```
idle: if (data_ready=0 and load_coefficient=0) goto idle
    if (load_coefficient=1) goto loadF0
;*************************************************
store: if (data_ready=0) goto eidle
    reg[5] = data                ; Store data in a register
    err = 0                      ; reset error

zero:
    reg[0] = 0 ; zero out accumulator

sort1:
    reg[1] = reg[2]              ; Reorder registers

sort2:
    reg[2] = reg[3]              ; Reorder registers

sort3:
    reg[3] = reg[4]              ; Reorder registers

sort4:
    reg[4] = reg[5]              ; Reorder registers

mul1:
    reg[10] = reg[1] * reg[6]          ; sample2* F2

add1:
    reg[0] = reg[0] + reg[10] ; add Large pos. coefficient
    if (V) goto eidle         ; On overflow, err condition

mul2:
    reg[10] = reg[2] * reg[7]  ; sample1* F1

sub1:
    reg[0] = reg[0] - reg[10]   ; sub Large neg. coefficient
    if (V) goto eidle         ; On overflow, err condition
    else goto idle

mul3:
    reg[10] = reg[3] * reg[8]          ; sample2* F2

add2:
    reg[0] = reg[0] + reg[10] ; add Large pos. coefficient
```

```
    if (V) goto eidle         ; On overflow, err condition

mul4:
    reg[10] = reg[4] * reg[9]  ; sample1* F1

sub2:
    reg[0] = reg[0] - reg[10]   ; sub Large neg. coefficient
    if (V) goto eidle         ; On overflow, err condition
    else goto idle
;*****************************************************
eidle:
    err = 1
    if (data_ready=1) goto store  ; wait until data_ready=1
    if (data_ready=0) goto eidle
;*****************************************************
loadF0:
    reg[9] = coef_data

waitF1:
    if (load_coefficient=0) goto waitF1

loadF1:
    data[8] = coeff_data

waitF2:
    if (load_coefficient=1) goto waitF2

loadF2:
    data[7] = coeff_data

waitF3:
    if (load_coefficent=1) goto waitF3

loadF3:
    data[6] = coeff_data
    goto idle
```