Прецизионные системы сбора данных семейства MSC12xx фирмы Texas Instruments

(часть 3)

Павел Редькин (г. Ульяновск)

Автор продолжает представлять устройства MSC12xx (см. СЭ № 2, 3, 2006). Рассмотрен порядок создания пользовательского проекта в среде IDE RIDE, а также аппаратная платформа и программные модули-драйверы для приложений на базе MSC12xx. Процесс создания пользовательских приложений на основе MSC12xx проиллюстрирован на примере одного из наиболее функционально насыщенных устройств этого семейства микросхем с максимальным объёмом встроенной памяти – MSC1211Y5. Приведён интерфейс кнопок управления и жидкокристаллического индикатора.

Создание проектов на основе MSC12xx в IDE RIDE

Интегрированная среда разработки-отладки для микроконтроллеров IDE RIDE представляет собой программный пакет, работающий под Windows Microsoft. Пакет включает в себя несколько отдельных программных продуктов: менеджер библиотек (LIB-51), редактор исходных текстов, программный отладчик-симулятор, компилятор ANSI C (RC-51), ассемблер (MA-51) и компоновщик (LX-51). Все они интегрированы в единую оболочку с удобным графическим интерфейсом. В результате инсталляции RIDE с CD на жёстком диске компьютера по умолчанию создаётся одноимённая директория, содержащая все вышеперечисленные компоненты RIDE. При щелчке мышью на иконке RIDE запускается исполняемый файл из этой директории и открывается главное окно RIDE, содержащее инструментальную панель.

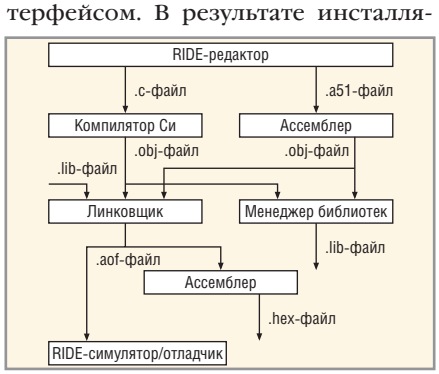

Рис. 12. Этапы разработки проекта в RIDE

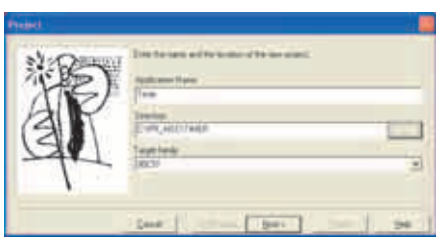

Рис. 13. Создание нового проекта в RIDE

Разработка встроенного программного обеспечения с помощью RIDE производится путём создания и редактирования пользовательских проектов. Графическая диаграмма, иллюстрирующая этапы разработки проекта с указанием используемых инструментальных средств RIDE, приведена на рис. 12.

В редакторе RIDE производится создание и редактирование файла исходного текста на Ассемблере (*.a51) или Си (*.c), который используется в дальнейшем в качестве файла источника проекта (source). Затем подготовленный файл исходного текста транслируется, если это ассемблерный файл, или компилируется, если это файл на Си. Результатом трансляции/компиляции является объектный файл (*.obj). Помимо объектного файла в проекте могут находиться подключаемые файлы для ассемблерного текста (*.a51, *.inc и т.д.) или заголовочные файлы и файлы библиотек (*.h, *.lib) для текста на Си. Компоновка всех файлов проекта и создание в результате абсолютного объектного файла (*.aof) производится компоновщиком. Полученный абсолютный объектный файл используется RIDE для создания файла кода (*.hex), который можно непосредственно загрузить в микроконтроллер, а также для отладки или симуляции написанной программы встроенными средствами RIDE. Помимо вышеперечисленных файлов, при создании проекта RIDE создаёт собственно файл проекта (*.prj), который содержит настройки конфигурации. Кроме того, в процессе компиляции/трансляции компилятор Си/Ассемблер генерирует файл листинга (*.lst), содержащий полезную информацию о размещении программы в памяти, которая может быть использована разработчиком при её отладке.

Порядок создания пользовательского проекта в RIDE может быть следующим. До начала создания проекта следует создать на жёстком диске директорию, в которой будут размещаться файлы проекта. Рекомендуется задать имя директории, совпадающее с именем создаваемого проекта. В директорию рекомендуется поместить все файлы, которые будут использованы файлом исходного текста будущего проекта в качестве подключаемых и заголовочных файлов. Затем в главном окне RIDE следует выбрать в меню Project опцию New, после чего откроется окно Project, показанное на рис. 13. В поле Application Name этого окна следует задать имя создаваемого проекта, а в поле Directory – указать путь к ранее созданной директории размещения файлов проекта, используя для этого дерево файлов и каталогов, открывающееся с помощью кнопки «...». После задания указанных параметров следует щёлкнуть левой клавишей мыши на кнопке Next. Откроется окно Target, в котором необходимо выбрать аппаратную платформу проекта (микроконтроллер), например MSC1211Y5, пометив его курсором, как показано на рис. 14. После выбора платформы следует щёлкнуть левой клавишей «мыши» на кнопке Finish, после чего откроется изменённое главное окно RIDE.

