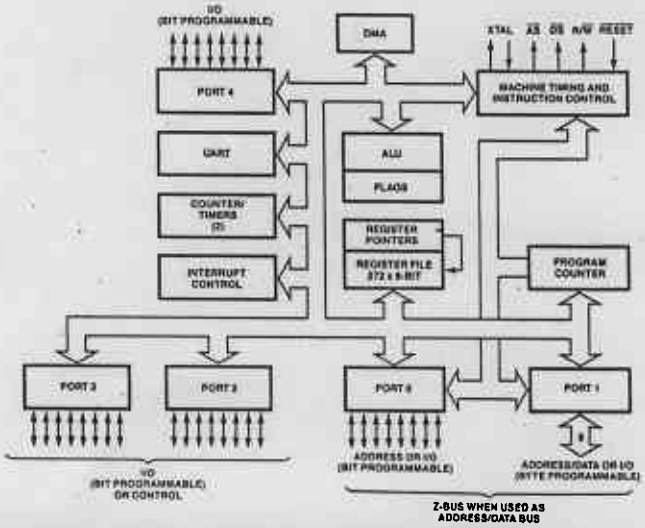
Jetzt verfügbar! Super8™-FORTH in Silicon

- FORTH im 8Kbyte On-Chip-ROM
- Extern wird nur ein RAM, ein V.24-Treiber und ein 19,6608 MHz Quarz benötigt
- 3 Ports verfügbar
- Transparente Entwicklungsumgebung
- Kompiliert ROM-fähigen Code
- Zugriff auf alle On-Chip-Resourcen
- Ausführliches Handbuch
- Quellcode von S8-Assembler und S8-FORTH werden mitgeliefert



Muster 0887520PSC bei:

FORTHWORKS Ulrike Schnitter
Nelkenstr. 52
W-8044 Unterschleißheim
Tel.: 089-310 33 85

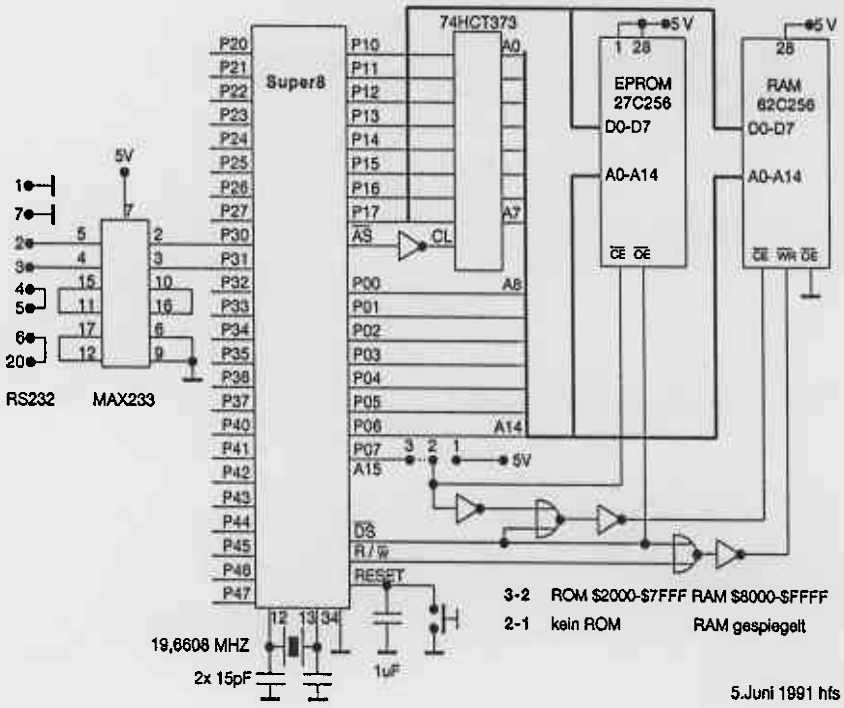
Leider hat sich im On-Chip-ROM ein Fehler eingeschlichen, der sich beim Start eines im EPROM erweiterten FORTH-Systems bemerkbar macht. Der Fehler liegt in dem Initialisierungsprogramm, das beim Neustart das im EPROM abgespeicherte Programm ins RAM an die ursprüngliche Adresse kopiert. Der Fehler tritt bei größeren Programmen (ab ca. 7kB) auf. Es gibt jedoch eine Möglichkeit, diesen Fehler zu beheben:

Die Adressierung von ROM und RAM muß wie auf beiliegendem Schaltbild geändert werden. In der Schaltung werden ausschließlich Standard-Bauteile verwendet. Ein GAL kann das sicherlich eleganter.

