Werner Stehling

Interrupts bei Prozessoren 680XX

Teil 2: Stack, Hardware und Programmierung

Nachdem im ersten Teil geschildert wurde, was Ausnahmen sind und wie der Prozessor nach Eintreten einer Ausnahmebedingung die Adresse der zuständigen Routine findet, geht es nun ganz ins Detail: Die Unterschiede der verschiedenen 680XX-CPUs, die Anforderungen an die Hardware und hilfreiche Regeln für die Programmierung von Interrupt-Routinen sind Inhalt dieses zweiten und letzten Teils unserer Einführung.

Auf die eine oder andere Art hat die CPU also nach Auftreten der Ausnahme eine Vektornummer ergattern können. Durch Multiplikation mit 4 erhält sie den Offset, d. h. die relative Adresse, in der Tabelle der Ausnahmevektoren. Sie weiß damit, wo die Routine beginnt, mit der sie auf die Unterbrechung des Programmes reagieren soll.

Der wichtigste Grundsatz bei der Ausnahmeverarbeitung ist, daß das unterbrochene Programm nichts davon merken darf. Es soll nach Beendigung der Ausnahmeverarbeitung so weiterlaufen, als ob diese überhaupt nicht stattgefunden hätte, wenigstens in den meisten Fällen. Die neueren Prozessoren 68010 und 68020 erlauben sogar eine Wiederholung des letzten Befehls, falls dieser durch einen Busfehler unterbrochen wurde.

Rettungsaktion

Um so intelligent agieren zu können, muß am Ende der Ausnahmeverarbeitung der innere Zustand der CPU genau so sein wie vor Beginn. Das Retten der Arbeitsregister bleibt dem Programmierer überlassen, dagegen gibt es einige interne Informationen, auf die der Anwender keinen Zugriff hat, die aber vom Prozessor vor dem Sprung in die Ausnahmeverarbeitungsroutine auf den Stack gerettet werden. Um nur einige Beispiele zu nennen: Inhalt von Statusregister und Programmzähler vor dem Auftreten der Ausnahme, Status der Funktionscode-Leitungen usw. Je nach Art der Ausnahme müssen mehr oder weniger Informationen aufbewahrt werden, bei einem Busfehler ist es zum Beispiel interessant, ob als letztes auf einen Befehl oder auf einen Operanden zugegriffen wurde; das wiederum ist bei einem normalen Interrupt völlig ohne Belang, da der laufende Befehl in jedem Fall vor der Annahme des Interrupts vollständig abgearbeitet wird.

Die einzige Ausnahme von dieser Regel ist ein externes Reset-Signal. Normalerweise wird es nur beim Einschalten der Stromversorgung oder als allerletzte Notbremse bei chaotischem Verhalten des Mikrocomputers angelegt. In beiden Fällen erfolgt sowieso eine völlige Neubesinnung des Systems auf seine Aufgabe. Es ist also nur logisch, wenn die CPU beim Reset erst gar keine Anstalten unternimmt, irgendwelche internen Informationen zu retten. Wohin sollte sie auch – der Stackpointer wird ja gleich zu Beginn der Ausnahme neu initialisiert! Zwischen den Prozessoren der Familie 68000 bestehen zusätzliche Unterschiede, immer getreu der Devise: Je intelligenter ich bin, desto mehr muß ich mir merken.

Wenden wir also unsere Aufmerksamkeit jetzt den verschiedenen Stackformaten zu. Im ritualisierten Ablauf der Ausnahmeverarbeitung rettet die CPU alles auf den Stack, von dem sie glaubt, daß es während oder nach der Ausnahmeverarbeitung von irgendeinem Nutzen sein könnte.

Verschiedene Stackformate

Wie es sich gehört, liegt der Supervisor-Stack im Supervisor-Datenbereich, was sich auch bei Stackzugriffen im Zustand der Funktionscode-Leitungen widerspiegelt. Es wird der Supervisor-Stackpointer aktiviert, der 68020 verwendet bei Interrupts zusätzlich einen speziellen Interrupt-Stackpointer. Der Stack wächst immer von höheren zu niedrigen Adressen; der kleinste Stackeintrag ist ein Wort (16 Bit = 2 Byte) und der Stackpointer zeigt immer auf den letzten gültigen Stackeintrag.

Je nach Art der Ausnahme rettet der Prozessor eine verschiedene Anzahl von Worten auf den Stack (auch der 68000). Nach Beendigung der Ausnahmeverarbeitung (genauer: beim Ausführen des RTE-Befehls) regeneriert der Prozessor seinen ursprünglichen Zustand, indem

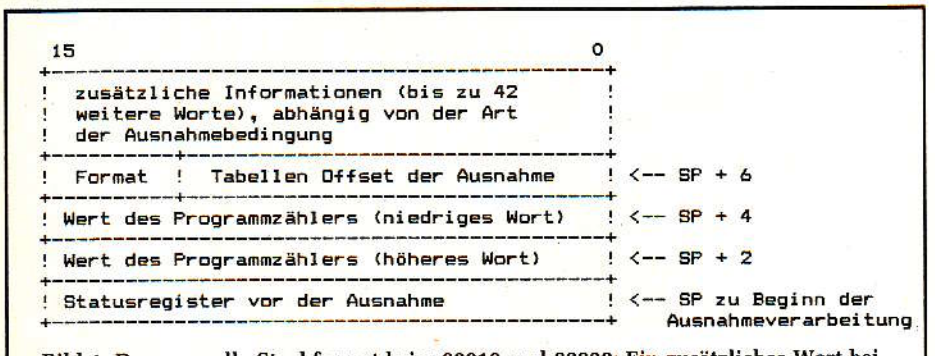

Bild 1. Das generelle Stackformat beim 68010 und 68020: Ein zusätzliches Wort bei SP + 6 enthält die Information, wie viele Worte wirklich auf dem Stack abgelegt wurden („Format") und identifiziert die Ausnahme

er die gleiche Zahl von Worten vom Stack zurückliest, die er zu Beginn dort abgelegt hat. Das Stackformat des 68000 ist grundsätzlich verschieden von dem der beiden anderen Prozessoren, während das Format des 68020 wenigstens aufwärtskompatibel zu dem des 68010 ist.

Beim 68000 können zwei verschiedene Stackformate auftreten, und zwar werden bei Bus- und Adreßfehlern insgesamt sieben Worte gerettet, in allen anderen Fällen nur Programmzähler und Statusregister (zusammen also drei Worte).

Der 68010 kennt ebenfalls nur zwei Formate, allerdings sind es bei Bus- und Adreßfehlern 29 Worte und in allen übrigen Fällen vier Worte. Bereits sechs verschiedene Formate gibt es beim 68020.

Bei 68010 und 68020 enthält das viertunterste Wort auf dem Stack (*Bild 1*) eine Angabe über das verwendete Stackformat (*Tabelle 1*). Die unteren Bit enthalten den Vektor-Offset für die Tabelle der Ausnahmevektoren (was nichts anderes ist, als die mit vier multiplizierte Vektornummer der Ausnahme).

