Varptr

Durch Varptr können also Adressen gewonnen werden.
Varptr ist mit Loadadr verwandt, jedoch bezieht sich Varptr auf den Datenbereich, also auf die Arbeitsregister, die Funktionsregister und das SRAM. Im Folgenden nennen wir diese Bereiche zusammengefasst, schlicht RAM. Auch wenn Varptr nicht auf Elemente des ROM-Bereiches, dem Programmspeicher anwendbar ist, kann das Resultat ein fixer Wert, welcher zu einem integralen Bestandteil des Programmcodes als Konstante werden kann. So lassen sich auch z.B. durch Varptr gebildete Konstanten mittels Data-Zeilen formieren.
Mit Varptr kann also zur Kompilierzeit eine Konstante liefern oder zur Laufzeit als Funktion.
Es gibt verschiedene Weisen mit diesem Befehl umzugehen und entsprechende Regeln in seiner Anwendung.

Anwendung auf Variablen

Eigentlich ist es für den Programmierer weniger interessant zu wissen, an welcher Adresse seine Variablen liegen, ist doch der vergebene Variablenname das Symbol für die Speicherstelle, bzw. für die Adresse selbst. Es kann aber Gründe dafür geben,warum die Adresse von Interesse sein kann.

Zuweisung an ein Variable
BASCOM-Quellcode

```
1. dim Var as word, Pointer as word
2. Pointer = varptr(Var)
```

Zuweisung an eine Konstante
BASCOM-Quellcode

```
1. const Var_ADR = varptr( "Var" )
```

Hierbei ist darauf zu achten, dass bei der Zuweisung an eine Konstante der Variablenname in Anführungszeichen steht, wohingegen bei der Variablenzuweisung nur der Variablenname in den Klammern steht.
Die so gewonnenen Adressen könnten als Parameter in Funktionsaufrufen dienen oder in einer Tabelle angelegt werden.

Anwendung auf Funktionsregister

Varptr(REGISTERNAME) liefert immer die RAM Adresse. Bei Registern innerhalb des IO-Adressraumes (00h bis 3fh, siehe Atmega Datenblatt), kann durch den in Anführungszeichen gesetzten Registernamen bestimmt werden, dass die IO-Adresse zurückgegeben wird. Die Rückgabe derIO-Adresse durch den Einsatz der Anführungszeichnung, wirkt natürlich nur beiFunktionsregistern innerhalb des IO-Bereiches. Ohne Anführungszeichen wird stets die RAM-Adresse zurückgegeben.
Das Prinzip entspricht ansonsten der Anwendung auf Variablen. Ausnahmen sind nur im IO-Bereich erlaubt.

Zuweisung an ein Variable
BASCOM-Quellcode

```
1. dim Reg_ADR as word
2. Reg_ADR = varprt(PORTB) 'Muss ohne Anführungszeichen erfolgen.
```

Zuweisung an eine Konstante
BASCOM-Quellcode

```
1. const a1 = varptr(UDR0)
```

```
2. const a2 = varptr("UDR0")
3. const a3 = varptr(PORTB)
4. const a4 = varptr("PORTB")
```

Beispiel

Die Aufgabe der folgenden Subroutine ist, einen beliebigen Port entsprechend der übergebenen Bitmasken einzurichten, also das dazugehörige Datenrichtungsregister zu stellen und die Pullups der Eingänge zu schalten bzw.den Initialpegel der Ausgangspins zu setzen. Nebenbei wird auch eine weitere Variante der Konstantenbildung, nämlich innerhalb der Parameterübergabe dargestellt (sieht man selten).

BASCOM-Quellcode

```
1. $regfile = "m2560def.dat"
2. $crystal=4000000
3. $hwstack=40
4. $swstack=16
5. $framesize = 32
6. declare sub configPort(byval IO_Port as word , byval DDR_bm as byte , byval Level_bm as byte)
9. configPort varptr(PortA) ,(2^PA0 + 2^PA1) ,(2^PA1 + 2^PA3)
12. configPort varptr(PortB) , &B01010101 , &B10101010
13. configPort varptr(PortC) ,(2^0 + 2^1) ,(2^1 + 2^3)
15. do
17. loop
19. '_____
21. sub configPort(byval IO_Port as word , byval DDR_bm as byte , byval Level_bm as byte)
24. out IO_Port , level_bm
25. decr IO_Port'DDR-Adresse bilden, siehe Datenblatt - Register summary
26. out IO_Port , DDR_bm
28. end sub
```

Alles anzeigen

Weitere Anwendungsfälle:

Array als Referenz direkt per Register übergeben