Сопряжение AVR-микроконтроллеров и ЖКИ

Татьяна Кривченко,
24 января 1999

1. Архитектура и система команд контроллера ЖКИ

Многие фирмы (**Optrex Corporation**, **Powertip**, **Seiko Instruments**, **Batron** и др.) выпускают жидкокристаллические индикаторы со встроенными контроллерами, облегчающими реализацию интерфейса ЖКИ и микропроцессора. В данной статье рассматриваются вопросы аппаратного и программного сопряжения микроконтроллеров AVR и символьных ЖКИ, построенных на базе контроллера **HD44780.**

Рассматриваемый ЖКИ при помощи стандартного 14-контактного разъема (**табл. 1**) обменивается информацией с управляющим микроконтроллером (в нашем случае с AVR). AVR-микроконтроллер посылает в ЖКИ команды (**табл. 2**), управляющие режимами его работы, и ASCII-коды выводимых символов. В свою очередь, ЖКИ может посылать AVR-микроконтроллеру по его запросу информацию о своем состоянии и данные из своих внутренних блоков памяти.

Таблица 1: Описание выводов стандартного разъема ЖКИ на базе HD44780

N	Название вывода	Описание
1	V_{SS}	(-) Питание. 0 V.
2	V_{DD}	(+) Питание.+5V.
3	V_0	Напряжение смещения, управляющее контрастностью
4	RS	Вход. Высокий уровень - Данные; Низкий - Команды
5	R/W	Вход. Высокий-Чтение, Низкий-Запись
6	E	Вход. Строб, сопровождающий сигналы на шине "команды/данные"
7	DB_0	Шина "команды/данные"
8	DB_1	
9	DB_2	
10	DB_3	
1	DB_4	

11		
12	DB_5	
13	DB_6	
14	DB_7	

Три вывода 14-контактного разъема предназначены для подачи питающего напряжения и напряжения смещения, которое управляет контрастностью дисплея. На **рис. 1** показана рекомендуемая схема подключения этих выводов.

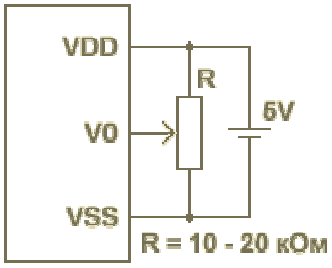

Рис. 1: Возможная схема питания ЖКИ

Из оставшихся 11 выводов 8 (DB0 - DB7) используются для организации мультиплексированной шины "команды / данные", и на 3 вывода (RS, R/W, E) AVR-микроконтроллер выставляет управляющие сигналы. На **рис. 2** изображены временные диаграммы этих сигналов при записи команд / данных в контроллер ЖКИ.

При помощи сигнала на линии RS микропроцессор сообщает контроллеру индикатора о том, что именно передается по шине: команда или данные. Сигнал на линии E является стробом, сопровождающим сигналы на шине "команды / данные". Запись информации в ЖКИ происходит по спаду этого сигнала. Потенциал на управляющем выводе R/W задает направление передачи данных: запись в RAM индикатора (R/W=0) или считывание оттуда (R/W=1).

Для случая, когда микроконтроллер имеет ограниченное количество линий ввода / вывода, предусмотрен второй вариант подключения ЖКИ с использованием 4-х разрядной шины "команды / данные". При этом каждый байт данных передается по линиям DB4 - DB7 последовательно двумя тетрадами, начиная со старшей.

Контроллер ЖКИ после приема байта команды или байта данных требует некоторого времени (**табл. 2**) для обработки полученной информации, в течение которого AVR-микроконтроллер не должен выполнять новых передач.

