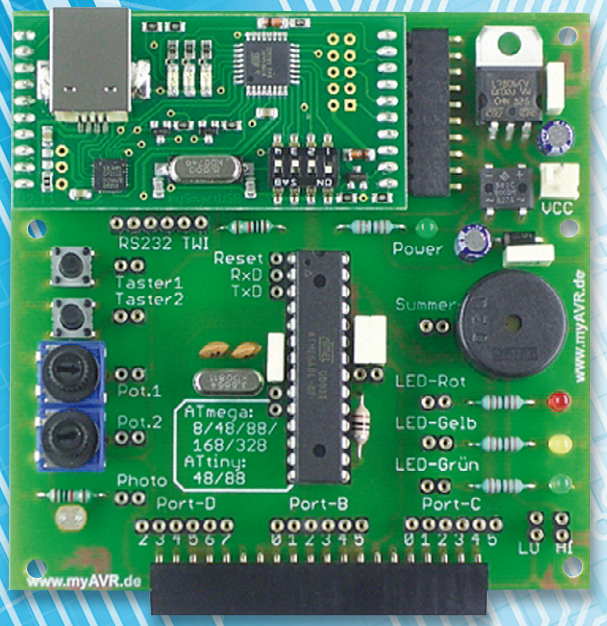
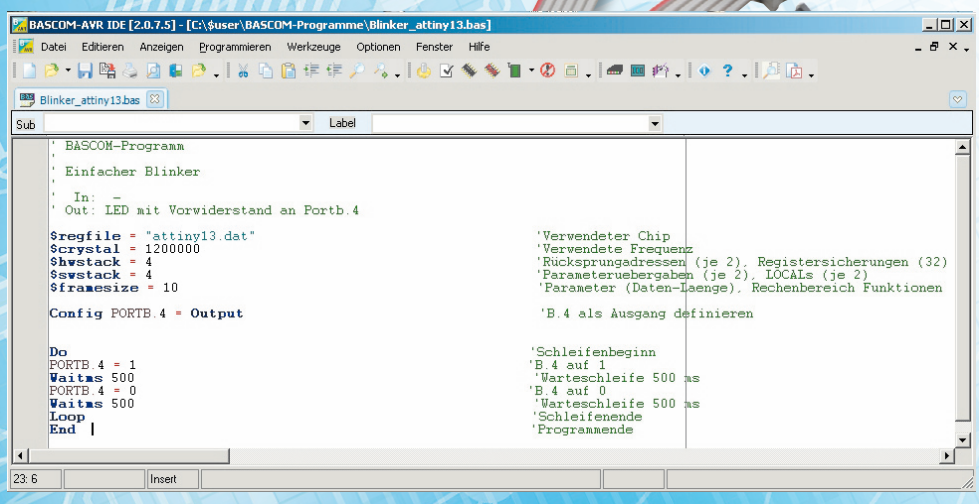
Mikrocontroller-Einstieg

Teil 4: Interrupts





mit BASCOM-AVR

Normalerweise werden die Befehle eines BASCOM-Programms einer nach dem anderen abgearbeitet. Bei bedingten Verzweigungen und Schleifen ist ebenfalls die Reihenfolge der Befehlsabarbeitung bereits bei der Programmierung festgelegt worden. Es gibt aber Situationen, in denen auf externe oder interne Ereignisse sofort reagiert werden soll. Diese ereignisgetriggerte Abweichung vom sequentiellen Programmablauf geschieht über sogenannte Interrupts (Unterbrechungen), deren Konzept und Umsetzung in BASCOM in dieser Folge unserer Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR" beschrieben werden.

Interrupts

Ein Interrupt ist eine Unterbrechung des „normalen" Ablaufs. Man kann sich als Vergleich jemanden vorstellen, der auf seinem Sofa sitzt und ein gutes Buch liest. Plötzlich klingelt es an der Haustür. Das Lesen des Buches wird unterbrochen, der Leser sieht an der Haustür nach, wer dort ist, und geht danach wieder zu dem Buch zurück, um das Lesen an genau der Stelle fortzusetzen, an der vorher die Unterbrechung kam. Eine andere Art der Unterbrechung könnte ein Telefonklingeln sein, für das das Lesen des Buches kurz unterbrochen und später wieder fortgesetzt wird. Um den Vergleich abzurunden, könnte es auch passieren, dass dem Leser plötzlich etwas einfällt – zum Beispiel, dass er kurz etwas trinken möchte –, er das kurz erledigt und dann zu seinem spannenden Buch zurückkehrt. Ebenso ist es bei einem BASCOM-Programm: Das Programm wird Schritt für Schritt abgearbeitet, und wenn plötzlich und zu einem nicht vorher festgelegten Zeitpunkt ein Interrupt – also ein bestimmtes Ereignis – eintritt, wird das eigentliche Programm verlassen, die dem Interrupt entsprechenden Aktionen ausgeführt und danach genau an die Stelle im Programm zurückgesprungen, an der es beim Auftreten des Ereignisses verlassen wurde. Bild 1 zeigt, wie beim Eintreten des Interrupt-Ereignisses die dem Interrupt-Ereignis zugeordneten Aktionen (von jeder Stelle des Programms) angesprungen werden können und nach der Interrupt-Routine – genannt Interrupt Service Routine (ISR) – an die Stelle der Unterbrechung zurückgekehrt wird. Wie bei dem Vergleich mit dem Buchleser so gibt es auch bei einem AVR-Mikrocontroller sowohl externe

