Ag102 Under-voltage, Change-over and Discharge Protection Circuit

Silvertel

This document describes how to use the Ag102 with a simple µ-controller. In this application, you can monitor the input supply voltage to change over to the battery and disconnect the load from the battery, when it is running low.

The Ag102 has a wide input supply operating range of 9V to 36V. But if the Ag102 supply drops <9V it can generate an over-current error. This requires the input voltage to be power cycled (removed and reapplied) before it will resume normal operation.

The circuit shown in Figure 1 uses a simple µ-controller to monitor the supply rail. In this example we have used a Microchip 16F676, but this can be changed to a suitable alternative.

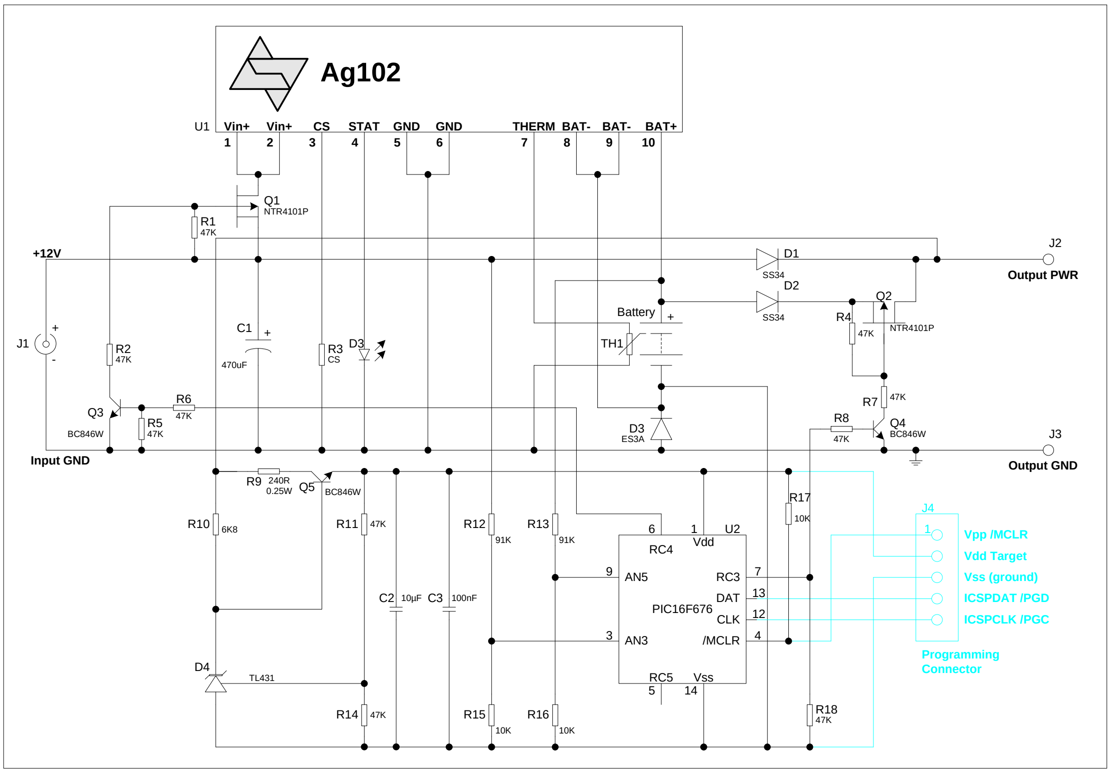

Figure 1: Example Circuit

When the input supply is present at J1, this is passed through D1 to the Output PWR pin. The µ-controller (U2) is powered from the Output PWR pin, via a simple linear regulator circuit.

However the Ag102 (U1) and the battery must both be connected to complete the return path. The Ag102 monitors the charge current across an internal sense resistor connected between the GND and BAT- pins, therefore these pins cannot be connected together. When powered from the input supply, the return path has to go through the Ag102, but this path will only be enabled when the battery is connected. So if the Ag102 and battery are not fitted the µ-controller will not be powered.

The linear regulator circuit uses a TL431 to generate the 5V supply. By having a reasonably accurate 5V rail this can then be used by the A/D converter.

R12 and R15 are used to divide the input supply voltage down to within the input range of µ-controller's A/D.