Теперь следует создать файл источника проекта. Для этого необходимо выбрать в меню File опцию New, а затем в открывшемся меню выбрать тип создаваемого файла источника: файл на Ассемблере (опция Assembler Files) или файл на Си (опция C Files). В открывшемся окне untitled с расширением соответственно *.a51 или *.c средствами редактора RIDE можно набрать файл исходного текста или выгрузить в это окно содержимое программного буфера с ранее помещённым туда файлом исходного текста. Затем созданный файл следует сохранить в директории проекта и под именем проекта, но с соответствующим расширением *.a51 или *.c, используя меню File и опцию Save as.

После этого необходимо задать созданный и сохранённый файл исходного текста в качестве файла источника проекта. Для этого следует в меню Project выбрать опцию Add node Source/Application. В появившемся дереве файлов и каталогов следует выбрать ранее созданный файл и щёлкнуть левой клавишей мыши на кнопке Open. При этом в окне дерева проекта RIDE появится заданный файл источника с указанием в квадратных скобках инструментальной программы его обработки, например tim.a51 [MA51].

На этом создание проекта закончено, и теперь можно перейти к работе с ним. Прежде всего, созданный проект следует оттранслировать/откомпилировать. Для этого в меню Project следует выбрать опцию Build all. После завершения трансляции в окне Make в нижней части экрана появятся сообщения об успешном создании hex-файла или об ошибках и замечаниях, обнаруженных в ходе трансляции. В случае появления сообщений об ошибках можно дважды щёлкнуть левой клавишей мыши на сообщении об ошибке, и курсор в окне файла исходного текста или подключаемого файла (соответствующий подключаемый файл откроется автоматически) укажет на местонахождение ошибки (будет выделено цветом). В результате безошибочной трансляции в директории проекта будет создан hex-файл, пригодный к непосредственной загрузке во Flash-память микроконтроллера.

Аппаратная платформа для приложений на основе MSC12xx

Процесс создания пользовательских приложений на основе MSC12xx проиллюстрируем на примере одного из наиболее функционально насыщенных устройств этого семейства микросхем с максимальным объёмом встроенной памяти – MSC1211Y5. Топологическая схема корпуса MSC1211 с указанием номеров и названий выводов показана на рис. 15. Функциональное назначение всех выводов устройства приведено в табл. 9.

Для упрощения внедрения и освоения устройств MSC12xx фирма TI предлагает готовые отладочные комплекты. В комплект входит печатная плата и компакт-диск. На плате установлено устройство MSC12xx, внешние компоненты схемы его включения и стабилизаторы источников питания. На диске находится программное обеспечение разработки-отладки. На сайте [5] приведено описание отладочного комплекта MSC1211EVM. На отладочной плате из этого комплекта установлены следующие основные компоненты:

- устройство MSC1211Y5;
- два стабилизированных источника питания +5 В для питания цифровой и аналоговой частей схемы;
- два разъёма портов RS-232 (подключенные к портам USART0 и USART1 MSC1211) с микросхемой-адаптером уровней RS-232 MAX3243;
- внешний кварцевый резонатор;
- внешнее ОЗУ ёмкостью 128 Кб с буферными и рабочими регистрами,
- звуковой излучатель;

Рис. 14. Выбор платформы для проекта

- микросхемы-супервизоры сброса (TPS3837L30DBVT) и перехода в режим программирования (TPS3838L30DBVT);
- клеммные колодки для подключения внешних цепей к входам встроенного АЦП MSC1211;
- перемычки и DIP-переключатели для задания конфигурации платы;
- разъёмы портов ввода/вывода общего назначения MSC1211;
- микросхема дополнительного АЦП ADS8325 с интерфейсом SPI;
- микросхема памяти EEPROM 24LC256 с интерфейсом I²C;
- кнопки RESET и LOAD соответственно для сброса и перехода в режим внутрисхемного программирования Flash-памяти.

Внутрисхемное программирование Flash-памяти MSC1211 производится с помощью программы-загрузчика TI Downloader через кабель RS-232 и микросхему-адаптер RS-232, подключенный к порту USART0.

