Внешняя оверлейная память ADSP-218x

Что я должен делать, когда мне нужно больше памяти, чем есть в DSP?

ADSP-2181 содержит встроенную память программ 16К x 24 и память данных 16К x 16. Что делать, когда для приложения не хватает 16К памяти программ? Самым простым выходом является добавление памяти. ADSP-2181 имеет возможность доступа к двум страницам внешней оверлейной памяти, каждая из которых содержит по 8К для памяти программ и памяти данных. Теперь DSP содержит по-прежнему 16К памяти программ и памяти данных, но пользователь может обращаться к другим 16К памяти программ и 16К памяти данных. В этом техническом замечании будут рассмотрены основные вопросы, связанные с использованием оверлейной памяти. Обратитесь к руководству *PM/DM Overlay & I/O Memory Expansion Board for ADSP-2181 EZ-Kit Lite* для дополнительной информации.

Как я должен работать с внешней оверлейной памятью?

Перед использованием внешней оверлейной памяти необходимо вспомнить, как память организована. Организация памяти контролируется выводом ММАР и битом оверлея. Когда ММАР=0, 16К памяти программ делится на два сегмента, образуя страницу 0 для внутренней памяти и страницы 1 и 2 для внешнего доступа. Нижний сегмент памяти программ от адреса 0x0000 до 0x1FFF всегда является внутренней памятью. В верхнем сегменте памяти программ от адреса 0x2000 до 0x3FFF размещаются оверлейные страницы. Бит оверлея контролирует, какая оверлейная страница активна. Для памяти программ, регистр PMOVLAY контролирует, к какой оверлейной странице осуществляется доступ. Если PMOVLAY=0, пользователь работает с внутренней памятью. Если PMOVLAY=1 или 2, пользователь работает с внешней памятью. При PMOVLAY=1, внешняя память программ адресуется в пределах от 0x0000 до 0x1FFF. При PMOVLAY=2, внешняя память программ адресуется в пределах от 0x2000 до 0x3FFF. Это справедливо и для памяти данных. Используя бит DMOVLAY, пользователь может контролировать, какая страница оверлея активна. Также как и внутренняя память, внешняя память данных 16-разрядная, а память программ 24-разрядная. Разница между оверлеями памяти данных и памяти программ состоит в организации памяти. Вместо доступа к верхней части памяти от адреса 0x2000 до 03FFF, используются нижние 8К сегмента памяти данных.

Если все линии данных и линии адреса мультиплексированы, почему используются только адреса от 0x2000 до 0x3FFF?

При использовании оверлеев в один момент времени могут быть доступны только 8К сегменты. DSP использует только 13 младших разрядов, означающих линии адреса A12 – A0. Четырнадцатая линия определяет, какая оверлейная страница активна. Если используется страница 1, A13 устанавливается в 0, если страница 2, A13 устанавливается в 1. Это происходит как с памятью программ, так и с памятью данных.

Могу ли я использовать внешнюю память для запуска кода?

Внешняя память программ 24-разрядная, поэтому она может использоваться для хранения кода. Так как внешняя память разделяет линии адреса и данных, ответственность ложится на программиста, чтобы следить, какая оверлейная страница используется DSP. Может быть опасно менять оверлейный бит при выполнении кода на оверлейной странице. Если по любой причине, оверлейный бит меняется тогда, когда код работает на оверлейной странице, программа может продолжить выполнять код на другой оверлейной странице, который мог быть в середине цикла. Это вызовет ошибки. По причине безопасности, советуется, чтобы только программы, находящиеся во внутренней памяти были единственными, кто управляет оверлейным битом. После того, как оверлейная страница изменена, программа может перейти на любую программу, расположенную на оверлейной странице. Затем, после того, как программа завершится, она должна перейти снова во внутреннюю память.

Как символьные имена могут разделять один адрес?

В отличие от данных или кода, символы не располагаются в реальной ячейке памяти. Символы используются для ссылки адресов именами вместо чисел. Теперь, внешняя оверлейная память включена. Как было замечено, внешние оверлеи разделяют те же адреса, а также код может выполняться из внешнего оверлея. Из-за этого, другие программы должны разделять те же адреса. Так как компоновщик не распознает физическую оверлейную память, возникнут выходные ошибки, если другие модули будут скомпонованы по одинаковому адресу. Например, если приложение включает в себя три программы, *routine1*, *routine2*, и *routine3*, которые начинаются с адреса 0x2000, возникнут ошибки, поскольку компоновщик не может расположить три символа по тому же адресу. Для того чтобы решить эту проблему, новый компоновщик версии 2.3 допускает, чтобы разные модули скомпонованы по одному адресу. Файл *readme*, который сопровождает последнюю версию компоновщика, объясняет, как надо использовать компоновщик, чтобы иметь другие модули проживать на том же адресе.

Как я могу загрузить код во внешнюю память?

Новый компоновщик версии 6.x содержит дополнительные расширения, которые передают программы через BDMA из ППЗУ во внутреннюю память, а затем передают данные из внутренней во внешнюю память. Используя новый компоновщик, программы могут связываться по одному адресу, и в памяти данных создается оверлейная таблица. Таблица состоит из расположений программы в ППЗУ. Распределитель (*splitter*) инициализирует таблицу соответствующими данными, чтобы ядро могло узнать, где начинать BDMA передачу.