Intelligent Dusk/Dawn Light Sensor

Richard S. LaBarbera *MSP430 Application*

ABSTRACT

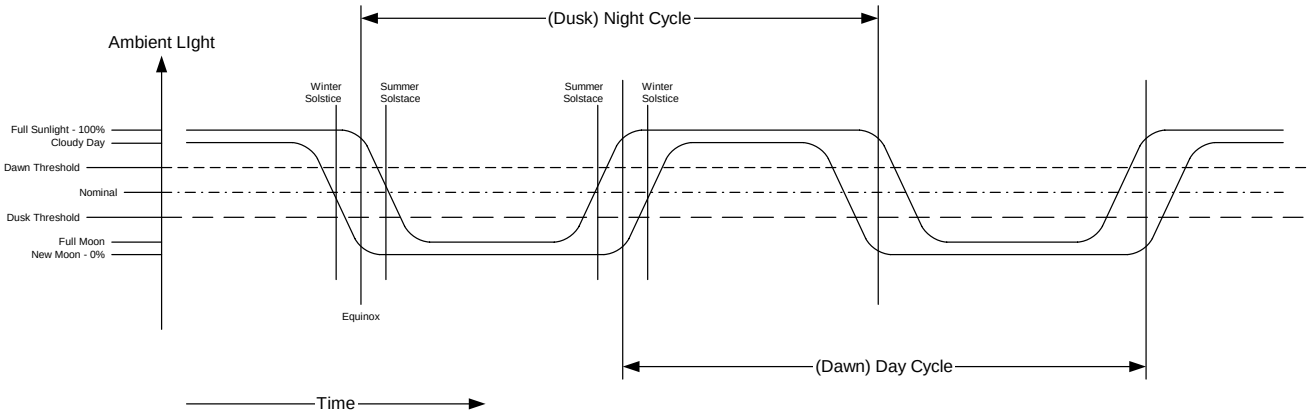
This document describes an application using the MSP430 (low pin count package) microcontroller to “intelligently” detect the daylight cycle transitions from light-to-dark (dusk) and dark-to-light (dawn) while filtering erroneous and spurious light level changes (noise) through a self-learning algorithm that automatically adapts to the normal daylight cycle. This application takes advantage of the rich feature set of the MSP430 family of microcontrollers by utilizing integral timers, analog-to-digital conversion (ADC), reference voltage generation, and clocks/oscillators; to minimize external circuit component requirement. Low power operation is also a major feature.

Introduction

How many times have you seen that rogue streetlamp that is on during the daytime or not on when needed at night? Or observed essential lighting needs turned-off due to a nearby lightning flash? These somewhat simple problems can be easily eliminated by adding just a small amount of intelligence to a standard photocell (or light sensitive resistor). A Cadmium-Sulfide (CdS) photocell similar to PN-120301 available from Jameco Electronics, will work well for this type of application. An external resistor used in series with the photocell to form a voltage divider that provides a voltage proportional to the light level observed by the photocell is all that is required for detection. This low component count circuit when combined with an MSP430 microcontroller device from Texas Instruments makes for an intelligent dusk/dawn sensor with the addition of a self-learning day-to-night and night-to-day algorithm implemented in software running on the MSP430.

Description of Normal Earth Daylight Cycle

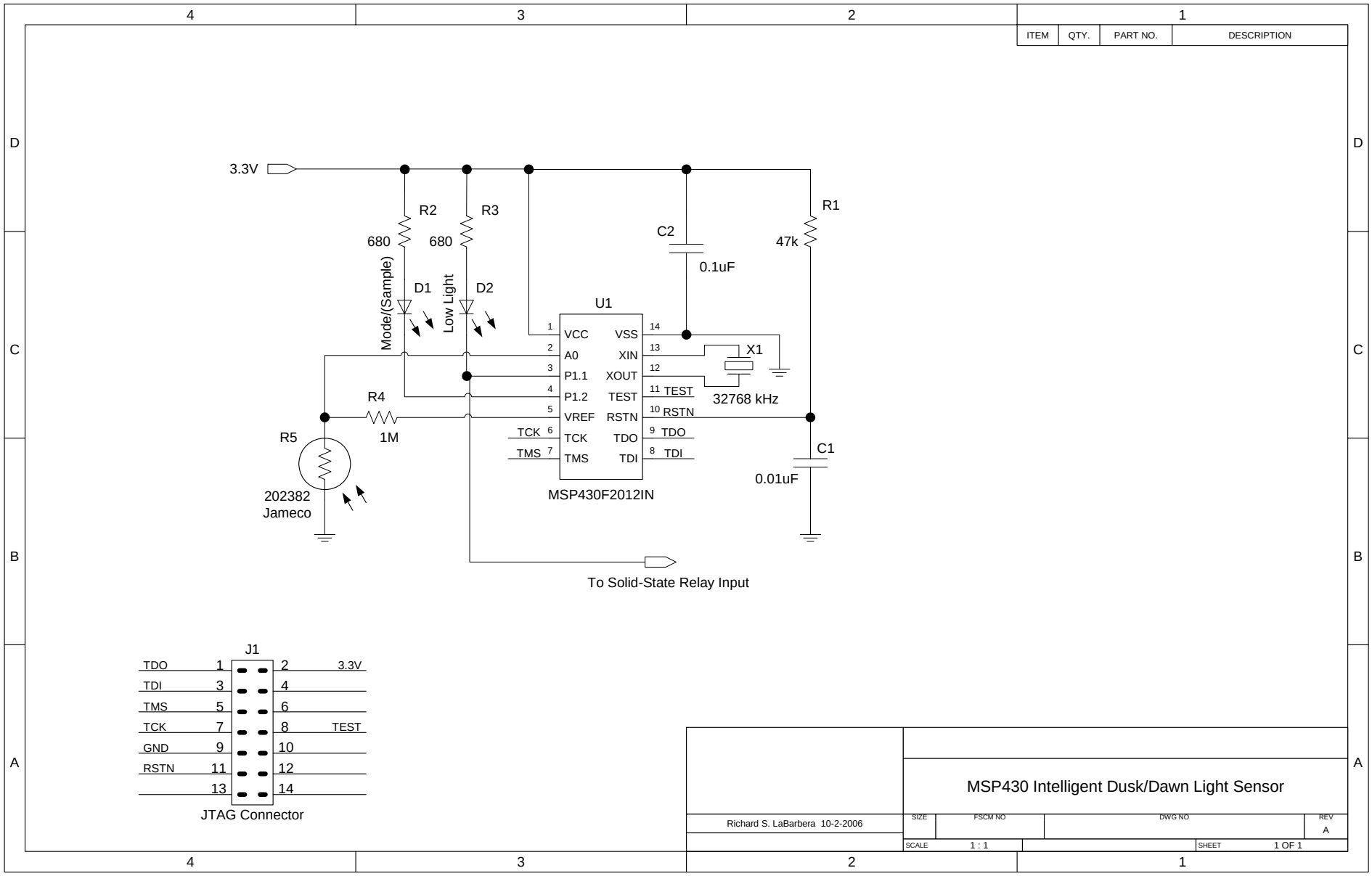
The following diagram shows a normalized cycle of a regular day and night as related to light levels falling on the earth's surface. (It is understood that other events can disrupt this "normal" cycle, but for simplicity this cycle depicts the expected light levels and intensity transitions as they usually occur.) The y-axis shows relative light intensity while the x-axis is advancing time from left to right.



