ПРИМЕНЕНИЕ ПРОЦЕССОРОВ СЕРИИ «МУЛЬТИКОР»

Работа с каналами DMA

ОАО НПЦ «ЭЛВИС»
support@elvees.com, www.multicore.ru

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

В документе поясняются принципы работы с каналами DMA в процессорах серии «Мультикор».

На примере микросхемы 1892ВМ10Я приведены алгоритмы работы с DMA. В приведенных примерах кода использованы адреса, значения и настройки для данной микросхемы.

Подробно рассматриваются четыре алгоритма:

- алгоритм обмена по DMA периферийных портов;
- алгоритм обмена по DMA периферийных портов с обработкой прерываний;
- алгоритм обмена по DMA типа “память-память”;
- алгоритм обмена по DMA цепочкой.

Документ применим к микросхемам 1892ВМ7Я, 1892ВМ8Я, 1892ВМ10Я, 1892ВМ12АТ, 1892ВМ15АФ.

2. ПРИНЦИПЫ ОБМЕНА ПО КАНАЛАМ DMA

Каналы DMA используются для обмена данными с внешними устройствами или между областями памяти без участия процессорных ядер. Это позволяет достичь двух целей:

- разгрузить процессор: во время обмена он может выполнять другие действия;
- увеличить скорость обмена, так как цикл вычитывания данных из устройства и сохранение данных в память - это минимум 5 инструкций процессора, что значительно дольше, нежели прямая передача данных с помощью DMA.

Обмен по DMA может происходить 32- или 64-разрядными словами. В общем случае, размер слова для обмена по DMA зависит от конкретного типа канала:

- DMA периферийных портов (кроме Ethernet) - 64 разряда;
- DMA типа "память-память" - 32/64 разряда (настраивается);
- DMA Ethernet - 8 разрядов.

Разрядность каналов DMA для конкретного процессора необходимо уточнять в руководстве пользователя.

2.1 Алгоритмы обмена по DMA

2.1.1 Алгоритм обмена по DMA периферийных портов

В этом разделе рассмотрен обмен по DMA через порт MFBSP в режиме LPORT.

Для каждого порта MFBSP предусмотрено по два независимых канала DMA - один на приём и один на передачу. Обмен происходит пачками. Пачка – это количество слов, переданное за одно предоставление прямого доступа. Размер пачки определяется полем WN регистра CSR соответствующего канала DMA.

В режиме работы DMA с периферийными портами обмены возможны только 64-разрядными словами. Поэтому следует проектировать протокол обмена таким образом, чтобы размер передаваемых пакетов был выровнен по границе 64-разрядного слова. Это позволит избежать ситуации, когда принято нечетное количество 32-разрядных слов, и последнее слово остается в FIFO MFBSP до прихода следующих слов данных. Если избежать такой ситуации нельзя, в настройках каналов DMA необходимо задавать длину передачи на одно 64-разрядное слово меньше, и по окончанию приема данных дополнительно программно вычитывать оставшееся 32-разрядное слово через псевдорегистр RX_MFBSP.

Предусмотрена работа с обработкой прерываний от каналов DMA: прерывание может формироваться по завершению передачи или приема всего блока данных и, применительно к порту MFBSP, при его возникновении устанавливается в единицу в

регистре QSTR2. Предварительно прерывание необходимо разрешить, установив соответствующие биты в регистре MASKR2.

Для корректного базового обмена между двумя периферийными портами необходимо произвести следующие действия с портами и каналами DMA:

- сформировать массивы для приема и передачи;
- как и при обмене без DMA, необходимо инициализировать порты;
- настроить каналы DMA на прием и передачу;
- ожидать окончания приема данных;
- проверить содержимое входного массива (опционально);
- обнулить регистры управления и состояния CSR.

Формирование массивов для приема и передачи подразумевает под собой создание двух массивов одинаковой длины, выравнивание их по границе 64-разрядного слова (пример кода приведен в примере MFBSP_LPORT_DMA составе MCStudio 3M/4), заполнение выходного массива OutputArray данными для передачи, заполнение входного массива InputArray нулями. Инициализация регистров управления и состояния портов MFBSP, то есть настройка одного порта на передачу, другого на приём, происходит аналогично их инициализации при обмене без DMA. В приведенном примере обмен производится между портами LPORT2 и LPORT3. Ниже приведены фрагменты кода на Си с комментариями.

2.1.1.1 Инициализация портов

```
void LPORT_Init() {
  unsigned int LCLK_RATE;
// TR_CRAM бит, вектора прерывания размещаются во внутренней памяти
//CRAM (адреса типа 0xB8000000)
  CSR = 1 << 1;
  LCLK_RATE = (FREQ / (2 * LPORT_FRQ)) - 1;
  // LPORT2 передает
  CSR_MFBSP2 = ((LCLK_RATE & 0x1E) << 10) | // LCLK_RATE[4:1]
               ((LCLK_RATE & 1) << 2) |     // LCLK_RATE[0]
               (1 << 6) | // LDW = 1 (длина слова - 8 бит)
               (1 << 1) | // LTRAN = 1 (передатчик)
               1;         // LEN = 1;
  // LPORT3 принимает
  CSR_MFBSP3 = ((LCLK_RATE & 0x1E) << 10) | // LCLK_RATE[4:1]
               ((LCLK_RATE & 1) << 2) | // LCLK_RATE[0]
               (1 << 6) | // LDW = 1 (длина слова - 8 бит)
               1;         // LEN = 1;
}
```

Далее необходимо настроить каналы DMA на прием и передачу (регистры управления и состояния CSR приемного и передающего каналов) и определить расположение выходного и входного массивов (регистры IR приемного и передающего каналов).

2.1.1.2 Настройка каналов DMA на прием и передачу

```
// Настройка DMA LPORT3 на прием
unsigned int v_to_phy(unsigned int addr) {
  // предполагаем, что работаем только в kseg1
  return (addr - 0xA0000000);
}
// Запись физического адреса в регистр индекса
IR_MFBSP_RX_CH3 = ((unsigned int) &InputArray[0]);
```

32-разрядный индексный регистр IR содержит физический начальный адрес памяти с точностью до байта. После каждой передачи данных к индексу IR прибавляется смещение, равное количеству переданных байтов.

DMA, в отличие от процессорного ядра, работает с физическими адресами, поэтому только их, но не виртуальные адреса нужно задавать при определении регистра IR.

```
// WCX = LEN/2
// WN = 15 (размер пачки - 16 слов)
// RUN = 1
CSR_MFBSP_RX_CH3 = ((ARRAY_LEN / 2) << 16) | (15 << 2) | 1;
// Настройка DMA LPORT2 на передачу и старт обмена
// Запись физического адреса в регистр индекса
IR_MFBSP_TX_CH2 = ((unsigned int) &OutputArray[0]);
// WCX = LEN/2
// WN = 15 (размер пачки - 16 слов)
// RUN = 1
CSR_MFBSP_TX_CH2 = ((ARRAY_LEN / 2) << 16) | (15 << 2) | 1;
```

В поле WCX должна содержаться информация о числе 64-разрядных слов данных, которые должен передать канал DMA (блок данных). Количество передаваемых слов: (WCX + 1). Содержимое этого поля уменьшается на 1 после передачи каналом DMA очередного слова данных.

В поле WN содержится информация о числе слов данных, передаваемых в одной пачке. Размер пачки равен WN+1 слов.

В поле RUN записывается единица (канал переводится в состояние обмена), то есть, канал начнет работать сразу после записи.

IM - маска разрешения установки признака END, используется только в процедуре инициализации. В поле IM должен быть записан ноль, поскольку цепочки передач не используются, состояние этого бита безразлично. Он разрешает прерывание по окончанию передачи одного звена цепочки.

В остальных полях регистра CSR записаны нули.

2.1.1.3 Ожидание окончания приема данных, верификация данных и обнуление CSR

Окончание обмена, то есть момент, когда приемный канал DMA закончит работу, можно проконтролировать, отслеживая признак завершения передачи данных – бит DONE регистра CSR DMA (он установится в единицу). Состояние этого бита читаем через псевдорегистр RUN, чтобы предотвратить сброс CSR после его чтения:

```
while ((RUN_MFBSP_RX_CH3&0x8000)==0) ;
```

В заключительных пунктах алгоритма проверим, совпадают ли входной и выходной массивы и обнулим регистры CSR.

В составе MCStudio 3M/4 существуют следующие примеры обмена по DMA с описанным выше алгоритмом: MFBSP_LPORT_DMA, MFBSP_I2S_DMA, MFBSP_SPI_DMA.

2.1.2 Алгоритм обмена по DMA периферийных портов с обработкой прерываний

Архитектура процессоров серии «Мультикор» предусматривает возникновение и обработку прерываний по окончании пересылки данных по каналам DMA.

Для корректного обмена по DMA с обработкой прерываний необходимо произвести следующие действия с портами и каналами DMA:

- установить значения регистра CP0. Status, разрешая прерывания;
- установить соответствующие биты масок системного регистра MASKR2;
- настроить режим размещения векторов прерываний TR_CRAM во внутренней памяти;
- инициализировать порты;
- настроить каналы DMA на прием и передачу;
- ожидать окончания обработки прерывания;
- проверить содержимое входного массива (опционально).
- обнулить регистры управления и состояния CSR.

2.1.2.1 Установка значения регистра CP0. Status, разрешающая прерывания

В регистр CP0. Status необходимо записать значение «0x1001», разрешая, таким образом, прерывания вообще (бит CP0.Status[0]) и, конкретно, прерывания от портов MFBSP (бит CP0.Status[11]).

Функция, реализующая запись в регистр CP0.Status:

```
// установка значения регистра CP0 Status
void SetCP0_Status(unsigned int value) {
  asm volatile("mtc0 %0, $12" ::"r"(value));
}
```

Непосредственно, запись:

```
SetCP0_Status(0x1001);
```

2.1.2.2 Установка соответствующих битов масок системного регистра MASKR2

Записывая единицу в соответствующий разряд регистра MASKR2, разрешается прерывание от конкретного канала DMA:

```
// установка регистра MASKR2 на прерывание
void MASKR2_set() {
  // настройка регистра маски прерываний (MASKR2)
  // выставление разряда MASKR2[29], разрешающего прерывание от DMA
  // канала
  SYS_REG.MASKR2.data |= (1 << 29);
```

2.1.2.3 Настройка режима размещения векторов прерываний TR_CRAM во внутренней памяти

Адрес обработчика прерываний зависит от состояния разрядов BEV, EXL регистра CP0.Status, IV регистра CP0.Cause, TR_CRAM регистра CSR. Работа с прерываниями подробна описана в отдельном документе, посему, не вдаваясь здесь в подробные описания, записываем единицу в бит TR_CRAM – разряд [1] системного регистра CSR.

2.1.2.4 Инициализация портов

Инициализация регистров управления и состояния портов LPORT2 и LPORT3 происходит аналогично их инициализации при базовом обмене по каналу DMA, описанном в предыдущей главе:

```
// инициализация регистров управления и состояния портов LPORT2 и
//LPORT3
void LPORT_Init() {
  unsigned int LCLK_RATE;

  // TR_CRAM бит, вектора прерывания размещаются во внутренней
  //памяти CRAM
  // (адреса типа 0xB8000000)
  SYS_REG.CSR.data = 1 << 1;

  LCLK_RATE = (FREQ / (2 * LPORT_FRQ)) - 1;
  // LPORT2 передает
  MFBSP2.CSR.bits.LCLK_RATE_1_4 = 2;
  MFBSP2.CSR.bits.LCLK_RATE_0 = 1;
  MFBSP2.CSR.bits.LDW = 1;   //длина слова-8 бит
  MFBSP2.CSR.bits.LTRAN = 1; //передатчик
  MFBSP2.CSR.bits.LEN = 1;
  // LPORT3 принимает
  MFBSP3.CSR.bits.LCLK_RATE_1_4 = 2;
  MFBSP3.CSR.bits.LCLK_RATE_0 = 1;
  MFBSP3.CSR.bits.LDW = 1; //длина слова- 8бит
  MFBSP3.CSR.bits.LEN = 1;
```

2.1.2.5 Настройка каналов DMA на прием и передачу

Далее необходимо настроить каналы DMA на прием и передачу (регистры управления и состояния CSR приемного и передающего каналов) и определить расположение выходного и входного массивов (регистры IR приемного и передающего каналов).

```
// Настройка DMA LPORT3 на прием
// запись физического адреса в регистр индекса
DMA_MFBSP_RX_CH3.IR.data = ((unsigned int)&InputArray[0]) -
0xA0000000;

// WCX =ARRAY_LEN/2
DMA_MFBSP_RX_CH3.CSR.bits.WCX = ARRAY_LEN / 2;

// WN = 15(размер пачки -16 слов)
DMA_MFBSP_RX_CH3.CSR.bits.WN = 15;

// RUN = 1
DMA_MFBSP_RX_CH3.CSR.bits.RUN = 1;

// Настройка DMA LPORT2 на передачу
// записываем физический адрес в регистр индекса
DMA_MFBSP_TX_CH2.IR.data = ((unsigned int)&OutputArray[0]) -
0xA0000000;

// WCX = ARRAY_LEN/2
DMA_MFBSP_TX_CH2.CSR.bits.WCX = ARRAY_LEN / 2;

// WN = 15 (размер пачки - 16 слов)
DMA_MFBSP_TX_CH2.CSR.bits.WN = 15;

// RUN = 1
DMA_MFBSP_TX_CH2.CSR.bits.RUN = 1;
```

2.1.2.6 Ожидание окончания обработки прерывания

Далее ожидается окончание обработки прерывания. Обработкой возникшего прерывания по окончании обмена занимается специальная программа – обработчик. Она сохраняет регистры 1-28,30,31 в стеке, когда приходит прерывание, и процессор его обрабатывает, затем программист снимает прерывание, сбрасывая биты DONE и END регистров управления и состояния DMA портов, затем обработчик возвращает программу по адресу дальнейшего исполнения, восстанавливая регистры 1-28,30,31 в стеке и восстанавливая указатель стека.

В составе MCStudio 3M/4 существует следующий набор примеров обмена по DMA с обработкой прерывания: MFBSP_LPORT_DMA_int, MFBSP_I2S_DMA_int, MFBSP_SPI_DMA_int.

2.1.3 Алгоритм обмена по DMA типа «память-память»

В процессорах серии «Мультикор» предусмотрена передача данных по каналу DMA из одной области памяти в другую. Имеются 4 канала MEM_CH, которые обеспечивают обмен данными между двумя областями любых блоков памяти (внутренних или внешних).

Такой тип передачи отличается от рассмотренных выше DMA-передач тем, что не нужно инициализировать порты, так как данные будут передаваться внутри адресного пространства процессора. Отличие также будет в настройке канала DMA.

В регистрах индексов IR1 и IR0 определяются начальные физические адреса источника и приемника данных:

```
//адрес приемника
DMA_MEM_CH0.IR0.data = v_to_phy((unsigned int)&InputArray[0]);
// адрес источника
DMA_MEM_CH0.IR1.data = v_to_phy((unsigned int)&OutputArray[0]);
```

Адрес источника и адрес приемной области памяти меняются местами в зависимости от бита DIR регистра CSR канала DMA, с которым происходит работа.

В регистре OR задаётся смещение в адресном пространстве, через которое DMA-канал считывает очередное слово из памяти для передачи и записывает в область памяти, являющуюся источником:

```
// Старшая часть регистра задает смещение регистра IR1, а младшая
// часть - смещение регистра IR0 после передачи каждого слова
OR_MEM_CH0 = 0x00010001;
```

В регистре управления и состояния определяется число 64-разрядных слов данных, которые должен передать канал DMA:

```
CSR_MEM_CH0 = ((LEN) << 16) | // WCX
              (1 << 1) |      // DIR =1
              1;              // RUN = 1
```

Разряд DIR регистра CSR_MEM_CH0 определяет направление обмена данными, в данном случае он равен единице и данные перемещаются из области памяти, указанной регистром IR1 в область памяти, указываемую регистром IR0.

Определения значений других регистров для такой передачи не требуется.

Далее ожидается флага окончания DMA-передачи:

```
  while (DMA_MEM_CH0.RUN.data &(1<<15) != 0);
```

Полученный массив сравнивается с отправленным, обнуляется регистр управления и состояния, индексные регистры.

Код программы для описанного обмена содержится в примере MEM_DMA в составе MCStudio 3M/4.

2.1.4 Алгоритм обмена по DMA цепочкой передач

Каналы DMA MEM_CH могут выполнять процедуру самоинициализации - выполнять передачи DMA цепочкой. Особенность данного режима в том, что в этом случае обмены по DMA выполняются друг за другом без участия процессора, что позволяет осуществлять быструю передачу больших объемов данных.

Как пример можно привести передачу данных на дисплей в порте видеовыхода (VPOUT). Максимальный объем передаваемых данных за один DMA-обмен - 65536 64-разрядных слов, то есть, 512 Кбайт. В то же время, один кадр в разрешении 640x480 занимает в памяти более 1 Мбайт. Поэтому для вывода видеоизображений на дисплей необходимо использовать цепочки передач DMA.

Для реализации цепочки передач необходимо формировать массив блоков параметров, содержащих значения регистров IR1, IR0, Y, OR, CSR, CP канала DMA MEM_CH. Блок параметров - это набор значений регистров, расположенных так, как показано на схеме ниже.

```
63                                                0
{IR132,                                    IR032};
{{WCY16,ORY16},   {OR116,OR016}};
{ CSR32,                                   CP32}.
```

Необходимо сформировать такое количество блоков параметров, сколько передач будет в цепочке. Эти параметры при самоинициализации аппаратно загружаются в 64-разрядном формате в соответствующие регистры канала DMA.

Для выполнения самоинициализации в каналах имеется 32-разрядный регистр CP, в котором хранится начальный адрес блока параметров очередного DMA обмена, то есть, физический адрес первого элемента первого вызываемого массива. Младший разряд регистра CP (бит CHEN) используется для старта режима самоинициализации.

Алгоритм обмена будет таким:

- инициализируется массив параметров (регистров IR1, IR0, Y, OR, CSR, CP);
- инициализируются массивы на передачу и прием, также как в обменах DMA, описанных в предыдущих главах;
- в регистр CP заносится физический адрес первого блока параметров;
- ожидается окончание DMA-передачи – прерывание от канала;
- проверяется значение входного массива, обнуляются регистры CSR, IR0, IR1.

Ниже приведена реализация алгоритма на языке Си.
Инициализируется массив параметров, создаётся структура с регистрами DMA в качестве ее полей:

```
 for (j = 0; j < N; j++) {
  //физический адрес "источника" определяется регистром индекса
  settings_array[j].IR1_reg =
      v_to_phy((unsigned int)&OutputArray[j * LEN / N]);

 //физический адрес "приемника" также определяется регистром
//индекса
  settings_array[j].IR0_reg =
      v_to_phy((unsigned int)&InputArray[j * LEN / N]);

  //двумерной адресации нет,
  //поэтому регистр смещения по направлению Y равен 0
  settings_array[j].Y_reg = 0;

  // смещение при чтении из "источника" - 1 слово
  settings_array[j].OR_reg = 0x00010001;
  if (j == (N - 1))
    // DATA_LEN | DIR (1->0) | RUN
    settings_array[j].CSR_reg = (LEN / N << 16) | (1 << 1) | 1;
  else
    // DATA_LEN | CHEN | DIR (1->0) | RUN,
    // если это последний блок параметров,
    //бит CHEN не выставляется CHEN – бит разрешения выполнения
    //процедуры самоинициализации.
    settings_array[j].CSR_reg = (LEN / N << 16) | (1 << 12) | (1 <<
1) | 1;
  // адрес следующего блока параметров
  if (j == (N - 1))
    settings_array[j].CP_reg = 0;
  else
settings_array[j].CP_reg = v_to_phy((unsigned int)&settings_array[j
+ 1]);
}
```

Далее задаётся адрес первого блока параметров DMA передачи через регистр CP.

Нулевой разряд CP устанавливается в единицу, это будет стартом самоинициализации:

```
CP_MEM_CH0 =  v_to_phy ((unsigned int)&settings_array[0]) |1;
```

Когда в бит CP[0] записывается единица, соответствующий канал DMA производит чтение блока параметров по адресу {CP[31:1],0}. Прочитанные значения заносятся в регистры данного канала DMA. Если в разряде CSR[0] окажется ноль - передача не запустится, канал DMA будет ждать записи единицы в этот разряд. После старта передачи обмен ничем не отличается от обмена без цепочек.

По окончании DMA-обмена всегда (даже если обмен запускался обычным путем, как единичный обмен) проверяется состояние бита CHEN в регистре CSR. Если он выставлен в единицу - контроллер DMA считывает блок параметров по адресу, содержащемуся в регистре CP, и заносит эти значения в регистр. Соответственно, последний блок параметров (если цепочка не зацикленная) должен содержать ноль в разряде CHEN регистра CSR.

Флагом окончания DMA передачи по используемому нами в примере каналу DMA_MEM_CH0, является нулевой бит регистра QSTR1 (см. главу 2.4 руководства пользователя на микросхему 1892ВМ10Я), он и отслеживается после старта процедуры самоинициализации:

```
while (!(QSTR1&1));
```

Дальше проверяется корректность принятого массива и обнуляется регистр управления и состояния и регистры индексов, как это было сделано в предыдущих примерах.

Код на языке Си для описанного обмена содержится в примере MEM_DMA_chain в составе сред разработки MCStudio 3M и MCStudio 4.

Цепочки передач могут использоваться как с каналами DMA типа "память-память", так и с каналами DMA портов. Процедура инициализации DMA портов MFBSP, EMAC, VPIN, VPOUT аналогична каналам MEM_CH. Параметры для самоинициализации размещаются в памяти в двух последовательных 64-разрядных словах, следующим образом (в порядке возрастания адресов):

Таблица 2.1. Параметры для самоинициализации

Смещение	Параметр
0x00	IR
0x04	-
0x08	CSR
0x0C	CP

Код программы для описанного обмена содержится в примере MEM_DMA_chain_LPORT в составе MCStudio 3M/4.

3. ИСТОРИЯ ИЗМЕНЕНИЙ

3.1 19 января 2017

- Обновлены колонтитулы и стили.
- Добавлена глава «Введение».