SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

CASCADING FUZZY MODULES WITH ST6 fUZZYTECH

By Lionel Picandet/Lim King Soon

INTRODUCTION

Note: This note requires knowledge of the ST6 fuzzyTECH fuzzy logic development tool and Software development tools and should be read in conjunction with the appropriate documentation.

It may seem as though it is is not possible to directly design several fuzzy modules with the ST6 fuzzyTECH graphic tool as from this point of view the ST6 fuzzyTECH tool can generate only one fuzzy module at a time. Therefore only one fuzzy project can be debugged at a time.

Nevertheless sometimes 2 fuzzy modules are needed to be included in the same application. It is possible to do this without modifying greatly the ST6 files generated by the ST6 fuzzyTECH tool. Keep in mind that these modules cannot work at the same time.

1 THE CONFIGURATION TECHNIQUES

We can consider two configurations as shown in the figures below:

Figure 1 : Modules with Independant Time Periods

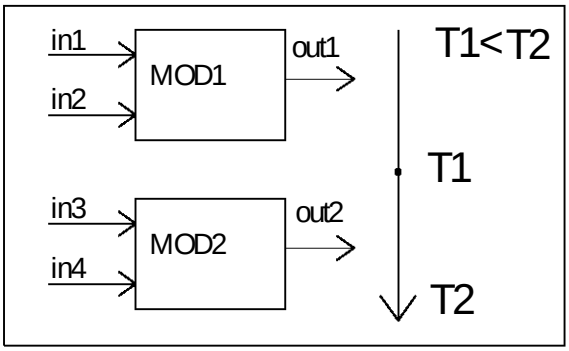
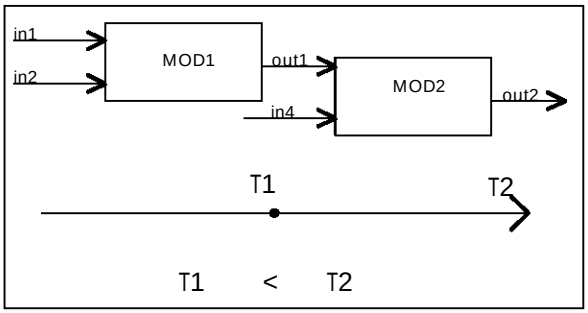


Figure 2 : Modules with Sequential Operation



In the first configuration the two fuzzy modules are independant. They can work sequentially or in different time periods.

In the second configuration the two fuzzy modules cannot be executed at the same time but they are however dependant. For example the first module can work at the beginning of the application, then stop and give information to the second module which starts and runs until the end of the process.

In this application note we describe 2 solutions to implement 2 fuzzy modules in the same application (as shown in the second configuration):

The first solution is easier to implement because it demands modification to only the USER.ST6 program. However its main drawback is that it includes the FUZZYLIB.ST6 library twice.

The second solution requires modification of the source files generated by the ST6 fuzzyTECH tool, but it has the advantage of including the FUZZYLIB.ST6 library only once.

1.1 First solution

ST6 procedure

This solution is described in the \FTEXPST6\SAMPLES\MYPROJ\CASCADE directory as a generic example.

To build the program you need:

- FUZZYLIB.ST6 fuzzy library
- MOD1.ST6 generated by fuzzyTECH MST6 (Source-File)
- MOD1.INC generated by fuzzyTECH MST6 (Include-File)
- MOD2.ST6 generated by fuzzyTECH MST6 (Source-File)
- MOD2.INC generated by fuzzyTECH MST6 (Include-File)
- USER1.ST6 generated by user
- USER2.ST6 generated by user

ST6 files description

The example described in the fuzzy package takes into account two independant fuzzy modules.

Several warnings must be noted:

- The program is mapped into page 0 and 1. Because there is not enough space to map windows in section 0, we have created a section 1 with just one instruction to be able to place data windows.

- As the two fuzzy modules do not work at the same time, fuzzy variables are located in the same place (”ftstart” is equal to the same value in the two modules). However the user has the option to avoid overlapping addresses of the fuzzy variables simply by changing their RAM address. If you define user variables relatively to ftend, take the ”ftend” symbol with the greater value to avoid overwriting the user variables with fuzzy variables.