Аналоговая и цифровая «земли» реализованы в MSC1211EVM в виде единой «земляной» шины, представляющей со-

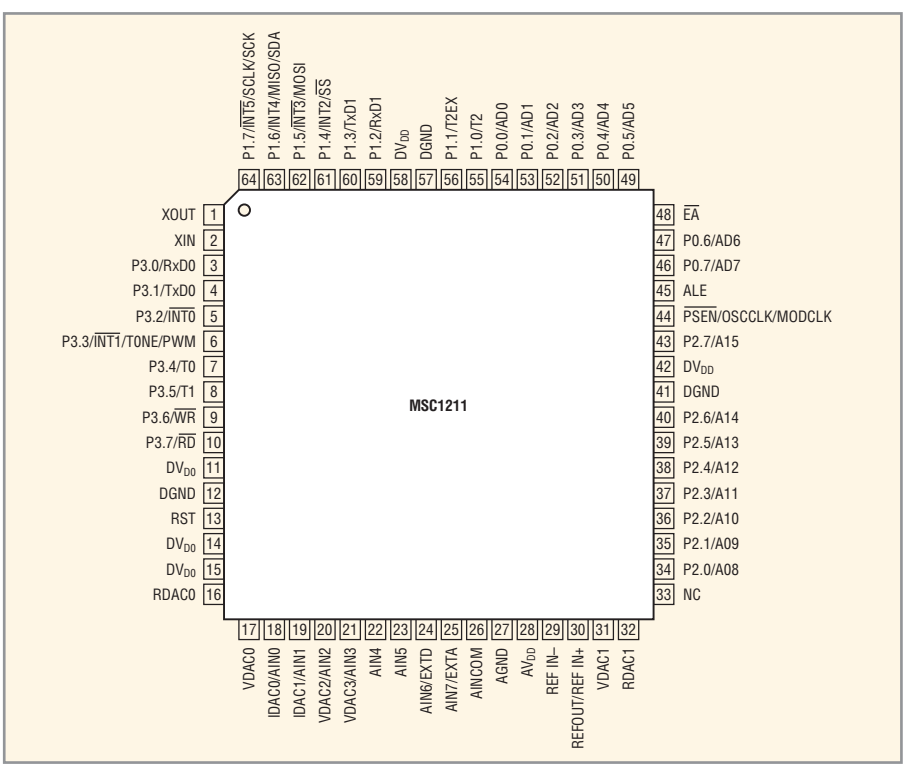

Рис. 15. Цоколёвка корпуса микросхемы MSC12xx

Таблица 9. Описание выводов MSC1211

Номер ножки	Название	Описание		
1	XOUT	Выход усилителя встроенного генератора, предназначенный для подключения параллельного кварцевого или керамического резонатора типа «абсолютное время» (AT)		
2	XIN	Вход усилителя встроенного генератора, предназначенный для подключения параллельного кварцевого или керамического резонатора типа AT. Также на этот вход можно подавать тактовый сигнал от внешнего источника		
3 – 10	P3.0 – P3.7	Линии двунаправленного порта ввода/вывода 3. Эти линии имеют альтернативные функции, которые описаны ниже		
		Линия порта	Альтернативная функция	Режим
		P3.0	RxD0	Вход последовательного порта 0
		P3.1	TxD0	Выход последовательного порта 0
		P3.2	$\overline{\text{INT0}}$	Вход внешнего прерывания 0
		P3.3	$\overline{\text{INT1}}$/TONE/PWM	Вход внешнего прерывания 1, выход генератора тонального сигнала/ШИМ
		P3.4	T0	Внешний вход таймера 0
		P3.5	T1	Внешний вход таймера 1
		P3.6	$\overline{\text{WR}}$	Выход стробирования записи во внешнюю память данных
		P3.7	$\overline{\text{RD}}$	Выход стробирования чтения из внешней памяти данных
11, 14, 15, 42, 58	DVdd	Входы подключения цифрового источника питания		
12, 41, 57	DGND	Входы подключения цифрового заземления		
13	RST	Вход сброса. Высокий уровень на этом входе, удерживаемый в течение двух тактовых циклов, вызовет сброс устройства		
16	RDAC0	Выход ЦАП RDAC0		
17	VDAC0	Выход ЦАП VDAC0		
27	AGND	Входы подключения аналогового заземления		
18	IDAC0/AIN0	Выход ЦАП IDAC0/Аналоговый вход канала 0		
19	IDAC1/AIN1	Выход ЦАП IDAC1/Аналоговый вход канала 1		
20	VDAC2/AIN2	Выход ЦАП VDAC2/Аналоговый вход канала 2		
21	VDAC3/AIN3	Выход ЦАП VDAC3/Аналоговый вход канала 3		
22	AIN4	Аналоговый вход канала 4		
23	AIN5	Аналоговый вход канала 5		
24	AIN6, EXTD	Аналоговый вход канала 6, вход детектора пониженного напряжения цифрового источника		
25	AIN7, EXTA	Аналоговый вход канала 7, вход детектора пониженного напряжения аналогового источника		
26	AINCOM	Аналоговый вход (может использоваться как общий провод для несимметричных аналоговых входов или как один из входов в дифференциальной паре)		
28	AVdd	Входы подключения аналогового источника питания		
29	REFIN–	Вход подключения отрицательного полюса внешнего ИОН		
30	REFOUT/REFIN+	Вход подключения положительного полюса внешнего ИОН/Выход встроенного ИОН		
31	VDAC1	Выход ЦАП VDAC1		
32	RDAC1	Выход ЦАП RDAC1		
33	NC	Не используется		
34 – 40, 43	P2.0 – P2.7	Линии двунаправленного порта ввода/вывода 2. Эти линии имеют альтернативные функции, которые описаны ниже		
		Линия порта	Альтернативная функция	Режим
		P2.0	A8	Бит 8 адреса
		P2.1	A9	Бит 9 адреса
		P2.2	A10	Бит 10 адреса
		P2.3	A11	Бит 11 адреса
		P2.4	A12	Бит 12 адреса
		P2.5	A13	Бит 13 адреса
		P2.6	A14	Бит 14 адреса
		P2.7	A15	Бит 15 адреса