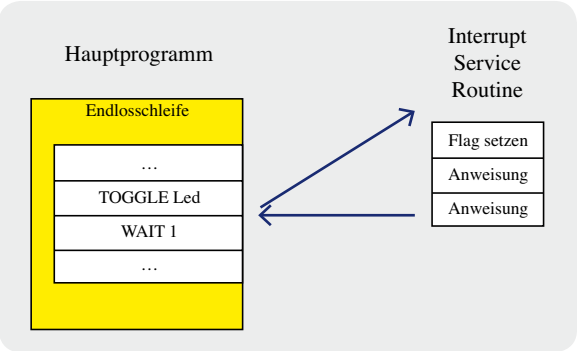


Bild 1: Programmablauf mit Interrupt

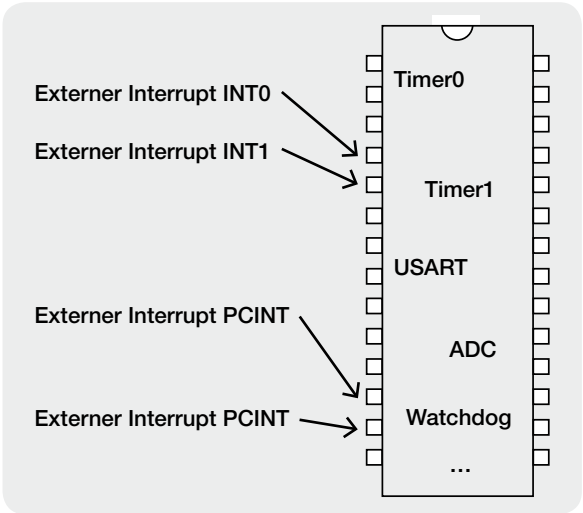


Bild 2: Externe und interne Interruptarten (schematisch)

Interrupt	Interrupt-Beschreibung
RESET	External pin, power-on reset, brown-out reset and watchdog system reset
INT0	External interrupt request 0
INT1	External interrupt request 1
PCINT0	Pin change interrupt request 0
PCINT1	Pin change interrupt request 1
PCINT2	Pin change interrupt request 2
WDT	Watchdog time-out interrupt
TIMER2 COMPA	Timer/Counter2 compare match A
TIMER2 COMPB	Timer/Counter2 compare match B
TIMER2 OVF	Timer/Counter2 overflow
TIMER1 CAPT	Timer/Counter1 capture event
TIMER1 COMPA	Timer/Counter1 compare match A
TIMER1 COMPB	Timer/Counter1 compare match B
TIMER1 OVF	Timer/Counter1 overflow
TIMER0 COMPA	Timer/Counter0 compare match A
TIMER0 COMPB	Timer/Counter0 compare match B
TIMER0 OVF	Timer/Counter0 overflow
SPI, STC	SPI serial transfer complete
USART, RX	USART Rx complete
USART, UDRE	USART, data register empty
USART, TX	USART, Tx complete
ADC	ADC conversion complete
EE READY	EEPROM ready
ANALOG COMP	Analog comparator
TWI	2-wire serial interface
SPM READY	Store program memory ready

