

SOFTWARE FOR POWER/SWR METER

Bill Kaune, W7IEQ

Microcontroller Overview

The purpose of this document is to provide a description of the software used in the power meter that I developed during the past few years and which is described in a QST article in the January, 2011 issue. To understand this document, you will need some basic familiarity with the PIC line of micro-controllers marketed by Microchip Inc and some familiarity with assembly language programming. If you have not already done so, I recommend that you download from Microchip's web site (www.microchip.com) and print a copy of the data sheet for the PIC16F87XA microcontroller. (The "data sheet" is actually 228 pages long). Reading this document can be tough going at times. I'll try to help by describing some of the tricks I learned to help me in understanding and remembering the contents of the data sheet.

I found it useful to make a second copy of pages 15-17, 19-21, and 160-165. Most of the questions that come up during the writing of a program are answered in these few pages.

I used the PIC16F876A for this project. This controller contains a 8-bit processor that is able to do addition, subtraction, and various logical, test, and data movement operations. The instruction set is fairly simple compared to most other processors that I am familiar with. There are only 35 instructions. By comparison the 8080 series of processors used in Windows (and now Macintosh) computers has many more instructions.

An interesting feature of this processor is that it contains separate memories for instructions and data. The PIC16F876A contains 8,192 flash memory locations for instructions. (Once a flash memory location is programmed, it retains its contents even when power is removed from the processor.) Each instruction is 14 bits wide. This processor also contains a set of 8-bit-wide file registers, described on page 17 of the manual. Some of these file registers are "special-purpose" registers used by the processor for various internal functions. Many of the special-purpose registers can be read and, in most cases, altered by the user. The remaining file registers are called general-purpose registers and are available to the user. In the PIC16F876A, there are 368 general-purpose file registers. The file registers are made from so-called dynamic access memory: They lose their contents whenever power is removed from the processor.

The PIC16F876A also contains 256 bytes (i.e., 8 bits wide) of electrically erasable programmable memory (EEPROM). These registers, which can be used to retain data even when the processor is turned off, are not used in my power/SWR meter program, but I have used them in other projects.

Each of the 8,192 instructions in the PIC16F876A instruction memory is given an address, starting at 0 and ending at 8,191. Since $8192 = 2^{13}$, the processor will need a total of 13 bits to encode all possible instruction addresses. The address of the next instruction to be executed is kept in an internal 13-bit register called the program counter register. Normally, instructions are executed in ascending order. However, two instructions (GOTO, CALL) contain address information that is used to determine the next instruction to be executed. Unfortunately, the instruction length of 14 bits is too short to contain both an operation code and a full 13-bit instruction address. Instead, these two instructions only contain the 11 least significant bits of an address. To complete the address, the instruction memory is divided into four pages. Page 0 extends from instruction address 0 to 2047, page 1 from 2048 to 4095, and so on. To fully specify an instruction location, an 11 bit instruction address has to be specified as well as the 2-bit page number. The page number is encoded using bits 3 and 4 of the PCLATH special purpose data register. Thus, if a program is executing in one page and you wish to go to an instruction in another page, or call a subroutine in another page, you must first set the appropriate bits in the PCLATH register before executing a GOTO or CALL instruction. Table 1 lists the PCLATH bit settings required to access various pages of instruction memory. For example, if it is desired to go to an instruction in page 3 from a location in page 0, it will be necessary to first set both bits 3 and 4 of PCLATH to 1 before executing the GOTO instruction.

The Power/SWR meter program being described here uses instruction located in all four pages of instruction memory.

A similar situation exists for the data file registers. Instructions that refer to the file registers have only room for 7 bits of an address, but in the 16F876A, 9 bits are required to fully specify a file register location. Thus, the file register memory is divided into four banks, numbered 0 through 3. The bank number is encoded by two bits (symbolically named RP0 and RP1) contained within the special-purpose STATUS register. The programmer has to make sure that these bits are set appropriately before executing any instruction that involves data stored in the file registers.

The Power/SWR meter program being described here uses data registers located both in banks 0 and 1. Table 2 shows how rp0 and rp1 must be set to access data in any of the four banks.

Table 1. Addressing for GOTO and CALL Instructions

Page	Range of addresses	PCLATH		Portion of address included in instruction
		Bit 4	Bit 3	
Page 0	0 to 2047 ¹	0	0	0 to 2047 ¹
Page 1	2048 to 4095 ²	0	1	0 to 2047 ¹
Page 2	4096 to 6143 ³	1	0	0 to 2047 ¹
Page 3	6144 to 8191 ⁴	1	1	0 to 2047 ¹

¹In hexadecimal, 0 to 7FF

²Hexadecimal 800 to FFF

³Hexadecimal 1000 to 17FF

⁴Hexadecimal 1800 to 1FFF

Table 2. File Register Addressing

Register Bank	Address Range	rp1	rp0	Address contained in instruction
0	0 to 127 ¹	0	0	0 – 127 ¹
1	128 to 255 ²	0	1	0 – 127 ¹
2	256 to 383 ³	1	0	0 – 127 ¹
3	384 - 511 ⁴	1	1	0 – 127 ¹

¹In hexadecimal, 0 to 7F

²In hexadecimal, 80 to FF

³In hexadecimal, 100 to 17F

⁴In hexadecimal, 180 to 1FF

What makes the PIC a micro-controller rather than just a microprocessor is that it incorporates various peripheral units in addition to its core numeric processor. The application being described here utilizes two of these peripheral units: A 10-bit analog-to-digital converter (ADC) and a pulse-width-modulated (PWM) output module. These will be described in more detail later.

Power/SWR Meter Hardware Overview

The hardware of the Power/SWR Meter is described in a QST article (January, 2011). In summary, RF power being generated by a transmitter is routed through a directional coupler on its way to an antenna tuner and an antenna (the load). The directional coupler contains passive circuits that sample the power flowing from the transmitter to the load (the so-called forward power) and, separately, the power flowing, after reflection, back from the load to the transmitter. Thus, the directional coupler has two outputs, one for the forward power sample and the other for the reflected power sample. Normally, the antenna and/or antenna tuner will be adjusted to minimize the reflected power flowing towards the transmitter.

The forward and reflected power samples are routed to two integrated circuits in the power meter whose outputs are voltages proportional to the logarithms of the rf input power. That is, the outputs are proportional to dBm (decibels relative to 1 milliwatt). The time constants in these two circuits are adjusted so that the outputs will follow the modulation envelopes of the input signals.