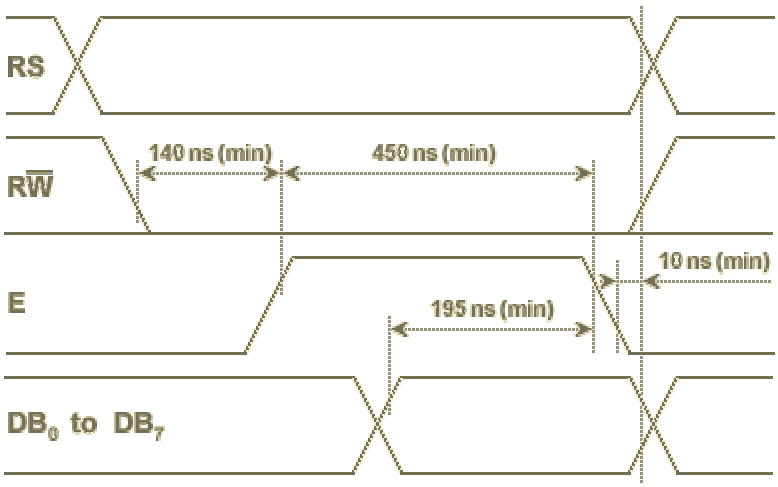

Рис. 2: Запись данных в ЖКИ

Для того, чтобы определить, когда контроллер ЖКИ закончит свои внутренние операции, AVR может опрашивать BUSY-флаг индикатора (команда "чтение busy-флага"), который сбросится только тогда, когда контроллер ЖКИ освободится. Второй, более простой способ заключается в том, что управляющий микроконтроллер, зная, сколько времени требуется ЖКИ на обработку той или иной команды, просто выполняет временную задержку после каждой передачи информации.

Если во время цикла записи AVR-микроконтроллер передает в контроллер индикатора код команды, то этот код записывается в регистр команд контроллера ЖКИ, и команда сразу же начинает выполняться. Если AVR-микроконтроллер передает в контроллер ЖКИ данные, которые представляют собой ASCII-коды отображаемых символов, то они записываются в буфер данных (DDRAM), который обычно содержит 80 ячеек (**рис. 3**). При записи или считывании буфера данных обращение осуществляется к ячейке, на которую в данный момент указывает курсор.

Таблица 2: Система команд контроллера HD4478

Код										Описание команды	Время исполнения команды (f_{osc}= 250кГ
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
0	0	0	0	0	0	0	0	0	1	Очистить дисплей и установить курсор в нулевую позицию (адрес 0)	82мкс до 1.64мс
0	0	0	0	0	0	0	0	1	*	Установить курсор в нулевую позицию (адрес 0). Установить дисплей относительно буфера DDRAM в начальную позицию. Содержимое DDRAM при этом не меняется.	40мкс до 1.6м
0	0	0	0	0	0	0	1	I/D	S	Установить направление сдвига курсора вправо (I/D=1) или влево (I/D=0) при записи/чтении очередного кода в	40мкс

										DDRAM. Разрешить (S=1) сдвиг дисплея вместе со сдвигом курсора.	
0	0	0	0	0	0	1	D	C	B	Включить(D=1)/выключить(D=0) дисплей. Зажечь(C=1)/погасить(C=0) курсор. Изображение курсора сделать мигающим (B=1).	40мкс
0	0	0	0	0	1	S/C	R/L	*	*	Переместить курсор (S/C=0) или сдвинуть дисплей (S/C=1) вправо (R/L=1) или влево(R/L=0).	40мкс
0	0	0	0	1	DL	N	F	*	*	Установить разрядность шины данных 4 бита (DL=0) или 8 бит (DL=1), количество строк дисплея - одна (N=0) или две (N=1), шрифт - 5x7 точек (F=0) или 5x10 точек (F=1).	40мкс
0	0	0	1	A_{CG}						Установка адреса CGRAM. После этой команды данные будут записываться/считываться в/из CGRAM.	40мкс
0	0	1	A_{DD}							Установка адреса DDRAM. После этой команды данные будут записываться/считываться в/из DDRAM.	40мкс
0	1	BF	AC							Чтение состояния busy-флага (BF) и счетчика адреса.	1мкс
1	0	Данные								Запись данных в DDRAM или CGRAM.	40мкс
1	1	Данные								Чтение данных из DDRAM или CGRAM.	40мкс