44	$\overline{\text{PSEN}}$, OSCCLK, MODCLK	Выход разрешения памяти программ. Связывает устройство с дополнительной внешней памятью. Сигнал $\overline{\text{PSEN}}$ имеет активный низкий уровень. В режиме программирования линия $\overline{\text{PSEN}}$ является входом (как и линия ALE) и используется для определения, какой режим программирования (параллельный или последовательный) установлен. На вход $\overline{\text{PSEN}}$ следует подать высокий уровень для установки режима параллельного программирования и низкий – для режима последовательного программирования. В случае, если внешняя память программ не используется, линия $\overline{\text{PSEN}}$ может быть соединена внутри устройства с выходом генератора тактовых импульсов, генератора модулятора или установлена в высокое или низкое состояние		
		ALE	$\overline{\text{PSEN}}$	Режим программирования, выбираемый в ходе сброса
		NC	NC	Нормальный режим
		0	NC	Параллельное программирование
		NC	0	Последовательное программирование
		0	0	Зарезервировано
45	ALE	Выход разрешения фиксации адреса. Используется для «защелкивания» младшего байта адреса в цикле доступа к внешней памяти. Частота генерации периодического сигнала ALE постоянна, равна 1/4 частоты тактового генератора и может использоваться для тактирования или синхронизации внешних устройств. Один импульс сигнал ALE генерируется в каждом цикле доступа к вешней памяти данных. В режиме программирования линия ALE является входом (как и линия $\overline{\text{PSEN}}$) и используется для определения, какой режим программирования (параллельный или последовательный) установлен. На вход ALE следует подать высокий уровень для установки режима последовательного программирования и низкий – для режима параллельного программирования		
48	$\overline{\text{EA}}$	Вход разрешения внешнего доступа. На входе $\overline{\text{EA}}$ необходимо установить низкий уровень, чтобы разрешить устройству выполнение кода из внешней памяти программ с адреса 0000h		
46, 47, 49 – 54	P0.0 – P0.7	Линии двунаправленного порта ввода/вывода 0. Эти линии имеют альтернативные функции, которые описаны ниже		
		Линия порта	Альтернативная функция	Режим
		P0.0	AD0	Бит 0 адреса/данных
		P0.1	AD1	Бит 1 адреса/данных
		P0.2	AD2	Бит 2 адреса/данных
		P0.3	AD3	Бит 3 адреса/данных
		P0.4	AD4	Бит 4 адреса/данных
		P0.5	AD5	Бит 5 адреса/данных
		P0.6	AD6	Бит 6 адреса/данных
		P0.7	AD7	Бит 7 адреса/данных
55, 56, 59 – 64	P1.0 – P1.7	Линии двунаправленного порта ввода/вывода 1 (конфигурирование выводов порта производится с помощью PCH P1DDRL и P1DDRH). Эти линии имеют альтернативные функции, которые описаны ниже		
		Линия порта	Альтернативная функция	Режим
		P1.0	T2	Вход T2
		P1.1	T2EX	Внешний вход T2
		P1.2	RxD1	Вход последовательного порта
		P1.3	TxD1	Выход последовательного порта
		P1.4	INT2/SS	Вход внешнего прерывания 2, линия «выбор ведомого»
		P1.5	$\overline{\text{INT3}}$/MOSI	Вход внешнего прерывания 3, линия MOSI SPI
		P1.6	INT4/MISO/SDA	Вход внешнего прерывания 4, линия MISO SPI, SDA I^2C
		P1.7	$\overline{\text{INT5}}$/SCK/SCLK	Вход внешнего прерывания 5, линия «последовательный синхросигнал»

