INU Object Toolkit for Internet of Things

June 20, 2015

SDT Lab
Incheon National University

Contents

Introduction

- **Hardware**
  - Smart Phone
    - ✓ Best Terminal
    - ✓ Sensors & Connections
  - Atmega128 Board
    - ✓ Bluetooth to UART
    - ✓ Bridge to "Things"

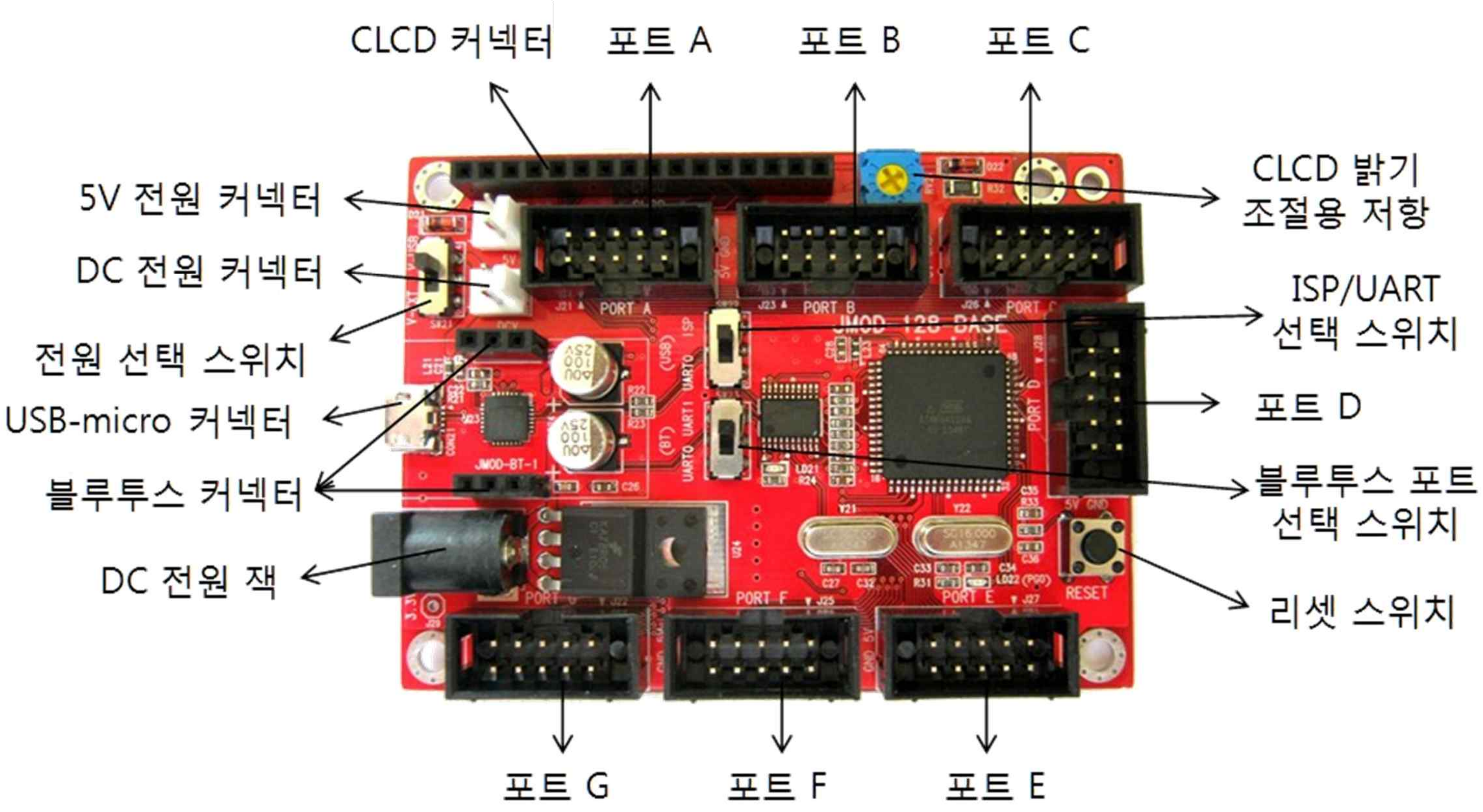
- **Software**
  - Android/Java for Smart Phone
  - C++ IoT Package for Atmega128
    - ✓ Timers
    - ✓ UARTs, BTs
    - ✓ CLCD
    - ✓ IR Remote Controller
  - C++: Open & Extendable
    - ✓ Different from Arduino
      - Atmel Studio 6
      - And Nothing in Toolkit
    - ✓ Closer to Industry

JMOD-128-BASE Hardware Platform

June 24, 2015

SDT Lab
Incheon National University

ATmega128 Extendable Module



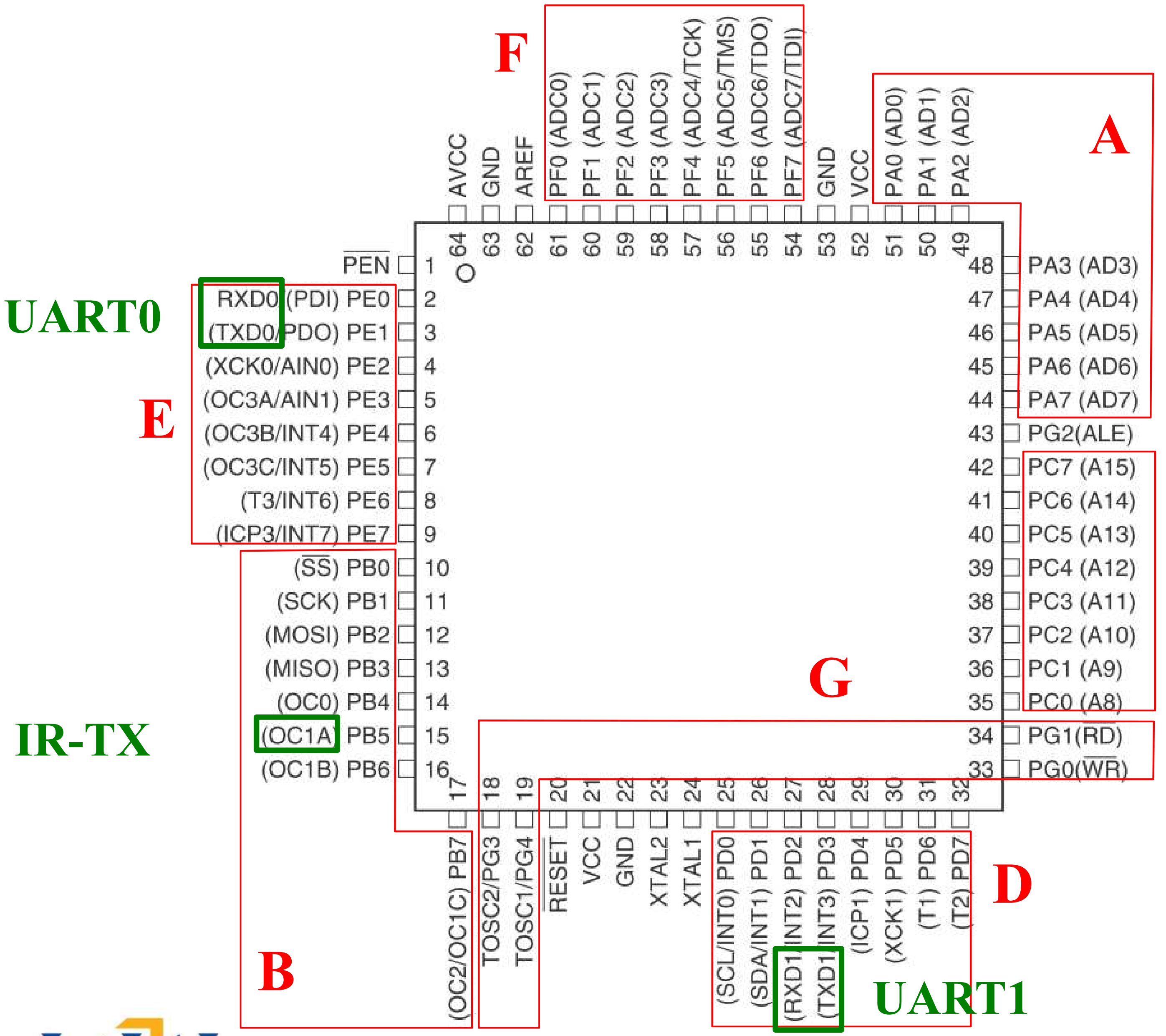
Extension Guideline

Unit	Function
USB-Micro 커넥터	JMOD-128-BASE와 PC와의 연결 커넥터 (전원, 다운로드, 직렬 통신)
전원선택 스위치	위쪽 (V-USB)으로 위치 시 USB에서 전원 (+5V)이 공급되며, 아래쪽 (V-EXT)으로 위치 시, DC 전원 잭 또는 DC 전원 커넥터에서 전원이 공급됨
DC 전원 잭 / 커넥터	6~12V DC 전원 입력으로 어댑터 등 연결 가능, DC 출력으로도 사용 가능
5V 전원 커넥터	5V 전원 입력으로 외부 전원 직접 연결 가능, 5V 출력으로도 사용 가능
ISP/UART 선택 스위치	프로그램 퓨징 시에는 위쪽 (ISP)으로 위치시키고, 프로그램 후 USB를 시리얼 포트 용도로 사용할 때는 아래쪽(UART0)로 위치시킴
블루투스 포트 선택 스위치	아래쪽(UART0)으로 위치 시 블루투스가 UART0 포트 쪽에 연결되고, 위쪽 (UART1)으로 위치 시 블루투스가 UART1 포트 쪽에 연결됨
블루투스 커넥터	블루투스 시리얼모듈 (JMOD-BT-1) 장착을 위한 커넥터
리셋 스위치	누를 시, 리셋 신호를 발생시켜 ATmega128를 초기화 함
포트 A, B, C, D, E, F, G	ATmega128의 해당 포트 (PA, PB, PC, PD, PE, PF, PG), 각 포트는 10핀으로 구성되며 이중 8핀은 포트 신호, 나머지 2핀은 +5V와 GND핀이 할당됨, (단, PG는 PG 0~4만 포트 신호로 사용하며 나머지 3핀은 내부 신호로 사용되므로 주의!)
CLCD 커넥터	Character LCD (문자 LCD) 연결을 위한 커넥터 (PORTA, PORTG)
CLCD 밝기 조절용 저항	CLCD 문자의 밝기를 조절하기 위한 가변 저항

Summary: Switch Combinations

기능 / 스위치	Download		USB 통신	Bluetooth 통신	
	유선	무선	UART0 포트	UART0 포트	UART1 포트
ISP/UART 선택 스위치	ISP	ISP	UART	UART	UART
Bluetooth 선택 스위치	UART1	UART0	UART1	UART0	UART1

Summary: I/O Ports

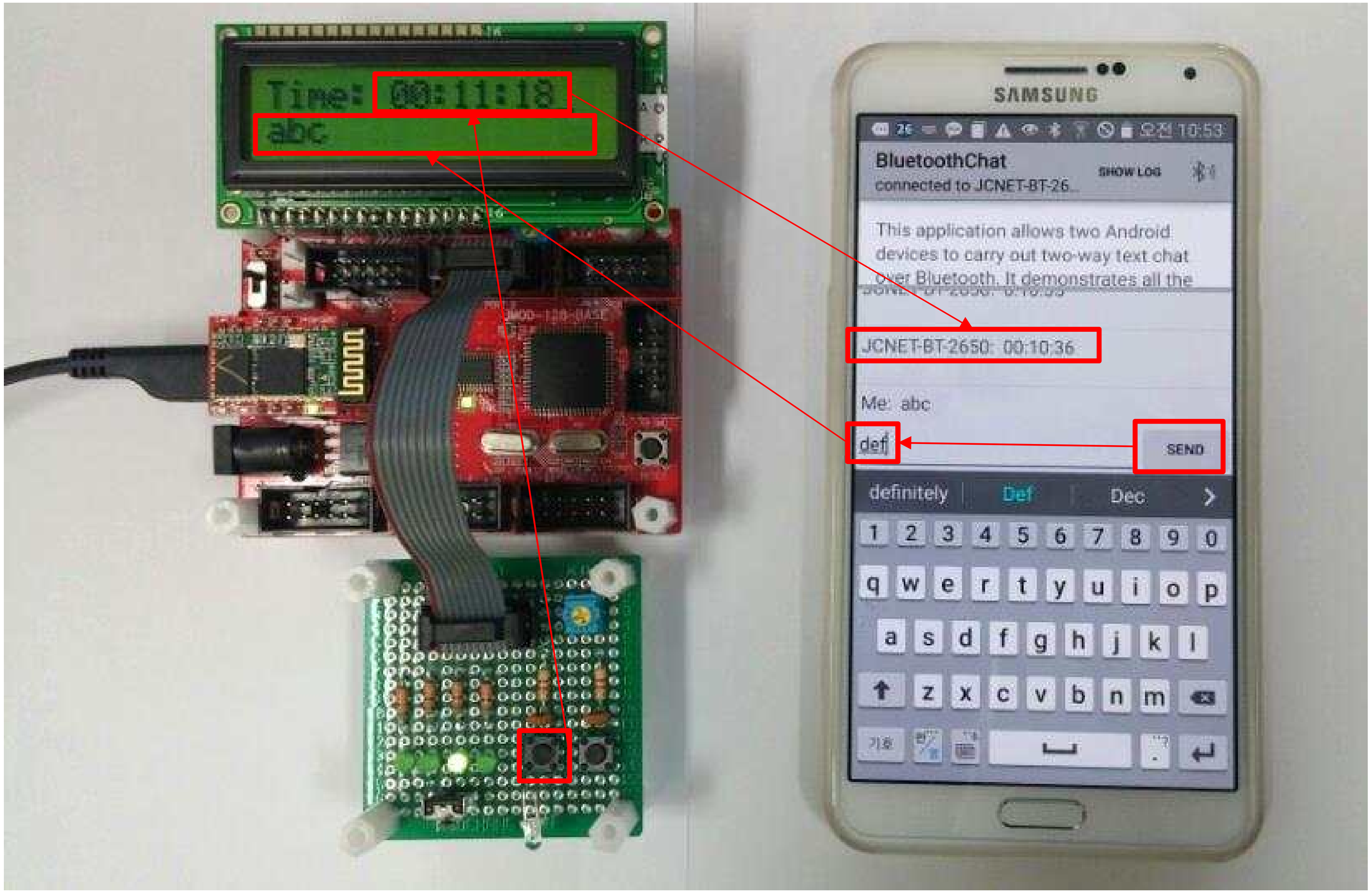


PWR/GND
Clock

Address Bus
Data Bus
Control Bus

GPIOs
INTs
Timer
ADC
SCI (UARTs)