When the input voltage is ≥11V the output pin "RC4" is set to logic 1, this turns Q3 and Q1 ON, connecting the supply to the Ag102 input.

When the input voltage is <10V the output pin "RC4" is set to logic 0, this turns Q3 and Q1 OFF, disconnecting the supply to the Ag102 input. By turning the Ag102 OFF before the input drops <9V, this prevents the Ag102's dc/dc converter from going into an over-current fault condition.

When the input voltage is <10V (and "RC4" resets to logic 0), the µ-controller then quickly checks that the battery voltage is ≥10V (using the potential divider R13 and R16). If it is the case "RC3" is set to logic 1, turning Q4 and Q2 ON, connecting the battery to the Output PWR pin. This is done quickly and C2 maintains the 5V supply to the µ-controller during the transition.

The µ-controller is now being powered by the battery and continues to monitor the input supply and the battery voltage. If the input power is restored to ≥11V, "RC2" will be reset to logic 0 to disconnect the battery and "RC4" set to logic 1 to reconnect power to the Ag102.

If the input power remains OFF and the battery voltage drops <10V, the µ-controller will reset "RC3" to logic 0 and action a 2 second delay loop. During this 2 second loop C2 will discharge and the µ-controller will turn itself OFF.

When the µ-controller is OFF, the only current drawn from the battery will be ~1mA through R12 and R15. The value of these may be increase, depending on the input impedance if the µ-controller's A/D input.

An example (asm) code is shown in Appendix A.

The Ag102 is not designed to be used with a solar panel; this is primarily due to the Ag102 going into an over-current error mode when the supply drops below 9V. Once in an over-current error mode the Ag102 needs to be power cycled before it can be returned to normal operation. But this application overcomes this problem by disconnecting the Ag102's when the input is <10V, preventing it from going into and staying in this error mode.

This application note is not a perfect solution for solar panels, because the Ag102 goes into bulk mode each time it starts-up. It then quickly goes through the charge profile until it reaches the point where the charge was terminated. If the solar panel goes into partial shade, it can end up going through this process many times.

