ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
СРЕДНЕГО (ПОЛНОГО) ОБЩЕГО ОБРАЗОВАНИЯ
ЛИЦЕЙ ПРИ СПБГУТ

Разработка в рамках программы «Умная школа»:
«Интеллектуальный эксперимент «Умный маятник»

Богураев М.В., Флаксман Д.А.

«Рабочие материалы за октябрь 2010 по разработке «Умный маятник»»

Санкт - Петербург
2010

Богураев М.В., Флаксман Д.А. Рабочие материалы за октябрь 2010 по разработке «Умный маятник». СПб: ГОУ «Лицей при СПбГУТ», 2010.

Рабочие материалы за октябрь 2010 по разработке «Умный маятник»

Аппаратное обеспечение.

Аппаратное обеспечение, измененное в августе (добавлен второй канал), видимо придётся подвергнуть доработке: установить кварцевый резонатор для первого таймера – это даст возможность измерять время в реальном масштабе (обсуждалось в конце сентября). Схема электрическая принципиальная изображена на вкладке.

Программное обеспечение.

Предварительные соображения

Испытания маятника в конце сентября показали неработоспособность программ. Потом и в моей и в Диминой (Дима Флаксман) программе обнаружились ошибки. В моей программе я обнаружил с десяток ошибок связанных с неправильной расстановкой меток, в том числе и в программе настройки нулевого таймера (программа в бесконечном цикле). Ещё я обнаружил алгоритмическую ошибку, которая не позволяла контроллеру после получения символа «r» переходить к анализу символа «e». Так же выяснилось, что переменная состояния X1 должна быть не мьютексом, а счётным семафором и иметь не два состояния, а три: ожидание символа «r» X1 = 0; работа X1 = r; завершение работы X1 = «e». После завершения работы X1 должно изменяться с «e» на «0», иначе контроллер всё время посылал 0D04 – этого нет в протоколе передачи, может вызвать неправильную работу компьютерной программы.

Второго октября исправлен алгоритм, переписана программа. Настройки периферийных модулей остались неизменными.

Пятого октября в ходе испытаний выяснилось, что нужно изменить протокол передачи данных. Были изменены ключи каналов и изменён порядок следования ключей. Ключи были изменены из тех соображений, что первый байт передаётся ADRESH, выравнивание результата АЦП левое, поэтому появление в первом байте числа больше 3 ситуация невозможная. Поэтому первый байт в ключах много больше трёх. Второй байт может быть любым, поэтому вторые байты ключей выбраны по произволу.

Одиннадцатого октября после очередных испытаний было выяснено, что нарушается протокол передачи, то есть: после FD FA передаётся FD F4 и Димина программа сбоит (после последней точки пропадает изображение – масштаб резко уменьшается). Было принято решение, о запрете прерываний в микроконтроллере на время передачи пакета. После этого нововведения проблема была исправлена. Пришлось немного переделать алгоритм (рис. 5) и подправить программу микроконтроллера.

Так же остаются открытыми два вопроса:

1. По какому маятнику стартует программа?
2. По какому маятнику заканчивается программа?

2.1.1. Пока не совсем понятно, какой ответ нужно выбрать в качестве удовлетворительного

1. Программа стартует после пересечения нуля каким-нибудь из маятников. А завершается после того, как самый длительно работающий маятник останавливается. Программа должна понять, что если один из маятников не запускался, то закончить нужно по тому маятнику, который был запущен.
2. Программа стартует по первому маятнику. А завершается после того, как самый длительно работающий маятник останавливается. Программа должна понять, что если один из маятников не запускался, то закончить нужно по тому маятнику, который был запущен.

3. Программа стартует по второму маятнику. А завершается после того, как самый длительно работающий маятник останавливается.
4. Программа стартует по первому маятнику, по первому же маятнику заканчивается.
5. Программа стартует по второму маятнику, по второму же маятнику заканчивается.
6. Записывать всё значимое на каждом канале по отдельности, а выводить… нужны маркеры реального времени.
7. Стартует по первому стоп по второму
8. Стартует по второму стоп по первому
9. Программа стартует и не заканчивается (возможен большой массив данных).
10. Частота выборки в 500 мкс может оказаться избыточной нужно предусмотреть возможность поменять это время.
11. Предусмотреть возможность передавать время, которое прошло от начала до конца эксперимента. Это время нужно передавать после 0Dh 04h? Тогда это будет максимально достоверное время для расчёта периода. Но контроллер должен будет передавать пакеты с одинаковым интервалом.

Пока решили, что нужно отображать оба маятника независимо друг от друга.

Протокол передачи

Протокол передачи изменён (5 октября 2010). Протокол состоит из пакетов. Каждый пакет является минимальной и неделимой единицей протокола передачи. Пакет – это посылка данных от микроконтроллера к компьютеру. Пакет состоит из восьми байт. Побайтная структура одного пакета данных такова:

1. FDh
2. FAh
3. ADRESH_UP – старший байт результата АЦП 1-го канала (верхнего маятника)
4. ADRESL_UP – младший байт результата АЦП 1-го канала (верхнего маятника)
5. FDh
6. F9h
7. ADRESH_DOWN – старший байт результата АЦП 2-го канала (нижнего маятника)
8. ADRESL_DOWN – младший байт результата АЦП 2-го канала (нижнего маятника)

Побайтная структура одного пакета данных изображена на рис. 1; временная диаграмма протокола изображена на рис. 2.

