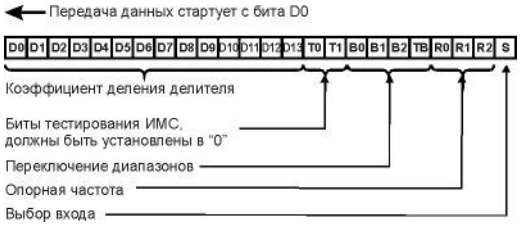
Итак, Вы решили использовать в своем приемнике (конструкции) схему цифрового синтезатора частоты. Расписывать тут подробно теорию синтезатора частот я не планировал. Вкратце - синтезатор состоит из трех узлов: формирователя опорной частоты (состоящего из генератора и делителя), ГУН-а (гетеродина) плюс управляемого делителя его выходной частоты, и схемы сравнения этих частот (поделенных - ГУН-а и опорной). Перестройка синтезатора по частоте производится изменением в делителе частоты ГУН-а его коэффициента деления. Схема сравнения частот при этом управляет ГУН-ом таким образом, чтобы его выходная частота (поделенная на коэффициент) совпала со сформированной опорной. Кому нужно подробнее, за сим, извиняюсь, - "к учебникам"...

В своем "Ресивере..." [1] в качестве основы синтезатора я использовал специализированную микросхему LM7001 фирмы Sanyo в типовой схеме включения. Выбор на нее пал по банальной причине доступности (на радиорынке г. Запорожья !!!) и низкой цены (около $1). Можете почитать datasheet [2] или [3] на микросхему (лично я настоятельно советую сделать это очень внимательно). Если же у Вас проблемы с "родным" английским языком, существует про нее информация и на русском языке (правда, более краткая): см. [4] "Справочный листок. – ж. Радио, 2003, №4 стр. 49,50". Его копия [5] "открыто жила" и в Интернете (я правда не знаю про "легитимность" и "долгосрочность" этой ссылки)...

Управляется синтезатор командами, передаваемыми по последовательному интерфейсу. И если в Вашей конструкции предполагается, что синтезатор всегда формирует одну-единственную частоту, то тут еще может можно обойтись тремя регистрами с параллельным входом и последовательным выходом, но в случае необходимости перестройки частоты, без процессора управления, увы, не обойтись. Смотрите сами - ниже приведена структура управляющего слова:




Как видим, нам необходимо "послать" в ИМС три байта (24 бита), содержимое которых полностью определяет ее работу. Может быть, я назвал эти биты несколько вольготно, но, ИМХО, так попонятней будет... Пройдемся по содержимому "управляющего слова" немного подробней (но !!! самое полное изложение все-равно - только в [2...5] )...

Коэффициент деления делителя - (биты D0-D13) Это как раз и есть делитель - значение, на которое делится измеряемая частота ГУН-а перед сравнением с опорной. Изменяя коэффициент, управляют частотой, на которую настроен синтезатор. При работе в диапазоне FM (бит S установлен в "1"; сигнал от гетеродина поступает на вход FMIN микросхемы) используются все 14 бит (с D0 по D13), в диапазоне AM (бит S сброшен в "0"; сигнал от гетеродина поступает на вход AMIN микросхемы) - только с D4 по D13.

Биты тестирования ИМС - (биты T0,T1) как следует из названия, используются в техпроцессе производства для тестирования произведенных микросхем. В "нормальной" эксплуатации всегда должны быть установлены в "ноль".

Переключение диапазонов - (биты B0-B2 и BT). Биты B0-B2 не имеют никакого отношения к частоте, на которую настроен синтезатор, а всего лишь определяют выходные уровни на выводах BO1-BO3 микросхемы. Выводы эти предназначены для управления внешними элементами коммутации, переключающими поддиапазоны в гетеродине и/или цепях тюнера (например, входные и нагрузочные контуры УВЧ). Если же биты B0-B2 установлены в "ноль", то в этом случае выходные уровни на выводах BO1-BO3 микросхемы определяются значением битов R0-R2. Бит BT управляет подачей сигнала т.н. измерительной (Time Base) частоты на вывод BO1 микросхемы. Когда бит BT сброшен в "ноль", состояние вывода BO1 определяется значением битов B0-B2 (или битов R0-R2), если же он установлен в "1", на выводе BO1 микросхемы присутствует меандр частотой 8 кГц.

