

Automatic Recovery and Reset Algorithms for System Controller Errors

Yon-Sik Lee*

*Professor, School of Computer Information and Communication Engineering, Kunsan National University,
Kunsan, Korea

[Abstract]

Solar lamp systems may not operate normally in the event of some system or controller failure due to internal or external factors, in which case secondary problems occur, which may cost the system recovery. Thus, when these errors occur, a technology is needed to recover to the state it was in before the failure occurred and to enable re-execution. This paper designs and implements a system that can recover the state of the system to the state prior to the time of the error by using the Watchdog Timer within the controller if a software error has occurred inside the system, and it also proposes a technology to reset and re-execution the system through a separate reset circuit in the event of hardware failure. The proposed system provides stable operation, maintenance cost reduction and reliability of the solar lamp system by enabling the system to operate semi-permanently without external support by utilizing the automatic recovery and automatic reset function for errors that occur in the operation of the solar lamp system. In addition, it can be applied to maintain the system's constancy by utilizing the self-operation, diagnosis and recovery functions required in various high reliability applications.

▶ **Key words:** Solar lamp system, Controller, Remote monitoring, Error recovery, Reset algorithm

[요 약]

본 논문은 시스템 내부에서 소프트웨어 오류가 발생하였을 경우 컨트롤러 내의 Watchdog Timer를 이용하여 시스템의 상태를 오류 발생시점 이전 상태로 복구하는 시스템을 설계 구현하고, 하드웨어 오류 발생 시 별도의 리셋 회로를 통해 시스템을 재실행할 수 있는 기술을 제안한다. 제안 시스템은 외부 지원 없이 시스템 자체적으로 반영구적으로 작동 할 수 있도록 함으로써, 시스템의 안정적인 작동, 유지비용 절감 및 신뢰성을 제공하며, 고 신뢰성 응용분야에서 요구되는 자가 동작, 진단 및 복구 기능을 통한 시스템의 항상성 유지를 위한 적용이 가능하다.

▶ **주제어:** 태양광 가로등 시스템, 컨트롤러, 원격 모니터링, 오류 복구, 리셋 알고리즘

-
- First Author: Yon-Sik Lee, Corresponding Author: Yon-Sik Lee
 - Yon-Sik Lee (yslee@kunsan.ac.kr), School of Computer Information and Communication Engineering, Kunsan National University
 - Received: 2019. 10. 11, Revised: 2019. 12. 04, Accepted: 2019. 12. 05.

I. Introduction

IoT 기술 기반의 원격 모니터링을 통한 원격 복구 및 리셋 기술은 아직 연구 단계에 있는 실정이며[1,2], 한국델파이와 S&T모터스의 OSEK-OS기반의 차량 전장품 개발 기술이나 Vector사의 AUTOSAR SWC 테스트 자동화 시스템 기술과 같은 임베디드 시스템 기반 기술은 별도의 운영 체제 환경에서 개발이 진행 중이다. 특히 이러한 고 신뢰성 응용서비스 기술들은 국방, 자동차 산업 분야는 현재 시작 단계이다[2,3,4]. 이러한 IoT 기반의 원격 모니터링 기능을 적용한 태양광 가로등 시스템은 자체 컨트롤러의 펌웨어에 의해 측정 데이터에 따라 배터리 충·방전 및 등기구 ON/OFF, 밝기조절 동작 등을 실행하면서 운영되며, 외부 신호선 잡음이나 펌웨어 상에서 비정상적 메모리 참조 및 오버플로우 등의 오류에 의한 오동작이 발생할 경우 정상적인 운영이 불가능하다. 또한 위치 확인이 어려운 경우 배터리의 자연 방전 등으로 2차적 문제 발생과 시스템 복구를 위한 비용이 요구된다[3,4]. 따라서, 현장에 설치된 시스템의 컨트롤러에서 오류가 발생할 경우, 이전 상태로 복구 및 재 동작이 가능한 기술이 필요하지만[5,6], 기존 시스템들은 IoT 기반의 원격 모니터링 및 트래픽 제어 기능을 포함할 뿐 시스템 오류 발생에 대한 자동 복구 및 리셋 기능을 적용한 사례는 미흡한 실정이다[1,2,6].

본 연구의 목표는 태양광 가로등 시스템에 이러한 고 신뢰성 응용서비스 기술을 적용하여 시스템이 외부 도움 없이 스스로 반영구적인 동작을 수행하도록 하는 기술의 개발이다. 이를 위하여, 본 논문에서는 소프트웨어적 오류 발생 시 측정 데이터의 저장과 컨트롤러 내의 Watchdog Timer를 이용하여 다운되는 시점 이전 상태로 복구하는 알고리즘과, 외부 또는 하드웨어적 오류 발생 시 별도의 알고리즘에 의하여 컨트롤러의 MCU 자체를 자동으로 리셋하는 회로 및 펌웨어 개발 기술을 제안한다.