Примечание:
DDRAM- Display data RAM - ОЗУ ASCII-кодов, отображаемых символов
CGRAM-Character generator RAM - ОЗУ знакогенератора

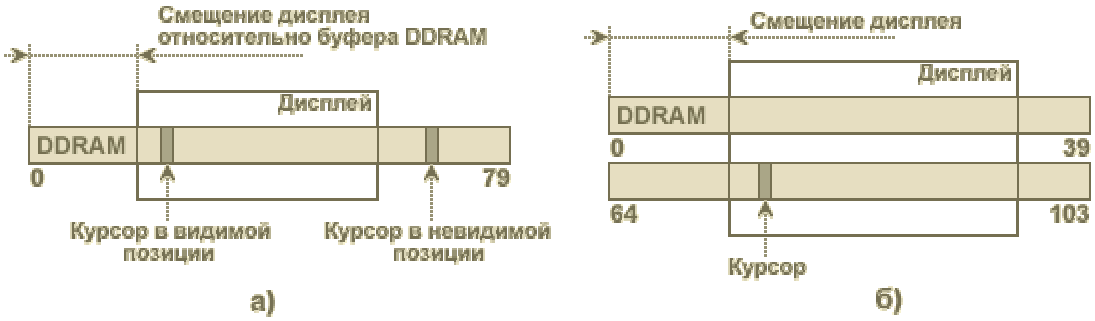

Рис. 3: Отображение на дисплее символов, ASCII-коды которых записаны в DDRAM (а - однострочный ЖКИ, б - двустрочный ЖКИ)

Буфер данных имеет больше ячеек, чем число знакомест дисплея. Смещая окно индикатора относительно буфера данных (см. систему команд), можно отображать на дисплее различные области буфера. У двустрочных индикаторов первые 40 ячеек буфера данных, обычно, отображаются на верхней строке дисплея, а вторые 40 ячеек - на нижней строке. Сдвиг окна дисплея относительно буфера данных для верхней и нижней строк происходит синхронно. Курсор будет виден на индикаторе только в том случае, если он попал в зону видимости дисплея (и если предварительно была подана команда отображать курсор).

Кроме DDRAM, контроллер ЖКИ содержит еще один блок памяти - знакогенератор. Его

"прошивка", то есть соответствие ASCII-кодов начертанию символов, обычно имеется в описании индикатора. Пример такой "прошивки" представлен в таблице 3. Знакогенератор состоит из двух частей. Основная его часть представляет собой ПЗУ (CGROM) и ее, следовательно, нельзя изменить. Вторая часть, в которой задаются начертания символов для первых 16-ти кодов таблицы знакогенератора, представляет собой перепрограммируемое ОЗУ(CGRAM). Имеется возможность задать начертание 8 символов, соответствующих кодам 0(8), 1(9), 2(10) ... 7(15). Для каждого из восьми перепрограммируемых символов в CGRAM отводится по 8 ячеек памяти, каждая из которых соответствует одной строке точек в изображении символа. Таким образом, перепрограммируемая часть знакогенератора содержит 64 байта памяти (8x8). Пример кодирования CGRAM для одной буквы приведен на **рис. 4**.

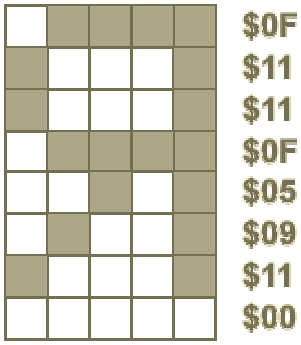

Рис. 4: Пример кодирования символа

Таблица 3: Кодовая таблица для русифицированного индикатора РС 1202-А

2. Интерфейс ЖКИ с AVR-микроконтроллером

На **рис. 5** и **рис. 7** приведены две возможные схемы сопряжения ЖКИ и AVR-микроконтроллера. В первой схеме (**рис. 5**) ЖКИ подключается к AVR как блок внешней памяти данных.