Experimental Setup


Time: 00:11:18
abc
SAMSUNG
BluetoothChat
connected to JCNET-BT-26...
SHOW LOG
This application allows two Android devices to carry out two-way text chat over Bluetooth. It demonstrates all the
JCNET-BT-2650: 00:10:36
Me: abc
def
SEND
definitely
Def
Dec

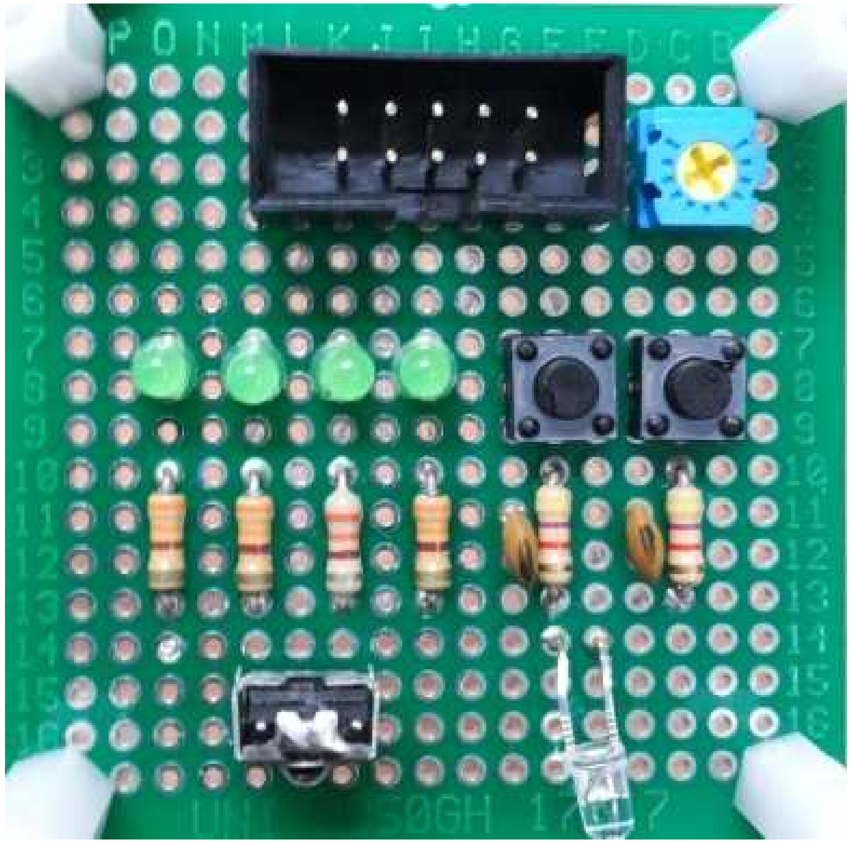
Infra-Red Communication Kit

January 20, 2015

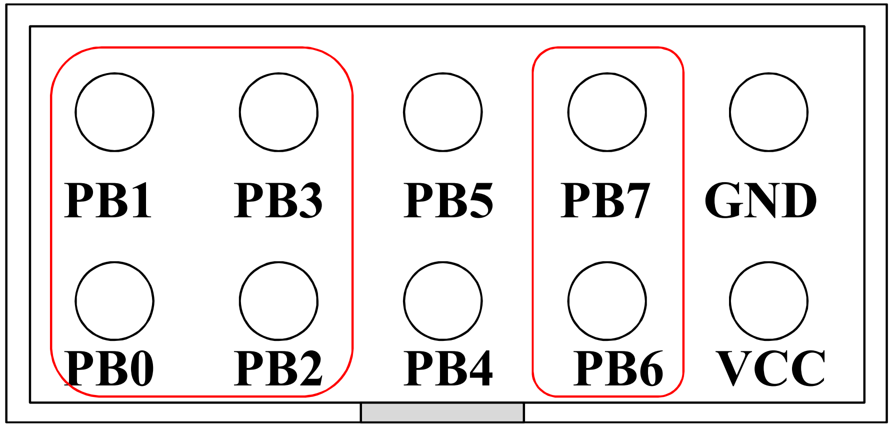
Incheon National University

Parts

- 양면기판 17×17
  - 10-PIN 박스헤더
- LED(녹) 4개
  - 330Ω 저항 4개
- 푸시 스위치(소형) 2개
  - 4.7Ω 저항 2개
  - 22pF 커패시터 2개
- 적외선 이미터 OED-EL-8L 송신부 1개
  - 가변저항 (0~500Ω) 1개
- KSM-603LM 적외선 수광부 1개

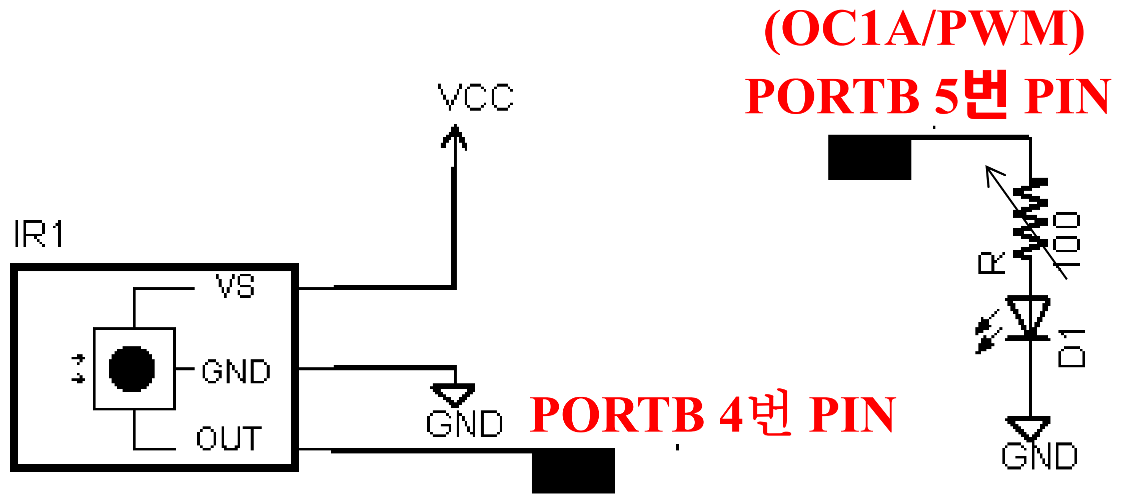


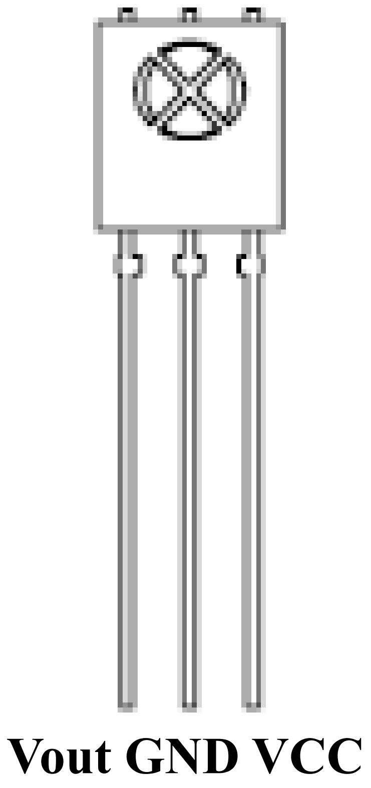
Port To 10-Pin Box Head Connection



- PORTB의 0~3번 PIN
  - LED 1, 2, 3, 4
- PORTB의 4번 PIN
  - KSM-603LM 트랜지스터 입력
- PORTB의 5번 PIN
  - OED-EL-8L 출력
  - Timer/Counter 1 - PWM
  - PB5/OC1A
    - ✓ PE3/OC3A
- PORTB의 6, 7번 PIN
  - Switch 1, 2

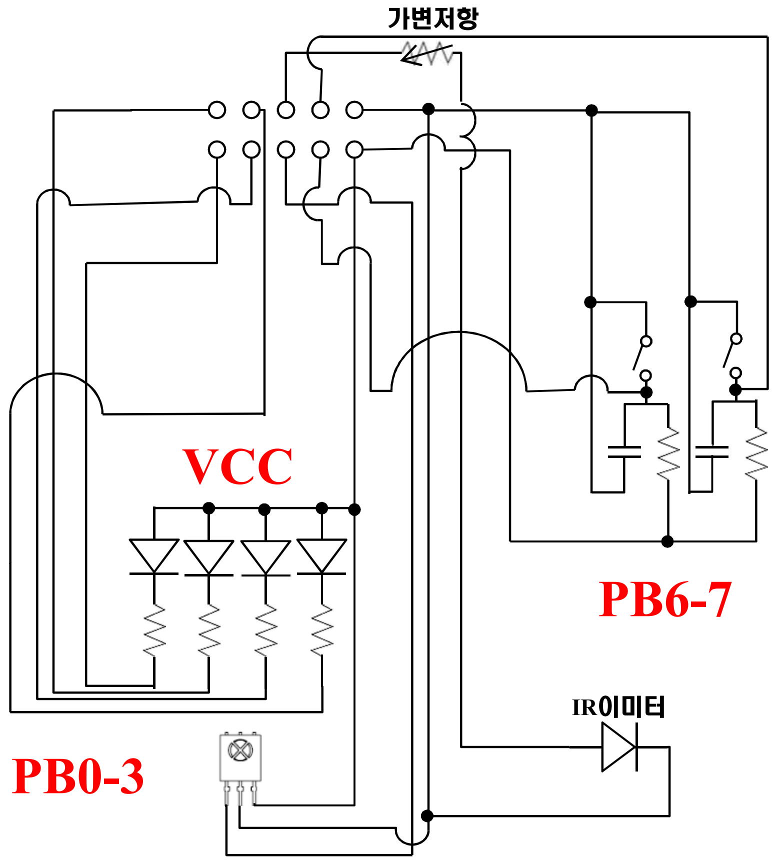
Infra-Red Transmitter & Receiver



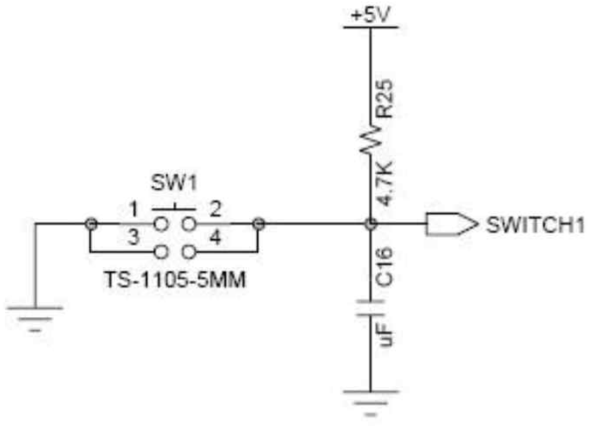


- ❑ **적외선 송수신부 회로 연결 방법**
  - ◆ 왼쪽 트랜지스터는 오른쪽 그림의 핀맵을 보고 연결
  - ◆ IR LED의 100Ω 저항은 가변저항으로 100Ω을 맞춘다.(적외선의 강도를 조절하기 위해 가변저항 이용)

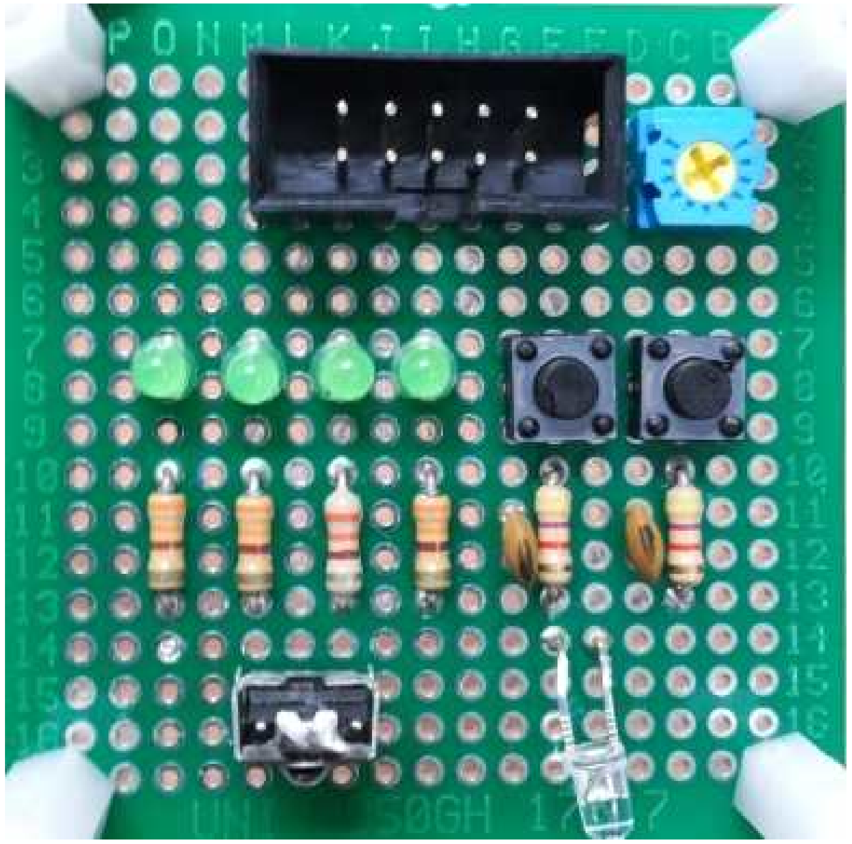
Circuit Diagram

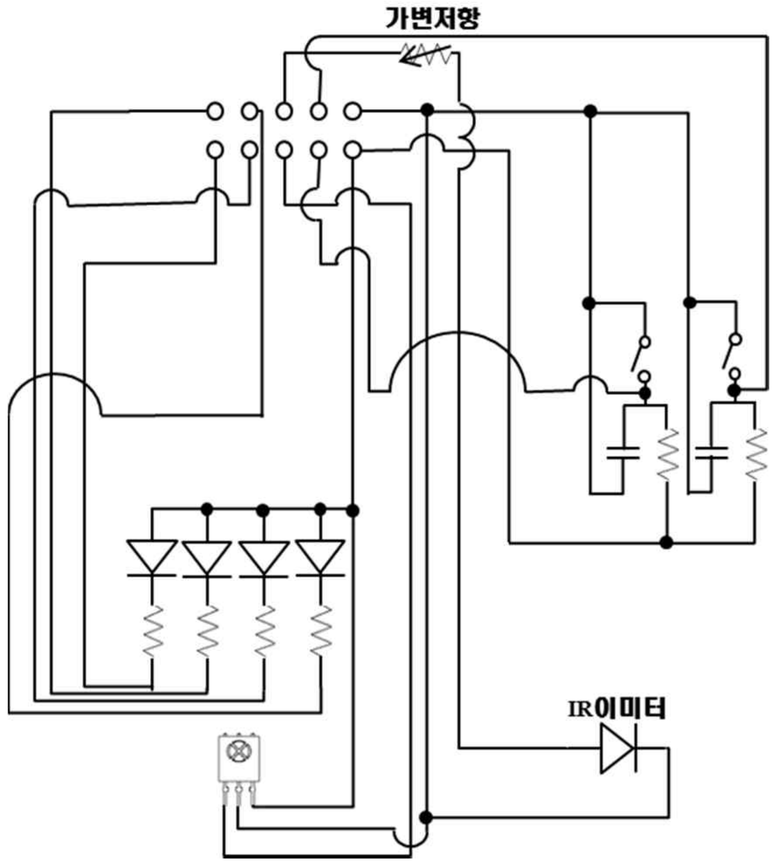


- 왼쪽 4개의 LED(녹) 아래 330Ω 4개가 연결 되어있다.
- 오른쪽 2개의 스위치 아래에 2개의 4.7㏀ 저항과 2개의 22㎌ 커패시터가 연결 되어있다.
  - 스위치의 연결은 조금 복잡하게 되어있는데 아래 회로와 같이 연결 되어있다.

