Московский государственный технический университет
имени Н.Э. Баумана

Методические указания

А.Ю. Попов

Проектирование радиоэлектронной аппаратуры на основе микроконтроллеров ARM7TDMI

Работа №3. Система прерываний микроконтроллера микроконтроллера и управление интерфейсом RS232.

Цель работы – изучение системы прерываний микроконтроллера NXP LPC2478 и принципов функционирования модуля универсального асинхронного приемо-передатчика UART.

В ходе работы студенту необходимо ознакомиться с теоретическим материалом, касающимся системы прерываний и модуля RS232, разработать и отладить программу функционирования микроконтроллера NXP LPC2478 с использованием отладочной платы SK-LPC2478-S3E.

Система прерываний микроконтроллера

Система прерываний является важной составляющей микроконтроллеров ARM7, так как позволяет построить эффективную обработку информации, поступающей от многочисленных периферийных устройств. Их количество и высокий темп передачи данных потребовал создания многоуровневой приоритетной системы обработки запросов прерываний.

Из семи режимов работы ARM7, два режима предназначены для обработки прерываний. Это режим быстрого прерывания Fast interrupt и режим векторных прерываний Interrupt. Время переключения на обработку быстрого прерывания (Fast interrupt) меньше, чем на обработку векторных прерываний, что может быть использовано разработчиком для приоритетного обслуживания наиболее важного события в системе.

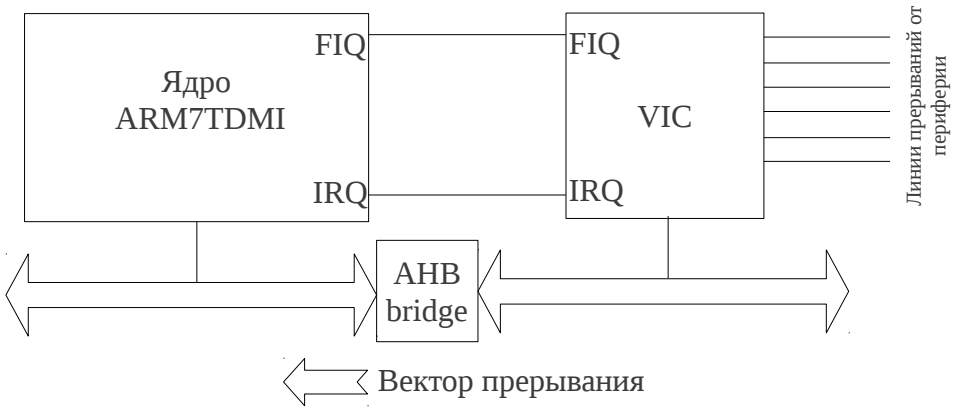

Рисунок 8.

Векторный контроллер прерываний.

Для организации приоритетной обработки запросов прерываний в микроконтроллерах LPC2478 предусмотрен контроллер прерываний Vector

Interrupt controller, VIC (Рисунок 8). Блок VIC обеспечивает приоритетную обработку 32 запросов прерываний, хранит и передает в ARM7 ядро вектор прерывания, позволяет программно изменять приоритет каждой линии (всего доступно 16 приоритетов) и тип прерываний (быстрое или векторное), а также служит для организации программных прерываний (Software Interrupt). Следует заметить, что сразу несколько входных линий прерываний могут иметь тип Fast interrupt. В этом случае в микропроцессорное ядро ARM7 передается функция ИЛИ всех таких запросов, а время реакции увеличивается за счет необходимости определения источника прерывания. В случае одновременного поступления сигналов прерываний, имеющих одинаковый приоритет, в первую очередь будет обработан источник с меньшим номером линии (смотри Приложение 1).