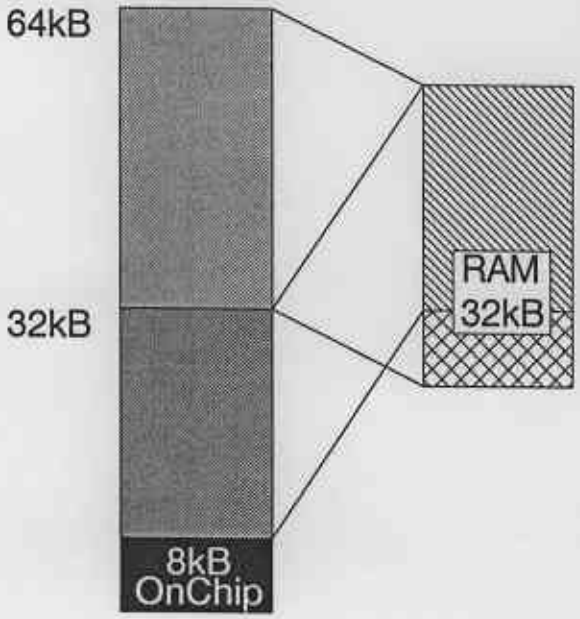
Jumper 1-2
Dieser Betriebsmode wird für die Programmentwicklung verwendet. Der Trick ist, daß sich das RAM auch im Bereich von $2000 bis $7FFF spiegelt. Der Speichertest 'glaubt' nun es wäre RAM von $2000 bis $FFFF vorhanden. Alle Erweiterungen werden beim Kompilieren ab Adresse $2050 ins Dictionary eingetragen. Es stehen jedoch nur 24kB zur Verfügung. Dies birgt die Gefahr, daß das S8-FORTH nicht feststellen kann, wie groß der freie RAM-Bereich noch ist, falls man mehr als 24kB kompiliert. Dies ist wohl auch der einzige Nachteil, den man sich mit dieser Schaltung einhandelt. Mit SAVESYSTEM wird die Erweiterung auf den Massenspeicher kopiert und dann in ein EPROM gebrannt.

Jumper 2-3
EPROM einstecken und die Jumper auf 2-3. Beachten Sie bitte, daß das Image ab $2000 beginnen muß. Der oben beschriebene Fehler tritt jetzt nicht mehr auf, da jetzt zwar das EPROM erkannt wird, das Programm aber nicht vom ROM in das RAM kopiert werden muß, da es sich an der selben Adresse befindet, an die es auch kompiliert wurde. Ein Vorteil: man hat jetzt wieder das **ganze** RAM zur Verfügung!! Der Ablauf der Initialisierung beim Reset ist im Handbuch in Kapitel 2.3 und 2.6.1 beschrieben.

Umseitig finden Sie eine graphische Darstellung der Memory-Map für beide Betriebszustände. Für weitere Fragen stehen wir Ihnen gerne zur Verfügung.



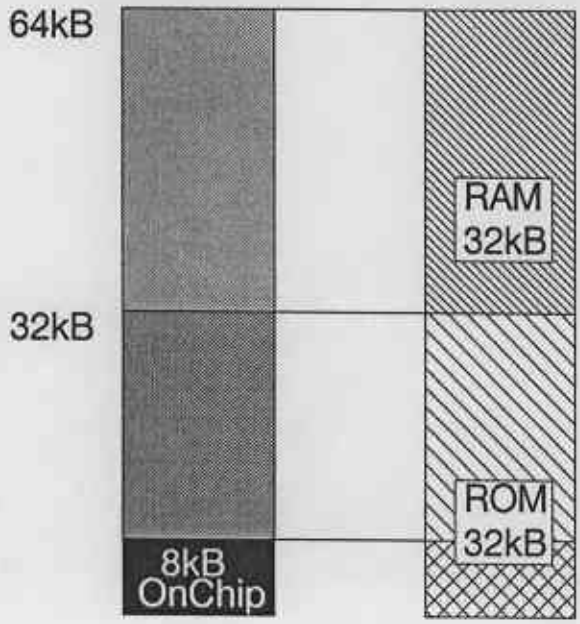
Jumper 1-2



Programmentwicklung:
das Programm wird ab $2000 kompiliert

- On Chip-ROM 8kB
- Programmentwicklung im RAM 24kB
- Gespiegeltes RAM ab $2000 24 kB
- nicht nutzbares RAM