Daylight Cycle

Circuit Implementation for the Intelligent Dusk/Dawn Light Sensor

The following diagram shows a circuit implementation for the intelligent dusk/dawn light sensor using the MSP430F2012IN (14-pin DIP) device. Using the DIP package for the MSP430 allowed for quick prototyping without the necessity of having surface mount capabilities readily available. All that was required beyond the sensing circuit (resistor and photocell) and the MSP430 device was the addition of a 32 kHz watch crystal and a header for JTAG connection to program the device for prototyping.


ITEM
QTY.
PART NO.
DESCRIPTION
3.3V
R2
680
R3
680
Mode/(Sample)
Low Light
D1
D2
C2
0.1uF
R1
47k
U1
VCC
A0
P1.1
P1.2
VREF
TCK
TMS
VSS
XIN
XOUT
TEST
RSTN
TDO
TDI
X1
32768 kHz
R4
1M
R5
202382
Jameco
C1
0.01uF
MSP430F2012IN
To Solid-State Relay Input
J1
TDO
TDI
TMS
TCK
GND
RSTN
3.3V
TEST
JTAG Connector
MSP430 Intelligent Dusk/Dawn Light Sensor
Richard S. LaBarbera 10-2-2006
SIZE
FSCM NO
DWG NO
REV
A
SCALE
1 : 1
SHEET
1 OF 1