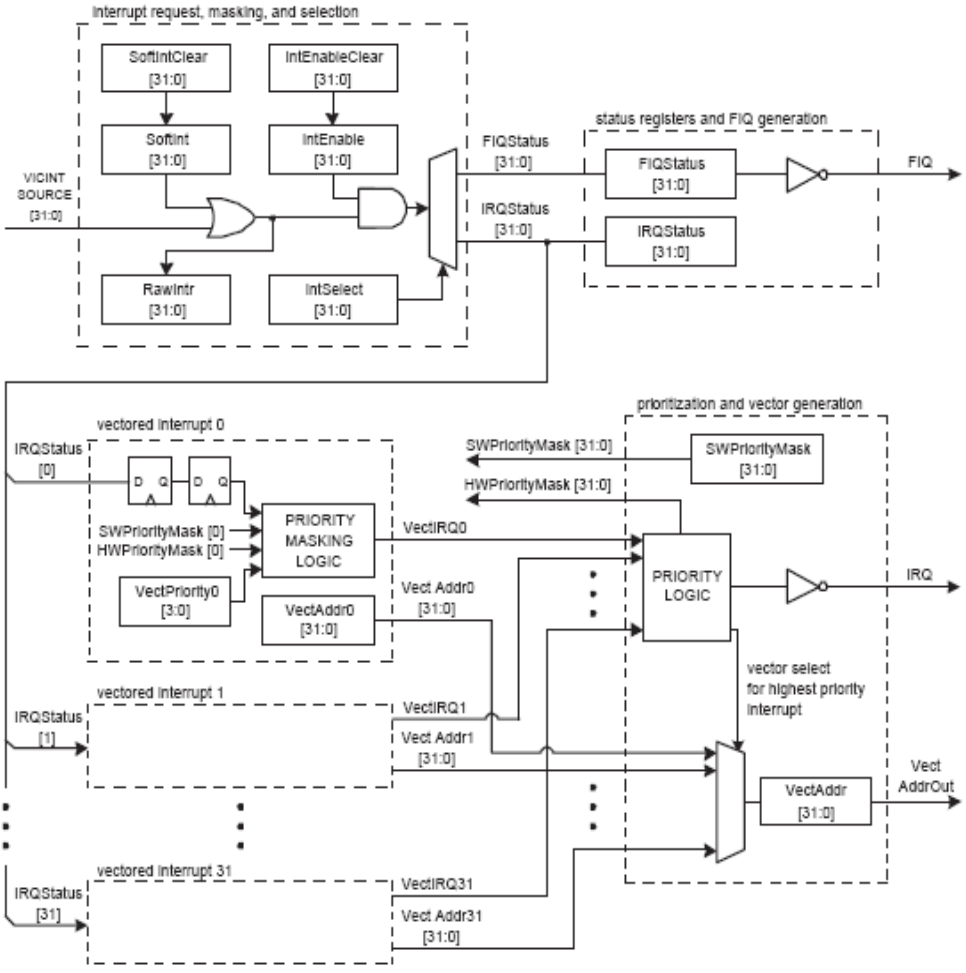

Рисунок 1 — Структура VIC

Программные прерывания (Software Interrupts) позволяют разработчику вызвать процедуры обработки FIQ и IRQ по программному событию, а не при возникновении внешних событий. Эта возможность может использоваться для

реализации операционных систем и сложных проектов.

Для описания процедур обработки прерывания на языке С, требуется использовать директиву __irq для указания обработчика векторного прерывания или быстрого прерывания.

```
void FIQ_int (void) __irq     //Векторное прерывание
{ ...
  }
```

При поступлении запроса быстрого прерывания, ядро ARM переключается в режим FIQ и выполняет команду LDR PC, FIQ_Addr записанную в таблице векторов прерываний по соответствующему смещению. Значение FIQ_Addr является точкой входа в процедуру обработки FIQ и может быть изменено пользователем в Startup файле. Например, для указания процедуры FIQ_int в качестве обработчика быстрого прерывания следует изменить таблицу векторов следующим оразом:

```
//Таблица векторов прерываний
Vectors     LDR   PC, Reset_Addr
            LDR   PC, Undef_Addr
            LDR   PC, SWI_Addr
            LDR   PC, PAbt_Addr
            LDR   PC, DAbt_Addr
            NOP                 ; Reserved Vector
;           LDR   PC, IRQ_Addr
            LDR   PC, [PC, #-0x0120]    ; Vector from VicVectAddr
            LDR   PC, FIQ_Addr
//Точки входа в обработчики прерываний
Reset_Addr      DCD   Reset_Handler
Undef_Addr      DCD   Undef_Handler
SWI_Addr        DCD   SWI_Handler
PAbt_Addr       DCD   PAbt_Handler
DAbt_Addr       DCD   DAbt_Handler
                DCD   0             ; Reserved Address
IRQ_Addr        DCD   IRQ_Handler
FIQ_Addr        DCD   FIQ_irq
```

При поступлении запроса прерываний IRQ исполняется команда LDR PC, [PC, #-0x0120], по которой в счетчик команд загружается вектор прерывания из контроллера VIC. Происходит это в связи с тем, что данная команда находится по адресу 0x00000018 и обращается в память по смещению 0x00000120 относительно текущего значения PC. Следует заметить, что значение PC для любой команды всегда превосходит ее адрес на 8, что связано с конвейерной организацией микропроцессора. Таким образом, точка входа в обработчик считывается по адресу 0x00000018 + 8 — 0x120 = 0xffffff00, т.е. регистра VICAddress контроллера VIC (см. таблицу 1).