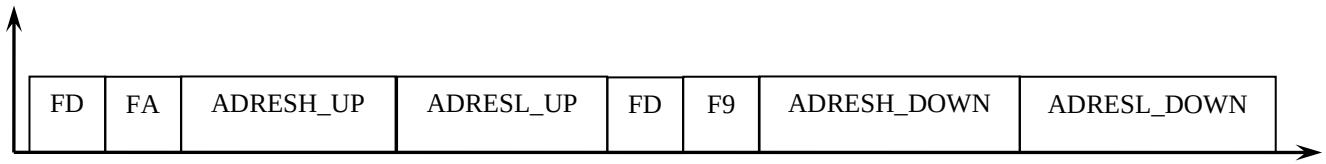
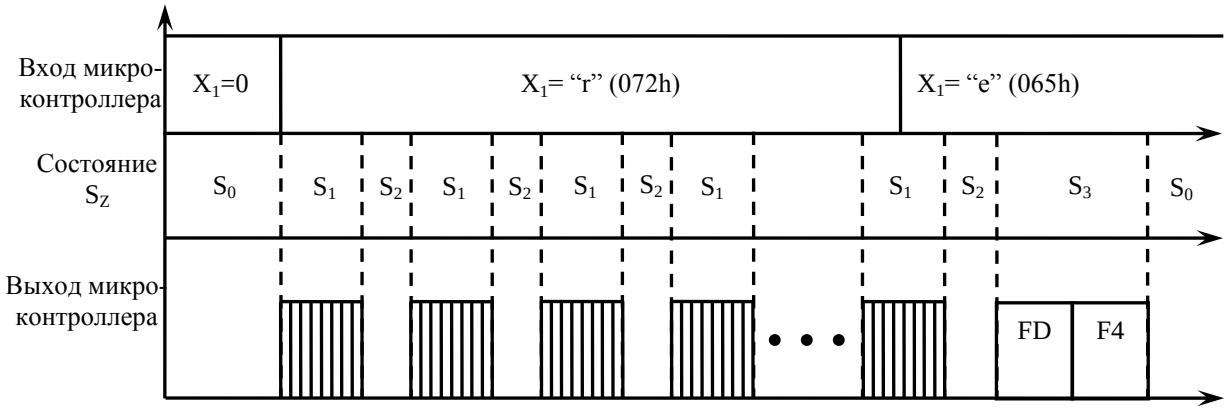


Рис. 1. Побайтная структура пакета данных.



S_0 - «Готовность» S_1 - «Передача данных» S_2 - «Опрос входа» S_3 - «Завершение сеанса»

X_1 - «Входная переменная»

Рис. 2. Временная диаграмма протокола.

Диаграмма состояний

В соответствии с поставленной задачей разумно будет предположить, что работающий микроконтроллер может находиться в одном из четырёх состояний. Вектор состояний S_Z – это четыре состояния S_0, S_1, S_2, S_3. Диаграмма состояний на рис. 3.

1. После включения микроконтроллер будет находиться в состоянии S_0 – «Готовность», при этом переменная состояния будет равна нулю ($X_1 = 0$). В состоянии готовности микроконтроллер проводит проверку входной переменной на равенство “r” (X_1=072h).
2. Как только на контроллер поступает символ “r” от компьютера, контроллер переходит в состояние S_1 – «Передача данных». В этом состоянии измеряются напряжения на аналоговых цепях, затем передаётся пакет, зажигается диод на RD0.
3. Когда пакет передан, микроконтроллер переходит в состояние S_2 – «Опрос входа». И, если входная переменная не равна “e” (065h) передаётся следующий пакет данных (состояние S_1).
4. Если от компьютера поступил символ “e” (065h) микроконтроллер переходит в состояние S_3, входная переменная равна “e” (065h). Контроллер передаёт 0D 04, для компьютера это значит, что сеанс закончен. Затем контроллер переходит в состояние S_0, устанавливает X1 = 0 чтобы микроконтроллер не переходил в состояние S_2 а находился в состоянии S_0 и ожидал ввода символа “r” (072h). Светодиод на RD0 гаснет. Если компьютер передаёт символ “r”, то программа продолжается с пункта 2.

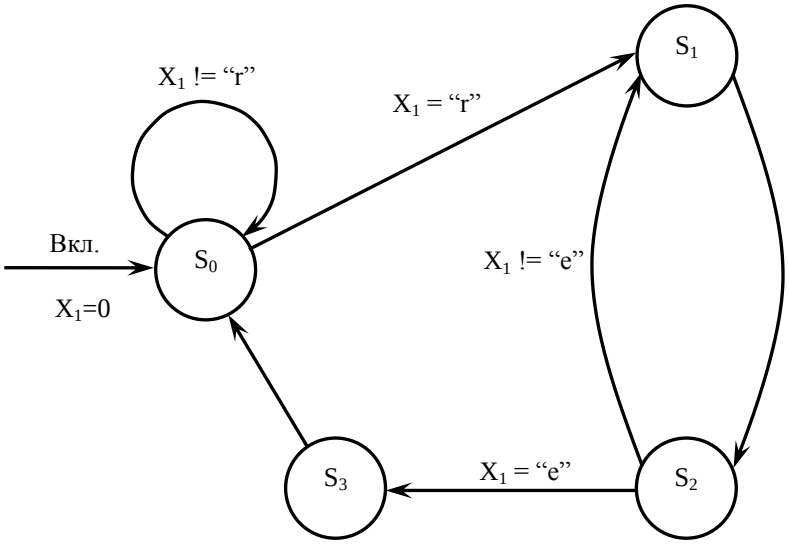


Рис. 3. Диаграмма состояний микроконтроллера.