Bild 3: Interruptarten ATmega88

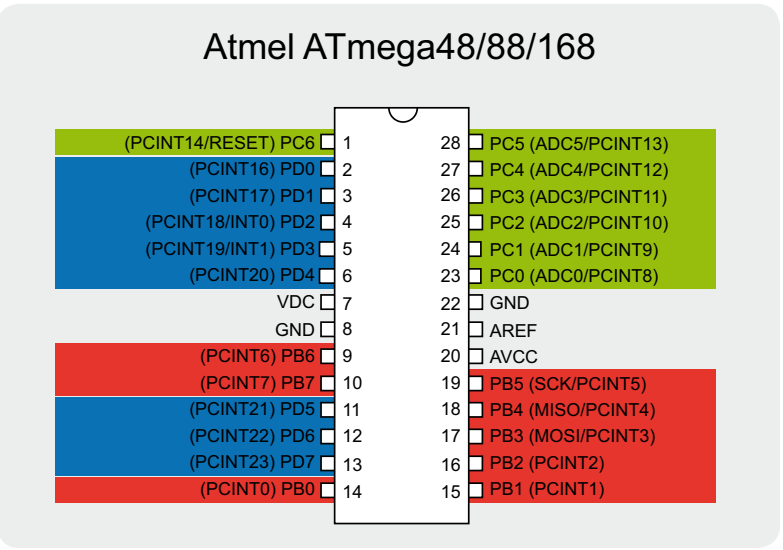


Bild 4: Pinbelegung ATmega88

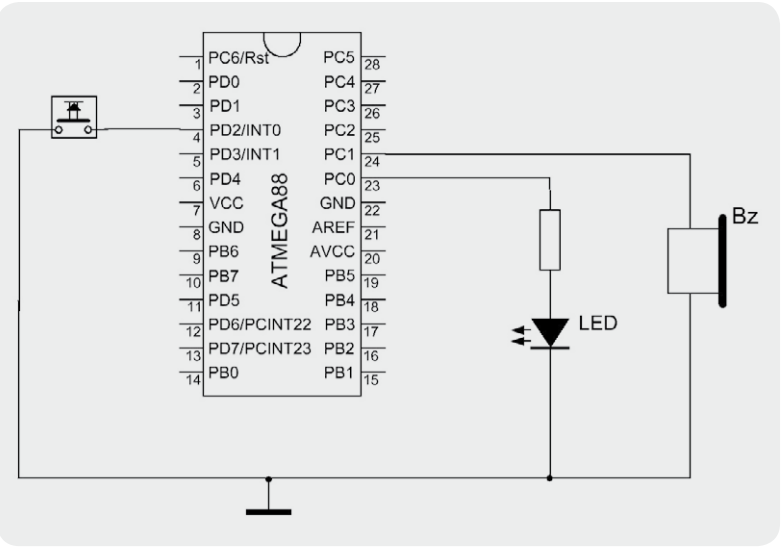


Bild 5: Schaltplan für INT0

Elektronikwissen A

Logik-Level, Flanken und Pin-Change

Man spricht (bei der hier betrachteten positiven Logik) von einem Logik-Pegel (oder Logik-Level) 1 oder High-Pegel (kurz High), wenn an einem Mikrocontroller-Pin in etwa die Versorgungsspannung anliegt bzw. ausgegeben wird. Gängige Versorgungsspannungen sind 5 Volt oder 3,3 Volt. Von einem Logik-Pegel (Logik-Level) 0 oder Low-Pegel (kurz Low) wird bei einer Spannung nahe Null Volt gesprochen. Die genauen Spannungsbereiche sind im jeweiligen Datenblatt nachzulesen.

Eine (ab)fallende Flanke (englisch: falling edge) ist ein Signalwechsel von High zu Low. Eine (an)steigende Flanke (englisch: rising edge) ist der Name für einen Low-High-Wechsel.

Als "Pin-Change" beteichnet man einen beliebigen Signalwechsel – also eine fallende oder steigende Flanke.

Ein INTx-Interrupt kann für Low-Level, steigende Flanke, fallende Flanke oder beliebige Flanke konfiguriert werden.

Ein PCINTxx-Interrupt kann nur nur auf Pin Change – also beliebige Flanke – reagieren.

Elektronikwissen B

Polling versus Interrupt

Man kann zwei Arten unterscheiden, programmtechnisch festzustellen, ob ein bestimmtes Ereignis stattgefunden hat: Polling und Interrupt. Beim Polling fragt man per Programm immer wieder – sehr schnell – ab, ob das Ereignis eingetreten ist. So kann man in einer Schleife immer wieder abfragen, ob ein Taster gedrückt wurde und dadurch an einem Pin des Mikrocontrollers 0V anliegt. Jedes Abfragen kostet etwas Zeit, und abgefragt wird nur zu bestimmten Zeiten. Dadurch ist es möglich, dass ein sehr kurzes Signal nicht erkannt wird, weil das Programm gerade etwas anderes ausführt. Für das vergleichsweise sehr langsame Signal einer gedrückten Taste kann das funktionieren, aber sehr kurze Signale von Sensoren können verlorengehen.

Bei der Interrupt-Technik wird nicht in einer Schleife im Programm immer wieder das Ereignis abgefragt, sondern es gibt eine Einheit im Mikrocontroller, die das Eintreten des Ereignisses unabhängig vom Programmablauf erkennt und bei dessen Auftreten eine Programmunterbrechung (Interrupt) auslöst. Es wird kein Programmcode für das Abfragen des Ereignisses benötigt, weil die entsprechende Einheit selbstständig im Hintergrund arbeitet, und es geht kein Ereignis verloren, wie es beim Polling passieren könnte.

Ereignisse als auch interne Ereignisse, die zu einer Unterbrechung des Programms führen können. Externe Ereignisse können Signale an bestimmten Pins des Mikrocontrollers sein, während interne Ereignisse durch Einheiten im Innern des Mikrocontrollers ausgelöst werden wie zum Beispiel bei einem Timer-Überlauf, beim Empfang eines Zeichens über die serielle Schnittstelle, bei der Fertigstellung einer Analog-zu-Digital-Umwandlung usw. (Bild 2). Eine komplette Liste der möglichen Interrupts eines ATmega88 zeigt Bild 3.

Externe Interrupts

Externe Interrupts reagieren auf von außen an den Mikrocontroller angeschlossene Signale.