Jumper 2-3



Das Programm befindet sich im ROM
ab Adresse $2000.
Das RAM ist wieder frei für Erweiterungen.

- On Chip-ROM 8kB
- RAM ab $8000
- Programm im ROM ab $2000 24kB
- nicht nutzbares ROM

Speicheraufteilung des Super8 FORTH

SCR 2 **Low-Level Interrupt laden**

```
\ Interrupt in High-Level                          hfs 20:42 18.04.91

\  High-Level Interrupt in FORTH
\  DECIMAL 2 7 THRU DECIMAL 13 LOAD

\  Low-Level Interrupt in Assembler
   DECIMAL 8 13 THRU
```

```
 Zilog Super8-FORTH V1.0
<Alt F>
Screenfile : S8.ASM <Enter>
empty-buffers 1 load <Enter> >>>>>>>>>>>>>>>
, is redefined
# is redefined >
@ is redefined >>>>>>>>>>>>
BL is redefined >>>>>
Length of Assembler: 5531 Bytes
 ok
<Alt F>
Screenfile : ZUSATZ.SCR <Enter>
empty-buffers 1 load <Enter> >>>>>>>>>>>>>>>>>>>
RP@ is redefined
SAVESYSTEM is redefined
DEFER is redefined > ok
<Alt F>
Screenfile : ISR.SCR <Enter>
empty-buffers 1 load <Enter> >>>>>>>>
singletask <Enter> ok
start_isr <Enter> ok                    Jetzt läuft die Interrupt-Service-Routine
words <Enter>
START_ISR  CONNECT  INT_OUTPUT  INT_INPUT  INT_EMIT  INT_KEY  (INT_EMIT
INT_EMIT?  INT_KEY?  (INT_KEY  KEY_ISR  INT_INIT  DEFER  SAVESYSTEM
RP@  SAVE  EMPTY  HCLEAR  FORGET  (FORGET  REMOVE  'REMOVE  | REM-VOCS
| REM-WORDS  | -LINK  | ENDPOINTS  INPUT:  INVECTOR:  OUTPUT:  OUTVECTOR:
2CONSTANT  2VARIABLE  (ERRORTEXT  | ERRTEXT  SINGLETASK  MULTITASK
| (PAUSE  | ((PAUSE  PAUSE  SCAN  SKIP  E!  E@  EC!  EC@  DMIN  DMAX
D>  D=  D0>  D0=  D0<  UD2/  D2/  D2*  2ROT  2OVER  HEAP?  UMAX
RLEN  SLEN  DISKORG  TASK#  LASTTASK  TASKORG  CODE  PROC  ;CODE
END-CODE  END-PROC  ASSEMBLER  COLD  'COLD  ABORT  'ABORT  IDENT
QUIT  INTERPRET  SAVESYSTEM  +LOOP  LOOP  DO  ?DO  REPEAT  UNTIL