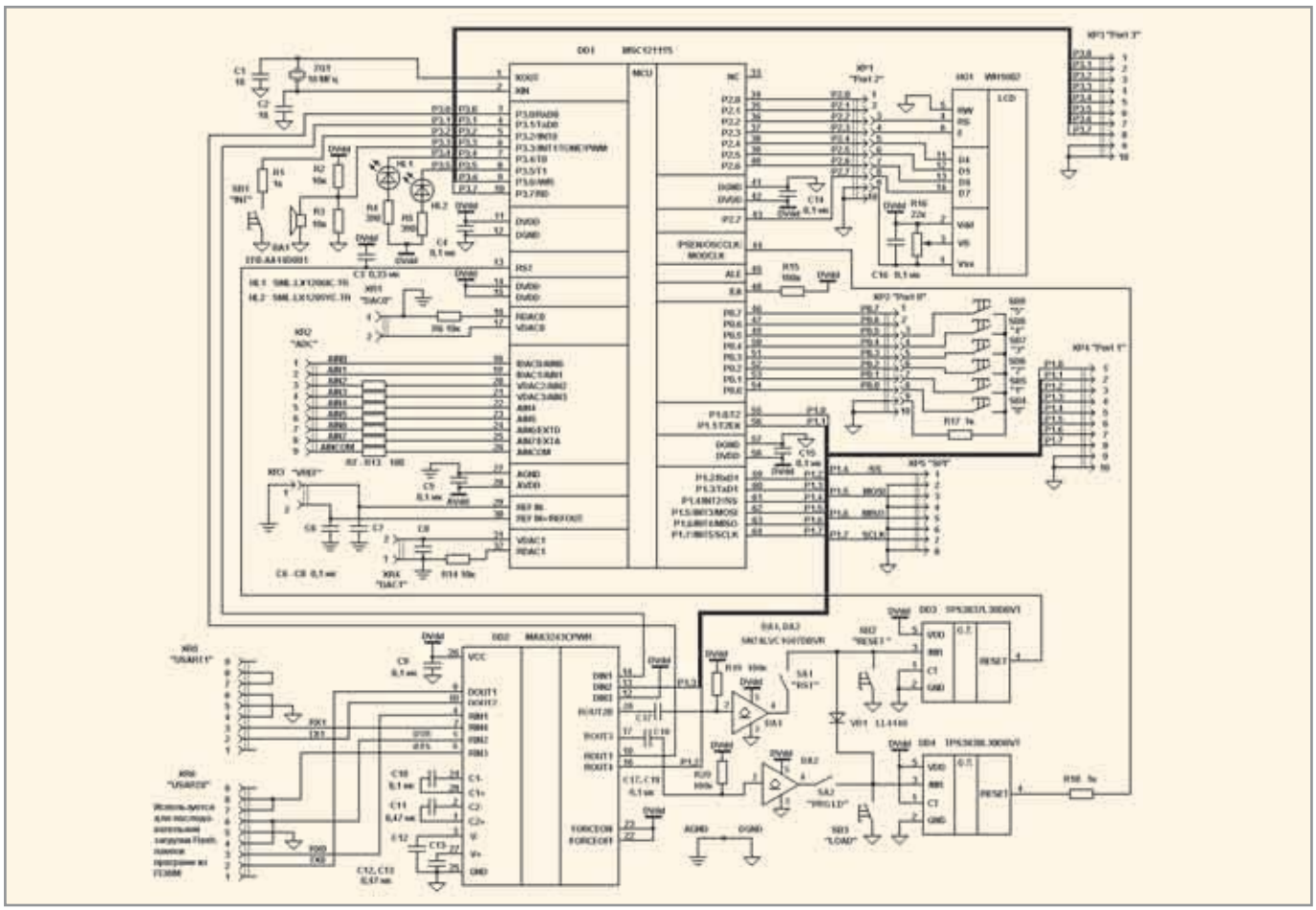

Рис. 16. Принципиальная схема отладочной платы с MSC12xx

бой слой фольги на стороне платы, свободной от компонентов. Упрощённая принципиальная схема отладочной платы MSC1211EVM (без источников питания) приведена на рис. 16. Принципиальная схема стабилизированных источников питания MSC1211EVM приведена на рис. 17. Элементы, реально установленные на плате, но не используемые встроенным программным обеспечением, которое будет описано ниже (например, внешнее ОЗУ с его регистрами, АЦП ADS8325, память 24LC256, а также большинство DIP-переключателей), на схеме рис. 16 не показаны. В то же время к компонентам, установленным на плате, могут быть подключены дополнительные внешние компоненты, не входящие в состав отладочного комплекта. Например, автор статьи подключил к разъёмам XP2 Port 0 и XP1 Port 2 схемы (рис. 16) линейку кнопок управления SB4 – SB9 и символьный жидкокристаллический индикатор со встроенным контроллером HG1 соответственно. Программные интерфейсы MSC12xx для этих устройств будут описаны ниже. Вывод REFIN– встроенного ИОН подключен к аналоговой «земле», таким образом, опорное напряжение на АЦП MSC1211 подаётся относительно «земли». Тактирование MSC1211 осуществляется от внутреннего генератора устройства с внешним кварцевым резонатором ZQ1 на частоту 16 МГц.