The dBm outputs are routed to two sample-and-hold integrated circuits that operate under the control of the 16F876A controller. The controller can send a signal to the two sample-and-hold units that will cause them to freeze (hold) their outputs at the input level present when the signal from the controller was received. In this way the forward and reflected power signals present at any particular time in the transmission line from the transmitter to the load can be held. The processor then reads in the forward signal and converts it (i.e., digitizes it) from an analog voltage to a corresponding 10-bit binary number. It requires 19.2 microseconds (μ s) to perform this conversion. It then reads the reflected power signal and converts it to a 10-bit number.

The analog-to-digital conversions are always done with respect to some voltage reference. An input voltage equal in amplitude to the reference voltage will yield a converted binary value of the maximum possible, in the case of a 10-bit converter 1023. The reference voltage used in my unit comes from a special integrated circuit that produces a very stable 2.50-volt output.

Once the forward and reflected powers are converted into internal 10-bit numbers, the processor computes the peak and average values of a large number of such samples and calculates the

standing-wave-ratio (SWR). The averaging time can be either short (about 1.6 seconds) or long (about 4.8 seconds) and is determined by the setting of the front panel “Averaging” switch shown in Figure 1. Depending on the setting of a second front-panel “Power” switch, the processor then calculates the peak and average forward powers, or the peak and average load (i.e., forward minus reflected) powers, present on the transmission line and displays these values using a two-line, 20 characters per line, liquid crystal display. In addition, every 0.1 second, the processor calculates the SWR and updates a front-panel analog meter, thus showing on an effectively continuous basis SWR. This is very useful for tuning purposes.

The remainder of this document will describe in more detail the software and how it accomplishes the tasks summarized above.

Power/SWR Meter Processor Circuit Connections

My article in QST (January, 2011) describes in detail the overall operation of the circuit in the Power/SWR Meter. Here, I will describe in more detail those parts of the circuit that directly connect to the PIC16F876A processor.

Figure 2 is the schematic of the meter. U5 is the PIC16F876A processor. Power (+5 V) is supplied to the processor through pin 20 and ground connection is through pins 8 and 19. Pin 1 is the so-called master clear input (MCLR). When power is first supplied to the processor, it will not start executing instructions until the signal on pin 1 goes high. It is important that pin 1 not go high until the source voltage to the processor has stabilized. R3 and C17 form a network whose effect is to prevent the signal on pin 1 going high for several tenths of second.

Pin 2 is the analog input for the forward channel power signal and pin 3 is the corresponding input for the reflected power channel signal. Pin 4 is configured is a digital output and provides the control signal for U3 and U4, the sample-and-hold chips for the forward and reflected power channels, respectively. Pin 5 is the input for the very stable +2.50 volts generated by U7 that is used as the reference against which analog-to-digital conversions are performed.

Pins 9 and 10 connect to a 20-MHz crystal that determines the clock frequency for the processor. The period of the clock is 50 ns (nanoseconds). In PIC 16-bit processors, four clock cycles are required for the execution of each instruction. Thus, each instruction requires 200 ns to execute.

Pins 11 and 12 are not used. Pin 13 provides a rectangular wave output (pulse height ≈ 5.0 V) whose frequency is fixed at about 4.9 kHz. The pulse width, however, can be varied in software from 0% (i.e., output is always 0 V) to 100% (output is always 5 V) in 256 steps. The signal from pin 13 is directed to the network R7 and C21 which act as a low-pass filter to extract the dc component, that is, the average of the pulse signal. This signal is buffered by U6 and drives the front panel analog meter M1 shown in Figure 1. R8 should be adjusted so that a signal with a 100% duty factor will result in a full-scale deflection of M1.

Pins 14 and 18 are configured as digital inputs and are used to sense the position of the two front panel switches (Figure 1). One switch (SW1) sets the averaging time as either short (about 1.6 s) or long (about 4.8 s). The other switch (SW2) determines whether the power meter displays the peak and average forward power, or the peak and average load power (i.e., forward – reflected power), using its LCD display.

Pins 15-17 are digital outputs that send control signals to the LCD display and pins 21-28 send data to the LCD display. The display I used features two lines of 20 characters each.

Software Overview

A complete listing of the processor program for my Power/SWR meter is included in the Appendix. Don't worry about the length. One half of the program is a series of lookup tables that will be described shortly, another quarter is a floating-point package that I obtained from the Microchip website, and the last quarter is the actual operating software.

When the power meter is turned on, the software starts at instruction address 0. Some initialization is performed, and the software then moves into its normal operation. The software proceeds to repetitively hold and acquire samples of the forward and reflected powers. With the parameters I currently use, it takes 1,111 samples during one "cycle". Each sample requires 90 μ s (microseconds) to acquire, so all 1,111 samples require 0.1 seconds. As they are acquired the software keeps track of the peak (i.e., maximum) forward power and the corresponding reflected power. As described earlier, the power signals that the processor directly measures are actually proportional to dBm. During each sample, the software converts the forward power from dBm to watts. In order to speed this process, the conversion is done using a lookup table. The lookup table for the forward power is held in page 2 of program memory. The software also sums all 1,111 values of the forward power, in watts, in preparation for calculating the average power

At the end of one cycle, the software checks to see if the new peak value exceeds the previous peak value then being displayed on the LCD display. If it does exceed the currently displayed value, the LCD display is updated to reflect the new peak forward (or load, depending on the setting of the front panel “Forward/Load” switch) power, and the SWR value being displayed is also updated, using the new peak forward power and corresponding reflected power for the SWR calculation. The new peak value will be held in the display until either a larger peak value is encountered in a subsequent cycle or a given number of cycles are executed without a larger peak value being encountered. In this latter event, after the given number of cycles is executed, the display is updated using the new peak value measured during the next cycle.

The SWR is calculated at the completion of each cycle, using the peak forward and corresponding reflected powers measured during that cycle. As noted above, this new value will be written to the display if the new peak forward power is larger than the forward power being displayed. However, it is always used to update the analog front panel meter display. Thus, this meter display is updated approximately every 0.1 seconds (actually the time to complete one cycle plus to overhead to calculate the SWR). The SWR is not displayed directly on the meter. Rather, the quantity $1 - 1/SWR$ is displayed, where a value of 1 corresponds to the full-scale position of the meter needle. Doing this makes the front panel meter a more effective tuning aid. If the SWR is 1, indicating a perfect match, the meter will read 0. If the SWR is 2, the meter will read 0.5, that is at half scale. If the SWR is 5, the meter reading will be 0.8, and so forth.

The procedure I use to update the peak forward (or load) power displayed on the LCD display may seem a little strange. I adopted this approach because I was concerned that updating the display after every cycle would result in a flickering and, thus, unreadable, displayed value of the peak power. The way I have it now programmed, the display will hold a peak power value for a maximum of 30 cycles (about 3 s) unless an even larger peak value is encountered before this period has expired. Once a larger peak value is measured, the display is updated and the 30-cycle waiting period is started again.