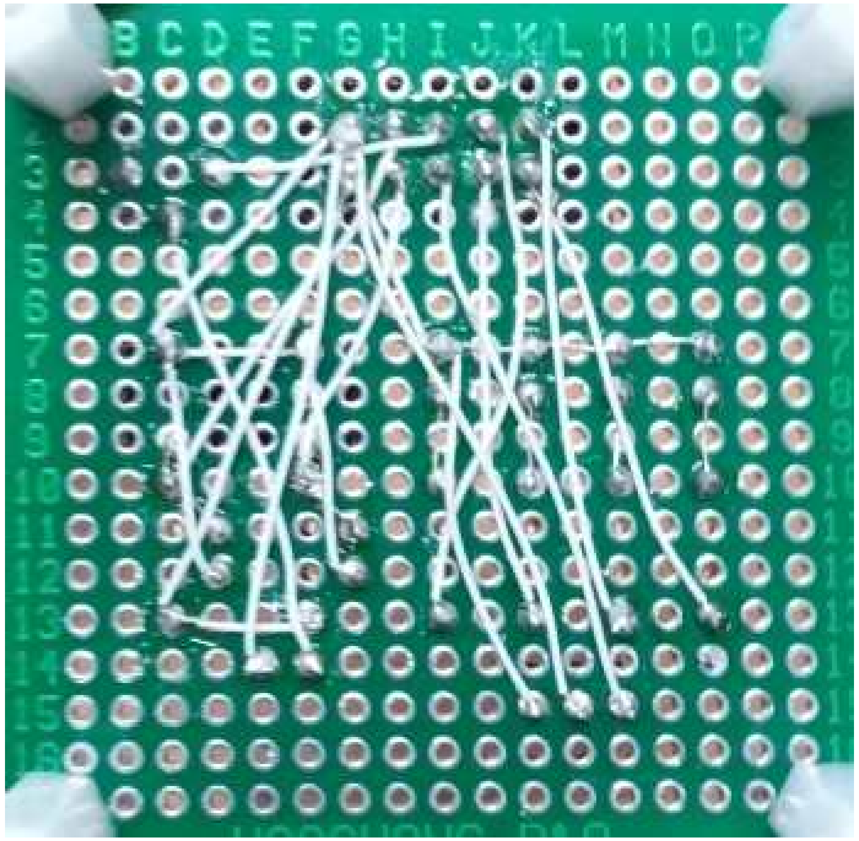


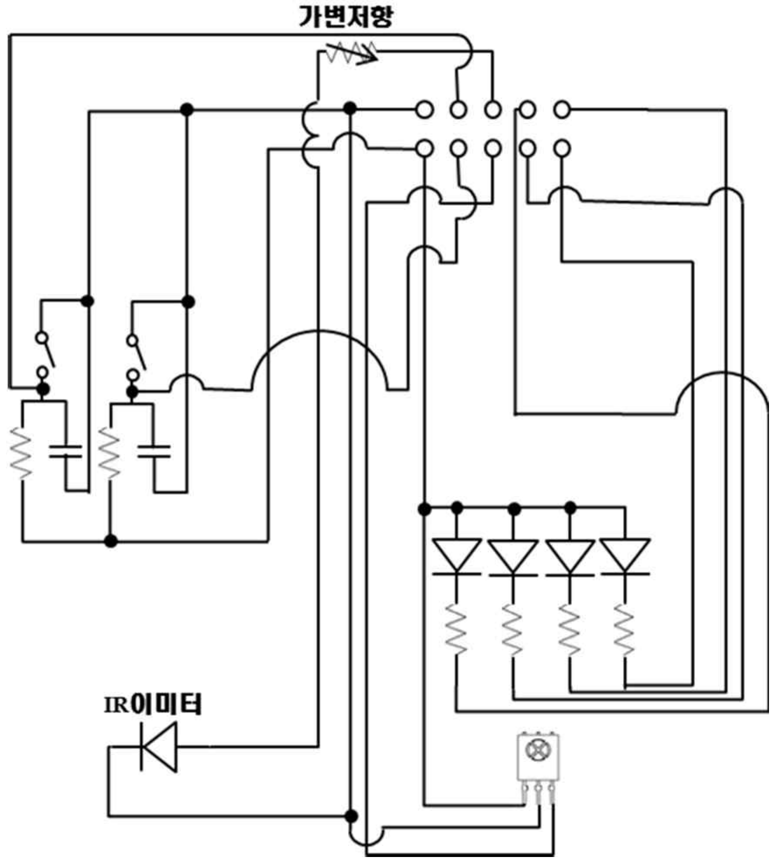
Front Side





Back Side





16-Bit Timer/Counter (x = 1, 3)

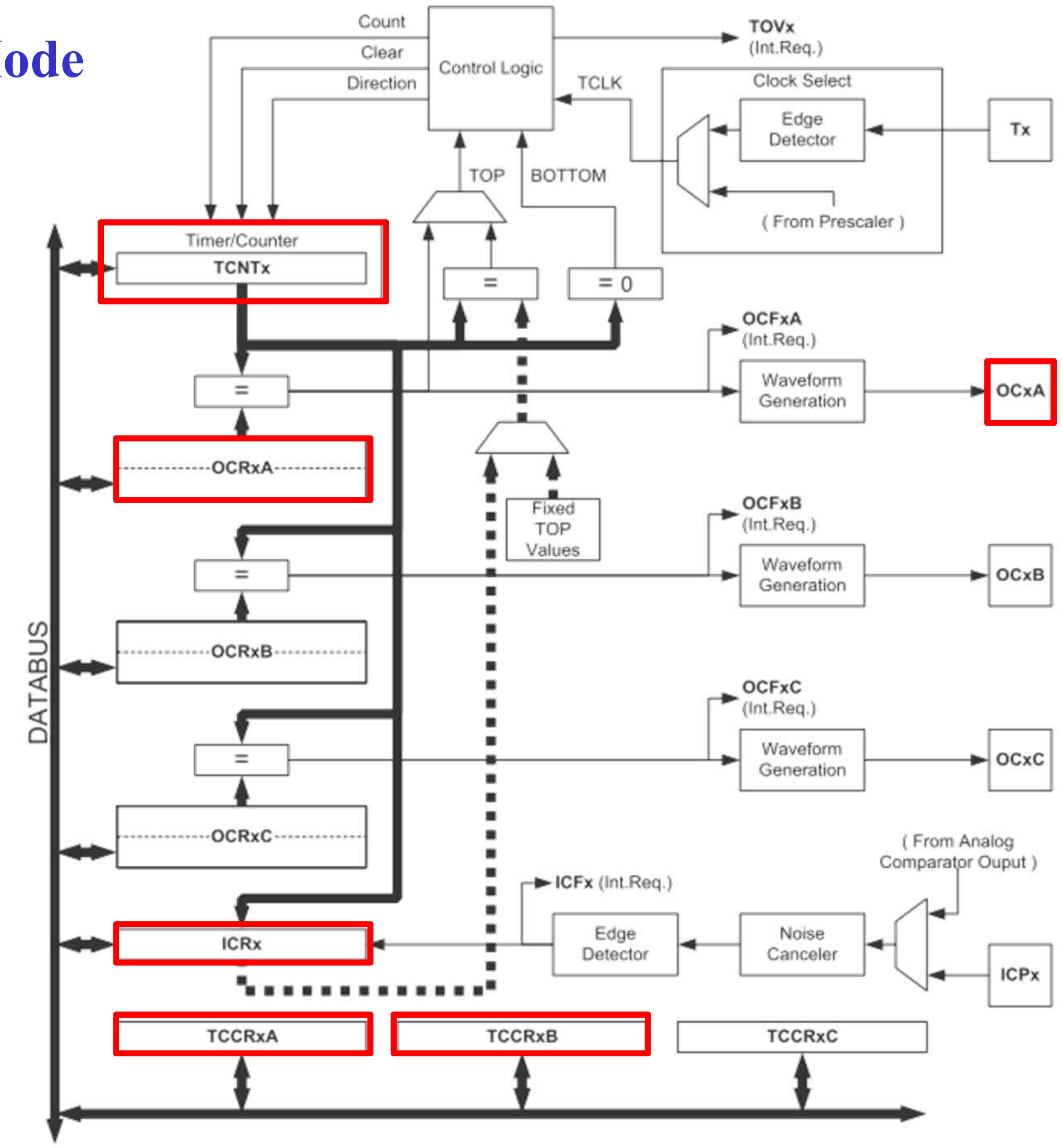
Phase Correct PWM Mode

Toggle Point

Top Count

Control Registers

PWM

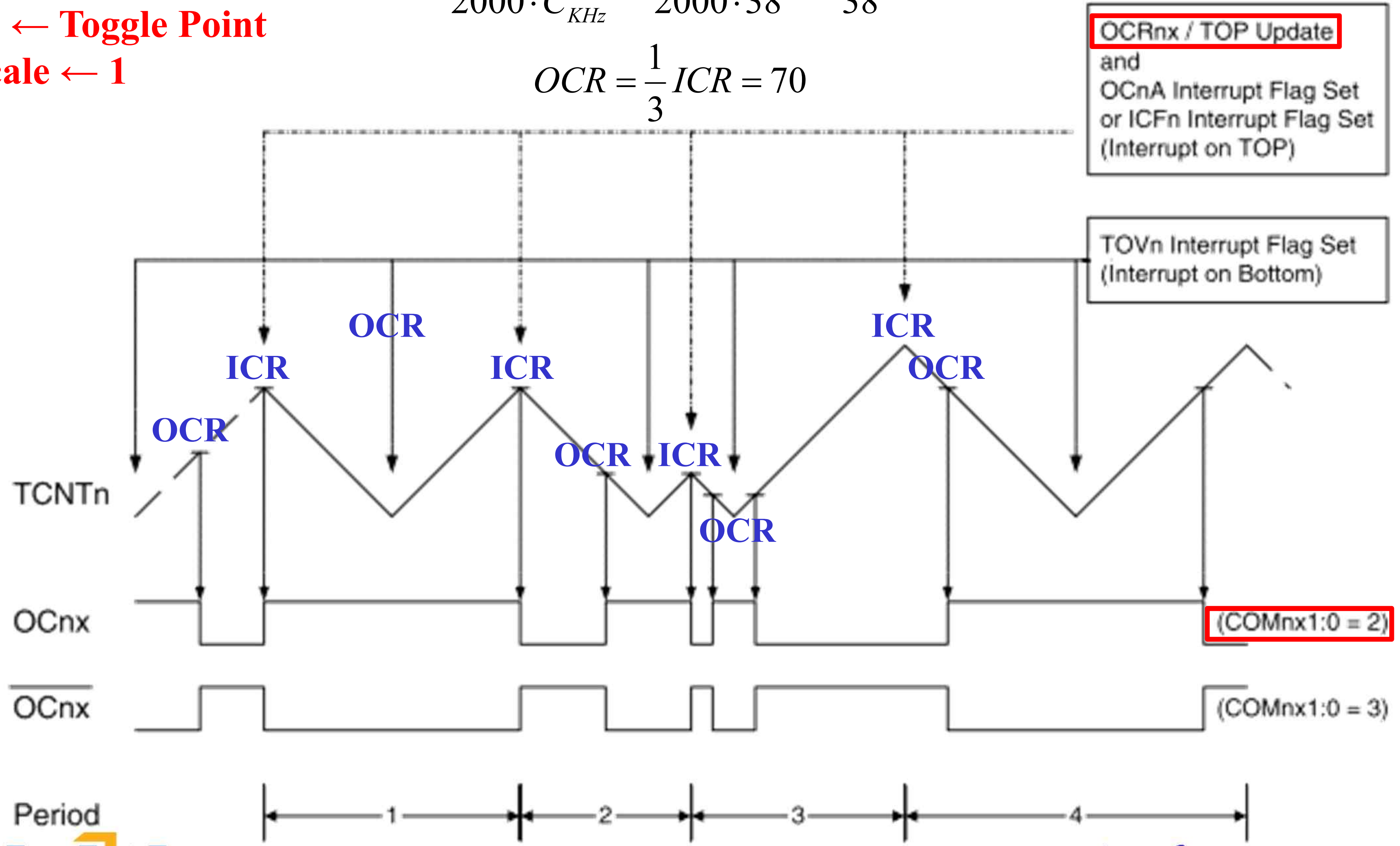


Phase Correct PWM Mode

ICRn ← Top Count
OCRn ← Toggle Point
Pre-Scale ← 1

$$ICR = \frac{f_{CPU}}{2000 \cdot C_{KHz}} = \frac{16MHz}{2000 \cdot 38} = \frac{8000}{38} = 210$$

$$OCR = \frac{1}{3} ICR = 70$$



SDT Lab

Timer/Counter Control Registers

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation[1]	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	TOP	TOP
15	1	1	1	1	Fast PWM	OCRnA	TOP	TOP

Timer/Counter Control Registers

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}/1$ (No prescaling
0	1	0	$clk_{I/O}/8$ (From pres
0	1	1	$clk_{I/O}/64$ (From pre
1	0	0	$clk_{I/O}/256$ (From p
1	0	1	$clk_{I/O}/1024$ (From
1	1	0	External clock sou
1	1	1	External clock sou

COMnA1/COMnB/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	WGMn3=0: Normal port operation, OCnA/OCnB/OCnC disconnected. WGMn3=1: Toggle OCnA on compare match, OCnB/OCnC reserved.
1	0	Clear OCnA/OCnB/OCnC on compare match when up-counting. Set OCnA/OCnB/OCnC on compare match when downcounting.
1	1	Set OCnA/OCnB/OCnC on compare match when up-counting. Clear OCnA/OCnB/OCnC on compare match when downcounting.