Рассмотрев диаграмму состояний и протокол передачи можно сделать вывод, что программу целесообразно разделить на два процесса. В первом процессе нужно отслеживать поступление на вход символов “r” и “e” и менять входную переменную в соответствии с полученным символом. Во втором процессе нужно передавать данные от маятника с определённым периодом времени (расчёт этого времени будет приведён ниже). Совершенно понятно, что в первом процессе изменения происходят гораздо реже, чем во втором. Поэтому целесообразно обрабатывать первый процесс в обработчике прерывания. Во втором процессе нужно точно выдерживать равные интервалы времени между передачами данных, чтобы график на экране не искажался. Так как по условиям задачи есть всего одна входная переменная X_1, то разумно будет использовать её как переменную состояний для связи между процессами. Получается, что входная переменная является семафором на три состояния.

Распределение памяти данных и описание переменных:

ADRESH_UP – старший байт результата АЦП 1-го канала (верхнего маятника)
ADRESL_UP – младший байт результата АЦП 1-го канала (верхнего маятника)
ADRESH_DOWN – старший байт результата АЦП 2-го канала (нижнего маятника)
ADRESL_DOWN – младший байт результата АЦП 2-го канала (нижнего маятника)
X1 – переменная состояния, она же входная переменная изменяется в прерывании по USART.

Результат АЦП расположен в разных банках памяти данных, Для упрощения обмена данными между банками памяти переменные лучше расположить по последним адресам памяти данных микроконтроллера PIC16F877, так как эти адреса отображаются на все банки памяти данных. Это адреса с 070h по 07Fh в нулевом банке и с 0F0h по 0FFh в первом банке памяти данных. Регистры USART TXREG (019h) RCREG (01Ah) расположены в нулевом банке памяти данных и поэтому удобно будет брать данные и не переключать банки.

Таблица описания ячеек памяти данных

Название переменной	Адрес	Описание	Прим.
status_temp	0x7F	Регистр для сохранения контекста в прерывании	
w_temp	0x7E	Регистр для сохранения контекста в прерывании	
ADRESH_UP	0x7D	Старший байт АЦП верхнего маятника	
ADRESL_UP	0xFC	Младший байт АЦП верхнего маятника	1 банк
ADRESH_DOWN	0x7B	Старший байт АЦП нижнего маятника	
ADRESL_DOWN	0xFA	Младший байт АЦП нижнего маятника	1 банк
X1	0x79	Переменная состояния	

Таблица соответствия ячеек памяти данных 0банка и 1банка

Название переменной	Адрес 0 банк	Адрес 1 банк	Описание	Прим.
status_temp	0x7F		Регистр для сохранения контекста в прерывании	
w_temp	0x7E		Регистр для сохранения контекста в прерывании	
ADRESH_UP	0x7D		Старший байт АЦП верхнего маятника	
ADRESL_UP	0x7C	0xFC	Младший байт АЦП верхнего маятника	1 банк
ADRESH_DOWN	0x7B		Старший байт АЦП нижнего маятника	
ADRESL_DOWN	0x7A	0xFA	Младший байт АЦП нижнего маятника	1 банк
X1	0x79		Переменная состояния	

Таблица соответствия X1 и состояний контроллера

Содержимое X1	Название состояния	Наименование состояния	Описание состояния
«r»	S1	«Передача данных»	Работа маятника
«e»	S2	«Опрос входа»	Ожидание ввода символа «e»
«0»	S3	«Завершение сеанса»	Окончание работы
«0»	S0	«Готовность»	Ожидание ввода символа «r»

Настройка нулевого таймера.

Расчёт времени, через которое будут передаваться данные на компьютер. Известно, что есть два маятника. Допустим, что каждый маятник может колебаться с максимальной частотой в 5 Герц. Чтобы получить плавный график на экране нужно делать выборки с частотой раз в десять выше, чем частота колебаний.

Период T одного колебания будет равен 1/5 = 0,2 с. Для каждого канала возьмём по десять измерений 10 + 10 = 20 измерений за период T. Если за один период колебания маятника нужно сделать 20 измерений, получаем время между измерениями

t = 0,2/20 = 0,01 = 10 мс.

Это приблизительный расчёт стадии «P». В результате экспериментов может выясниться, что это время нужно изменить.

Для отсчёта времени будем использовать нулевой таймер.

Нулевой таймер будет работать от микроконтроллера (Fosc/4= 20/4=5MHz). Посчитаем количество машинных циклов (N) за 10 мс:

$$N = \frac{10 \cdot 10^{-3}}{0{,}2 \cdot 10^{-6}} = \frac{10^{-2}}{2 \cdot 10^{-7}} = \frac{10^{-2} \cdot 10^{7}}{2} = \frac{10^{5}}{2} = \frac{100000}{2} = 50000$$

теперь поделим полученное число на установку предделителя 256 и найдём число, которое даст 10 мс.

$$\frac{50000}{256} = 195{,}3125 \approx 0xC3$$

Это число нужно отнять от 0FFh для того, чтобы счётчик переполнялся каждые 10 мс. Рассчитаем число, загружаемое в TMR0:

0xFF–0xC3=0x3C.

Настроим таймер 0 в регистре OPTION_REG. Таймер получает тактовые импульсы от микроконтроллера. Предделитель включаем перед таймером (PSA=0). Коэффициент предделителя 1:256 (<PS2:PS0> 111). Переполнение таймера определяем опросом флага T0IF.

Таблица настройки нулевого таймера.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
OPTION_REG	1, 3	81h, 181h			T0CS		PSA	PS2	PS1	PS0	1111 1111	1111 1111
					0		0	1	1	1		
INTCON	0, 1, 2, 3	0Bh,8Bh, 10Bh,18Bh	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
					0			1/0				
TMR0	0, 2	01h, 101h	0x3C								xxxx xxxx	uuuu uuuu

u – не изменяется, x – не определено.

