Современные 32-разрядные ARM-микроконтроллеры серии STM32: прямой доступ к памяти

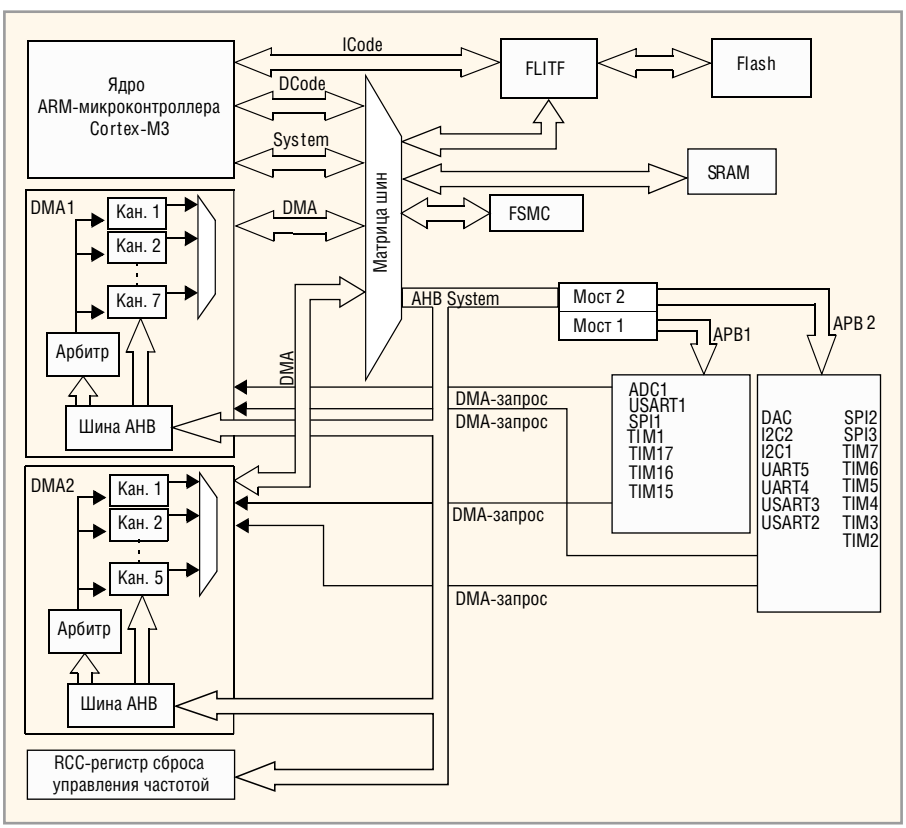
Олег Вальпа (г. Миасс, Челябинская обл.)

В статье приведено описание блока прямого доступа к памяти 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрена архитектура и состав его регистров, а также приведены практические примеры программ.

Введение

Многие микроконтроллеры имеют встроенный блок прямого доступа к памяти (ПДП), именуемый в англоязычной литературе как Direct Memory Access (DMA). Блок DMA позволяет обеспечить высокоскоростную передачу данных между внешними устройствами и памятью микроконтроллера, а также передачу данных типа память-память без участия процессора. Это позволяет освободить процессор от рутинной операции по пересылке данных, ускорить данную процедуру за счёт прямой пересылки данных и выполнять различные операции в микроконтроллере одновременно и независимо самим процессором и блоком DMA.

Во многих программах процессор часто бывает загружен операциями, требующими непрерывной передачи данных. Это и опрос состояния датчиков, и регулярные расчёты. Примером таких процедур могут служить: вывод информации на графический индикатор, опрос клавиатуры, чтение данных с карты памяти и тому подобное. Если в это время ещё потребуется обслуживать запрос от компьютера по последовательному порту, то процессор может начать притормаживать выполнение некоторых операций. В подобных случаях DMA может помочь процессору. Для этого достаточно будет сформировать задание блоку DMA в виде начала буфера в памяти с указанием количества байтов этого буфера для передачи в регистр последовательного порта по мере его готовности. После передачи всех данных из буфера, DMA с помощью флага сообщит об этом процессору. Таким образом, процедура обслуживания запроса от компьютера по последовательному порту будет выполнена блоком DMA, что защитит процессор от перегрузки.