The *average* forward (or load) power is not updated after every cycle. Instead, it is updated after a specified number of cycles have been completed, a period which I call a “series”. The number of cycles in a series is controlled by the front-panel averaging switch. In the “Short” position, the average will be updated every 16 cycles (a little more than 1.6 s), whereas in the “Long” position, the average will be updated every 48 cycles. I generally operate with the “Short” setting.

Software Details

Development System

In this section, I will get into the details of the software. It will help if you refer to the program listing in the Appendix. The software was written in assembly language and was assembled using the free assembler available from Microchip (www.microchip.com). The program was debugged using Microchip's IDE development system, which is also free. Once the program was debugged, a 16F876A controller was programmed using Microchip's PicStart Plus Development Programmer. This programmer is not cheap. There are other less expensive programmers that might be used for this purpose.

When I use the Microchip assembler, I always set software flags so that numbers are interpreted as decimal, unless they are explicitly preceded by "0x" in which case they are interpreted as hexadecimal. Thus, the number 71 will be interpreted as decimal 71, whereas 0x71 is hexadecimal (= 113 decimal).

The assembler program understands a substantial number of so-called assembler directives. For example, if one wishes to set the bank number to some value (between 0 and 3), one can write the code directly to do this (see Table 1) or one can, instead, use the assembler directive

bank Address

where Address is an address located in the desired bank. When the program is actually assembled, the assembler will replace this directive with the required machine instructions.

Similarly, if one wants to set the page number to a desired value before executing a GOTO or CALL instruction, one can directly manipulate bits 3 and 4 (see Table 2) or one can instead include the directive

page Address

where Address is an address located in the desired page. For various reasons (which might not make much sense), I have mostly chosen not to use these directives but to, instead, write code that directly sets/clears bits 3 and 4 of PCLATH and bits rp0 and rp1 of the status register.

I will mention other assembler directives as they are used.

Simulator and Simulate Mode

Microchip's IDE development system includes a simulator that I used to develop and test most parts of the program. This simulator, however, is not very effective at testing the analog-to-digital converter, the pulse-width-modulated unit, the front-panel analog meter, and the liquid crystal display. (In fact, the program will hang if you attempt to run it in the simulator.) To deal with this problem, I wrote the program to enable the user to select a simulate mode, where several of the components of the program are bypassed. At the very beginning of the program (Appendix 1, page 1), the user can set a variable, "Simulate" to 0 or 1. Setting it 1, by replacing the line

```
Simulate equ 0
```

with

```
Simulate equ 1
```

enables simulate mode. (The assembler directive equ simply defines a symbolic variable—Simulate in this example—and set its value to the indicated number. Whenever the assembler encounters this symbolic variable, it will replace it with its defined numerical value.)

Following this are two constants, ForwardOrLoadWatts and LongOrShortAvgTime, that are used only in simulate mode.

Note that, in any line, text following a semicolon character is treated as a comment and is ignored by the assembler.

Next in the listing is the definition of the variable LowPowerThreshold. When the peak forward power measured during a cycle of 1,111 measurements results in a digitized 10-bit value less than LowPowerThreshold, the processor writes the word "Low" to the LCD display. This variable is currently set to a value corresponding to an input forward power of about 1 mW (milliwatt).

Calibrate Mode

A little later on the first page, a new constant, “Calibrate” is defined. If Calibrate = 0, the normal mode of the program is enabled. However, if Calibrate = 1, a different program is assembled. The calibrate program is used for calibrating my power/SWR meter. This program continually reads the forward and reflected power channels 1024 times, then computes the average of the string of digitized 10-bit values, and displays the resulting 10-bit averages on the front panel display. As discussed in my QST article (January, 2011), the response of the power meter to an input RF signal can be written

$$dBm = \alpha \times ADC + \beta ,$$

where dBm is the measured known power input to the power meter and ADC is the recorded digitized value of the power. By inputting known powers and recording the resulting ADC values, one can experimentally determine the slope, α and intercept, β , for the above equation and, thus, calibrate the channel.

In addition, the calibrate routine outputs a maximum signal to the front panel meter (i.e., the pulse train on pin 13 of U5 has a 100% duty cycle). The variable resistor R8 should be adjusted so that the meter reads full scale.

Constant Definitions

Next in the program listing is a series of constants. The first constant is MaxNumPeakWaits. As noted above, the program will display a peak value measured during its first cycle and then either display a new peak value measured during a subsequent cycle or, if a larger peak value is not encountered, continue to display the first peak value for a specified number of cycles before updating the display with the next measured peak value. The specified maximum number of cycles during which a peak value is held is defined by the value of MaxNumPeakWaits. I use 30 for this, which corresponds approximately to a maximum wait of about 3 s.

The next constant is MaxNumSWRMeterWaits. Assume that a valid forward and reflected power measurement is followed by a series of measurements with peak forward powers less than 1 mW, that is, too low to display and trust. The power meter will continue to display on its front-panel analog meter the SWR measured during the last valid cycle until a specified number of

cycles of low power readings is taken, at which point the SWR meter will be set to display 0. MaxNumSWRMeterWaits specifies the number of cycles.

The next constant is NumSamples. This sets the number of samples taken in one cycle. Currently it is set to 1,111 so that one cycle takes 0.1 s to complete. (Actually, NumSamples is set equal to $100,000/\text{SamplePeriod}$. SamplePeriod is the time in microseconds to make one measurement, currently set to 90. 0.1 s expressed in μs is 100,000. Dividing the latter by the former yields 1,111.11111111..., which rounds to 1,111.)

The following two constants, NumCyclesShort and NumCyclesLong, define the number of cycles in a “Short” or “Long” series. Remember that a front panel switch selects between Short and Long (Figure 1).

The next section of the listing in Appendix 1 describes the 16-bit floating point format that I use in part of the program. I will discuss this format a little later in this document.

Memory Allocations

Go to page 2 of Appendix 1. Part way down the page begins a section labeled “Constant and Memory Allocations”. The section provides the assignment of data memory addresses. The first statement is

```
cblock    0x20
```

This informs the assembler that this block of data memory starts with the address 0x20 (a hex number = 32 decimal). If you look at page 17 in the data sheet for the PIC16F876A, you will see that 0x20 is the first address of the general-purpose data registers located in bank 0. The following lines define the symbolic names for the registers in this block. Thus, 0x20 is given three names “exp”, “aexp”, and “aarg”. These registers, as well as all the registers through “bargb2” are used by the 24-bit floating point package supplied by Microchip. I will describe most of the following registers as I describe the routines that use them. Note that a symbolic name without any modifier causes the allocation of one 8-bit data register. However, if the name is followed immediately by a colon and number, the number of registers allocated to the name will be determined by the number. Thus, the appearance of a name like FP:2 will cause the allocation of two consecutive data registers. On the other hand, aexp:0 will not allocate any data registers to the name.