WHILE  BEGIN  THEN  ELSE  IF  >RESOLVE  >MARK  <RESOLVE  <MARK  IS
DEFER  VOCABULARY  DOES>  ;  :  CONSTANT  VARIABLE  CREATE  HEADERS
-HEADERS  |  RESTRICT  IMMEDIATE  REVEAL  HIDE  ABORT"  ERROR"  (ERROR
\\  \  (  .(  ."  "  [']  '  ASCII  $,  LITERAL  [COMPILE]  COMPILE
]  [  FIND  ?CAP  WORD  .S  DUMP  WORDS  VOCS  ORDER  ONLYFORTH
FORTH  ALSO  DEFINITIONS  CONTEXT  >LINK  >BODY  >NAME  LINK>  BODY>
NAME>  L>NAME  N>LINK  -->  THRU  LOAD  BLOCK  BUFFER  UPDATE  FLUSH
EMPTY-BUFFERS  SAVE-BUFFERS  R/W  COMMAND!  NUMBER?  CONVERT  .
.R  U.  U.R  D.  D.R  #>  SIGN  #S  #  HOLD  <#  HEX  DECIMAL  EXPECT
STOP?  CR  SPACES  SPACE  TYPE  SIO_INPUT  KEY  KEY?  SIO_OUTPUT
EMIT  EMIT?  INTERROR  NOTFOUND  NODEFER  ?PAIRS  ?STACK  ?ERROR
ERROR  V,  VALLOT  H,  HALLOT  ,  C,  ALLOT  ->VAR  DEPTH  WDP@
HERE  HEAP  PAD  TIB  FIRST  BLANK  ERASE  FILL  CMOVE>  CMOVE  -TRAILING
COUNT  SPR@  SPR!  GR!  GR@  OFF  ON  +!  */  */MOD  MOD  /  /MOD
M/  M/MOD  *  M*  UM/MOD  UM*  DABS  D<  CASE?  UWITHIN  MAX  MIN
ABS  U<  >  =  <  0>  0=  0<  DNEGATE  D-  D+  2+  1+  1-  2-  NEGATE
-  +  FLIP  U2/  2/  2*  NOT  XOR  OR  AND  J  I  2DROP  2SWAP  2DUP
ROLL  PICK  -ROT  ROT  NIP  TUCK  OVER  ?DUP  PUSH  RDROP  R@  R>
>R  RP!  RP@  DROP  SWAP  DUP  DCLEAR  SP!  SP@  !  @  C!  C@  ?HEAD
LAST  CURRENT  ERRORHANDLER  OUTPUT  INPUT  DPL  HLD  BASE  #TIB
SPAN  R#  SCR  >IN  BLK  STATE  R0  S0  BL  TRUE  FALSE  NOOP  HDP
VLEN  VDP  DP  VOC-LINK  RAMORG  LEAVE  ?BRANCH  BRANCH  EXECUTE
EXIT  ok
<Ctrl S>  xlerb <Enter>    Ctrl S (nicht sichtbar) sperrt den Output.
                           Die Zeichen xlerb werden in den Buffer eingelesen, aber
                           nicht am Bildschirm angezeigt
<Ctrl Q> xlerb "XLERB" ?  ???    Ctrl Q (nicht sichtbar) gibt den Output wieder frei.
```