Показанные на схеме на рис. 16 цепи DTR, RTS разъёма XR6 USART0 через микросхему-адаптер RS-232 DD2, конденсаторы C17, C18, буферные элементы DA1, DA2 и DIP-переключатели SA1, SA2 подключены ко входам микросхем-супервизоров сброса (TPS3837L30DBVT) и перехода в режим программирования (TPS3838L30DBVT) соответственно. Выходы DA1 и DA2 подключены параллельно кнопкам SB2 RESET («Сброс») и SB3 LOAD («Загрузка»). Это даёт возможность производить сброс микроконтроллера и загрузку его встроенной Flash-памяти с помощью сигналов DTR и RTS COM-порта. При этом COM-порт должен быть подключён к отладочной плате компьютера (терминала). Для внутрисхемного программирования Flash-памяти с помощью программы-загрузчика TI Downloader, а также для работы порта USART0 в составе пользовательского приложения в этих цепях

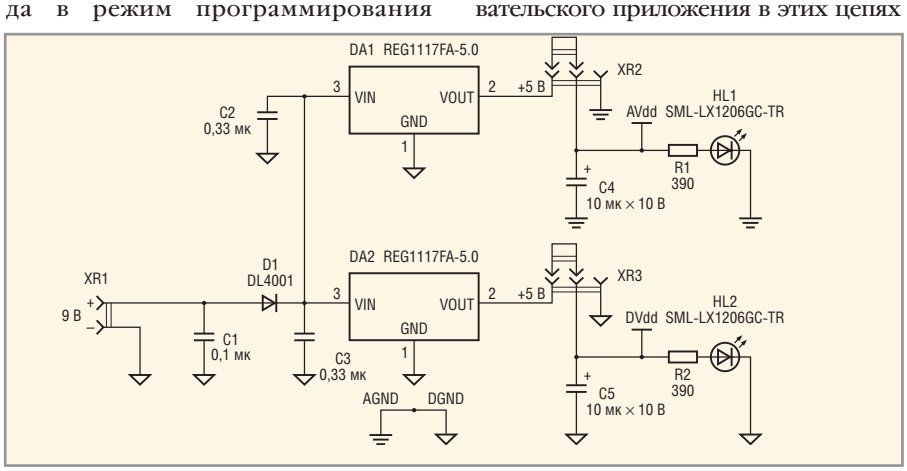

Рис. 17. Принципиальная схема источников питания отладочной платы

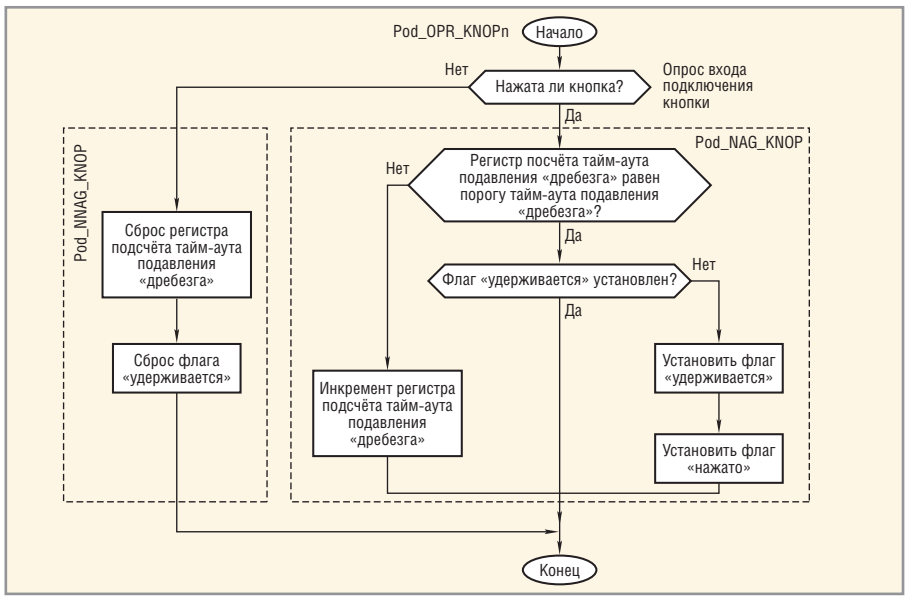

Рис. 18. Блок-схема алгоритма интерфейса опроса кнопки управления Pod_OPR_KNOPn

нет необходимости, и поэтому выключатели SA1, SA2 в указанных режимах могут быть разомкнуты.