Структурная схема DMA

Функциональное описание DMA

Микроконтроллеры серии STM32 [1] также имеют DMA. Причём более насыщенные модели имеют два независимых блока DMA: DMA1 и DMA2, каждый из которых имеет несколько независимых каналов. Блок DMA1 содержит 7 каналов, а DMA2 – 5. Структурная схема DMA приведена на рисунке.

Блок DMA STM32 может быть использован такими основными периферийными устройствами микроконтроллера STM32, как SPI, ЦАП, I^2C, USART, таймеры и АЦП.

К каждому каналу можно подключить одно из периферийных устройств, закреплённых за этим каналом. Например, регистр передачи данных модуля USART1 можно подключить к каналу 4, а регистр приёма данных – к каналу 5. К этим же каналам можно подключить другую периферию. Например, за каналом 4 закреплена следующая периферия: TIM1_CH4, TIM1_TRIG, TIM1_COM, TIM4_CH2, SPI, I2C2_RX, I2C2_TX.

Блок DMA микроконтроллера STM32 имеет определённые особенности. В первую очередь это наличие 12 независимо конфигурируемых каналов: 7 для DMA1 и 5 для DMA2. Каждый из этих каналов связан с выделенными аппаратными DMA-запросами. На каждом канале поддерживается программный пусковой механизм. Конфигурация канала задаётся программно.

Приоритеты между запросами от каналов одного DMA контроллера задаются программно. Имеется четыре уровня приоритета: низкий, средний, высокий и очень высокий уровень. В случае равенства программных при-

оритетов, запросы регулируются аппаратно. Запрос с номером 1 имеет приоритет над запросом 2, и так далее.

Блок DMA обеспечивает независимые размерности данных для обмена между источником и адресатом в виде байта, полуслова и 32-разрядного слова. Поддерживается режим эмуляции упаковки и распаковки данных. Адреса источника и приёмника обмена могут быть выровнены по размерности элемента данных.

DMA поддерживает три флага событий: половина обмена DMA, завершение обмена DMA и ошибка обмена DMA. Эти флаги логически объединяются с помощью функции «ИЛИ» в единый запрос на прерывание для каждого канала.

Блок DMA способен работать в разных режимах обмена: «память–память», «периферия–память», «память–периферия» и «периферия–периферия».

Он обеспечивает доступ к памяти Flash, SRAM, памяти периферии, шинам периферии APB1, APB2 и AHB, как к источнику и как к приёмнику данных.

Программируемый размер данных для обмена может достигать значения 65536.

Если блок DMA и процессор микроконтроллера обращаются к одному и тому же устройству, то время обращения между ними будет распределено поровну. Механизм работы DMA позволяет занять шину данных на несколько рабочих тактов, а затем освобождает её.

Блок DMA выполняет прямой обмен с памятью, разделяя системную шину с ядром микроконтроллера. DMA-запрос может приостановить доступ процессора к системной шине на несколько тактов шины, если процессор и DMA работают с одним адресатом памяти или внешнего устройства. Матрица шин работает по алгоритму циклического планирования, которое гарантирует, по крайней мере, половину пропускной способности системной шины для процессора при одновременном обращении как к памяти, так и к периферии.

Описание регистров DMA

Для настройки контроллера DMA предусмотрено две категории регистров:

- регистры настройки и управления DMA в целом;
- регистры настройки и управления каждого канала.

Для настройки и управления DMA в целом предназначены регистры:

- DMA_ISR – регистр флагов прерывания от каналов;
- DMA_IFCR – регистр очистки флагов прерываний.

Для каждого канала служат следующие регистры:

- DMA_CCR – управляет режимом работы канала;
- DMA_CNDTR – содержит количество необходимых для передачи байт;

Таблица 1. Карта регистров DMA

Сдвиг	Регистр	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Резерв				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Исх. значение					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	Резерв				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Исх. значение					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Резерв																	MEM2MEM	PL[1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Исх. значение																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Резерв																	NDT[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Резерв																																
0x01C	DMA_CCR2	Резерв																	MEM2MEM	PL[1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Исх. значение																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Резерв																	NDT[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Резерв																																
…															…																		
0x080	DMA_CCR7	Резерв																	MEM2MEM	PL[1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Исх. значение																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_CNDTR7	Резерв																	NDT[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CMAR7	MA[31:0]																															
	Исх. значение	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Резерв																																

- DMA_CPAR – содержит указатель на периферийное устройство, которое подключено к каналу;
- DMA_CMAR – содержит указатель области памяти, в которой будут записаны или прочитаны данные.

