Mikrocontroller-Grundlagen

Teil 15

Im fünfzehnten Teil dieser Artikelserie erläutern wir ausführlich die Interrupt-Struktur der MCS51-Familie.

6 Interrupts

Interrupts sind Programmunterbrechungen, die die Abarbeitung des laufenden Programmcodes vorübergehend stoppen. Sie werden ausschließlich durch Hardwareereignisse ausgelöst, worauf der Prozessor mit der Abarbeitung der Interrupt-Service-Routine beginnt. Interrupts werden vornehmlich dort eingesetzt, wo es um eine schnelle Reaktion auf bestimmte Ereignisse geht. Ein weiterer Einsatzfall liegt in dem Auslösen eines regelmäßigen Timer-Interrupts.

Ein typisches Beispiel für die erstgenannte Anwendung ist die Reaktion auf einen Tastendruck, der in der Regel eine sofortige Reaktion des Programmes erfordert. Das Hauptprogramm, welches beispielsweise gerade eine Berechnung ausführt, wird zwar kurzfristig in seiner Funktion unterbrochen, arbeitet aber nach Beendigung der Programmunterbrechung an der ursprünglichen Stelle weiter.

Stünde für einen solchen Einsatzfall keine Interrupt-Möglichkeit zur Verfügung, so müßte das Hauptprogramm seine Arbeit ständig unterbrechen und die Abfrage möglicher externer oder interner Ereignisse vornehmen, was nicht nur speicherplatzintensiv ist, sondern das Programm auch unübersichtlicher macht.

Die regelmäßigen Timer-Interrupts dienen z. B. zur Ansteuerung von gemultiplexten 7-Segment-Anzeigen, deren regelmäßige Ansteuerung unbedingt erforderlich ist.

Es gibt fast kein Programm, das ohne eine Interrupt-Routine auskommt. Das führt unter Umständen sogar so weit, daß im Hauptprogramm keine Aktivitäten mehr ablaufen und alle Arbeiten über Interrupts ausgeführt werden.

6.1 Interrupt-Ausführung

Abbildung 126 zeigt die komplette Interrupt-Logik der MCS51-Familie, die sich über verschiedene bit- und byte-adressierbare Register steuern läßt. Der 8031/51 besitzt 5 Interrupt-Quellen (2 externe und 3 interne).

Die externe Interrupt-Auslösung erfolgt durch die Prozessor-Abschlußpins $\overline{INT0}$ und $\overline{INT1}$, die eine Unterbrechungsanforderung mit fallender Flanke oder Low-Pegel auslösen. Die Auslösung der internen Interrupts erfolgt mit dem Überlauf eines der beiden Timer oder über die serielle Schnittstelle. Beim 8032/52 kommt jeweils noch ein externer bzw. interner Interrupt hinzu.

Jeder der externen Interrupts $\overline{INT0}$ und $\overline{INT1}$ kann, wie Abbildung 126 zeigt, entweder pegel- oder flankengetriggert sein, welches von den Bits IT0 und IT1 gesteuert wird. Die Steuerbits IT0 und IT1 sowie IE0 und IE1 sind in dem bit- und byteadressierbaren Timer-Control-Register TCON untergebracht, das in Tabelle 20 („ELVjournal" 2/95, Seite 84) bereits beschrieben wurde.

Ist das IT0- bzw. IT1-Bit = 0, ist der Interrupt Pegel-getriggert und wird ausgelöst solange Low-Pegel an dem $\overline{INT0}$- bzw. $\overline{INT1}$-Pin anliegt. Bei gesetztem IT0 bzw. IT1-Bit erfolgt die Interrupt-Auslösung mit der negativen Flanke an dem entsprechenden Anschlußpin.

Die Flankensteilheit des Signals sollte eine Mikrosekunde nicht überschreiten.

