

```
// Ein kleines ISR Programm, um LEDs mit konfigurierbarer Sequenz blinken zu lassen.
// Das rote ist zum testen eingefügt, um zu sehen, wie der Timer Interrupt sich verhält,
// wenn ein LED-Kanal behindert wird.
```

```
***** D E F I N E S *****
```

```
// besser lesbar
#define byte      uint8_t
#define word      uint16_t
#define dword     uint32_t

// Allgemein
#define NC        255 // Default für "unbenutzt" (z.B. Pin)

// Timer 2 Interrupt: Haupttakt konfigurieren
#define MAIN_CLK  160 // Interrupts pro Sekunde (Timer-Takt)
#define OCR2A_KOR  2 // Manueller OCR2A Korrekturfaktor

/* LM: LED-Management */

// LM: Allgemeine Einstellungen
#define LM_COUNT   4 // Anzahl Kanäle
#define LM_CLOCK   20 // Timer-Prescaler Default

// LM_01: Kanal 1 konfigurieren (LED rot)
#define LM_01_IDX  0 // Index im Arbeitsbereich
#define LM_01_PIN  37 // Pin-Nr.

// LM_02: Kanal 2 konfigurieren (LED gelb)
#define LM_02_IDX  1 // Index im Arbeitsbereich
#define LM_02_PIN  39 // Pin-Nr.

// LM_03: Kanal 3 konfigurieren (LED grün)
#define LM_03_IDX  2 // Index im Arbeitsbereich
#define LM_03_PIN  41 // Pin-Nr.

// LM_04: Kanal 4 konfigurieren
#define LM_04_IDX  3 // Index im Arbeitsbereich
#define LM_04_PIN  NC // Pin-Nr.

// LM: Verwendete Kanäle (Sprechender Tabellenindex)
#define LED_ROT    LM_01_IDX // LED rot
#define LED_GELB   LM_02_IDX // LED gelb
#define LED_GRUEN  LM_03_IDX // LED grün
```

```
***** T Y P E S *****
```

```
// LM: Struktur des Arbeitsbereichs
struct tLM {
    byte Pin; // Pin-Nr. (NC=ungenutzt)
    word TPrescaler; // Timer-Prescaler (0=Kanal deaktiviert)
    word TCounter; // Prescaler-Counter
    byte InUse; // ISR-Sperrflag (1=Kanal wird gerade verwendet)
    byte Pattern; // LED-Blinkmuster
    byte BitPointer; // Bit-Zeiger
};
```

```
***** G L O B A L E   V A R I A B L E N *****
```

```
// LM: LED-Arbeitsbereich anlegen
volatile tLM LM[LM_COUNT];
```

```
***** F O R W A R D S *****
```

```
void mySetup();
```

```
***** A D M I N I S T R A T I O N S T O O L S *****
```

```
/* Zeigt die Millisekunden mit Nachkommastellen seit dem letzten Aufruf an */
/* fNoPrint : 1=initialisiert nur den Zeitstempel (kein Ausgabe) */
void PrintMillisDelta(byte iKanal=0, byte fNoPrint=0) {
```

```

static dword Timestamp[10];
if (!fNoPrint)
    {Serial.println( (micros() - Timestamp[iKanal]) / 1000.0);}
Timestamp[iKanal] = micros();
}

```

```

/*****
/* Byte als Binärzahl seriell ausgeben */
*****/

```

```

void PrintBinaer(byte i, byte fMitBitNr=0, byte fNoLN=0) {
    if (fMitBitNr)
        {Serial.println("76543210");}
    for (byte n=0; n<8; n++)
        {Serial.print(i & (128 >> n) ? 1 : 0);}
    if (!fNoLN)
        {Serial.println("");}
}

```

```

/***** I N I T I A L I S I E R U N G *****/

```

```

void setup() {

    /*****/
    /* Serielle Schnittstelle und Admintool */
    /*****/
    PrintMillisDelta(0, 1); // Zeitmesser 0 initialisieren
    Serial.begin(9600); // Geschwindigkeit der Schnittstelle zum PC
    delay(250); // kurz warten
    Serial.println("(0) Hallo Meister");

    /*****/
    /* LM: LED-Management initialisieren */
    /*****/
    for (byte ix=0; ix<LM_COUNT; ix++) {
        tLM *p; // Hilfspointer
        p = &LM[ix]; // Zeiger auf Arbeitsbereich

        // LM: Allgemeine Einstellungen
        p->TPrescaler = 0; // Timer-Prescaler
        p->InUse = 0; // ISR-Sperrflag

        // LM: Spezifische Einstellungen
        switch (ix) {
            case LM_01_IDX: // Kanal 1
                p->Pin = LM_01_PIN; // Pin-Nr.
                break;
            case LM_02_IDX: // Kanal 2
                p->Pin = LM_02_PIN; // Pin-Nr.
                break;
            case LM_03_IDX: // Kanal 3
                p->Pin = LM_03_PIN; // Pin-Nr.
                break;
            case LM_04_IDX: // Kanal 4
                p->Pin = LM_04_PIN; // Pin-Nr.
                break;
            default:
                p->Pin = NC; // Ungültiger Index, keine Pin-Nr.
        } // Ende Spezifische Einstellungen

        // LM: Pin als Ausgang definieren
        if (p->Pin != NC) { // Nur wenn eine Pin definiert ist
            pinMode(p->Pin, OUTPUT); // Pin auf OUTPUT schalten
            digitalWrite(p->Pin, LOW); // LED ausschalten
        } // Ende Pin als Ausgang definieren
    } // Ende LED-Management initialisieren

    /*****/
    /* SM: Interrupt konfigurieren und Timer starten */
    /*****/
    cli(); // Interrupts sperren
    TCNT2 = 0; // Timer Counter 2
    TCCR2A = 0; // Timer Counter Controll Register A löschen
    TCCR2B = 0; // Timer Counter Controll Register B löschen
    OCR2A = (16000000.0/(1024.0*MAIN_CLK))+OCR2A_KOR; // Vergleichswert in Output Compare Register A
    TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20); // Prescaler 1024
}