Настройка АЦП.

Рекомендованная последовательность работы модуля АЦП такова:

1. Настроить входы АЦП (ADCON1 <PCFG3:PCFG0>)
2. Выбрать источник тактовых импульсов
   2.1. Для частоты 20 МГц возможен только один вариант 32 Tosc (ADCS1:ADCS0 = 10)
3. Включить АЦП (BSF ADCON0, ADON)
4. Выбрать канал
5. Выдержать паузу, необходимую для зарядки конденсатора C hold.
   5.1. Пауза для зарядки C hold не менее 20 мкс
6. Запустить преобразование (BSF ADCON0,GO)
7. Ожидать сброса бита GO или установку бита ADIF.
8. Считать результат из ADRESH:ADRESL
9. Если нужно, продолжить с п. 4.

С работой АЦП связаны регистры ADCON0 (01Fh), ADCON1 (09Fh), PIR1 (0Ch), PIE1 (8Ch). Прерывания от модуля АЦП запрещены. В работе модуля будут участвовать два канала. Каналы будут переключаться для получения пакета данных (рис. 1). Данные будут передаваться в формате 2 байта с правым выравниванием. АЦП будет использовать вывод контроллера 2 RA0/AN0 для верхнего маятника или 1-го канала. И вывод 3 RA1/AN1 для нижнего маятника или 2-го канала. Эти ножки контроллера должны быть настроены на вход (TRISA,0 = 1 и TRISA,1 = 1).

Заполним настроечные таблицы для АЦП.

Выводы порта PORTA,0 PORTA,1 настраиваем на вход.

Таблица TRIS для АЦП.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
TRISA	1	85h	-	-					1	1	--11 1111	--11 1111

Из всех вариантов настройки каналов наиболее подходит под задачу

<PCFG3:PCFG0> = 0100 AN0, AN1, AN3 аналоговые, остальные цифровые Vref берётся от напряжения питания.

Таблица настройки АЦП.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
PIE1	1	8Ch	PSP IE	AD IE	RC IE	TX IE	SSP IE	CCP1 IE	TMR2 IE	TMR1 IE	0000 0000	0000 0000
				0								
PIR1	0	0Ch		ADIF							0000 0000	0000 0000
				0/1								
ADCON0	0	1Fh	ADCS1	ADCS0	CHS2	CHS1	CHS0	go/ -done	—	ADON	0000 00-0	0000 00-0
			1	0	0	0	0 RA0 1 RA1	1/0		1		
ADCON1	1	9Fh	ADFM				PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	0--- 0000
			1				0	1	0	0		

Настройка USART

Рекомендованная последовательность включения модуля USART в асинхронном режиме:

1. Задать скорость обмена (SPBRG)
2. Настроить режим работы модуля на передачу (TXSTA)
3. Разрешить прерывания (PIE1)
4. Настроить режим работы модуля на приём (RCSTA)

Биты TRISC<6:7> должны быть настроены на вход.

По протоколу микроконтроллер должен передавать и принимать данные компьютера по USART в асинхронном режиме со скоростью 9600 бит за секунду. Прерывания разрешены от приёмника USART.

Чтобы задать скорость обмена нужно поместить в регистр SPBRG определённое число. Рассчитать его нужно по формуле:

Baud Rate = Fosc/(64·(X+1)), если BRGH = 0 (Low Speed)

Baud Rate = Fosc/(16·(X+1)), если BRGH = 1 (High Speed),

где X – число в SPBRG (должно находиться в диапазоне 0 – .255), Fosc – частота тактового генератора. В таблице документа на микроконтроллер приведено значение .129 (081h при BRGH = 1, High Speed), дающее погрешность 0,16% на основной частоте 20 МГц.

Режим работы модуля USART таков: скорость 9600 битов за секунду, 8 – битная передача, асинхронный режим, SPBRG настроен в режиме High Speed.

Для настройки модуля на передачу в регистре TXSTA нужно установить BRGH и TXEN (включить модуль). Передаваемые данные нужно поместить в регистр TXREG. Окончание передачи вызовет установку бита TRMT в регистре TXSTA. После этого можно загружать следующий байт в TXREG.

Для настройки модуля на приём в регистре RCSTA нужно установить SPEN, CREN. Принятые данные будут находиться в регистре RCREG. После принятия данных нужно прочитать бит RCREG и тогда RCIF сбрасывается аппаратно. Этот бит только для чтения.

Таблица TRISC для USART.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
TRISC	1	87h	1	1					1	1	1111 1111	1111 1111

Таблица состояния битов для настройки USART.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
SPBRG	1	099h	081h								0000 0000	0000 0000
TXSTA	1	098h	CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D	0000 -010	0000 -010
			0	0	1	0		1	1/0			
RCSTA	0	018h	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
			1	0	0	1	0	0	0	0		
TXREG	0	019h	Передаваемые на компьютер данные								0000 0000	0000 0000
RCREG	0	01Ah	Принятые от компьютера данные								0000 0000	0000 0000
PIR1	0	0Ch			RCIF						0000 0000	0000 0000
					0/1							

u – не изменяется, x – не определено.

Настройка устройства вывода микроконтроллера.

Во время работы микроконтроллер сигнализирует о своём состоянии. Если микроконтроллер ожидает приёма данных, то светодиод, подключенный к RD0 не горит. Если происходит передача данных, то светодиод горит. Бит 0 регистра TRISD сброшен, это настраивает на выход PORTD,0.

Таблицы настроек микроконтроллера.