Es gibt zwei unterschiedliche Arten von externen Interrupts. Die externen Interrupts INT0 und INT1 sind fest den Pins 4 und 5 des Mikrocontrollers zugeordnet (Bild 4). Man kann sie für steigende oder fallende Flanke oder Level-Interrupt definieren (siehe „Elektronikwissen A"). Zusätzlich gibt es bei einem ATmega88 die externen Pin-Change-Interrupts PCINT0 bis PCINT23, die bei jedem Level-Wechsel einen Interrupt auslösen können und deren Zuordnung zu den Pins in Bild 4 zu sehen ist. Für andere AVR-Mikrocontroller ist das jeweilige Datenblatt zu benutzen. Die Interruptarten INT0/INT1 und PCINTxx werden in BASCOM etwas unterschiedlich benutzt.

INT0/INT1

Bild 5 zeigt den Schaltplan für ein Testprogramm für INT0. Aus Gründen der Übersichtlichkeit wurden die Anschlüsse für die Spannungsversorgung und Abblock-Kondensatoren in Bild 5 nicht dargestellt.

Die externen Interrupts INT0 und INT1 funktionieren nur dann wenn

1. der jeweilige Interrupt definiert wurde (CONFIG INTx...),
2. die zugehörige Interrupt Service Routine definiert wurde (ON INTx ...),
3. der Interrupt aktiviert wurde (ENABLE INTx ..) und
4. Interrupts generell aktiviert wurden (ENABLE INTERRUPTS).

Das folgende Programm lässt eine Leuchtdiode langsam blinken.

Wenn während der Wartezeit (WAIT 3) unabhängig vom Programmablauf ein (kurzes) Signal eintritt, würde es normalerweise verlorengehen. Da aber die Interrupt-Einheit den INT0-Pin ständig überwacht und bei Auftreten des Ereignisses der Interrupt ausgelöst und die Interrupt Service Routine angesprungen und ausgeführt wird, geht selbst ein sehr kurzes Signal während der Wartezeit nicht verloren. Hier und in weiteren Beispielen werden Taster nur zu Zwecken der Veranschaulichung verwendet. Tasteneingaben könnte man in BASCOM auch mit dem DEBOUNCE-Befehl oder sogar mit Polling (siehe „Elektronikwissen B") abfragen, weil es sich um sehr langsame Signale handelt. Typische Geräte, die extern einen Interrupt für den Mikrocontroller erzeugen, sind beispielsweise Echtzeit-Uhren (Real-Time-Clocks), Porterweiterungsbausteine, Temperatursensoren mit Alarm-Temperaturen, Bewegungssensoren, serielle Eingabegeräte usw.

```
' BASCOM-Programm
'
' INT0 Interrupt
' LED blinkt langsam
' Bei Tastendruck Interrupt und Ton
'
' In:  Taster an INT0 (D.2)
' Out: LED mit Vorwiderstand an C.0
'      Buzzer ohne Elektronik an C.1 (oder LED mit Widerstand)

$regfile  = "M88def.dat"                'Verwendeter Chip
$crystal = 1000000                      'Verwendete Frequenz
$hwstack = 40                           'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                           'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                         'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Pind.2 = Input                   'Taster an INT0 Interrupt-Eingang (D.2)
Eingang Alias Pind.2
Portd.2 = 1                             'Interner Pullup-Widerstand

Config Portc.0 = Output                 'Ausgang fuer blinkende LED
Led Alias Portc.0

Config Portc.1 = Output                 'Ausgang fuer Buzzer
Buzzer Alias Portc.1

Config  Int0 = Falling                  'INT0 fuer fallende Flanke
On Int0 Int0_isr                        'Bei INT0-Interrupt zu INT0_isr springen

Enable Int0                             'INT0 ermoeglichen
Enable Interrupts                       'Alle Interrupts ermoeglichen



Do
  Toggle Led                            'LED langsam blinken lassen
  Wait 3
Loop
End



Int0_isr:                               'Name der Interruptroutine wie auch im ON-Befehl beschrieben
  Sound Buzzer , 55 , 189               'Kurzer 440Hz-Ton
Return                                  'Jede Interrupt-Routine muss mit Return beendet werden
```

Erläuterungen:

Der INT0-Eingang (D2) wird mit internem Pull-up-Widerstand als Eingang benutzt. Mit CONFIG INT0 = Falling wird der INT0-Interrupt so eingestellt, dass er (nur) auf fallende Flanken (also High-Low-Wechsel) reagiert. Durch ON wird angegeben, zu welchem Label bei Auftreten des Interrupts gesprungen werden soll. ENABLE INT0 schaltet den INT0-Interrupt ein und ENABLE INTERRUPTS schaltet generell alle Interrupts ein. Die Endlosschleife (Do-Loop) lässt eine Leuchtdiode langsam blinken, indem der LED-Ausgabepin zwischen High und Low wech-

selt (toggelt) und dann 3 Sekunden gewartet wird. Mit diesem kleinen Demo-Programm kann man sehr schön testweise zu jeder beliebigen Zeit während der 3 Sekunden Wartezeit den INT0-Interrupt auslösen, wodurch zu der INT0-ISR gesprungen wird, was man durch einen kurzen Piepton aus dem Piezo-Buzzer nachvollziehen kann. Natürlich kann man während einer Warteperiode auch mehrfach kurz den Interrupt auslösen und dadurch mehrfach den Buzzer ertönen lassen. Der SOUND-Befehl von BASCOM dient zum Erzeugen einfacher Töne.

