Mikro Forth Vocabulary Status: 01.11.2012 A: Assembler Word F: Forth Word C: Compiler Word Juergen Pintaske August 2017

Word	Stack	Comment	Type
.	(n –)	Sends TOS to Port B; Data Direction Bits of Port B are all set to 1 (OUTPUT)	A
-	(a b – res)	Res = a – b TOS is subtracted from TOS-1	A
/	(a b – res rem)	Divides a by b res = a/b (no remainder), remainder is TOS	A
:		: starts the definition of a new Forth Word	C
;		; ends the definition of this new word	C
<	(a b – flag)	a b < a larger than b ? puts 1 (TRUE) flag onto the stack if a < b ist, else the flag 0 onto the stack.	A
>	(a b – flag)	a b > puts 1 (TRUE) onto the stack if a > b ist, else 0.	F
>com	(n –)	sends TOS to the serial COM Interface. BUT: before, the COM Interface has to be initialized using INITCOM.	A
>eprom	(w a –)	Writes the value w to address a of the EEPROM. . See as well eprom>	A
>R	(a –)	Moves the top of stack TOS to the Return Stack See as well R>	A
>sram	(w a –)	Saves the value w into a RAM cell at address a	A
1 to 255	(– n)	Transfers the number onto the stack.	AC
and	(a b – res)	res = a and b	A
begin.until	(–) (n –)	begin Bef1 Bef2..Befn until repeats the instructions Bef1, Bef2, . Befn, until the word until reads TOS = 1 flag.	A
blink	(b hp –)	*bitpattern hp* blink results in *bitpattern* on Port B, waits for *hp* milliseconds, then outputs 0 on Port B, waits again for *hp ms*	F
com>	(– n)	Receives a byte via the COM Interface and saves it onto the stack. See as well >com.	A
DDBitB	**(bit flag –)**	***bit flag* DDBitB sets the relevant bit line of Port B to OUTPUT, if *flag* = 1, else sets it to INPUT.**	**A**
DDBitD	(bit flag –)	*bit flag* DDBitD sets the relevant bit line of Ports D to |OUTPUT, if *flag* = 1, else sets bit to INPUT.	A*
DDRB	(b –)	Writes b into Data Direction Register of Port B.	A
DDRD	(d –)	Writes d into Data Direction Register of Port D.	A*
do ... loop	(e a –) (–)	*e a* do Bef1 Bef2 ... Befn loop repeats the instructions Bef1, Bef2, ..., Befn; The loop starts with the index a and runs until value e (included) This loop runs at least once. Within the loop, the program can check the current value via I. do loop	A
drop	(n –)	Discards the TOS value	A
dup	(n – n n)	Duplicates the TOS value	A
end	(–)	Executes an endless loop and is suggested as last instruction at the end of a program.	A
eprom>	(a – w)	Reads the value at the EEPROM address a, and copies the value onto the stack. See as well >eprom	A
getOSCCAL	(– n)	Puts the value OSCCAL onto the Stack. See as well. SetOSCCAL	A
I	(– n)	Copies the loop index I of a do-loop onto the stack. Only allowed to be executed between do and loop.	A
i2cread	(ACK – value)	Read a value from a Slave Chip; if ACK = 0, then an Acknowledge Signal is sent.	A

