Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

АППАРАТНЫЕ СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

«Битовый процессор 2»

Преподаватель:

Рафиков А.Г. _____
(подпись, дата)

Студент:

Девяткин Е.Д., группа ИУ8-74 (4 курс) _____
(подпись, дата)

Москва, 2025

Содержание

Цель работы

Собрать в Proteus модель центра управления сигналами автомобиля, используя МК i8051. КЗ должен быть равен 30%. В качестве индикации должен быть выбран дисплей.

Теоретическая часть

В программе используется таймер 0 микроконтроллера в режиме 1 (16-битный), настроенный на генерацию прерываний с фиксированным периодом. Прерывание таймера – это автоматический вызов подпрограммы обработки (в данном случае `TL0_PROC`) при переполнении таймера, что позволяет выполнять периодические действия без блокировки основной программы.

При инициализации (подпрограмма `INIT`) устанавливаются разрешения глобальных прерываний (`EA`) и прерываний от таймера 0 (`ET0`), а также загружаются начальные значения в регистры `TL0` и `TH0` для задания периода:

```
INIT:
   MOV TMOD, #00000001B
   MOV TL0, #0
   MOV TH0, #-16
   MOV SUB_DIV, #244
   MOV R5, #61
   SETB ET0
   SETB EA
   SETB TR0
   CLR RS
   CLR E
   CLR F0
   MOV R7, #1
   MOV F, #255
   MOV R6, #48
   RET
```

Значение -16 в `TH0` задаёт начальное счётное значение таймера, при котором он переполняется каждые 16 машинных циклов, обеспечивая высокую частоту прерываний (~15,625 кГц при частоте кварца 12 МГц).

Для снижения частоты мигания используется программный делитель `SUB_DIV`, который декрементируется при каждом прерывании и сбрасывается после достижения нуля:

```
TL0_PROC:
    MOV TL0, #0
    MOV TH0, #-16        ; перезагрузка таймера
    DJNZ SUB_DIV, T0_SERV ; уменьшить SUB_DIV, если не ноль — перейти к T0_SERV
    MOV SUB_DIV, #244    ; сбросить делитель
```

Когда SUB_DIV достигает нуля, его биты используются для формирования двух условных уровней частоты: бит 0 (SUB_DIV.0) обозначает высокую частоту (HI_FREQ), а бит 7 (SUB_DIV.7) – низкую (LO_FREQ).

По условию скважность сигнала должна быть 30% (сигнал включён 30% времени, выключен – 70%). Для этого в программе анализируются младшие три бита регистра SUB_DIV (биты 2, 1 и 0). Таблица истинности показывает, при каких комбинациях выход должен быть включён:

SUB_DIV.1	SUB_DIV.0	1 * 0	30%
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	0
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	1

Реализация в программе:

```
MOV C, SUB_DIV.1
ANL C, SUB_DIV.2
MOV DIM, C
```

В обработчике прерывания TL0_PROC сначала таймеру снова задают начальное значение, чтобы он продолжал отсчитывать одинаковые промежутки времени. Потом уменьшают счётчик SUB_DIV, и по состоянию его битов определяют, должен ли сейчас гореть или гаснуть сигнал – этот результат временно сохраняют во флаге DIM.

Основной цикл программы пуст (SJMP $), так как вся логика управления световыми индикаторами реализована в фоновом режиме через прерывания таймера.