Узел формирования стробирующего сигнала E здесь выполнен так же, как и на фирменных платах STK200, STK300 и использует сигнал записи WR AVR-микроконтроллера и старший разряд адреса A15. Таким образом, информация на индикатор поступает при выполнении AVR-микроконтроллером команды записи данных во внешнюю память с любым адресом, имеющим A15=1.

На **рис. 6** представлена временная диаграмма записи данных во внешнюю память AVR-микроконтроллера.

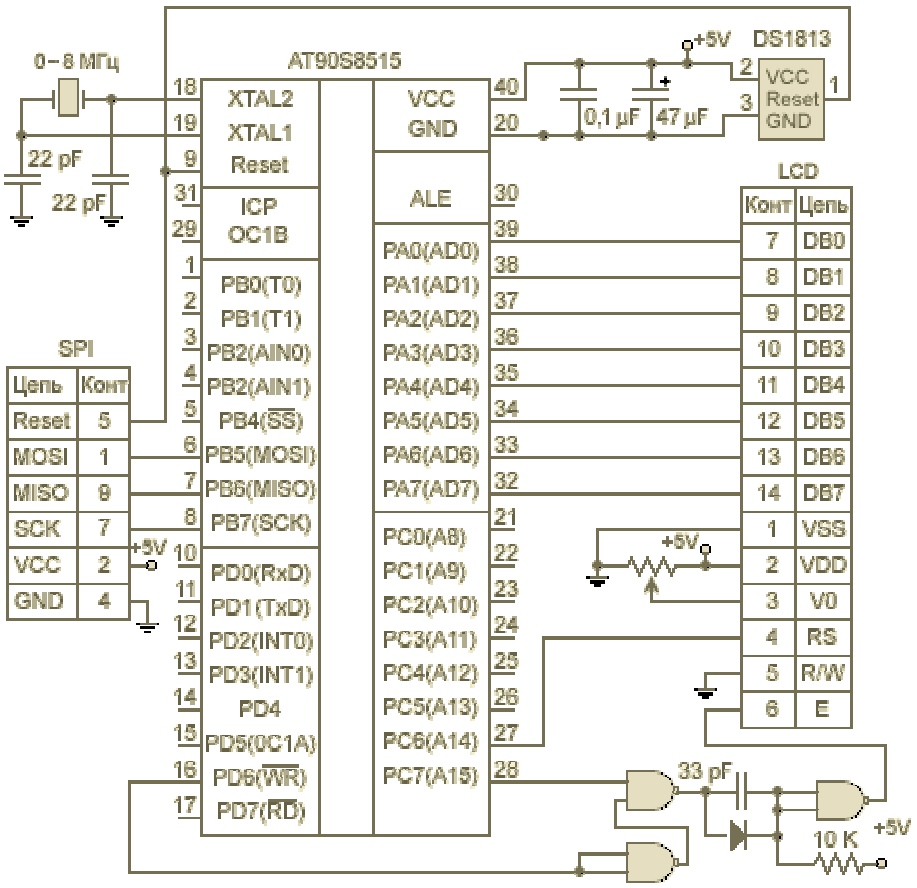

Рис. 5: Включение ЖКИ как блока внешней памяти

Сравнивая ее с диаграммой сигналов записи данных в контроллер ЖКИ (**рис. 2**), не трудно заметить, что сигналы E и WR имеют разную полярность. Различаются также временные соотношения этих сигналов с сигналами на шине данных. Поэтому узел формирования сигнала E на **рис. 5** имеет дополнительный инвертор и дифференцирующую цепочку, укорачивающую импульс записи.

В качестве сигнала выбора регистра RS в рассматриваемой схеме используется еще один разряд адреса - A14. Таким образом, операции записи в контроллер ЖКИ команд или данных отличаются между собой только адресом. Например, данные можно записывать по адресу $C000, а команды - по адресу $8000.

Вывод R/W ЖКИ в предлагаемой схеме непосредственно присоединен к линии "земля", тем самым задается единственно возможное направление передачи информации от AVR-микроконтроллера на ЖКИ. При этом потеря возможности опрашивать BUSY-флаг и считывать данные из блоков памяти контроллера ЖКИ окупается экономией одного управляющего вывода AVR.