Definitions of the data registers in bank 0 continue until the “endc” statement that can be found on page 3 of the listing. Later on this page are the definitions of data registers that the program uses at the very top of bank 0 (starting with the address 0x7D), and data registers located in bank 1, starting with the address 0xA0.

The material following defines a large number of symbolic names used by the Microchip floating point library routines and by my program.

Macro Definitions

We next come to a series of macros. Macros define sections of code that are used at multiple points in the program. Rather than having to type out these instructions repeatedly, we define a macro. Anytime this macro appears in the stream of instructions, the assembler replaces it with the block of code specified by the macro definition.

The first macro is MoveBytes. This macro name is defined by the statement

```
MoveBytes    macro    AA, BB, nbytes
```

The code that defines this macro follows until the “endm” assembler directive is encountered. This macro moves a number of bytes, given by the argument “nbytes”, from the data registers starting at the address “AA” to the data registers starting at the address “BB”. This macro will properly move bytes from bank 0 to bank 0, bank 0 to bank 1, bank 1 to bank 0, or bank 1 to bank 1. For example, suppose that one wishes to move three contiguous bytes of data, with the starting address aexp, to a location with starting address t1. To accomplish this, one need only write the macro

```
MoveBytes    aexp,t1,3
```

and the appropriate machine instructions will automatically be generated by the assembler.

The next macro, StoreFPLiteral, stores a 24-bit literal floating point value into a given data register located in the current bank. (Details about 24-bit floating point values will be given shortly.) For example, suppose it is desired to store the number 1.0 into the registers beginning at

“aarg”. (The 24-bit floating point literal for 1.0 is 0x7F0000.) This can be done by writing the macro

```
StoreFPLiteral 0x7F0000, aarg
```

The next macro is named LCDDisplayText. The purpose of this macro is to write to the LCD display text and/or numbers that do not change during the course of the program’s operation. I am going to spend some time describing this macro because it is the first use of lookup tables, which are implemented in PIC processors in what was for me, at least, unique and interesting.

Suppose we want to be able to look up the various characters within a text string, for example, the string “abcd”. The way to do this using PIC processors is to first construct a series of four instructions as follows:

```
retlw    "a"
retlw    "b"
retlw    "c"
retlw    "d"
```

The instruction “retlw” means to return from a subroutine with a 8-bit literal value in the w register. (The w, or “working” register, is a special register in the PIC that is involved in most operations.) In this case, the first “retlw” will have encoded into it the ASCII code for the letter a, the second the code for “b”, and so on.

The second thing we need is a pointer to the letter we wish to retrieve. In this case, the pointer will have the values 0 (for “a”), 1 (“b”), 2 (“c”), and 3 (“d”).

Finally, we need a subroutine that is called with a value of the pointer and which, somehow, arranges to execute the “retlw” instruction that corresponds to the pointer value. The PIC 16-bit series of processors has what, in my experience, is a unique way to accomplish this. The PIC has an internal 13-bit register that keeps track of the next instruction that it is to be executed, the so-called program counter register. The lower 8 bits of this register are located in a special purpose file register named the PCL register. The upper 5 bits are held internally and are not directly accessible to the programmer. The interesting thing about the PIC 16-bit series is that a program can write directly to the PCL, in which case the next address to be executed is constructed from the 5 least significant bits in the PCLATH register and the new 8 bits in the PCL. An example of

this usage is my macro `LCD_DisplayText` that starts on page 5 of Appendix 1. To use this macro, the program must contain a statement with the following format:

```
LCD_DisplayText    TextString, UniqueAddress
```

`TextString` is the text to be printed (surrounded by quotation marks) and `UniqueAddress` is unique symbol that has the general format of an address. During the assembly of the program, the macro is replaced with the code following the macro definition starting on page 5. The code starts with 20 PIC instructions and then is followed by the assembler directive

```
dt    TextString,0
```

Suppose that `TextString = "abc"`. The assembler directive, `dt`, above results in the inclusion of four `"retlw"` instructions. The first includes the ASCII code for "a", the second code for "b", the third code for "c", and the fourth the literal value 0. This last value is used to signal the end of the lookup table.

Now take the instructions that precede the `"dt"` directive. The first instruction sets the file register bank to 0 and the next two instructions set the instruction memory page to 0. The next instruction sets the text pointer, named `LCDTextPtr`, to 0 so that it points to the first character in `TextString`. We then move into a section of code that loops through all characters in `TextString`. We first call the subroutine that immediately follows this section of code. For this call, we use a relative address, `$+7`. The symbol `$` means the current address. Thus, we are called a subroutine that begins 7 instructions further on in the code.

Upon entry, the subroutine temporarily stores the pointer and then loads the top 5 bits of the address of the lookup table (at `$+7`) into `PCLATH`. It then adds the pointer value to the bottom 8 bits of the address of the lookup table and increment `PCLATH` by 1 if there is a carry resulting from this addition. Then the bottom 8 bits are loaded into `PCL`, which causes the top 5 bits of the address counter to be taken from `PCLATH`. The next instruction executed is thus the appropriate entry in the lookup table. Since this instruction is `"retlw"`, control is returned to the instruction following the original subroutine call, but now with the appropriate ASCII code in the `w` register. The program then checks to see if the `w` register contains 0, which would signify that all characters in the lookup table have been retrieved. If `w` is not 0, the character in `w` is written to the LCD display, via a call to the subroutine `LCD_SendChar`, the pointer is incremented by 1, and control is sent to the beginning of the loop to process the next character.

The next macro we encounter is `ADC_Initialize`. As its name suggests, this macro simply initializes the analog-to-digital converter internal to the PIC16F876A processor. This processor contains two special purpose file registers (`ADCON0` and `ADCON1`) that control the ADC. In my application, the ADC is configured to use two input channels, channels 0 (denoted `AN0` in PIC literature) and 1 (`AN1`). In addition, the ADC is configured to utilize an external reference voltage. Externally, channel 0 is connected to pin 2 of the processor chip, channel 1 to pin 3, and the reference voltage to pin 5. The ADC module requires its own clock signal, which is normally obtained from the main processor clock through a division process. The period of the ADC clock must be at least 1.6 μ s, which means that since the period of the processor clock is 50 ns, the processor clock must be divided by (at least) 32. I use 32.