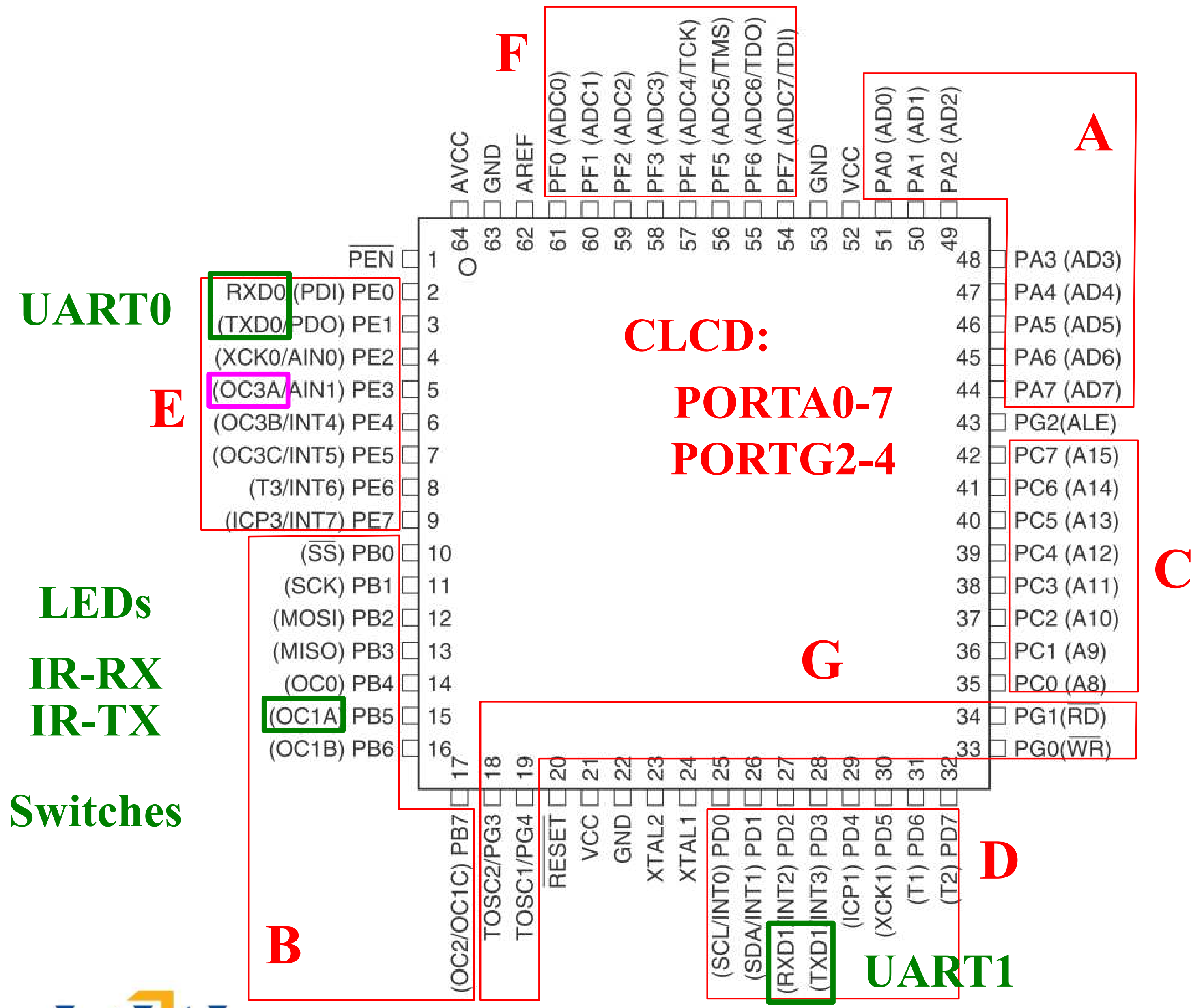
C++ Object Toolkit

June 24, 2015

SDT Lab

Incheon National University

Example: Usage of I/O Ports



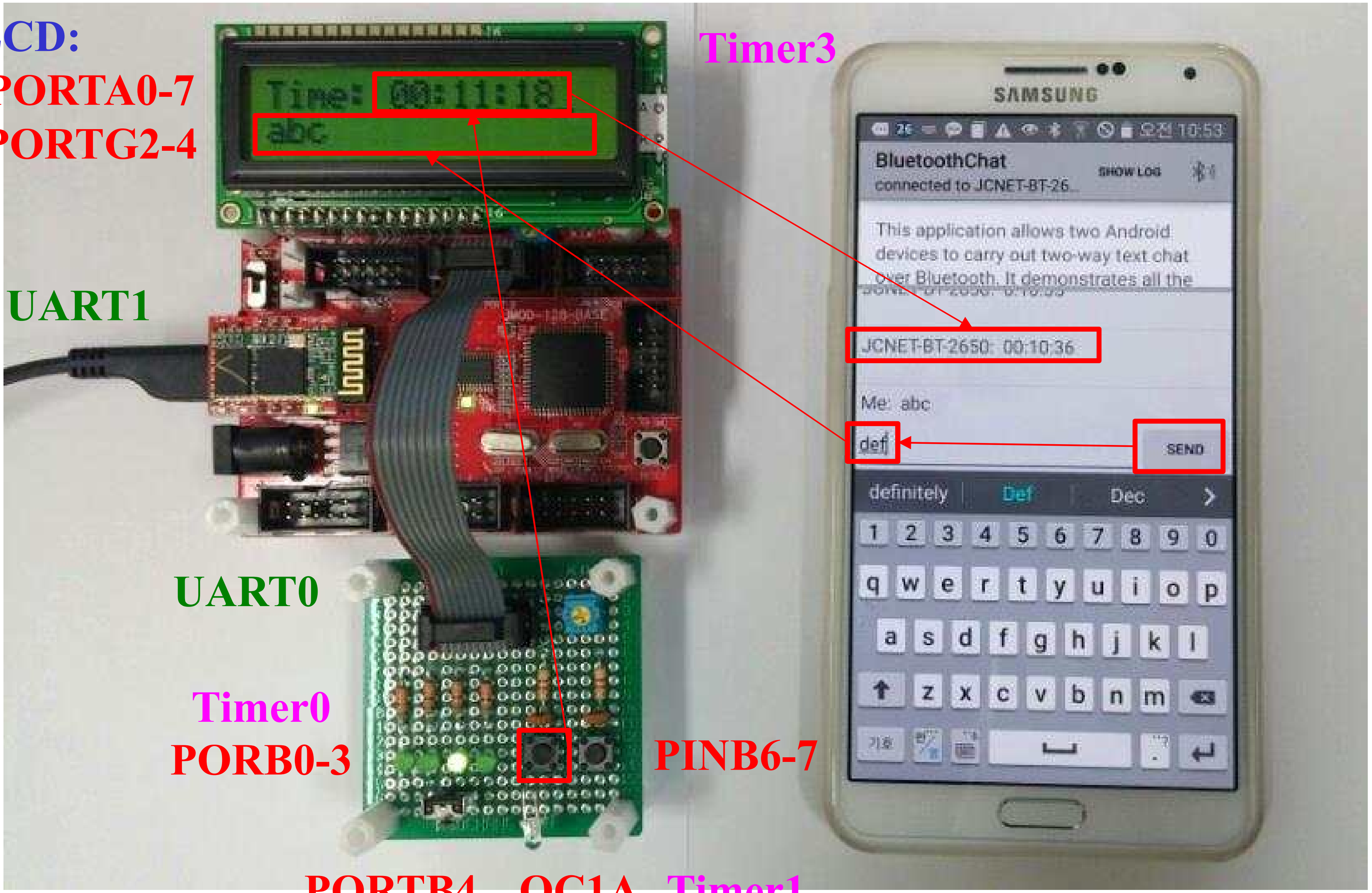
PWR/GND
Clock

Address Bus
Data Bus
Control Bus

GPIOs
INTs
Timer
ADC
SCI (UARTs)

SDT Lab

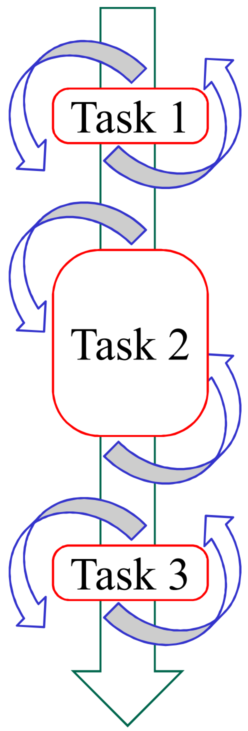
Example: Hardware Configuration

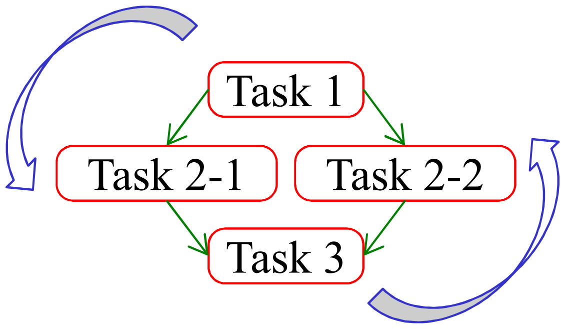


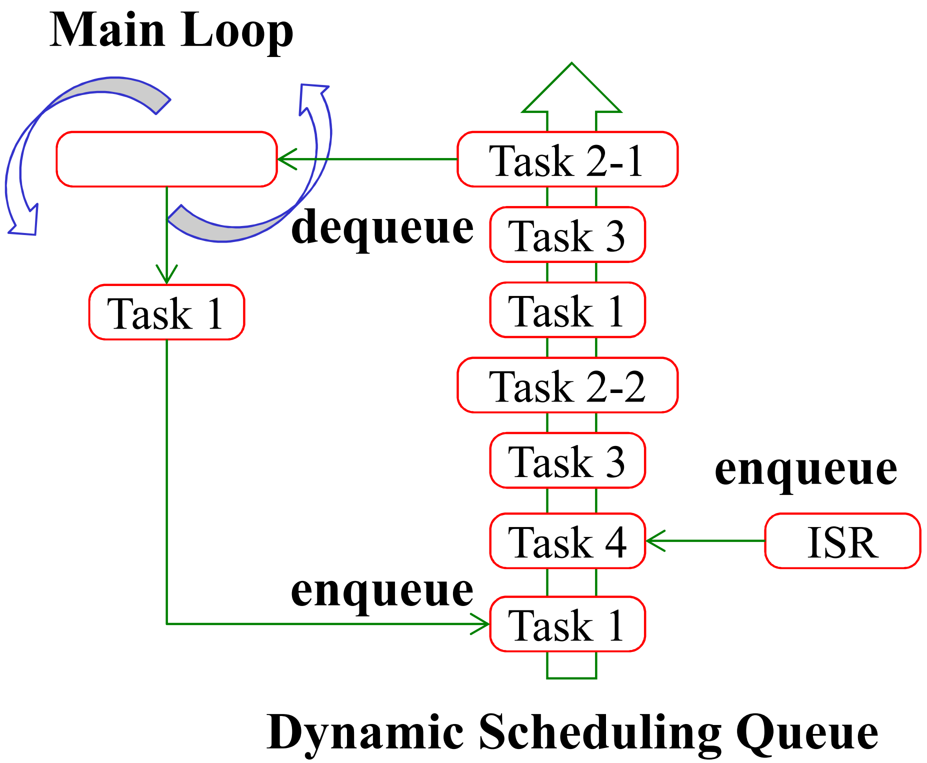
This application allows two Android devices to carry out two-way text chat over Bluetooth. It demonstrates all the

Process Control

- Two Types of Threads: Main Loop & ISRs
  - Main Loop – Scheduled Tasks (Synchronous)
    - ✓ Tasks are executed sequentially according to a schedule.
    - ✓ A time consuming task can be split into short tasks to enable other short tasks to be interleaved.