При использовании прерываний FIQ следует внести изменения в таблицу векторов прерываний:

```
IMPORT FIQ_irq ; Импорт адреса обработчика
FIQ_Addr      DCD    FIQ_irq
; FIQ_Handler B FIQ_Handler
```

Следует иметь ввиду, что запросы прерываний не сбрасываются автоматически при переходе в режим IRQ или FIQ в вызвавших их периферийных модулях. Разработчик должен самостоятельно сбросить биты запросов прерываний в соответствующем модуле с помощью регистров сброса запросов.

Для правильной работы системы прерываний необходимо выполнить инициализацию регистров VIC. В таблице 1 дано краткое их описание.

Таблица 1 - Регистры VIC

Название	Описание	Доступ
VICIRQStatus	**Регистр статуса IRQ,** позволяющий прочитать биты разрешения прерывания для каждой линии запроса IRQ	Чтение
VICFIQStatus	**Регистр статуса FIQ,** позволяющий прочитать биты разрешения прерывания для каждой линии запроса FIQ	Чтение
VICRawIntr	**Регистр состояния прерываний,** позволяющий определить активность каждой из 32-х линии запроса или программного прерывания	Чиение
VICIntSelect	**Регистр выбора прерывания,** позволяющий для каждой из 32 линий выбрать ее тип: FIQ или IRQ.	Чтение/ Запись
VICIntEnable	**Регистр разрешения прерываний.** При чтении регистра, единичный разряд указывает на разрешение линии или программного прерывания. При записи, единичный разряд разрешает обработку прерывания по линии FIQ или IRQ или программного прерывания.	Чтение/ Запись
VICIntEnClr	**Регистр сброса разрешения прерывания.** При записи единичный разряд запрещает обработку соответствующего прерывания FIQ, IRQ или программного прерывания.	Запись
VICSoftInt	**Регистр программных прерываний** служит для генерации запросов программных прерываний.	Чтение/ Запись
VICSoftIntClear	**Регистр сброса запроса прерывания.** При записи единичный разряд сбрасывает запрос программного прерывания.	Запись
VICProtection	**Регистр защиты VIC** позволяет ограничить доступ к VIC программам, запущенным в режиме User	Чтение/ Запись

VICSWPriorityMask	**Регистр маски приоритетов программных прерываний** позволяет замаскировать использование нескольких уровней приоритетов для программных прерываний.	Чтение/ Запись
VICVectAddr0 - 31	**Регистры векторов прерываний** линий 0 — 31 позволяют сохранить адрес обработчика прерывания для 32-х векторных прерывания.	Чтение/ Запись
VICVectPriority0 - 31	**Регистры приоритетов прерываний** линий 0 - 31 позволяют указать приоритет для каждого векторного прерывания	Чтение/ Запись
VICAddress	**Регистр вектора прерывания** хранит адрес текущего обрабатываемого прерывания	Чтение/ Запись

Рассмотрим пример использования прерывания таймера для управления вводом и выводом в порт.

```
/*    Пример 1.
      Управление портами ввода/вывода.
*/

#include <LPC24xx.H>                        /* Описание LPC22xx */

unsigned int n;

void Timer0_Init(void){
//Предделитель таймера = 15000
      T0PR = 15000;
//Сбросить счетчик и делитель
      T0TCR = 0x00000002;
//При совпадении сбрасываем таймер и вызываем прерывание
      T0MCR = 0x00000003;
//Регистр совпадения = 1000 (1 Гц)
      T0MR0 = 1000;
}

void Timer0_Int (void) __irq
{
    if (n!=0x0000080) {n<<=1;}
      else {n = 0x00000001;}
      T0IR = 0x00000001; /*Сбросить флаг прерывания в Timer0*/
//Бегущая единица
    IOCLR0 = 0x000000FF;
    IOSET0 = n;   /* Установить состояние порта */
    VICVectAddr = 0; /*Перевести VIC в исходное состояние*/
}

int main (void) {

  IODIR0 = 0x000000FF; /* P0.0..7 программируем на вывод, остальные на ввод */
  IOCLR0 = 0x000000FF; /* Устанавливаем ноль на выходах */
```

```
  n = 0x00000001;
  IOSET0 = n;   /* Установить состояние порта */

  Timer0_Init(); /* Настроить таймер */

//Записать адрес обработчика прерывания в таблицу векторов
  VICVectAddr4 = (unsigned)Timer0_Int;
//Разрешить прерывания
  VICIntEnable |= 0x00000010;
//Запустить таймер
  T0TCR = 0x00000001;

  for (;;) {}          /* Бесконечный цикл */
}
```