Карта размещения регистров DMA в пространстве памяти представлена в таблице 1. В ней также показано значение регистров после сброса.

Каждый блок DMA включает в свой состав несколько групп регистров в соответствии с количеством обслуживаемых им каналов. На карте показаны группы регистров для каналов 1, 2 и 7. Регистры для остальных каналов имеют аналогичную структуру и одноимённые названия, отличающиеся лишь номером канала и смещением относительно базового адреса.

Рассмотрим подробнее назначение всех разрядов этих регистров. В последующих описаниях регистров все биты для каналов 6 и 7 относятся только к DMA2, так как DMA1 имеет всего 5 каналов.

Регистр статуса прерываний DMA_ISR имеет нулевое смещение адреса и обнуляется после сброса микроконтроллера. Назначение его бит следующее:

- биты 31...28 зарезервированы и при чтении имеют нулевое значение;
- биты 27, 23, 19, 15, 11, 7, 3 представляют собой флаги TEIFx ошибки обмена в канале x, где x – номер канала от 1 до 7. Эти биты устанавливаются аппаратно и обнуляются программно путём записи логической «1» в соответствующий бит регистра DMA_IFCR. Значение флага «0» означает отсутствие ошибки обмена в канале x, а значение «1» – наличие ошибки обмена в канале x;
- биты 26, 22, 18, 14, 10, 6, 2 представляют собой флаги HTIFx завершения половины обмена в канале x. Их установка и обнуление производится аналогично флагам TEIFx;
- биты 25, 21, 17, 13, 9, 5, 1 являются флагами TCIFx полного завершения обмена в канале x;
- биты 24, 20, 16, 12, 8, 4, 0 – это флаги GIFx, указывающие на возникшие прерывания в канале x.

Регистр DMA_IFCR служит для очистки флагов прерываний DMA, описанных выше. Его биты 31...28 зарезервированы, а остальные биты предназначены для обнуления описанных выше разрядов регистра DMA_ISR и имеют соответствующие им названия с добавлением префикса «C» от слова Clear (пер. с англ. – очистить). Установка и обнуление этих флагов производится программно.

Регистры DMA_CCRx нужны для конфигурации соответствующего канала x:

- биты 31...15 в нём зарезервированы и всегда читаются как «0»;
- бит 14 MEM2MEM позволяет запретить или разрешить режим работы «память-память» для DMA;
- биты 13...12 PL[1:0] определяют уровень приоритета канала и могут принимать следующие значения:
  - 00 – низкий приоритет,
  - 01 – средний приоритет,
  - 10 – высокий приоритет,
  - 11 – очень высокий приоритет;
- биты 11...10 MSIZE[1:0] задают размер данных в памяти и могут принимать значения:
  - 00 – 8 бит,
  - 01 – 16 бит,
  - 10 – 32 бита,
  - 11 – резерв;
- биты 9...8 PSIZE[1:0] определяют размер данных для периферии аналогично битам MSIZE[1:0];
- бит 7 MINC задаёт режим инкремента указателя в памяти и может принимать следующие значения:
  - 0 – режим инкремента указателя в памяти отключён,
  - 1 – режим инкремента указателя в памяти разрешён;
- бит 6 PINC служит для запрета и разрешения режима инкремента указателя периферии;
- бит 5 CIRC запрещает и разрешает режим цикличности;
- бит 4 DIR задаёт направление обмена данных:
  - 0 – чтение из периферии,
  - 1 – чтение из памяти;
- бит 3 TEIE запрещает и разрешает прерывания от ошибки обмена;
- бит 2 HTIE запрещает и разрешает прерывания от события завершения половины обмена;
- бит 1 TCIE запрещает и разрешает прерывания от события завершения обмена;
- бит 0 EN отключает и включает соответствующий канал в работу.

Регистр DMA_CNDTRx задаёт размер данных для канала x DMA:

- биты 31...16 зарезервированы и всегда читаются как «0»;
- биты 15...0 NDT[15:0] задают размер данных обмена в байтах от 0 до 65535.

В этот регистр можно производить запись только тогда, когда канал выключен. Как только канал будет разрешён, этот регистр можно будет только читать, чтобы определить число байт, оставшееся для обмена. Данный регистр декрементируется после каждой DMA-транзакции.

