RN-Wissen.de - Bascom Inside

from RN-knowledge, the free knowledge base

Contents

Bascom Inside

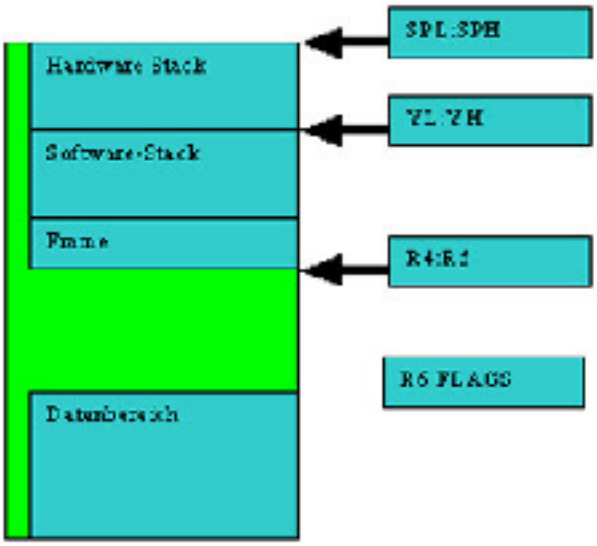
Stacks & Frame

Bascom works with three areas:

- HW stack,
- Software Stack and
- Frame.

Described is the "HELP". Additionally, the

- Register r6 for special flags used (bit # 2 in register 6 is the ERR bit)



Hw- (hardware) stack

The hardware stack is like the "normal" stack, which is supported by the controllers through the stack pointer and corresponding commands. Bascom, of course, use it for CALL / RET / RETI / PUSH and POP.

Sw (software) Stack

For pointer to temporary data and the parameters for SUB and FUNCTION Bascom has defined its own stack. It is below the hardware stack.

Frame

For the temporary data even the frame area is used

Definition

```
$ Framesize = 32 'the size of the frame
$ Swstack = 32 'the size of the software stacks
$ Hwstack = 32 'the size of the hardware stack
```

The actual sizes required are difficult to calculate. Some clues:

- For each interrupt before 32 bytes HW stack will certainly overdone, just to insure all the stops.
- Each "LOCAL" Data Frame consumes space in its size + 2 bytes on the stack Soft
- Each parameter of a "SUB" needs 2 bytes Soft Stack + data size when in addition is "byval" specified.
- data conversions number -> string need the string length (PRINT!)

```
 For a decent program, the default values are much too small in any case.
If in doubt as large as possible, especially the soft stack.
```

Local data

Place a course must first be created for local data. Bascom used to the frame, the address of the data it sends

it to the software stack. All data types are treated the same, only the reserved byte length is relevant. Bascom writes the required length of the item by R24 and calls an internal function auf.Diese ensures only the frame pointer (r4: r5) on the software stack, increasing the frame pointer to the required value and returns

reserve space

LOCAL single byte AS BYTE

```
 ....
LDI r24 $ 01 // desired length = immediate value
CALL AddFrame
 ....
```

```
 AddFrame:
ST -=Y, r5 // Store & Save Frampointer Hi
ST -=Y, r4 // Store & Save Frampointer LO
ADD r4, r24 // add the desired length of the frame pointer
CLR r24
ADC r5, r24 // ev carry =. Overflow
RET
```

As a result, the address of the local date is on

```
 Y + 0 // Address LSB
Y + 1 // address MSB
```

and is available there (of course only within the function or subroutine) to available

unreserve

Before the "return" of the subroutine or function, of course, everything must be restored Bascom, of course counted how much space was needed, and he also knows how many such addresses he has put on the soft stack.

```
 ... ..
ADIW YL, $ 0002 // Add to the software stack pointer
LDI r24 $ 01 // subtracting from the frame pointer (eg 1 byte)
CALL SubR4
// Return RET Sub or Function
```

```
 SubR4:
SUB r4, r24
CLR r24
SBC r5, r24
RET
```

Call Sub & Function

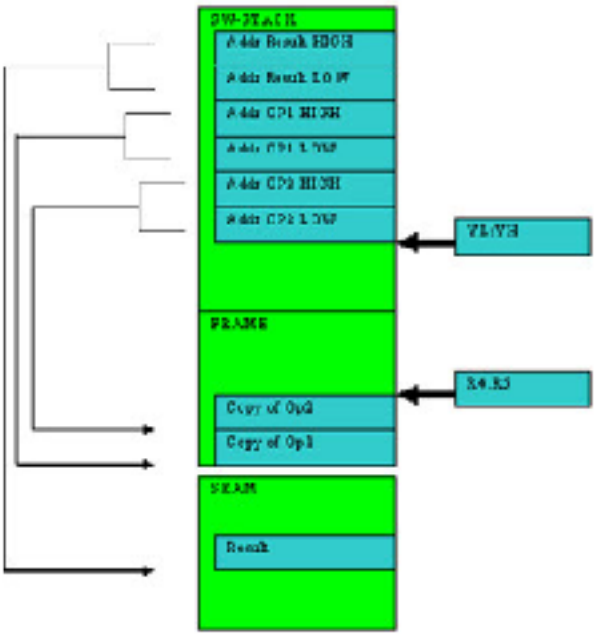
The parameters to the function / sub are passed via the soft stack. Except for the actual CALL the Function / Sub HW stack is not used here.

There are to be two transfer variants: BYVAL and BYREF, but it is always BYREF (address of) used, the only difference is that in BYVAL the address of a data copy is passed instead of the original. The order is from left to right, wherein the address of the result (function) is first placed on the stack is soft.

Following declarations have in our program the same code

```
 Declare Function MyFunc (byval as byte Op1, Op2 byval as byte) as byte
Declare Sub MySub (byref result as byte, byval as byte Op1, Op2 byval as byte)
```

The situation when the function / sub now looks like this, the HW stack not taking into account



The function / sub is now (at entry)

```
 OP2 Y + 0/1
OP1 Y + 2/3
Result on Y + 4/5
```

Uses the function / sub even now the SW stack or frame, it must of course be taken into account in the Y-offset and before the return of the above condition must also be restored.

See also

- Bascom
- Bascom Inside Code

From " http://rn-wissen.de/wiki/index.php?title=Bascom_Inside&oldid=13218 "

Categories : Microcontrollers | Basics | Software | Source Bascom