To initiate an ADC conversion, bit 2 of `ADCON0` must be set. Bit 2 will remain set until the conversion is complete, at which point the ADC module will clear point bit 2. The 10-bit result of the conversion is returned in two special purpose registers, `ADRESH` in bank 0 and `ADRESL` in bank 1. The ADC can be configured to return the 10-bit result either right-justified or left-justified in these registers. I chose to have it left-justified (i.e., most significant 8 bits in `ADRESH`, least significant 2 bits in bit positions 7 and 6 of `ADRESL`). I no longer remember why I made this choice, but were I writing this program again, I might select right justification.

The next macro, `ADC_StartRead`, initiates an ADC conversion. The following macro, `ADC_ReadFinished`, simply monitors bit 2 of `ADCON0` until it returns to 0 thereby indicating that the ADC conversion is finished. The following two macros, `ADC_SetCh0` and `ADC_SetCh1`, configure the ADC to read either channel 0 or 1, respectively.

Initialization Code

We now come to the actual instruction code for the processor. When power is first applied to the processor, and the Master Clear signal on pin 1 goes high, the processor starts by executing the instruction at location 0 in its instruction memory. The processor incorporates a fairly elaborate interrupt system (more details later), and any interrupt causes control to transfer to the instruction located at address 4. My program uses one type of interrupt, so there is an interrupt service routine located at address 4. Thus, when the processor initially starts, the first instruction is a `GOTO` instruction that simply jumps around the interrupt service routine to the address `StartUp`. This address is located on page 7 of the program listing in Appendix 1.

In StartUp, the first thing the processor does is wait for 1 second (actually 10 tenth-of-one-second waits). It then sets the appropriate states of the various input-output pins of the processor. These states are controlled by three special purpose registers, TRISA (i.e., Tri-State Port A), TRISB, and TRISC. A bit set to 1 in one of these registers configures the respective pin in the respective port as an input whereas 0 makes an output.

My program includes three subroutines that can be used to wait for various lengths of time. We have already mentioned WaitTenthSeconds. The number of periods that one desires to wait is entered in the w register. Any value from 1 (0.1 s wait) to 255 (25.5 s wait) can be entered into w. The next routine is WaitMilliseconds, which will wait from 1 ms ($w = 1$) to 255 ms ($w = 255$). Finally, WaitMicroSeconds will wait from 3 μ s ($w = 3$) to 255 μ s ($w = 255$). This routine will not act properly if $w = 1$ or $w = 2$. (These subroutines will only wait for these times on processors clocked at 20 MHz.)

Next the LCD display is initialized by calling the subroutine LCD_Init. I won't discuss the initialization procedure here. Suffice it to say that the procedure is rather complicated and, in some respects, mysterious. The display I used is documented in a manual². This manual, however, is of little use because it does not provide any information on programming the display. The display uses a Hitachi controller/driver integrated circuit, and the programming of this device is well documented in a Hitachi user manual³. I also purchased a book⁴ on programming PIC processors that I found helpful for programming this kind of display.

After the display is initialized, the text "W7IEQ Power Meter" is written to the first line of the display. (After the display is initialized, the "cursor" is set to character 1 of line 1.) The program next moves the cursor to character 1 of line 2 and writes out the text "Version 2". The program then waits for 3 seconds, enough time for the user to read the display.

The next step is to initialize the analog-to-digital converter module. This is done with the macro "ADC_Initialize" which has already been discussed. We next place the two sample-and-hold chips into sample mode by setting bit 2 of IO port A to 1. The pulse-width module (PWM) is next initialized by calling the subroutine "PWM_Initialize". The module is set to produce a steady train of pulses at a frequency of about 4.9 kHz. Initially, the pulse width is set to 0%, but it can be changed in software with 256 steps between 0 and 100%.

The next three steps in the program set up the interrupt system. (I will give details later.) These three instructions are:


```

bsf      status,rp0
bsf      0x7F & PIE1,TMR1IE
bsf      INTCON,PEIE

```

The two special-purpose registers, PIE1 (Peripheral Interrupt #1) and INTCON (Interrupt Control) that we wish to manipulate are located in bank 1 of the file registers. To reach bank 1, we must set the rp0 bit in status to 1, which the first instruction listed above does. The next instruction sets the bit TMR1IE (timer 1 interrupt enable) to 1.

Notice the odd construction 0x7F & PIE1 in the second instruction listed above. PIE1 is an address of a register located in bank 1 of the file registers. The Microchip assembler seems to always issue a warning whenever an address in a bank other than bank 0 is referenced in an instruction. Since, in my program, there are many such references, the list of warning messages the assembler produces is very lengthy. To avoid this, I precede all such addresses referenced in an instruction with the construction “0x7F & “. The symbol means logical bit-by-bit AND and has the effect of eliminating all but the least significant 7 bits of the address. But, all such 7 bit addresses are located in bank 0, so doing this avoids the warning message.

(I suspect there must be another way to avoid this problem. Perhaps the assembler has a switch that disables this particular warning message. However, I have looked and have never been able to find such a switch.)

24-Bit Floating Point Routines

This power meter is designed to respond to power levels from 1 mW to over 1,000 W. To handle such a range of powers, I early-on decided to use floating point arithmetic in most of the program. I searched the Microchip web site (specifically Application Notes) and found both 24-bit and 32-bit floating-point libraries. I chose to use the 24-bit floating-point library. The Microchip floating-point library is located in page 1 of instruction memory.

It is common to express decimal numbers (particularly very large and very small numbers) in power-of-ten format. For example, the speed of light, in meters per second, is 299,790,000 m/s. This would, most commonly, be written 2.9979×10^8 m/s.

A similar situation exists with binary arithmetic. Two formats are typically used, fixed point and floating point. In floating point, all binary numbers are written in the format

$$1.b_{-1}b_{-1}\cdots b_{-m}\times 2^n,$$

where m is the number of significant digits and n is an integer exponent. Note that this number starts with the digit 1. Because binary has only two bit values, 0 and 1, *all* numbers start with the digit 1 except for 0. For the 24-bit format used by the Microchip library, $m = 15$. Thus, for example the numbers 1.0, 3.14159, and -0.1 would be written in this binary floating-point format as

$$\begin{aligned} 1.0 &= 1.000\ 0000\ 0000\ 0000 \times 2^0 \\ 3.14159 &= 1.100\ 1001\ 0001\ 0000 \times 2^1 \\ -0.1 &= -1.100\ 1100\ 1100\ 1101 \times 2^{-4} \end{aligned}$$

The binary value is called the mantissa.