Рассмотренная схема требует минимальных программных затрат на обслуживание ЖКИ, но, очевидно, годится только для **AT90S4414**, **AT90S8515** и **megaAVR**, которые допускают подключение внешней памяти.

На **рис. 7** изображена схема подключения ЖКИ, которую можно использовать для AVR-микроконтроллеров, не имеющих возможности подключения внешней памяти данных. В этой схеме управляющие сигналы E и RS формируются программно на обычных линиях ввода/вывода AVR. В приведенном примере шина данных состоит из 4 разрядов. Каждый байт данных при этом, как упоминалось выше, передается за две последовательные

посылки, начиная со старшей тетрады.

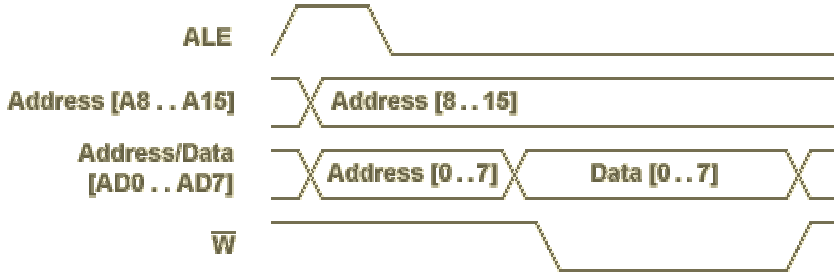

Рис. 6: Запись данных во внешнюю память AVR-микроконтроллера

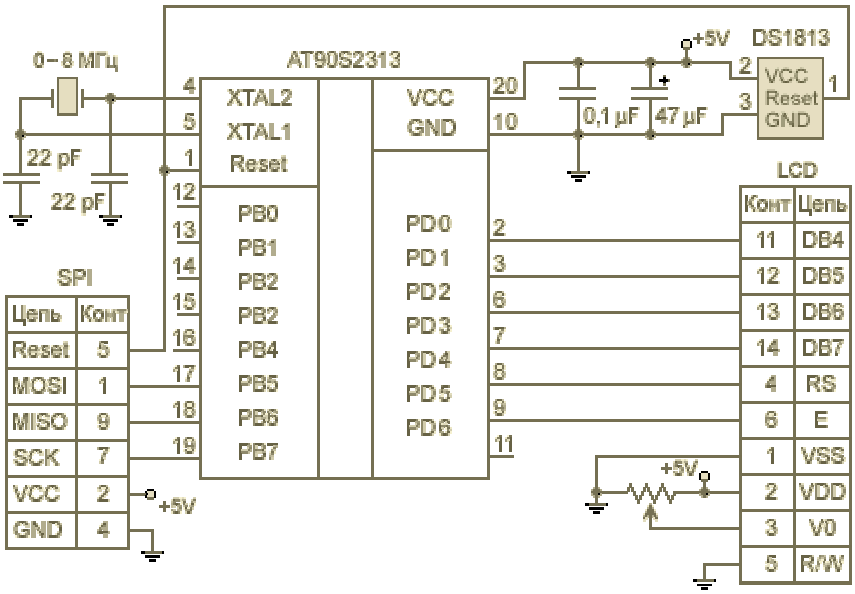

Рис. 7: Подключение ЖКИ при помощи 6 цифровых выводов

3. Программирование вывода информации на ЖКИ

Драйвером какого-либо аппаратного узла называют набор подпрограмм, которые учитывают все аппаратные особенности схемы включения этого узла и максимально облегчают работу с ним головной программе.

Простейшими составными "кирпичиками" драйвера ЖКИ могут быть подпрограммы, перечисленные в **таблице 4** и приведенные на **рис. 9** и **рис. 10**. Первые две из них ("icom_XXXX" и "idat_XXXX") для вывода в контроллер индикатора байта команды и байта данных реализуют рассмотренные ранее (**рис. 2**) временные диаграммы обмена сигналами между контроллером индикатора и AVR-микроконтроллером. Построение этих подпрограмм целиком зависит от схемы включения индикатора.