```

```

TCCR2A |= (1<<WGM21); // CTC-Mode ein
TIMSK2 |= (1<<OCIE2A); // Output Compare A Match Interrupt Enable
sei(); // Interrupts erlauben

/*****/
/* Vordergrundprogramm initialisieren */
/*****/
Serial.print("(0) HG-Setup beendet. Dauer: ");
PrintMillisDelta(0);
mySetup(); // VG-Setup
Serial.print("(0) VG-Setup beendet. Dauer: ");
PrintMillisDelta(0);

} // Ende setup()

/***** T I M E R E V E N T *****/

/*****/
/* Timer 2 Interrupt 160x pro Sekunde */
/*****/
ISR(TIMER2_COMPA_vect) {

/*****/
/* Nur zum Test */
/*****/
sei(); // Das ist eine Männer-ISR!

/*****/
/* LM: LEDs versorgen */
/*****/
for (byte ix=0; ix<LM_COUNT; ix++) {
    tLM *p; // Hilfspointer
    p = &LM[ix]; // Zeiger auf Arbeitsbereich

    // LM: Kanal aktiviert?
    if (p->TPrescaler == 0) { /*schon fertig*/} // Prescaler=0 sperrt die Ausführung

    // LM: Kanal gesperrt?
    else if (p->InUse) { /*besetzt*/} // ISR-Sperrflag: Kanal wird gerade verwendet

    // LM: Prescaler abgelaufen?
    else if (++p->TCounter >= p->TPrescaler) { // Der DM-Timerevent verwendet einen Prescaler
        p->TCounter = 0; // Auslösewert erreicht: Counter wieder löschen

        // LM: Kanal sperren
        p->InUse = 1; // Nicht stören

// XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
// Eingefügt zum Testen des Timers
if (ix == LED_GELB) { // Kanal der gelben LED
    PrintMillisDelta(2, 1); // blockieren, um zu sehen,
    Serial.println("(2) Gelbe LED 5 Sek blockiert"); // ob die anderen LEDs noch
    word iTS = millis() + 5000; // blinken. Sie blinken nur
    while(iTS > millis()); // dann, wenn zu ISR-Beginn
    Serial.print("(2) Gelbe LED wieder frei. Dauer: "); // ein sei() eingefügt wurde.
    PrintMillisDelta(2);
}
// XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    // LM: 8 Bits ausgegeben?
    if (++p->BitPointer > 8) // Wenn 8 Bits ausgegeben sind
        {p->BitPointer = 0;} // Bitzeiger wieder auf 0

    // LM: LED ansteuern
    byte f = p->Pattern & (1 << p->BitPointer); // Bit auslesen
    digitalWrite(p->Pin, f); // LED umschalten

    // LM: Kanalsperre aufheben
    p->InUse = 0; // Kanal wieder freigeben

} // Ende Prescaler abgelaufen?
} // Ende LEDs versorgen
} // Ende ISR

/***** B A S I S F U N K T I O N E N *****/

```

```

/*****
/* LM: Konfiguriert einen LED-Kanal */
/*
/* iKanal      : Nummer der LED (Tabellenindex, beginnend ab 0) */
/* iPattern    : Bitmuster */
/* iPrescaler  : Blinkgeschwindigkeit Standard: 8 Bit pro Sekunde) */
*****/
void LM_SetLED(byte iKanal,
               byte iPattern,
               word iPrescaler = LM_CLOCK) {

    tLM *p;                                // Hilfspointer
    p = &LM[iKanal];                      // Zeiger auf Arbeitsbereich

    // LM: Kanal konfigurieren
    cli();                                // Interrupts verbieten
    p->Pattern    = iPattern;              // LED-Bitmuster
    p->BitPointer = 0;                     // Bit-Zeiger auf 0
    p->TCounter   = 0;                     // Prescaler-Counter löschen
    p->TPrescaler = iPrescaler;            // Prescaler übergeben
    sei();                                  // Interrupts erlauben
}

```

```

/***** V O R D E R G R U N D P R O G R A M M *****/

```

```

void mySetup() {
    LM_SetLED(LED_ROT,    0b11110000);
    LM_SetLED(LED_GELB,   0b11001100);
    LM_SetLED(LED_GRUEN,  0b10101010);

    Serial.println("(1) Start delay(1000)");
    PrintMillisDelta(1, 1);
    delay(1000);
    Serial.print("(1) Ende delay(1000). Dauer: ");
    PrintMillisDelta(1);

    LM_SetLED(LED_GRUEN, 0b00001111);
}

void loop() {
}

```