Выполнение работы

Задание 1

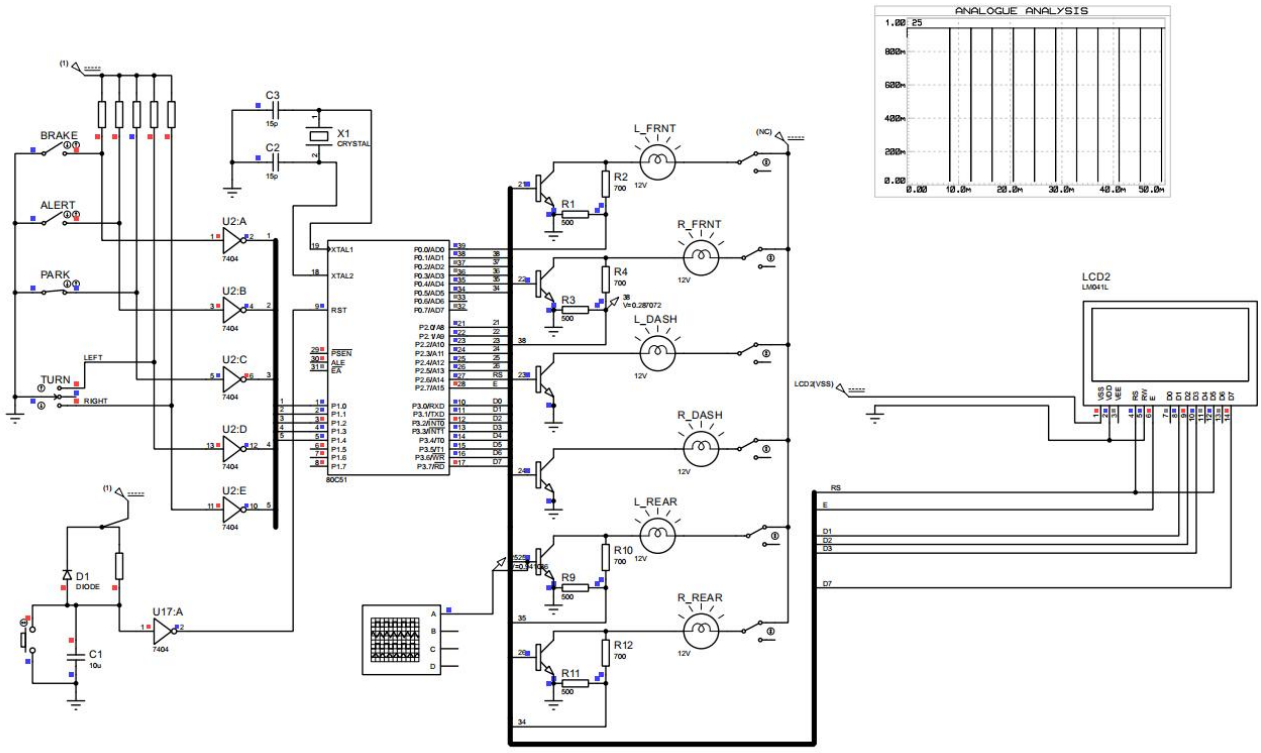

Рис. 1 - Схема.

```
$NOMOD51
$INCLUDE (8051.MCU)

BRAKE BIT P1.0
EMERG BIT P1.1
PARK BIT P1.2
L_TURN BIT P1.3
R_TURN BIT P1.4
L_FRNT BIT P2.0
R_FRNT BIT P2.1
L_DASH BIT P2.2
R_DASH BIT P2.3
L_REAR BIT P2.4
R_REAR BIT P2.5
RS BIT P2.6
E BIT P2.7
DD DATA P3
F DATA 21h
SUB_DIV DATA 20H
HI_FREQ BIT SUB_DIV.0
ME_FREQ BIT SUB_DIV.5
LO_FREQ BIT SUB_DIV.7

ORG 0000H

ACALL INIT
JMP MAIN
```

```
ORG 000BH

ACALL TMR0_PROCCESING
RETI

ORG 150H

MAIN:
   SJMP $

ORG 200H

INIT:
   MOV TMOD, #00000001B
   MOV TL0, #0
   MOV TH0, #-16
   MOV SUB_DIV, #244
   MOV R5, #61
   SETB ET0
   SETB EA
   SETB TR0
   CLR RS
   CLR E
   CLR F0
   MOV R7, #1
   MOV F, #255
   MOV R6, #48
   RET

TMR0_PROCCESING:
   MOV TL0, #0
   MOV TH0, #-16
   DJNZ SUB_DIV, T0_SERV
   MOV SUB_DIV, #244

T0_SERV:
   CLR E
   MOV A, R7
   JNZ INIT_LCD1
   DJNZ R5, NEXT
   MOV R5, #61

CHECK00:
   CLR L_FRNT
   CLR R_FRNT
   CLR L_DASH
   CLR R_DASH
   CLR L_REAR
   CLR R_REAR
   MOV F, P0

NEXT:
   DJNZ R6, CHECK
   MOV R6, #24

CHECK:
   MOV A, R6
   ANL A, #00010000B
   JNZ CHECK_456
```

```
CHECK_123:
   MOV A, R6
   ANL A, #00001000B
   JNZ CHECK_23

CHECK_1:
   MOV A, R6
   ANL A, #00000010B
   JNZ SET_ADDR1
   JB F.0, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR1:
   MOV A, #10000000B
   JMP SET_ADDR

CHECK_23:
   MOV A, R6
   ANL A, #00000100B
   JNZ CHECK_3

CHECK_2:
   MOV A, R6
   ANL A, #00000010B
   JNZ SET_ADDR2
   JB F.1, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR2:
   MOV A, #10000010B
   JMP SET_ADDR

INIT_LCD1:
   LJMP INIT_LCD

CHECK_3:
   MOV A, R6
   ANL A, #00000010B
   JNZ SET_ADDR3
   JB F.2, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR3:
   MOV A, #10000100B
   JMP SET_ADDR

CHECK_456:
   MOV A, R6
   ANL A, #00001000B
   JNZ CHECK_6

CHECK_45:
   MOV A, R6
   ANL A, #00000100B
   JNZ CHECK_5

CHECK_4:
   MOV A, R6
   ANL A, #00000010B
```

```
   JNZ SET_ADDR4
   JB F.3, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR4:
   MOV A, #10000110B
