Mikrocontroller-Einstieg

Teil 10: Serielle (UART-)Datenübertragung: Empfang mit Interrupt

mit BASCOM-AVR

Im Teil 9 der Artikelserie wurden die grundlegen Methoden des seriellen Datenempfangs mit BASCOM und die BASCOM-Befehle INPUT/INPUTBIN, WAITKEY und INKEY vorgestellt. In diesem Teil geht es nun einen Schritt weiter, damit das laufende BASCOM-Programm weniger mit dem Empfang belastet wird.

Vom PC empfangen mit URXC-Interrupt

Die eleganteste Methode des seriellen Datenempfangs erfolgt mit Hilfe des seriellen Interrupts. Dabei kann das Programm beliebige andere Aktionen ausführen, ohne dass es zwischendurch den seriellen Puffer abfragen oder gar anhalten muss. Sobald von der eigenständig arbeitenden UART-Einheit des Mikrocontrollers ein Zeichen empfangen wird, wird der entsprechende URXC-Interrupt ausgelöst und es kann auf den Empfang reagiert werden.

```
' BASCOM-Programm
'
' Serieller Empfang vom PC (über USB-UART-Umsetzer UM2102 o. Ä.)
' Mit URXC-Interrupt
' Am PC senden mit BASCOM-Terminal, HTerm oder anderem Programm
'
'  In: Serielles Signal an Pin  d.0  = Rxd
' Out: LEDs mit Vorwiderständen an B.1 (rot), B.2 (gelb)

$regfile = „M88def.dat“                 'Verwendeter Chip
$crystal = 3686400                      'Verwendete Frequenz. Fuse-Bits: Externer Quarz.
$hwstack = 40                           'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                           'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                         'Parameter (Daten-Laenge), Rechenbereich Funktionen
$baud = 9600                            '9600 Bit pro Sekunde, 8 Datenbit, Keine Parität, 1 Stoppbit
Led_rot Alias Portb.1
Config Led_rot = Output
Led_gelb Alias Portb.2
Config Led_gelb = Output

On Urxc Zeichenempfangen                'Sprung zu Zeichenempfangen wenn Empfangs-Interrupt
Enable Urxc                             'Empfangs-Interrupt einschalten
Enable Interrupts                       'Interrupts generell einschalten

Dim Empfang_komplett As Bit             'Anzeiger für ‚Empfang komplett‘
Dim Eingabe_zeichen As String * 1       'Eingelesenes Zeichen
Dim Eingabe_string As String * 5        'Eingabezeichenkette
```

```
Empfang_komplett = 0
Print "ELVjournal"
Print "Mikrocontroller-Einstieg ";       'Semikolon unterdrückt CRLF
Print "mit BASCOM-AVR" : Print
Print "Eingabe Schaltbefehl (rot bzw. gelb)"

Do
If Empfang_komplett = 1 Then
  Print "Empfangen: " ; Eingabe_string
  Select Case Eingabe_string
    Case "rot" : Toggle Led_rot
    Case „gelb“ : Toggle Led_gelb
    Case Else : Print "Eingabe unbekannt: " ; Eingabe_string
  End Select
  Empfang_komplett = 0
  Eingabe_string = ""
End If
Loop
End

Zeichenempfangen:                                   'Interruptroutine
Eingabe_zeichen = Inkey()                           'Zeichen aus seriellem Puffer lesen

'--- Einlesen bis Anzahl Zeichen erreicht ist: ---
Eingabe_string = Eingabe_string + Chr(eingabe_zeichen)    'An Eingabestring anfügen
If Len(eingabe_string) = 3 Then Empfang_komplett = 1      'Wenn Länge erreicht, dann fertig

'(
'--- Alternativ: Bis zu bestimmtem Zeichen einlesen: ---
If Eingabe_zeichen = „$“ Then                       'oder z. B. für Enter:  .. = Chr(13)
  Empfang_komplett = 1                              'Wenn bestimmtes Zeichen empfangen, dann fertig
Else                                                'sonst
  Eingabe_string = Eingabe_string + Chr(eingabe_zeichen)  'An Eingabestring anfügen
End If
')
Return
```