Appendix A – Example (asm) Code

```
list      p=16f676                                    ; list directive to define processor
#include <p16F676.inc>                                ; processor specific variable definitions

errorlevel  -302                                      ; suppress message 302 from list file

__CONFIG     _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_ON & _WDT_OFF & _PWRTE_ON &
_INTRC_OSC_NOCLKOUT

; '__CONFIG' directive is used to embed configuration word within .asm file.
; The lables following the directive are located in the respective .inc file.
; See data sheet for additional information on configuration word settings.

RAMTRIS                     RES   1             ; this is a baseline part so have to create
                                                ; own tris register in RAM to keep track of
                                                ; input and output pins (very important!)

;***********************************************************************
;***** VARIABLE DEFINITIONS
;***********************************************************************

Battery            equu 0x03                          ; battery backup switch
Power              equ 0x04                           ; Ag102 power on switch
TestPin            equ 0x05                           ; test pin


         cblock 20h                                   ; list of variables used in the program

                   Delay:3                            ; three delay loop bytes
                   Counter                            ; loop counter
                   LowCount                           ; low loop counter
                   AD:2                               ; A/D reading 2 bytes (low then high)
                   LO:2                               ; lower limit 2 bytes
                   RES_HI                             ; working result register higher bits
                   TestFlag                           ; test status flag
                   temp
                   twoseconds                         ; 2 second deley loop

         endc

;***********************************************************************
;***** VARIABLE DEFINITIONS
w_temp             EQU      0x20                      ; variable used for context saving
status_temp        EQU      0x21                      ; variable used for context saving

FlagClear          EQU      B'00000000'               ; clear all flags - pass
ResLow             EQU      B'00000001'               ; result is lower
ResHigh            EQU      B'00000010'               ; result is higher

LowBit             EQU      H'0000'                   ; low bit use after testing
HighBit            EQU      H'0001'                   ; high bit



;***********************************************************************
         ORG     0x000          ; coding begins here
;***********************************************************************
         goto      start                              ; go to beginning of program


         ORG       0x004                              ; interrupt vector location
         movwf     w_temp                             ; save off current W register contents
         movf      STATUS,w                           ; move status register into W register
         movwf     status_temp                        ; save off contents of STATUS register

; isr code can go here or be located as a call subroutine elsewhere
```

```
        movf     status_temp,w            ; retrieve copy of STATUS register
        movwf    STATUS                   ; restore pre-isr STATUS register contents
        swapf    w_temp,f
        swapf    w_temp,w                 ; restore pre-isr W register contents
        retfie                            ; return from interrupt

start

        banksel  OSCCAL                   ; select bank1
        movlw    b'00000000'
        movwf    OSCCAL                   ; update register with factory cal value
        movlw    B'00101000'              ; set AN3 & AN5 to analog inputs
        movwf    ANSEL                    ;
        movlw    B'11111111'              ; RA0, RA1, RA2, RA3, RA4 & RA5 to inputs
        movwf    TRISA                    ;
        movlw    B'11000111'              ; set RC0 (AN4), RC1 (AN5) & RC2 to input,
                                          ; RC3, RC4 & 5 to outputs
        movwf    TRISC                    ;
        movlw    B'00100000'              ; fosc/32
        movwf    ADCON1

        banksel  PORTA                    ; select bank0
        movlw    B'10001101'              ; configure A/D justified right, Vref Vdd, Channel = AN3,
                                          ; A/D = ON
        movwf    ADCON0

        clrf     PORTA                    ; clear port A
        clrf     PORTC                    ; clear port C

;***********************************************************************
;       main loop
;***********************************************************************

mainloop

;***********************************************************************
; set the under-voltage limit to 10V
; ignore the upper limit by setting to maximum
; divide ratio 10V x 0.099 = 0.99
; FSD = 5V = 1024, bit resolution 4.88mV
; 0.99 / 0.00488 = 202.868
; set lower limit to 203 (11001011)
;***********************************************************************

        movlw    B'11001011'              ; set the PSU limit to 10V
        movwf    LO
        clrf     LO+1

        call     Measure_PSU
        call     TestLimits               ; test the result and set the appropriate condition flags

        btfss    TestFlag,HighBit         ; skip next instruction, connect the Ag102 and disconnect the
                                          ; battery
        goto     batloop                  ; goto to battey loop if PSU voltage is <10V
        bsf      PORTC,Power              ; connect Ag102  to PSU
        bcf      PORTC,Battery            ; ensure that the battery is disconnected
        goto     mainloop                 ; go back to mainloop to monitor the PSU voltage

batloop

;***********************************************************************
; set the under-voltage limit to 11V
; ignore the lower limit by setting to minimum
; divide ratio 11V x 0.099 = 1.089
; FSD = 5V = 1024, bit resolution 4.88mV
; 1.089 / 0.00488 = 223.155
; set upper limit to 223 (11011111)
;***********************************************************************

        movlw    B'11011111'              ; set the PSU limit to 11V
        movwf    LO
```

```
        clrf    LO+1

        call    Measure_PSU
        call    TestLimits                  ; test the result and set the appropriate condition flags

        btfss   TestFlag,LowBit             ; skip next instruction,
        goto    mainloop                    ; the PSU is >11V exit batloop and return to mainloop

        bcf     PORTC,Power                 ; disconnect Ag102

;**********************************************************************
; set the under-voltage limit to 10V