Как только обмен завершён, этот регистр может либо остаться обнулённым, либо автоматически перезагрузиться ранее запрограммированным значением, если канал сконфигурирован в режиме автоматической перезагрузки, то есть цикличности.

Если значение этого регистра равно нулю, никакая транзакция не может быть обслужена, причём независимо от того, разрешён канал или нет.

Регистр DMA_CPARx позволяет задать адрес периферии для канала x DMA. В этот регистр нельзя производить запись, когда канал разрешён. Значение его бит:

- биты 31...0 PA[31:0] задают базовый адрес регистра данных периферии, который будет участвовать в операциях чтения и записи.

Когда в регистре DMA_CCRx биты PSIZE[1:0]=01 задают 16-битный размер элемента данных, то бит PA[0] регистра DMA_CPARx игнорируется. Доступ к данным автоматически выравнивается по адресу полуслова. Когда биты PSIZE[1:0] =10 задают 32-битный размер элемента данных, игнорируются биты PA[1:0] регистра DMA_CPARx. Доступ к данным периферии автоматически выравнивается по адресу слова.

Регистр DMA_CMARx даёт возможность задать адрес памяти для канала x DMA. В этот регистр также нельзя производить запись, когда канал разрешён. Значение бит регистра:

биты 31...0 MA[31:0] задают базовый адрес в памяти микроконтроллера для данных, которые будут участвовать в операциях чтения и записи.

Когда в регистре DMA_CCRx биты MSIZE[1:0]=01 задают 16-битный размер элемента данных, то бит MA[0] регистра DMA_CMARx игнорируется. Доступ к данным автоматически выравнивается по адресу полуслова. Когда биты MSIZE[1:0]=10 задают 32-битный размер элемента данных, игнорируются биты MA[1:0] регистра DMA_CMARx, и доступ к данным памяти автоматически выравнивается по адресу слова.

DMA-транзакции и особенности работы

После возникновения запланированного события, периферия посыла-

ет сигнал запроса для DMA. Блок DMA обслуживает этот запрос в зависимости от приоритета канала. Как только DMA получает доступ к периферии, он посылает ей сигнал уведомления. Внешнее устройство снимает свой запрос после получения сигнала уведомления от DMA. Как только снимается сигнал запроса от периферии, DMA, в свою очередь, снимает свой сигнал уведомления. Если появятся новые запросы, периферия может инициализировать следующую транзакцию.

В итоге каждый DMA-обмен состоит из трёх операций:

1. Загрузка данных из регистров данных периферии или из области памяти, производимая с помощью внутреннего регистра текущего адреса периферии или памяти. Стартовый адрес, который используется для первого обмена, является базовым адресом периферии или памяти, и записывается в регистр DMA_CMARx или DMA_CPARx.

2. Сохранение данных путём загрузки в регистры периферии или в область памяти, адресуя их через внутренний регистр текущего адреса периферии или памяти. Стартовый адрес, который используется для первого обмена, является базовым адресом периферии или памяти, и программируется в регистре DMA_CMARx или DMA_CPARx.

3. Пост-декремент регистра DMA_CNDTRx, который содержит число транзакций, которые ещё необходимо выполнить.

Каждый канал может обработать DMA-обмен между регистром периферии, имеющим фиксированный адрес, и адресом в памяти. Количество данных, которые будут переданы, задаётся программно от 1 до 65535 элементов. Регистр, содержащий количество данных, которые необходимо передать, будет декрементироваться после каждой транзакции.

Размеры элементов данных обмена для периферии и памяти полностью программируется с помощью битовых полей PSIZE и MSIZE в регистре DMA_CCRx.

Указатели для периферии и памяти могут автоматически инкрементироваться после каждой транзакции, в зависимости от битов PINC и MINC в регистре DMA_CCRx. Если разрешён режим инкремента, то адрес следующей транзакции будет адресом предыдущей транзакции, увеличенный на 1, 2 или 4, в зависимости от выбранного размера элемента данных. Первый адрес обмена – это тот адрес, который запрограммирован в регистрах DMA_CPARx и DMA_CMARx. Во время операций обмена эти регистры сохраняют первоначально запрограммированное значение. Текущие адреса обмена, хранимые во внутреннем регистре текущего адреса периферии или памяти, недоступны программе.

Если канал сконфигурирован в нециклическом режиме, то после последней транзакции DMA-запрос не обслуживается, поскольку число элементов данных, которые нужно передавать, достигло нуля. Чтобы загрузить в регистр DMA_CNDTRx новое число элементов данных, которые надо передавать, DMA-канал должен быть заблокирован.