II. Operational Information Store and Error Automatic Recovery Algorithm

다음 Fig. 1은 컨트롤러의 동작 감지 및 오류 자동복구 기능을 지원하기 위하여 설계한 컨트롤러 내부 구성도이며, 본 논문의 주요 연구 및 구현 범위는 Firmware Main을 기반으로 연동되는 Memory, Watchdog Timer, Reset IC 등의 적용기술이다.

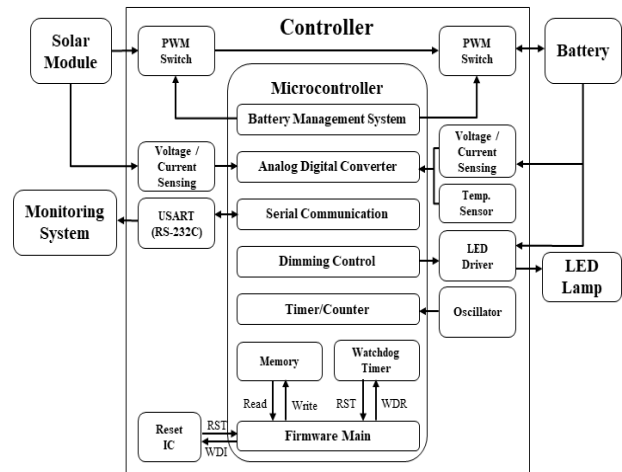


Fig. 1. Configurations of the controller in the solar lamp system

태양광 가로등 시스템은 Figure 1과 같이 태양광 모듈, 컨트롤러, 배터리, LED 등기구, 모니터링 시스템으로 구성되며, 컨트롤러에서 시스템을 운영한다. 컨트롤러는 마이크로 컨트롤러, PWM Switch, 전압/전류 센싱 모듈, USART (시리얼통신), 온도센서, LED Driver, Oscillator, Reset IC 등의 회로로 구성된 하드웨어와 마이크로 컨트롤러 내부 펌웨어로 동작한다[7]. 펌웨어는 배터리 충·방전 및 보호를 위한 BMS, 태양광 모듈에서 발전되는 전압/전류와 배터리에서 방전되는 전압, 전류 및 온도 측정을 위한 ADC, 모니터링을 위한 시리얼 통신, 가로등의 밝기조절을 위한 디밍 제어, 동작 시간 및 경과 시간 계산을 위한 Timer/Counter 등의 기능들로 구성되며, 운영정보 등의 데이터 저장을 위하여 마이크로 컨트롤러 내부 메모리를 사용하고, Watchdog Timer에 주기적으로 WDR 신호를 전송하여 시스템이 리셋되는 현상을 방지한다.

1. Selection and store of the measured data

시스템 운영에 필요한 데이터는 시스템의 고장 분석 및 동작 상황 모니터링을 위한 측정 데이터와 시스템의 고장 시점 및 계절별 낮/밤 시간 계산을 통한 점등시간 자동 조절을 위한 운영 데이터이다. 컨트롤러의 동작 감지 및 오류 자동복구를 위한 이와 같은 데이터들을 컨트롤러의 제한된 메모리에 효율적으로 저장하기 위하여 본 논문에서는 AVR 계열의 ATmega128 마이크로 컨트롤러[9]를 사용하고, 메모리는 프로그램 메모리, 데이터 메모리, EEPROM으로 구성하며, 시스템 운영에 필요한 데이터들은 4K Byte 크기의 EEPROM에 1 Byte 단위로 저장한다. EEPROM의 데이터를 읽는 횟수는 제한을 두지 않고, 10만 번 까지 쓰기가 가능하도록 효율적으로 공간을 설계하

여 운영한다. 이와 같은 데이터들을 저장하는 EEPROM 데이터 메모리의 저장구조 및 주소 초기화 내용은 다음 Table 1과 같다.

Table 1. Storage structure of EEPROM data memory

Stored Item		Unit	Size	Memory Addr.	Total Size
System Operation Time		min.	4	0x000~0x003	4
Storage address for the next day's day/night hours		hexa	2	0x004~0x005	2
Day/night hours of the day	Daytime	min.	2	0x006~0x011	12 (3day)
	Night time	min.	2		
Storage address of measured data after 10 minutes		hexa	2	0x012~0x013	2
measured data (ave.)	Photovoltaic module voltage	V	2	0x014~0xD93	3,456 (12*288)
	Charging current	A	2		
	Battery1 voltage	V	2		
	Battery2 voltage	V	2		
	Discharge current	A	2		
	Controller temperature	°C	2		

시스템 구동 후 (2^{32})분 동안의 컨트롤러 동작 시간저장이 가능하며, 안정화 이후 조정이 가능하다. 다음날 낮/밤 시간이 저장될 메모리의 주소는 밤에서 낮으로 전환된 경우 다음날로 인지하여 0x004~0x005의 범위에 2 Byte로 저장하고, 당일 낮/밤 시간은 각각 2 Byte로 구분하여 4 Byte의 1일 데이터를 3일간 갱신 저장한다. 측정데이터들의 평균은 각각 2 Byte를 사용하여 set (12 Byte)로 저장하며, 1분 간격으로 측정 후 10분 동안의 평균값으로 약 2일간 (288회)의 데이터를 저장하고, 10분경과 후 측정 데이터가 저장될 메모리의 주소를 갱신한다.