; ignore the upper limit by setting to maximum
; divide ratio 10V x 0.099 = 0.99
; FSD = 5V = 1024, bit resolution 4.88mV
; 0.99 / 0.00488 = 202.868
; set lower limit to 203 (11001011)
;**********************************************************************

        movlw   B'11001011'                 ; set the battery disconnect threshold to 10V
        movwf   LO
        clrf    LO+1

        call    Measure_BAT
        call    TestLimits                  ; test the result and set the appropriate condition flags

        btfsc   TestFlag,HighBit            ; skip next instruction and disconnect the battery if <10V
        goto    bat2
        bcf     PORTC,Battery               ; ensure that the battery is disconnected
        call    delay2s                     ; delay 2 seconds to allow the micro's supply rail to collapes
        goto    batloop                     ; if the power supply is connected before the rail collapeses
                                            ; or if the supply <10V go back to start of the battery loop

bat2

        bsf     PORTC,Battery               ; connect battery if >10V
        goto    batloop


;**********************************************************************
; delay routine using simple loops
;**********************************************************************

delay2s

        movlw   D'20'                       ; ~2S delay
        movwf   twoseconds

delay2sloop

        call    delay100ms
        decf    twoseconds
        skpz
        goto    delay2sloop

        return

delay100ms

        movlw   D'100'                      ; ~100mS delay
        movwf   Delay+2
        goto    delayLoop3

delay10ms

        movlw   D'010'                      ; ~10mS delay
        movwf   Delay+2
        goto    delayLoop3

delay1ms

        movlw   D'001'                      ; ~1mS delay
        movwf   Delay+2
```

```
delayLoop3

        movlw    0x02
        movwf    Delay+1

delayLoop2

        movlw    0x80
        movwf    Delay

delayLoop1

  decfsz Delay, f
        goto delayLoop1

  decfsz Delay+1, f
  goto delayLoop2

  decfsz Delay+2, f
  goto delayLoop3

  retlw 0

;**********************************************************************
; Analogue inputs
;**********************************************************************

Measure_PSU

        movlw    B'10001101'              ; configure A/D justified right, Vref Vdd,
                                          ; Channel = AN3(psu), A/D = ON
        movwf    ADCON0
        call     GetResult
        return

Measure_BAT

        movlw    B'10010101'              ; configure A/D justified right, Vref Vdd,
                                          ; Channel = AN5(bat), A/D = ON
        movwf    ADCON0
        call     GetResult
        return

;**********************************************************************
; measure A/D and store result
;**********************************************************************

GetResult

        banksel  ADRESH                   ; select bank0
        bsf      ADCON0,GO                ; start A/D

MeasureLoop

        btfsc    ADCON0,GO_DONE           ; read A/D status bit, jump past loop when done (low)
        goto     MeasureLoop

        movfw    ADRESH                   ; get upper bits
        movwf    AD+1                     ; store upper bits

        banksel  ADRESL                   ; select bank1
        movfw    ADRESL                   ; get lower bits
        banksel  ADRESH                   ; select bank0
        movwf    AD                       ; store lower bits

        return                            ; return from measurement routine (in bank0)

;**********************************************************************
; test the AD result against a nominal limit
;**********************************************************************

TestLimits
```

```
            clrf      TestFlag                    ; clear the test flag register

            movfw     AD+1                        ; get higher result
            movwf     RES_HI                      ; store in working register

            movfw     LO                          ; put lower 8 bits of the lower limit into the working register
            subwf     AD,W                        ; subtract the lower limit from test result and store result in
                                                  ; working register
                                                  ; if the measurement is above the limit Zero = 0, Carry = 1
                                                  ; if the measurement is equal to the limit Zero = 1, Carry = 1
                                                  ; if the measurement is low Zero = 0, Carry = 0
            skpnc                                 ; jump past the next statement if the result is low
            goto      TestLO_HI                   ; result equal or higher

            movf      RES_HI,F                    ; move upper bits to test for zero
            skpz                                  ; if the result is zero then nothing can be borrowed,
                                                  ; skip and set result low flag
            goto      DecLoLimit                  ; AD HI is not zero, jump to decrement and upper bit test

            goto      ResLowExit                  ; goto low fail

DecLoLimit

            decf      RES_HI,F                    ; decrement 1 from the upper bit and continue
                                                  ; with the lower limit test

TestLO_HI

            movfw     LO+1                        ; move higher limit bits into the working register and test if
zero
            skpnz                                 ; skip next command if it is not zero and test higher bits
            goto      ResHiExit                   ; the result equal or greater than the lower limit,
                                                  ; go to result high exit
            subwf     RES_HI,W                    ; subtract the lower limit from test result
                                                  ; and store result in working register
            skpnc                                 ; jump past the next statement if the result is low
            goto      ResHiExit                   ;

ResLowExit

            movlw     ResLow                      ; get result low flag
            movwf     TestFlag                    ; set the test status flag(as low)
            goto      TestExit                    ; exit the testlimits routine

ResHiExit

            movlw     ResHigh                     ; get result high flag
            movwf     TestFlag                    ; set the test status flag(as pass)

TestExit

            return                                ; return from result test routine (bank0)

;**********************************************************************
            END              ; directive 'end of program'
;**********************************************************************

; initialize eeprom locations

ORG         0x2100
DE          0x00, 0x01, 0x02, 0x03
```