Опорная частота - (биты R0-R2) позволяют выбрать опорную частоту поступающую на схему сравнения (из ряда: 1, 5, 9, 10, 25, 50 и 100 кГц), и, собственно, определяющую (равную) шаг перестройки

синтезатора по частоте. Кроме того, если биты B0-B2 установлены в "ноль", то значения битов R0-R2 определяют также и выходные уровни на выводах BO1-BO3 микросхемы. По сути, эти биты устанавливают коэффициент деления сигнала кварцевого генератора внутри ИМС, а т.н. "шаг" по сути - частота, подаваемая на схему сравнения в качестве опорной.

Выбор входа - (бит S) переключение диапазона: FM (бит равен "1"; при этом сигнал от гетеродина должен поступать на вход FMIN микросхемы) или AM (бит равен "0"; при этом сигнал от гетеродина должен поступать на вход AMIN микросхемы).

Еще одно необходимое отступление перед тем, как перейти к непосредственно процедурам управления. Про приемники. Точнее - про частоту, которую должен выдавать синтезатор частот. Итак, супергетеродинный приемник состоит из следующих узлов: УВЧ (может и отсутствовать), смеситель, гетеродин, УПЧ, детектор. На выходе детектора присутствует уже принятый сигнал, что с ним дальше делать - дело Ваше, тут я рассматривать не буду. Детектор тоже опустим. Начнем с того, что основное усиление супергетеродинного приемника происходит на промежуточной частоте и сосредоточено в УПЧ. Чтобы получить эту самую промежуточную частоту, в приемник установлены смеситель и гетеродин (роль которого в нашем случае выполняет тот самый синтезатор частот на LM7001). Промежуточная частота формируется на выходе смесителя и фактически является разностью частоты принимаемого сигнала и частоты гетеродина. Причем, вычитать можно как принимаемый сигнал из гетеродина (т.н. "верхняя настройка гетеродина"), так и гетеродин из принимаемого сигнала (т.н. "нижняя настройка гетеродина"). Неоднократно читал, что лучшим считается вариант с "верхней настройкой". Почему - всех доводов я уж и не помню, но там "что-то было про помехи"... Как следствие, выбрал я для себя "верхнюю настройку" гетеродина. При этом частота, на которой должен работать гетеродин будет выше частоты принимаемого сигнала на величину промежуточной частоты:

Fгет = Fсиг + Fпч

Теперь, собственно, про управление. Для начала определимся с тем, что и как мы будем делать. Собственно, у меня в "Ресивере..." тюнер работает только в "буржуйском" диапазоне FM. Других диапазонов нет, переключать их не нужно. Выходы BO1-BO3 микросхемы никуда не подключены, управлять ими тоже - никакой необходимости. Следовательно, все биты R0-R2, B0-B2 и BT "управляющего слова" в моем случае сброшены в "ноль". Сигнал гетеродина подается на вход FMIN микросхемы. И как следствие - бит S установлен в "единицу". Итого, вроде как в бинарном виде последний (третий) байт будет выглядеть так B'00000001'. Только вот еще один нюанс - если внимательно посмотреть на последовательность "управляющего слова", то видно, что данные передаются младшим битом вперед (это видно по полю делителя). Следовательно, чтоб не мудрить с двумя байтами делителя, разворачивая их справа-налево, проще всего один раз в уме (или на бумажке) развернуть третий байт. Получаем B'10000000' или 0x80 (hex). Именно такое значение мы и видим в процедуре передачи "управляющегшо слова" (сначала передается два байта делителя).