Таблица 4: Базовые подпрограммы вывода информации на ЖКИ

Назначение подпрограммы (пп)	**Название пп для схемы на рис. 5**	**Название пп для схемы на рис. 7**
Вывод байта-команды в контролер ЖКИ	icom_8515	icom_2313
Вывод байта данных в контроллер ЖКИ	idat_8515	idat_2313

Инициализация ЖКИ	initlcd_8515	initlcd_2313

Подпрограмма инициализации контроллера ЖКИ "initlcd_XXX", посылая в индикатор последовательность команд при помощи подпрограммы "icom_XXXX", задает режим работы ЖКИ. Эта подпрограмма в дальнейшем при программировании реальной задачи может дополняться зависимыми от этой задачи фрагментами. Например, в подпрограмме инициализации удобно выполнить вывод постоянного (не изменяемого во все время работы программы) текста на дисплей. Здесь же мы будем задавать начальные значения управляющим битам (флагам), при помощи которых драйвер ЖКИ будет обмениваться статусной информацией с головной программой.

Для того, чтобы лучше понять работу перечисленных подпрограмм, рассмотрим программу "runstr" (**рис. 11**) вывода на ЖКИ бегущей строки. В этой программе показано, как, перепрограммируя начальные ячейки знакогенератора, можно при помощи нерусифицированного ЖКИ выводить на дисплей русские буквы. Обратите внимание на то, что выводимый текст хранится во flash-памяти программ в виде таблицы. При извлечении данных из этой таблицы указатель адреса приходится умножать на 2. Это происходит из-за того, что память программ, в которой размещается таблица выводимых символов, имеет 16-разрядные ячейки, в каждой из которых хранится по два ASCII-кода данных. Задавая начальный адрес таблицы при помощи директивы .org, мы задаем адрес 16-разрядной ячейки. Но для команды LPM извлечения байта данных из памяти программ требуется задать адрес байта, который, очевидно, в два раза больше адреса слова.

В программе "runstr" длительность задержек между выводами на индикатор задает сам процессор, подсчитывая количество холостых операций. При этом никакой полезной работы процессор не выполняет. Понятно, что если единственной целью программы является организация бегущей строки, то используемый в программе "runstr" способ задания программных задержек является вполне приемлемым.

Рассмотрим теперь случай, когда AVR-микроконтроллер предназначен для управления каким-либо объектом, а ЖКИ используется для отображения контролируемого параметра, например, частоты, количества импульсов и т. п. В такой реальной задаче программное обеспечение всегда строится с использованием системы прерываний AVR-микроконтроллера. И в этом случае вести отсчет временных интервалов в головной программе становится невозможным, так как не известно, когда и какие происходили прерывания и сколько на их обслуживание потребовалось времени. Поэтому в подобных случаях подсчет временных интервалов возлагают на таймер. А процессор при этом освобождается для решения более интеллектуальных задач.

Схема взаимодействия головной программы и отдельных программных модулей, обслуживающих ЖКИ по прерыванию от таймера, изображена на **рис. 8**. Тексты соответствующих программ "main", "initlcd" и "tim0_lcd" приведены на **рис. 12.**

Функцией основной программы "main", так же как и раньше, является задание режима работы ЖКИ в инициализирующей части программы. Во время инициализации, когда прерывания еще не разрешены, можно воспользоваться рассмотренными выше подпрограммами задержки.

В основном цикле головной программы задачей процессора является получение очередного отсчета и сохранение этого отсчета в промежуточном буфере ЖКИ "ind_buf", который организуется в RAM. Сохранение отсчета в RAM происходит приблизительно в 100 раз быстрее, чем его вывод на индикатор, так как при этом головной программе не нужно ждать после записи каждого байта.

Подпрограмма "tim0_lcd", которую можно считать основной частью драйвера ЖКИ, вызывается по прерыванию от таймера 0 с временным интервалом, достаточным для обработки информации внутри контроллера ЖКИ (в приведенном примере около 64 мкс). Задачей подпрограммы "tim0_lcd" является вывод в ЖКИ очередного байта данных из буфера "ind_buf".