- The fuzzy library FUZZYLIB.ST6 is included in every fuzzy module, so ROM space is wasted.

FUZZYLIB.ST6

The fuzzyTECH kernel for the ST6 microcontroller family. This kernel contains configurable modules for fuzzification, defuzzification and rule inference. The FUZZYLIB.ST6 file comes with the the fuzzyTECH ST6 Explorer Edition in the \LIB subdirectory.

MOD1.ST6

The output file of the ST6 Explorer Edition precompiler. Nothing can be modified in this file to fit the ST6 application.
MOD1.INC

Also generated by the ST6 Explorer Edition precompiler. MOD1.INC contains public variable definitions for the fuzzy logic system function. You have to include this file in USER1.ST6. Nothing can be modified in this file to fit the ST6 application.

MOD2.ST6

The output file of the ST6 Explorer Edition precompiler. Nothing can be modified in this file to fit the ST6 application.

MOD2.INC

Also generated by the ST6 Explorer Edition precompiler. MOD2.INC contains public variable definitions for the fuzzy logic system function. You have to include this file in USER2.ST6. Nothing can be modified in this file to fit the ST6 application.

USER1.ST6

Your main program, containing all interfaces and pre- and post-processing of input and output data of the first fuzzy module.

MOD1.ST6.

This file included in the \FTEXPST6\SAMPLES\MYPROJ subdirectory has been copied to the \FTEXPST6\SAMPLES\MYPROJ\CASCADE subdirectory and then modified to fit the program:

```
;---------------------------------------------------------------------
;---------------------- File: USER1.ST6 ------------------------------
;---------------------------------------------------------------------

      .VERS"ST62xx"
      .ROMSIZE 2
      .INPUT     "mod1.inc"

      .EXTERN initmod1, mod1
      .EXTERN    cascade


            .PP_ON

      .SECTION        0

;---------------------- define used registers --------------------

a     .DEF  0FFH

;------------------ entry point in the user program --------------

main:

;--------------------- exit from the reset status ----------------

       reti
```

```
;---------------------- user initializations --------------------

       call    initmod1

;---------------------- loop on fuzzy routines ------------------

loop1:

     ldi   lv0_in1, 020h         ; give inputs to Fuzzy (examples)
     ldi   lv1_in2, 040h
     ldi   lv2_in3, 080h
     call  mod1             ; call fuzzy routines
     ld    a, invalidflags
     jrnz  no_fire1           ; test if a rule fire
     jp    loop1

no_fire1:

     ld    a, lv3_out1           ; crisp result is stored in lv3_out1
     cpi   a, 0ffh
     jrz   fuzz1_achieved
     jp    loop1

fuzz1_achieved

     jp    cascade               ; jump to the second fuzzy module

;-------------------------- END MAIN -----------------------------

     .SECTION   1

     nop

;------------------- No interrupt handle routines ---------------

     .SECTION   32

adc        nop
           reti

timer      nop
           reti

portbc          nop
           reti

porta      nop
           reti

     .block     4

nmi        nop
           reti

reset      jp    main
```

```
;---------------------------------------------------------------------
```

USER2.ST6

The file containing all interfaces and pre- and post-processing of input and output data of the second fuzzy module MOD2.ST6.

This file included in the \FTEXPST6\SAMPLES\MYPROJ subdirectory has been copied to the \FTEXPST6\SAMPLES\MYPROJ\CASCADE subdirectory then modified to fit the program.

```
;---------------------------------------------------------------------
;---------------------- File: USER2.ST6 ------------------------------
;---------------------------------------------------------------------

      .VERS"ST62xx"
      .ROMSIZE 2
      .INPUT     "mod2.inc"

      .EXTERN    initmod2, mod2
      .GLOBAL    cascade

      .PP_ON

      .SECTION   0

;---------------------- define used registers --------------------

a     .DEF  0FFH

; we have to define the output variable of the first module again
; in this module because we use it

lv3_out1   .DEF  ftstart + 003H       ; crisp i/o variable

;---------------------- user initializations ---------------------

cascade:
      call  initmod2

;---------------------- loop on fuzzy routines --------------------

loop2:

      ld    a,lv3_out1 ; because the two modules are located in
      ld    lv3_in7, a ; the same place we have to deal with this
                       ; output variable first

      ldi   lv0_in4, 020h   ; give inputs to Fuzzy (examples)
      ldi   lv1_in5, 040h
      ldi   lv2_in6, 060h
      call  mod2       ; call fuzzy routines

      ld    a, invalidflags

      jrnz  no_fire2   ; test if a rule fire
      jp    loop2
```

```
no_fire2

     ld   a, lv4_out2     ; crisp result is stored in lv4_out2
     jp   loop2      ; jump direct (closed loop)
     ret

;-------------------------------------------------------------
```

CASCADE1.BAT

This is an example of the required build procedure.