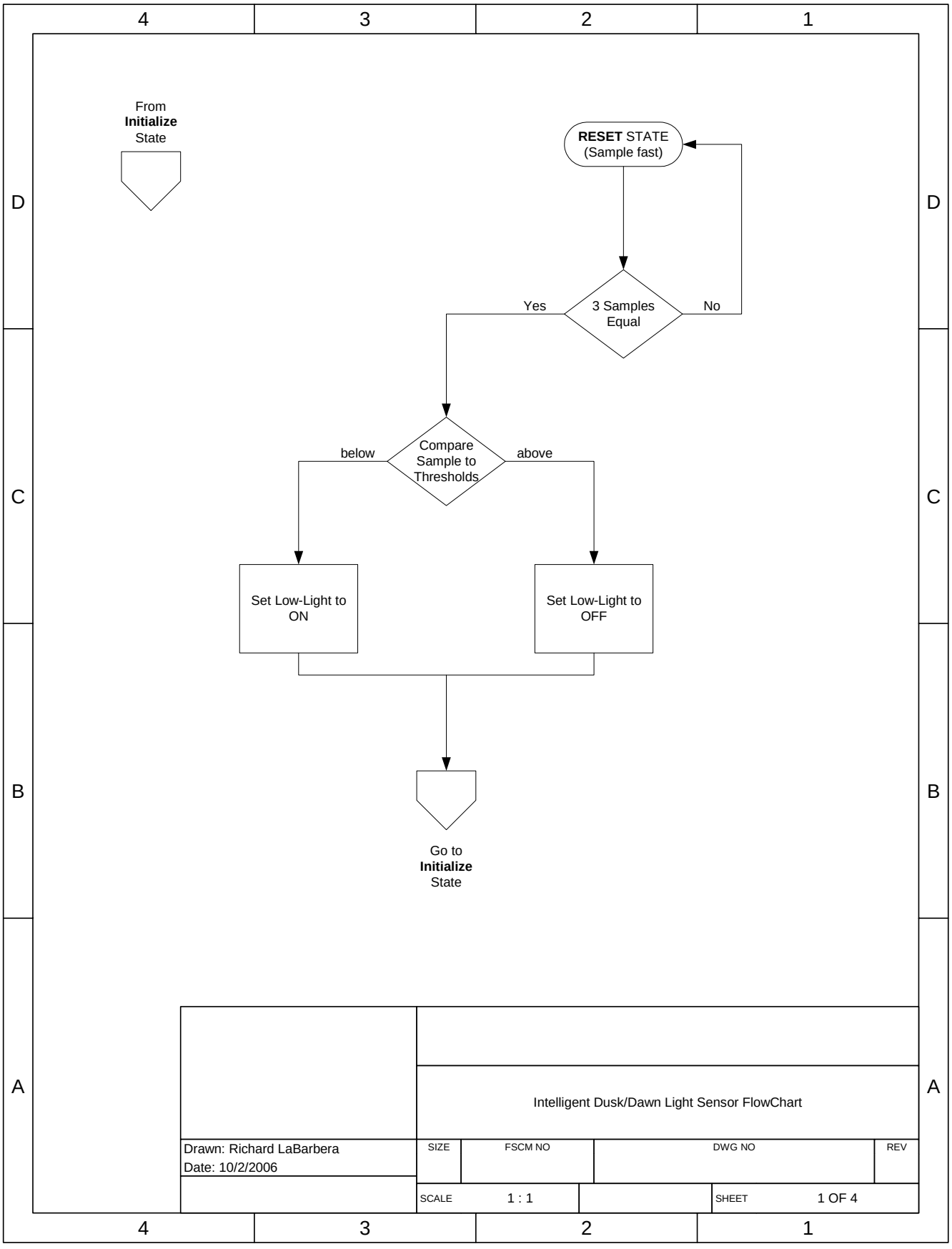
Intelligent Dusk/Dawn Algorithm

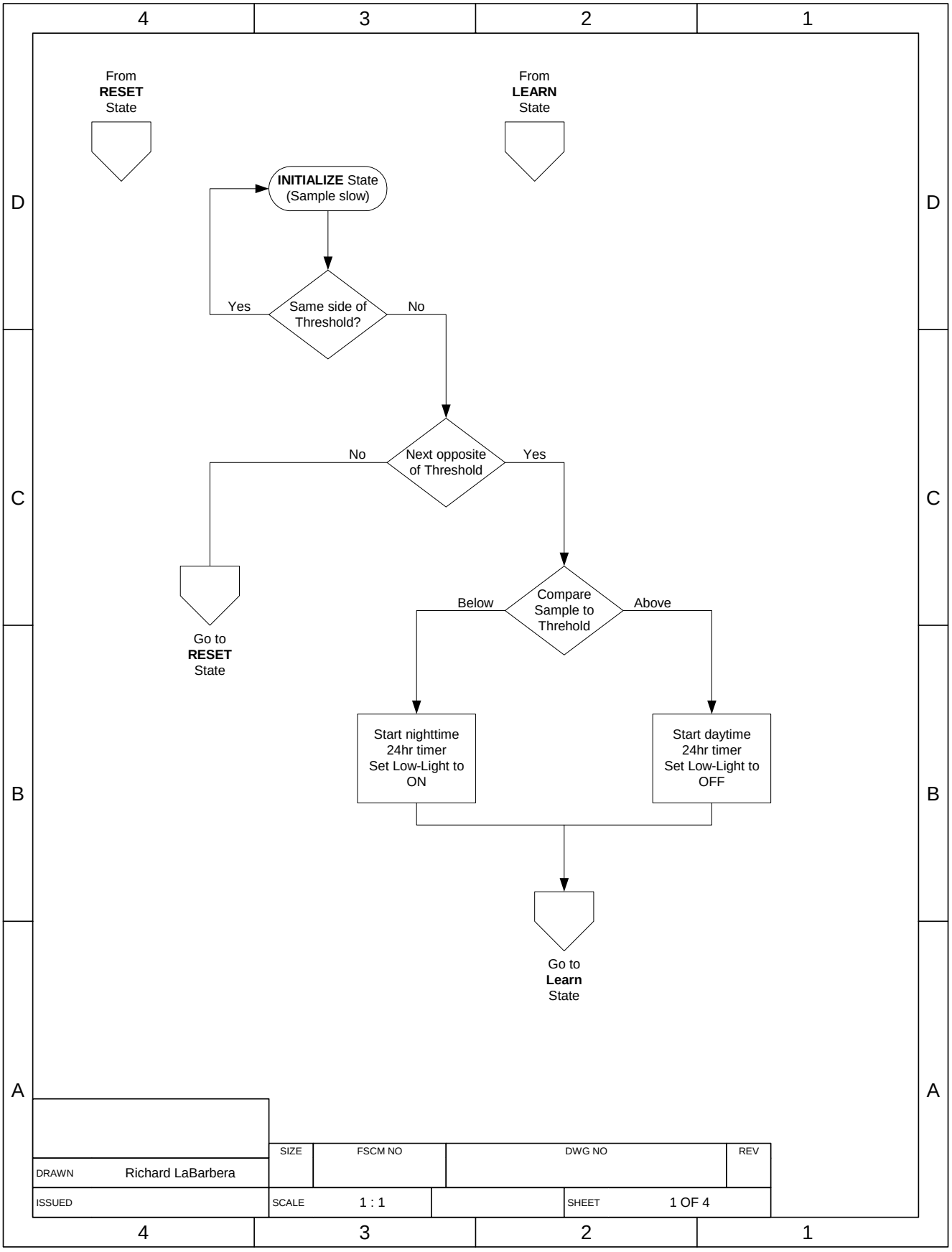
The 10-bit analog to digital converter included in the MSP430 is used to sample the voltage of the photocell/resister network for light level indication to the algorithm. Sampling time vary depending on the state of the algorithm. Four states are defined and transition from one state to the next is dependent upon decisive “filtering” criteria. One huge advantage in using this microcontroller is that not only can filtering be done on levels (light intensity) but with respect to time. And by time, not only is sampling time used for filtering, but expected transition time is used. This is the “intelligent” or learned part of the algorithm. Knowing that dusk-to-dusk and dawn-to-dawn occur close to 24-hours apart while dusk-to-dawn and dawn-to-dusk can vary depending upon the tilt of the earth (seasonal change – short days in winter and long days in summer) can be used to intelligently predict the next upcoming transition. To properly track time, a 32 kHz watch crystal is used (connected directly to the MSP430 low frequency oscillator inputs) to produce an accurate time-of-day clock with 1-second watch dog timer interrupts to increment time keeping counters. Levels from the ADC are also used to filter noise (from other light sources). Different thresholds are used for night-to-day detection and day-to-night detection, providing hysteresis. The internal reference voltage of the MSP430 is also used to provide a stable 1.5-volt reference for the sensor circuit.