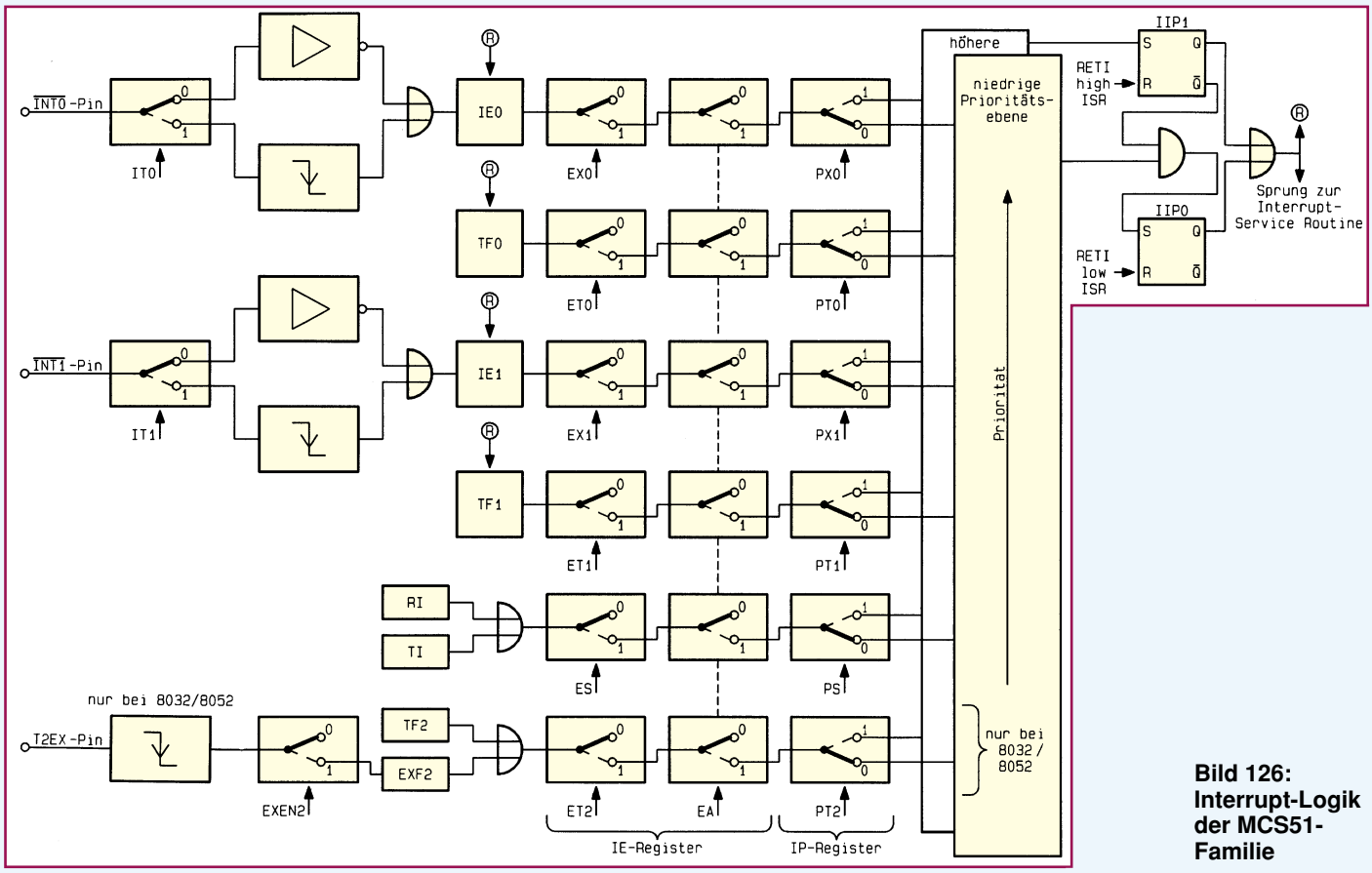


Bild 126: Interrupt-Logik der MCS51-Familie

Die High- bzw. Low-Pegel müssen jeweils mindestens einen Maschinenzyklus lang anliegen, da der Mikrocontroller jeweils den Zustand der Anschlußpins nur einmal pro Maschinenzyklus prüft. Daraus ergibt sich, daß der gültige Pegel jeweils mindestens für 12 Oszillatorperioden stabil an den Pins anliegen muß.