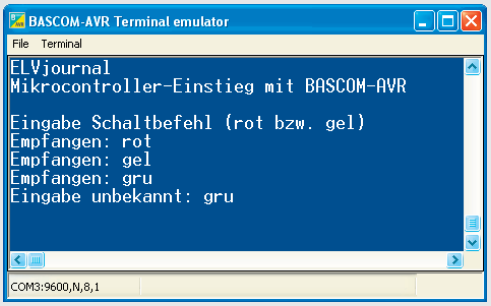

Bild 1 BASCOM-Terminal: Eingabe mit URXC-Interrupt

Erläuterungen:
Mit ON URXC Zeichenempfangen wird festgelegt, dass beim Auftreten des seriellen Empfangsinterrupts URXC zu der Routine Zeichenempfangen gesprungen werden soll. Mit ENABLE werden der serielle Empfangsinterrupt URXC und allgemein alle Interrupts eingeschaltet. Sobald ein Zeichen empfangen wird, erfolgt der Empfangsinterrupt und es wird zu der Interrupt-Serviceroutine Zeichenempfangen gesprungen. Dort wird als Erstes das empfangene Zeichen mit „Inkey()“ in die Variable „Eingabe_zeichen“ eingelesen. (Man liest auch oft „Eingabe_zeichen = UDR“, was das Gleiche bewirkt.) In der Interrupt-Routine kann man nun das neu empfangene Zeichen an die Zeichenkettenvariable „Eingabe_string“ anhängen. Wenn eine festgelegte Zeichenkettenlänge – hier zum Beispiel 3 – erreicht ist, wird die Variable „Empfang_komplett“ auf 1 gesetzt. In der Hauptschleife wird bei jedem Schleifendurchlauf die Variable „Empfang_komplet“ abgefragt. Ist der Inhalt der Variablen 0, passiert nichts bzw. es könnten andere Aktionen abgearbeitet werden. Wenn der Inhalt der Variablen „Empfang_komplett“ 1 ist, wird mit PRINT die empfangene Zeichenkette angezeigt und dann im „SELECT CASE“-Konstrukt je nach Inhalt der Zeichenkette eine Aktion ausgeführt. Dann werden die Variablen „Empfang_komplett“ und „Eingabe_String“ wieder zurückgesetzt für den nächsten Durchlauf. Im Bild 1 lässt sich erkennen, dass immer genau 3 Zeichen empfangen und sofort ausgewertet werden.

In vielen Situationen möchte man nicht eine bestimmte Anzahl von Zeichen empfangen, sondern man möchte Daten von der seriellen Schnittstelle einlesen, bis ein bestimmtes Zeichen, wie zum Beispiel ein Dollarzeichen („$“) oder CR (ASCII 13), empfangen wird. Im Kommentarbereich am Ende des Programms sieht man das Schema für diese Situation.