Теперь разберемся с делителем. Собственно, делитель (число) это частное частоты гетеродина и опорной частоты (которая фактически равна шагу настройки):

DIVISOR = Fгет / Fопорн

Частота гетеродина нам уже почти известна. Берем нижнюю частоту диапазона FM (88 МГц) и суммируем (настройка-то "верхняя") с промежуточной частотой (у меня в тюнере 10,7 МГц). Получаем 98,7 МГц. Теперь прибавим промежуточную к верхней частоте диапазона FM (108 МГц) - "в ответе" 118,7 МГц. То есть, для приема радиостанций в диапазоне частот от 88 до 108 МГц нужно, чтобы гетеродин перестраивался в диапазоне от 98,7 до 118,7 МГц. Итак, что делить, у нас есть. Будем определяться с тем на что делить. В принципе, для FM можно использовать шаг настройки как 50, так и 100 кГц (использовать 25 кГц, думаю не стоит - ну зачем такая точность?). С другой стороны, по формуле:

Nшагов = ((Fмакс - Fмин) / Fопорн) + 1

посчитаем число шагов необходимых для перестройки по всему диапазону FM (при шаге 50 кГц): ((108-88)/0,05)+1=401. Больше одного байта. Жалко! Если же использовать шаг настройки 100 кГц то получим всего ((108-88)/0,1)+1=201 шаг, это число прекрасно помещающается в один байт памяти. Вот это уже "ближе к народу". И, как бы оправдывая самого себя - а нужно ли настраиваться с точностью в 50 кГц? Порылся в памяти - все FM-станции, вещающие в Запорожье, рекламируя свою частоту, указывают только одну цифру после запятой. Следовательно, сетка частот с шагом в 100 кГц меня устраивает с головой. Вот и определились, на что делить - на 100 кГц. Подставив в приведенную выше формулу значения принимаемой и опорной частот (вторая, как мы помним, является одновременно и шагом настройки), получаем значение коэффициента деления. Проделав это дважды (для крайних частот FM-диапазона) мы получаем два коэффициента: минимальный - 987 (соответствует настройке применика на частоту 88,0 МГц), и максимальный - 1187 (принимаемая частота 108,0 МГц). Меняя этот коэффициент в пределах от 987 до 1187 и отсылая его в LM7001, мы сможем перестраивать приемник в диапазоне от 88 до 108 МГц. Что собственно, нам и требовалось...

Теперь еще одно обстоятельство. Раз уж мы будем использовать микропроцессор для управления, было бы очень неплохо хранить в его памяти несколько (а может и несколько десятков) станций. Что же в таком случае запоминать? Коэффициент деления (с его пределами от 987 до 1187) в один байт памяти не поместится, а использовать для хранения одной частоты аж две ячейки памяти - "жаба давит". Непосредственно частоту настройки (например, 101,8 МГц) тоже хранить как-то неудобно. Проще всего оказалось сохранять в памяти номер шага в сетке частот. Что это за номер такой? Да все просто - всего шагов в сетке у нас 201, 0-й шаг соответствует частоте 88,0 МГц, 1-й - 88,1 МГц и так далее. Можете посчитать сами (в уме, на бумажке, на калькуляторе) - 200-й шаг соответствует частоте 108,0 МГц. Так вот их "родимых", эти самые порядковые номера мы и будем сохранять.

Итак, мы определились - нам нужна одна переменная, назовем ее FREQVENCY. Назначим ей (а также другим, использованным в приводимых фрагментах программ переменным) адреса в ОЗУ микропроцессора. Управляя тюнером, нам нужно менять эту переменную в пределах от 0 до 200. Эту процедуру, я думаю, Вы напишите сами. Мы можем также в энергонезависимой памяти микропроцессора сохранить требуемое количество значенний этой переменной, представив их как память настроек на разные станции, и вызывая в случае необходимости. И это я оставлю на Вашей совести. Я опишу лишь три основных действия, которые необходимо выполнить, уже ПОСЛЕ того, как Вы изменили значение переменной FREQVENCY (или же считали ее из ЭНЗУ процессора), а именно: вычисление коэффициента для делителя ГУН-а, отправка "управляющего слова" на ИМС синтезатора и отображение принимаемой частоты (ну хоть на каком-нибудь дисплее). Собственно, приведенная ниже процедура FM_SET выполняет вызов именно трех необходимых нам подпрограмм:

```
;=============================================================================
; Full setting FM PLL Synthesizer
;=============================================================================

FM_SET
CALL FREQ_CONVERT ; Процедура преобразования частоты
CALL FREQ_SEND ; Отправка частоты в синтезатор
CALL SHOW_FM ; Показываем "FM", номер ячейки памяти и частоту настройки
RETURN ; возврат
```