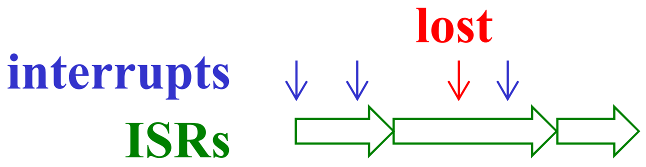




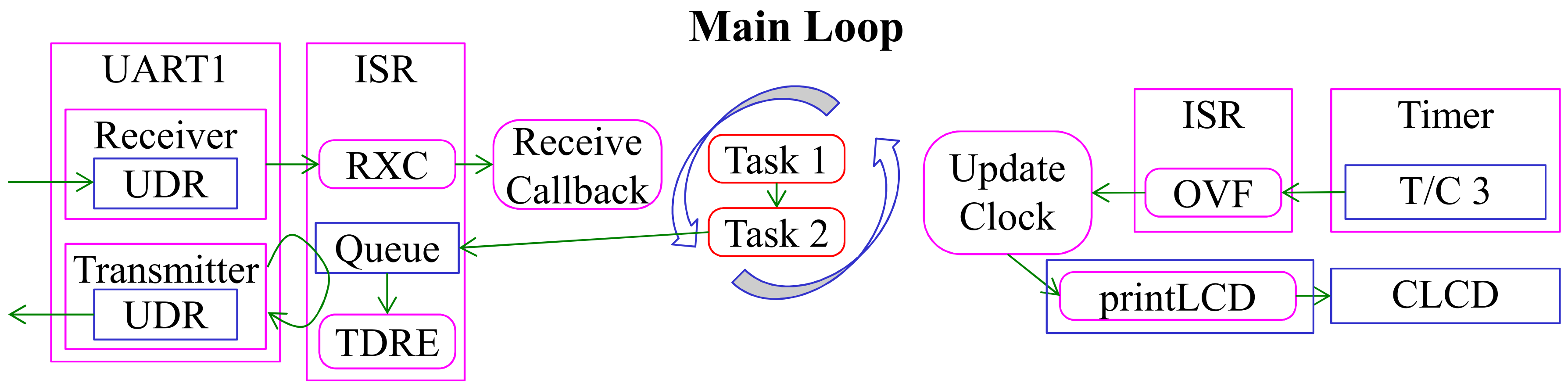


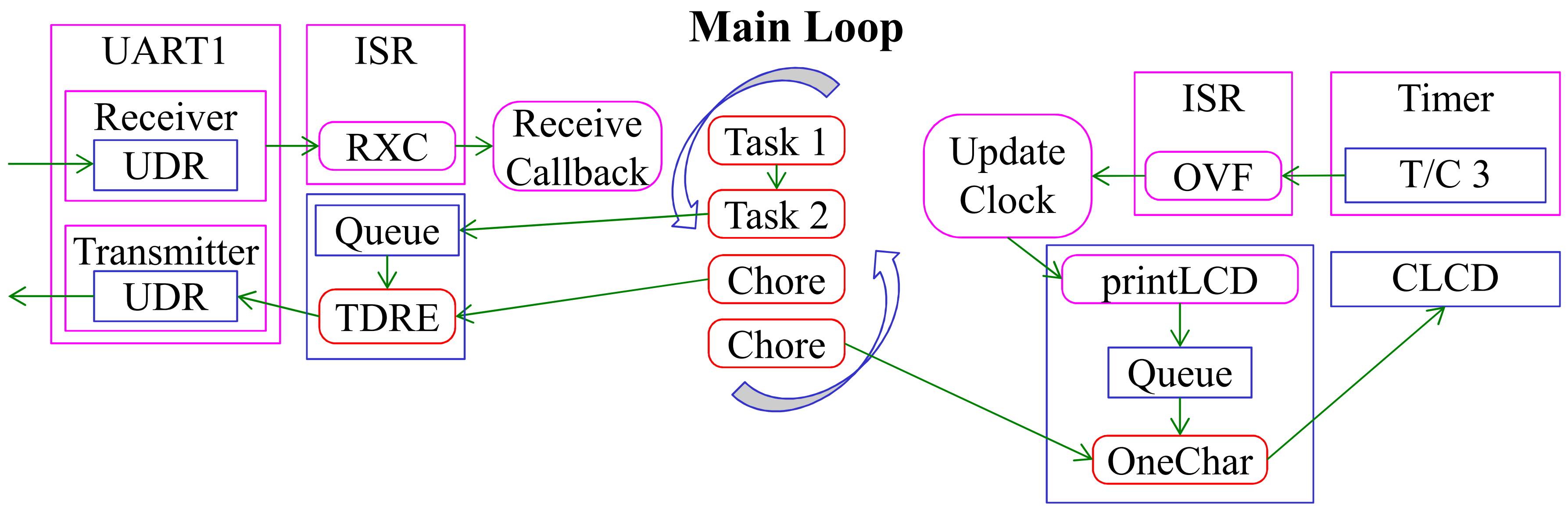
Process Control

- Two Types of Threads: Main Loop & ISRs (Cont'd)
  - Interrupt Service Routines (ISRs)
    - ✓ ISRs are all urgent, and executed asynchronously.
    - ✓ Interrupts are automatically disabled and enabled before and after ISR calls, respectively.
    - ✓ ISRs should be brief. Otherwise, some interrupts are lost.
    - ✓ Schedule non-urgent callbacks as tasks in the main loop.
    - ✓ Separate time consuming parts of ISRs if any, and schedule them as tasks in the main loop.



Process Control



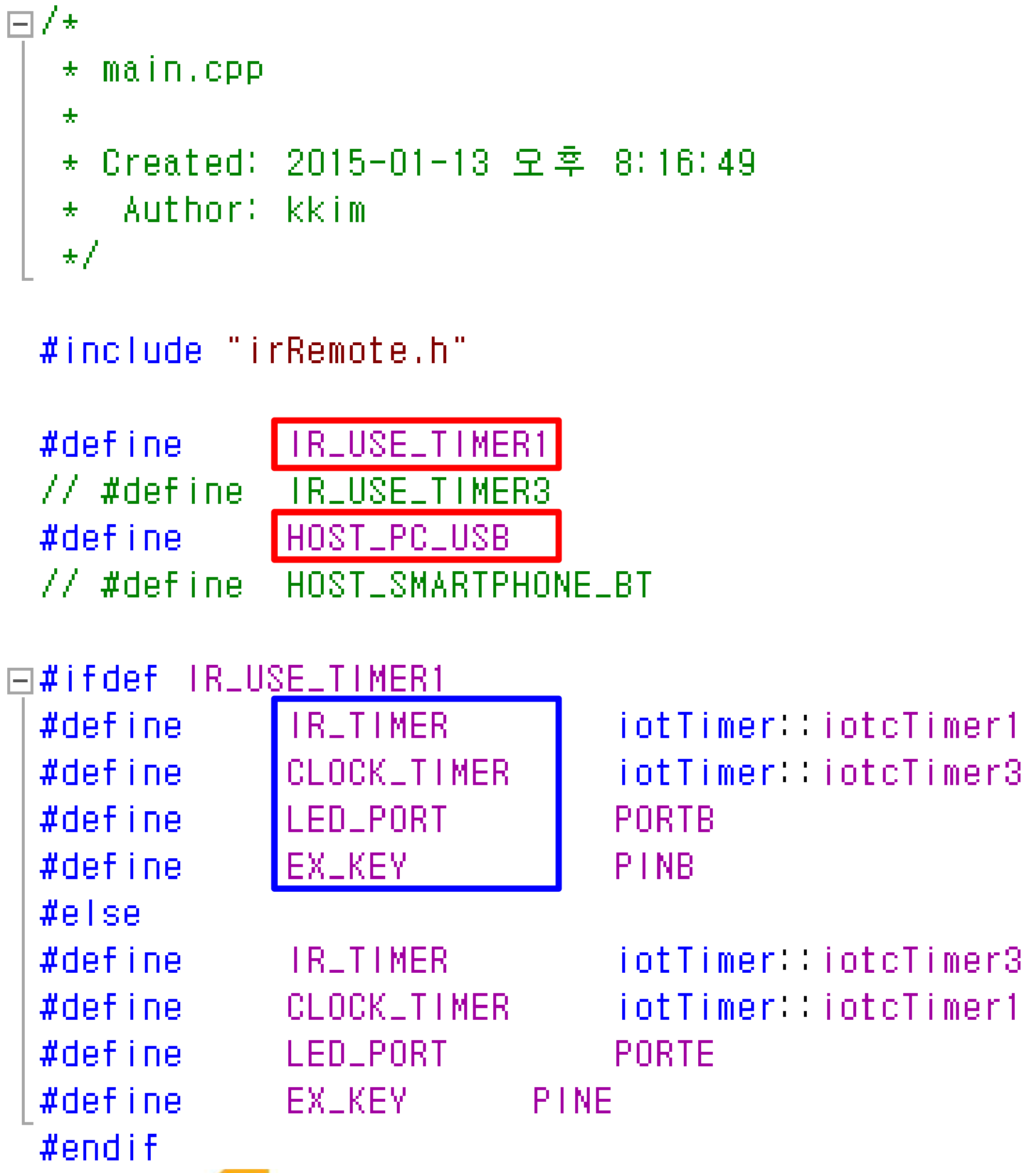


Process Control

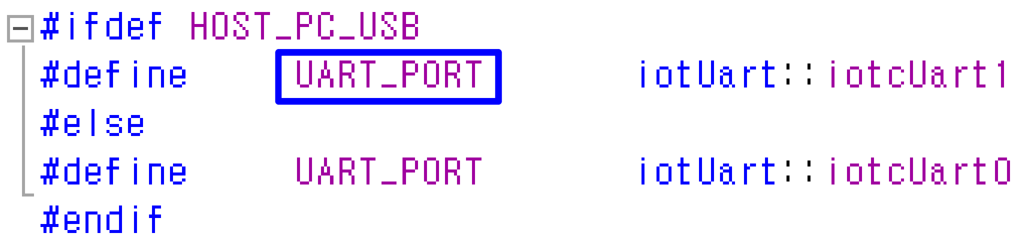
- It's Racing Condition Free!
  - No Racing Conditions
    - ✓ Between Tasks in Main Loop (Scheduled to be done Sequentially)
    - ✓ Between ISRs (Interrupts are disabled in ISRs)
  - Between Tasks & ISRs
    - ✓ Racing Conditions are Prevented by Atomic Operations
    - ✓ Users' Responsibility.
  - Synchronization Overdone!
    - ✓ Irrelevant (Sharing No Variables) Tasks & ISRs are Also Synchronized.

- In Java Programming
  - Numerous & More Flexible Synchronization Techniques are Available.

main.cpp: Configuration

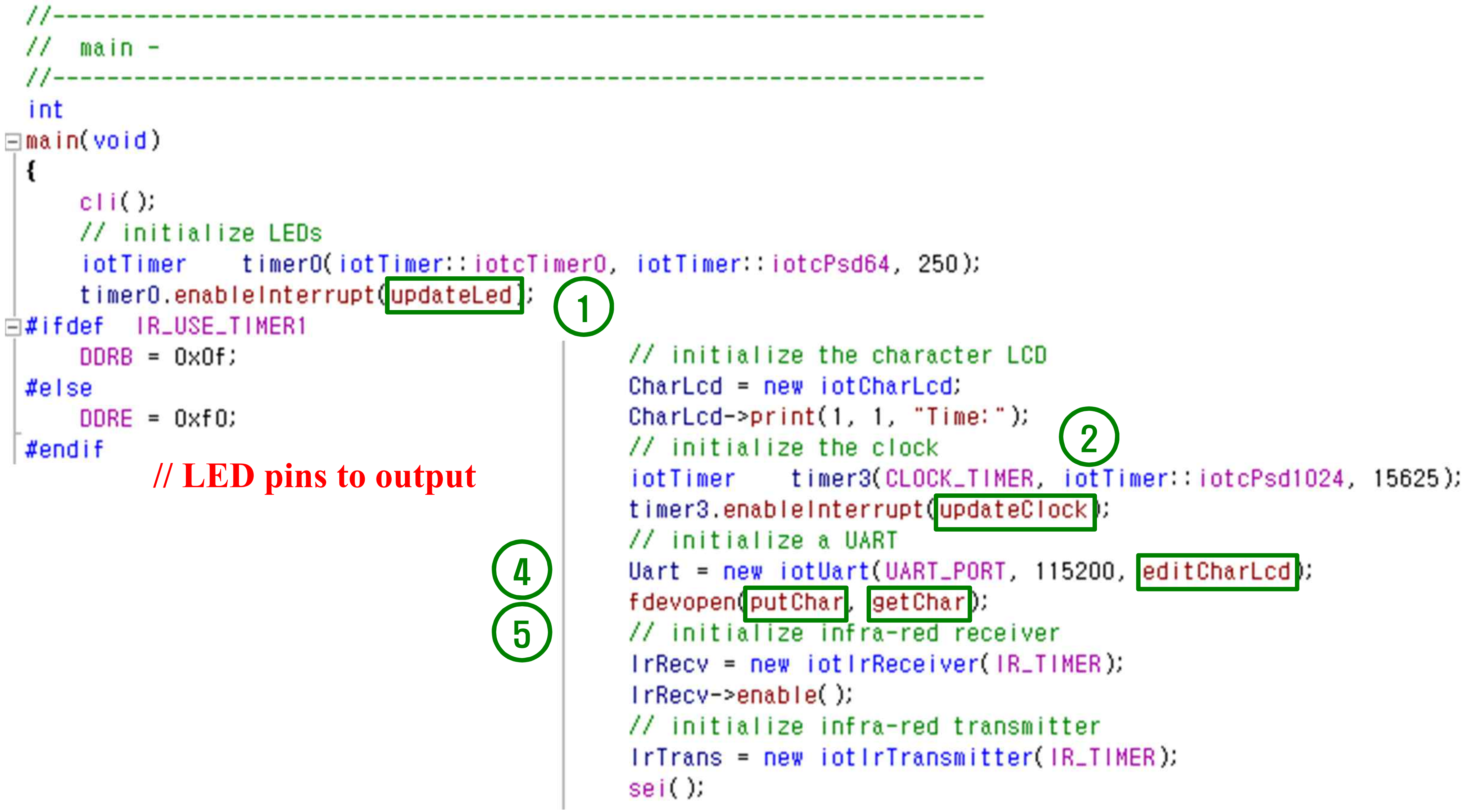


```
/*
 * main.cpp
 *
 * Created: 2015-01-13 오후 8:16:49
 *  Author: kkim
 */

#include "irRemote.h"

#define     IR_USE_TIMER1
// #define  IR_USE_TIMER3
#define     HOST_PC_USB
// #define  HOST_SMARTPHONE_BT

#ifdef IR_USE_TIMER1
#define     IR_TIMER        iotTimer::iotcTimer1
#define     CLOCK_TIMER     iotTimer::iotcTimer3
#define     LED_PORT        PORTB
#define     EX_KEY          PINB
#else
#define     IR_TIMER        iotTimer::iotcTimer3
#define     CLOCK_TIMER     iotTimer::iotcTimer1
#define     LED_PORT        PORTE
#define     EX_KEY      PINE
#endif
```



```
#ifdef HOST_PC_USB
#define     UART_PORT       iotUart::iotcUart1
#else
#define     UART_PORT       iotUart::iotcUart0
#endif
```