Stackformate bei Bus- und Adreßfehlern

Bild 2 zeigt die Anordnung der Informationen auf dem Stack, wie sie vom 68000 vorgenommen wird. Der Wert des Programmzählers zeigt in die Nähe des Befehls, während dessen Ausführung der Fehler auftrat. Der Wert des Instruktionsregisters gibt den Befehl an, der gerade ausgeführt wurde. Außer der Adresse, die zum Fehler führte, enthält das unterste Wort auf dem Stack noch die Prozessor-Statussignale FC0...FC2. Das dritte Bit im speziellen Statuswort (I/N) ist Null, wenn zum Zeitpunkt des Fehlers gerade eine Ausnahmeverarbeitung der Gruppen 0 oder 1 bearbeitet wird, es ist gesetzt bei „normaler" Programmausführung oder bei einer Ausnahmeverarbeitung eines Befehls aus Gruppe 2. Bit 4 zeigt an, ob der Fehler beim Lesen ($R/\overline{W} = 1$) oder bei einem Schreibzugriff ($R/\overline{W} = 0$) auftrat.

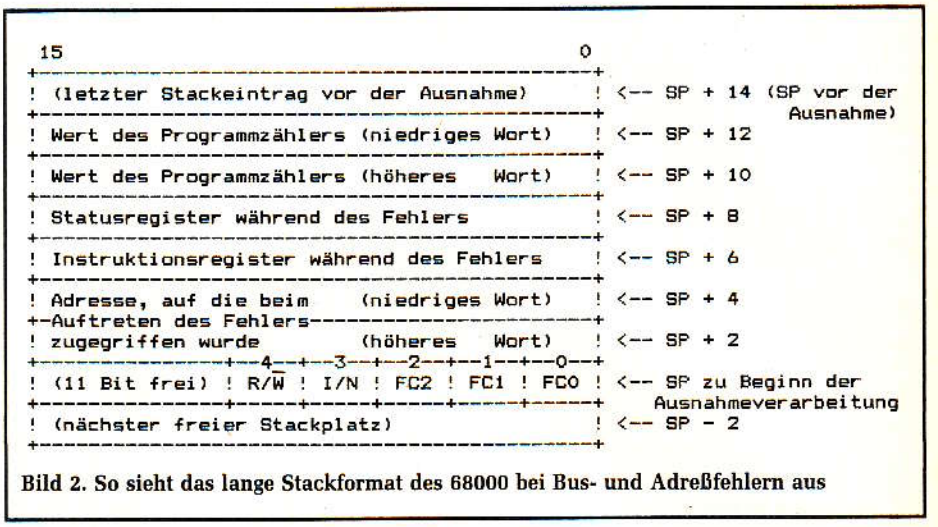

Bild 2. So sieht das lange Stackformat des 68000 bei Bus- und Adreßfehlern aus

Der RTE-Befehl des 68000 holt grundsätzlich nur drei Worte vom Stack zurück, also ACHTUNG:

Bei Bus- oder Adreßfehlern muß die Ausnahmeverarbeitungsroutine dafür sorgen, daß der Stack von allem nicht mehr benötigten Unrat gesäubert wird!

Der 68010 rettet wesentlich mehr Informationen auf den Stack (*Bild 3*) und gibt damit dem Anwender die Möglichkeit, die genaue Fehlerursache lokalisieren und gezielt darauf reagieren zu können. Ob das Leben eines geplagten Programmierers dadurch wirklich einfacher wird, sei jetzt mal dahingestellt. Statusregister, Programmzähler, Befehlsregister und Fehleradresse entsprechen den Informationen beim 68000. Auch hier kann der Wert des Programmzählers auf eine Adresse verweisen, die 1 bis 5 Worte hinter dem eigentlichen Befehl liegt. Die Ursache liegt im „Prefetch"-Mechanismus des Prozessors, der während der Befehlsausführung bereits die nächsten Speicherstellen einliest (Auffüllen der internen Pipeline zur Geschwindigkeitssteigerung).

Im speziellen Statuswort bei SP + 8 sind im Vergleich zum 68000 ebenfalls mehr Informationen enthalten (*Bild 4*). Es gibt den internen Zustand der CPU zum Zeitpunkt der Fehlerbedingung wieder:

- FC0...FC2 ist der bekannte Funktionscode (Status),
- RW ist das Lese-/Schreibflag wie beim 68000,
- BY ist ein Byte-Flag, das anzeigt, ob ein Byte (BY = 1) oder ein Wort (BY = 0) übertragen wurde;
- HB gibt an, ob das niedere (HB = 0) oder das höhere (HB = 1) Byte übertragen wurde,
- RM wird bei einem Read-Modify-Write-Zyklus gesetzt,
- DF ist gesetzt, wenn Daten in den Daten-Eingangspuffer geladen wurden,
- IF ist gesetzt, wenn ein Befehl in den Befehls-Eingangspuffer geladen wurde;
- RR bedeutet ReRun-Flag und muß vor der Rückkehr aus der Ausnahmeverarbeitungsroutine vom Programmierer entsprechend gesetzt werden: RR = 1 bedeutet, daß der Fehler auf Grund der Informationen im speziellen Statuswort durch die Ausnahmeverarbeitung (d. h. per Software) behoben wurde, während RR = 0 dem Prozessor signalisiert, daß er es im Anschluß an den RTE-Befehl noch einmal versuchen soll.

Der 68020 kennt sogar zwei unterschiedliche Stackformate bei Bus- oder Adressfehlern. Wenn der Fehler beim Lesen des Befehlswortes auftritt, muß der Prozessor wesentlich weniger Information

Tabelle 1: Bedeutung des Formatcodes bei 68010 und 68020

Code	68010	68020	Worte	Ausnahme-Bedingung
0000	*	*	4	Normalfall
0001	-	*	4	Interrupt-Dummy-Information
0010	-	*	6	Ausnahme-Befehl (z. B. TRAP)
1000	*	-	29	Busfehler
1001	-	*	10	Coprozessor-Ausnahme
1010	-	*	16	Busfehler bei Befehls-Code
1011	-	*	46	Busfehler bei Parameter

aufbewahren als zum Zeitpunkt des Lesens oder Schreibens der Parameter. Bedingt durch die unterschiedliche Länge der internen Befehls-Pipeline im Vergleich zum 68010 und den zusätzlichen (32 Bit breiten) Langwort-Transfer hat das spezielle Statuswort ebenfalls einen anderen Aufbau. Darauf soll hier jedoch nicht genauer eingegangen werden.

Stackformat bei den übrigen Ausnahmen

Bild 5 zeigt den Inhalt des Supervisor-Stacks des 68000, bevor der Sprung in die Ausnahmeverarbeitung erfolgt. Bei allen Ausnahmen außer bei Bus- oder Adreßfehlern werden nur Statusregister und Programmzähler abgelegt; damit ist die Sache noch bequem zu überschauen. Als Referenz für den Stackpointer (SP) wird wieder sein Wert zu Beginn der Ausnahmeverarbeitungsroutine angegeben.

