

```

;*****
; Program:          RCMICRO.MDS
; Description:      Model programs for development
; Function:         Model type processing source code
; Author:          John F. Fitter B.E.
; Language:        MDL V1.0
; Platform:        RCMicro V1.0
;
; Revisions:       00 15jan97 Original
;                  01 13jun97 Re-written for new language syntax
;                  02 14jun97 Added new models
;                  03 21jun96 New language elements added
;                  04 04dec97 Interpreter working - program refinements
;                  05 04dec97 Program changed to relect new syntax
;
; Copyright c 1999 Eagle Air Australia Pty. Ltd.
;
;*****

;*****
; F3A Aerobatic contest aircraft
;
; Implements:      Dual rates on elevator, aileron and rudder
;                  Idle-up enabled by switch
;                  Mixture control from traditional throttle trim control
;                  Left, right and invert snap roll switches
;                  Mixing of throttle into mixture
;                  Retracting undercarriage switch
;*****

#descr          "F3A Aero"

; Analog input definitions

#define elev_in 0 ; elevator input
#define ail_in 1 ; aileron input
#define rud_in 2 ; rudder input
#define thr_in 3 ; throttle input
#define aux1_in 5 ; aux1 input
#define aux2_in 6 ; aux2 input
#define elev_tr 7 ; elevator trim
#define ail_tr 8 ; aileron trim
#define rud_tr 9 ; rudder trim
#define mixt_in 4 ; mixture input

; Output channel definitions

#define elev_out 7 ; elevator output
#define ail_out 6 ; aileron output
#define rud_out 5 ; rudder output
#define thr_out 4 ; throttle output
#define retr_out 3 ; retract output
#define aux1_out 2 ; aux1 output
#define aux2_out 1 ; aux2 output
#define mixt_out 0 ; mixture output

; Switch definitions

#define elev_dr 7 ; elevator dual rate
#define ail_dr 6 ; aileron dual rate
#define rud_dr 5 ; rudder dual rate
#define thr_iu 0 ; throttle idle-up
#define snapl 4 ; left snap roll
#define snapr 3 ; right snap roll
#define snapi 2 ; invert snap roll
#define retract 8 ; retract

; Parameter definitions

#define sn_elev 0 ; snap roll elevator amount
#define sn_ail 1 ; snap roll aileron amount
#define sn_rud 2 ; snap roll rudder amount
#define idle_amt 3 ; idle-up amount
#define elev_rate 4 ; elevator dual rate amount
#define ail_rate 5 ; aileron dual rate amount
#define rud_rate 6 ; rudder dual rate amount
#define mix_mix 7 ; throttle - mixture mix amount

```

## ; User interface definitions

```

#assign    CHA0      "Elev"
#assign    CHA1      "Ail"
#assign    CHA2      "Rudd"
#assign    CHA3      "Thro"
#assign    CHA4      "Retr"
#assign    CHA5      "Aux1"
#assign    CHA6      "Aux2"
#assign    CHA7      "Mixt"

#assign    PAR0      "SrEle"
#assign    PAR1      "SrAil"
#assign    PAR2      "SrRud"
#assign    PAR3      "IdlUp"
#assign    PAR4      "DrEle"
#assign    PAR5      "DrAil"
#assign    PAR6      "DrRud"
#assign    PAR7      "MTMix"

THR:      IN        thr_in      ; push throttle into X
          SSC        thr_iu      ; is idleup off ? (contacts closed is off)
          JMP        THR0        ; yes, then skip all this
          SETX       idle_amt    ; push idleup amount into X, throttle into Y
          SGE        ; is the throttle less than the idleup amount
          POP        ; no, then pop throttle, else use idleup amount

THR0:     OUT        thr_out     ; send throttle to output

MIXT:     IN        mixt_in     ; push mixture into X
          PUSH       ; triple range of mixture
          DBX
          ADD
          EXCH      ; throttle in X, mixture in Y
          MIX        mix_mix     ; mix throttle into mixture
          OUT        mixt_out    ; send mixture to output

AUX1:     IN        aux1_in     ; get auxilliary 1
          DBX        ; double range
          OUT        aux1_out    ; send to output

AUX2:     IN        aux2_in     ; get auxilliary 2
          DBX        ; double range
          OUT        aux2_out    ; send to output

RETR:     SETS      ; preload stack with maximum deflections
          SSS        retract    ; is retract on ?