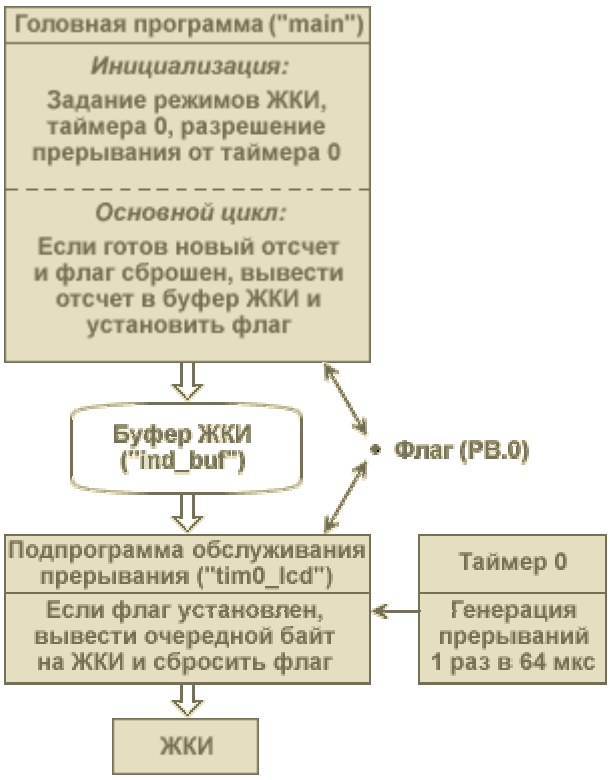

Рис. 8: Взаимодействие головной программы и драйвера ЖКИ, работающего по прерыванию

Таким образом, головная программа "main" быстро помещает результат в буфер, а подпрограмма обработки прерывания "tim0_lcd" по мере готовности ЖКИ байт за байтом выводит результат из буфера на индикатор.

Следует иметь в виду, что при использовании прерываний и организации взаимодействия программ через буфер данных, в системе возникают несколько (в нашем случае два) параллельных асинхронных процесса: головная программа работает в своем ритме, зависящем от конкретной задачи, а периодичность подпрограммы обслуживания прерывания определяется быстродействием ЖКИ.

Из-за асинхронности программ возможна ситуация, когда обе программы одновременно начнут обращаться к буферу "ind_buf". Например, после того, как на индикатор выведена только часть результата, головная программа может обновить данные в буфере. И тогда последующие байты, выводимые на индикатор, будут относиться уже к новому отсчету, что, конечно, будет являться недопустимой ошибкой. Для того, чтобы таких ситуаций не возникало, необходимо решить задачу взаимоисключения доступа программ к промежуточному буферу. Для решения этой задачи используют специальные биты управления, которые называют семафорами или флагами. В рассматриваемой программе в качестве такого флага выбран бит PB.0 и принято, что если PB.0=0, то открыт доступ к буферу для головной программы; если PB.0=1, то открыт доступ к буферу для подпрограммы обслуживания прерывания. В самый первый раз флаг устанавливается при инициализации, разрешая доступ к буферу головной программе, а затем каждая

программа, завершив работу с буфером, переключает флаг в противоположное состояние.

В заключение хочется отметить, что в статье рассмотрены только некоторые принципы управления ЖКИ, которые могут быть полезными при разработке программного обеспечения для AVR-микроконтроллеров. В зависимости от конкретной задачи драйвер ЖКИ может быть значительно видоизменен и усовершенствован. Например, можно заставить мигать отдельные поля индикатора постоянно или в зависимости от состояния объекта управления; можно рассматривать буфер как две отдельные страницы памяти между которыми, по команде с клавиатуры, например, можно быстро переключаться; можно организовывать на ЖКИ систему меню и многое другое, что подскажет Вам Ваша фантазия.

Татьяна Кривченко
E-mail: tkr@efo.spb.su