Следует отметить, что многие примеры встроенных управляющих программ для MSC12xx, разработанные производителем, предусматривают в качестве интерфейса ввода/вывода использование компьютерной терминальной программы с условием постоянного подключения системы к компьютеру (терминалу) через встроенный порт USART. Ввод информации в систему производится с клавиатуры компьютера с контролем вводимых данных в передающем окне терминальной программы, а вывод – через приёмное окно терминальной программы. Указанный подход весьма удобен и значительно упрощает встроенное программное обеспечение MSC12xx, однако системы с таким пользовательским интерфейсом не являются автономными. Чтобы преодолеть этот недостаток, для MSC12xx был разработан и отлажен программный интерфейс ввода/вывода кнопок и ЖКИ [2]. Это позволило сделать систему на базе MSC12xx полностью автономной. Помимо программных модулей поддержки кнопок и ЖКИ в качестве подключаемых файлов также использован файл стандартных мнемоник Reg1211.inc, поставляемый в составе IDE RIDE, и файл дополнительных мнемоник P0_P2_A.inc, которые доступны на сайте журнала.

Интерфейс кнопок управления

Предлагаемый интерфейс кнопок управления для MSC12xx обеспечивает программное подавление «дребезга» механических контактов. Никаких дополнительных внешних элементов для этого не требуется. Блок-схема алгоритма интерфейса обслуживания одной кнопки приведена на рис. 18. Алгоритм реализован в подпрограмму, которая отслеживает моменты нажатия и отпускания опрашиваемой кнопки. При этом производится фильтрация импульсов «дребезга» механических контактов кнопки.

Для построения программного интерфейса одной кнопки разработчику необходимо выделить следующие ресурсы микроконтроллера целевой системы: один регистр общего назначения, служащий для подсчёта тайм-аута подавления «дребезга» кнопки, и два пользовательских флага (бита общего назначения), служащие для индикации момента нажатия на кнопку (флаг «нажато») и длительности удержания кнопки (флаг «удерживается»). Кроме того, подпрограмма использует некоторую константу, задающую значение порога тайм-аута подавления «дребезга».

Для корректной работы кнопочного интерфейса при первоначальной инициализации флаги «нажато» и «удерживается» всех опрашиваемых кнопок должны быть сброшены. Как можно видеть из блок-схемы, при каждом вызове подпрограммы при условии, что опрашиваемая кнопка не нажата, регистр подсчёта тайм-аута и флаг «удерживается» будут сбрасываться, т.е., будет происходить постоянная установка цифрового программного фильтра «дребезга» в начальное состояние. Следует заметить, что вызовы подпрограммы должны производиться циклически с частотой от нескольких десятков до нескольких сотен герц, например, по прерыванию от какого-нибудь таймера. Когда кнопка нажимается, содержимое регистра подсчёта тайм-аута начинает инкрементироваться при каждом вызове подпрограммы, причём устойчиво увеличиваться оно станет только после окончания «дребезга», когда регистр перестанет сбрасываться из-за пауз между импульсами «дребезга». Спустя некоторое время после окончания импульсов «дребезга» при циклических вызовах подпрограммы опроса наступит момент, когда содержимое регистра подсчёта тайм-аута станет равным константе значения порога тайм-аута и, следовательно, произойдет установка флагов «нажато» и «удерживается» опрашиваемой кнопки.

Состояние флага «удерживается» прямо или инверсно отображает текущее состояние входа подключения соответствующей ему кнопки с учётом подавления импульсов «дребезга», т.е. он будет оставаться в единичном состоянии столько времени, сколько удерживается кнопка после окончания «дребезга», и сбросится при первом же импульсе «дребезга» в момент отпускания кнопки. Этот флаг можно использовать в программе для реализации каких-то дополнительных функций данной кнопки в интерфейсе ввода разрабатываемого устройства, например, для задания какого-либо режима, условием которого служит текущее нажатое состояние этой кнопки.

Флаг «нажато» в предлагаемом алгоритме обладает свойством «защёлкивания», т.е. при отпускании кнопки он в подпрограмме не сбрасывается. Опрос состояния с целью обнаружения установки и последующего сброса флага «нажато» возлагается на ту часть основной программы, которая производит обработку нажатия на данную кнопку.

Конкретное значение константы порога тайм-аута «дребезга» может задаваться разработчиком для обеспечения требуемого времени реакции устройства на нажатую кнопку в зависимости от значения периода вызова подпрограммы кнопочного интерфейса и конкретного типа (линейных размеров) кнопки.

Исходные тексты подпрограмм, написанных на Ассемблере и реализующих описанный выше алгоритм обслуживания кнопок, содержатся в

файле knop.a51, который доступен на сайте журнала.

Далее этот файл будет указываться как подключаемый в исходных текстах пользовательских программ для MSC12xx. Предложенный программный код может использоваться с любым 8051-совместимым микроконтроллером.