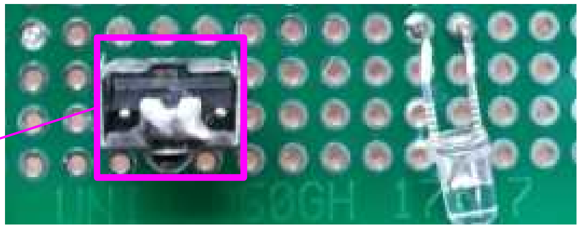
main.cpp: main

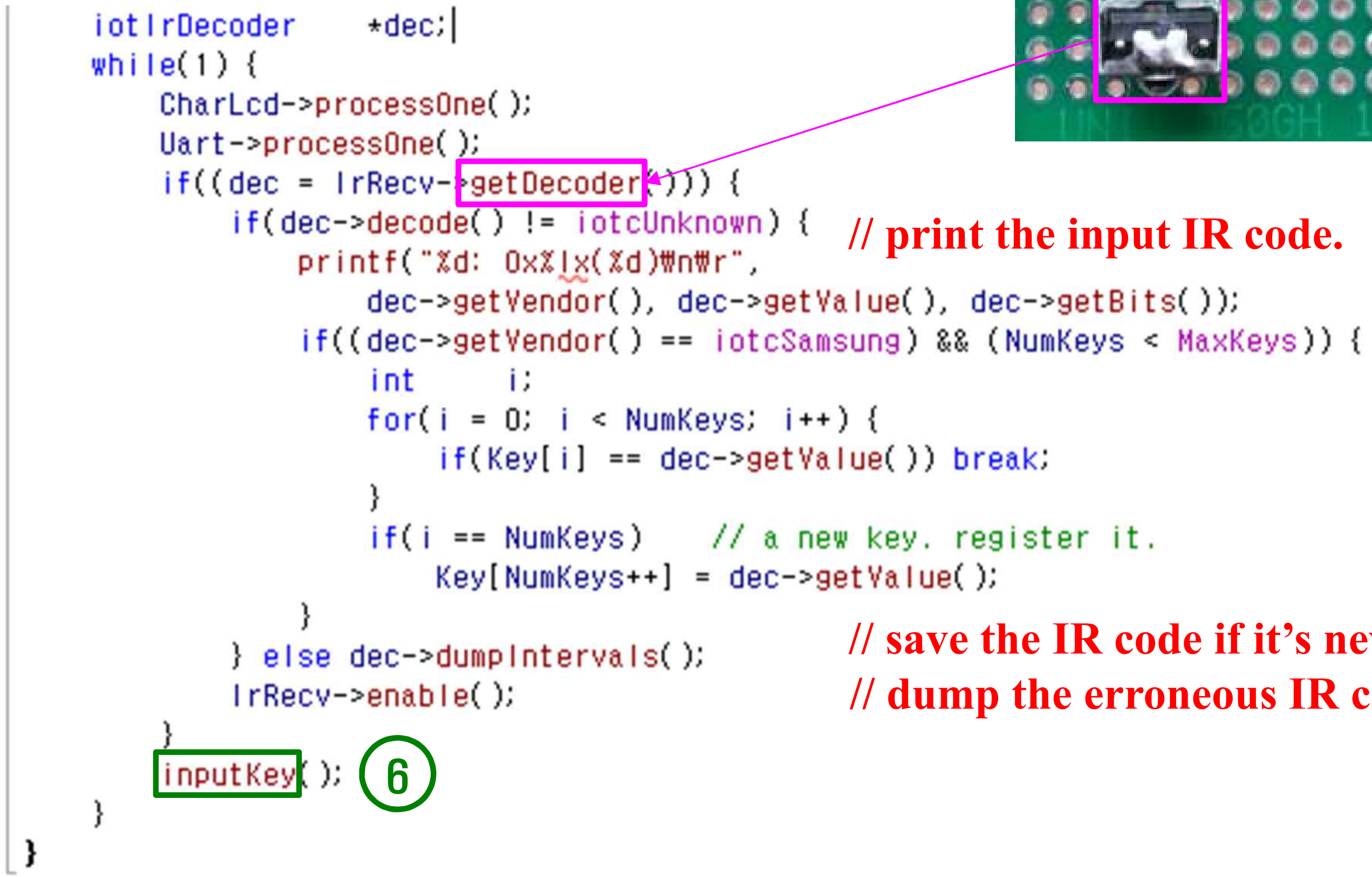


```
//-----------------------------------------------------------------
//  main -
//-----------------------------------------------------------------
int
main(void)
{
    cli();
    // initialize LEDs
    iotTimer    timer0(iotTimer::iotcTimer0, iotTimer::iotcPsd64, 250);
    timer0.enableInterrupt(updateLed);
#ifdef  IR_USE_TIMER1
    DDRB = 0x0f;
#else
    DDRE = 0xf0;
#endif
    // initialize the character LCD
    CharLcd = new iotCharLcd;
    CharLcd->print(1, 1, "Time:");
    // initialize the clock
    iotTimer    timer3(CLOCK_TIMER, iotTimer::iotcPsd1024, 15625);
    timer3.enableInterrupt(updateClock);
    // initialize a UART
    Uart = new iotUart(UART_PORT, 115200, editCharLcd);
    fdevopen(putChar, getChar);
    // initialize infra-red receiver
    IrRecv = new iotIrReceiver(IR_TIMER);
    IrRecv->enable();
    // initialize infra-red transmitter
    IrTrans = new iotIrTransmitter(IR_TIMER);
    sei();
```

SDT Lab

main.cpp: main (Cont'd)





```
    iotIrDecoder     *dec;
    while(1) {
        CharLcd->processOne();
        Uart->processOne();
        if((dec = IrRecv->getDecoder())) {
            if(dec->decode() != iotcUnknown) {
                printf("%d: 0x%lx(%d)₩n₩r",
                    dec->getVendor(), dec->getValue(), dec->getBits());
                if((dec->getVendor() == iotcSamsung) && (NumKeys < MaxKeys)) {
                    int      i;
                    for(i = 0; i < NumKeys; i++) {
                        if(Key[i] == dec->getValue()) break;
                    }
                    if(i == NumKeys)    // a new key. register it.
                        Key[NumKeys++] = dec->getValue();
                }
            } else dec->dumpIntervals();
            IrRecv->enable();
        }
        inputKey(); 6
    }
}
```

// print the input IR code.

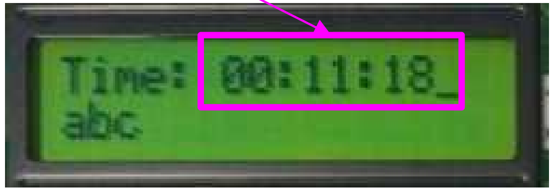
// save the IR code if it's new.

// dump the erroneous IR code.

main.cpp: Clock/LED

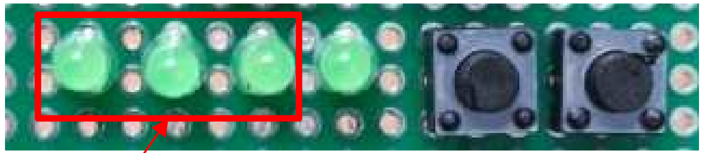
```
//---------------------------------------------------
//  clock -
//---------------------------------------------------
long int     Seconds = 0;
char         HhMmSs[9] = "00:00:00";
iotCharLcd   *CharLcd = (iotCharLcd *) NULL;

void
updateClock()
{
    Seconds++;
    if(Seconds < 0) Seconds = 0;

    HhMmSs[0] = '0' + (Seconds % 86400) / 36000;
    HhMmSs[1] = '0' + (Seconds % 36000) / 3600;
    HhMmSs[3] = '0' + (Seconds % 3600) / 600;
    HhMmSs[4] = '0' + (Seconds % 600) / 60;
    HhMmSs[6] = '0' + (Seconds % 60) / 10;
    HhMmSs[7] = '0' + (Seconds % 10);

    if(CharLcd) CharLcd->print(1, 7, HhMmSs);
}
```

2



```
//-------------------------------------------------------
//  led -
//-------------------------------------------------------
#define     LED_LEFT    0
#define     LED_RIGHT   1

unsigned char   Num1kHzCycles = 0;  // # of 500 Hz clocks
unsigned char   LedData = 1;
unsigned char   LedMode = LED_LEFT;

void
updateLed()
{
    Num1kHzCycles++;
    if(Num1kHzCycles == 200) {     // 0.2 sec
        if(LedMode == LED_LEFT) {
            if(LedData == 0x04) LedData = 0x01;
            else LedData = LedData << 1;
        } else {
            if(LedData == 0x01) LedData = 0x04;
            else LedData = LedData >> 1;
        }
#ifdef  IR_USE_TIMER1
        LED_PORT = ~LedData;
#else
        LED_PORT = ~(LedData << 4);
#endif
        Num1kHzCycles = 0;
    }
}
```

1



main.cpp: UART

```
//----------------------------------------------
//  uart -
//----------------------------------------------
iotUart      *Uart = (iotUart *) NULL;
unsigned     char     CurX = 0;

void
editCharLcd()
{
    if(!Uart || !CharLcd) return;

    int      c;
    while(Uart->peekChar() > -1) {
        c = Uart->getChar();
        if(c == 8) {
            CharLcd->print(2, CurX, 32);
            if(CurX > 0) --CurX;
        } else {
            CharLcd->print(2, ++CurX, c);
            CurX %= iotcCharLcdColumns;
        }
    }
}
```

3

// backspace: erase by printing space

// called by printf ( )

4

```
int
putChar(char c, FILE *f)
{
    if(Uart) return(Uart->putChar(c));
    return(-1);
}
```

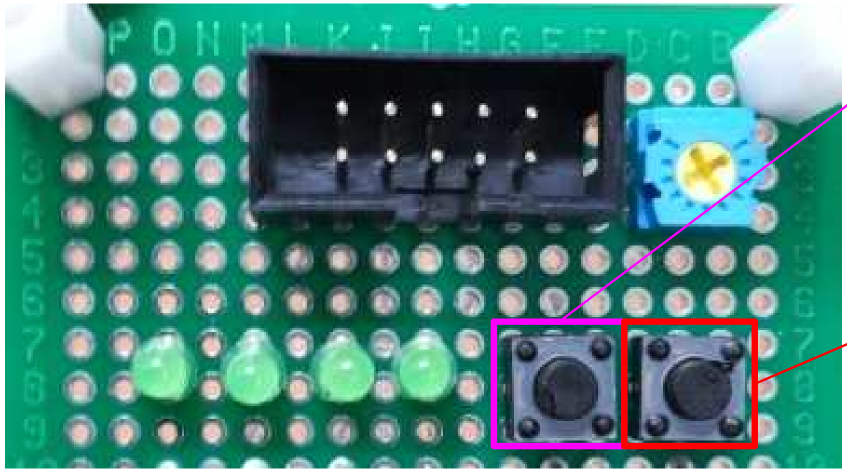
// called by scanf ( )

5

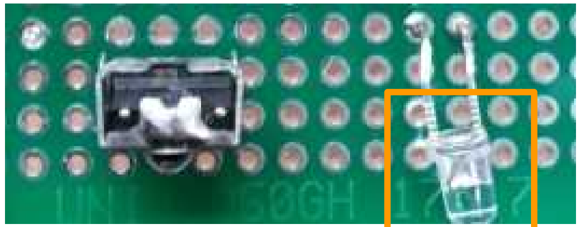
```
int
getChar(FILE *f)
{
    if(Uart) return(Uart->getChar());
    return(-1);
}
```

main.cpp: Switch

```
//---------------------------------------------------
//  switch
//---------------------------------------------------
char            LastKey = 0;

iotIrReceiver         *IrRecv;
iotIrTransmitter      *IrTrans;

#define               MaxKeys      5
#define               NumBits      32
int                   NumKeys = 0;
int                   CurKey = 0;
unsigned long         Key[MaxKeys] = { 0, 0, 0, 0, 0 };
```



```
void
inputKey( )   ⑥
{
    volatile char    key;

#ifdef IR_USE_TIMER1
    key = ~EX_KEY & 0xc0;
#else
    key = ~EX_KEY << 6;
#endif
    if(key == LastKey) return;
    LastKey = key;
    iotCharLcd::delay(200);
    switch(key) {
        case 0x40:
            if(Key[CurKey] == 0) printf("%s\n\r", HhMmSs);
            else {
                IrTrans->sendSamsung(Key[CurKey], NumBits);
                IrRecv->enable( );
            }
            CurKey = (CurKey + 1) % MaxKeys;
            break;
        case 0x80:
            if(LedMode == LED_LEFT) LedMode = LED_RIGHT;
            else LedMode = LED_LEFT;
            break;
        default: break;
    }
}
```



// print date to UART

// send IR code

// redirect LEDs

SDT Lab

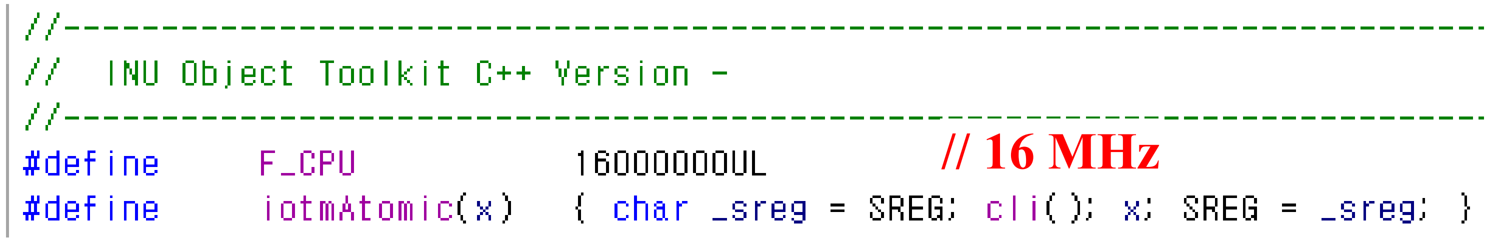
iot.h: C++ Compatibility Code

```
//=====================================================================
// iot.h
//  The header of the iot package.
//      (INU Object Toolkit for the Internet of Things)
//  Created: 2015-01-09 오후 2:33:04
//  Author: kkim@inu.ac.kr / SDTLAB INU (Incheon National University)
//=====================================================================
```

```
#ifndef IOT_H_
#define IOT_H_

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
```

```
//-------------------------------------------------------------------
//  cplusplus -
//-------------------------------------------------------------------

//  This is applicable if using virtual inheritance.
__extension__ typedef int __guard __attribute__((mode (__DI__)));

extern "C" int __cxa_guard_acquire(__guard *);
extern "C" void __cxa_guard_release (__guard *);
extern "C" void __cxa_guard_abort (__guard *);

//  This is applicable if using pure virtual inheritance.
extern "C" void __cxa_pure_virtual(void);

//  Operators required for C++
void *operator new(size_t size);
void operator delete(void *size);
```