메모리에 데이터를 저장하는 방법은 쓰기와 갱신으로 구분하고, 1 Byte 단위로 저장한다. 쓰기는 1 Byte 이상의 데이터 set을 Byte 단위로 덮어쓰기로 수행하고, 갱신은 1 Byte 이상의 데이터 set을 저장 메모리에 기록된 데이터와 Byte 단위로 비교하여 다른 데이터만 덮어쓰기로 수행한다.

시스템 운영에 필요한 운영 데이터 저장은 시스템 동작 시간 계산을 위한 저장 시간의 간격 (1분)을 설정하고, 메모리 쓰기 제한에 따라서 저장 간격 조절이 가능하도록 한

다. 낮/밤 시간의 계산을 위한 데이터는 태양광 모듈 전압을 기준으로 낮과 밤을 구분하며, 저장된 각각의 데이터를 메모리로부터 읽어 들여 1분 단위로 갱신한다. 또한, 측정 데이터는 1분마다 시스템 동작 시간을 증가시켜 측정 후 메모리에 갱신하며, 데이터는 6개의 데이터를 하나의 set으로 측정하고, 10분간 측정 데이터를 임시 저장 후 평균을 계산하여 메모리에 저장한다. 데이터 저장 후 메모리 주소를 측정 데이터의 크기만큼 증가시키고, 이를 다음 데이터가 저장될 주소 저장 메모리에 갱신한다.

다음 *Algorithm 1*은 데이터 저장을 위한 EEPROM 주소 및 파라미터를 초기화 시킨 후, 낮/밤 시간과 측정된 데이터 저장 알고리즘이다.

Algorithm 1: Data Store

```
// Initialization of EEPROM address for data store
#define CTIME_ADR      0x000
#define DATE_ADR      0x004
#define DDATA_ADR      0x006
#define SAVE_ADR      0x012
#define MDATA_ADR      0x014
#define MAX_MDATA_ADR 0xD93

// Initialization of parameters for data store
#define SAVE_DAY      3
#define SAVE_TIME     10
#define SAVE_ITEM      6
#define SAVE_PERIOD    1
#define PRINT_SIZE     8
#define DAY_IDX       6
#define MOON_IDX      7
#define NULL_IDX      6
#define TIME_IDX      7

// Day/Night time store function
void dayMoonTime_Save(void)
{ char dayOrMoon = DETECT_Get_isDayOrMoon();
  eeprom_read_block((void *)dayMoon,
    (const void *) dayAddress, sizeof(dayMoon));
  if(dayOrMoon == DAY)    dayMoon[DAY]++;
  else    dayMoon[MOON]++;
  eeprom_update_block((const void *)dayMoon,
    (void *)dayAddress, sizeof(dayMoon)); }

// Measured data store function
void dataSet_Save(void) {
  int i, j;
  int totalDataSet[SAVE_ITEM] = {0,};
  int averageDataSet[SAVE_ITEM] = {0,};
  dataSet[saveTime][PV_VAD] = ADC_Get(PV_VAD);
  dataSet[saveTime][PV_IAD] = ADC_Get(PV_IAD);
  dataSet[saveTime][BV_VAD0] = ADC_Get(BV_VAD0);
  dataSet[saveTime][BV_VAD1] = ADC_Get(BV_VAD1);
  dataSet[saveTime][BV_IAD] = ADC_Get(BV_IAD);
  dataSet[saveTime][TP_IC] = ADC_Get(TP_IC);
  saveTime++;
  timeCount++;
  eeprom_update_dword((uint32_t *)
    CTIME_ADR, timeCount);
  if(saveTime >= SAVE_TIME) {
    for(i = 0; i < SAVE_ITEM; i++) {
```

```

for(j = 0; j < SAVE_TIME; j++) {
    totalDataSet[i] += dataSet[j][i];
    averageDataSet[i] = totalDataSet[i] / SAVE_TIME;
    eeprom_update_block((const void *)averageDataSet,
        (void *)saveAddress, eof(averageDataSet));
    saveAddress += sizeof(dataSet);
    if(saveAddress >= MAX_MDATA_ADR)
        saveAddress = MDATA_ADR;
    saveTime = 0;
    for(i = 0; i < SAVE_ITEM; i++) {
        for(j = 0; j < SAVE_TIME; j++)
            dataSet[j][i] = 0;
        totalDataSet[i] = 0;
        averageDataSet[i] = 0;
    }
}