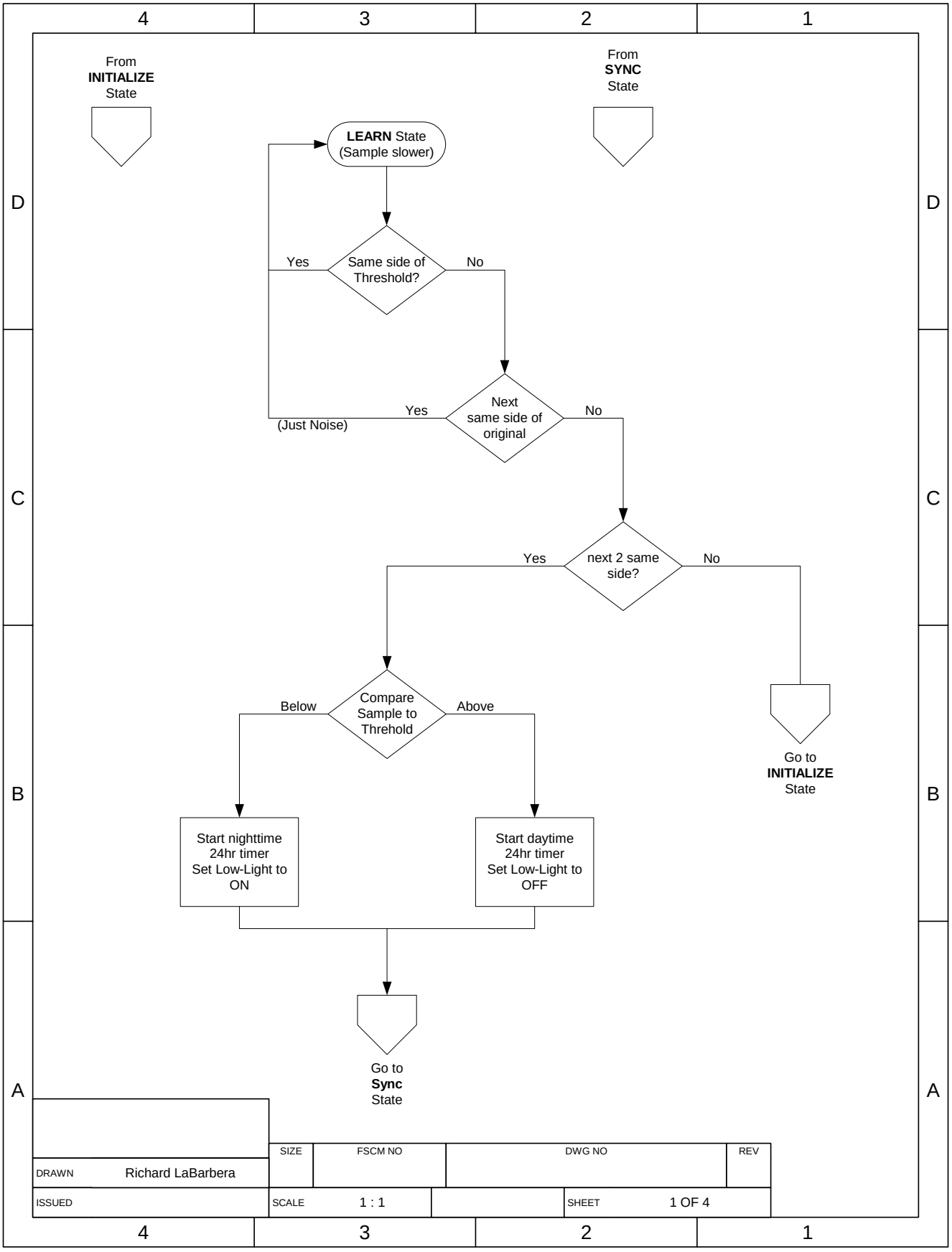
The four states of the algorithm include RESET, INITIALIZE, LEARN and SYNC. These states are shown in the next section in the algorithm flow diagram and transition criteria can be seen going between each of these states. A power-up sequence or a hardware/software reset will put the algorithm back in the RESET state.