Während der Abarbeitung einer Interrupt Service Routine sind übrigens alle Interrupts abgeschaltet – werden aber zwischengespeichert, falls sie auftreten, und nach dem Ende der Interrupt Service Routine aktiv.

Mit Flag

Es gibt Situationen, in denen die Aktion nicht sofort ausgeführt, sondern der Interrupt zunächst nur vorgemerkt werden soll. In dem Fall kann man bei Eintreten des Interrupt-Ereignisses ein Flag (eine Variable) setzen und an geeigneter Stelle im Programm dieses Flag abfragen, um dann an dieser Stelle die Aktion ausführen zu lassen.

```
' BASCOM-Programm
'
' INT0 Interrupt
' LED blinkt langsam
' Bei Tastendruck Flag setzen für Ton
'
' In:  Taster an INT0 (D.2)
' Out: LED mit Vorwiderstand an C.0
'      Buzzer ohne Elektronik an C.1

$regfile  = "M88def.dat"                  'Verwendeter Chip
$crystal = 1000000                        'Verwendete Frequenz
$hwstack = 40                             'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                             'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                           'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Pind.2 = Input                     'Taster an INT0 Interrupt-Eingang (D.2)
Eingang Alias Pind.2
Portd.2 = 1                               'Interner Pullup-Widerstand

Config Portc.0 = Output                   'Ausgang fuer blinkende LED
Led Alias Portc.0

Config Portc.1 = Output                   'Ausgang fuer Buzzer
Buzzer Alias Portc.1

Config  Int0 = Falling                    'INT0 fuer fallende Flanke
On Int0 Int0_isr                          'Bei INT0-Interrupt zu INT0_isr springen

Enable Int0                               'INT0 ermoeglichen
Enable Interrupts                         'Alle Interrupts ermoeglichen
Dim Interrupt_flag As Bit                 'Anzeiger fuer INT0

Do
  Toggle Led
  Wait 3
  If Interrupt_flag = 1 Then
     Sound Buzzer , 55 , 189              '440Hz
     Reset Interrupt_flag                 'Flag wieder auf 0 setzen
  End If
Loop
End


Int0_isr:                                 'Name der Interrupt-Routine
Set Interrupt_flag                        'Flag auf 1 setzen
Return
```

Erläuterungen:
Konfiguration und Einschalten des INT0 erfolgen hier wieder mit CONFIG INT0 bzw. ENABLE INT0. Alle Interrupts werden mit ENABLE INTERRUPTS eingeschaltet. Sobald das Ereignis (High-Low-Wechsel an INT0) auftritt, wird in der INT0-ISR die Bit-Variable Interrupt_Flag gesetzt. In der Hauptschleife wird in diesem Beispielprogramm erst die Wartezeit (WAIT 3) komplett abgearbeitet und danach geprüft, ob während der Wartezeit der Interrupt ausgelöst worden war. Wenn das der Fall ist, wird an dieser Stelle im Programm die Aktion ausgeführt (SOUND Buzzer).

PCINT

Neben den INTx-Interrupts stellen die meisten AVR-Mikrocontroller eine große Anzahl sogenannter Pin-Change-Interrupts (PCINTxx) zur Verfügung.

Das Arbeiten mit externen PCINT-Interrupts unterscheidet sich etwas von der Vorgehensweise bei INT0/INT1-Interrupts. Eine Konfiguration entfällt bei PCINT, da sie auf jeden „Logic Level Change" reagieren – also auf jeden Übergang von High zu Low und umgekehrt. Deshalb heißt es auch PCINT für „Pin Change Interrupt". Analog wie bei INT0/INT1 wird mit ON PCINTx definiert, zu welcher Routine im Interruptfall gesprungen werden soll. Die PCINTxx sind allerdings zu drei Gruppen zusammengefasst: PCINT0, PCINT1 und PCINT2. Zu jeder dieser PCINTx-Gruppen gehören einzelne PCINT-Interrupts.
PCINT0: PCINT0 bis PCINT7
PCINT1: PCINT8 bis PCINT14
PCINT2: PCINT16 bis PCINT23

Wenn EINER der PCINT-Interrupts einer Gruppe ausgelöst wird, löst der „Gruppen-Interrupt" aus. ON PCINT2 definiert also die Sprungadresse für jeden der zugeordneten Interrupts PCINT16 bis PCINT23.

Um einzelne Interrupts der Gruppe einzuschalten, wird das entsprechende Bit im Register PCMSKx auf 1 gesetzt (PCMSK-Register = Pin Change Mask Register). Für PCINT22 wird das Bit 6 im PCMSK2-Register auf 1 gesetzt (Bild 6).