```

EEPROM Address	Contents	EEPROM Data	Unit	Data Size
0x000 - 0x003	Operating Time	96	Minute	4Byte
0x004 - 0x005	Daylight Time Next Address	0x006	Hexadecimal	2Byte
0x006 - 0x007	Day 1 Daylight Time	0	Minute	2Byte
0x008 - 0x009	Day 1 Night Time	96	Minute	2Byte
0x00A - 0x00B	Day 2 Daylight Time	0	Minute	2Byte
0x00C - 0x00D	Day 2 Night Time	0	Minute	2Byte
0x00E - 0x00F	Day 3 Daylight Time	0	Minute	2Byte
0x010 - 0x011	Day 3 Night Time	0	Minute	2Byte
0x012 - 0x013	Measure Data Next Address	0x080	Hexadecimal	2Byte
0x014 - 0x093	Measure Data	See the next table	-	3,458Byte

Address	PV(V)	PV(A)	BAT1(V)	BAT2(V)	Discharge(A)	Temp(C)
0x014	1.4	0.0	11.8	11.8	1.3	22.7
0x020	1.4	0.0	11.8	11.8	1.3	24.2
0x02C	1.4	0.0	11.8	11.8	1.3	24.8
0x038	1.4	0.0	11.8	11.8	1.4	25.0
0x044	1.4	0.0	11.8	11.8	1.3	25.2
0x050	1.4	0.0	11.8	11.8	1.4	25.2
0x05C	1.4	0.0	11.8	11.8	1.4	25.5
0x068	1.4	0.0	11.8	11.8	1.3	25.6
0x074	1.4	0.0	11.8	11.8	1.3	25.6
0x080	0.0	0.0	0.0	0.0	0.0	0.0
0x08C	0.0	0.0	0.0	0.0	0.0	0.0
0x098	0.0	0.0	0.0	0.0	0.0	0.0
0x0A4	0.0	0.0	0.0	0.0	0.0	0.0
0x0B0	0.0	0.0	0.0	0.0	0.0	0.0
0x0BC	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 2. Monitoring data stored in the EEPROM of micro-controller

Fig. 2는 모니터링 프로그램을 이용하여 EEPROM에 저장된 데이터를 확인한 결과를 나타낸 것으로, 컨트롤러 자체로는 저장 데이터 확인이 불가능하므로 시리얼 통신으로 저장 데이터를 전송하고 컨트롤러와 연결을 확인 후 EEPROM Read 기능으로 전송 요청한 결과이다. 이는 오류 복구를 위하여 요구되는 자료 (낮/밤 시간 데이터, 발전량, 충전 및 방전량, 시스템 내부 온도 등)들이 주어진 조건에 따라 정확하게 저장되어 있음을 보여준다.

2. Error automatic recovery algorithm

시스템 운영 중 오류 발생은 다양한 형태로 나타나며 오동작 및 동작 멈춤 상태를 유발시킨다. 오류 발생 시 강제로 시스템을 리셋하여 재 동작시키면 이전의 측정 데이터와 운영 데이터가 삭제되므로, 오류 발생 이전의 이들 데이터를 메모리에 저장하여 시스템 리셋 후에도 시스템의 원활한 운영을 위하여 이들이 사용될 수 있도록 해야 한다[1, 2]. 이를 위하여 본 논문에서 구현한 오류 자동복

구 알고리즘(Algorithm 2)은 메모리에 저장된 데이터 복구 함수와 데이터를 출력하여 모니터링 프로그램에 전송하는 함수 등으로 구성된다.

Algorithm 2: Automatic Error Recovery

// Stored data recovery function

```

void HISTORY_Init(void)
{ NoWait_ByMin_T2(&savePeriod, 0);
  eeprom_busy_wait();
  timeCount =
    eeprom_read_dword((uint32_t*)CTIME_ADR);
  if(timeCount != 0) {
    dayAddress =
      eeprom_read_word((uint16_t*)DATE_ADR);
    saveAddress =
      eeprom_read_word((uint16_t*)SAVE_ADR);
  } else {
    dayAddress = DATE_ADR;
    saveAddress = SAVE_ADR; } }

```

// Stored data output function

(transfer to the monitoring program)

```

void HISTORY_Print(void)
{ uint16_t t_dayAddress, t_saveAddress;
  uint32_t t_timeCount;
  eeprom_busy_wait();
  t_timeCount =
    eeprom_read_dword((uint32_t*)CTIME_ADR);
  eeprom_busy_wait();
  t_dayAddress =
    eeprom_read_word((uint16_t*)DATE_ADR);
  eeprom_busy_wait();
  t_saveAddress =
    eeprom_read_word((uint16_t*)SAVE_ADR);
  tx_EVENT_INFO(SND_TIMECOUNT, (int *)
    &t_timeCount, sizeof(t_timeCount));
  _delay_ms(100);
  print_DayMoon(t_dayAddress);
  print_DataSet(t_saveAddress); }

```

오류 자동복구 알고리즘에 의한 데이터 저장 및 복구 과정은 다음과 같다.