Der 68010 schreibt ein Wort mehr auf den Stack (*Bild 6*). Die für die CPU interessante Information ist der Formatcode 0000, an dem sie beim Ausführen des RTE-Befehls erkennen kann, daß sie genau vier Worte vom Stack zurückholen darf, um damit die Ausnahmeverarbeitung zu beenden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RR		IF	DF	RM	HB	BY	RW						FC2	FC1	FC0

Bild 4. Das spezielle Statuswort, das der 68010 bei Bus- oder Adreßfehlern auf dem Stack ablegt, enthält viele Informationen

Im Unterschied zu den Verhältnissen bei Bus- oder Adreßfehlern gibt der Wert des Programmzählers die Adresse des nächsten nicht ausgeführten Befehls an, bei dem nach Beendigung der Ausnahmeverarbeitung die Arbeit schön ordentlich fortgesetzt wird – oder besser gesagt: werden soll, denn die Ausnahmeverarbeitungsroutine kann natürlich etwas völlig anderes im Sinn haben.

Das gleiche Format (Bild 6) verwendet normalerweise auch der 68020. Speziell bei Interrupts wird diese Information auf den Supervisor-Stack geschrieben. Ist das M-Bit im Statusregister (Bild 1, erster Teil) gesetzt, wurde also zum Zeitpunkt des Interrupts mit dem Supervisor-Stack gearbeitet, so wird das M-Bit gelöscht. Das S-Bit ist zu diesem Zeitpunkt ohnehin gesetzt und die ganze Information wird noch einmal auf den Interrupt-Stack gelegt. Allerdings erhält sie jetzt den Formatcode 0001 (siehe Tabelle 1), damit die CPU später beim RTE-Befehl merkt, daß nicht hier, sondern auf dem Supervisor-Stack die eigentlichen Rückkehr-Daten stehen.
Wenn die Ausnahmeverarbeitung durch einen Befehl herbeigeführt wurde (z. B. CHK oder TRAP), wird zusätzlich noch die Adresse dieses Befehls auf den Stack gerettet (*Bild 7*). Der Wert des Programmzählers zeigt dann auf den nächsten Befehl, mit dem es nach der Ausnahme weitergeht.

Wenn man bedenkt, was die CPU bis jetzt schon alles geleistet hat, ist der Rest nur noch ein Kinderspiel. Der Prozessor nimmt den Vektoroffset der Ausnahme als Index, ein 68010 oder 68020 zählt den Inhalt seines Vektorbasisregisters dazu, und lädt dann den Programmzähler mit der Adresse, die er in der Tabelle 1 (erster Teil) der Ausnahmevektoren findet. So erfolgt ein unbedingter Sprung zur Ausnahmeverarbeitungsroutine, die dann wie ein ganz normales Programm ausgeführt wird.

15 … 0	
(letzter Stackeintrag vor der Ausnahme)	<-- SP + 58 (SP vor der Ausnahme)
zusätzliche interne Informationen (16 weitere Worte), z.B. die Versionsnummer des Prozessors	<-- SP + 56 …
Wert des Instruktionsregisters	<-- SP + 24
nicht benutzt (reserviert)	<-- SP + 22
Inhalt des Dateneingangsregisters der CPU	<-- SP + 20
nicht benutzt (reserviert)	<-- SP + 18
Inhalt des Datenausgangsregisters der CPU	<-- SP + 16
nicht benutzt (reserviert)	<-- SP + 14
Adresse, auf die beim Auftreten des Fehlers zugegriffen wurde (niedriges Wort)	<-- SP + 12
	<-- SP + 10
spezielles Statuswort	<-- SP + 8
1 0 0 0 ! Tabellen Offset der Ausnahme	<-- SP + 6
Wert des Programmzählers (niedriges Wort)	<-- SP + 4
Wert des Programmzählers (höheres Wort)	<-- SP + 2
Statusregister vor der Ausnahme	<-- SP zu Beginn der Ausnahmeverarbeitung
(nächster freier Stackplatz)	<-- SP - 2

Bild 3. So packen 68010 und 68020 ihre Informationen bei Bus- und Adreßfehlern auf den Stack: Der Formatcode 1000 zeigt an, daß der Eintrag 29 Worte lang ist

Ende der Fahnenstange

Im Normalfall wird die Ausnahmeverarbeitung durch einen RTE-Befehl (= ReTurn from Exception) abgeschlossen. Die Routine hat nach bestem Wissen und Gewissen alle Aufgaben erledigt, für die sie gedacht war; jetzt geht es ans große Stackbereinigen: 68010 und 68020 können am Formatcode erkennen, wieviele Worte sie vorher auf den Stack geschaufelt haben, der 68000 vermutet grundsätzlich nur Statusregister und Programmzähler. Beim Regenerieren des internen Zustandes nimmt es der 68000, wie es gerade kommt, die anderen beiden Prozessoren gehen etwas differenzierter zu Werke:

- Statusregister und Programmzähler werden vom Stack gelesen. Damit ist der 68000 fertig und holt den nächsten Befehl von der Adresse, auf die der Programmzähler zeigt.
- 68010 und 68020 lesen den Formatcode und überprüfen, ob er für den Prozessor überhaupt definiert ist – wenn nicht, dann folgt umgehend eine weitere Ausnahmeverarbeitung über Vek-

tor 14 (Formatfehler). Da der Wert des Stackpointers bis zu diesem Zeitpunkt noch nicht geändert wurde, wird die gesamte zum Fehler führende Stack-Information nicht überschrieben und steht für eine Analyse zur Verfügung.
- Falls der Formatcode das kurze Stackformat (vier Worte) anzeigt, sind auch 68010 und 68020 fertig mit der Ausnahmeverarbeitung und fahren normal im Programm fort.
- Findet der 68020 das kurze Dummy-Format 0001, so lädt er nur das Statusregister vom Stack und führt sofort einen weiteren RTE-Befehl aus. Welcher Stack dabei verwendet wird, bestimmen das M- und S-Bit des gerade gelesenen Statuswortes, es darf also jeder sein.
- Bei einem Busfehler-Format versucht die CPU, das erste Wort auf dem Stack zu lesen. Wenn dabei kein Busfehler auftritt, glaubt der Prozessor (mit einer gewissen Berechtigung), daß die dazwischenliegenden Daten auch alle lesbar sind; im anderen Fall, Sie ahnen es schon, setzt es selbstverständlich eine Busfehler-Ausnahme. Es scheint hier fast, als ob man aus einer Ausnahmeverarbeitung gar nicht mehr herauskommt, wenn man erst einmal damit angefangen hat, aber das scheint eben nur so.
- Die CPU pumpt nun eifrig die nächsten Worte vom Stack und stoppt erst wieder bei einer nicht näher spezifizierten internen Information (SP + 26 in Bild 3, SP + 54 beim 68020). Darin enthalten ist die Versionsnummer des Prozessors, der die Daten auf den Stack geladen hat, und die muß mit der Nummer des lesenden Prozessors übereinstimmen, damit die noch folgenden Stackdaten richtig interpretiert werden. Sollte dem nicht so sein, dann haben wir wieder einen Formatfehler (Sie wissen schon, Vektornummer 14). Das ist z. B. in Multiprozessorsystemen wichtig.
- Wenn wir wirklich bis hierher gekommen sind, ohne eine weitere Ausnahme provoziert zu haben, werden schnell noch die restlichen Daten vom Stack geholt und erst dann wird der Inhalt des Supervisor-Stackpointers korrigiert – bis zu diesem Zeitpunkt hätten ja immer noch Busfehler auftreten können mit der ganzen schönen dazugehörigen Ausnahmeverarbeitung und so fort.
- Jetzt sind wir endgültig fertig.