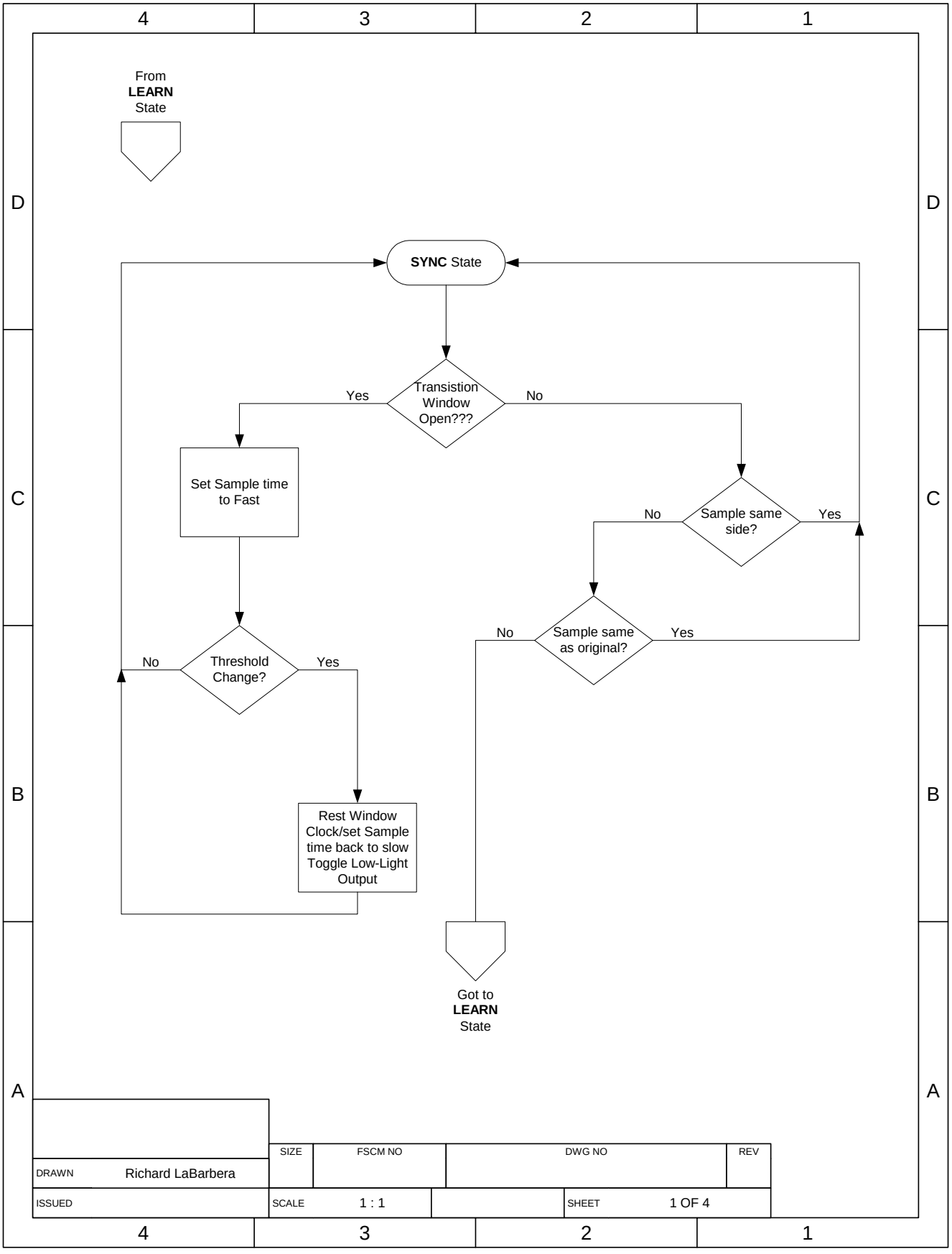
Flow Diagram for Software Algorithm

The following flow chart diagrams the intelligent algorithm used to filter out spurious light level changes by using sampling time settings and voltage threshold levels.


4
3
2
1
D
C
B
A
From **Initialize** State
RESET STATE (Sample fast)
3 Samples Equal
Yes
No
Compare Sample to Thresholds
below
above
Set Low-Light to ON
Set Low-Light to OFF
Go to **Initialize** State
Intelligent Dusk/Dawn Light Sensor FlowChart
Drawn: Richard LaBarbera
Date: 10/2/2006
SIZE
FSCM NO
DWG NO
REV
SCALE 1 : 1
SHEET 1 OF 4


4
3
2
1
D
C
B
A
From
RESET
State
From
LEARN
State
INITIALIZE State
(Sample slow)
Same side of
Threshold?
Yes
No
Next opposite
of Threshold
No
Yes
Go to
RESET
State
Compare
Sample to
Threhold
Below
Above
Start nighttime
24hr timer
Set Low-Light to
ON
Start daytime
24hr timer
Set Low-Light to
OFF
Go to
Learn
State
DRAWN
Richard LaBarbera
ISSUED
SIZE
FSCM NO
DWG NO
REV
SCALE
1 : 1
SHEET
1 OF 4


4
3
2
1
From
INITIALIZE
State
From
SYNC
State
LEARN State
(Sample slower)
Same side of
Threshold?
Yes
No
Next
same side of
original
Yes
(Just Noise)
No
next 2 same
side?
Yes
No
Compare
Sample to
Threhold
Below
Above
Go to
INITIALIZE
State
Start nighttime
24hr timer
Set Low-Light to
ON
Start daytime
24hr timer
Set Low-Light to
OFF
Go to
Sync
State
SIZE
FSCM NO
DWG NO
REV
DRAWN
Richard LaBarbera
ISSUED
SCALE
1 : 1
SHEET
1 OF 4


4
3
2
1
D
C
B
A
From
LEARN
State
SYNC State
Transistion
Window
Open???
Yes
No
Set Sample time
to Fast
Sample same
side?
No
Yes
Sample same
as original?
No
Yes
Threshold
Change?
No
Yes
Rest Window
Clock/set Sample
time back to slow
Toggle Low-Light
Output
Got to
LEARN
State
DRAWN
Richard LaBarbera
ISSUED
SIZE
FSCM NO
DWG NO
REV
SCALE
1 : 1
SHEET
1 OF 4

Assembly Code for Intelligent Dusk/Dawn Light Sensor

The algorithm for the intelligent dusk/dawn light sensor has been written in assembly language. There are four “include” files required in conjunction with the main controller source code. These include files are:

ADC_sensor.h
sensor_filter.h
time_isr.h
msp430x20x2.h

The include file ***msp430x20x2.h*** is the standard file provided by the IAR Embedded Workbench IDE.