1) 제한된 메모리의 공간과 저장 횟수를 고려하여 시스템의 운영 데이터 저장 주기를 초기화 (시스템의 동작 시간과 낮/밤 시간은 1분마다, 측정 데이터는 10분마다 저장)

2) EEPROM 메모리에 저장된 시스템 동작 시간을 읽어 시스템의 최초 운영 시점부터 1분 단위로 기록하며, 동작 시간이 0분일 경우 최초 동작이나 시스템 리셋으로 간주하여 낮/밤 시간과 측정 데이터가 저장될 메모리 주소를 읽어 해당 메모리에 운영 데이터를 저장하고, 동작 시간이 0분이 아닌 경우 낮/밤 시간과 측정 데이터가 저장될 주소를 메모리에서 읽어 들임

3) 낮/밤 시간 데이터를 메모리에서 읽고, 매 분 간격으로 동작 시간과 낮 시간과 밤 시간을 각각 증가시킴

4) 매 분 간격으로 데이터 (전압, 전류, 온도 등)들을 측

정하고, 측정 시간이 10분이 되면 측정 데이터의 평균값을 해당 메모리 주소에 저장

5) 측정된 데이터의 평균값을 메모리에 저장 후 다음 측정 데이터가 저장될 메모리 주소를 측정 데이터의 크기만큼 증가하여 저장

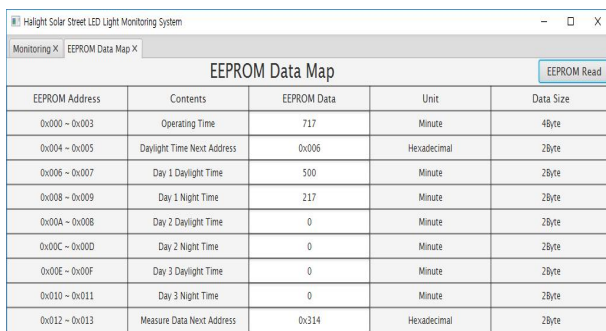
6) 측정 시간 및 측정 데이터를 초기화

7) 밤에서 낮으로 전환되는 경우를 1일이 바뀌는 기준으로 설정하고, 1일이 바뀌면 다음 낮/밤 시간이 저장될 메모리 주소를 낮/밤 시간 데이터의 크기만큼 증가하여 저장

8) 다음 낮/밤 시간이 저장될 메모리 공간 초기화

9) EEPROM 메모리에 저장된 데이터 출력이 요구되면, 모니터링 프로그램에서 메모리 데이터 전송을 요청하고 시리얼 통신으로 메모리에 저장된 데이터를 모니터링 프로그램으로 전송

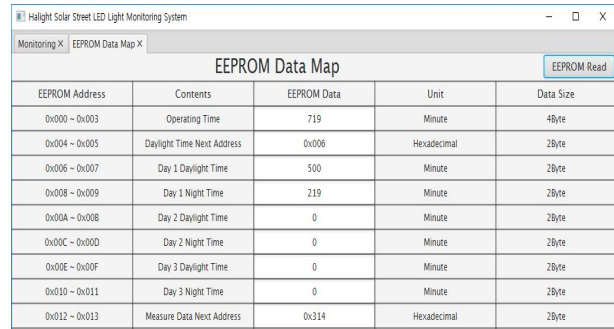
본 논문에서는 위와 같은 오류 자동복구 알고리즘에 의한 데이터 저장 및 오류 복구 과정의 올바름을 증명하기 위하여, 시스템의 배터리 및 태양광 모듈의 전원을 차단하여 오류 발생 상태로 설정하고 시스템 리셋 후 재 작동한 상태를 오류 복구 상태로 설정하여, 태양광 모듈을 연결하지 않아 밤 상태로 가정한 후 점등 확인 및 데이터 복구 상태를 실험하였다. 다음 Fig. 3은 실험을 위한 오류 발생 전의 EEPROM 데이터를 나타낸다.



EEPROM Address	Contents	EEPROM Data	Unit	Data Size
0x000 ~ 0x003	Operating Time	717	Minute	4Byte
0x004 ~ 0x005	Daylight Time Next Address	0x006	Hexadecimal	2Byte
0x006 ~ 0x007	Day 1 Daylight Time	500	Minute	2Byte
0x008 ~ 0x009	Day 1 Night Time	217	Minute	2Byte
0x00A ~ 0x00B	Day 2 Daylight Time	0	Minute	2Byte
0x00C ~ 0x00D	Day 2 Night Time	0	Minute	2Byte
0x00E ~ 0x00F	Day 3 Daylight Time	0	Minute	2Byte
0x010 ~ 0x011	Day 3 Night Time	0	Minute	2Byte
0x012 ~ 0x013	Measure Data Next Address	0x314	Hexadecimal	2Byte

Fig. 3. EEPROM data before the error occurred

다음 Fig. 4는 오류에 대한 자동 리셋 후의 EEPROM 데이터를 비교한 실험 결과로써, 시스템의 운영 시간과 밤 시간은 오류 발생 전부터 복구 후까지의 데이터들이 손실 없이 누적되어 기록됨을 보여준다.