Soviel zum Normalfall. Die zweite Möglichkeit, eine Ausnahmeverarbeitung zu beenden, funktioniert völlig ohne RTE-Befehl. Dann aber ist der Programmierer der Ausnahmeroutine dafür verantwortlich, daß der Stack bereinigt und die CPU in einen gebrauchsfähigen Zustand versetzt wird. Das bedeutet konkret: Stackpointer um die entsprechende Anzahl Worte korrigieren (wenn Sie jetzt nicht wissen, wie viele, dann setzen Sie einmal beim Würfeln aus und gehen drei Kapitel zurück), das Statusregister mit einem vernünftigen Wert laden und anschließend normal irgendwo im Programmspeicher fortfahren. Die Konsequenz ist natürlich, daß das ursprünglich unterbrochene Programm nicht weitergeführt wird, aber das ist ja vielleicht auch beabsichtigt – so ungefähr nämlich funktioniert Multitasking.

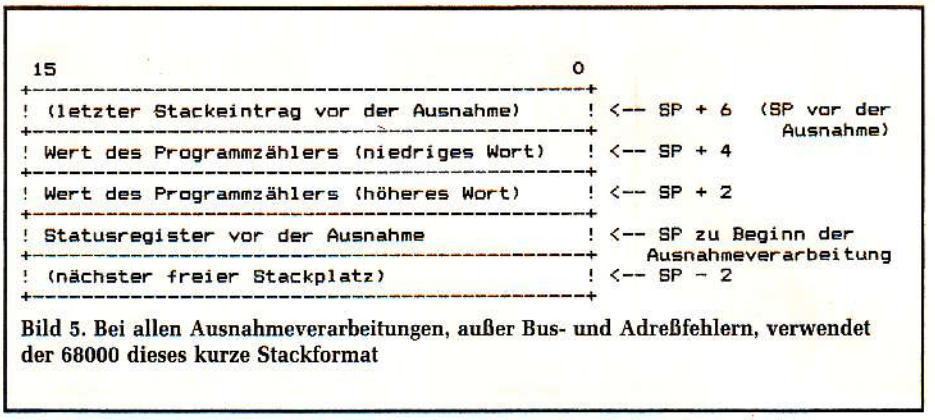

Bild 5. Bei allen Ausnahmeverarbeitungen, außer Bus- und Adreßfehlern, verwendet der 68000 dieses kurze Stackformat

Hardware-Voraussetzungen

Wenn man den prinzipiellen Aufbau eines Mikrocomputers betrachtet, können fünf verschiedene Funktionsgruppen unterschieden werden:

(1) Das Gehirn des Ganzen besteht aus der CPU und eventuell vorhandenen Coprozessoren (Arithmetikeinheit, DMA-Controller).

(2) Um das Rechnersystem überhaupt starten zu können, muß ein nichtflüchtiger Speicher vorhanden sein, z. B. ein EPROM oder ein batteriegepuffertes RAM. Häufig werden darin zusätzlich ein Monitor, ein Betriebssystem (komplett oder teilweise) und ein Hochspracheninterpreter (nein, ich sage nicht, welcher!) abgelegt.

(3) Bei Kleinstanwendungen könnte man sich vorstellen, allein mit den CPU-Registern auszukommen. Aber normalerweise wird ein Schreib-Lese-Speicher für irgendwelche Variable und den (System-)Stack vorhanden sein.

(4) Damit der Rechner überhaupt einen Sinn bekommt, muß er irgendwie mit der Außenwelt in Verbindung treten können, etwa über eine Tastatur-Schnittstelle, den Disk-Controller usw.

(5) Das Knochengerüst, das die ganze Chose zusammenhält, wird in den allermeisten Fällen äußerst stiefmütterlich behandelt; es gehört schon riesiges Glück dazu, wenn in offiziellen Verlautbarungen zu einem Computer etwas über Adreß-Decodierung, Power-On-Schaltung oder Interruptbehandlung enthalten ist.

Betrachten wir jetzt einmal den Aufbau unseres Mikrocomputers unter einem etwas ungewöhnlichen Gesichtspunkt: Wie optimiere ich die Hardware bezüglich Ausnahmeverarbeitung? Das Problem ist mit unserem jetzigen Wissensstand nicht mehr sehr schwer zu lösen, da die CPU ja alle Randbedingungen vorgibt. Wir müssen sie nur im Design der Hardware gebührend würdigen.

Anforderungen an die Adreßdecodierung

Nach einem externen Reset, etwa beim Einschalten des Systems, liest der Prozessor als erstes unter den Adressen 0...7 den Startwert des Supervisor-Stackpointers und des Programmzählers. Es bietet sich daher an, auf den untersten Adressen den EPROM-Bereich anzuordnen, da der Programmzähler üblicherweise auch auf eine Initialisierungsroutine im EPROM zeigt. Der Startwert des Stackpointers muß natürlich auf einen RAM-Bereich zeigen. In einem modernen Computersystem wäre es schön, wenn das verfügbare RAM erst während der Initialisierungsphase gesucht und erst dann der Stackpointer geladen würde. Dummerweise aber müssen alle RAM-Suchroutinen die Busfehler-Ausnahme

benutzen, und dazu benötigt man bereits den Stack.

Womit wir ganz zwanglos bei der Tabelle der Ausnahmeverarbeitungsvektoren wären. Bei einem anständigen Computer gehört diese Tabelle ins RAM, unwiderruflich und ohne Ausflüchte! In 99 % aller Fälle werden einige Vektoren vom Betriebssystem oder vom Anwender während des Programmlaufs verändert; wenn dann die Ausnahmeverarbeitung erst über eine vordefinierte Tabelle im EPROM auf eine Hilfsroutine (meist auch im EPROM) verzweigt, die ihrerseits eine zweite Tabelle (diesmal im RAM) mit den derzeit gültigen Vektoren benötigt, um die eigentliche Ausnahmeverarbeitungsroutine zu finden, dann hat der Hardware-Designer ganz entschieden fünf Minuten zu wenig nachgedacht. Speziell wenn Interrupts im System herumgeistern, besteht der Witz ja gerade darin, die gesamte Unterbrechung so kurz wie möglich zu halten und die Reaktion auf dem schnellsten Weg auszulösen.