File: Dusk_Dawn_MSP430F2012.s43

```
;-------------------------------------------------------------------------
; MSP430F2012IN Microcontroller - Intelligent Dusk/Dawn Light Sensor
; File: Dusk_Dawn_MSP430F2012.s43
; Author: Richard LaBarbera
; October 2, 2006
;
;
;
;
;
;
;                MSP430F2012
;             -----------------
;         /|\|              XIN|-
;          | |                 |
;          --|RST          XOUT|-
;            |                 |
;  sensor>---|P1.0/A0      P1.1|-->LED
;            |                 |
;            |             P1.2|-->LED
;            |                 |
;            |             P1.4|-->Vref
;            |                 |
;
;
;
;   ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = DCO
;   //* External watch crystal installed on XIN XOUT is required for ACLK *//
;-------------------------------------------------------------------------
#include  "msp430x20x2.h"

;-------------------------------------------
; PORT definitions
;-------------------------------------------

#define     DRIVE_LED BIT1,&P1OUT           /* LED & DRIVE     P1.1      */
#define     MODE_LED  BIT2,&P1OUT           /* LED & STATUS    P1.2      */
;-------------------------------------------
; Constants
;-------------------------------------------

#define     lo_v_th   00CDh                 /*300mV threshold           */
#define     hi_v_th   0111h                 /*400mV threshold           */
#define     w_open    059Bh                 /*1435minutes (24hr-5min)   */
#define     w_close   05A5h                 /*1445minutes (24hr+5min)   */
#define     s0_time   0001h                 /*1sec - state0 sample time  */
#define     s1_time   000Ah                 /*10sec - state1 sample time */
#define     s1_ftime  0001h                 /*1sec - state1 sample time  */
#define     s2_time   000Ah                 /*10sec - state2 sample time */
#define     s2_ftime  0001h                 /*1sec - state2 sample time  */
#define     s3_time   003Ch                 /*1min - state3 sample time  */
#define     s3_ftime  000Ah                 /*10sec - state3 fast sample */
```

```
;-----------------------------------------
;test values for time constants –
;     - uncomment to use
;     - remember to comment out the ones above!
;-----------------------------------------
;#define     s0_time   0001h                /*1sec - state0 sample time  */
;#define     s1_time   0002h                /*2sec - state1 sample time  */
;#define     s1_ftime  0001h                /*1sec - state1 sample time  */
;#define     s2_time   0005h                /*5sec - state2 sample time  */
;#define     s2_ftime  0001h                /*1sec - state2 sample time  */
;#define     s3_time   000Ah                /*10sec - state3 sample time */
;#define     s3_ftime  0002h                /*2sec - state3 fast sample  */

;-----------------------------------------
; Register variables
;-----------------------------------------
#define     VOLTAGE   R7

;-----------------------------------------
; Storage Area
;-----------------------------------------
sample      equ     0200h                  ;sample storage area

;-----------------------------------------
; Data variables
;-----------------------------------------
            org     0210h                  ;RAM address

daytime_counter        dw      1
nighttime_counter      dw      1
seconds                dw      1
sample_pointer         dw      1
prev_sample_pointer    dw      1
day_night_flag         dw      1
conv_flag_value        dw      1
adc_conv_flag_sensor   dw      1
sensor_state           db      1


;-----------------------------------------
            org     0f800h                 ;Code memory
;-----------------------------------------
#include  "include\ADC_sensor.h"
#include  "include\sensor_filter.h"
#include  "include\time_isr.h"

;-----------------------------------------
            RSEG    CSTACK                 ; Define stack segment
;-----------------------------------------
            RSEG    CODE                   ; Assemble to Flash memory
;-----------------------------------------
;-----------------------------------------
;  A RESET Starts Here
;-----------------------------------------
RESET       mov.w   #SFE(CSTACK),SP        ; Initialize stackpointer
SetupWDT    mov.w   #WDT_ADLY_1000,&WDTCTL ;init WDT for 1 second
```

```
;-------------------------------------------
;Set Up ADC Reference
;-------------------------------------------
            mov     #REFON+REFOUT+ADC10ON+ADC10SHT_2,&ADC10CTL0  ;+REF1_5V
                                            ;Turn on 1.5V ref, set sample time
            bis.b   #01h,&ADC10AE0          ;P1.0 ADC10 option select
            bis     #SREF_1,&ADC10CTL0      ;
            bis     #INCH_0,&ADC10CTL1      ;Sample channel = 0
            mov     #2600h,R12              ;delay for needed ref start-up
L$3         dec     R12                     ;See datasheet for details
            jnz     L$3                     ;Loop

;-------------------------------------------
;Initialize ADC Sample Storage
;-------------------------------------------
            mov.w   #0000h,sample_pointer   ;
            mov.w   #01111h,R5              ;
            mov.w   #0000h,R6               ;
fill_sample
            mov.w   R5,sample(R6)           ;load sample memory
            add.w   #01111h,R5              ;shift contents to make unique
            add.w   #0002h,R6               ;increment pointer
            cmp     #0008h,R6               ;check for end
            jne     fill_sample             ;

;-------------------------------------------
;Set Up LED Output
;-------------------------------------------
            bis.b   #BIT1+BIT2,&P1DIR       ;LED outputs
            bis.b   #MODE_LED               ;init LED OFF
            bis.b   #DRIVE_LED              ;init LED OFF

;-------------------------------------------
;Initialize Flags
;-------------------------------------------
            mov   #0000h,adc_conv_flag_sensor;
            mov   #0001h,conv_flag_value    ;1 second for state 0
            mov.b   #01,day_night_flag      ;initalize to Daytime

;-------------------------------------------
;Initialize Sensor State
;-------------------------------------------
            mov.b   #00,sensor_state        ;

;-------------------------------------------
;Initialize Counters
;-------------------------------------------
            mov.w   #00,daytime_counter     ;
            mov.w   #00,nighttime_counter   ;

;-------------------------------------------
;Enable Interrupts
;-------------------------------------------
            bis.b   #WDTIE,&IE1             ;enable WDT interrupts
            eint                            ;

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