Если DMA-канал блокируется, то DMA-регистр не сбрасывается. Регистры DMA-каналов (DMA_CCRx, DMA_CPARx и DMA_CMARx) сохраняют начальные значения, запрограммированные на этапе конфигурации канала.

В циклическом режиме, после последней транзакции, регистр DMA_CNDTRx автоматически перезагружается первоначально запрограммированным значением. Внутренние регистры текущего адреса периферии или памяти перезагружаются значениями из регистров базового адреса DMA_CPARx или DMA_CMARx соответственно.

Конфигурация каналов

Чтобы сконфигурировать DMA-канал, необходимо выполнить следующую последовательность:

- задать адрес регистра периферии в регистре DMA_CPARx;
- задать адрес памяти в регистре DMA_CMARx;
- задать общее число данных в байтах, которые необходимо переместить, в регистре DMA_CNDTRx;
- задать приоритет канала с помощью битов PL[1:0] в регистре DMA_CCRx;
- задать направление перемещения данных, режим цикличности, режим инкремента периферии и памяти, размер элемента данных периферии и памяти и источник прерывания для завершения половины или всего обмена в регистре DMA_CCRx;
- активировать канал установкой бита ENABLE в регистре DMA_CCRx.

Как только канал будет разрешён, он сможет обслуживать DMA-запросы от периферии, подключённой к каналу.

Как только будет передана половина байтов, будет установлен флаг HTIF и сгенерировано прерывание, если такое прерывание разрешено битом HTIE. В конце обмена будет установлен флаг его завершения TCIF и сгенерировано прерывание, если такое прерывание разрешено битом TCIE.

Режим цикличности

Режим цикличности используется для управления непрерывным потоком данных с помощью кольцевых буферов, например, от АЦП в режиме сканирования. Эта особенность может быть разрешена битом CIRC в регистре DMA_CCRx. Когда режим цикличности активизирован, то значение числа данных, которые будут переданы за один цикл, автоматически перезагружается начальным значением, запрограммированным на этапе конфигурации канала, и DMA-запросы продолжают обслуживаться.

Режим «память–память»

Каналы DMA могут также работать без запроса от периферии. Такой режим называют «память–память». Если установлен «1» бит MEM2MEM в регистре DMA_CCRx, то канал инициализирует обмен сразу, как только он программно разрешается битом EN в регистре DMA_CCRx. Обмен останавливается в тот момент, когда значение в регистре DMA_CNDTRx достигает нуля. Режим «память–память» не может использоваться одновременно с режимом цикличности.

Обработка ошибок

Генерация ошибки обмена по DMA может возникнуть при операции чтения или записи в зарезервированном адресном пространстве. Если происходит ошибка обмена по DMA во время доступа на чтение или запись, то канал автоматически отключается посредством аппаратного сброса бита EN в регистре конфигурации соответствующего канала DMA_CCRx. Выставляется флаг запроса прерывания от ошибки обмена в регистре DMA_IFR канала TEIF и генерируется прерывание, если оно разрешено битом TEIE в регистре DMA_CCRx.

Таблица 2. Запросы прерываний от DMA

Событие прерывания	Флаг события	Бит управления разрешением
Завершение половины обмена	HTIF	HTIE
Завершение полного обмена	TCIF	TCIE
Ошибка обмена	TEIF	TEIE

Таблица 3. Сводная таблица запросов для DMA1

Периферия	Канал 1	Канал 2	Канал 3	Канал 4	Канал 5	Канал 6	Канал 7
ADC1	ADC1						
SPI/I2S		SPI1_RX	SPI1_TX	SPI/I2S2_RX	SPI/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

Таблица 4. Сводная таблица запросов для DMA2

Периферия	Канал 1	Канал 2	Канал 3	Канал 4	Канал 5
ADC3					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/DAC1			TIM6_UP/DAC1		
TIM7/DAC2				TIM7_UP/DAC2	
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

Прерывания

Запрос на прерывание может генерироваться событиями «завершение половины обмена», «завершение полного обмена» или «ошибка обмена» в каждом DMA-канале. Для удобства обработки доступны отдельные биты разрешения прерываний, представленные в таблице 2.