In der Interrupt Service Routine wird abgefragt, WELCHER Pin der Gruppe den Interrupt ausgelöst hat.

Bit	7	6	5	4	3	2	1	0	
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2

Bild 6: PCMSK2-Register

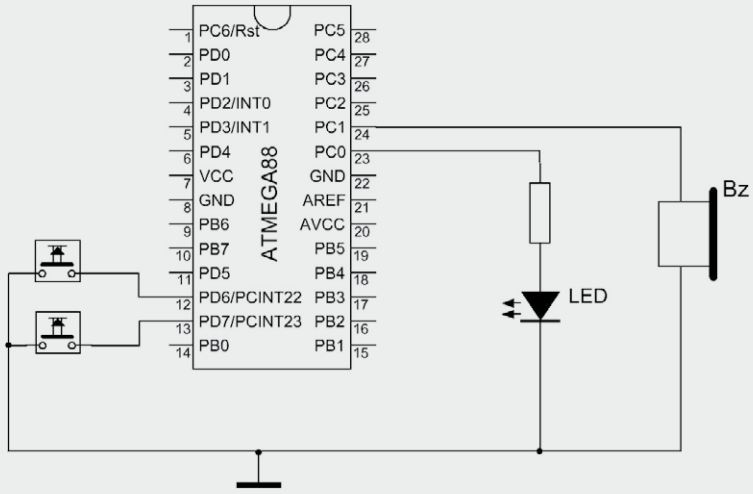


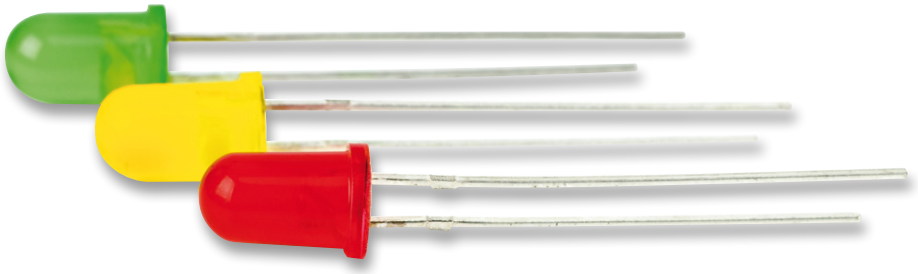
Bild 7: Schaltplan für PCINT

Elektronikwissen C

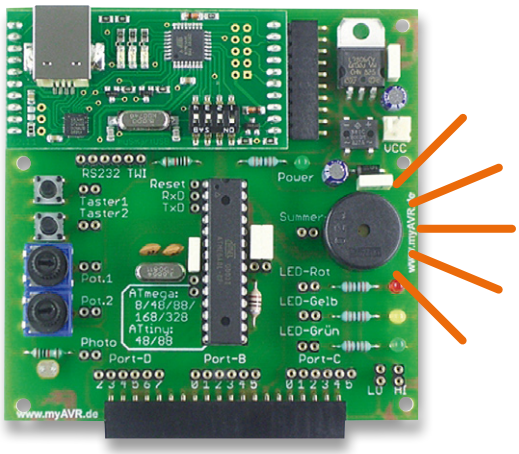
Informationsquellen

Bei der Arbeit mit Mikrocontrollern und BASCOM muss sehr präzise gearbeitet werden. Man muss daher immer richtige und aktuelle Informationen vorliegen haben und verwenden. Die wichtigsten Informationsquellen sind:

1. Das **Datenblatt** des verwendeten Mikrocontrollers von der ATMEL-Seite. Dort liest man auf der ersten Seite die genaue Spezifikation der wichtigsten Werte wie Speichergrößen, erlaubte Betriebsspannungen usw. Auf der zweiten Seite findet man die Pinbelegung des Mikrocontrollers. Weiter hinten die detaillierten Beschreibungen der Komponenten des Mikrocontrollers und auch der Register wie die der PCINT-Register.
2. In der **BASCOM-Hilfe** liest man sowohl grundsätzliche Themen als auch die genaue Beschreibung der einzelnen BASCOM-Befehle. Die BASCOM-Hilfe erreicht man entweder durch Drücken von F1 aus der BASCOM-Umgebung heraus oder online auf der MCSELEC-Internetseite.
3. Weitere Informationsquellen stellen **Bücher bzw. Internet-Tutorials** dar, die von unabhängigen Autoren geschrieben wurden.
4. In **Internet-Foren** kann man als „stiller Mitleser" sehr viel lernen oder eigene Fragestellungen beantwortet bekommen.



```
' BASCOM-Programm
'
' PCINT Interrupt
' LED blinkt langsam
' Bei Tastendruck Interrupt und Ton
'
' In:  Taster an PCINT22 (D.6)
'      Taster an PCINT23 (D.7)
' Out: LED mit Vorwiderstand an C.0
'      Buzzer ohne Elektronik an C.1

$regfile  = "M88def.dat"                'Verwendeter Chip
$crystal = 1000000                      'Verwendete Frequenz
$hwstack = 40                           'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                           'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                         'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Pind.6 = Input                                   'Taster an PCINT22 Interrupt-Eingang (D.6)
Taster_pcint22 Alias Pind.6
Portd.6 = 1                             'Interner Pullup-Widerstand

Config Pind.7 = Input                                   'Taster an PCINT23 Interrupt-Eingang (D.7)
Taster_pcint23 Alias Pind.7
Portd.7 = 1                             'Interner Pullup-Widerstand

Config Portc.0 = Output                 'Ausgang fuer blinkende LED
Blink_led Alias Portc.0

Config Portc.1 = Output                 'Ausgang fuer Buzzer
Buzzer Alias Portc.1

On Pcint2 Pcint2_isr                    'Wenn ein Pin der Gruppe PCINT2 (16-23) Pegelwechsel hat

Enable Pcint2                           'PCINT 16 bis 23 einschalten
Pcmsk2.6 = 1                            'PCINT22 einschalten
Pcmsk2.7 = 1                            'PCINT23 einschalten
Enable Interrupts                       'Alle Interrupt einschalten


Do
  Toggle Blink_led
  Wait 3                                'Wartezeit
Loop
End


Pcint2_isr:                             'Interrupt-Routine für PCINT2-Gruppe
If Taster_pcint22 = 0 Then Sound Buzzer , 55 , 189      'Bei falling Edge: Ton
'If Taster_pcint22 = 1 Then Sound Buzzer , 110 , 189    'Bei rising Edge: anderer Ton

If Taster_pcint23 = 0 Then                              'Bei falling Edge: 2 mal Ton
  Sound Buzzer , 55 , 189
  Waitms 50
  Sound Buzzer , 55 , 189
End If
Return
```



Erläuterungen:

Mit Config Pind.6 = Input bzw. Config Pind.7 = Input werden die Pins gemäß Bild 7 als Eingänge definiert. Aliase und Pull-up-Widerstände werden ebenfalls definiert. ENABLE Pcint2 und ENABLE Interrupts schalten die PCINT-Gruppe sowie alle Interrupts ein. Durch Setzen einer 1 im Bit des der Gruppe entsprechenden PCMSK-Registers werden die zugeordneten PCINT-Interrupts eingeschaltet. Die anderen PCINT-Gruppen können analog verwendet werden. In der Interrupt Service Routine wird abgefragt, ob am entsprechenden Pin eine 0 anliegt.

Dadurch erreicht man eine Reaktion auf fallende Flanke, weil bei steigender Flanke zwar der PCINT-Interrupt ausgelöst wird, aber nach Einsprung in die ISR am Pin eine 1 liegen würde. Durch die Abfragen in der ISR kann man auf die unterschiedlichen PCINTxx unterschiedlich reagieren.

Interne Interrupts

Neben den beschriebenen externen Interrupts gibt es einige wichtige interne Interrupts. Die am meisten verwendeten internen Interrupts werden durch interne Einheiten des Mikrocontrollers wie Timer, Watchdog-Timer, USART-Einheiten, ADCs usw. ausgelöst (vgl. Bilder 2 und 3).

Timer-Interrupt

Ein Timer ist eine interne Einheit eines Mikrocontrollers, die selbstständig und unabhängig vom normalen Programm einen Zähler hochzählt. Bei einem 8-Bit-Timer (Timer 0 und Timer 2) zählt der Zähler von 0 bis 255 und bei einem 16-Bit-Timer (Timer 1) wird von 0 bis 65.535 gezählt. Wenn der jeweilige höchste Wert erreicht ist, dann fängt der Timer wieder bei 0 an und löst dabei einen Timer-Interrupt aus.

Im folgenden Programm ist ein Timer so eingestellt, dass unabhängig vom normalen Programmablauf ca. alle 4 Sekunden ein Timer-Interrupt ausgelöst wird. In der Hauptschleife blinkt eine LED langsam, und sobald der Timer-Interrupt auftritt, erfolgt in der Interrupt-Routine ein kurzer Piepton.

```
' BASCOM-Programm
'
' Timer-Interrupt
' LED blinkt langsam
' Bei Timer-Überlauf Ton
'
' In:  Taster an INT0 (D.2)
' Out: LED mit Vorwiderstand an C.0
'      Buzzer ohne Elektronik an C.1

$regfile  = "M88def.dat"                  'Verwendeter Chip
$crystal = 1000000                        'Verwendete Frequenz
$hwstack = 40                             'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                             'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                           'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Timer1 = Timer , Prescale = 64     'Timer für Überlauf alle ca. 4 Sekunden
On Timer1 Timer1_isr
Enable Timer1
Start Timer1
Enable Interrupts

Config Portc.0 = Output                   'Ausgang fuer blinkende LED
Led Alias Portc.0

Config Portc.1 = Output                   'Ausgang fuer Buzzer
Buzzer Alias Portc.1

Do
  Toggle Led
  Wait 3
Loop
End


Timer1_isr:
Sound Buzzer , 55 , 189                   '440Hz
Return
```

Erläuterungen:

Mit Config Timer1 = Timer, Prescale = 64 wird der Timer definiert. Bei einer Taktfrequenz von 1 MHz bewirkt ein Prescaler (Vorteiler) von 64, dass der Timer mit 1 MHz : 64 = 15.625 Hz getaktet wird. Da der Timer 1 ein 16-Bit-Timer ist, wird nach dem Timer-Überlauf nach 65.536 Takten ein Interrupt ausgelöst. 15.625 Hz : 65.536 = 0,24 Hz. Ein Timerüberlauf und dadurch der Timer-Interrupt erfolgt also ca. alle 4 Sekunden.