Принцип действия универсального асинхронного приемопередатчика (UART) микроконтроллера LPC2478

Универсальный асинхронный приемопередатчик UART (Universal Asynchronous Receiver-Transmitter) предназначен для асинхронной дуплексной передачи пакетов данных по последовательному интерфейсу типа RS232C или «токовая петля».

Для интерфейса RS232C характерно подключение устройств с помощью двух сигнальных линий TX и RX таким образом, что передатчик устройства 1 подключается выходом TX к входу RX устройства 2, а приемник устройства 1 подключается входом RX к выходу TX устройства 2.

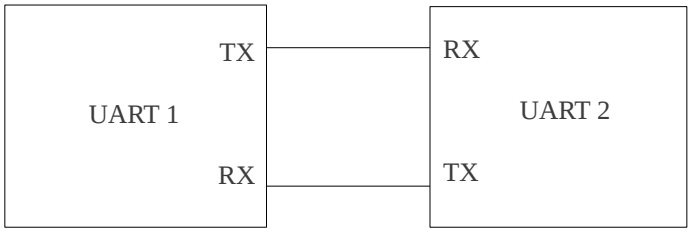

Рисунок 2 — Схема подключение двух универсальных асинхронных приемо-передатчиков

При отсутствии передачи на линии TX поддерживается высокий уровень сигнала. Передача начинается установкой сигнала TX в низкий уровень («стартовый бит»). После этого с заданной частотой на линию передаются от 5 до 8 бит данных. Последовательность бит передается начиная с младших и может дополняться контрольным разрядом. Передача завершается установкой сигнала в единичный уровень («стоповый бит»). Таким образом, один пакет может содержать от 7 до 11 бит. Для правильного приема информации передающая и принимающая стороны должны использовать одинаковые настроечные параметры, такие как: частота синхронизации приемопередатчика, количество бит данных, наличие стопового бита.

В микроконтроллерах NXP LPC2478 четыре приемо-передатчика UART (UART0/1/2/3), три из которых (UART0/2/3) имеют одинаковую структуру и набор регистров. Дальнейшее описание будет относиться в равной мере к данным трем устройствам.

Отличительными особенностями UART0/2/3 является:

- Наличие 16 байтных FIFO буферов передатчика и приемника.
- Наличие схем контроля уровней заполнения FIFO буферов (1,2,4,8,14 слов) с возможностью генерации прерываний.
- Встроенный генератор задающей частоты с возможностью умножения и деления частоты, автоподстойки частоты, программного управления приемом и передачей.

В универсальный асинхронный приемопередатчик микроконтроллера LPC2478 входят следующие регистры (Таблица 2):

Название	Описание	Доступ
RBR (при DLAB=0)	Регистр буфера приемника позволяет прочитать данные из FIFO буфера приемника. Чтение данных из регистра RBR возможно при сброшенном бите DLAB (бит фиксации делителя частоты) в регистре статуса LCR	чтение
THR (при DLAB=0)	Регистр данных передатчика позволяет записать данные в FIFO буфер передатчика. Запись возможна только при сброшенном бите DLAB (бит фиксации делителя частоты) в регистре статуса LCR	запись
DLL (при DLAB=1)	Младший байт делителя частоты позволяет задать частоту приема и передачи информации через UART.	чтение/ запись
DLM (при DLAB=1)	Старший байт делителя частоты позволяет задать частоту приема и передачи информации через UART.	чтение/ запись
IER (при DLAB=0)	Регистр разрешения прерываний позволяет управлять тремя прерываниями: прерывание по приему, прерывания по передаче, прерывание по событию на линии RX.	чтение/ запись
IIR	Регистр прерываний позволяет идентифицировать прерывание	чтение
FCR	Регистр управления UART позволяет: разрешить использование FIFO буфера, сбросить содержимое FIFO RX и TX буфера, указать уровень заполнения FIFO, вызывающий прерывание	запись
LCR	Регистр управления передачей позволяет задать: длина слова (от 5 до 8 бит), стоповый бит (от 1 до 2 бит), контроль по четности, возможность разрыва передачи, бит DLAB фиксации делителя частоты	чтение/ запись
LSR	Регистр статуса линий позволяет получить доступ к состоянию линий RX и TX.	чтение
SCR	Дополнительный пользовательский регистр	чтение/