          NEGX      ; no, then negate retract
          OUT        retr_out    ; send retract to output

SNAP:     SSC        snapl      ; snap roll determination
          JMP        SNAP0      ; is left snap on ?
          SSS        snapr      ; no, then is snap right on ?
          GTO        ELEV       ; no snaps, then jump to elevator processing
SNAP0:    SCL        sn_elev    ; snap left or right must be on, get elevator amount
          RLU        ; roll amount of elevator up stack
          SCL        sn_ail     ; get amount of aileron
          RLU        ; roll amount of aileron up stack
          SCL        sn_rud     ; and get amount of rudder
          SSS        snapl      ; is snap left on ?
          JMP        SNAP1      ; no, then skip negations
          NEGX      ; snap left is on then negate aileron & rudder
          NEGY

SNAP1:    SSS        snapi      ; is snap invert on ?
          JMP        SNAP2      ; no, then skip negations
          NEGX      ; yes, then negate elevator, aileron, & rudder
          NEGY
          NEGZ

SNAP2:    OUT        rud_out    ; send rudder to output
          POP
          OUT        ail_out    ; send aileron to output
          POP
          OUT        elev_out   ; send elevator to output
          RET        ; done, return from processing

ELEV:     IN        elev_in     ; push elevator into X
          SSC        elev_dr    ; is dual rate on ?

```

```

        SCL      elev_rate    ; yes, multiply elevator by it's rate
        TRIM     elev_tr     ; trim the elevator
        OUT      elev_out    ; send elevator to output

AIL:    IN      ail_in      ; push aileron into X
        SSC      ail_dr     ; is dual rate on ?
        SCL      ail_rate    ; yes, multiply aileron by it's rate
        TRIM     ail_tr     ; trim the aileron
        OUT      ail_out    ; send aileron to output

RUD:    IN      rud_in     ; push rudder into X
        SSC      rud_dr     ; is dual rate on ?
        SCL      rud_rate    ; yes, multiply rudder by it's rate
        TRIM     rud_tr     ; trim the rudder
        OUT      rud_out    ; send rudder to output

        RET      ; done, return from processing

#end

;*****
; F3B Glider contest aircraft *
; *
; Implements:  Dual rates on elevator, aileron and rudder *
; Spoiler on auxillary channel *
; Towhook release on switched channel *
; Flaperon mixing (on always) *
; Switched flap mixed into elevator (elevator/flap trim) *
; Switched spoiler mixed into elevator (elevator/spoiler trim) *
; Switched aileron mixed into rudder (rudder/aileron coupling) *
; Servo speed control on flap and spoiler functions *
; Switched speed flight setting - sets elevator trim and flap position *
;*****

#descr      "F3B Glider"

; Analog input definitions

#define     elev_in      0      ; elevator input
#define     ail_in      1      ; aileron input
#define     rud_in      2      ; rudder input
#define     flap_in     3      ; flap input
#define     spoil_in    6      ; spoiler input
#define     aux1_in     5      ; auxillary input
#define     elev_tr     7      ; elevator trim
#define     ail_tr      8      ; aileron trim
#define     rud_tr      9      ; rudder trim
#define     flap_tr     4      ; flap trim

; Output channel definitions

#define     elev_out    7      ; elevator output
#define     ail_out_1  6      ; LH aileron output
#define     rud_out    5      ; rudder output
#define     ail_out_2  4      ; RH aileron output
#define     spoil_out  3      ; spoiler output
#define     hook_out   2      ; towhook release output
#define     aux1_out   1      ; auxillary 1 output
#define     aux2_out   0      ; auxillary 2 output

; Switch definitions

#define     elev_dr     0      ; elevator dual rate
#define     ail_dr     1      ; aileron dual rate
#define     rud_dr     2      ; rudder dual rate
#define     mix_fe     3      ; mix flap into elevator
#define     mix_se     4      ; mix spoiler into elevator