1.2 Second solution

The second solution is more restrictive than the first solution.

ST6 procedure

This solution is described in the \FTEXPST6\SAMPLES\MYPROJ\CASCADE directory as a generic example.

To build the program you need:

- FUZZYLIB.ST6 fuzzy library
- MOD3.ST6 generated by fuzzyTECH MST6 (Source-File)
- MOD3.INC generated by fuzzyTECH MST6 (Include-File)
- MOD2.ST6 generated by fuzzyTECH MST6 (Source-File)
- MOD2.INC generated by fuzzyTECH MST6 (Include-File)
- USER3.ST6 generated by user

The fuzzy linguistic variables for both modules 2 and 3 are located at the same RAM space by the ST6 fuzzyTECH tool. In order to use this solution with the overlapping RAM space, the user has to make one modification. That is, if the number of linguistic input variables are different for
both modules, the user has to relocate the RAM register ”invalidflags” to the end of the register definition.

This modification will not be necessary if the number of input variables used for both modules are the same. However the user has the option to avoid overlapping addresses simply by changing the RAM address of the fuzzy linguistic variables from second module. In this case the contents of the linguistic variables of the first module will not be overwritten while executing the second.

Due to the duplication of the fuzzy library in every module the first solution wastes a lot of ROM space. To avoid the duplication of the fuzzy library the source files generated by the ST6 fuzzyTECH tool need to be modified. The approach is to concentrate all the calls to the fuzzy library routines within one module.

The steps are as follows:

- append the module with the smallest size of data RAM to the greatest one (to do this look at every .INC files and compare their FTEND symbols).
- append the linguistic variables defined in the .INC file of the second module (the module with the smallest size of data RAM) to the .INC file of the first one.
- rename the data RAM registers fuzout in either module using a different name as before.
- delete the duplicated register definition in the module that has the smallest amount of data RAM because these registers have been defined earlier.
- rename the lookup tables (rt0, rt1 ,.. , xcom , tpts1..) from the second module using different names as before.
- delete the include file of the library in the second module
- verify call directives to fuzzy library for the second module; if call directives are equal, delete the call directives to this module; if call directives are different, add extra call directives to the first module and delete the call directives to this module and rename the label parameters used in the called routine.
- change the RAM registers ”fuzout” in the modified module using the modified name whenever it has been called
- change the calling of the tables in the second module according to the modified names made previously.
- delete the initialization routine of the second module because it is exactly the same as the first module.

ST6 files description

The example described in the fuzzy package takes into account two independant fuzzy modules. To make the difference between the first solution and the second one we have renamed the generated source file MOD1.ST6 to MOD3.ST6.

FUZZYLIB.ST6

The fuzzyTECH kernel for the ST6 microcontroller family. This kernel contains configurable modules for fuzzification, defuzzification and rule inference. The FUZZYLIB.ST6 file comes with the fuzzyTECH ST6 Explorer Edition in the \LIB subdirectory.
MOD3.ST6