Mit Registrierung einer gültigen Flanke bzw. eines Pegels werden durch die Hardware des Prozessors die korrespondierenden Bits IE0 bzw. IE1, die ebenfalls Bestandteil des TCON-Registers sind, gesetzt. Der Zustand dieser Bits, der sich von der Steuersoftware auch beliebig setzen oder löschen läßt, löst den eigentlichen Interrupt aus. War der Interrupt flankengetriggert, löscht die Hardware beim Ausführen der Interrupt-Routine automatisch das entsprechende IE-Bit.

War der Interrupt-Pegel getriggert, bleibt das IE-Bit gesetzt, solange der zugehörige INT-Pin auf Low-Pegel liegt. Das Löschen kann erst erfolgen, sobald der INT-Pin auf High-Pegel liegt.

Bei der Konzeption der Hard- und Software ist deshalb folgendes zu beachten:

1. Die Interrupt-Anforderung muß mindestens so lange anliegen, daß die Hardware eine einwandfreie Detektierung vornehmen kann.

2. Daß bis zum Ende der Interrupt-Service-Routine die Interrupt-Anforderung nicht mehr anliegt, da ansonsten der Interrupt mehrfach ausgelöst und durchlaufen würde.

In den meisten Fällen wird daher durch geeignete Hard- und Softwaremaßnahmen dafür gesorgt, daß während der Interrupt-Ausführung das Interrupt-Anforderungssignal zurückgenommen wird.

Mit dem Überlauf des Zählers 0 bzw. 1 wird automatisch das entsprechende Timer-Overflow-Flip-Flop TF0 bzw. TF1 gesetzt.

Die TF0- und TF1-Flags sind Bestandteil des bit- und byte-adressierbaren Registers TCON.

Bei entsprechender Interrupt-Freigabe wird damit automatisch der zugehörige Interrupt ausgelöst. Ein Löschen dieser Bits erfolgt entweder durch einen Befehl oder durch die Hardware, sobald in die entsprechende Interrupt-Routine verzweigt wird. Eine Ausnahme bildet lediglich der Zähler im Timer-Mode 3 (Kapitel 4.4 „ELVjournal" 3/95, Seite 21).

Die Auslösung des seriellen Interrupts erfolgt durch das Setzen des RI- oder TI-Bits. Die TI- und RI-Flags sind Bestandteil des bit- und byte-adressierbaren Registers SCON, die im „ELVjournal" 5/95 auf Seite 89 beschrieben wurden. Das RI-Flag wird automatisch vom Controller gesetzt, sobald dieser ein Zeichen über die seriellen Schnittstellen empfangen hat. In der Interrupt-Service-Routine läßt sich dann das empfangene Zeichen auswerten und weiter verarbeiten.

Nachdem ein Zeichen über die serielle Schnittstelle ausgegeben wurde, setzt die Logik das TI-Bit. Mit Auslösung der seriellen Interrupts werden die RI- und TI-Flags von der Hardware nicht gelöscht, um die spätere Ermittlung der Interrupt-Ursache zu ermöglichen. Am Ende der Interrupt-Service-Routine sind die entsprechenden Flags softwareseitig zu löschen.

Der 8032/52 hat noch eine weitere Interrupt-Quelle, die durch das Bit TF2 oder EXF2 ausgelöst wird. Der Timer/Zähler2-Überlauf setzt das TF2-Bit, während durch eine negative Flanke an dem T2EX-Anschlußpin des Mikroprozessors das EXF2-Flip-Flop gesetzt wird, freigegeben durch EXEN2. Die 3 direkt- und byte-adressierbaren Bits sind Bestandteil des T2CON-Registers (Beschreibung erfolgte im „ELVjournal" 4/95 auf Seite 60).

Die TF2- und EXF2-Bits werden beim Eintritt in die Interrupt-Service-Routine nicht gelöscht, so daß hier die Überprüfung der Interrupt-Ursache erfolgen kann. Auch hier muß wie bei der seriellen Schnittstelle die Interrupt-Service-Routine das gesetzte Flag zurücksetzen.