		запись
ACR	Регистр управления автоподстойкой частоты служить для запуска и управления процессом автоматического определения частоты при получении данных.	чтение/ запись
ICR	Регистр управления передачей через инфракрасный порт служит для организации интерфейсов IrDA	чтение/ запись
FDR	Регистр подстройки делителя частоты содержит два поля: DIVADDVAL (0 ≤ DIVADDVAL < 15) деления частоты и поле MULVAL умножения частоты (0 < MULVAL ≤ 15), DIVADDVAL<MULVAL	чтение/ запись
TER	Регистр разрешения передачи	чтение/ запись

Частота передачи UART задается значениями, хранимыми в регистрах DLL, DLM, FDR по формуле:

$$UART_{частота} = \frac{PCLK}{16 \times (256 \times DLM + DLL) \times (1 + \frac{FDR_{DIVADDVAL}}{FDR_{MULVAL}})} ,$$

где PCLK - частота синхронизации UART, DLM и DLL байты делителя частоты, FDRxxx – поля регистра подстройки делителя частоты. Запись параметров делителя возможно только при единичном значении разряда DLAB регистра LCR.

В приложении 2 приведен алгоритм определения параметров делителя частоты и примеры их вычислений. Пример использования универсального асинхронного приемо-передатчика приведен ниже.

Пример 2

```
#include <LPC24xx.H>
#include <string.h>


char str[255],cur_log[9],cur_pas[9],rx;
int i;
const char log[9] = "root";
const char pas[9] = "12345678";

//Обработчик прерываний UART0 RDA и CTI
void UART0_Int(void) __irq
{
            unsigned int j;
        void *pos;

             //Читать из FIFO буфера байты данных
             while (U0LSR & 0x01) {
                //Прочитать байт и сбросить прерывание
                  rx=U0RBR;
```

```
                //Сохранить в str
                memmove(&str[strlen(str)],&rx,1);
                if (rx==0xD) {i++;}
            }
            if (i>=2) {
                  //Получены две строки
                  i=0;
            memset(cur_log,0,9);
            memset(cur_pas,0,9);
            pos = memchr (str, 0xD, sizeof (str));
            if (pos != NULL) {
                  //Получить поле login
                j=(int)pos-(int)str;
                  memmove(&cur_log,&str,(int)pos-(int)str);
                      memmove(&str,&str[j+1],strlen(str));
                pos = memchr (str, 0xD, sizeof (str));
                if (pos != NULL) {
                      //Получить поле password
                   j=(int)pos-(int)str;
                        memmove(&cur_pas,&str,j);
                        memmove(&str,&str[j+1],strlen(str));
                       //Ожидание готовности передатчика
                        while (!(U0LSR & 0x20));
                        if
((memcmp(cur_log,log,9)==0)&&(memcmp(cur_pas,pas,9)==0)) {
                        //Идентификация закончилась успешно!
                              U0RBR=0x31;}
                        else {
                        //Идентификация закончилась неудачей!
                              U0RBR=0x30;}
                }
            }
            }
    VICVectAddr = 0; /*Перевести VIC в исходное состояние*/

}


void UART0_Init (void)
{
//Разрешить альтернативные UART0 функции входов/выходов P0.2 и P0.3: RxD и TxD

  PINSEL0   = 0x00000050;
//Установить параметры передачи: 8 бит, без контроля четности, 1 стоповый бит
//+Разрешить запись делителя частоты CLK_UART0
  U0LCR = 0x00000083;
//Установить делитель частоты на скорость 115200 при частоте CLK_UART0 = 15MHz
  U0DLL = 0x00000008;
//Фиксировать делитель частоты
  U0LCR = 0x00000003;
//Программировать FIFO буфер на прием 8-ми байт.
  U0FCR = 0x00000081;
//Разрешить прерывание по приему
  U0IER = 0x00000001;
//Записать адрес обработчика прерывания в таблицу векторов
  VICVectAddr6 = (unsigned)UART0_Int;
//Разрешить прерывания
  VICIntEnable |= 0x00000040;
```

```
}

int main(void)
{
      UART0_Init();
      for (;;){}
}
```

Практическая часть

Задание 1. Ознакомиться с теоретическим материалом на стр. 2-10.

Задание 2. Создать проект С программы в среде Keil uVision для микроконтроллера NXP LPC2478 с частотой генератора, указанной в индивидуальном задании.

Задание 3. Определить параметры M, N, CLKSEL(7:0), PCLKSEL0, PCLKSEL1, DLL, DLM, FDR для синхронизации приема и передачи информации по UART0.

Задание 4. Разработать программу функционирования микроконтроллера по индивидуальному заданию. Реализовать два варианта процедуры обработки прерываний: быстрое прерывание FIQ и векторное прерывание IRQ.

Задание 5. Протестировать правильность функционирования программы с помощью среды Keil uVision. Результаты занести в отчет.

Требования к отчету

Отчет по работе должен содержать: задание, листинги программ функционирования микроконтроллера для FIQ и IRQ прерываний, текст программы, результаты тестирования программы, выводы о работоспособности программы.

Контрольные вопросы

1. Для чего предназначен блок VIC микроконтроллера?
2. В чем отличие прерывания IRQ от прерывания FIQ?
3. Сколько линий прерываний подключено к VIC?
4. Как выглядит пакет, передаваемый UART по интерфейсу RS232 для передачи 8-бит числа 0x0f с одним стоповым битом и контролем по четности.

5. Перечислите программно задаваемые параметры, определяющие частоту передачи по UART.

Приложение 1. Назначение линий прерываний каналам VIC.

Канал	Модуль	Описание источников прерываний
0	WDT	Watchdog Interrupt (WDINT)
1	-	Зарезервировано за Software Interrupt
2	Ядро ARM	Embedded ICE, DbgCommRx
3	Ядро ARM	Embedded ICE, DbgCommTx
4	TIMER0	Match 0 - 1 (MR0, MR1), Capture 0 - 1 (CR0, CR1)
5	TIMER1	Match 0 - 2 (MR0, MR1, MR2), Capture 0 - 1 (CR0, CR1)
6	UART0	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
7	UART1	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
8	PWM0, PWM1	Match 0 - 6 of PWM0, Capture 0 of PWM0, Match 0 - 6 of PWM1, Capture 0-1 of PWM1
9	I2C0	SI (state change)
10	SPI, SSP0	SPI Interrupt Flag of SPI (SPIF), Mode Fault of SPI (MODF), Tx FIFO half empty of SSP0, Rx FIFO half full of SSP0, Rx Timeout of SSP0, Rx Overrun of SSP0
11	SSP 1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
12	PLL	PLL Lock (PLOCK)
13	RTC	Counter Increment (RTCCIF) Alarm (RTCALF) Subsecond Int (RTCSSF)

Канал	Модуль	Описание источников прерываний
14	System Control	External Interrupt 0 (EINT0)
15	System Control	External Interrupt 1 (EINT1)
16	System Control	External Interrupt 2 (EINT2), LCD
17	System Control	External Interrupt 3 (EINT3)
18	ADC0	A/D Converter 0 end of conversion
19	I2C1	I2C1 SI (state change)
20	BOD	Brown Out detect
21	Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.
22	USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA
23	CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx
24	SD/ MMC interface	RxDataAvlbl, TxDataAvlbl, RxFifoEmpty, TxFifoEmpty, RxFifoFull, TxFifoFull, RxFifoHalfFull, TxFifoHalfEmpty, RxActive, TxActive, CmdActive, DataBlockEnd, StartBitErr, DataEnd, CmdSent, CmdRespEnd, RxOverrun, TxUnderrun, DataTimeOut, CmdTimeOut, DataCrcFail, CmdCrcFail
25	GP DMA	IntStatus of DMA channel 0, IntStatus of DMA channel 1
26	Timer 2	Match 0-3 Capture 0-1
27	Timer 3	Match 0-3 Capture 0-1
28	UART2	Rx Line Status (RLS), Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
29	UART3	Rx Line Status (RLS),

Канал	Модуль	Описание источников прерываний
		Transmit Holding Register Empty (THRE), Rx Data Available (RDA), Character Time-out Indicator (CTI), End of Auto-Baud (ABEO), Auto-Baud Time-Out (ABTO)
30	I2C2	SI (state change)
31	I2S	irq_rx irq_tx

Приложение 2. Определения параметров делителя частоты UART0/2/3.

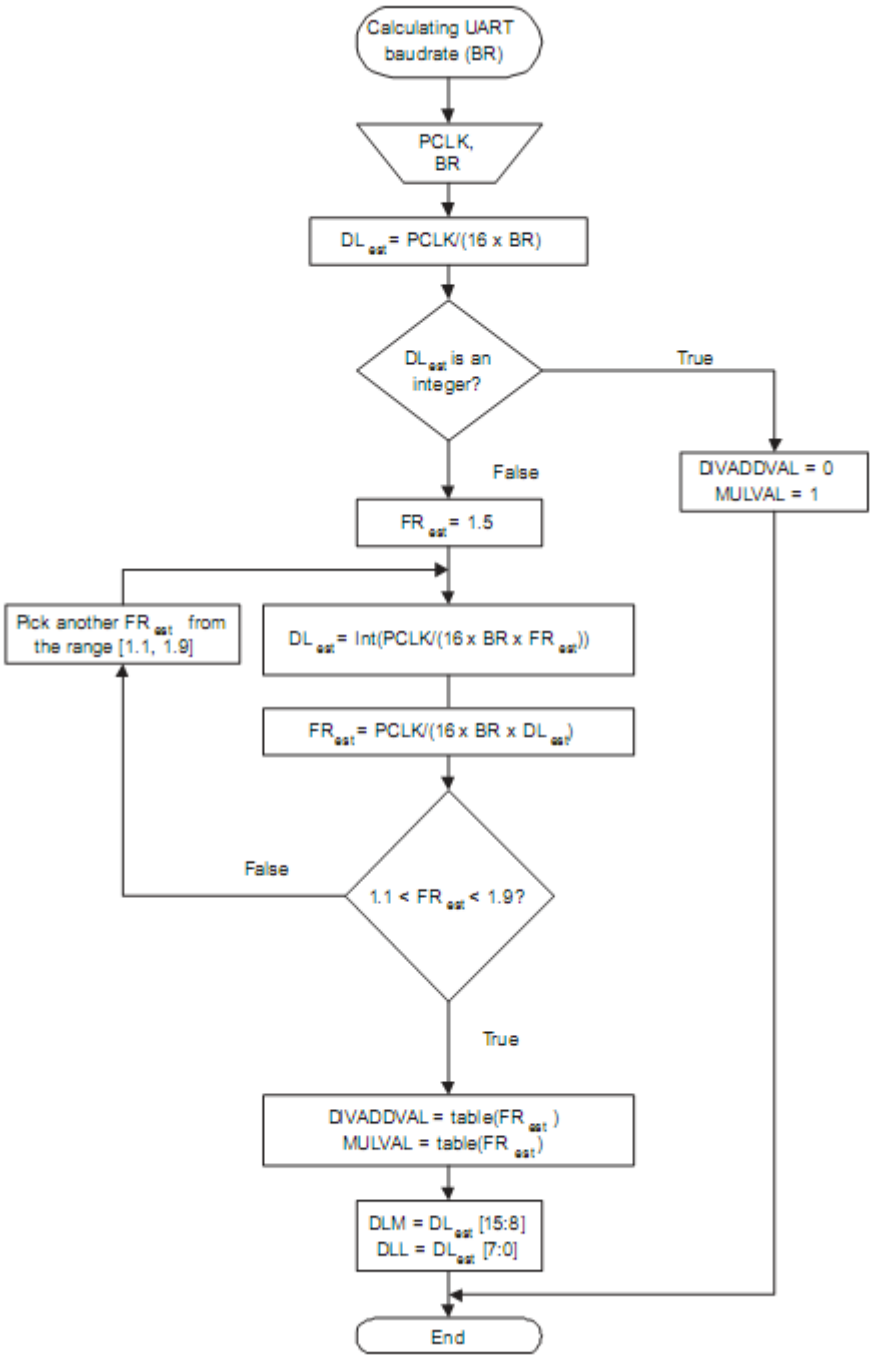

Table 393. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15