The output file of the ST6 Explorer Edition precompiler. Call directives to fuzzy library kernel of the second module must eventually be added to this one. This is the file with the greatest amount of data RAM.

```
;--------------------------------------------------------------------
```

```
;-------------------------- project: MOD3 ---------------------------
;--------------------------------------------------------------------

                .PP_ON
                .W_ON
                .NOTRANSMIT

a           .DEF  0FFH
x           .DEF  080H
y           .DEF  081H
v           .DEF  082H
w           .DEF  083H

drwr        .DEF  0C9H

            .INPUT     "MOD3.INC"
            .TRANSMIT

fvs         .DEF  ftstart + 005H        ;(f)uz(v)als (s)tart
fuzvals           .DEF  ftstart + 005H
fuzout            .DEF  ftstart + 014H
fuzout1           .DEF  ftstart + (value depends on second module)

;------------ reserved locations for ST6 fuzzyTECH kernel ---------

var1        .DEF  ftstart + 019H
var2        .DEF  ftstart + 01AH
var3        .DEF  ftstart + 01BH
var4        .DEF  ftstart + 01CH
var5        .DEF  ftstart + 01DH
var6        .DEF  ftstart + 01EH

;-------------------------- name mapping ----------------------------

crisp       .DEF  var1
rulecnt           .DEF  var1
otcnt       .DEF  var1
x1          .DEF  var2
x3          .DEF  var2
incnt       .DEF  var2
firecnt           .DEF  var2
aslope            .DEF  var3
curmin            .DEF  var3
curmax            .DEF  var3
fireval           .DEF  var3
icnt        .DEF  var4
itcnt       .DEF  var4
numl        .DEF  var4
outcnt            .DEF  var4
numh        .DEF  var5
numext            .DEF  var6

;---------------------- term definition -------------------------

            ; nothing is changed in this part
```

```
;------------------- defines for included kernel -----------------

          ; eventually add extra call directives for the second
module

_fmax_          .EQU 0
_max_      .EQU 0
_fmin_          .EQU 0
_min_      .EQU 1
_com_      .EQU 0
_mom_      .EQU 0


          .SECTION    0

          .INPUT     ”FUZZYLIB.ST6”  ; fuzzyTECH kernel for ST6

;---------------------- fuzzy controller function -----------------

          ; nothing is changed in this part

          .GLOBAL    mod1
          .GLOBAL    initmod1

;-------------------------------------------------------------------
;------------------------ project: MOD2 --------------------------
;-------------------------------------------------------------------

          ; delete register definition

;------------------------ term definition -------------------------

          ; rename the window symbols



          .WINDOW
cs_tpts1

          .BYTE 00000H, 00000H, 0003FH, 00004H
          .BYTE 0007EH, 00004H, 000FFH, 00000H

          .WINDOWEND

;-------------------------- rule table ------------------------

          .WINDOW

cs_rt0:

               .WINDOWEND

          .WINDOW

cs_rt13:

          .WINDOWEND
```

```
;------------------- xcom table (defuzzification) ----------------

           .WINDOW

cs_xcom:

           .WINDOWEND


;------------------- fuzzy controller function --------------------

mod2:

;------------------------- fuzzification --------------------------

           ; nothing is changed in this part

;--------------------------- inference ----------------------------

           ; rename window symbols

           ldi   drwr, cs_rt0.w
           ldi   y, cs_rt0.d
           call  Min        ; min aggregation

           ldi   drwr, cs_rt13.w
           ldi   y, cs_rt13.d

           call  Min        ; min aggregation

;----------------------- defuzzification --------------------------

           clr   invalidflags
           ldi   drwr, cs_xcom.w
           ldi   y, cs_xcom.d
           ldi   x, fuzout1 ; rename ”fuzout” in ”fuzout1”
           ldi   lv4_out2, 080H
           ldi   otcnt, 03H
           call  com
           ld    lv4_out2, a
           ret              ; end of fuzzy controller

           .GLOBAL    mod2

;---------------------------------------------------------------
```