Итак, мы ранее приняли, что у нас есть переменная FREQVENCY, значение которой изменяется в пределах от 0 до 200 и непосредственно определяет частоту, на которую наш синтезатор на LM7001 будет настраивать приемник. С другой стороны, нам необходимо получить значение делителя (DIVISOR), которое впоследствии отсылается в микросхему синтезатора, определяя частоту, на которую настраивается гетеродин приемника. Так как для диапазона FM и "верхней" настройки гетеродина требуемое значение делителя находится в пределах от 987 до 1187, в один байт памяти микропроцессора оно не поместится. Как следствие - результат подпроцедуры преобразования размещается в двух регистрах памяти - DIVISOR_LOW и DIVISOR_HIGH. Собственно говоря, преобразование переменной FREQVENCY в значение делителя - это простое прибавление к переменной FREQVENCY числа 987, которое по сути (для лучшей "доходчивости") является суммой числа 880 (пропорционального самой

нижней частоте диапазона FM - 88,0 МГц) и числа 107 (оно определяется значением промежуточной частоты приемника, в нашем случае равной 10,7 МГц). Именно это сложение и выполняет первая часть подпроцедуры FREQ_CONVERT (до той части, которая начинается фразой SECOND PART). Так как десятичное число 987 в шестнадцатеричном виде выглядит как 0x03DB(hex), а наша переменная FREQVENCY меняется в пределах одного байта, то процедура сложения максимально упрощена - сначала в регистр DIVISOR_HIGH записывается значение 0x03(hex), а потом производится сложение текущего значения переменной FREQVENCY и числа 0xDB(hex) - получаем значение DIVISOR_LOW. Потом проверяется регистр STATUS на предмет того, был ли перенос (бит "C" - CARRY), и если "да", то содержимое регистра DIVISOR_HIGH увеличивается еще на единицу.
HINT: Если Вам нужно настраивать приемник на другую полосу частот, то нужно вычислить новое значение числа прибавляемого к переменной FREQVENCY. Тут всё просто - сначала берем начальную (нижнюю) частоту принимаемого диапазона в МГц и умножаем ее на 10 (как помните, мы приняли, что шаг перестройки по диапазону равен 100 кГц) - получаем "X". Потом то же самое проделываем с промежуточной частотой (тоже умножаем на 10) - получаем "Y". Дальнейшее наше действие зависит от того, какую настройку гетеродина мы выберем - "нижнюю" или "верхнюю". В случае "нижней" настройки "Y" от "X" нужно отнять, в случае "верхней" прибавить. Полученное число и будет тем самым значением, которое нужно будет прибавлять к переменной FREQVENCY. И не забывайте, что для требующегося Вам диапазона может измениться число шагов настройки n=(Fв-Fн)/0,1. Соответственно, изменяя переменную FREQVENCY, проверяйте, чтобы она находилась в пределах 0 < FREQVENCY < n
...
Вторая часть процедуры FREQ_CONVERT выполняет преобразование переменной FREQVENCY в четырехзначное число (точнее, в четыре цифры, сохраняющиеся в регистрах FOURTH_DIGIT, THIRD_DIGIT, SECOND_DIGIT и FIRST_DIGIT), которое впоследствии непосредственно индицируется дисплеем устройства. Почему четыре цифры? Три из них служат для отображения целого значения, а четвертое показывает десятые доли МГц (например, 104,5). Данная процедура использует другую подпроцедуру - BCD_CONVERT, преобразующую бинарное значение переменной FREQVENCY в десятичное число.

```
;=================================================================================
; Frequency conversion from abstract values 0~200 to divisor and display.
; FIRST PART - convert freqvency register to divisor values for PLL
; synthesizer stored in two parts: DIVISOR_LOW and DIVISOR_HIGH.
; DIVISOR=(FREQVENCY+880+107)
;=================================================================================
FREQ_CONVERT
MOVLW 0x03
MOVWF DIVISOR_HIGH
MOVF FREQVENCY,W
ADDLW 0xDB
MOVWF DIVISOR_LOW
BTFSC STATUS,C
INCF DIVISOR_HIGH,F
;=================================================================================
; SECOND PART - converting FREQVENCY to four digit value for display
;=================================================================================
MOVFF FREQVENCY, TMP_VALUE
CALL BCD_CONVERT
CLRF FOURTH_DIGIT
MOVF DEC_ONES,W
ANDLW 0x0F
MOVWF FIRST_DIGIT
SWAPF DEC_ONES,W
ANDLW 0x0F
ADDLW 0xFE
BTFSC STATUS,C
GOTO CARRY_1
ADDLW D'10'
DECF HUNDREDS,F
CARRY_1
MOVWF SECOND_DIGIT
INCF HUNDREDS,W
ADDLW 0xFE
BTFSC STATUS,C
GOTO CARRY_2
ADDLW D'10'
DECF FOURTH_DIGIT,F
CARRY_2
MOVWF THIRD_DIGIT
INCF FOURTH_DIGIT,F
RETURN
```