EEPROM Address	Contents	EEPROM Data	Unit	Data Size
0x000 ~ 0x003	Operating Time	719	Minute	4Byte
0x004 ~ 0x005	Daylight Time Next Address	0x006	Hexadecimal	2Byte
0x006 ~ 0x007	Day 1 Daylight Time	500	Minute	2Byte
0x008 ~ 0x009	Day 1 Night Time	219	Minute	2Byte
0x00A ~ 0x00B	Day 2 Daylight Time	0	Minute	2Byte
0x00C ~ 0x00D	Day 2 Night Time	0	Minute	2Byte
0x00E ~ 0x00F	Day 3 Daylight Time	0	Minute	2Byte
0x010 ~ 0x011	Day 3 Night Time	0	Minute	2Byte
0x012 ~ 0x013	Measure Data Next Address	0x314	Hexadecimal	2Byte

Fig. 4. EEPROM data after system reset

III. System Automatic Reset Algorithm

1. Design of automatic reset circuit and its algorithm

자동리셋 기능을 위하여 프로세서가 안정적으로 동작하는지 감시하는 기능을 수행하는 마이크로 컨트롤러 내부에 Watchdog Timer를 내장하여, 다음 Fig. 5와 같이 지정한 주기 이상이 경과 되면 1 clock cycle의 리셋 펄스를 발생시키고, 리셋 펄스가 끝난 후 일정 시간 (t_{TOUT})이 경과될 때까지 리셋 상태를 유지하고 시스템을 리셋한다 [8]. 주기적으로 동작하는 Watchdog Timer를 설정하면 주기적으로 위치독 동작 금지 명령을 실행하여 시스템 리셋 동작을 방지한다.

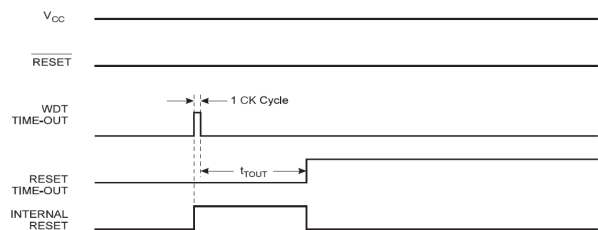


Fig. 5. Timing diagram of the Watchdog reset operation

Firmware 및 외부요인으로 인하여 리셋이 불가능할 경우에는 독립적 리셋 신호 공급이 필요하므로, 마이크로 컨트롤러 외부에 리셋 신호를 제공하는 Reset IC (TPS3823)를 추가하여 마이크로 컨트롤러의 동작을 감시하고, 마이크로 컨트롤러는 Reset IC에 주기적으로 시스템 리셋 동작 금지신호를 송신한다. 다음 Fig. 6은 Reset IC의 타이밍도를 나타내며[9], Fig. 7은 Reset IC를 이용한 자동 및 수동 리셋 회로도를 나타낸다.

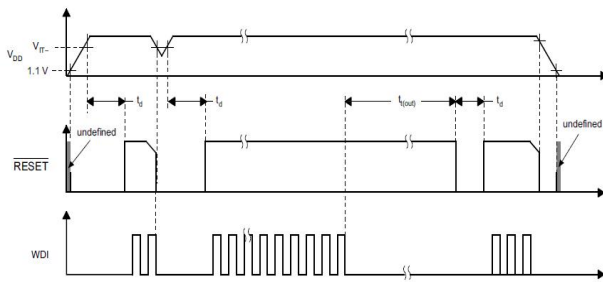


Fig. 6. Timing diagram of the Reset IC

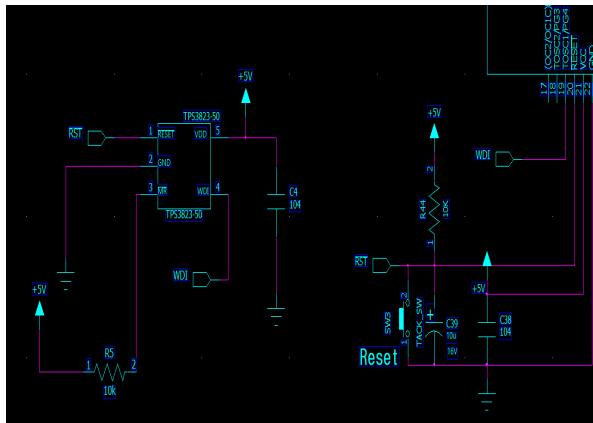


Fig. 7. Automatic and manual reset circuit diagrams using Reset IC

본 논문에서 설계 구현한 자동리셋 알고리즘은 다음 Algorithm 3과 같다.