Die einen Interrupt erzeugenden Bits IE0, IE1, TF0, TF1, RI, TI, TF2 und EXF2 können durch die Software gesetzt oder gelöscht werden, als wäre dies durch die Hardware erfolgt. Somit können Interrupts, beispielsweise für die Testphase, durch die Software gesetzt oder schwebende Interrupt-Anforderungen gestrichen werden.

Jede der beschriebenen Interrupt-Quellen läßt sich individuell freigeben oder sperren, durch die Steuerbits EX0, ET0, EX1, ET1, ES und ET2, die im Interrupt-Enable-Register IE untergebracht sind (Tabelle 21). In Bit 7 des IE-Registers ist das Steuerbit EA (Enable All) untergebracht, welches die generelle Interrupt-Freigabe steuert. Zur Auslösung eines Interrupts muß also eine der Interrupt-Quellen durch das Setzen der zugehörigen Steuerbits und die allgemeine Interrupt-Freigabe durch das Setzen des EA-Bits erfolgen.

Bei der Ausführung von kritischen Programmoperationen, die nicht unterbrochen werden dürfen, ist dann lediglich das EA-Bit auf 0 zu setzen, um die generelle Interrupt-Auslösung zu verhindern. Nach Abarbeitung der kritischen Routine ist das EA-Bit wieder zu setzen, um den reibungslosen Programmablauf zu gewährleisten.

6.2. Interrupt-Prioritäten

Für jede Interrupt-Quelle läßt sich, wie Abbildung 126 zeigt, die Interrupt-Priorität auf eine hohe oder niedrige Prioritätsebene einstellen. Die Steuerung übernimmt das IP-Register, welches die Prioritäts-Steuerbits für jede einzelne Interrupt-Quelle enthält (Tabelle 22). Bei gelöschtem Bit wird der zugehörige Interrupt jeweils mit niedrigerer Priorität behandelt, während ein gesetztes Bit den Prozessor veranlaßt, diesen Interrupt mit hoher Priorität zu behandeln.

Das Prioritäts-Problem tritt nur auf, wenn mehr als eine Interrupt-Quelle gleichzeitig eine Programm-Unterbrechung anfordert. Führt der Prozessor gerade eine Interrupt-Service-Routine aus, die mit niedrigerer Priorität versehen ist, so läßt sich diese Programmabarbeitung noch durch einen Interrupt mit höherer Priorität unterbrechen. Umgekehrt ist es demnach nicht möglich, daß ein Interrupt auf hoher Prioritätsebene durch einen niedrigeren oder gleichgesetzten Interrupt unterbrochen wird.

Eine gegenseitige Unterbrechung der Interrupts auf gleicher Prioritätsebene ist ebenfalls nicht möglich. Soll nun generell verhindert werden, daß mehr als ein Interrupt gleichzeitig bearbeitet wird, so sind die Prioritäten der Interrupt-Quellen alle auf gleiche Ebene zu setzen.

Ist beispielsweise in einem Programm ein regelmäßiger Timer-Interrupt für die Zeit-Zählung und ein Interrupt für die serielle Schnittstelle vorgesehen, so ist es sinnvoll, den Timer-Interrupt mit niedrigerer Priorität und den Schnittstellen-Interrupt mit hoher Priorität zu versehen. Sollte dann einmal während der Abarbeitung des Timer-Interrupts ein Interrupt der seriellen Schnittstelle auftreten, so wird zunächst dieser abgearbeitet, woraufhin anschließend die Abarbeitung des Timer-Interrupts fortgesetzt wird.

Tabelle 21: Interrupt-Enable-Register IE

Bedeutung	EA	–	(ET2)	ES	ET1	EX1	ET0	EX0
Bitadresse	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
Byteadresse	IP 0A8H							

Tabelle 22: Interrupt-Prioritäts-Register IP

Bedeutung	–	–	(PT2)	PS	PT1	PX1	PT0	PX0
Bitadresse	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H
Byteadresse	IP 0B8H							

Tabelle 23: Interrupt-Einsprungadressen

Quelle	Bit	Interrupt-Adresse	Priorität
Externer Interrupt 0	IE0	0003H	1
Zähler0-Überlauf	TF0	000BH	2
Externer Interrupt 1	IE1	0013H	3
Zähler1-Überlauf	TF1	001BH	4
Serielle Schnittstelle	RI oder TI	0023H	5
Zähler 2 oder externer Interrupt 2 (8032/52)	TF2 oder EXF2	002BH	6

Die Reaktion auf die serielle Schnittstelle ist in der Regel wichtiger, um einen Datenverlust zu vermeiden, während es im allgemeinen von untergeordneter Bedeutung ist, wenn die Zeitaktualisierung bzw. -übernahme einige Mikrosekunden später erfolgt. Eine Beeinflussung der absoluten Zeitzählung erfolgt ohnehin nicht, da die Timer meistens im Auto-Reload-Mode arbeiten.

Beim Auftreten mehrerer Interrupts gleicher Priorität werden diese vom Prozessor in einer festgelegten Reihenfolge abgearbeitet (Tabelle 23). Bearbeitet die CPU beispielsweise gerade einen Timer-Interrupt und treten währenddessen der externe Interrupt 1 und etwas später der externe Interrupt 0 auf, so erscheinen für den Prozessor nach der Timer-Interrupt-Routine die beiden externen Interrupts als gleichzeitig.

In jedem Maschinenzyklus läuft daher eine Abfragesequenz für anstehende Interrupts, d. h., es würde zunächst der Interrupt 0 (höhere Priorität) und anschließend die Interrupt1-Service-Routine abgearbeitet. Eine Änderung dieser Abarbeitungsreihenfolge läßt sich nur dadurch vornehmen, daß der bevorzugte Interrupt die höhere Priorität, während alle anderen Interrupts die niedrigere Priorität bekommen (IP-Register).

Zusammenfassend ist zu sagen, daß bei gleichzeitig anstehenden Interrupts zunächst die Interrupts mit hoher Priorität in der beschriebenen Prioritätsreihenfolge abgearbeitet werden und anschließend die Interrupts der niedrigeren Prioritätsebene.

6.3 Interrupt-Verarbeitung

Nach der Auswertung der Interrupt-Anforderung springt der Prozessor mit einem Long-Call zu den der Interrupt-Quelle gehörenden Einsprungadressen, die Tabelle 23 zeigt. Bei dem Einsprung werden gleichzeitig die folgenden Adressen des gerade bearbeiteten Befehls auf dem Stack abgelegt (CALL), so daß am Ende der Interrupt-Routine genau an der unterbrochenen Stelle wieder aufgesetzt wird.

Die zeitlichen Zusammenhänge der Interrupt-Reaktion zeigt Abbildung 127. Die Speicherung der Interrupt-Anforderung erfolgt zum Ende eines Maschinenzyklus, während die eigentliche Interrupt-Abfrage zu Beginn des darauf folgenden Maschinenzyklus erfolgt. Nach Erkennung einer solchen Interrupt-Anforderung springt der Prozessor in den folgenden Maschinenzyklen zu der festgelegten Interrupt-Adresse und beginnt dort mit der Abarbeitung der Interrupt-Service-Routine.

In Abbildung 127 ist die kürzestmögliche Antwortzeit auf eine Interrupt-Anforderung dargestellt. Wird nun gerade während der Interrupt-Anforderung ein Multiplikations- oder Divisionsbefehl, der 4 Zyklen benötigt, abgearbeitet, so können bis zu 8 Maschinentakte bis zur Ausführung der Interrupt-Routine vergehen. Die Zeit für die Beantwortung einer Interrupt-Anforderung beträgt also immer mindestens 3 bis maximal 8 Maschinenzyklen. Längere Wartezeiten können sich ergeben, wenn ein Interrupt gleicher oder höherer Priorität gerade in Bearbeitung ist.