Карта запросов DMA

Сводная таблица запросов на обслуживание для DMA1 и DMA2 приведена соответственно в таблицах 3 и 4. Для каждого канала существует несколько источников запроса, которые объединяются с помощью логической операции «ИЛИ». Любой из каналов может быть разрешён или запрещён программно. Высшей приоритетностью обслуживания обладает первый канал.

Программирование DMA

Для использования блока DMA в программе необходимо выполнить его инициализацию и настройку параметров. Рассмотрим эти операции на примере подключения передатчика последовательного порта USART1_TX к каналу 4 DMA1 для передачи 64 байт. Начнём с настройки адреса источника и приёмника данных канала DMA.

Инициализация контроллера DMA, так же как и любого периферийного устройства, начинается с подачи на него тактовых импульсов. Делается это с помощью операции:

```
if ((RCC->AHBENR & RCC_AHBENR_
DMA1EN) != RCC_AHBENR_DMA1EN)
RCC->AHBENR |=RCC_AHBENR_DMA1EN;
```

Далее следует указать адрес регистра периферии, с которой будет работать канал DMA:

```
DMA1_Channel4->CPAR =
(uint32_t)&USART1->DR; // Адрес
регистра периферии
```

Теперь определимся с источником данных. Пусть это будет массив символов:

Листинг 1

```
unsigned char BuffTxd[32]; // Инициализация буфера передатчика
// Функция инициализации DMA1 для передачи в порт USART1
void USART1_TX_DMA1_Init (void)
{
 // Включить тактирование DMA1
 if ((RCC->AHBENR & RCC_AHBENR_DMA1EN) != RCC_AHBENR_DMA1EN)
 RCC->AHBENR |= RCC_AHBENR_DMA1EN;
 // Задать адрес источника и приемника и количество данных для обмена
 DMA1_Channel4->CPAR = (uint32_t)&USART1->DR; // Задать адрес регистра периферии
 DMA1_Channel4->CMAR = (uint32_t)&BuffTxd[0]; // Задать адрес буфера в памяти
 DMA1_Channel4->CNDTR = 32; // Задать количество данных для обмена
 // Настройка конфигурации
 DMA1_Channel4->CCR = 0; // Обнуление регистра конфигурации
 DMA1_Channel4->CCR &= ~DMA_CCR4_CIRC; // Отключить циклический режим
 DMA1_Channel4->CCR |= DMA_CCR4_DIR; // Задать направление чтения из памяти
 // Настроить работу с периферийным устройством
 DMA1_Channel4->CCR &= ~DMA_CCR4_PSIZE; // Задать размерность данных 8 бит
 DMA1_Channel4->CCR &= ~DMA_CCR4_PINC; // Запретить инкремент указателя
 // Настроить работу с памятью
 DMA1_Channel4->CCR &= ~DMA_CCR4_MSIZE; // Задать размерность данных 8 бит
 DMA1_Channel4->CCR |= DMA_CCR4_MINC; // Включить инкремент указателя
 USART1->CR3 |= USART_CR3_DMAT; // Разрешить передачу USART1 через DMA
}

// Функция проверки флага окончания обмена в канале
// Результат: 0 - обмен не завершен, 1 - обмен завершен
unsigned char GetStateDMAChannel4(void)
{
 if(DMA1->ISR & DMA_ISR_TCIF4) return 1; // Если флаг TCIF установлен - обмен завершен
 return 0; // Иначе - обмен продолжается
}

// Функция запуска обмена в канале
// Аргумент LengthBufer - количество данных для обмена
void StartDMAChannel4(unsigned int LengthBufer)
{
 DMA1_Channel4->CCR &= ~DMA_CCR4_EN; // Запретить работу канала
 DMA1_Channel4->CNDTR = LengthBufer; // Загрузить количество данных для обмена
 DMA1->IFCR |= DMA_IFCR_CTCIF4; // Обнулить флаг окончания обмена
 DMA1_Channel4->CCR |= DMA_CCR4_EN; // Разрешить работу канала
}
```

```
unsigned char BuffTxd[64];
```

Тогда адрес буфера в памяти можно задать так:

```
DMA1_Channel4->CMAR = (uint32_t)&BuffTxd[0]; // Адрес буфера в памяти
```

Далее укажем количество байт, которое следует передать:

```
DMA1_Channel4->CNDTR = 64; // Количество данных для передачи
```

На этом настройка адресов источника и приёмника канала DMA заканчивается и далее необходимо настроить режимы работы канала.