In 68010- und 68020-Systemen bereitet es keine Mühe, die Tabelle im RAM abzulegen, da die Prozessoren dieses hübsche Vektorbasisregister besitzen. Es können sogar mehrere Tabellen vorhanden sein, um blitzschnell den Kontext zu wechseln. Etwas aufwendiger gestaltet sich die Angelegenheit, wenn ein 68000 das Sagen hat. Im linearen Adreßbereich der CPU schließen die Ausnahmevektoren nahtlos am Resetvektor an – und der liegt ja nun schlauerweise im EPROM. Diese widersprüchliche Forderung ist wohl auch schon den Erfindern des 68000 aufgefallen, deshalb liegt der Resetvektor im Supervisor-Programmbereich und alle übrigen Ausnahmevektoren im Supervisor-Datenbereich. Durch Berücksichtigung der Funktionscode-Leitungen in der Adreß-Decodierung kann deshalb dieses Problem leicht aus der Welt geschafft werden. Eine andere, häufig angewandte Methode besteht darin, nach der Initialisierungsphase den EPROM-Bereich per Software auszublenden und durch einen RAM-Bereich zu ersetzen. Diese Lösung ist hardwaremäßig jedoch aufwendiger. Die letzte (und billigste) Möglichkeit, nämlich das eingangs erwähnte Führen einer zweiten Tabelle, ist keine Lösung, sondern ein Krampf.

Wer träumt nicht davon, seinen Rechner im vollen Adreßbereich mit Speicher auszustatten (abzüglich einer unbedeutenden Handvoll I/O-Adressen)? Abgesehen vom zur Zeit nicht unbeträchtlichen finanziellen Aufwand möge der geneigte Leser seinen Blick nochmals auf den ersten Teil dieses Beitrages richten: Die obersten 16 Adressen sind für die Interrupt-Bestätigung reserviert! Was glauben Sie, wird die CPU unternehmen, wenn während des Pseudo-Lesezyklus einer Interrupt-Bestätigung die Hardware ordnungsgemäß eine Vektornummer auf den Datenbus legt und gleichzeitig Ihre neuerworbene 4-MByte-RAM-Karte ihren Senf dazugibt, nur weil sie sich ebenfalls angesprochen fühlt? Die Frage ist leicht zu beantworten: Was auch immer Sie erwarten, die CPU wird sicher etwas anderes tun, nämlich Ihnen etwas husten. Und die Moral von der Geschichte: Man meide den obersten Adreßbereich, es sei denn – und hier kommt wieder die geniale Lösung – die Adreß-Decodierung berücksichtigt die Funktionscode-Leitungen (Tabelle 2, erster Teil), damit die Speicherkarte brav den Mund hält, wenn der CPU-Bereich dran ist.

Merke: Die Funktionscode-Leitungen sind nicht nur purer Luxus, sie können bei der Adreß-Decodierung an Stellen benötigt werden, an die man zunächst gar nicht gedacht hat.

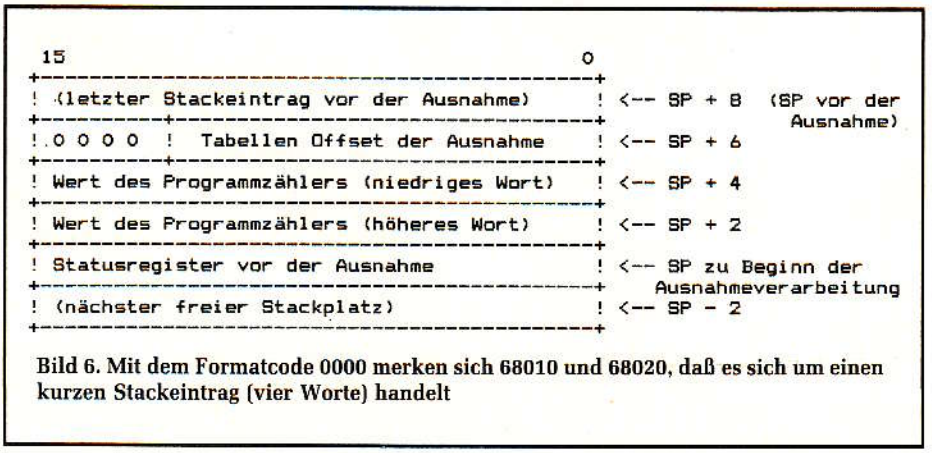

Bild 6. Mit dem Formatcode 0000 merken sich 68010 und 68020, daß es sich um einen kurzen Stackeintrag (vier Worte) handelt

Diese Überlegungen gelten für einfache Computersysteme; über die Komplikationen, die etwa durch den Einsatz einer MMU (Memory Management Unit) mit der Übersetzung von logischen in physikalische Adressen entstehen, wollen wir hier großzügig schweigen.

Reset- und Busfehler-Signal

Es gibt zwei Fälle, in denen ein externer Reset des Computers nötig wird: Beim Einschalten der Betriebsspannung und wenn ein entnervter Benutzer eine entsprechend bezeichnete Taste betätigt. Ein Busfehler tritt auf, wenn die CPU eine Adresse auf den Adreßbus gelegt hat und sich nach einer gewissen Zeitspanne noch niemand angesprochen fühlt. Sowohl beim Reset als auch bei Busfehlern muß ein Signal mit gewissen Eigenschaften erzeugt werden.
Dieser Aufgabe kann man sich sauber und elegant unter Verwendung von Zählern und Gattern entledigen (synchrone Methode) oder mit Hilfe von Monoflops (wer das tut, weiß genau, daß er sich eigentlich schämen müßte). Wie auch immer, diese beiden Signale sind lebenswichtig für den Mikrocomputer und dürfen in keinem System fehlen.

Interrupt-Erzeugung

Die meisten Überlegungen zum Hardware-Design müssen sicherlich in die Handhabung von Interrupts gesteckt werden. Ein Einplatinenrechner wird vielleicht nur einen oder zwei Interruptquellen haben; ein größeres System kann viele Quellen besitzen, die sich dann verschiedene Interrupt-Level teilen müssen. Je nach Intelligenz der Peripheriebausteine ist eine Erzeugung von externen Interruptvektoren einfach zu implementieren oder man muß mit Autovektoren auskommen.

Verhältnismäßig einfach liegen die Dinge, wenn weniger als sieben (bzw. drei beim 68008) unterschiedliche Interruptquellen vorhanden sind. Dann kann jedem Erzeuger eine Interruptstufe zugeteilt werden, ohne daß sich zwei gegenseitig stören. Level 7 (nicht maskierbar) sollte man nur verwenden, wenn man genau weiß, warum. Bei solch kleinen Systemen ist es normalerweise völlig ausreichend, mit Autovektoren zu arbeiten. Der Hardware-Designer kann sich das Leben einfach machen, denn er muß während der Interrupt-Bestätigung nur ein $\overline{\text{VPA}}$-Signal erzeugen. Wird ein 68010 oder 68020 eingesetzt, so müssen in dieser Decodierung neben den Funk-

tionscode- auch die Adreßleitungen ab A4 aufwärts enthalten sein, da diese Prozessoren mit dem Funktionscode 111 noch andere Tätigkeiten signalisieren. Die zeitliche Referenz dieser Signale wird durch den Adreß-Strobe (AS) geliefert.