Algorithm 3: Automatic Reset

```
static int is_wdr;
// Signal transmission to WDI pin and Watchdog timer
reset inhibit function
void run_ResetIC(void)
{ if(NoWait_By2ms(&is_wdr, 250)) {
  PORTG |= 0x10;
  asm("wdr");
  NoWait_By2ms(&is_wdr, 0); } }
int main(void)
{
// System initialization
DETECT_RESET();
cli();
CFG_init();
NoWait_By2ms(&is_wdr, 0);/
TIMER0_init();
TIMER1_init();
TIMER2_init();
TIMER3_init();
USART_init();
ADC_Init();
sei();
PORTC = 0B00000011;
ACTIVE_Func = IdelFunc;
Check_SystemIO();
PORTC = 0;
```

```
PARAM_Init();
DIMMING_Set_UserMode();
DETECT_Init();
HISTORY_Init();
_delay_ms(100);
// System operation
for(;;) {
  run_ResetIC();
  ADC_preCalculate();
  PARSER_SndMeasureValue();
  DETECT_DayMoon();
  ACTIVE_Func();
  HISTORY_Save();
  do_Sensing_LED();
  do_ModeTest(KEY_IsKeyPress()); / }
return 0; }
```

시스템 자동리셋 처리 과정은 다음과 같다.

먼저, 시스템 재실행 및 초기 실행 상태를 확인 (컨트롤러의 녹색 LED와 적색 LED가 동시에 켜졌다 꺼짐 상태로 확인)하고, 주기적으로 Reset IC에 신호를 전송하기 위한 출력 포트 설정한 후, 신호 전송 시간 계산을 위하여 타이머를 초기화한다. 0.5초 주기의 타이머 시간에 따라 출력 포트에 ON 신호를 전송하고, 0.5초 주기로 위치독 동작 금지 명령을 실행한 후 1.8초 동안 금지 명령이 실행되지 않으면 리셋을 실행하고 타이머 시간을 초기화한다. Reset IC는 WDI 핀에서 신호를 감지하고, 일정 시간 동안 신호가 없는 경우에 리셋 신호를 전송하며, 컨트롤러는 Reset IC의 리셋 신호를 받아 시스템을 재 구동한다.

2. Experiment of automatic reset operation

다음 Fig. 8은 구현된 Watchdog Timer 리셋 기능에 대한 실험 결과이다.

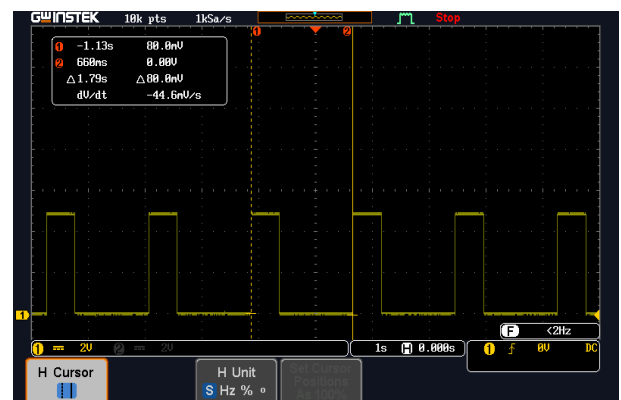


Fig. 8. Reset output waveform by Watchdog timer

Fig. 8은 위치독 동작 금지 명령을 삭제할 경우 자동으로 리셋 동작이 실행됨을 보여주는 출력 파형을 나타내며, 실험은 오실로스코프를 이용하여 컨트롤러의 초기 실행

상태를 확인하는 적색 LED의 + 단자에서 신호 및 주기를 측정할 것이다. 출력 파형에서 파형이 상승된 구간은 적색 LED가 켜진 상태이며, 다음 상승 구간은 시스템이 재시작된 경우이다. 파형의 상승 구간과 다음 상승 구간의 주기는 약 1.79s로 일정하다. 따라서 Watchdog Timer에 의해 시스템이 리셋 됨을 확인할 수 있다.

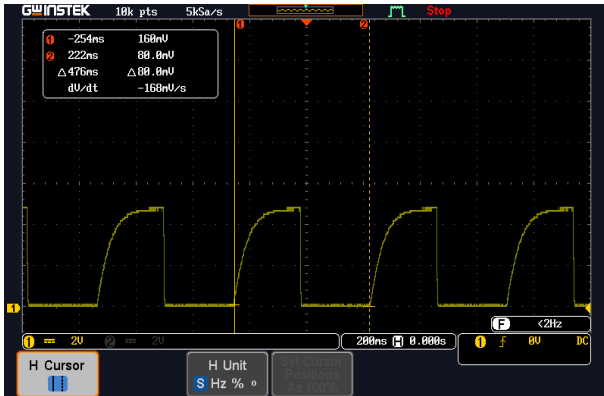


Fig. 9. Reset output waveform by Reset IC

Reset IC를 이용한 리셋 기능에 대한 실험 결과는 Fig. 9와 같다. 이는 마이크로 컨트롤러에서 Reset IC에 WDI 신호를 주지 않을 경우 자동으로 리셋 동작을 수행하는 실험으로, Fig. 7에서 Reset IC의 RST 단자가 컨트롤러의 리셋 버튼과 연결되어 있고, 오실로스코프를 이용하여 리셋 버튼의 단자를 측정한 것이다.

