|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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) |

**Содержание**

[Цель работы 3](#_Toc10)

[Теоретическая часть 3](#_Toc779)

[Выполнение работы 5](#_Toc9964)

[Задание 1 5](#_Toc13521)

[Вывод 10](#_Toc24054)

# 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**

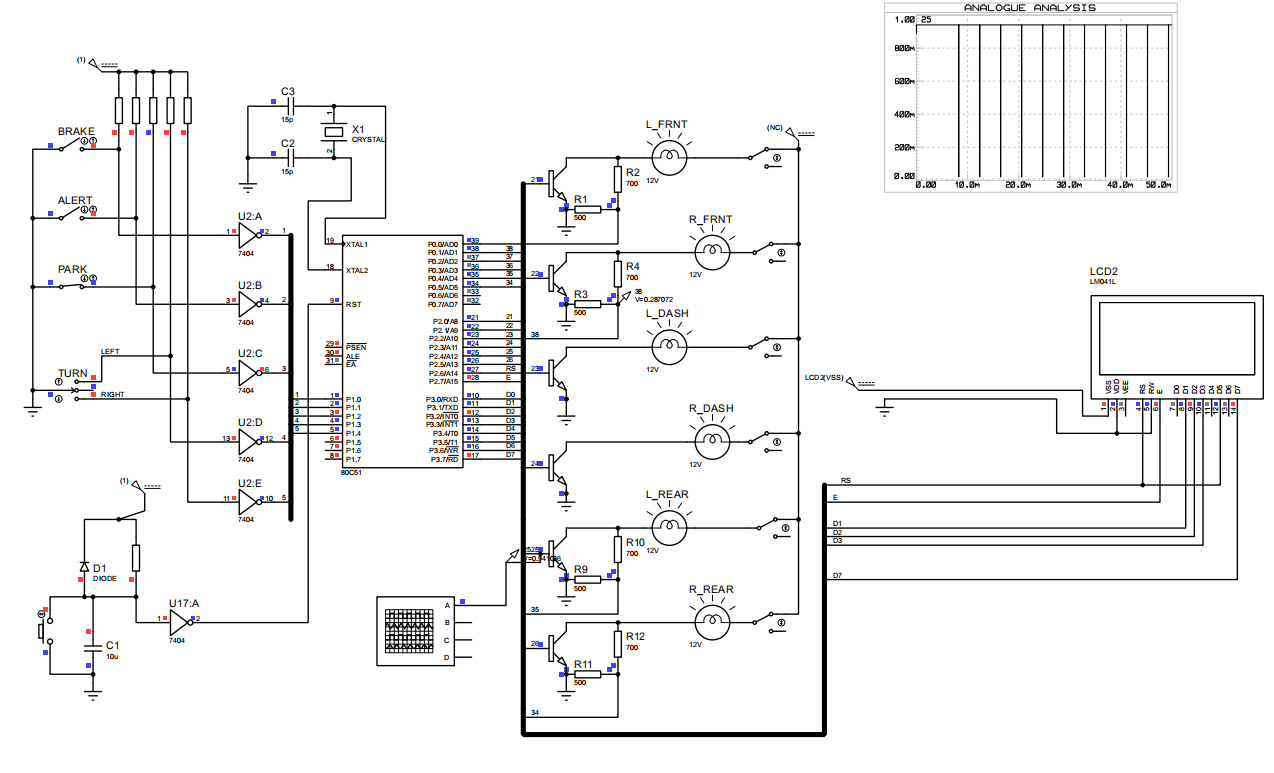


Рис. 1 - Схема.

$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.