В программе разрешены прерывания только от приёмника USART.

Таблица маски прерываний.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
INTCON	0,1,2,3	0Bh, 8Bh, 10Bh, 18Bh	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000x
			1	1	0	0	0	0/1	0	0		
PIR1	0	0Ch		ADIF	RCIF						0000 0000	0000 0000
				0/1	0/1							
PIE1	1	8Ch	PSP IE	AD IE	RC IE	TX IE	SSP IE	CCP1 IE	TMR2 IE	TMR1 IE	0000 0000	0000 0000
					1	0						

Таблица настройки регистров TRIS.

Название регистра	Номер банка	Адрес	Название, номер бита, значение бита.								Состояние после POR	Состояние после сброса
			7	6	5	4	3	2	1	0		
TRISA	1	85h	-	-					1	1	--11 1111	--11 1111
TRISC	1	87h	1	1							1111 1111	1111 1111
TRISD	1	88h								0	1111 1111	1111 1111

Алгоритм программы и обработчика прерываний

Алгоритм основной программы представлен на рис. 4 (исправлен 2 октября), рис. 5 (исправлен 5 октября), рис. 6. Алгоритм прерывания на рис. 7.

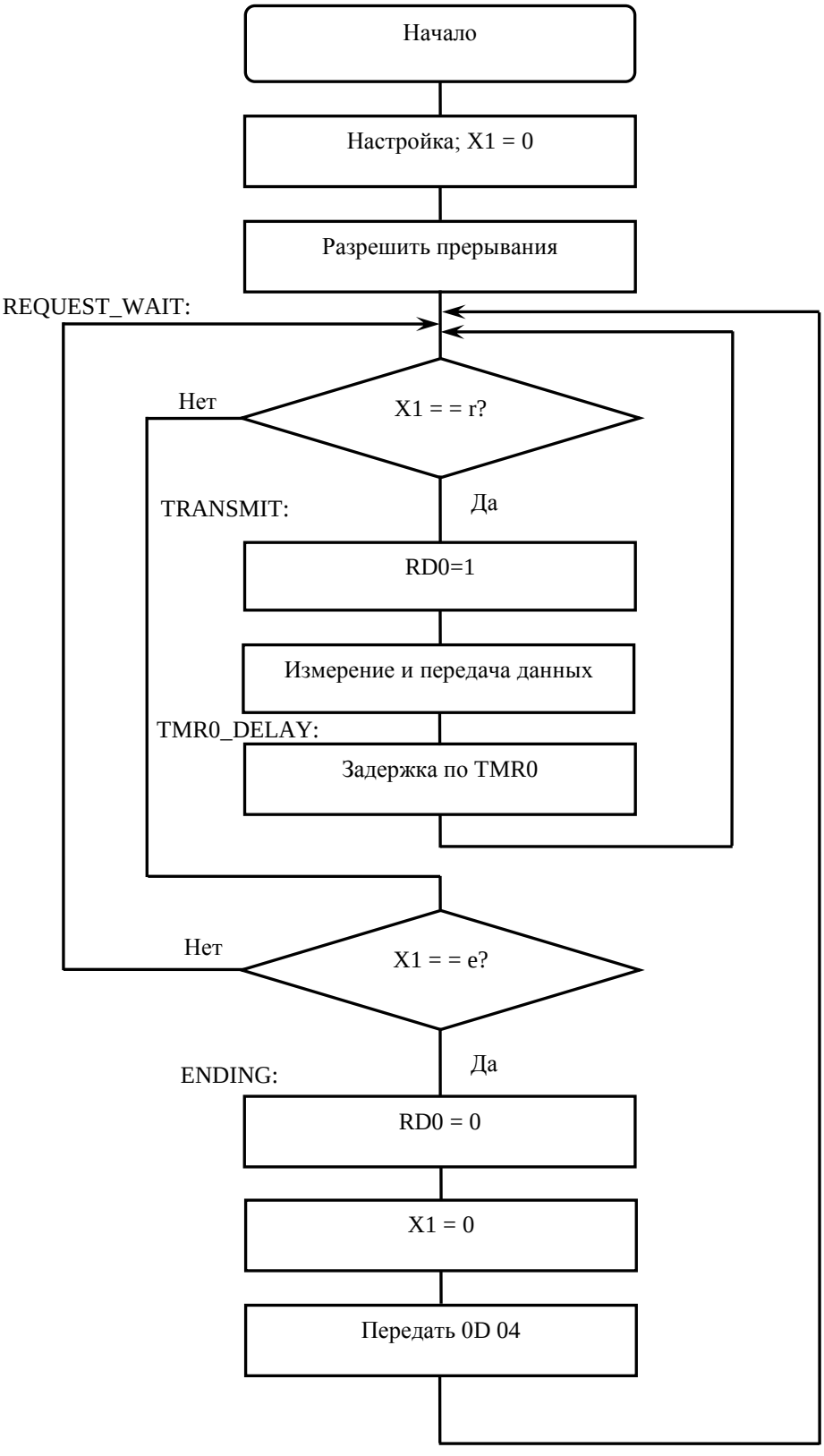


Рис. 4. Алгоритм основной программы.