Следующая (приведенная ниже) процедура передает в LM7001 24-х битное слово управления (три байта), определяющие ее режим и частоту настройки. Перед ее запуском необходимо, чтобы в регистрах DIVISOR_LOW и DIVISOR_HIGH уже находилось вычисленное и преобразованное значение коэффициента деления. Сначала устанавливаем "1" на выводе CE (в данном примере - SSP_CE). Этим самым мы разрешаем LM7001 принимать поступающие в нее данные. Затем загружаем во временный регистр PLL_BYTE_TEMP байт данных, который необходимо передать, и вызываем подпроцедуру SEND_BYTE. Так мы поступаем три раза (как помните, нам ведь необходимо в LM7001 передать три байта). В начале подпроцедуры SEND_BYTE в другой временный регистр (COUNT) мы записываем "8" (число бит в байте). После этого начинаем содержимое регистра PLL_BYTE_TEMP "сдвигать вправо" (если Вы еще не забыли - данные на LM7001 подаются младшим байтом вперед). После каждого сдвига проверяем бит CARRY (C) регистра STATUS и соответствующее значение ("0" или "1") выставляем на выводе данных (в данном примере - SSP_DATA). После этого формируем сигнал строба на соответствующем выводе микроконтроллера (в данном примере - SSP_CLK). Сначала вызываем подпроцедуру DELAY_PLL, формирующую короткую задержку (длительность задержки определяется

значением, записываемым в самом начале подпроцедуры DELAY_PLL во временный регистр COUNT_1). Затем ставим "1" на SSP_CLK, снова вызываем DELAY_PLL, после - ставим "0" на выводе SSP_CLK и еще раз вызываем DELAY_PLL. Таким образом мы получаем короткий строб на фоне импульса данных. "Изюминка" заключается в том, что и фронт импульса CLK и его спад происходят при устоявшихся данных на выводе DATA. Это позволяет процедуру SEND_BYTE использовать при выводе данных на устройства стробируемые как по фронту, так и по спаду синхроимпульса. Закончив передачу всех трех байт (24 бит), ставим "0" на выводе SSP_CE.
HINT: Если нужно использовать другой порт микроконтроллера (например, не C, а B), или же другие разряды порта, то нужно выполнить соответствующие изменения в приведенных ниже процедурах определения мнемонических имен.

```
;=============================================================================
; Передача управляющих команд в синтезатор частоты
;=============================================================================
FREQ_SEND
BSF SSP_CE
MOVFF DIVISOR_LOW, PLL_BYTE_TEMP
CALL SEND_BYTE
MOVFF DIVISOR_HIGH, PLL_BYTE_TEMP
CALL SEND_BYTE
MOVLW 0x80 ; Третий байт "управляющего слова"
MOVWF PLL_BYTE_TEMP
CALL SEND_BYTE
BCF SSP_CE
RETURN
SEND_BYTE
MOVLW 0x08
MOVWF COUNT
SEND_BYTE_1
RRCF PLL_BYTE_TEMP,F
BNC BIT_0
BSF SSP_DATA
BIT_0
CALL DELAY_PLL
BSF SSP_CLK
CALL DELAY_PLL
BCF SSP_CLK
CALL DELAY_PLL
BCF SSP_DATA
DECFSZ COUNT
GOTO SEND_BYTE_1
RETURN
DELAY_PLL
MOVLW 0x0F
MOVWF COUNT_1
DEL
DECFSZ COUNT_1
GOTO DEL
RETURN
```