Vom PC empfangen mit CONFIG SERIALIN

Eine weitere Methode, seriell Daten zu empfangen, lässt sich in BASCOM mit CONFIG SERIALIN realisieren. Durch CONFIG SERIALIN wird erstens ein Puffer für den seriellen Empfang definiert, so dass nicht bei jedem seriellen Empfang ein Interrupt erfolgt, und zweitens wird ein Zeichen definiert, bei dessen Empfang ein Interrupt ausgelöst wird.

```
' BASCOM-Programm
'
' Serieller Empfang vom PC (über USB-UART-Umsetzer UM2102 o. Ä.)
' Mit CONFIG SERIALIN = BUFFERED
' Während die meisten Serialin=Buffered Beispiele auf CR warten und dann mit
' INPUT den Puffer einlesen, ist diese Vorgehensweise mit beliebigen Match-Zeichen verwendbar.
'
'  In: Serielles Signal an Pin d.0 = Rxd
' Out: LEDs mit Vorwiderständen an B.1(rot), B.2(gelb)

$regfile  = „M88def.dat“                  'Verwendeter Chip
$crystal = 3686400                        'Verwendete Frequenz
$hwstack = 40                             'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                             'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                           'Parameter (Daten-Laenge), Rechenbereich Funktionen
$baud = 9600                              '9600 Bit pro Sekunde, 8 Datenbit, Keine Parität, 1 Stoppbit
Led_rot Alias Portb.1
Config Led_rot = Output
Led_gelb Alias Portb.2
Config Led_gelb = Output

Const Matchzeichen = 65                                          '65 fuer „A“   13 fuer CR  usw.
Dim Matchzeichenstring As String * 1
Matchzeichenstring = Chr(matchzeichen)                           'Als Zeichen zuweisen für Stringoperation

Config Serialin = Buffered , Size = 10 , Bytematch = Matchzeichen  'Puffer und Matchzeichen definieren
'bzw. direkt eingesetzt:
'Config Serialin = Buffered , Size = 10 , Bytematch = 13          '65 für A   oder 13 für CR

Enable Interrupts                                                'Interrupts generell einschalten
Dim Empfang_komplett As Bit                                      'Anzeiger für ‚Empfang komplett‘
Dim Eingabe_string As String * 10                                'Eingabezeichenkette
Dim Puffer_overlay As String * 10 At _rs232inbuf0 Overlay        'Puffer-Overlay
Dim Laenge As Byte

Empfang_komplett = 0
Print „Eingabe Schaltbefehl (rot bzw. gelb)“

Do
If Empfang_komplett = 1 Then                                     'Wenn das Flag für ‚Empfang komplett‘ gesetzt ist
  Laenge = Instr(eingabe_string , Matchzeichenstring)            'Position des Match-Zeichens
  Decr Laenge                                                    'Laenge ist eins weniger als Position des Match-Zeichens
  Eingabe_string = Left(eingabe_string , Laenge)                 'String bis Match-Zeichen
  Select Case Eingabe_string
    Case "rot" : Toggle Led_rot
    Case "gelb" : Toggle Led_gelb
    Case Else : Print "Eingabe unbekannt: " ; Eingabe_string
  End Select
  Empfang_komplett = 0
End If
Loop
End

Serial0charmatch:
Eingabe_string = Left(puffer_overlay , _rs_bufcountr0)           'Zeichen aus seriellem Puffer lesen bis Ende=Matchzeichen
Clear Serialin
Empfang_komplett = 1                                             'Flag setzen
Return
```

Erläuterungen:

Im Programm wird mit CONFIG SERIALIN die Größe des Empfangspuffers mit der Länge 10 festgelegt. Außerdem wird mit dem Parameter Bytematch das Zeichen festgelegt, bei dessen Empfang zur Interrupt-Routine gesprungen werden soll. Man kann typischerweise das ASCII-Zeichen 13 (für CR) oder aber auch ein anderes Zeichen wie zum Beispiel ein A (ASCII 65) als Matchzeichen

definieren. Die Interrupt-Routine muss den Namen „Serial0charmatch“ haben und die Interrupts müssen global mit Enable-Interrupts eingeschaltet werden. In der Interrupt-Routine werden die Zeichen bis zum und einschließlich des Matchzeichens einer Zeichenkettenvariablen „Eingabe_string“ zugewiesen und dann mit CLEAR SERIALIN der Puffer geleert. Schließlich wird wie oben (bei URXC) die Variable „Empfang_komplett“ auf 1 gesetzt, damit sie in der Hauptschleife sinnvoll abgefragt werden kann. Wie man sieht, wird die Hauptschleife auf diese Weise erst unterbrochen, wenn die Zeichenkette komplett in den Puffer empfangen wurde.

Es wurden hier zwei von BASCOM erzeugte Variablen verwendet: „_rs232inbuf0“ ist ein Byte-Array, welches als Ring-Puffer für die empfangenen Zeichen fungiert. „_rs_bufcountr0“ ist eine Bytevariable, die die Anzahl der im Puffer vorhandenen Zeichen enthält.

Bis hierher wurden die wesentlichen Methoden für seriellen Datenempfang mit BASCOM beschrieben. Je nach Kenntnissen und Vorlieben des jeweiligen Programmierers findet man weitere Varianten, die sich im Wesentlichen durch mehr oder weniger intensiven direkten Gebrauch von Registern von den hier vorgestellten unterscheiden.

Wetterdatenempfang und FS20-Datenempfang

Mit dem ELV-Modul FS20 WUE (Best.-Nr. J3-10 38 66) wird ein interessantes Modul für den Empfang von Wetter- oder FS20-Daten angeboten. Man kann mit diesem Modul und einem BASCOM-Programm eine sehr individuelle Wetterstation mit Funksensoren aufbauen, aufgrund bestimmter Wetterdaten Aktionen ausführen lassen oder ein FS20-Diagnose-Tool wie das FS20 DT (Best.-Nr. J3-06 68 13) selbst aufbauen bzw. die FS20-Sender in eine eigene Anwendung integrieren. Das Modul kann mit einer 5-V-Spannung betrieben werden. Als Verbindung zwischen dem Modul und dem Mikrocontroller dienen die Leitungen für TxD, RxD und GND gemäß Bild 2. In Bild 2 ist außerdem eine Verbindung vom Interrupt-Ausgang des Moduls zu einem Interrupt-Eingang des Mikrocontrollers dargestellt. Wenn auch ein FS20-Interrupt genutzt werden soll, ist die entsprechende Verbindung vom Modul zum Interrupt-Eingang des Mikrocontrollers zu verbinden.