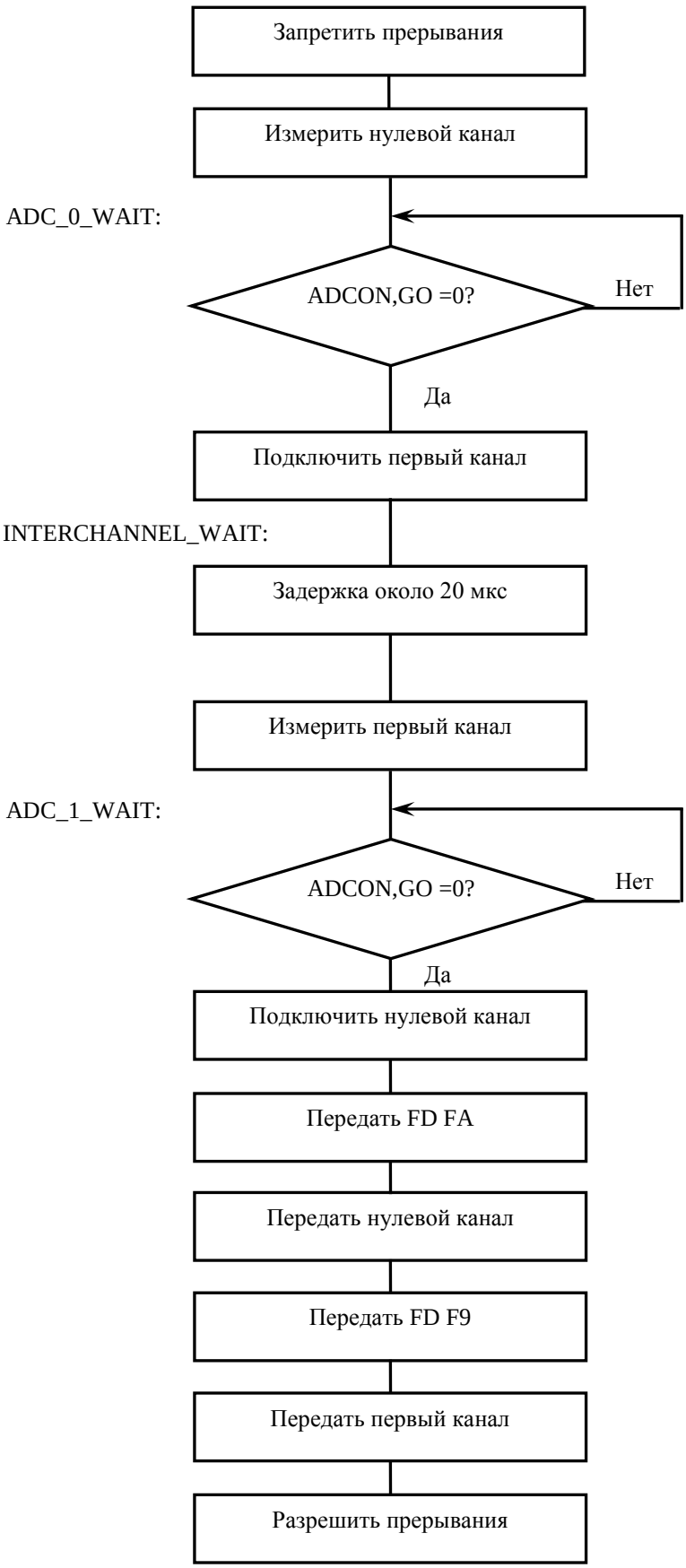


Рис. 5. Алгоритм измерения и передачи данных.