Bild 128: Aufbau des Programmspeichers anhand eines Beispiels

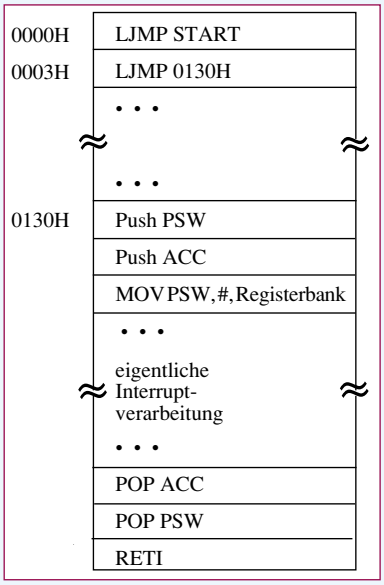
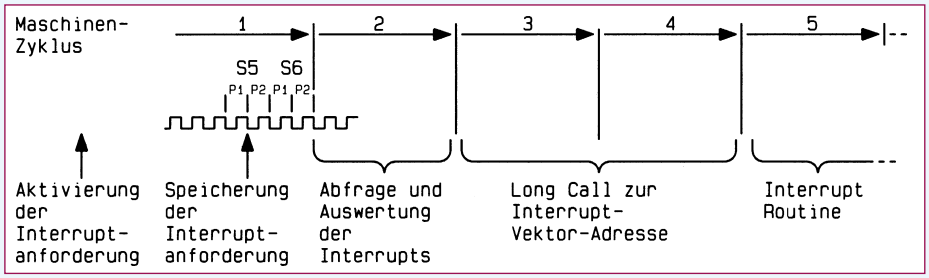


Bild 127: Zeitliche Reaktion auf eine Interrupt-Anforderung



Wie aus Tabelle 23 ersichtlich, stehen zwischen den Interrupt-Einsprungadressen maximal 8 Byte zur Verfügung. In den meisten Fällen wird daher an den Interrupt-Einsprungadressen ein Long-JMP-Befehl stehen, der unmittelbar zur eigentlichen Interrupt-Service-Routine verzweigt.

Abbildung 128 zeigt den grundsätzlichen Aufbau des Programmspeichers für die Abarbeitung einer INT0-Unterbrechungsanforderung. Nach dem Ansprung der für die Interrupt-Bearbeitung des externen Interrupts INT0 zuständigen Adresse (0003H) erfolgt in dem Beispiel ein weiterer Sprung zur Adresse 0130H, ab der die eigentliche Interrupt-Service-Routine beginnt.

Da die Interrupt-Service-Routine das laufende Hauptprogramm nicht beeinflussen darf, müssen zunächst die in der Interrupt-Service-Routine verwendeten Register und Flags gesichert werden. Dieses ist am einfachsten möglich durch die Ablage der Informationen auf dem Stack.

Üblicherweise benötigt die Interrupt-Service-Routine nicht nur das Programm-Statuswort und den Akkumulator, sondern auch einen Register-Satz. Sinnvollerweise wird dann für diese Interrupt-Service-Routine auf einen anderen Register-Satz umgeschaltet, der für die Interrupt-Service-Routine reserviert ist. Danach erfolgt die eigentliche Interrupt-Verarbeitung.

Zum Ende der Interrupt-Service-Routine müssen durch die entsprechenden Befehle die Inhalte des Programm-Statuswortes und des Akkumulators auf den Originalstand zurückgebracht werden. Die Selektierung der zuvor benutzten Registerbank erfolgt automatisch mit dem „POP PSW"-Befehl.

Zum Abschluß der Interrupt-Verarbeitungs-Routine steht der Befehl „RETI", der gleichzeitig 2 Aufgaben erfüllt. Der Prozessor setzt damit einerseits das zuständige IIP-Flip-Flop zurück, um weitere Interrupts abarbeiten zu können. Ferner holt der Befehl den auf dem Stack abgelegten Inhalt des Programmzählers zurück, um mit der Befehlsausführung im Hauptprogramm an der Stelle fortzufahren, an der der Interrupt auftrat.

Der RET-Befehl würde ebenfalls einen Rücksprung aus der Interrupt-Service-Routine bewirken, könnte aber dem Prozessor nicht die Mitteilung über die Freigabe neuer Interrupts machen, womit im allgemeinen die weitere Funktion stark beeinträchtigt wäre.

Im nächsten Teil der Mikrocontroller-Grundlagen-Serie zeigen wir anhand praktischer Beispiele die Verwendung und den Einsatz von Interrupt-Routinen. ELV