Die Funktion dieser Beispielanwendung ist wie folgt: Das Modul läuft in der mikrocontrollertypischen Endlosschleife, bis durch ein empfangenes Wetter- oder FS20-Signal ein Interrupt ausgelöst und in der Interrupt-Routine das entsprechende Kennzeichen auf 1 gesetzt wird, welches in der DO-LOOP abgefragt wird.

Da die Wetter-Sender nur ca. alle 3 Minuten senden, ist etwas Geduld nach dem Einschalten nötig.

```
' BASCOM-Programm
'
' Wetterdaten-Empfang mit ELV-Modul FS20 WUE
' Version für Empfang von 2 Sensoren UND FS20-Sender

'  In: Signal von WUE für Wetterempfang an PCINT0 (B.0). Für FS20-Interrupt an PCINT1 (B.1)
'  In: Serielles Signal von WUE an D.0
' Out: Serielles Signal zu WUE an D.1
' Out: Buzzer an Portb.2, LCD Portd.2 bis Portd.7

$regfile  = „M88def.dat“                  'Verwendeter Chip
$crystal = 3686400                        'Verwendete Frequenz
$hwstack = 40                             'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                             'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                           'Parameter (Daten-Laenge), Rechenbereich Funktionen

$baud = 4800                              '4800 Bit pro Sekunde, 8 Datenbit, Keine Parität, 1 Stoppbit

Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Cls
Cursor Off

On Pcint0 Pcint0_isr                      'Wenn ein Pin der Gruppe PCINT0 (0-7) Pegelwechsel hat
Enable Pcint0                             'PCINT 0 bis 7 enablen
Pcmsk0.0 = 1                              'PCINT0 enablen
Pcmsk0.1 = 1                              'PCINT1 enablen
Enable Interrupts                         'Alle Interrupt enablen
Dim Adresse As Word
Declare Function Zahl_zu_4er(byval Zahl As Byte) As Word

Dim Neue_wetterdaten As Bit               'Flag wenn neuer Befehl empfangen wurde
Dim Neue_fs20daten As Bit                 'Flag wenn neuer Befehl empfangen wurde

Dim Wetterdaten(14) As Byte               'Daten vom WUE-Modul
Dim Temperatur_mal_10 As Integer          '10facher Temperaturwert
Dim Temperatur_string As String * 5
```

ELV Funk-Außensensor ASH 2200

ELV Temperatursensor S300 IA

```
Dim Fs20daten(8) As Byte                    'Daten vom WUE-Modul
Dim Hauscode_word As Word At Fs20daten(4) Overlay

Config Pinb.0 = Input                       'Signal an PCINT0 Interrupt-Eingang (B.0)
Wetter_empfang Alias Pinb.0
'Portb.0 = 1                                'Interner Pullup-Widerstand

Config Pinb.1 = Input                       'Signal an PCINT1 Interrupt-Eingang (B.1)
Fs20_empfang Alias Pinb.1
'Portb.1 = 1                                'Interner Pullup-Widerstand

Config Portb.2 = Output                     'Buzzer piept zu Kontrollzwecken wenn Daten empfangen werden
Buzzer Alias Portb.2


Lcd "Wetterdaten/FS20"
Locate 2 , 1
Lcd "mit ELV FS20 WUE"
Wait 2
Cls
Lcd "erwarte Empfang"

'FS20 WUE vorbereiten:
Printbin &H02 ; &H02 ; &HF1 ; &H01 ;      'FS20-Daten sofort seriell ausgeben
Printbin &H02 ; &H02 ; &HF2 ; &H01 ;      'Wetterdaten sofort seriell ausgeben

Do
'--- Wenn neue Wetterdaten vom WUE empfangen wurden Ausgabe an LCD: ---
If Neue_wetterdaten = 1 Then
  Sound Buzzer , 100 , 250                                          'Kontrollton
  Neue_wetterdaten = 0
  Temperatur_mal_10 = Makeint(wetterdaten(7) , Wetterdaten(6))      'Integer aus 2 Byte zusammenfügen

  If Wetterdaten(5) = 1 Then
    Locate 1 , 1 : Lcd Spc(16)                                      '1. Zeile löschen
    Locate 1 , 1                                                    'Sensor 1 in Zeile 1
  Else
    Locate 2 , 1 : Lcd Spc(16)                                      '2. Zeile löschen
    Locate 2 , 1                                                    'anderen Sensor in Zeile 2
  End If
  Lcd "T" ; Wetterdaten(5) ; ": ";                                  'Sensornummer anzeigen
  Temperatur_string = Str(temperatur_mal_10)                'Temperaturwert in String umwandeln..
  Temperatur_string = Format(temperatur_string , " 0.0")    '..und aufbereiten..
  Lcd Temperatur_string ; Chr(223) ; "C  "                  '..für Ausgabe
End If
'--- Wenn neue FS20-Daten vom WUE empfangen wurden Ausgabe an LCD: ---
If Neue_fs20daten = 1 Then
  Sound Buzzer , 50 , 150                                   'Kontrollton
  Neue_fs20daten = 0
```