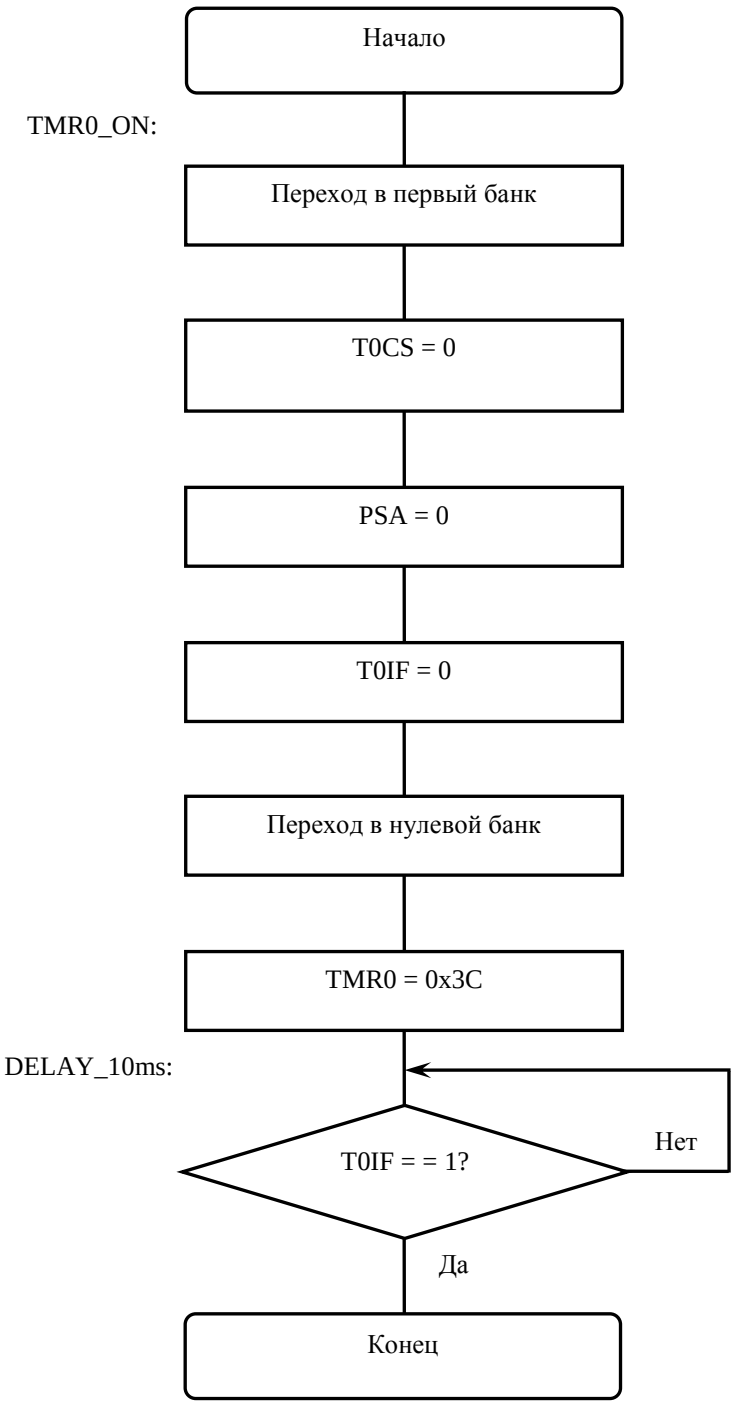


Рис. 6. Алгоритм задержки по TMR0 (по нулевому таймеру).

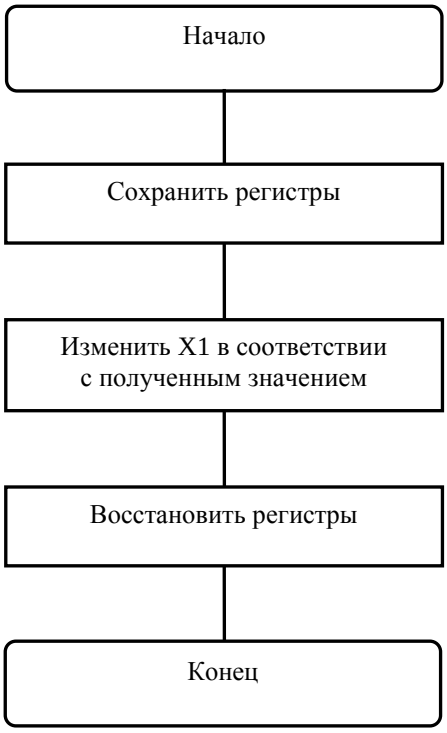


Рис. 7. Алгоритм прерывания.