Diese einfache Erzeugung des VPA-Signals hat noch einen großen Vorteil: Es können durchaus mehrere (hardwaremäßig unterschiedliche) Interrupts auf demselben Level auftreten. Durch Verwendung von Treibern mit offenen Kollektoren können problemlos mehrere Quellen auf derselben Interruptleitung zusammengeschaltet werden. Die Bestätigung des Autovektors erfolgt zentral, und es ist Aufgabe der Interrupt-Service-Routine, die in Frage kommenden Quellen abzufragen und zu bedienen. Im Grunde genommen handelt es sich hier um eine Kombination aus Interrupt und Polling. Der Nachteil besteht darin, daß das Ganze eben nur mit Autovektoren funktioniert.

In großen Computersystemen mit sehr vielen Interruptquellen kann das Abfragen aller Außenstellen, die auf dem gleichen Level liegen, zu unerwünscht langen Verzögerungen führen. In diesem Fall ist der Vektorinterrupt vorzuziehen, der von den meisten komplexen Peripheriebausteinen der Familie 680XX unterstützt wird. Die Zeiteinsparung in der Software muß jedoch mit einem erhöhten Aufwand in der Hardware ausgeglichen werden. Da nun jede Quelle ihre eigene Vektornummer abliefert, wird ohne zusätzlichen Verwaltungsaufwand gleich in die richtige Ausnahmeverarbeitung verzweigt. Der Zeitgewinn kann beträchtlich sein.

Vektor-Komplikationen

Alle denkbaren Kombinationen lassen sich auf den Fall zurückführen, daß zwei Vektorinterrupts gleichzeitig auftreten. Wenn sie verschiedene Prioritäten besitzen, ist das Problem schnell gelöst: Für jede Interruptquelle muß die Bestätigung durch die CPU separat und vollständig decodiert werden, das bedeutet die zusätzliche Auswertung der Adreßleitungen A1...A3, die den akzeptierten Level angeben. Dadurch wird sichergestellt, daß nur die Quelle mit der höchsten Priorität ihre Vektornummer auf den Datenbus legt.

Wenn zusätzlich noch Autovektorinterrupts auf anderen Leveln möglich sind, muß die Erzeugung des VPA-Signals selbstverständlich auch individuell für

```
 15                                                  0
+-------------------------------------------------+
! (letzter Stackeintrag vor der Ausnahme)          ! <-- SP + 12 (SP vor der
+----------+--------------------------------------+                 Ausnahme)
! Adresse der Instruktion, (niedriges Wort)        ! <-- SP + 10
+-die die Ausnahmebedingung----------------------+
! herbeigeführt hat        (höheres   Wort)        ! <-- SP + 8
+----------+--------------------------------------+
! 0 0 1 0  !  Tabellen Offset der Ausnahme         ! <-- SP + 6
+----------+--------------------------------------+
! Wert des Programmzählers (niedriges Wort)       ! <-- SP + 4
+-------------------------------------------------+
! Wert des Programmzählers (höheres Wort)         ! <-- SP + 2
+-------------------------------------------------+
! Statusregister vor der Ausnahme                 ! <-- SP zu Beginn der
+-------------------------------------------------+     Ausnahmeverarbeitung
! (nächster freier Stackplatz)                    ! <-- SP - 2
+-------------------------------------------------+
```

Bild 7. Diese Stackeinteilung verwendet der 68020, wenn es sich um eine Ausnahme handelt, die per Befehl eingeleitet wurde

jede Priorität, also unter Beachtung der Adreßleitungen A1...A3 erfolgen. Sonst wird während des Vektorlesezyklus die besagte Einfach-Decodierung vorschlagen, man könne doch genauso gut den Autorvektor nehmen.

Das Ganze ist natürlich ein Schuß in den Ofen, wenn die zwei Vektorinterrupts auf derselben Prioritätsstufe liegen. Dann fühlen sich während der Bestätigung beide angesprochen, und die CPU erhält so etwas wie das logische Und der beiden Vektornummern, was dann weiter geschieht, muß wohl nicht näher erläutert werden. Die Lösung des Problems ist wahrhaftig nicht neu, stammt sie doch aus den Anfängen der Computerei, als eine anständige CPU sowieso nur einen Interrupt-Eingang besaß: Alle Quellen werden in einer Reihe hintereinander geschaltet (figürlich natürlich), wobei der wichtigste Kandidat vorne steht. Wenn die Interruptbestätigung der CPU kommt, schaut der Vorderste nach, ob er überhaupt ein Interrüptchen angefordert hatte. Wenn ja, dann legt er seinen Vektor auf den Datenbus und beendet den Pseudo-Lesezyklus des Prozessors. Wenn nicht, dann reicht er die schöne Bestätigung einfach an seinen Hintermann weiter. Und schon haben Sie ganz zwanglos begriffen, was „Daisy Chain" (= „Gänseblümchen-Kette") bedeutet.

Wenn Sie sich beim Aufbau Ihres Rechners einfach nicht für eine bestimmte Reihenfolge der Interruptquellen entscheiden können, oder anders ausgedrückt, wenn die totale Gleichberechtigung unter allen Kandidaten ausgebrochen ist, kann man den Aufwand auch noch etwas weiter treiben. Jeder, der seinen Interrupt gehabt hat, ist im Anschluß daran das Schlußlicht in der Abfrage-Kette. So etwas nennt man „Round Robin"-(= „Ringelreihn"-)Methode. Obwohl diese Begriffe aus der Computersteinzeit stammen, sind sie absolut modern. Beispielsweise schlagen sich Konstrukteure lokaler Netzwerke mit genau dem gleichen Problem herum: Wie vermeide ich Chaos auf meinem Bus.

Um den Hardware-Aufwand wirklich maximal zu gestalten, können jetzt sieben solcher Ketten aufgebaut werden, für jede Prioritätsstufe eine. Man darf natürlich auch Interruptquellen einordnen, die „nur" autovektorfähig sind, der CPU ist das ziemlich egal. Aber wie gesagt: So etwas geht auch einfacher, wenn nur Autointerrupts für die betreffenden Prioritäten vorgesehen sind.

Faustregeln

Bei wenigen Interruptquellen im System werden nur Autovektoren verwendet. Für die Decodierung des gemeinsamen VPA-Signals (AVEC beim 68020) werden unbedingt die Funktionscode-Leitungen benötigt. Soll das System für 68010 und 68020 tauglich sein, so müssen zusätzlich die Adreßleitungen A4...A23 (...A31 beim 68020) mit decodiert werden. Autovektoren funktionieren auch bei mehreren Interrupt-Quellen auf der gleichen Prioritätsstufe.

Wenn Vektorinterrupts möglich sind, muß die Interruptbestätigung vollständig decodiert werden, d. h. unter Berücksichtigung des akzeptierten Levels auf den Adreßleitungen A1...A3. Bei vielen Quellen im System bringen Vektorinterrupts großen Zeitgewinn. Es muß aber erhöhter Hardware-Aufwand getrieben werden, damit sich im Interrupt-Bestätigungszyklus auch wirklich nur eine Quelle meldet.