#define     mix_ar     5      ; mix aileron into rudder
#define     speed      6      ; set trims for speed
#define     hook_rel   7      ; release towhook

; Parameter definitions

#define     fa_amt     0      ; flap - aileron mix amount
#define     fe_amt     1      ; flap - elevator mix amount
#define     se_amt     2      ; spoiler - elevator mix amount
#define     ar_amt     3      ; aileron - rudder mix amount
#define     elev_rate  4      ; elevator dual rate amount

```

```

#define    ail_rate    5            ; aileron dual rate amount
#define    rud_rate    6            ; rudder dual rate amount
#define    sp_ele      7            ; elevator trim offset for speed
#define    sp_flap     8            ; flap trim setting for speed
#define    flap_sp     9            ; speed of operation of flap
#define    spoil_sp    10           ; speed of operation of spoiler

; Register definitions

#define    flap_reg    0            ; flap temporary storage
#define    spoil_reg   1            ; spoiler temporary storage

; User interface definitions

#assign    CHA0        "Elev"
#assign    CHA1        "Ail"
#assign    CHA2        "Rudd"
#assign    CHA3        "Flap"
#assign    CHA4        "Spoil"
#assign    CHA5        "Hook"
#assign    CHA6        "Aux1"
#assign    CHA7        "Aux2"

#assign    PAR0        "FAMix"
#assign    PAR1        "FEMix"
#assign    PAR2        "SEMix"
#assign    PAR3        "ARMix"
#assign    PAR4        "DrEle"
#assign    PAR5        "DrAil"
#assign    PAR6        "DrRud"
#assign    PAR7        "SpEle"
#assign    PAR8        "SpFla"
#assign    PAR9        "FlSpd"
#assign    PAR10       "SpSpd"

HOOK:      LMX                ; set towhook release
           SSS                hook_rel ; is release switch on
           NEGX               ; no, then negate X
           OUT                hook_out ; send to release

FLAP:      IN                flap_in   ; push flap into X
           SSC                speed    ; is speed switch on
           SETX               sp_flap ; yes, then push speed flap trim into X
FLAP1:     RCL                flap_reg ; get previous flap position into X
           SADJ               flap_sp  ; speed adjust flap
           TRIM               flap_tr  ; add flap trim
           STO                flap_reg ; store flap in register

SPOIL:     IN                spoil_in  ; push spoiler into X
           DBX                ; double range
           RCL                spoil_reg ; get previous position of spoiler into X
           SADJ               spoil_sp ; speed adjust spoiler
           STO                spoil_reg ; store in spoiler register
           OUT                spoil_out ; send to spoiler

RUD:       IN                ail_in    ; push aileron into X
           PUSH               ; and push a copy into Y
           IN                rud_in    ; push rudder into X
           SSC                rud_dr   ; is dual rate on
           SCL                rud_rate ; yes, scale the rudder
           TRIM               rud_tr   ; trim the rudder
           SSC                mix_ar   ; is aileron-rudder mixing on ?
           MIX                ar_amt   ; yes, then mix aileron into rudder
           OUT                rud_out  ; send result to rudder
           POP                ; get the aileron amount into X
           POP

AIL:       SSC                ail_dr   ; is dual rate on ?
           SCL                ail_rate ; yes, scale the aileron
           TRIM               ail_tr   ; trim the aileron
           RCL                flap_reg ; push flap into X
           MIX                fa_amt   ; mix flap with aileron
           OUT                ail_out_1 ; send to flaperon_1
           POP                ; get other flaperon into X
           OUT                ail_out_2 ; send to flaperon_2

ELEV:     IN                elev_in   ; push elevator into X

```

```

        SSC          elev_dr      ; is dual rate on ?