A 24-bit floating-point value consists of 24 bits (3 bytes) that encode this value. The first 8 bits contain the value $n + 127$, where n is the exponent (by adding 127 to the exponent, we avoid negative values) and n is restricted to the values $-126 \leq n \leq 128$. With this restriction, $1 \leq n + 127 \leq 255$. The value of 0 for the exponent byte is then reserved to signify that the overall number is 0. Since the leading bit of all nonzero mantissas is 1, we only include the 15 bits to the right of the decimal point. The 16th bit is used to indicate whether the number is positive or negative, with the value 1 indicating the number is negative. Thus the above three numbers, written in Microchip format, are

$$\begin{aligned} 1.0 &= 0111\ 1111\ 0000\ 0000\ 0000\ 0000 = 0x7F0000 \\ 3.14159 &= 1000\ 0000\ 0100\ 1001\ 0001\ 0000 = 0x804910 \\ -0.1 &= 0111\ 1011\ 1100\ 1100\ 1100\ 1101 = 0x7BCCCD \end{aligned}$$

The Microchip floating-point library contains subroutines to add (FPA24), subtract (FPS24), multiply (FPM24), and divide (FPD24) values. In all cases, the first argument must be placed in the data register `aarg`, `aarg+1`, `aarg+2` and the second in `barg`, `barg+1`, `barg+2`. The result of the operation is returned in `aarg`, `aarg+1`, and `aarg+2`. The library also includes routines that convert 16-bit binary integers into 24-bit floating-point values (FLO1624), 24-bit integers into 24-bit floating point values (FLO2424), and 24-bit floating point values into 16- or 24-bit integers

(INT2416 and INT2424, respectively). Finally, the library includes subroutines that determine whether the number in aarg is $<$ the number in barg (TALTB24), $\text{aarg} \leq \text{barg}$ (TALEB24), $\text{aarg} > \text{barg}$ (TAGTB24), $\text{aarg} \geq \text{barg}$ (TAGEB24), $\text{aarg} = \text{barg}$ (TAEQB24), and $\text{aarg} \neq \text{barg}$ (TANEB24). The entire Microchip library occupies page 1 of instruction memory. In addition to the Microchip routines, I included one additional routine that I wrote named AargTimes10 that multiplies the value in aarg by 10. This routine uses the fact that $10 * \text{aarg} = 2 * \text{aarg} + 2^3 * \text{aarg}$ to more quickly do this multiplication.

There were many times during the development and testing of this program where I needed to convert decimal numbers into the Microchip 24-bit floating-point format, or visa versa. To simplify this, I wrote a program for my programmable hand calculator (a HP 48GX calculator). Since this calculator has built-in capability to convert decimal integer numbers to binary integer numbers, or the inverse, it was relatively simple to write this program. The program consisted of the following steps.

1. If $x = 0$, set the floating point value to 0x000000 and quit.
2. If $x < 0$, make it positive and continue with the following steps.
3. Let x be the number to be converted. If $x < 1$, multiply repetitively by 2 until $x \geq 1$. Keep track of the number of required multiplications. The negative of this value becomes the exponent.
4. If $x \geq 2$, divide repetitively by 2 until $x < 2$. Keep track of the number of required divisions. This value becomes the exponent.
5. Add 127 to the exponent deduced in Step 3 or Step 4 to obtain the offset exponent. Convert the resulting value to an 8-bit binary value. This is the first byte of the floating-point value.
6. Multiply the remaining value of x by 2^{15} , round it to the near integer value, and convert the resulting integer to a 16-bit binary value. (When the fractional value is exactly 0.5, round to the nearest even value. For example, if multiplying x by 2^{15} yields 32768.5, round to 32768. But if the resulting value is 32769.5, round to 32770.)
7. If the original value of x was positive, change the 16th (i.e., most significant bit) to 0. If x was negative, leave the 16th bit = 1. The resulting 16-bit value comprises the second and third bytes of the floating-point value.

More Initialization

Beginning on page 8 of the program listing in Appendix 1, the Microchip floating-point package is initialized by setting certain flags. Following this, the program proceeds to calculate two normalization constants, `FPIInvTotalSamplesShort` and `FPIInvTotalSamplesLong`. Earlier, I explained that the power meter program computes the average forward power during a series of samples. A series consists of a specified number of cycles, `NumCyclesShort` for the short averaging time (selected from the front panel “Averaging” switch) and `NumCyclesLong` for the long averaging time. Furthermore, each cycle consists of `NumSamples` measurements of the forward power. Thus, to compute the average, we must sum all the forward power samples and then divide by `NumSamples` multiplied by `NumCyclesShort` for short averaging or `NumSamples` times `NumCyclesLong` for long averaging. This initialization simply calculates the quantities $1/(\text{NumSamples} * \text{NumCyclesShort})$ and $1/(\text{NumSamples} * \text{NumCyclesLong})$ for later use by the averaging routine.

Following along in the listing, the routine first loads the 16-bit integer value of `NumSamples` into `aarg` and `aarg+1` and converts it to a floating-point value. Note that when I call a subroutine in page 1, I mask the address to keep only the least significant 11 bits. (This masking is done by performing a bit-wise logical and between the subroutine address and the number `0x7FF`.) I do this to prevent the assembler issuing messages warning that the page number has to be set correctly. I do set the page number to page 1 via the following instruction:

```
bsf    PCLATH,psel0
```

The resulting floating-point value of `NumSamples` is saved in `t1` and is also placed in `barg`. Next, the `NumCyclesShort` is loaded into `aarg+1` and `aarg` is cleared. (`NumCyclesShort` must be less than 256.) This value is then converted to floating point, and then multiplied by `barg`, that is, by `NumSamples`. The result is transferred to `barg`, and `aarg` is then loaded with the floating point value for 1.0, namely `0x7F0000`. `Aarg` is then divided by `barg`, yielding finally $1/(\text{NumSamples} * \text{NumCyclesShort})$. This result is saved in the 24-bit register `FPIInvTotalSamplesShort`.

The next segment of code calculates and saves $1/(\text{NumSamples} * \text{NumCyclesLong})$.

The next step in the initialization is to clear (i.e., set to 0) a data register named `ModeStatusFlags`. The various bits in this 8-bit register are used to keep track of certain conditions, as defined on page 2 of the listing. The conditions are covered are:

Bit 0 = 0 (or 1) means the peak forward power (in watts) measured during a cycle is greater than or equal to (or is less than) the low power threshold, defined by the constant `LowPowerThreshold`.

Bit 2 = 0 (or 1) means that the LCD display is showing the peak and average load (or forward) powers.

Bit 4 = 0 (or 1) means the average time switch in the in short (or long) position.