```
;
mainloop                                    ;MAIN Loop Begins Here
;
;-------------------------------------------
check_sensor_flag
            cmp     conv_flag_value,adc_conv_flag_sensor;
            jlo     main_continue           ;
            bic.b   #MODE_LED               ;
            call    #check_sensor           ;
main_continue
            mov     #2600h,R12              ;delay to see LED
L$1         dec     R12                     ;
            jnz     L$1                     ;Loop
            bis.b   #MODE_LED               ;
;-------------------------------------------
;
            jmp     mainloop                ;MAIN Loop Ends Here
;
;-------------------------------------------

ERRVEC                                      ;
            jmp     ERRVEC                  ;Trap for Error
            reti                            ;

;-------------------------------------------
; Interrupt Vectors for MSP430F2012
;-------------------------------------------
            rseg    INTVEC
            dw      ERRVEC                  ;0 reserved
            dw      ERRVEC                  ;1 reserved
            dw      ERRVEC                  ;2 port 1 Vector
            dw      ERRVEC                  ;3 port 2 Vector
            dw      ERRVEC                  ;4 reserved
            dw      ERRVEC                  ;5 reserved
            dw      ERRVEC                  ;6 TIMER_A CC0 Vector
            dw      ERRVEC                  ;7 reserved
            dw      ERRVEC                  ;8 TIMER_A CC1-2 TA Vector
            dw      ERRVEC                  ;9 reserved
            dw      WDT_ISR                 ;10 WDT Vector
            dw      ERRVEC                  ;11 Comparator Vector
            dw      ERRVEC                  ;12 reserved
            dw      ERRVEC                  ;13 reserved
            dw      ERRVEC                  ;14 NMI Vector
            dw      RESET                   ;15 RESET Vector
            END
```

File: ADC_sensor.h

```
;-------------------------------------------------------------------------
; MSP430F2012IN Microcontroller - Intelligent Dusk/Dawn Light Sensor
; File: ADC_sensor.h
; Author: Richard LaBarbera
; October 2, 2006
;-------------------------------------------------------------------------
;-----------------------------------------
; Analog to Digital conversion of Sensor Voltage
;-----------------------------------------
adc_conv_sensor
            bis     #ENC,&ADC10CTL0         ;enable conversions
            bis     #ADC10SC,&ADC10CTL0     ;Start conversion
testIFG2    bit     #ADC10BUSY,&ADC10CTL1   ;Conversion done?
            jnz     testIFG2                ;No, test again
            bic     #ADC10SC,&ADC10CTL0     ;End conversion
            bic     #ENC,&ADC10CTL0         ;disable conversions
            mov.b   #0,adc_conv_flag_sensor ;
            and.w   #03fch,&ADC10MEM        ;mask LSBs ADC mem1
            mov     &ADC10MEM,VOLTAGE       ;hold voltage
            mov.w   sample_pointer,R6       ;
            mov.w   &ADC10MEM,sample(R6)    ;store results
            mov.w   sample_pointer,prev_sample_pointer
                                            ;save pointer for later use
            add.w   #02h,sample_pointer     ;
            and.w   #07h,sample_pointer     ;modulo pointer
            ret
```

File: ADC_sensor.h

```
;-------------------------------------------------------------------------
; MSP430F2012IN Microcontroller - Intelligent Dusk/Dawn Light Sensor
; File: sensor_filter.h
; Author: Richard LaBarbera
; October 2, 2006
;-------------------------------------------------------------------------
;-----------------------------------------
; Filter Sensor Voltage to set dusk/dawn indication
;-----------------------------------------
check_sensor
            call    #adc_conv_sensor        ;

;-----------------------------------------
; check for current state
;-----------------------------------------
            cmp.b   #03, sensor_state       ;sync STATE???
            jnz     state_2                 ;not state_3

;-----------------------------------------
; state_3
;-----------------------------------------
check_day_window
            cmp     #w_open,daytime_counter ;
            jlo     check_night_window      ;
            cmp     #w_close,daytime_counter;
            jhs     window_closed_day_rst   ;
                                            ;new day window is OPEN
            mov     #s3_ftime,conv_flag_value;

            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jhs     sensor_exit             ;
            bis.b   #DRIVE_LED              ;set as daytime
            bic     #BIT0,day_night_flag    ;
            clr     daytime_counter         ;
            mov     #s3_time,conv_flag_value;
            jmp     sensor_exit             ;

check_night_window
            cmp     #w_open,nighttime_counter;
            jlo     window_closed           ;
            cmp     #w_close,nighttime_counter;
            jhs     window_closed_night_rst ;
                                            ;new night window is OPEN
            mov     #s3_ftime,conv_flag_value;