        SCL          elev_rate    ; yes, scale the elevator
        TRIM         elev_tr      ; trim the elevator
        SSS          mix_fe       ; is elevator - flap mixing on
        JMP          ELEV1        ; no, then jump
        RCL          flap_reg     ; push flap into X
        MIX          fe_amt       ; mix flap into elevator
ELEV1:   SSS          mix_se       ; is spoiler - elevator mixing on
        JMP          ELEV2        ; no, then jump
        RCL          spoil_reg    ; push spoiler into X
        MIX          se_amt       ; mix spoiler into elevator
ELEV2:   SSS          speed       ; is speed switch on
        JMP          ELEV3        ; no, then done
        LMX          LMX         ; yes, compute speed trim offset
        SCL          sp_ele
ELEV3:   OUT          elev_out    ; send to elevator

AUX1:   IN           aux1_in     ; push aux1 into X
        DBX          DBX         ; double range
        OUT          aux1_out    ; send to aux1

        RET              ; return from processing
#end

;*****
; Sport aircraft for testing - implements basic operation - not a real aeroplane *
;*****

#descr      "Sport-Test1"

; Analog input definitions

#define     elev_in      0          ; elevator input
#define     ail_in      1          ; aileron input
#define     rud_in      2          ; rudder input
#define     thr_in      3          ; throttle input
#define     aux1_in     4          ; aux1 input
#define     aux2_in     5          ; aux2 input
#define     elev_tr     6          ; elevator trim
#define     ail_tr      7          ; aileron trim
#define     rud_tr      8          ; rudder trim
#define     thr_tr      9          ; throttle trim

; Output channel definitions

#define     elev_out     0          ; elevator output
#define     ail_out_1   1          ; aileron output 1
#define     rud_out     2          ; rudder output
#define     thr_out     3          ; throttle output
#define     retr_out    4          ; retract output
#define     aux1_out    5          ; aux1 output
#define     aux2_out    6          ; aux2 output
#define     ail_out_2   7          ; aileron output 2

; Switch definitions

#define     elev_dr     0          ; elevator dual rate
#define     ail_dr     1          ; aileron dual rate
#define     rud_dr     2          ; rudder dual rate
#define     retract    3          ; retract

; Parameter definitions

#define     elev_rate   0          ; elevator dual rate amount
#define     ail_rate    1          ; aileron dual rate amount
#define     rud_rate    2          ; rudder dual rate amount

; User interface definitions

#assign    CHA0        "Elev"
#assign    CHA1        "Ail"
#assign    CHA2        "Rudd"
#assign    CHA3        "Thro"
#assign    CHA4        "Retr"
#assign    CHA5        "Aux1"
#assign    CHA6        "Aux2"

```

```
#assign PAR0 "DrEle"
#assign PAR1 "DrAil"
#assign PAR2 "DrRud"

ELEV: IN elev_in ; push elevator into X
      SSC elev_dr ; is dual rate on ?
      SCL elev_rate ; yes, multiply elevator by it's rate
      TRIM elev_tr ; trim the elevator
ELEV1: OUT elev_out ; send elevator to output

AIL: IN ail_in ; push aileron into X
      SSC ail_dr ; is dual rate on ?
      SCL ail_rate ; yes, multiply aileron by it's rate
      TRIM ail_tr ; trim the aileron
AIL1: OUT ail_out_1 ; send aileron 1 to output
      NEGX
      OUT ail_out_2 ; send aileron 2 to output

RUD: IN rud_in ; push rudder into X
      SSC rud_dr ; is dual rate on ?
      SCL rud_rate ; yes, multiply rudder by it's rate
      TRIM rud_tr ; trim the rudder
RUD1: OUT rud_out ; send rudder to output

THR: IN thr_in ; push throttle into X
      TRIM thr_tr ; trim the throttle
      OUT thr_out ; send throttle to output

AUX1: IN aux1_in ; get auxilliary 1
      DBX ; double it
      OUT aux1_out ; send to output

AUX2: IN aux2_in ; get auxilliary 2
      DBX ; double it
      OUT aux2_out ; send to output

RETR: SETS ; preload stack with maximum deflections
      SSS retract ; is retract on ?
      NEGX ; no, then negate retract
      OUT retr_out ; send retract to output

      RET ; return from processing

#end
```