkürzungen:

AS Aussensperre
IS Innensperre
UE Unterbrechungs-Eingang
V3 Verknüpfungsglied 3
(für Eingänge gleicher Unterbrechungspriorität)

Bild 4 Beispiel zur Gruppierung der unterbrechungseingänge bei dem Unterbrechungsebenen und Gruppen mit gemeinsamer Aussen- und Innensperre nicht zusammenfallen

d) Gemeinsame Anfangsadresse des Unterbrechungsanwortprogramms

Allen Unterbrechungseingängen dieser Gruppe ist die gleiche Anfangsadresse des Antwortprogramms im Rechner zugeordnet.

Die Gruppierung der Unterbrechungseingänge nach den in a) - d) genannten Eigenschaften kann zu verschiedenen Strukturen führen. Bild 3 zeigt ein Beispiel, bei dem die Gruppierung nach gleicher Unterbrechungspriorität mit der Gruppierung nach gemeinsamer Aussen- und Innensperre zusammenfällt; im Beispiel des Bildes 4 fallen diese Gruppierungen dagegen nicht zusammen (das in Bild 3 und 4 verwendete Schaltsymbol UE ist in Bild 2 erläutert).

5. Auswertung der Priorität der Unterbrechungssignale

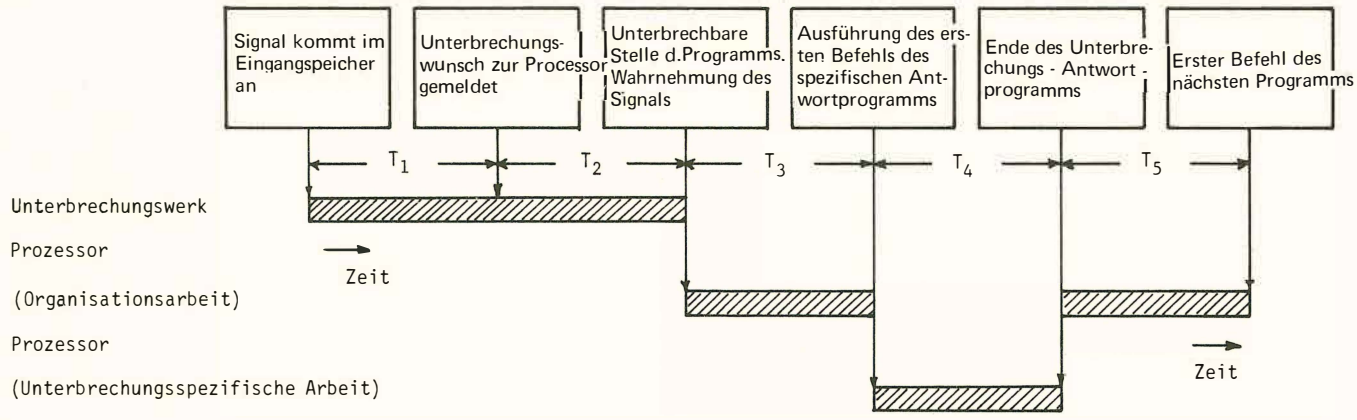
Es wird unterschieden zwischen Unterbrechungspriorität und Prozessor-priorität.

Die Unterbrechungspriorität ist diejenige Priorität, die einer Unterbrechungsebene statisch oder dynamisch zu einem bestimmten Zeitpunkt zugeordnet ist.

Die Prozessorpriorität wird einem Prozessor entsprechend der gerade von ihm bearbeiteten Aufgabe zugeordnet.

Die Prioritätsauswertung umfasst folgende Funktionen:

a) Zuordnung der Unterbrechungspriorität zu den aktivierten Unterbrechungsebenen.
b) Ermittlung der Ebene mit höchster Priorität.
c) Vergleich zwischen Prozessor-Priorität und höchster Unterbrechungs-Priorität (bei Mehrprozessor-Systemen Auswahl eines geeigneten Prozessors).
d) Entscheidung über eine Unterbrechung auf Grund folgender Kriterien:
   α) Vergleich nach c).
   β) Generelle Unterbrechungssperre gesetzt?
   γ) Vorliegen einer unterbrechbaren Stelle im Prozessor.
e) Innerhalb der Ebene mit höchster Unter-



Definitionen:

T_1 = Durchlasszeit
T_2 = Latenzzeit
T_3 = Erkennungszeit
T_4 = Ausführungszeit
(umfasst nur den Ablauf des unterbrechungsspezifischen Antwortprogramms. Identifizierung der Unterbrechungsursache und Bereitstellung der Betriebsmittel sind zu T_3 zu rechnen)
T_5 = Rückkehrzeit

$T_1 + T_2$ = Wartezeit
$T_1 + T_2 + T_3$ = Reaktionszeit
$T_3 + T_4 + T_5$ = Unterbrechungszeit
$T_1 + T_2 + T_3 + T_4$ = Antwartzeit
$T_3 + T_5$ = Organisationszeit

Bild 5 Definition charakteristischer Zeitintervalle

brechungs-Priorität Bestimmung der Unterbrechungsanforderung mit höchster Subpriorität.

Ausserdem muss die der Unterbrechungs-Anforderung zugeordnete Anfangsadresse festgelegt werden.

6. Definition charakteristischer Zeitintervalle

Bild 5 beschreibt ein Zeitdiagramm einer Unterbrechung. Im zeitlichen Ablauf werden sechs charakteristische Zeitpunkte fixiert und dadurch fünf Zeitintervalle T_1 bis T_5 definiert.

Die Zeitangaben für diese Intervalle können je nach angenommener Belastung des Rechners sehr unterschiedlich sein. Die günstigsten Zeitangaben erhält man, wenn man annimmt, dass der Ablauf der Unterbrechungsverarbeitung nicht durch andere Aktivitäten gestört wird. In allen anderen Fällen sind Angaben über die angenommene Belastung des Rechners mit anderen Aufgaben erforderlich. Zu Vergleichszwecken wäre eine Auswahl von Standardaufgaben durchaus sinnvoll.