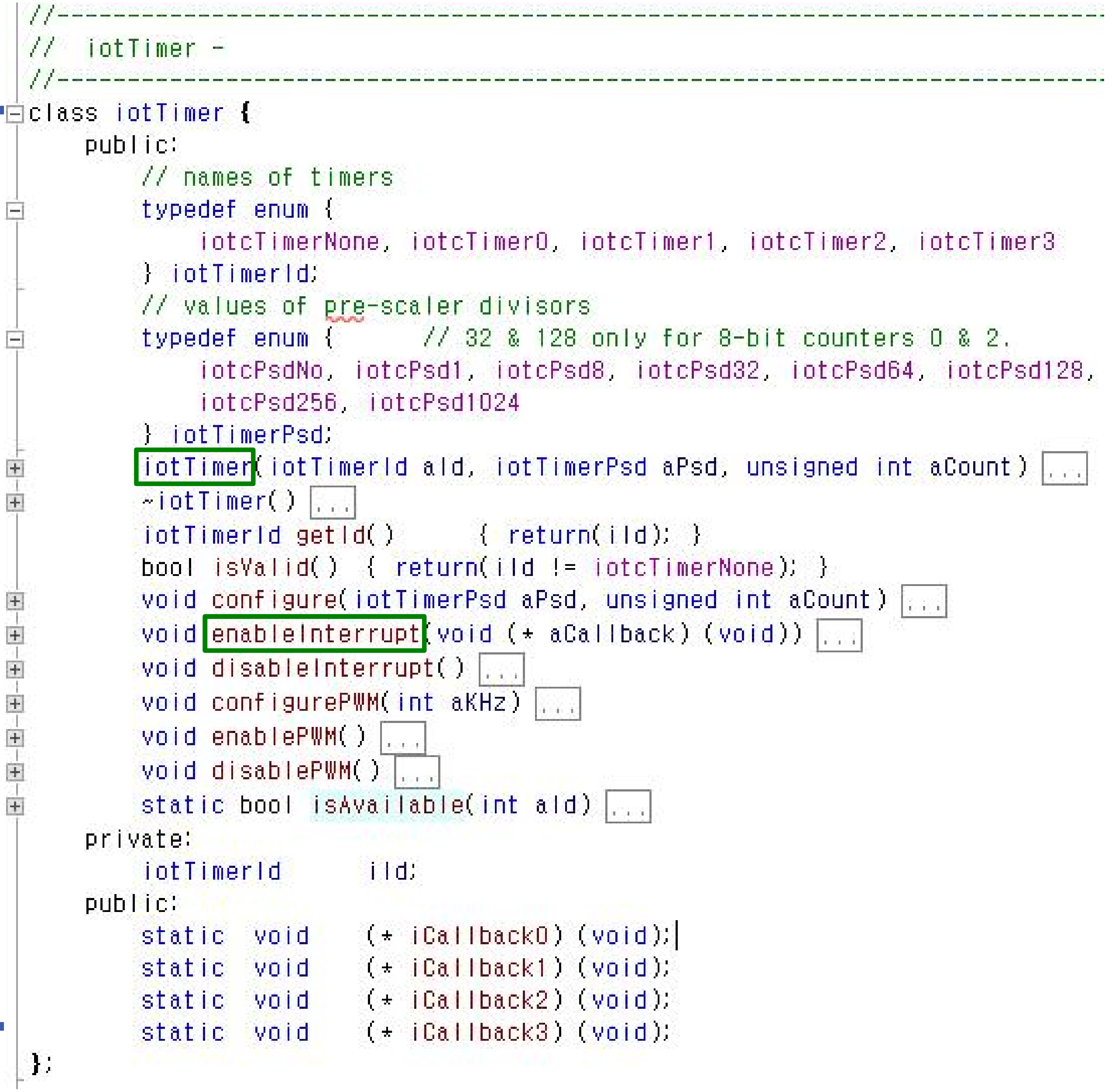
iot.h: INU Object Toolkit



```
//------------------------------------------------------------------------
//  INU Object Toolkit C++ Version -
//------------------------------------------------------------------------
#define     F_CPU               16000000UL
#define     iotmAtomic(x)   { char _sreg = SREG; cli(); x; SREG = _sreg; }
```

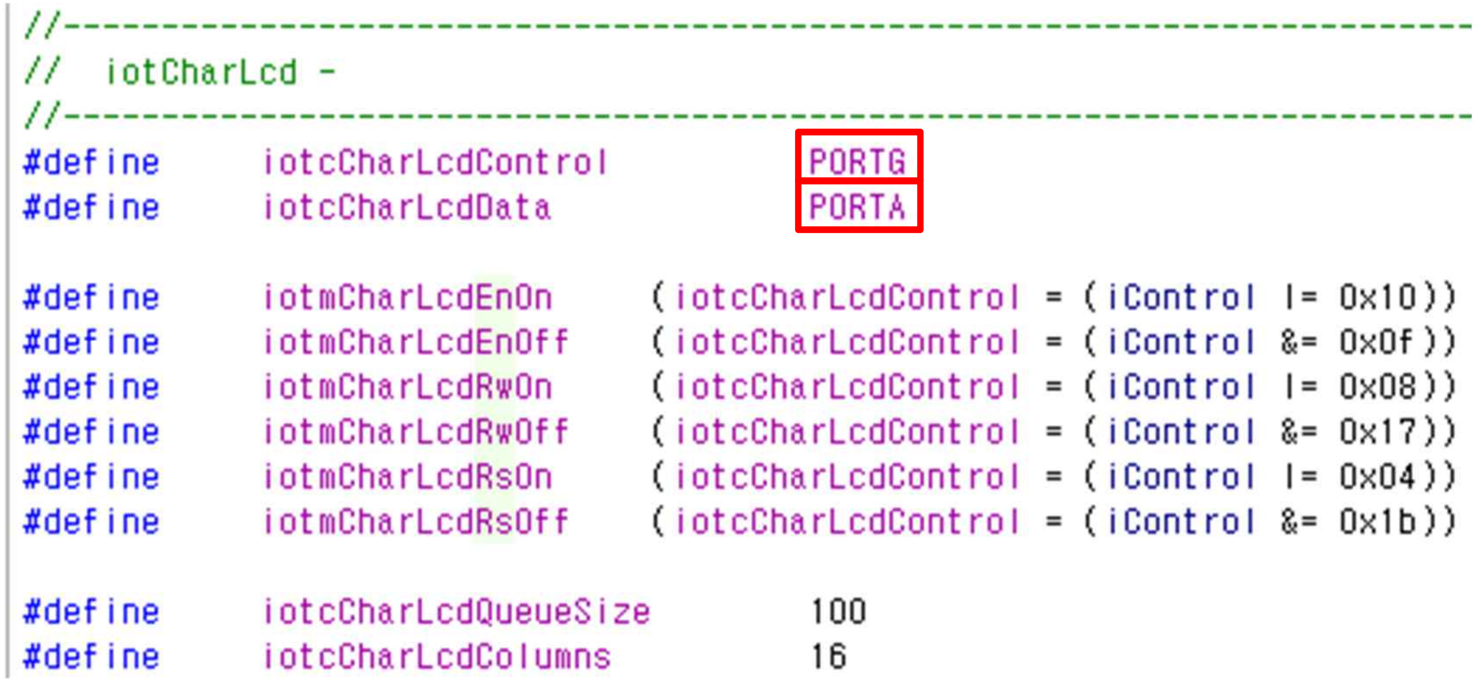
// 16 MHz

iot.h: Timer



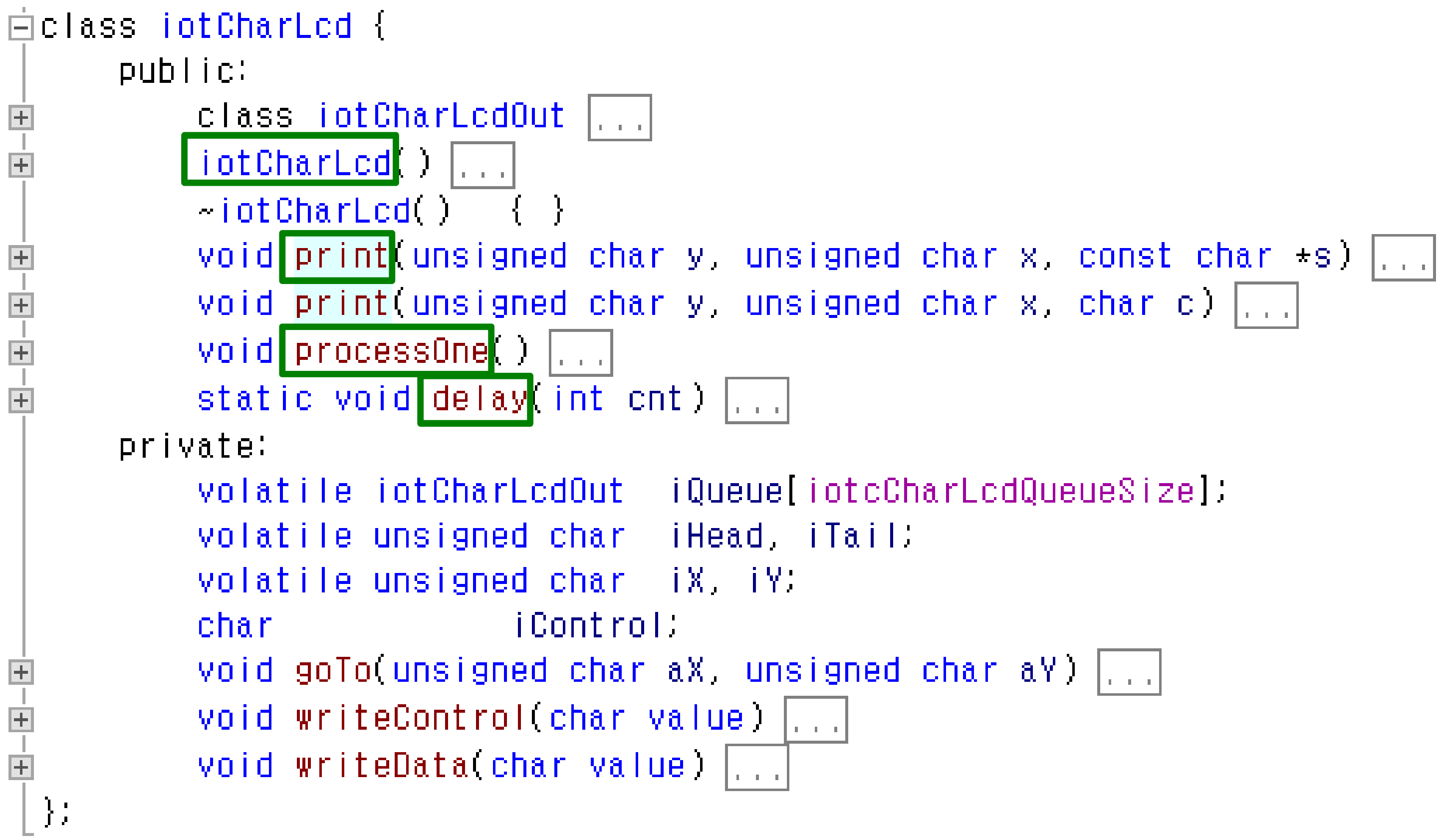
```
//-------------------------------------------------------------------------
//   iotTimer -
//-------------------------------------------------------------------------
class iotTimer {
    public:
        // names of timers
        typedef enum {
            iotcTimerNone, iotcTimer0, iotcTimer1, iotcTimer2, iotcTimer3
        } iotTimerId;
        // values of pre-scaler divisors
        typedef enum {      // 32 & 128 only for 8-bit counters 0 & 2.
            iotcPsdNo, iotcPsd1, iotcPsd8, iotcPsd32, iotcPsd64, iotcPsd128,
            iotcPsd256, iotcPsd1024
        } iotTimerPsd;
        iotTimer(iotTimerId aId, iotTimerPsd aPsd, unsigned int aCount) ...
        ~iotTimer() ...
        iotTimerId getId()      { return(iId); }
        bool isValid()  { return(iId != iotcTimerNone); }
        void configure(iotTimerPsd aPsd, unsigned int aCount) ...
        void enableInterrupt(void (* aCallback) (void)) ...
        void disableInterrupt() ...
        void configurePWM(int aKHz) ...
        void enablePWM() ...
        void disablePWM() ...
        static bool isAvailable(int aId) ...
    private:
        iotTimerId      iId;
    public:
        static  void    (* iCallback0) (void);
        static  void    (* iCallback1) (void);
        static  void    (* iCallback2) (void);
        static  void    (* iCallback3) (void);
};
```

iot.h: CharLcd



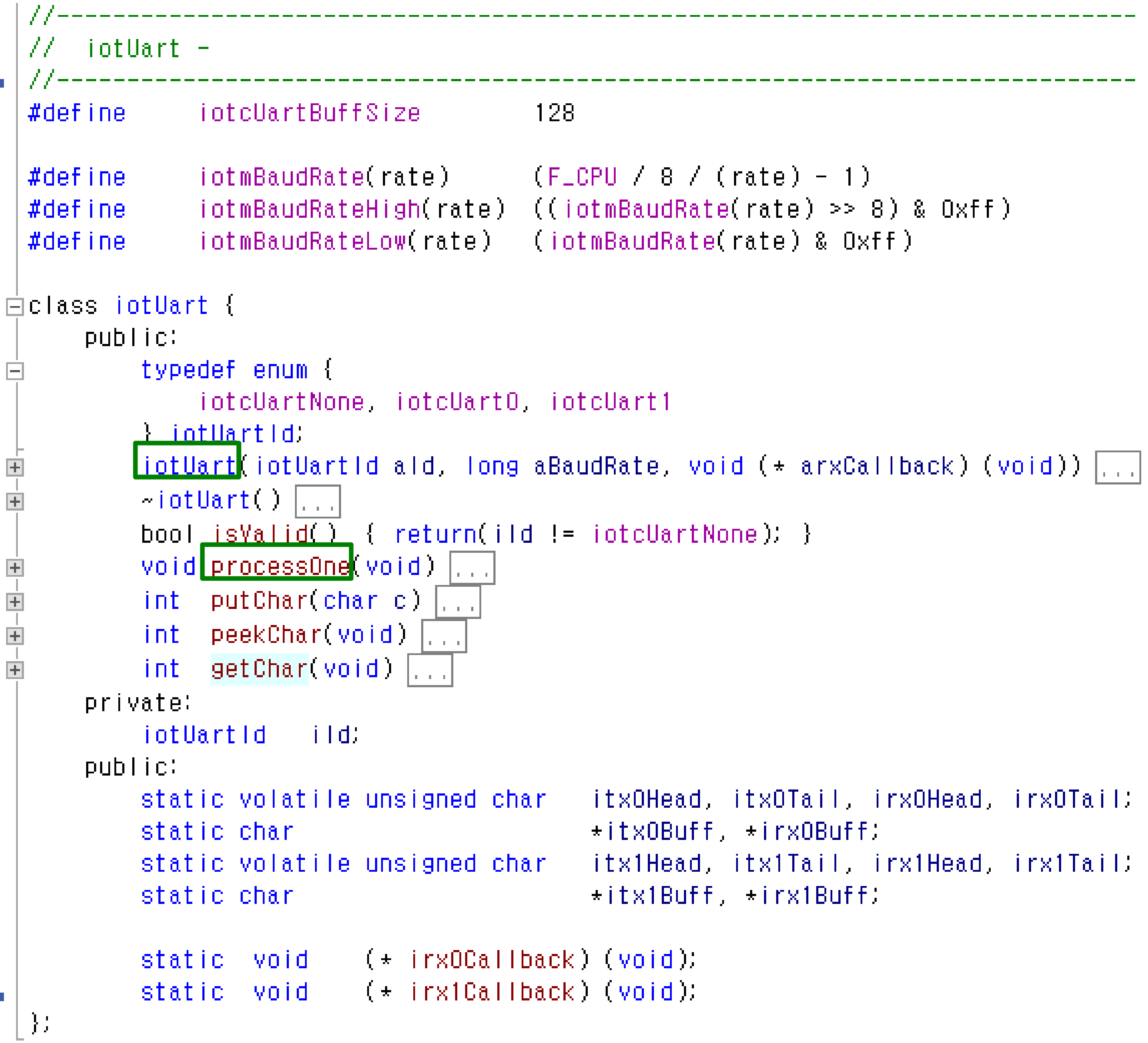
```
//----------------------------------------------------------------------
//  iotCharLcd -
//----------------------------------------------------------------------
#define     iotcCharLcdControl            PORTG
#define     iotcCharLcdData               PORTA

#define     iotmCharLcdEnOn     (iotcCharLcdControl = (iControl |= 0x10))
#define     iotmCharLcdEnOff    (iotcCharLcdControl = (iControl &= 0x0f))
#define     iotmCharLcdRwOn     (iotcCharLcdControl = (iControl |= 0x08))
#define     iotmCharLcdRwOff    (iotcCharLcdControl = (iControl &= 0x17))
#define     iotmCharLcdRsOn     (iotcCharLcdControl = (iControl |= 0x04))
#define     iotmCharLcdRsOff    (iotcCharLcdControl = (iControl &= 0x1b))

#define     iotcCharLcdQueueSize          100
#define     iotcCharLcdColumns            16
```