Bit 5 = 0 (or 1) means peak and average powers are being displayed with units of watts (or milliwatts)

The program next initializes the 16-bit register `bForwardPeak` to 0, the 8-bit register `PeakWaitCounter` to the value of the constant `MaxNumPeakWaits`, and the 8-bit register `SWRMeterWaitCounter` to the constant `MaxNumSWRMeterWaits`. The functions of these registers will be discussed later. Finally, the two-line LCD display is cleared and the text “SWR” is written near the right edge of the first line.

We now come to the line with the address `StartSeries`. (Any symbol that starts in column 0 of the listing is interpreted as an address by the assembler.) This point in the program will be entered whenever a new series of cycles is started. At this point, the program first sets bits to insure that page 0 of program memory and bank 0 of data memory are set. It then determines the setting of the front-panel Forward/Load Power switch (Figure 1). This switch is wired to bit 3 of input/output port C. Placing the switch in the “Forward” position will cause the level on this input to be high (i.e., 5 V) as shown in Figure 2.

The program then continues by further initializing the display by writing additional text that does not change during a measurement series. The program then determines the setting of the front-panel averaging time switch (Figure 1). This switch, wired to bit 7 of input/output port C, determines whether the averaging time is short or long. It sets or clears the appropriate bit in `ModeStatusFlags` and places the appropriate number of cycles per series into the 8-bit register `CycleCounter`. It then clears the 32-bit accumulator `Fpaccum` that is used to accumulate the running sum of measured forward powers. At the end of a cycle, the value in this accumulator will be divided by the total number of measurements in the cycle to determine the average The

program loads the 16-bit register SampleCounter with the number of measurements to be taken in the cycle and sets/clears one bit in the 8-bit register Flags. It then clears the 16-bit register bForward and calls the subroutine ISR to start the cycle.

During a cycle, a substantial number of measurements are taken. With the current settings in the program, 1,111 measurements are taken during one cycle, and measurements are repeated at 90 μ s intervals. This timing is set using the interrupt system and an internal timer in the processor. Thus, let me digress briefly to discuss the interrupt system.

Interrupt System

Together with many types of processors, PIC processors have the capability of responding immediately to certain external and/or internal conditions. When such a condition occurs, the processor is interrupted, the instruction being executed is finished, the location of the next instruction to be executed is saved, and the processor's execution stream is diverted to an "interrupt service routine (ISR)" that starts at address 4 in program memory. Normally, this routine will perform actions to "service" the interrupt condition and will then return execution to the normal instruction stream using the memory location saved earlier. Let me now explain how this capability is used in my program.

The most basic activity of the program is to repetitively sample the forward and reflected powers between a transmitter and a load. As noted earlier, a sample is taken every 90 μ s. This timing is implemented by having a clock (timer) internal to the processor generate interrupt signals every 90 μ s to initiate a sample. To do this, I used the processor's internal 16-bit timer (called TIMER1 in the PIC16F876A data sheet). The contents of this timer are incremented by 1 every instruction cycle. When it reaches its maximum value (0xFFFF) and the next increment arrives, the counter's contents overflow to 0 and an interrupt is generated. The processor will then stop whatever it is doing, save its current location in instruction memory (by placing the address of the next instruction to be executed at the top of an internal stack), and then transfer to location 4 in instruction memory. The program has an interrupt service routine (ISR) starting at location 4 that saves the contents of the w and status registers, initializes TIMER1, takes samples of the forward and reflected power, and processes them.

Now to details of the ISR, whose listing starts on page 6 of Appendix 1. The first task is to save the w and status registers. This must be done in such a way that none of the status bits are changed. I followed a method described in the PIC16F876 data sheet. After that, bit 2 of Port A

is cleared, which causes the two sample-and-hold chips to go into hold mode. A bit in the register `Flags` is then cleared. (Note that `Flags` is located in bank 1 of the data file registers, so bit `rp0` of status must be set before it is accessed.)

The next step is to turn the `TIMER1` counter off. We then initialize this counter and turn it back on. The 16-bit counter will overflow when it reaches the count $2^{16} = 65,536$. The counter is incremented 4 times every microsecond, so to set it to overflow every `SamplePeriod` (in μ s, currently set to 90), we must place the following count in the register: $65,536 - 4 * \text{SamplePeriod} + \text{Deadtime}$, where `Deadtime` is the dead time in `TIMER1` counts between the time the ISR was initially entered and the time that `TIMER1` is turned back on after being initialized. In this case, `Deadtime` = 16 counts.

Next, the `BusyFlag` bit is set in `Flags`, indicating that the routine is currently processing a measurement. We then wait 16 additional microseconds to allow time for the Sample-and-Hold chips to stabilize their held data. We then read in the contents of channel 0 of the ADC. Channel 0 is connected to the forward power channel. The ADC is immediately switched to read channel 1, but the actual read is delayed until the channel 1 electronics stabilize around the reflected power channel data. The forward power ADC data are transferred to the 16-bit register `bForwardNew`, and the new value is compared to the previous peak value, held in the 16-bit register `bForward`. If the new value is greater, it will replace the value in `bForward`.

Comparison of `bForwardNew` and `bForward` is done using the subtract operation. I found this procedure quite confusing at first, particularly the function of the carry bit, until I discovered a way of thinking about it that made the interpretation easy. To illustrate this, suppose we have two 8-bit quantities, A and B , in memory that we wish to compare. We can do this by subtracting B from A . Subtraction in the PIC processors is performed using 2's complement arithmetic. To subtract B from A , the 2's complement of B is first formed. I will denote the 2's complement as $-B$. To do this, every bit in B is interchanged, that is, 1's become 0's and 0's become 1's. It is easy to see that this is exactly equivalent to subtracting B from 255, that is, $255 - B$. Then, 1 is added to the result. Thus,

$$-B = 256 - B$$

Once the 2's complement is formed, subtraction of B from A is accomplished by simple addition, that is

$$A - B = A + (-B) = A + (256 - B) = 256 + (A - B)$$

If the result of the addition is greater than 255, the carry bit will be set. On the other hand, if the result is less than or equal to 255, the carry bit will be cleared. Thus, we arrive at the rule:

If $A \geq B$, the carry bit = 1

If $A < B$, the carry bit = 0

Now go back to the program listing (p. 7). Remember that the 10-bit result of an ADC conversion is left justified. Thus, the most significant 8 bits of the result are in bForwardNew and the least significant 2-bits are the left-most bits in bForwardNew+1. The first step is to compare the most significant 8 bits of the bForward and bForwardNew. If bForwardNew > bForward, the program jumps to the address ISR_012 where bForward is replaced by bForwardNew. If on the other hand, if bForward = bForwardNew, the bottom two bits are compared.