Текст программы

```
;*****************************************************************
;   Filename:                    Pendulum_2_3.as m                *
;   Date:                        12.10.2010                       *
;   File Version:                0.0.5                            *
;   Author:                      Boguraev_Mikhail e-mail boguraev@bk.ru *
;   Company:                     GOU SPB SUT                      *
; Запретил прерывания на время АЦП и передачи                    *
; И увеличил время INTERCHANNEL_WAIT с 0x64 на 0xFF              *
;*****************************************************************
;   Files required:                                              *
;*****************************************************************
;   Notes: PIC16F877 @ 20 MHz                                    *
;*****************************************************************
list      p=16f877          ; list directive to define processor
#include <p16f877.inc>        ; processor specific variable definitions
__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC &
_WRT_ENABLE_ON & _LVP_OFF & _DEBUG_ON & _CPD_OFF
;***** VARIABLE DEFINITIONS
w_temp              EQU     0x7E;   variable used for context saving
status_temp         EQU     0x7F;   variable used for context saving
ADRESH_UP           EQU     0x7D;   Старший байт АЦП верхнего маятника
ADRESL_UP           EQU     0xFC;   Младший байт АЦП верхнего маятника 1 банк
ADRESH_DOWN         EQU     0x7B;   Старший байт АЦП нижнего маятника
ADRESL_DOWN         EQU     0xFA;   Младший байт АЦП нижнего маятника 1 банк
X1                  EQU     0x78;   Переменная состояния
INTERCHANNEL_COUNTER    EQU     0x77;   Ячейка для задержки ADC для Asc Time
;*****************************************************************
      ORG     0x000           ; processor reset vector
      nop
      clrf    PCLATH          ; ensure page bits are cleared
      goto    main            ; go to beginning of program
      ORG     0x004           ; interrupt vector location
      movwf   w_temp          ; save off current W register contents
      movf    STATUS,w        ; move status register into W register
      movwf   status_temp     ; save off contents of STATUS register
      bcf     STATUS,RP0
      bcf     STATUS,RP1
      movf                RCREG,W
      movwf   X1
end_of_isr:
      movf    status_temp,w   ; retrieve copy of STATUS register
      movwf               STATUS          ; restore pre-isr STATUS register contents
      swapf   w_temp,f
      swapf   w_temp,w        ; restore pre-isr W register contents
      retfie                ; return from interrupt
main
      clrf    PORTD
      clrf    X1
      bsf     STATUS,RP0
      movlw   b'00000000'
      movwf   TRISD
```

```
                        movlw   081h
        movwf   SPBRG
        bsf     TXSTA,BRGH
        bsf     TXSTA,TXEN
        bsf     ADCON1,ADFM
        bsf     ADCON1,PCFG2
        bsf     PIE1,RCIE
        bcf     STATUS,RP0
        bsf     RCSTA,CREN
        bsf     RCSTA,SPEN
        bsf     ADCON0,ADCS1
        bsf     ADCON0,ADON
        bsf     INTCON,PEIE
        bsf     INTCON,GIE
REQUEST_WAIT:
        movlw   "r"
        subwf   X1,W
        btfsc   STATUS,Z
        goto    TRANSMIT
; проверка X1 на "e"
        movlw   "e"
        subwf   X1,W
        btfsc   STATUS,Z
        goto    ENDING
        goto    REQUEST_WAIT
TRANSMIT:
; Зажечь светодиод
        movlw   b'00000001'
        movwf   PORTD
        bcf     INTCON,GIE
        call    MEASURING_TRANSMISSION
        bsf     INTCON,GIE
        call    TMR0_DELAY
        goto    REQUEST_WAIT
ENDING:
        bcf     PORTD,0
        clrf    X1
        movlw           0x0FD
        movwf           TXREG
                        bsf             STATUS,RP0
WAIT_0xFD_ENDING:
                        btfss   TXSTA,TRMT
                        goto    WAIT_0xFD_ENDING
                        bcf             STATUS,RP0
                        movlw           0x0F4
                        movwf           TXREG
                        bsf             STATUS,RP0
WAIT_0xF4_ENDING:
                        btfss   TXSTA,TRMT
                        goto    WAIT_0xF4_ENDING
                        bcf             STATUS,RP0
        goto    REQUEST_WAIT
```

```
MEASURING_TRANSMISSION
; пошло измерение и передача данных MEASURING_TRANSMISSION
; ADCON0 по умолчанию настраивает нулевой канал (UP) RA0/AN0 ножка 2
; При первом входе установки по умолчанию, при последующих входах
; нулевой канал устанавливается после первого
    bsf    ADCON0,GO
ADC_0_WAIT:
                    btfsc  ADCON0,GO
                    goto   ADC_0_WAIT
; если преобразование закончено, тогда младший байт АЦП 0 канала поместить в память
    bsf    STATUS,RP0
    movf   ADRESL,W
    movwf  ADRESL_UP
    bcf    STATUS,RP0
; теперь старший байт АЦП 0 канала поместить в память
    movf   ADRESH,W
    movwf  ADRESH_UP
; переключаем на первый канал RA1/AN1 ножка 3
    bsf    ADCON0,CHS0
; после переключения каналов АЦП надо подождать время Tasq - около 20 мкс.
    movlw  0xFF; 0x64
    movwf  INTERCHANNEL_COUNTER
INTERCHANNEL_WAIT:
    decfsz  INTERCHANNEL_COUNTER,F
    goto    INTERCHANNEL_WAIT
    bsf    ADCON0,GO
ADC_1_WAIT:
                    btfsc  ADCON0,GO
                    goto   ADC_1_WAIT
; если преобразование закончено, тогда младший байт АЦП 1 канала поместить в память
    bsf    STATUS,RP0
    movf   ADRESL,W
    movwf  ADRESL_DOWN
    bcf    STATUS,RP0
; теперь старший байт АЦП 1 канала поместить в память
    movf   ADRESH,W
    movwf  ADRESH_DOWN
; переключаем на нулевой канал RA0/AN0 ножка 2
; пока идёт передача данных и задержка в 500 мкс - Tasq пройдёт.
    bcf    ADCON0,CHS0
                    movlw       0x0FD
                    movwf       TXREG
                    bsf         STATUS,RP0
WAIT_0xFD_FIRST:
                    btfss  TXSTA,TRMT
                    goto   WAIT_0xFD_FIRST
                    bcf         STATUS,RP0
                    movlw       0x0FA
                    movwf       TXREG
                    bsf         STATUS,RP0
WAIT_0xFA_FIRST:
                    btfss  TXSTA,TRMT
```

```
                    goto  WAIT_0xFA_FIRST
                    bcf         STATUS,RP0
    movf    ADRESH_UP,W
    movwf   TXREG
    bsf     STATUS,RP0
WAIT_RECEIVED_ADRESH_UP:
                    btfss TXSTA,TRMT
                    goto  WAIT_RECEIVED_ADRESH_UP
    bcf     STATUS,RP0
    movf    0x7C,W; ADRESL_UP Младший байт АЦП верхнего маятника 1 банк
                    movwf       TXREG
    bsf     STATUS,RP0
WAIT_RECEIVED_ADRESL_UP:
                    btfss TXSTA,TRMT
                    goto  WAIT_RECEIVED_ADRESL_UP
                    bcf         STATUS,RP0
                    movlw       0x0FD
                    movwf       TXREG
                    bsf         STATUS,RP0
WAIT_0xFD_SECOND:
                    btfss TXSTA,TRMT
                    goto  WAIT_0xFD_SECOND
                    bcf         STATUS,RP0
                    movlw       0x0F9
                    movwf       TXREG
                    bsf         STATUS,RP0
WAIT_0xF9_SECOND:
                    btfss TXSTA,TRMT
                    goto  WAIT_0xF9_SECOND
                    bcf         STATUS,RP0
; RECEIVE_DATA_1_CHANNEL
    movf    ADRESH_DOWN,W
    movwf   TXREG
    bsf     STATUS,RP0
WAIT_RECEIVED_ADRESH_DOWN:
                    btfss TXSTA,TRMT
                    goto  WAIT_RECEIVED_ADRESH_DOWN
    bcf     STATUS,RP0
    movf    0x7A,W; ADRESL_DOWN Младший байт АЦП верхнего маятника 1 банк
                    movwf       TXREG
    bsf     STATUS,RP0
WAIT_RECEIVED_ADRESL_DOWN:
                    btfss TXSTA,TRMT
                    goto  WAIT_RECEIVED_ADRESL_DOWN
                    bcf         STATUS,RP0
    return
TMR0_DELAY
    bsf     STATUS,RP0
    bcf     OPTION_REG,T0CS
    bcf     OPTION_REG,PSA
    bcf     STATUS,RP0
    bsf     INTCON,T0IF
```

```
        movlw   0x3C
                        movwf   TMR0
DELAY_10ms
        btfss   INTCON,T0IF
        goto    DELAY_10ms
        return
        END                     ; directive 'end of program'
```

Оглавление:

Список рисунков: