**Блок** **Init\_DMA\_BASE\_out** **бло**к  **возвращает буфер значений полученный от периферийного устройства**

|  |  |  |
| --- | --- | --- |
| Картинка блока в палитре | Картинка блока на схеме | Векторизован ли блок, генерится ли блок в СИ код (да) |

**Прямой доступ к памяти** (англ. DMA – Direct Memory Access) представляет собой механизм, позволяющий периферийным устройствам напрямую, без использования  
центрального процессора, обращаться к оперативной памяти или памяти  
программ. Поскольку для сокращенного обозначения наибольшей  
популярностью пользуется английская аббревиатура **DMA**, то и мы также  
будем использовать ее.  
Использование DMA позволяет существенно разгрузить центральный  
процессор, не отвлекая его на взаимодействие с периферийными устройствами.

Блок использует контроллер DMA для получения массива заданного размера от входящего тракта периферийного устройства и формирует выходной массив заданного размера. Но так же может использоваться для копирования областей памяти микроконтроллера.

Блок обращается к функции DMA\_BASE() находящейся в dma.h проекта шаблона сборки кода. Функция может быть изменена пользователем.

**Входные порты**

**Dev\_init** определяет последовательность включения блока в работу используется после настройки периферийного устройства

**sort\_block** необходимо для определения порядка сортировки блока

**Выходные порты**

**ready** значение 1, заданный массив заполнен

**Uint32\_t Buf** возвращает массив со значениями полученными от периферийного устройства

**Свойства**

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Имя | Описание | По умолчанию |
| **Адрес приемника\источника данных** | **addr** | Адрес регистра данных периферийного устройства АЦП | 0x40088018 (АЦП1) |
| **Канал DMA** | **Chanal** | Канал DMA | 8 |
| **Размер выходного массива** | **SIZE** | Размер массива выходного буфера | 32 |
| **Автоувеличение адреса для источника** | **DMA\_SourceIncSize** | Автоматическое увеличение адреса источника | DMA\_SourceIncNo |
| **Автоувеличение адреса для приемника** | **DMA\_DestIncSize** | Автоматическое увеличение адреса приемника | DMA\_DestIncWord |
| **Размер посылки** | **DMA\_MemoryDataSize** | Размер элемента массива | Word |
| **Использовать прерывание** | **Intr** | После завершения цикла DMA возникает прерывание, для продолжения работы необходимо пери инициализировать контроллер DMA, после чего контроллер DMA будет готов к следующему циклу работы. В результате получение данных будет проходить в асинхронном режиме. | нет |
| **Запрос к DMA** | **Request** | Если устройству не определен канал DMA, то можно выбрать свободный и для того, что бы происходило выполнения цикла DMA в момент обращения к блоку, а не асинхронно, как в варианте, когда используем пере инициализацию в обработчике прерывания, можно реализовать запрос к каналу. | Нет |

Примечание! В случае если устройству назначен канал и мы хотим получать результаты работы блока при обращении к нему, то не стоит использовать ни режим прерывания, ни режим запроса, пере инициализация будет происходить в момент обращения к блоку простым обращением к функции DMA\_Init().

В случае если устройству не выделен канал DMA, а мы хотим выполнения цикла в момент обращения к блоку то следует выбрать режим Request, в этом случае после обращения к функции DMA\_Init(), будет произведено обращение к функции Request() и произойдет цикл работы контроллера DMA.

**Пример dma\_acd.prt, dma\_acd\_dac.prt**

**Блок** **Init\_DMA\_BASE\_in** **бло**к **передает буфер на периферийное устройство**

|  |  |  |
| --- | --- | --- |
| Картинка блока в палитре | Картинка блока на схеме | Векторизован ли блок, генерится ли блок в СИ код (да) |

Блок использует контроллер DMA для передачи массива заданного размера на регистры периферийного устройства. Но так же может использоваться для копирования областей памяти микроконтроллера.

Блок обращается к функции DMA\_BASE() находящейся в dma.h проекта шаблона сборки кода. Функция может быть изменена пользователем.

**Входные порты**

**Dev\_init** определяет последовательность включения блока в работу используется после настройки периферийного устройства

**buf** буфер с цифровыми значениями для передачи периферийным устройством

**sort\_block** необходимо для определения порядка сортировки блока

**Выходные порты**

**ready** значение 1, заданный массив заполнен

**Свойства**

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Имя | Описание | По умолчанию |
| **Адрес приемника\источника данных** | **addr** | Адрес регистра данных периферийного устройства | 0x40090008 (ЦАП2) |
| **Канал DMA** | **Chanal** | Канал DMA | 8 |
| **Размер выходного массива** | **SIZE** | Размер массива выходного буфера | 32 |
| **Автоувеличение адреса для источника** | **DMA\_SourceIncSize** | Автоматическое увеличение адреса источника | DMA\_SourceIncNo |
| **Автоувеличение адреса для приемника** | **DMA\_DestIncSize** | Автоматическое увеличение адреса приемника | DMA\_DestIncWord |
| **Размер посылки** | **DMA\_MemoryDataSize** | Размер элемента массива | Word |
| **Использовать прерывание** | **Intr** | После завершения цикла DMA возникает прерывание, для продолжения работы необходимо пери инициализировать контроллер DMA, после чего контроллер DMA будет готов к следующему циклу работы. В результате получение данных будет проходить в асинхронном режиме. | нет |
| **Запрос к DMA** | **Request** | Если устройству не определен канал DMA, то можно выбрать свободный и для того, что бы происходило выполнения цикла DMA в момент обращения к блоку, а не асинхронно, как в варианте, когда используем пере инициализацию в обработчике прерывания, можно реализовать запрос к каналу. | нет |

Примечание! В случае если устройству назначен канал и мы хотим получать результаты работы блока при обращении к нему, то не стоит использовать ни режим прерывания, ни режим запроса, пере инициализация будет происходить в момент обращения к блоку простым обращением к функции DMA\_Init().

В случае если устройству не выделен канал DMA, а мы хотим выполнения цикла в момент обращения к блоку то следует выбрать режим Request, в этом случае после обращения к функции DMA\_Init(), будет произведено обращение к функции Request() и произойдет цикл работы контроллера DMA.

Пример **dma\_dac.ptr, dma\_acd\_dac.prt**

**Важно! При использовании DMA при приеме/передачи информации от периферийных устройств использование прерываний от внешних источников может привести к сбою в работе контроллера DMA. При использовании контроллера DMA режим отладки DEBUG рекомендовано отключить.**