iot.h: CharLcd



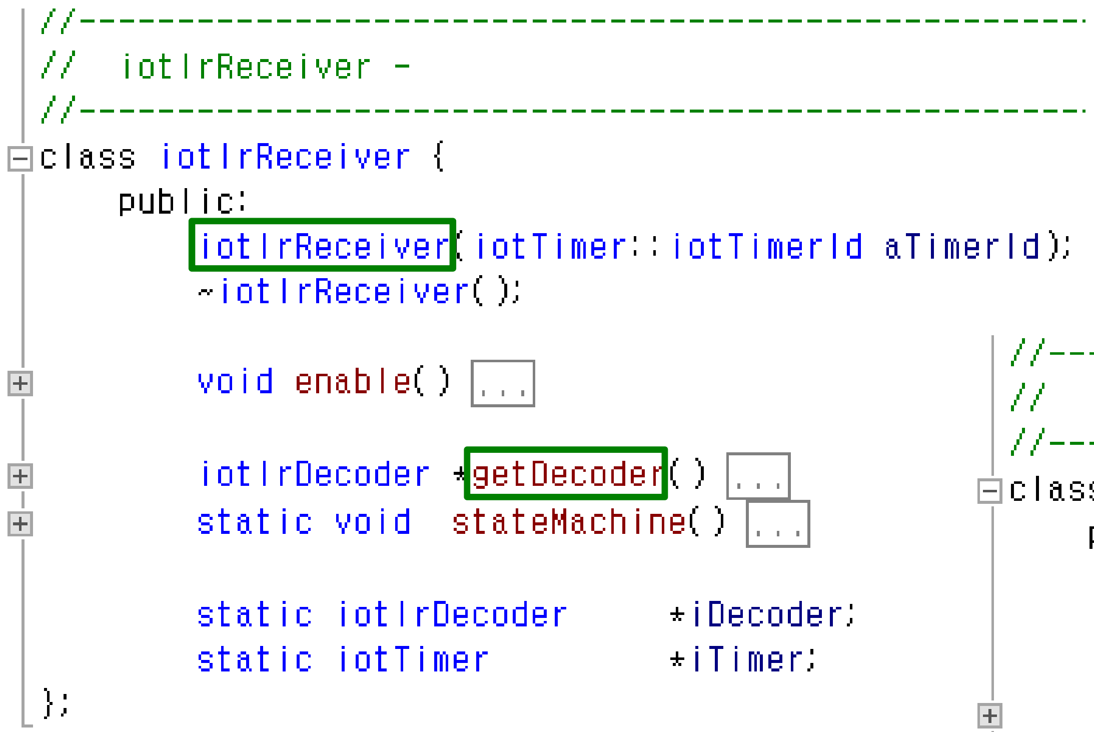
```
class iotCharLcd {
    public:
        class iotCharLcdOut ...
        iotCharLcd() ...
        ~iotCharLcd()   { }
        void print(unsigned char y, unsigned char x, const char *s) ...
        void print(unsigned char y, unsigned char x, char c) ...
        void processOne() ...
        static void delay(int cnt) ...
    private:
        volatile iotCharLcdOut   iQueue[iotcCharLcdQueueSize];
        volatile unsigned char   iHead, iTail;
        volatile unsigned char   iX, iY;
        char                     iControl;
        void goTo(unsigned char aX, unsigned char aY) ...
        void writeControl(char value) ...
        void writeData(char value) ...
};
```

iot.h: Uart

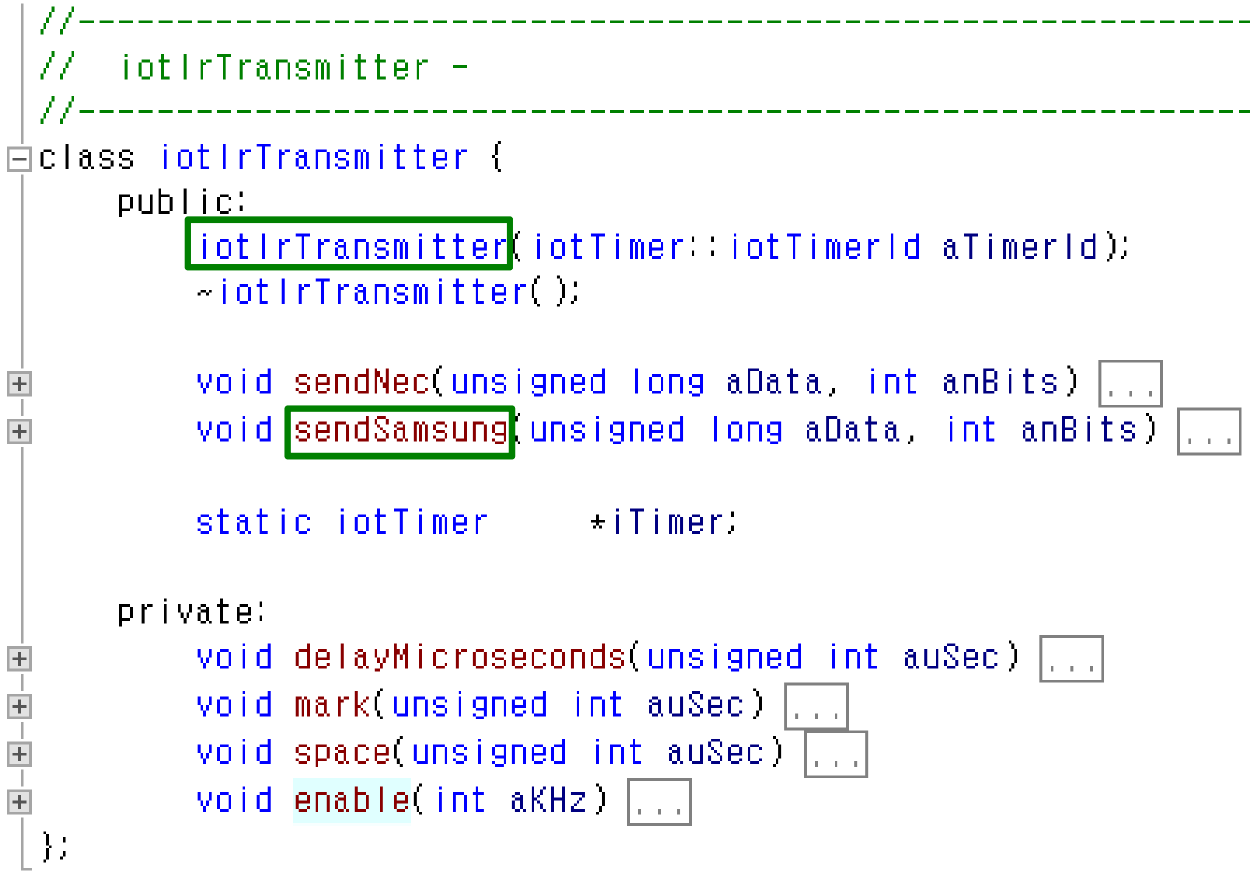


```
//-----------------------------------------------------------------------------
//  iotUart -
//-----------------------------------------------------------------------------
#define     iotcUartBuffSize        128

#define     iotmBaudRate(rate)      (F_CPU / 8 / (rate) - 1)
#define     iotmBaudRateHigh(rate)  ((iotmBaudRate(rate) >> 8) & 0xff)
#define     iotmBaudRateLow(rate)   (iotmBaudRate(rate) & 0xff)

class iotUart {
    public:
        typedef enum {
            iotcUartNone, iotcUart0, iotcUart1
        } iotUartId;
        iotUart(iotUartId aId, long aBaudRate, void (* arxCallback) (void)) ...
        ~iotUart() ...
        bool isValid() { return(iId != iotcUartNone); }
        void processOne(void) ...
        int  putChar(char c) ...
        int  peekChar(void) ...
        int  getChar(void) ...
    private:
        iotUartId   iId;
    public:
        static volatile unsigned char   itx0Head, itx0Tail, irx0Head, irx0Tail;
        static char                     *itx0Buff, *irx0Buff;
        static volatile unsigned char   itx1Head, itx1Tail, irx1Head, irx1Tail;
        static char                     *itx1Buff, *irx1Buff;

        static  void    (* irx0Callback) (void);
        static  void    (* irx1Callback) (void);
};
```

irRemote.h

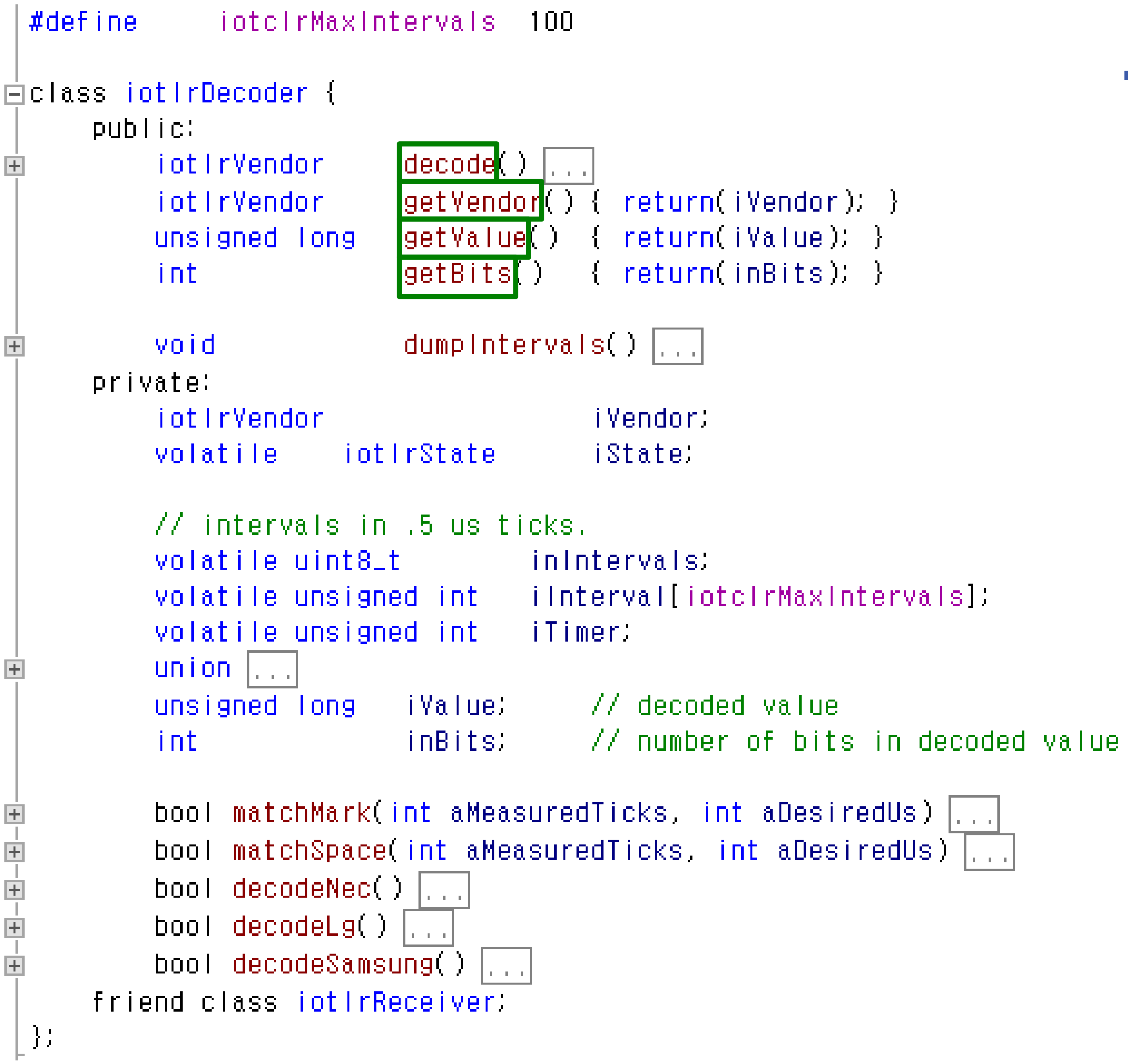


```
//-----------------------------------------------------------
//  iotIrReceiver -
//-----------------------------------------------------------
class iotIrReceiver {
    public:
        iotIrReceiver(iotTimer::iotTimerId aTimerId);
        ~iotIrReceiver();

        void enable() ...

        iotIrDecoder *getDecoder() ...
        static void  stateMachine() ...

        static iotIrDecoder      *iDecoder;
        static iotTimer          *iTimer;
};
```



```
//-----------------------------------------------------------
//  iotIrTransmitter -
//-----------------------------------------------------------
class iotIrTransmitter {
    public:
        iotIrTransmitter(iotTimer::iotTimerId aTimerId);
        ~iotIrTransmitter();

        void sendNec(unsigned long aData, int anBits) ...
        void sendSamsung(unsigned long aData, int anBits) ...

        static iotTimer      *iTimer;

    private:
        void delayMicroseconds(unsigned int auSec) ...
        void mark(unsigned int auSec) ...
        void space(unsigned int auSec) ...
        void enable(int aKHz) ...
};
```

irRemote.h



```
#define     iotcIrMaxIntervals  100

class iotIrDecoder {
    public:
        iotIrVendor     decode() ...
        iotIrVendor     getVendor() { return(iVendor); }
        unsigned long   getValue()  { return(iValue); }
        int             getBits()   { return(inBits); }

        void            dumpIntervals() ...
    private:
        iotIrVendor                     iVendor;
        volatile    iotIrState          iState;

        // intervals in .5 us ticks.
        volatile uint8_t            inIntervals;
        volatile unsigned int       iInterval[iotcIrMaxIntervals];
        volatile unsigned int       iTimer;
        union ...
        unsigned long   iValue;     // decoded value
        int             inBits;     // number of bits in decoded value

        bool matchMark(int aMeasuredTicks, int aDesiredUs) ...
        bool matchSpace(int aMeasuredTicks, int aDesiredUs) ...
        bool decodeNec() ...
        bool decodeLg() ...
        bool decodeSamsung() ...
    friend class iotIrReceiver;
};
```