위와 같은 동작 실험을 통하여 마이크로 컨트롤러에서 입치독 동작 금지 명령과 Reset IC의 WDI 신호가 없을 경우 주기적으로 컨트롤러가 재동작 하는 것을 확인함으로써, 펌웨어의 알 수 없는 실행 오류 및 하드웨어 오류 발생 시 Watchdog Timer와 Reset IC를 이용하여 시스템을 자동으로 재 동작하도록 하는 알고리즘과 개발된 펌웨어의 신뢰성과 적응성을 확인할 수 있다.

IV. Conclusion

IoT 기반의 원격 모니터링 기능을 적용한 태양광 가로등 시스템은 측정 데이터에 따라 컨트롤러의 펌웨어에 의해 필요한 동작들을 실행한다. 그러나 내외부적 오류에 의한 오동작이 발생할 경우 정상적인 운영이 불가능하고 2차적 문제 발생과 시스템 복구를 위한 비용이 증가하므로, 오류 발생 시 이전 상태 데이터를 통한 자동복구 및 리셋

기술이 필요하다. 이러한 고 신뢰성 응용서비스 기술 적용을 위하여, 본 논문은 태양광 가로등 시스템을 대상 시스템으로 설정하여 시스템 운영에 필요한 측정 데이터와 운영 데이터를 컨트롤러의 제한된 메모리 공간에 효율적으로 저장하고, 컨트롤러 내의 Watchdog Timer를 이용하여 다운되는 시점 이전 상태로 복구하는 알고리즘과 하드웨어 시스템의 자동리셋 기능을 포함한 컨트롤러 회로 설계 및 펌웨어 개발 기술을 구현 적용하였다.

제안 시스템은 신뢰성과 강건성을 요구하는 제어응용 시스템들의 오류 환경에 대한 적응적 제어기술을 제공함으로써, 시스템 운영의 안정성과 신뢰성을 확보하고 유지 보수 효율성을 제공하며, 다양한 원격 감시 및 제어 시스템들에 대한 자가 진단 및 복구와 자율 동작 지원에 적용이 가능하다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (018R1D1A1B07051045)

(This work was published in part in the Proceedings of the 60th Korean Society of Computer Information Conference, July 2019)

REFERENCES

- [1] J. J. Lee, et al., "IoT based Mobile Smart Monitoring System for Solar Power Generation," *Journal of IEK*, 54(8), pp. 55-64, 2017 <http://dx.doi.org/10.5573/ieie.2017.54.8.55>
- [2] M. Saifuzzaman, et. al., "IoT Based Street Lighting And Traffic Management System," *Humanitarian Technology Conference*, 2017 IEEE Region 10, pp. 121-124, 2017 DOI: 10.1109/R10-HTC.2017.8288921
- [3] H. O. Song, "IoT Automatic Management System based on Situational Task Control," Ph.D. thesis of Paijue University, 2017
- [4] C. Lee, Y. Lai, "Design and Implementation of a Universal Smart Energy Management Gateway based on the IoT Platform," 2016 IEEE Int'l Conference on Consumer Electronics, pp. 67-68, 2016
- [5] M. H. Park, et al., "A Study on the Development of Energy IoT Platform," *Journal of KIPS*, 5(10), pp. 311-318, 2016 <http://dx.doi.org/10.3745/KTCCS.2016.5.10.311>

- [6] A. Jain, C. Nagarajan, "Efficient Control Algorithm for a Smart Solar Street Light," 2015 9th Int'l Conference on NGMAST, pp. 376-381, 2015. DOI: 10.1109/NGMAST.2015.40
- [7] Y. Lee, Y. Mun, "Design and implementation of IoT based controllers and communication module interfaces for stand-alone solar system," Journal of The Korea Society of Computer and Information, Vol.24, No.1, pp.129-135, 2019 <http://doi.org/10.9708/jksci.2019.24.01>
- [8] ATmega128 Manual, ATMEL
- [9] TPS382x Voltage Monitor With Watchdog Timer, TI

Authors



Yon-Sik Lee received the B.S. and M.S. degrees in Computer Science from Chonnam National University, Korea, in 1982 and 1984, respectively. And, his Ph.D. degree in Computer Application Engineering from

Chonbuk National University, Korea, in 1994. Dr. Lee joined the faculty of the School of Computer Information and Communication Engineering, Kunsan National University, Kunsan, Korea, in 1986. He is interested in sensor network middleware, active rule system, agent system and cloud computing.