i2cstart	(–)	The Start Signal for an I2C Bus transmission is sent. (SDA changes from 1 to 0; then SCL from 1 to 0)	A
i2cstop	(–)	Initializes the I2C Bus (SCL and SDA set to 1); Data Direction Bits for SDA (PortB.5) and SCL (PortB.7) are set.	A
i2cwrite	(value/Addr – ACK)	One single Byte is sent to the Slave; the Acknowledge Signal is copied to the Stack.	A
init		This is a System Word, and should not be changed or removed.	F
initCom	(–)	Initializes the COM Interface: Pin 2 D0 = RxD Pin 3 D1 = TxD Baud rate = 9600 8 Bit No parity bit	A
initInt0	(signaltyp –)	*signaltyp* initInt0 configures INT0 (Port D2) as Interrupt Input and sets this Input to High. Dependant on the *signaltyp* value, different Input Signals will trigger an Interrupt: 0: change HIGH to LOW 1: change LOW to HIGH Interrupts are generally allowed.	A
initInt1	(signaltyp –)	Same as initInt0, but related to Input INT1 (Port D3).	A
initT0ovf	(typ preset –)	typ preset initT0ovf initializes Timer0 Interrupt: typ 0: Stop Timer / deactivate 1: System clock /1 2: System clock /8 3: System clock /64 4: System clock /256 5: System clock /1024 6: ext. Clockt, decending on T0 7: ext. Clock, ascending on T0 Timer Interrupts Timer-are allowed, all other Interrupts are allowed *Preset* value within a Interrupt routine has always to be set again.	A
inPortB	(bit – flag)	*bit* InPortB Reads the relevant INPUT bit of Port B and puts 0/1 onto stack when Input is High or Low. See DDRB, DDBitB	A
inPortD	(bit – flag)	*bit* InPortD Reads relevant INPUT bit of Port D, puts 0/1 onto the stack when Input is High or Low.See DDRD, DDBitD Reads the relevant INPUT bit of Port D and puts 0/1 onto the stack when Input is High or Low. See DDRD and DDBitD	A
int0	(–)	This word is called when INT0 Interrupt is triggered. Construction of an INT0 word:: : int0 pushreg ... <any words> ... popreg reti;	F

		During the execution of the `int0` word all other Interrupts are inhibited.	
`int1`	(–)	This word is called when the INT1 Interrupt is triggered. This can be freely defined.	F
`not`	(*flag* –)	Replaces *flag* by its logic complement.	F
`or`	(a b – res)	res = a or b	A
`outPortB`	(bit flag –)	*bit flag* `outPortB` sets the Output bit of Port B to High/Low if the *flag* value is `0 / 1`. See `DDRB` and `DDBitB`	A
`outPortD`		*bit flag* `outPortD` sets the Output bit of Port D to High/Low if the *flag* value is `0 / 1`. See. `DDRD` and `DDBitD`	A
`over`		Copies the second stack element onto TOS.	A
`popreg`		All internal registers r16 to r29 are set	A
`pushreg`		All internal registers r16-r29 are saved into (r2 - r15).	A
`R>`		Moves the highest element of the Return Stack onto the Parameter Stack. See `>R`	A
`reti`		Interrupts are allowed	A
`rot`		Rotates the top 3 stack values	A
`sei`		Same as `reti`	A
`setOSCCAL`		Writes the value n into the OSCCALRegister.	**A**
`setTimer0`	(preset –)	Sets the Preset Value of Timer0 (TCNT0).	**A**
`skipIf`	(n –)	Skips over the next Instruction if TOS equals 1 (TRUE) ist.	A
`sram>`	(a – w)	Reads the value of the SRAM cell at address a and copy value to stack	A
`stackInit`		A System word, this should not be modified or removed.	A
`swap`	(n m – m n)	SWAP exchanges to two numbers at TOS and TOS-1 of the stack.	A
`T0ovf`	(–)	This word is called when the Timer0 Overflow Interrupt is triggered, to build an Interrupt word See `int0`. Within the `T0ovf` word, there might be the need to set the Preset value of the Timer using `setTimer0`.	F
`Ta0?`	(– bit)	Puts 1/0 onto the Stack, if Ta0 is open / closed (D2=1/0) PortD.2 is configured automatically.	F
`Ta1?`		Puts 1/0 onto the Stack, if Ta1 is open / closed (D3=1/0) PortD.3 is configured automatically.	F
`toggleB`	(–)	Toggles the register of Port B.	A
`VARIABLE`		Starts the definition of a Variable. `VARIABLE` *abc* defines Variable *abc*. As result of this definition, the Compiler reserves a memory location in EEPROM. Next, *abc* puts the address of the relevant memory location onto the stack.	C
`wait`	(s –)	Waits for s seconds.	F
`wait1ms`	(–)	Waits for 1 millisecond	A

waitms	(n –)	Waits for n ms.	A
wdogOff	(–)	Switches the Watchdog Timer off.	**A**
wdogOn	(–)	Switches the Watchdog Timer on.	A
xor	(a b – res)	res = a xor b	A
=	(a b – flag)	*a b* = puts 1 (TRUE) onto the Stack, if *a* = *b* , else puts 0 (FALSE).	A
+	(a b – res)	res = a + b	A
*	(a b – res)	res = a * b	A