   JMP SET_ADDR

CHECK_5:
   MOV A, R6
   ANL A, #00000010B
   JNZ SET_ADDR5
   JB F.4, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR5:
   MOV A, #10001000B
   JMP SET_ADDR

CHECK_6:
   MOV A, R6
   ANL A, #00000010B
   JNZ SET_ADDR6
   JB F.5, PRINT_POINT
   JMP CLEAR_POINT

SET_ADDR6:
   MOV A, #10001010B
   JMP SET_ADDR

PRINT_POINT:
   SETB RS
   MOV DD, #00101011B
   MOV A, R6
   ANL A, #00000001B
   JNZ SAVE
   JMP DEC1

CLEAR_POINT:
   SETB RS
   MOV DD, #00110000B
   MOV A, R6
   JMP SAVE

SET_ADDR:
   CLR RS
   MOV DD, A
   MOV A, R6

SAVE:
   SETB E

DEC1:
   DEC R6


PARK_SCRIPT:
    ; ??????? ????????? ??????? ??? ordinary_mode
    MOV C, L_TURN
    ORL C, EMERG
```

```
    ANL C, LO_FREQ
    MOV L_DASH, C
    MOV L_FRNT, C
    MOV L_REAR, C

    MOV C, R_TURN
    ORL C, EMERG
    ANL C, LO_FREQ
    MOV R_DASH, C
    MOV R_FRNT, C
    MOV R_REAR, C

    ; ???? ??????????? ordinary_mode (??????????? ??? ????????),
    ; ?? ?????????? ??????????? ?????
    JNB L_TURN, CHECK_R_TURN
    JMP STOP_SCRIPT

CHECK_R_TURN:
    JNB R_TURN, CHECK_EMERG
    JMP STOP_SCRIPT

CHECK_EMERG:
    JNB EMERG, CHECK_PARK
    JMP STOP_SCRIPT

    ; ???? ordinary_mode ?? ???????, ????????? ??????????? ?????
CHECK_PARK:
    JNB PARK, STOP_SCRIPT

    ; ??????????? ?????
    CLR L_DASH
    CLR R_DASH
    MOV C, HI_FREQ
    MOV C, SUB_DIV.2
    ANL C, SUB_DIV.1
    ANL C, PARK
    MOV L_REAR, C
    MOV R_REAR, C
    MOV L_FRNT, C
    MOV R_FRNT, C

STOP_SCRIPT:
    ; ?????? ?????? ????? ????????? ??? ??????? ????????
    MOV C, BRAKE
    JNC BRAKE_NOT_PRESSED
    SETB L_REAR
    SETB R_REAR
BRAKE_NOT_PRESSED:
    RET

INIT_LCD:
    MOV A, #00001100B
    MOV DD, A
    JNB HI_FREQ, MISS_CYCLE
    JMP LCD_CMD

MISS_CYCLE:
    RET

LCD_CMD:
```

```
    SETB E
    DEC R7
    RET

END
```

Вывод

Модель центра управления сигналами автомобиля собрана с использованием МК i8051. КЗ равен 90%. В качестве индикации выбран дисплей. Переход к подпрограммам, реализующим проверку ламп, индикации и проверки выбранного состояния сигнализации осуществлен через прерывания.