Теперь (ниже) процедура вывода на дисплей - тут абсолютно ничего "военного". Естественно, перед ней должна быть выполнена процедура конвертации, которая в числе прочего сохраняет в регистрах FOURTH_DIGIT, THIRD_DIGIT, SECOND_DIGIT и FIRST_DIGIT числа для отображения частоты настройки на дисплее. Используемая ниже подпроцедура LCDPUTCHAR - это уже "слова из другой песни" (часть библиотеки по работе с ЖКИ-дисплеем). То есть, что мы делаем - берем символ "F" и посылаем на дисплей, потом - символ "M". Далее - берем переменную FM_NUMBER - в моей программе это собственно номер ячейки памяти, в которой хранится частота, на которую настроен приемник. Номера ячеек у меня с 1-й по 9-ю. Чтобы ЖКИ-дисплей (на чипе HD44780U или его "клонах") показал

"1" на него нужно послать 0x31 (hex), для "2" - 0x32 (hex), и т. д. Как следствие, перед выводом каждой цифры стоит дополнительная команда ADDLW 0x30, вводящая требуемый офсет. Исключение составляет вывод содержимого регистра FOURTH_DIGIT, в котором хранится значение сотен МГц - при его отображении сначала производится проверка того, что оно не равно нулю. Если же сотни МГц все-таки равны нулю, то вместо них выводится пробел. Больше никаких "фичей" в этой процедуре нет. Итог (на ЖКИ-дисплее) выглядит примерно так:

FM2 101,8 MHz

```
;=================================================================================
; Showing "FM", number of selected FM memory and value of Frequency
;=================================================================================
SHOW_FM
MOVLW 'F'
CALL LCDPUTCHAR
MOVLW 'M'
CALL LCDPUTCHAR
MOVF FM_NUMBER,W
ADDLW 0x30
CALL LCDPUTCHAR
MOVLW 0x20
CALL LCDPUTCHAR
MOVF FOURTH_DIGIT,W
BTFSC STATUS,Z
MOVLW D'239'
ADDLW 0x30
CALL LCDPUTCHAR
MOVF THIRD_DIGIT,W
ADDLW 0x30
CALL LCDPUTCHAR
MOVF SECOND_DIGIT,W
ADDLW 0x30
CALL LCDPUTCHAR
MOVLW ','
CALL LCDPUTCHAR
MOVF FIRST_DIGIT,W
ADDLW 0x030
CALL LCDPUTCHAR
MOVLW 0x20
CALL LCDPUTCHAR
MOVLW 'M'
CALL LCDPUTCHAR
MOVLW 'H'
CALL LCDPUTCHAR
MOVLW 'z'
CALL LCDPUTCHAR
RETURN
```