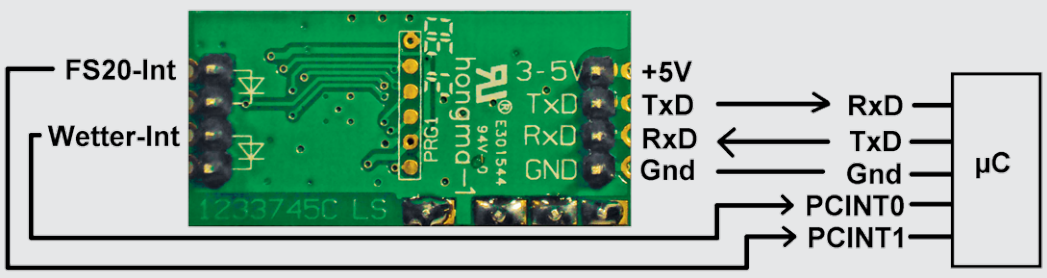

Bild 2: FS20 WUE

```
  Locate 1 , 1
  Lcd Spc(16)                                   '1. Zeile mit Blanks füllen
  Locate 1 , 1
  Lcd "HC:  "                                   'Hauscode
  Adresse = Zahl_zu_4er(fs20daten(4))
  Adresse = Adresse + 1111
  Lcd Adresse
  Adresse = Zahl_zu_4er(fs20daten(5))
  Adresse = Adresse + 1111                      'Jede Stelle um 1 erhöhen
  Lcd Adresse

  Locate 2 , 1
  Lcd Spc(16)
  Locate 2 , 1
  Lcd "Adr: "
  Adresse = Zahl_zu_4er(fs20daten(6))
  Adresse = Adresse + 1111                      'Jede Stelle um 1 erhöhen
  Lcd Adresse
  Locate 2 , 12
  Lcd Hex(fs20daten(7)) ; " "                   'Befehl
  Lcd Hex(fs20daten(8))                         'Befehlserweiterung
End If
Loop
End

Pcint0_isr:
If Wetter_empfang = 1 Then                      'Wetterdaten empfangen
  Inputbin Wetterdaten(1) , 14                  'Daten vom WUE einlesen. 14 Byte.
  Neue_wetterdaten = 1
End If
If Fs20_empfang = 1 Then                        'Wetterdaten empfangen
  Inputbin Fs20daten(1) , 8                     'Daten vom WUE einlesen. 8 Byte.
  Neue_fs20daten = 1
End If
Return

Function Zahl_zu_4er(byval Zahl As Byte) As Word
'Eingabe: Byte-Dezimalzahl wie von WUE empfangen
'Ausgabe: Zahl im 4er-Zahlensystem  (nur Ziffern 0 bis 3)
Local Temp_string As String * 4                 'Zum Zusammenbauen der Zahl als Zeichenkette
Local Stelle As Byte
Local Rest As Byte
Local Zeichen As String * 1
Temp_string = "    "
Stelle = 4                                      'Byte-Zahl (max 255) wird maximal 4 stellig (3333).
Rest = 0
Zeichen = " "
Do
  Rest = Zahl Mod 4
  Zeichen = Str(rest)
  Mid(temp_string , Stelle , 1) = Zeichen
  Decr Stelle
  Zahl = Zahl / 4
Loop Until Zahl = 0
Zahl_zu_4er = Val(temp_string )
End Function
```