Programmierers Lust und Frust

Aus der Sicht des Programmierers müssen für jede Ausnahmeverarbeitung zwei völlig getrennte Phasen unterschieden werden: Bevor die Ausnahme überhaupt auftreten darf, muß eine entsprechende Routine im System installiert sein, die dann die Reaktion auf die Ausnahme vornimmt. Vor allem muß ein Zeiger auf den Start dieses Programmstücks in die Vektortabelle eingetragen sein.

Die zweite, davon unabhängige Phase beginnt jedesmal dann, wenn die Ausnahmebedingung wirklich auftritt. Die CPU führt nun alle Schritte aus, die sich der Programmierer vorher so mühevoll überlegt hat, und mit etwas Glück entspricht das Resultat sogar seinen Erwartungen.

Initialisierung der Ausnahmeverarbeitung

Da die Einleitung wieder so kompliziert klingt, hier ein einfaches Beispiel: Jeder Computer benötigt ein Startprogramm, das er nach dem Einschalten der Stromversorgung durchläuft. Aus relativ einleuchtenden Gründen befindet sich dieses Programm in einem nicht flüchtigen Speicher (z. B. im EPROM). Die Anfangsadresse des Programmes ist gerade der Startwert des Programmzählers und der ist ja Bestandteil des Reset-Vektors in der Vektortabelle. Die Initialisierungsphase ist hier also während des Zusammenbaus des Computers erfolgt, nämlich indem das EPROM programmiert und in seinen Sockel gesteckt wurde.

In vielen Fällen besteht die erste und wichtigste Aufgabe des Startprogramms im Initialisieren der übrigen Ausnahmevektoren. Um nur einige zu nennen: die Vektoren für Busfehler und Spurious Interrupt werden mit den Adressen geeigneter (eventuell provisorischer) Prozeduren geladen, die sich häufig auch im EPROM befinden, denn noch bevor das Betriebssystem hochgefahren ist, können diese Ausnahmen hereinfunken. Später kann das Betriebssystem die Zeiger ja immer noch auf eine schönere und komfortablere Routine umlenken, denn die letztendlich wirksame Tabelle mit den Ausnahme-Vektoren liegt ja wie gesagt im RAM, auch beim 68000, unwiderruflich!

Dem Programmierer stellen sich nun zwei Probleme: Zum einen muß er herausfinden, ab welcher Adresse im Speicher der Loader oder Linker des Betriebssystems seine eben geschriebene Ausnahmeverarbeitungsroutine abgelegt hat. In vielen Fällen ist dazu eine intime Kenntnis des Betriebssystems notwendig, andererseits gibt es auch Hochsprachen, die dieses Begehren perfekt unterstützen. Nehmen wir zum Beispiel Modula-2: Es gibt dort den wunderschönen Typ „Prozedur" und einer Variablen dieses Typs kann man dann die Ausnahmeverarbeitungsroutine zuweisen. Wenn die Gedankengänge des Compilerbauers nicht allzu hinterlistig waren, besteht die interne Darstellung dieses Typs aus einer 32-Bit-Zahl – und zwar gerade aus der gesuchten Startadresse der Routine. Die zweite Schwierigkeit besteht im Finden der Vektortabelle. Bei 68010 und 68020 gibt es das Vektorbasisregister, dem man zumindest auf Assembler-Ebene zu Leibe rücken kann, und das Problem ist gelöst. Mit dem 68000 ist es etwas mühsamer. Wenn keine weiteren Informationen über den Aufbau des Betriebssystems vorliegen und wenn die ursprüngliche Vektortabelle doch im EPROM steht, kann man sich mit Hilfe eines Disassemblers aus der Affäre ziehen. Soll etwa der Interrupt-Autovektor Nr. 2 installiert werden, dann schaut man zuerst nach, welche Adresse in den Speicherplätzen 104...107 (Tabelle 1, 1. Teil) abgelegt ist. Von da aus gelangt man durch schrittweises Vorgehen und Disassemblieren irgendwann zu einem Programmstück, das die endgültige Referenz auf die RAM-Vektortabelle enthält. Während damit für die meisten Ausnahmen alle Vorbereitungen beendet sind muß ein Interrupt noch speziell freigegeben werden. In vielen Fällen, besonders bei intelligenten Peripheriebausteinen, sagt man zunächst der Peripherie Bescheid, daß sie jetzt darf, wenn sie will. Anschließend wird die Interrupt-Maske im Statusregister der CPU so gesetzt, daß der Interrupt die Chance bekommt, gehört zu werden. Das geschieht nun nicht etwa, indem man ganz brutal die gewünschte Priorität ins Register donnert (ein sehr beliebter Fehler), sondern man sieht tunlichst nach, welche Priorität eigentlich gerade zugelassen ist. Wenn nämlich bereits früher ein Interrupt niedrigerer Priorität freigegeben worden ist, würde dem durch die Holzhammer-Methode ziemlich herzlos das Wasser abgegraben, und so etwas tut man doch nicht, oder?

Im Fall eines einigermaßen intelligenten Peripheriebausteins (etwa aus der Familie 68xxx oder 63xxx) muß eventuell noch dessen Interruptvektorregister mit der gewünschten Vektornummer geladen werden. Für nicht initialisierte Vektorinterrupts ist in der Tabelle 1, 1. Teil, die Nummer 15 reserviert und alle Mitglieder der Familie 68xxx geben nach einem Reset diese Vektornummer ab, solange sie nicht eines Besseren belehrt wurden.

Können, Dürfen und vor allem Lassen

Tritt die Ausnahmebedingung nun wirklich einmal auf, ob gewünscht oder gefürchtet, so landet die CPU über kurz oder lang in der ordentlich installierten Routine. Was sie da tut, hängt natürlich von der Aufgabenstellung ab. Aber ein paar allgemeine Richtlinien sind sicher angebracht.

Als erstes muß der Inhalt aller CPU-Register, die in der Routine selbst benötigt werden, gerettet werden, und zwar auf den Stack. Sehr häufig findet man den Fehler, daß ein Registerinhalt in einer festen Speicheradresse abgelegt wird. Und da man ja ein bißchen haushalten muß, nutzt eine andere Ausnahmeverarbeitung diesen Speicherplatz gleich mit. Das geht dann vielleicht sogar eine ganze Zeit lang gut, nämlich solange beide Routinen auf der gleichen Priorität liegen und sich somit gegenseitig nicht unterbrechen können, aber wenn man dann nach Wochen die Priorität der einen Routine ändert, ergeben sich plötzlich ganz unerklärliche Fehler...

Natürlich müssen vor Beendigung der Ausnahmeroutine die Register wieder in ihren alten Zustand versetzt werden, wenn möglich ohne sie zu verwechseln. Durch Verwendung des MOVEM-Befehls kann fast nichts schiefgehen. Apropos Stack: Wenn jemand glaubt, er müsse das Adreßregister A7 für etwas anderes verwenden und könne den Inhalt mal schnell zwischenspeichern, dann ist er selber schuld. Sowie eine Ausnahmebedingung auftritt, fängt der Prozessor an, mit Hilfe des Supervisor-Stackpointers irgendwelche Daten zu retten und es ist ihm dabei ziemlich egal, wohin A7 in diesem Moment zeigt. Oft werden in Ausnahmeverarbeitungen gewisse Informationen vom Stack benötigt. Dazu hat man bequemen Zugriff mit Push und Pop, indirekter, indizierter, relativer oder sonst irgendeiner Adressierung, aber niemals sollte der Stackpointer direkt verändert werden. Notfalls kopiert man A7 in ein anderes Register, das man dann ungestraft manipu-

lieren darf. Mit dem Link-Befehl spart man dabei das vorherige Retten dieses Registers. Mit dem Anwender-Stapelzeiger kann man natürlich beliebig (oder jedenfalls beinahe) herumturnen, aber bei System- oder Interrupt-Stackpointer sind wir nur Untermieter, und die Hausordnung ist streng!

Merke: Die nächste Ausnahme kommt bestimmt und normalerweise gerade dann, wenn man sie am wenigsten erwartet. Wenn dann der Stackpointer nicht auf den Stack zeigt, dann geht die Angelegenheit mit ziemlicher Sicherheit in die H§$&.

Sofern die Hardware nicht schon dafür sorgt, muß eine Interrupt-Service-Routine veranlassen, daß die Quelle ihre Anforderung auch zurücknimmt, im anderen Falle befindet man sich nach dem Ende der Ausnahmeverarbeitung gleich wieder an deren Anfang und das geht so weiter, bis jemand die Geduld verliert. Bei einem Bus- oder Adreßfehler und einem Prozessor 68000 muß man den Stack selbst in Ordnung bringen.
Noch ein Wort zum Schluß: Niemand verlangt, daß eine Ausnahmeverarbeitungsroutine komplett in Assembler geschrieben wird. Beim Initialisieren, Retten der Registerinhalte, Reinstallieren sowie beim RTE-Befehl wird es sich nicht vermeiden lassen, aber sonst spricht (außer vielleicht Zeitgründen) nichts gegen die Verwendung von Hochsprachen. Vorausgesetzt, man weiß, wohin die Prozedur oder Funktion vom Betriebssystem geladen wurde.

Polling von Interrupt-Quellen

Sehr häufig tritt der Fall ein, daß für einen Interrupt mehrere verschiedene Quellen in Betracht kommen, sei es, daß ein komplexer Peripheriebaustein mehr als eine mögliche Ausnahmebedingung herbeiführen kann oder daß sich verschiedene Hardware-Quellen die gleiche Priorität teilen. Damit die Ausnahmeverarbeitungsroutine den eigentlichen Verursacher des Interrupts herausfinden kann, werden nacheinander die Statusregister der einzelnen Kandidaten abgefragt. Auch dabei schleicht sich schnell ein bestimmter Fehler ein: Wenn ein spezielles Statusbit (etwa für eine Handshake-Leitung in 6522 oder 68230) gesetzt ist, bedeutet das noch nicht, daß der Interrupt von diesem Baustein kam; ausschlaggebend ist in diesem Fall, ob dieser Interrupt überhaupt durch die Baustein-interne Maske freigegeben war, und das muß der Programmierer nun leider selbst überprüfen. Wer hier nicht aufpaßt, läßt Interrupts bearbeiten, die eigentlich gar nicht durchkommen sollten.

Ein anderer beliebter Fehler ist es, beim Bearbeiten eines Interrupts aus Versehen gleich eine zweite Anforderung im gleichen Baustein zu löschen. Also: In jeder Service-Routine nur die Quelle zurücksetzen, die auch wirklich schon bedient wurde. Der gleiche Grundsatz gilt natürlich auch für die Freigabe in der Interrupt-Maske des Bausteins; es ist gefährlich, immer pauschal alle möglichen Quellen freizugeben oder zu sperren, irgend ein Programmstück hat ja vielleicht gute Gründe, einen Interrupt zeitweilig abzuklemmen oder zu erlauben.
Durch die Reihenfolge, in der die möglichen Quellen abgefragt werden, ist es auf einfache Art möglich, Software-Prioritäten zu implementieren: Die wichtigsten Interrupts werden immer zuerst geprüft.

Unterschiede

Jetzt sind wir auch soweit, die eingangs gestellte Frage nach den Unterschieden zwischen den einzelnen Familienmitgliedern bezüglich Ausnahmeverarbeitung beantworten zu können. Von allen Prozessoren wird im wesentlichen die gleiche Hardware benötigt (von den verschiedenen Busbreiten einmal abgesehen) und wenn von Anfang an die Interrupt-Bestätigung vollständig decodiert wird (Funktionscode- und alle Adreßleitungen), kann gefahrlos ein Prozessor gegen den anderen ausgetauscht werden. Nicht ganz so einfach liegen die Verhältnisse bei der Software. 68010 und 68020 sind sehr gut miteinander verträglich, aber es bestehen ziemlich starke Abweichungen zum 68000. Unterschiede und Gemeinsamkeiten lassen sich wie folgt zusammenfassen:

(1) Das fehlende Vektorbasisregister des 68000 erfordert eine andere Initialisierung der Ausnahmen.

(2) Das Stackformat des 68000 bei Bus- und Adreßfehlern ist grundsätzlich anders aufgebaut. Zur Fehlerbehandlung werden aber gerade die Informationen über die Ursache vom Stack gebraucht. Auch 68010 und 68020 besitzen unterschiedliche Stackformate und retten andere interne Zustände. Diese Ausnahmeverarbeitungen müssen also für jede CPU individuell programmiert werden.

(3) Bei allen anderen Ausnahmen werden Statusregister und Programmzähler an den gleichen Stellen auf dem Stack abgelegt (relativ zum Stackpointer). Wenn keine weiteren Informationen benötigt werden, sind dieselben Verarbeitungsroutinen auf allen Prozessoren lauffähig.

Zusammenfassend: Wie zu erwarten war, sind die Mikroprozessoren der Familie 680xx untereinander und übereinander kompatibel, aber...

Spruch des Monats

„Es sieht so aus als mache die Wirklichkeit einen Umweg über die Träumer."

Titel eines Buches von Jörg Löhken, das im Verlag „Unter uns" erschienen ist

Literatur:

[1] Datenblatt MC68008, Motorola 1982
[2] Datenblatt MC68000, Motorola 1985
[3] Datenblatt MC68010, Motorola 1983
[4] MC68020 Users Manual, Second Edition, Motorola 1985
[5] *Hilf, Werner/Nausch, Anton:* MC68000-Familie, Teil 1 und 2, Te-Wi-Verlag München 1984
[6] *Kane, Gerry:* 68000 Mikroprozessorhandbuch, McGraw-Hill Book Company GmbH, Hamburg 1981
[7] *Grohmann, Bernd/Eichler, Lutz:* Das Prozessorbuch zum 68000, Data Becker, Düsseldorf 1985
[8] *Johnson, Thomas L.:* A Comparison Of MC68000 Family Processors, Byte 9/1986, Seite 205