Ниже приведена подпроцедура преобразования бинарного значения, хранящегося в регистре TMP_VALUE в десятичное (например, для последующего вывода на дисплей). Я не писал эту подпроцедуру, а взял ее готовую в интернете. Перед ее запуском необходимо, чтобы в регистре TMP_VALUE уже находилось число, которое нужно преобразовать из бинарного в десятичное. Результат преобразования сохраняется в двух регистрах: HUNDREDS и DEC_ONES. В первом из них получаются "сотни" (0,1 или 2). Во втором - "десятки" и "единицы", причем, "десятки" находятся в старших 4-х битах регистра DEC_ONES, а "единицы" - в младших.

```
;=============================================================================
; Binary to decimal convert I don't know, how it work, I took it from Internet
;=============================================================================
BCD_CONVERT
CLRF HUNDREDS
SWAPF TMP_VALUE,W
ADDWF TMP_VALUE, W
ANDLW B'00001111'
BTFSC STATUS,DC
ADDLW 0X16
BTFSC STATUS,DC
ADDLW 0X06
ADDLW 0X06
BTFSS STATUS,DC
ADDLW -0X06
BTFSC TMP_VALUE,4
ADDLW 0X16-1+0X6
BTFSS STATUS,DC
ADDLW -0X06
BTFSC TMP_VALUE,5
ADDLW 0X30
BTFSC TMP_VALUE,6
ADDLW 0X60
BTFSC TMP_VALUE,7
ADDLW 0X20
ADDLW 0X60
RLCF HUNDREDS,F
BTFSS HUNDREDS,0
ADDLW -0X60
MOVWF DEC_ONES
BTFSC TMP_VALUE,7
INCF HUNDREDS,F
RETURN
```

Вообще-то, приведенный ниже фрагмент является описанием используемых переменных (назначением адресов этим переменным в ОЗУ микропроцессора) и размещается в самом начале листинга программы. По сути каждая строка назначает одной переменной адрес ОЗУ, в которой та будет храниться. Например, строка "FM_NUMBER EQU 0x010" обозначает, что переменная FM_NUMBER будет храниться в ячейке ОЗУ с адресом 0x010(hex). Первые две переменные - FM_NUMBER и FREQVENCY - определяют, какая выбрана ячейка памяти и на какую частоту должен быть настроен приемник.

```
FM_NUMBER EQU 0x010 ; номер выранной ячейки памяти FM
FREQVENCY EQU 0x011 ; Текущее значение частоты приемника
```

Следующие шесть переменных - результат преобразований - первые две посылаются в LM7001 в качестве делителя, а последующие четыре выводятся на дисплей, показывая, на какую частоту настроен приемник.

```
DIVISOR_HIGH EQU 0x012 ; Старший байт делителя частоты приемника
DIVISOR_LOW EQU 0x013 ; Младший байт делителя частоты приемника
FIRST_DIGIT EQU 0x014 ; Младший разряд при отображении частоты
SECOND_DIGIT EQU 0x015 ; ....
THIRD_DIGIT EQU 0x016 ; ....
FOURTH_DIGIT EQU 0x017 ; Старший разряд при отображении частоты
```

Еще семь переменных, выполнющих вспомогательные функции.

```
HUNDREDS EQU 0x018 ; Регистр "сотен" результата децимального преобразования
DEC_ONES EQU 0x019 ; Регистр "десятков/единиц" результата децимального преобразования
TMP_VALUE EQU 0x01A ; Временный регистр
PLL_BYTE_TEMP EQU 0x01B ; Временный регистр
COUNT EQU 0x01C ; Счетчик
COUNT_1 EQU 0x01D ; Счетчик
DELAY EQU 0x01E ; Счетчик
```

И напоследок, еще три строки, определяющие соответствие используемым в листинге мнемоническим именам конкретных битов в конкретных регистрах микропроцессора. Просто, удобнее один раз в начале программы выполнить такое присвоение имени, чем потом (в случае необходимости что-то изменить) рыскать по всему листингу. Например, Вам понадобилось вывод SSP_CLK подать не на порт С, разряд 0, а на порт А, разряд 3. Меняете в соответствующей строке имя порта, номер разряда и все.

```
#define SSP_CLK PORTC,0
#define SSP_DATA PORTC,1
#define SSP_CE PORTC,2
```

СПИСОК ЛИТЕРАТУРЫ


1. Харций Д. Ресивер с цифровой обработкой сигнала. “Радиохобби”, 2004, №1 с. 48.
2. Datasheet на микросхему LM7001 на сайте ф. Sanyo
3. Datasheet на микросхему LM7001, лежащий у меня на сайте
4. Справочный листок. - ж. Радио, 2003, №4 стр. 49,50
5. Копия "справочного листка" из ж. Радио