Im Beispielprogramm wird in der Hauptschleife eine LED alle 3 Sekunden umgeschaltet. Unabhängig davon läuft der Timer und löst ca. alle 4 Sekunden einen Interrupt aus, wodurch in der Interrupt Service Routine ein kurzer Piepton ausgegeben wird. Mehr über Timer erfahren Sie im nächsten ELVjournal.

Die Tabelle 1 fasst die Schritte der Interrupt-Programmierung für externe bzw. Timer-Interrupts zusammen. Andere interne Interrupts wie zum Beispiel Watchdog-Timer werden analog verwendet. Die vollständige Liste der für den jeweils verwendeten Mikrocontroller verfügbaren Interrupts sowie deren Namen findet man im BASCOM-Verzeichnis in der Definitionsdatei des Mikrocontrollers, die dem Compiler mit der $REGFILE-Direktive angegeben wurde – zum Beispiel in der Datei m88def.dat. Die detaillierte Verwendung der Interrupts wird im Datenblatt des Mikrocontrollers beschrieben.

Generell sollte eine Interrupt Service Routine immer sehr kurz und schnell abarbeitbar gehalten werden. Es sollten keine komplexen Berechnungen oder Programmschleifen und auch keine aufwändigen Anweisungen in der ISR stattfinden. Grundsätzlich sollten nur wenige kurze Aktionen stattfinden – eventuell nur das Setzen eines Merkers/Flags.

Tabelle 1

Schritte bei INTx/PCINT/Timer-Interrupt

INT0/INT1	PCINT	Timer
Config INTx ..	---	CONFIG Timer ..
ON INTx ..	ON PCINTx ..	ON Timerx ..
ENABLE INTx	ENABLE PCINTx	ENABLE Timerx
---	PCMSKx.y = 1	---
ENABLE Interrupts	ENABLE Interrupts	Enable Interrupts
…	…	…
…	…	….
Interrupt_Routine:	Interrupt_Routine:	Interrupt_Routine:
..	IF Pinx.y=0 …	…
…	…	..
Return	Return	Return

Ausblick

Im nächsten Artikel unserer BASCOM-Artikelserie werden Timer und Counter vorgestellt, welche eine zentrale Rolle bei der Mikrocontrollerprogrammierung darstellen. Mit ihnen kann man Signale/Töne erzeugen, Zeitabläufe messen und Signale zählen. ELV

Weitere Infos:

- Stefan Hoffmann: Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM. Systematische Einführung und Nachschlagewerk mit vielen Anregungen. ISBN 978-3-8391-8430-1
- www.bascom-buch.de
- www.mcselec.com
- www.atmel.com

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics	–	–
Atmel AVRISP mkII Programmer	JW-10 03 55	€ 39,95
oder myAVR Board MK2	JW-10 90 00	€ 49,–
Netzteil für myAVR Board MK2	JW-10 90 01	€ 6,95
ATtiny13	JW-10 03 39	€ 1,95
ATmega8	JW-05 29 71	€ 3,20
ATmega88	JW-10 07 62	€ 3,95
100-nF-Kondensator	JW-10 03 17	€ 0,08
Batteriehalter für 3x Mignon	JW-08 15 30	€ 0,75
Batterieclip für 9-V-Block-Batterie	JW-08 01 28	€ 0,30
BASCOM-Buch	JW-10 90 02	€ 54,–
Experimentier-Board 1202B	JW-07 72 89	€ 12,95
Schaltdraht-Sortiment	JW-05 47 68	€ 5,95
LED-Set	JW-10 63 56	€ 3,95
oder Leuchtdioden	JW-10 66 60	€ 1,95
und Widerstände	JW-10 66 57	€ 1,85
Piezo-Signalgeber	JW-00 73 87	€ 0,95
Mikroschalter und -taster	JW-10 66 67	€ 2,80
Mini-Erschütterungssensor MES 1	JW-07 38 22	€ 11,95
3-Achsen-Beschleunigungssensor	JW-09 15 21	€ 6,95
I2C-Real-Time-Clock	JW-10 34 13	€ 6,50
Mini-USB-Modul	JW-09 18 59	€ 5,95
Nützlich: Pin-Ausrichter	JW-00 84 63	€ 4,95

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Preisstellung April 2013 – aktuelle Preise im Web-Shop