MOD3.INC

Also generated by the ST6 Explorer Edition precompiler. MOD3.INC contains public variables definitions for the fuzzy logic system function. You have to include this file in USER3.ST6 file. You have to append the linguistic variables definition of the second module to this file.

```
;---------------------------------------------------------------
```

```
;---------- MOD3.INC allows general access to crisp values ----
;--------------------------------------------------------------

          .ifc  ndf ftstart

ftstart         .EQU  0084H      ; 1st free RAM byte

          .endc

;------------- input/output interface of controller ------------

lv0_in1         .DEF  ftstart + 000H  ; crisp i/o variable
lv1_in2         .DEF  ftstart + 001H  ; crisp i/o variable
lv2_in3         .DEF  ftstart + 002H  ; crisp i/o variable
lv3_out1        .DEF  ftstart + 003H  ; crisp i/o variable
invalidflags    .DEF  ftstart + 004H

;------------- auxilary variables for multiplication -----------

m_res           .DEF  ftstart + 01FH
l_res           .DEF  ftstart + 020H
counter         .DEF  ftstart + 021H
op1             .DEF  ftstart + 021H
op2             .DEF  ftstart + 022H
m_op1           .DEF  ftstart + 023H
l_op1           .DEF  ftstart + 024H
save_a          .DEF  ftstart + 025H
remaind         .DEF  ftstart + 026H


;----------------- 1st free location in data space ---------------

ftend      .EQU  ftstart + 027H

     ; add variables definition of the second module



;-------------- input/output interface of controller ---------------

lv0_in4         .DEF  ftend + 000H    ; crisp i/o variable
lv1_in5         .DEF  ftend + 001H    ; crisp i/o variable
lv2_in6         .DEF  ftend + 002H    ; crisp i/o variable
lv3_in7         .DEF  ftend + 003H    ; crisp i/o variable
lv4_out2        .DEF  ftend + 004H    ; crisp i/o variable

;--------------------------------------------------------------
```

MOD2.ST6

The output file of the ST6 Explorer Edition precompiler. You have to append this file to the MOD3.ST6 file.

MOD2.INC

Also generated by the ST6 Explorer Edition precompiler. MOD2.INC contains public variable definitions for the fuzzy logic system function. You have to put the linguistic variables definition of this file into the MOD3.INC file.

USER3.ST6

Your main program, containing all interfaces and pre- and post-processing of input and output data of the two fuzzy modules.

This file, included in the \FTEXPST6\SAMPLES\MYPROJ subdirectory, is copied to the \FTEXPST6\SAMPLES\MYPROJ\CASCADE subdirectory and then modified to fit the program.

```
;---------------------------------------------------------------------
;---------------------- File: USER3.ST6 ------------------------------
;---------------------------------------------------------------------

      .VERS"ST62xx"
      .ROMSIZE   2
      .INPUT     "mod3.inc"

      .EXTERN    initmod1, mod1 , mod2



      .PP_ON

      .SECTION   0

;----------------------- define used registers --------------------

a     .DEF  0FFH

;------------------ entry point in the user program --------------

main:

;-------------------- exit from the reset status -----------------

      reti


;----------------------- user initializations --------------------

      call  initmod1

;---------------------- loop on fuzzy routines -------------------

loop1:
      ldi   lv0_in1, 020h         ; give inputs to Fuzzy (examples)
      ldi   lv1_in2, 040h
      ldi   lv2_in3, 080h
      call  mod1             ; call fuzzy routines

      ld    a, invalidflags
```

```
      jrnz  no_fire1        ; test if a rule fire
      jp    loop1

no_fire1
      ld    a, lv3_out1          ; crisp result stored in lv3_out1
      cpi   a, 0ffh
      jrz   fuzz1_achieved

      jp    loop1

fuzz1_achieved

      call  initmod1

;---------------------- loop on fuzzy routines -------------------

loop2:
      ldi   lv0_in4, 020h        ; give inputs to Fuzzy (examples)
      ldi   lv1_in5, 040h
      ldi   lv2_in6, 060h
      ld    a, lv3_out1
      ld    lv3_in7, a
      call  mod2            ; call fuzzy routines

      ld    a, invalidflags
      jrnz  no_fire2        ; test if a rule fire

      jp    loop2

no_fire2:

      ld    a, lv4_out2          ; crisp result stored in lv4_out2
      jp    loop2           ; jump direct (closed loop)

;--------------------------- END MAIN -----------------------------


; tips to map windows as there is not enough space to map them in
section 0

      .SECTION   1
      nop

;------------------- No interrupt handle routines ---------------

      .SECTION   32

adc   nop
      reti

timer nop
      reti

portbc     nop
      reti

porta nop
```

```
      reti

      .block     4

nmi   nop
      reti

reset jp    main

;---------------------------------------------------------------------
```

CASCADE2.BAT

This is an example of the required build procedure.

2 SUMMARY

This application note demonstrates practical techniques for using two fuzzy logic modules produced by the ST6 fuzzyTECH EXPLORER EDITION fuzzy logic development tool into one application. Code and examples are supplied for the ST6 microcontroller.