В файле содержатся подпрограммы обслуживания шести кнопок, каждая из которых подключена к отдельной линии ввода/вывода общего назначения стандарта 8051. Эти линии в подпрограммах имеют имена _IN_KNOP0 – _IN_KNOP5. Для оптимизации кода каждая ветвь управления в подпрограммах Pod_OPR_KNOP0 – Pod_OPR_KNOP5 реализована подпрограммами следующего уровня вложенности вызова, одинаковыми для всех кнопок:

- Pod_NNAG_KNOP – последовательность действий, производимых алгоритмом в случае не нажатой кнопки;
- Pod_NAG_KNOP – последовательность действий, производимых алгоритмом в случае нажатой кнопки.

При этом предполагается, что кнопки являются нормально разомкнутыми и включены между входами MSC12xx и общим проводом устройства, а линии подключения кнопок сконфигурированы в микроконтроллере как линии ввода/вывода стандарта 8051 (подтянуты к «плюсу» источника питания внутренними резисторами). Перед тем как вызвать подпрограмму обслуживания какой-либо кнопки, следует в основной программе косвенно адресовать регистр подсчёта тайм-аута этой кнопки с помощью указателя R1, а регистр, содержащий флаги «нажато» и «удерживается» этой кнопки, – с помощью указателя R0. Флаг «удерживается» – младший значащий бит (МЗБ) в регистре с флагами, флаг «нажато» – МЗБ+1 (этот регистр может и не иметь побитную адресацию). Константа значения порога тайм-аута «дребезга» в подпрограмме имеет имя POROG_K.

Интерфейс ЖКИ

Интерфейс вывода в устройстве на базе микроконтроллера в большинстве случаев должен включать в себя какие-то аппаратные средства индикации с соответствующей программной поддержкой. В настоящее время на рынке индикаторов весьма широко распространены символьные (буквенно-цифровые) жидкокристаллические модули со встроенным контроллером управления, совместимым с контроллером HD44780 фирмы Hitachi. Они выпускаются несколькими производителями в различных исполнениях и модификациях. Эти индикаторы имеют малое энергопотребление, простой интерфейс (для взаимодействия с микроконтроллером им требуется от шести до одиннадцати линий, не считая питания и общей шины) и предоставляют очень широкие возможности в плане отображения информации.

В данном проекте использован индикатор WH1602 фирмы Winstar (2 строки, 16 символов в строке, русифицированный). Таблицу фонтов для этого и для других русифицированных ЖКИ можно найти в книге [4], а электрические параметры, временные диаграммы сигналов управления и набор команд – в [6]. Передача информации из микроконтроллера в HD44780-совместимый индикатор возможна по параллельной 8- или 4-проводной шине данных. В первом случае передача производится быстрее, но дополнительно занимаются четыре линии ввода/вывода микроконтроллера. Логический уровень на входе ЖКИ, определяющий его режим (запись/чтение) RW (выв. 5 HG1), аппаратно установлен для режима записи (подключен к общей шине), как показано на рис. 16. По этой причине готовность ЖКИ к приёму данных микроконтроллер не проверяет, а вместо этого программно генерирует временные задержки, заведомо большие интервалов времени, необходимых ЖКИ на подготовку к приёму.

Исходные тексты набора подпрограмм, реализующих интерфейс обмена с ЖКИ по 4-проводной шине данных без опроса его состояния, содержатся в файле lcd_del.a51. Подпрограммы обслуживания HD44780-совместимого ЖКИ для MSC12xx доступны на сайте журнала.

Далее этот файл будет указываться как подключаемый в исходных текстах программ для MSC12xx. В файле содержатся следующие подпрограммы:

- подпрограмма начальной инициализации ЖКИ после сброса Pod_INIT_LCD;
- подпрограмма очистки экрана ЖКИ Pod_CLEAR_LCD;
- подпрограмма передачи в ЖКИ одной команды Pod_PER_COM_LCD;
- подпрограмма записи в ОЗУ ЖКИ одного байта данных (индикации на экране ЖКИ одного символа) по произвольному адресу ОЗУ ЖКИ Pod_PER_DAT_LCD;
- подпрограмма записи в ОЗУ ЖКИ одного байта данных (индикации на экране ЖКИ одного символа) по текущему адресу ОЗУ ЖКИ Pod_TEK_DAT_LCD;
- вспомогательные подпрограммы следующего уровня вложенности.

Продолжение следует

Литература

1. www.ti.com.
2. *Редькин П.П.* Прецизионные системы сбора данных семейства MSC12xx Texas Instruments: архитектура, программирование, разработка приложений (+CD). М.: Додэка-XXI, 2006.
3. www.analog.com.
4. *Фрунзе А.В.* Микроконтроллеры? Это же просто! В 3-х т. М.: СКИМЕН, 2003.
5. MSC1211 Precision ADC with 8051 Microcontroller and Flash Memory Evaluation Module. SBAU086, 2003, Texas Instruments Incorporated (www.ti.com).
6. Жидкокристаллические индикаторы фирмы DATA International. Библиотека электронных компонентов. Вып. 8. М.: Додэка, 1999.