            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
```

```
            jlo     sensor_exit             ;
            bic.b   #DRIVE_LED              ;set as nighttime
            bis     #BIT0,day_night_flag    ;
            clr     nighttime_counter       ;
            mov     #s3_time,conv_flag_value;
            jmp     sensor_exit             ;


window_closed_day_rst
            clr     daytime_counter         ;
            jmp     window_closed           ;
window_closed_night_rst
            clr     nighttime_counter       ;
window_closed
            mov     #s3_time,conv_flag_value;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_1_s3     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     first2different_s3      ;
            jmp     first2same_s3           ;
last_check_day_1_s3
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jhs     first2different_s3      ;
first2same_s3
            mov     #s3_time,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_2_s3     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jhs     sensor_exit             ;
            jmp     back2state_2            ;
last_check_day_2_s3
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jlo     sensor_exit             ;
back2state_2
            mov.b   #02h,sensor_state       ;go back to state_1
            mov     #s2_time,conv_flag_value;
            jmp     sensor_exit             ;
first2different_s3
            mov     #s3_ftime,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_3_s3     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     threshold_tripped_s3    ;
            jmp     sensor_exit             ;
last_check_day_3_s3
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jlo     sensor_exit             ;
threshold_tripped_s3                        ;crossed the threshold from d to n
            mov.b   #02,sensor_state        ;go to state_2
```

```
            mov     #s2_time,conv_flag_value;
            jmp     sensor_exit             ;


;-----------------------------------------
; check for next state
;
state_2
;-----------------------------------------
            cmp.b   #02, sensor_state       ;learn STATE???
            jnz     state_1                 ;not state_2
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_1_s2     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     first2different_s2      ;
            jmp     first2same_s2           ;
last_check_day_1_s2
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jhs     first2different_s2      ;
first2same_s2
            mov     #s2_time,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_2_s2     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jhs     sensor_exit             ;
            jmp     back2state_1            ;
last_check_day_2_s2
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jlo     sensor_exit             ;
back2state_1
            mov.b   #01h,sensor_state       ;go back to state_1
            mov     #s1_time,conv_flag_value;
            jmp     sensor_exit             ;
first2different_s2
            mov     #s2_ftime,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_3_s2     ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     threshold_tripped_s2    ;
            jmp     sensor_exit             ;
last_check_day_3_s2
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jlo     sensor_exit             ;
threshold_tripped_s2                        ;crossed the threshold from d to n
            bit     #BIT0,day_night_flag    ;
            jz      change2night_s2         ;
            bis.b   #DRIVE_LED              ;set as daytime
            bic     #BIT0,day_night_flag    ;
            clr     daytime_counter         ;
```

```
            jmp     go_sync                 ;
change2night_s2
            bic.b   #DRIVE_LED              ;set as nighttime
            bis     #BIT0,day_night_flag    ;
            clr     nighttime_counter       ;
go_sync
            mov.b   #03,sensor_state        ;go to state_3
            mov     #s3_time,conv_flag_value;
            jmp     sensor_exit             ;


;-----------------------------------------
; check for next state
;
state_1
;-----------------------------------------
            cmp.b   #01, sensor_state       ;init STATE???
            jnz     state_0                 ;not state_1
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_1        ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     first2different         ;
            jmp     first2same              ;
last_check_day_1
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jhs     first2different         ;
first2same
            mov     #s1_time,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_2        ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jhs     sensor_exit             ;
            jmp     back2state_0            ;
last_check_day_2
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jlo     sensor_exit             ;
back2state_0
            mov.b   #00h,sensor_state       ;go back to state_0
            mov     #s0_time,conv_flag_value;
            jmp     sensor_exit             ;
first2different
            mov     #s1_ftime,conv_flag_value;
            mov.w   sample_pointer,R6
            add.w   #04h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            mov     sample(R6),VOLTAGE      ;
            bit     #BIT0,day_night_flag    ;
            jz      last_check_day_3        ;
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     threshold_tripped       ;
            jmp     sensor_exit             ;
last_check_day_3
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
```

```
            jlo     sensor_exit             ;
threshold_tripped                           ;crossed the threshold from d to n
            bit     #BIT0,day_night_flag    ;
            jz      change2night            ;
            bis.b   #DRIVE_LED              ;set as daytime
            bic     #BIT0,day_night_flag    ;
            clr     daytime_counter         ;
            jmp     go_learn                ;
change2night
            bic.b   #DRIVE_LED              ;set as nighttime
            bis     #BIT0,day_night_flag    ;
            clr     nighttime_counter       ;
go_learn
            mov.b   #02,sensor_state        ;go to state_2
            mov     #s2_time,conv_flag_value;
            jmp     sensor_exit             ;

;-------------------------------------------
state_0
;-------------------------------------------
cmp_samples
            mov.w   sample_pointer,R6
            add.w   #02h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            cmp     sample(R6),VOLTAGE      ;cmp last to first samples
            jne     sensor_exit             ;no match
            add.w   #02h,R6                 ;point next sample
            and.w   #07h,R6                 ;modulo pointer
            cmp     sample(R6),VOLTAGE      ;cmp last to previous samples
            jne     sensor_exit             ;no match
match
            mov.w   prev_sample_pointer,R6  ;get current sample
            mov     sample(R6),VOLTAGE      ;hold voltage - last sample
            cmp.w   #hi_v_th,VOLTAGE        ;cmp to high threshold
            jhs     sense_complete_night    ;
            mov     sample(R6),VOLTAGE      ;hold voltage - last sample
            cmp.w   #lo_v_th,VOLTAGE        ;cmp to low threshold
            jlo     sense_complete_day      ;
            bit     #BIT0,day_night_flag    ;
            jz      sense_complete_day      ;already set as Daylight
sense_complete_night
            bic.b   #DRIVE_LED              ;set as nighttime
            bis     #BIT0,day_night_flag    ;
            jmp     sense_complete          ;
sense_complete_day
            bis.b   #DRIVE_LED              ;set as daytime
            bic     #BIT0,day_night_flag    ;
sense_complete
            mov.b   #01,sensor_state        ;go to state_1
            mov     #s1_time,conv_flag_value;
;-------------------------------------------
sensor_exit
;-------------------------------------------
            ret
```