Программа и схема алгоритмы:

В основном теле программы сначала настраиваем порты, таймер и UART. Затем циклически проверяем нажатые кнопки и ждём, пока не нажмут <- или ->. Как только нажали, если стрелка влево, то вводим через UART, иначе с матричной клавиатуры. Затем бесконечный цикл опроса кнопок клавиатуры. Если нажаты стрелки, то переход на следующий или предыдущий часовой пояс. Если цифры "0" "1" или "2", то ждём ввода следующей цифры. Эти два ввода и дадут нам часы, на которые следует заменить старое значение выбранного пояса (выбирать через стрелки влево-вправо)

Функция TIMER0_OVF_vect отвечает за расчёт времени, вывод на индикаторы текущего времени и проверяет матричную клавиатуру на нажатые клавиши.
Input_uart() отвечает за ввод исходного времени (HH:MM) и часов четырёх поясов (HH 4 раза)

input_keyaboard() полностью копирует функционал input_uart(), но ввод происходит с клавиатуры

Функциональная схема

4 для показа времении один спереди для показа номера поясатам будет или 1 или 2 или 3 или 4 светиться

Слева сверху подключены 5 семи-сегментных индикаторов. Подключаются они через драйвер MAX7219. Их выходы a-b-c-..-g соединены с выходами PA0-PA1-..-PA6 соответственно. Эти соединения отвечают за то, какие сегменты на индикаторах зажигаются, а какие нет(на картинке каждого индикатора каждый сегмент подписан конкретной буквой). Соединения DIG0-DIG4 индикаторов соединены с выходами PB0-..-PB4. При подаче сигнала на них индикатор отображает то, что пришло по a-g выходам, иначе он выключен

Снизу к линии порта B подключен программатор. Мы уже использовали его на лабораторных, при загрузке кода в плату. Подключается по SPI.
Справа снизу подключается ПЭВМ (ПК) через UART. Всего есть две линии у этой связи-RXD и TXD. Receive = приём, Transmit = передача. Поэтом стрелки от МК к ПК по TXD и к МК от ПК по RXD, а дальше они при помощи драйвера соединены с разъёмом RS232

Сверху справа матричная клавиатура, каждая кнопка подписана. По идее, двух кнопок в виде стрелок достаточно для управления. На выходы PC6-PC5-PC4 подаётся ток (1), а на PC0-PC1-PC2-PC3 ничего (0). Если ничего не нажато, то ток не может пройти дальше кнопок. Если кнопка нажата, то ток проходит дальше на землю через резисторы и на каком-то выходе PC6-PC5-PC4 становится (0) (изменение тока на противоположное) (тем самым мы знаем столбец кнопки), а также на конкретный выход из PC0-PC1-PC2-PC3 (тем самым мы знаем строку кнопки)

Но при такой клавиатуре нельзя обработать одновременное нажатие кнопок (вроде как)

Справа на линию RESET подключены питание и кнопка сброса с землёй, как только она нажата, МК перезагружается. Генератор подключен к кварцевому генератору (резонатору) с нек частотой. Это нужно, потом что внутренний генератор МК может выдавать только 1, 2, 4, 8, 16 МГц, а для ТОЧНОГО измерения времени нужна более точная частота работы (она влияет на вычисления таймера)

Справа снизу при помощи драйвера мы используем Юарт

Где разъём для подключения программатора

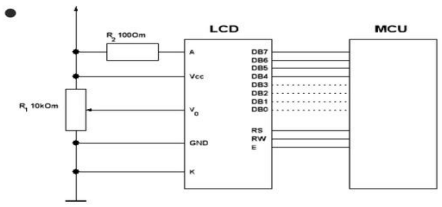
Это такое устройство, которое подключается по spi интерфейсу к нашему МК и загружает в него код

Кнопки и так понятно d1-d7 это входы дисплея, по которым мы передаем символы, которые нужно выводить на сам дисплей

тыре ножки контроллера меньше. Есть еще одна особенность работы в 8-битном режиме — к некоторым контроллерам можно подрубить этот дисплей **как внешнее ОЗУ** и засылать данные простыми командами пересылки. Лично я подключил его в режиме полного порта у меня один фиг выводы уже девать некуда было, так что не жалко.

- Выводы **DB7…DB0** это шина данных/адреса.
- **E** — стробирующий вход. Дрыгом напряжения на этой линии мы даем понять дисплею что нужно забирать/отдавать данные с/на шину данных.
- **RW** — определяет в каком направлении у нас движутся данные. Если 1 — то на чтение из дисплея, если 0 то на запись в дисплей.
- **RS** — определяет что у нас передается, команда (RS=0) или данные (RS=1). Данные будут записаны в память по текущему адресу, а команда исполнена контроллером.

Со стороны питания все еще проще:



Подключение дисплея