Erläuterungen:
Bei Wetterdaten- oder FS20-Empfang bewirkt der jeweilige Interrupt, dass die Interrupt-Routine der PCINT0-Interrupt-Gruppe angesprungen wird. In der Interrupt-Routine wird geprüft, welche der beiden Interruptarten vorliegt (Wetterdaten oder FS20). Dementsprechend werden die anliegenden seriellen Daten mit INPUTBIN in das jeweilige Datenarray eingelesen und ein Kennzeichen („Neue_Wetterdaten" bzw. „Neue_FS20_Daten") auf 1 gesetzt. In der Hauptschleife werden diese Kennzeichen abgeprüft, und wenn sie vorhanden sind, erfolgt die entsprechende Auswertung und Anzeige.

Direkt vor der Hauptschleife wird das Modul mit PRINTBIN so konfiguriert, dass sowohl Wetter- als auch FS20-Daten bei Empfang sofort seriell an den Mikrocontroller gesendet werden.

Selbstverständlich könnte das Modul auch nur für Wettterdaten oder nur für FS20-Daten verwendet werden.

Die vorgestellten BASCOM-Befehle für seriellen Datenempfang und serielles Datensenden lassen sich in Verbindung mit vielen weiteren Sensoren oder auch ELV-Modulen wie beispielsweise der Realtime-Clock mit DCF77-Funktion RTC-DCF (Best.-Nr. J3-13 05 41) oder dem USB-Stick-Interface STI 100 (Best.-Nr. J3-07 59 50) einsetzen.

Ausblick

Im nächsten Artikel der Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR" werden die Grundlagen der I²C-Kommunikation dargestellt und anhand von BASCOM-Programmen die Kommunikation mit ELV-Modulen gezeigt.

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
- Produktübersicht Bascom: **www.elv.de/bascom.html**

Empfohlene Produkte/Bauteile:	**Best.-Nr.**	**Preis**
BASCOM-(Demo-)Lizenz von MCS Electronics www.mcselec.com	–	–
Atmel-AVRISP-mkII-Programmer	J3-10 03 55	€ 39,95
oder myAVR-Board MK2	J3-10 90 00	€ 49,–
Netzteil für myAVR-Board MK2	J3-10 90 01	€ 6,95
ATmega88	J3-10 07 62	€ 3,95
100-nF-Kondensator	J3-10 03 17	€ 0,08
Batteriehalter für 3x Mignon	J3-08 15 30	€ 0,75
Batterieclip für 9-V-Block-Batterie	J3-08 01 28	€ 0,30
BASCOM-Buch	J3-10 90 02	€ 54,–
Experimentier-Board 1202B	J3-07 72 89	€ 12,95
Schaltdraht-Sortiment	J3-05 47 68	€ 5,95
LED-Set	J3-10 63 56	€ 3,95
oder Leuchtdioden	J3-10 66 60	€ 1,65
und Widerstände	J3-10 66 57	€ 1,85
Piezo-Signalgeber	J3-00 73 87	€ 0,95
Mikroschalter und -taster	J3-10 66 67	€ 2,80
LC-Display, 2x 16 Zeichen	J3-05 41 84	€ 6,95
oder myAVR-LCD-Add-on		
Pin-Ausrichter	J3-00 84 63	€ 4,95
USB-UART-Umsetzer UM2102	J3-09 18 59	€ 5,95
USB-UART-Umsetzer mit Optokopplung UO2102	J3-10 49 66	€ 12,95
Real-Time-Clock mit DCF77 RTC-DCF	J3-13 05 41	€ 11,95
FS20- und Wetterdaten-Empfänger FS20WUE	J3-10 38 66	€ 14,95
Funk-Temperatursensor S300IA	J3-07 36 06	€ 39,95
Funk-Außensensor ASH2200-1	J3-07 36 05	€ 29,95
GPS-Empfangsmodul NL-552ETTL	J3-09 42 41	€ 34,95
USB-Stick-Interface STI 100	J3-07 59 50	€ 27,95
FS20-Diagnosetool FS20 DT	J3-06 68 13	€ 59,95

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

Preisstellung April 2014 – aktuelle Preise im Web-Shop