SCR 2 **High-Level Interrupt laden**

```
\ Interrupt in High-Level                          hfs 20:42 18.04.91

\  High-Level Interrupt in FORTH
   DECIMAL 2 7 THRU DECIMAL 13 LOAD

\  Low-Level Interrupt in Assembler
\  DECIMAL 8 13 THRU
```

```
 Zilog Super8-FORTH V1.0
<Alt F>
Screenfile : S8.ASM <Enter>
empty-buffers 1 load <Enter> >>>>>>>>>>>>>>>
, is redefined
# is redefined >
@ is redefined >>>>>>>>>>>>
BL is redefined >>>>>
Length of Assembler: 5531 Bytes
 ok
<Alt F>
Screenfile : ZUSATZ.SCR <Enter>
empty-buffers 1 load <Enter> >>>>>>>>>>>>>>>>>>>
RP@ is redefined
SAVESYSTEM is redefined
DEFER is redefined > ok
<Alt F>
Screenfile : ISR.SCR <Enter>
empty-buffers 1 load <Enter> >>>
UTC is redefined
UIE is redefined
IMR is redefined
UIO is redefined >>>>>>> ok
singletask <Enter> ok
start_isr <Enter> ok                          Jetzt läuft die Interrupt-Service-Routine
words <Enter>
START_ISR  CONNECT  INT_OUTPUT  INT_INPUT  INT_EMIT  INT_KEY  (INT_EMIT
INT_EMIT?  INT_KEY?  (INT_KEY  KEY_ISR  INT_INIT  DIS_INT  ENA_INT
UIO  IMR  UIE  UTC  WR_P  CNT  RD_P  XQS  ^S  ^Q  ;ISR  ISR:  DEFER
SAVESYSTEM  RP@  SAVE  EMPTY  HCLEAR  FORGET  (FORGET  REMOVE  'REMOVE
| REM-VOCS  | REM-WORDS  | -LINK  | ENDPOINTS  INPUT:  INVECTOR:
OUTPUT:  OUTVECTOR:  2CONSTANT  2VARIABLE  (ERRORTEXT  | ERRTEXT
SINGLETASK  MULTITASK  | (PAUSE  | ((PAUSE  PAUSE  SCAN  SKIP  E!
E@  EC!  EC@  DMIN  DMAX  D>  D-  D0>  D0=  D0<  UD2/  D2/  D2*
2ROT  2OVER  HEAP?  UMAX  RLEN  SLEN  DISKORG  TASK#  LASTTASK  TASKORG
CODE  PROC  ;CODE  END-CODE  END-PROC  ASSEMBLER  COLD  'COLD  ABORT
'ABORT  IDENT  QUIT  INTERPRET  SAVESYSTEM  +LOOP  LOOP  DO  ?DO
REPEAT  UNTIL  WHILE  BEGIN  THEN  ELSE  IF  >RESOLVE  >MARK  <RESOLVE
<MARK  IS  DEFER  VOCABULARY  DOES>  ;  :  CONSTANT  VARIABLE  CREATE
HEADERS  -HEADERS  |  RESTRICT  IMMEDIATE  REVEAL  HIDE  ABORT"
ERROR"  (ERROR  \\  \  (  .(  ."  "  [']  '  ASCII  $,  LITERAL
[COMPILE]  COMPILE  ]  [  FIND  ?CAP  WORD  .S  DUMP  WORDS  VOCS
ORDER  ONLYFORTH  FORTH  ALSO  DEFINITIONS  CONTEXT  >LINK  >BODY
>NAME  LINK>  BODY>  NAME>  L>NAME  N>LINK  -->  THRU  LOAD  BLOCK
BUFFER  UPDATE  FLUSH  EMPTY-BUFFERS  SAVE-BUFFERS  R/W  COMMAND!
NUMBER?  CONVERT  .  .R  U.  U.R  D.  D.R  #>  SIGN  #S  #  HOLD
<#  HEX  DECIMAL  EXPECT  STOP?  CR  SPACES  SPACE  TYPE  SIO_INPUT
KEY  KEY?  SIO_OUTPUT  EMIT  EMIT?  INTERROR  NOTFOUND  NODEFER
?PAIRS  ?STACK  ?ERROR  ERROR  V,  VALLOT  H,  HALLOT  ,  C,  ALLOT
->VAR  DEPTH  WDP@  HERE  HEAP  PAD  TIB  FIRST  BLANK  ERASE  FILL
CMOVE>  CMOVE  -TRAILING  COUNT  SPR@  SPR!  GR!  GR@  OFF  ON  +!
*/  */MOD  MOD  /  /MOD  M/  M/MOD  *  M*  UM/MOD  UM*  DABS  D<
CASE?  UWITHIN  MAX  MIN  ABS  U<  >  =  <  0>  0=  0<  DNEGATE
D-  D+  2+  1+  1-  2-  NEGATE  -  +  FLIP  U2/  2/  2*  NOT  XOR
OR  AND  J  I  2DROP  2SWAP  2DUP  ROLL  PICK  -ROT  ROT  NIP  TUCK
OVER  ?DUP  PUSH  RDROP  R@  R>  >R  RP!  RP@  DROP  SWAP  DUP  DCLEAR
SP!  SP@  !  @  C!  C@  ?HEAD  LAST  CURRENT  ERRORHANDLER  OUTPUT
INPUT  DPL  HLD  BASE  #TIB  SPAN  R#  SCR  >IN  BLK  STATE  R0
S0  BL  TRUE  FALSE  NOOP  HDP  VLEN  VDP  DP  VOC-LINK  RAMORG
LEAVE  ?BRANCH  BRANCH  EXECUTE  EXIT  ok
<Ctrl S>  xlerb <Enter>     Ctrl S (nicht sichtbar) sperrt den Output.
                            Die Zeichen xlerb werden in den Buffer eingelesen, aber
                            nicht am Bildschirm angezeigt
<Ctrl Q> xlerb "XLERB" ?  ???   Ctrl Q (nicht sichtbar) gibt den Output wieder frei.
```