Листинг 2

```
unsigned char BuffTxd[32]; // Инициализация буфера передатчика
void main()
{
// Загрузка буфера
BuffTxd[0] = 'Д';
BuffTxd[1] = 'о';
BuffTxd[2] = 'б';
BuffTxd[3] = 'р';
BuffTxd[4] = 'ы';
BuffTxd[5] = 'й';
BuffTxd[6] = ' ';
BuffTxd[7] = 'д';
BuffTxd[8] = 'е';
BuffTxd[9] = 'н';
BuffTxd[10] = 'ь';
BuffTxd[11] = '!';
USART1_TX_DMA1_Init(); // Инициализация режима DMA1 для USART1
StartDMAChannel4(12); // Пуск передачи
while(GetStateDMAChannel4()==0) {}; // Ждать окончания передачи
StartDMAChannel4(12); // Повторный пуск передачи
while(GetStateDMAChannel4()==0) {}; // Ждать окончания передачи
}
```

Данную процедуру можно разделить на три части:

- настройка режима связи с периферийным устройством;
- настройка режима связи с памятью;
- настройка канала в целом.

Настройка режима связи с периферийным устройством подобна настройке режима связи с памятью. Необходимо задать размерность данных и активировать инкремент указателя.

Размерность данных может составлять 8, 16 или 32 бита. В рассматриваемом примере данные будут перемещаться из буфера памяти в регистр данных USART1_TX. Для этого необходимо будет поместить в один и тот же регистр данных USART1_TX первый байт из буфера, затем второй и так далее до конца буфера. Значит, инкрементация указателя на память необходима, а инкрементация указателя на периферийное устройство не требуется.

Настройки канала в целом включают в себя:

- задание режима цикличности в виде однократного или циклического обмена;
- задание направления чтения из памяти или периферии;
- задание уровня приоритета канала;
- разрешение при необходимости прерываний.

Данные настройки выполняются с помощью битов регистра управления канала CCR, описанных ранее. Запись данных в управляющий регистр можно выполнять побитно или в один приём 32-разрядным словом. Для наглядности рассмотрим пример побитного занесения данных.

Вначале выполним предварительную очистку управляющего регистра:

```
DMA1_Channel4->CCR=0; // Очистка регистра конфигурации четвертого канала
```

Далее зададим размер элемента данных размером 8 бит путём обнуления соответствующего поля:

```
DMA1_Channel4->CCR &= ~DMA_CCR4_MSIZE; // 8 бит
```

Для задания другого размера можно использовать следующие операции:

```
DMA1_Channel4->CCR |= DMA_CCR4_MSIZE_0; // 16 бит
DMA1_Channel4->CCR |= DMA_CCR4_MSIZE_1; // 32 бита
```

Аналогично задаём размер элемента данных для периферии:

```
DMA1_Channel4->CCR &= ~DMA_CCR4_PSIZE; // 8 бит
```

Для передачи всего массива данных из буфера активируем режим инкремента указателя в памяти:

```
DMA1_Channel4->CCR |= DMA_CCR4_PINC; // Инкремент указателя
```

Запретим инкремент указателя для периферии:

```
DMA1_Channel4->CCR &= ~DMA_CCR4_PINC;
```

В обычном режиме DMA останавливается после выполнения одного цикла обмена. Чтобы инициировать следующий цикл обмена, необходимо выполнить следующие действия: отключить канал DMA, перезагрузить регистр CNDTR, вновь включить канал DMA. Но можно изначально задать режим циклического обмена. Тогда по окончании передачи DMA автоматически начнёт новый цикл. Для этого необходимо выполнить следующие операции:

```
DMA1_Channel4->CCR |= DMA_CCR4_CIRC; // Включить циклический режим
DMA1_Channel4->CCR |= DMA_CCR4_DIR; // Задать направление передачи данных из памяти
DMA1_Channel4->CCR |= DMA_CCR4_EN; // Разрешить работу канала
```

Кроме этого необходимо разрешить работу периферии через DMA, воспользовавшись описанием периферийного устройства, которое нужно подключить к DMA. В нашем случае для подключения USART1 используем следующий код:

```
USART1->CR3 |= USART_CR3_DMAT; // Разрешить передачу USART1 через DMA
```

Узнать об окончании процедуры обмена через DMA помогут следующие флаги в регистре ISR: TEIF – ошибка обмена в канале, HTIF – передана половина буфера, TCIF – передан весь буфер и GIF – общий флаг прерывания в канале, который устанавливается, если установлен любой из флагов TEIF, HTIF или TCIF.

Для проверки соответствующего флага канала 4 необходимо выполнить следующие процедуры:

```
if (DMA1->ISR & DMA_ISR_TEIF4) {} // Проверить флаг TEIF ошибки обмена
if (DMA1->ISR & DMA_ISR_HTIF4) {} // Проверить флаг HTIF передачи половины буфера
if (DMA1->ISR & DMA_ISR_TCIF4) {} // Проверить флаг TCIF окончания обмена
```

Для обнуления установленного флага канала 4 существует регистр IFCR. Очистка соответствующего флага выполняется так:

```
DMA1->IFCR |= DMA_ISR_TEIF4; // Очистить флаг ошибки обмена TEIF
DMA1->IFCR |= DMA_ISR_HTIF4; // Очистить флаг передачи половины буфера HTIF
DMA1->IFCR |= DMA_ISR_TCIF4; // Очистить флаг окончания обмена TCIF
```

Примеры программ

В листинге 1 приведён пример программного кода набора функций настройки DMA для работы с USART1_TX, созданных на основе изложенного материала.

Приведённые функции используем в программе, которая позволит передать содержимое буфера в порт USART1 через DMA без участия процессора. Пример такой программы приведён в листинге 2. В качестве данных буфера будет загружена фраза «Добрый день!». Естественно, перед использованием программы необходимо будет выполнить инициализацию USART1.

Данная программа обладает определённым недостатком, связанным с необходимостью проверять флаг окончания передачи. Избавиться от этого недостатка поможет использование прерываний от канала DMA.

В качестве нового примера использования прерываний от DMA рассмотрим программу, приведённую в листинге 3.

В этой программе формируется буфер данных для передачи в порт USART2 и производится инициализация канала 6 DMA1 с разрешением прерываний. По окончании передачи произойдёт прерывание, которое вызовет обработчик DMA1_Channel6_IRQHandler. В этом обработчике обнуляется бит прерывания и отключается 6-й канал DMA1.

Подробнее познакомиться с блоком DMA поможет первоисточник [2].

Литература

1. www.st.com.
2. www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf.

Листинг 3

```
uint16_t Buffer[10] = {0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
0x38, 0x39, 0x3A};
// Обработчик прерывания DMA1
void DMA1_Channel6_IRQHandler(void)
{
DMA_ClearITPendingBit(DMA1_IT_TC6);
DMA_Cmd(DMA1_Channel6, DISABLE);
}
// Функция инициализации DMA1 для USART2
void init_DMA1_USART2()
{
RCC->APB2ENR |= RCC_APB2ENR_IOPDEN | RCC_APB2ENR_AFIOEN;
RCC->APB1ENR |= RCC_APB1ENR_USART2EN;
AFIO->MAPR |= AFIO_MAPR_USART2_REMAP;
GPIOD->CRL &= !GPIO_CRL_CNF5;
GPIOD->CRL |= GPIO_CRL_CNF5_1 | GPIO_CRL_MODE5_0 | GPIO_CRL_CNF6_0;
USART2->BRR = USART_BRR;
USART2->CR1 = USART_CR1_UE | USART_CR1_TE | USART_CR1_RE;
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
DMA_InitTypeDef DMA_InitStruct;
DMA_StructInit(&DMA_InitStruct);
DMA_InitStruct.DMA_DIR = DMA_DIR_PeripheralDST;
DMA_InitStruct.DMA_PeripheralBaseAddr = (uint32_t) &(USART2->DR);
DMA_InitStruct.DMA_MemoryBaseAddr = (uint32_t) Buffer;
DMA_InitStruct.DMA_BufferSize = 10;
DMA_InitStruct.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStruct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_InitStruct.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_InitStruct.DMA_Mode = DMA_Mode_Normal;
DMA_Init(DMA1_Channel6, &DMA_InitStruct);
USART_DMACmd(USART2, USART_DMAReq_Tx, ENABLE);
DMA_Cmd(DMA1_Channel6, ENABLE);
DMA_ITConfig(DMA1_Channel6, DMA_IT_TC, ENABLE);
NVIC_EnableIRQ(DMA1_Channel6_IRQn);
}
// Главный модуль программы
void main()
{
init_DMA1_USART2();
while(1);
}
```