The program next reaches the instruction location ISR_019, where the subroutine GetForwardWatts is called. Remember that the signal from the forward channel is a voltage proportional to logarithmic power in units of dBm. The purpose of GetForwardWatts is to convert from dBm to watts. I use lookup tables to accomplish this conversion as rapidly as possible.

The program occupies much of page 0 of instruction memory and the Microchip floating-point package occupies much of page 1. Thus, pages 2 and 3 were available for the lookup tables, a total of 4096 values. A 10-bit ADC can output 1024 values. Thus, the forward and reflected power channels have 2048 possible inputs, which means that, at most, there are two instruction locations available to table an ADC value. Unfortunately, a 24-bit floating-point value requires three locations to encode a value. Thus, I developed a 16-bit floating point format that I used for this purpose. The next section describes this format.

16-bit Floating-Point Format

Since this format will be used to convert 10-bit binary integers, representing forward and reflected power in units of dBm, to 16-bit floating point values equal to the corresponding power in watts, we only need 10 bits of mantissa to retain accuracy. Furthermore, all of these values will be positive, so there is no need to have a sign bit. Finally, the range of values is limited, from about 1 μ W (microwatt) to several thousand watts. Thus, this format consists of a 6-bit exponent, occupying the most significant 6 bits of the first byte, and a 10-bit “reduced”

mantissa. The mantissa is reduced by not including the leading 1 bit, which is present in all values except for 0. The exponent is offset by adding 27 to avoid negative values. Finally, an exponent of 0 is used to indicate that the number is 0.

For example, consider the number 1, which can be written in binary as 1.0000000000×2^0 . The offset exponent is $27 + 0 = 27 = 011011$. The mantissa is just 0000000000 since the leading 1 is suppressed. Thus, the 16-bit floating-point value is 0110 1100 0000 0000 = 0x6C00.

As another example, take 439. In binary, this is $1.1011\ 0111\ 00 \times 2^8$. Thus, the exponent is $27 + 8 = 35 = 100011$ and the mantissa is 10 1101 1100, so the floating-point value is $1000\ 1110\ 1101\ 1100 = 0x8EDC$.

The largest and smallest numbers that be encoded in this format are approximately 1.37×10^{11} and 1.49×10^{-8} , respectively.

Interrupt Service Routine (Continued)

The program continues at location ISR_019 by calling the subroutine GetForwardWatts to obtain the forward power in watts. This subroutine uses the Microchip standard method to implement lookup tables (see the discussion of lookup tables in the Macro section earlier in this report). The lookup tables for the forward power conversion occupy all of page 2 of memory and are organized into eight 256-word subtables. The first four subtables contain the right-hand bytes (low bytes) of the forward floating point power, and the next four subtables contain the left-hand bytes (high bytes). The least-significant 2 bits of the ADC result are used as an index into the four low-byte and the four high-byte subtables, and the most significant 8 bits are used as an index into the selected 256-word subtables. The result of this conversion is returned in the 16-bit data register FP.

After the lookup is complete, the processor initiates the actual ADC conversion of the reflected power signal. While this conversion is underway, the processor calls the subroutine SumP16, located in page 1 of instruction memory. This subroutine adds the forward power, in watts, to a running sum located in FPaccum. The program then checks to see if the ADC read of the reflected power is finished. (If it is not finished, the macro will wait until it is finished.) Once the read is finished, the processor sets the ADC to channel 0, checks to see if a new peak forward

power was measured and, if so, places the corresponding reflected power in the 16-bit data register bReflected.

Next, starting at the instruction following the address ISR_021, the 16-bit register, SampleCounter, is decremented by 1 and tested to see if it is 0. If it is 0, a bit named SampleDoneFlag is set in the bank 1 data register Flags. Then, at ISR_040, the status and w registers are restored to their state before the interrupt occurred and program control is returned to the program location just prior to the interrupt. This return is accomplished using the retfie instruction, which returns to the appropriate location and also arms the interrupt system so that it will respond to future interrupts.

Routine at the End of a Cycle

The code immediately following the “call ISR” instruction on page 10 of Appendix 1 simply repetitively clears the bit BusyFlag in Flags and checks the SampleDoneFlag in Flags. When it finds this flag set, indicating that all samples for the current cycle have been taken, the program disarms the interrupt system with the instruction

```
bcf    INTCON,GIE
```

and proceeds to process the results obtained during the last cycle.

The first step in the process is to determine whether the peak forward power measured during the last cycle, and held in the 16-bit register bForward, is larger than the current peak power being displayed on the front panel LCD display, and held in the 16-bit register bForwardPeak. If the latest peak is not larger, control passes to location hh003 in the program. The register PeakWaitCounter is decremented by 1 and checked to see if it is 0. If it is not yet 0, nothing more is done, i.e., the previous peak power will continue to be displayed on the front panel LCD display. If, however, PeakWaitCounter has been decremented to the point that it is 0, or the forward peak measurement during the latest cycle is larger, the program will jump to the location hh001 where the value bForwardPeak will be replaced by the new peak value in bForward. This value is then converted to watts, using the subroutine GetForwardWatts, the resulting value is converted to a 24-bit floating-point value via the subroutine P16ToFP24, and saved in PeakFowardWatts. (It is necessary to convert to a 24-bit value so that the Microchip 24-bit floating-point library can be used for subsequent analyses.) Then, the corresponding reflected power will be converted to 16-bit floating-point format and then to 24-bit floating-

point format. Then, the subroutine CalculateSWRandSetMeter is called to determine the SWR from the 24-bit floating-point values of forward and reflected power and set the front panel meter to reflect this new value. Next, the new SWR and peak power values are written to the LCD display using the routine DisplayPeakPowerAndSWR. Finally, the 8-bit counter PeakWaitCounter is initialized to the value MaxNumPeakWaits.

If a new peak forward power was not found during the last cycle, the peak forward and corresponding reflected powers that were measured during this cycle are used to update the front-panel analog SWR meter. This code starts at location hh002.

The program next arrives at the location c002 (p. 12) where the CycleCounter is decremented and checked to see if it is 0. If it is not 0, control is passed to the location NextCycle where processing for a new cycle begins. If, on the other hand, CycleCounter = 0, the series of cycles is complete and the program moves to the processing that follows the completion of a series.

Routine Following Completion of a Series

The only processing that occurs at the end of a series involves the calculation and displaying of the average forward (or load) power. First, the three most significant bytes of the 32-bit floating point accumulator are moved to the register FP, converted to 24-bit floating-point Microchip format, and are multiplied by the inverse of the total number of measurements, contained in the data registers FPIInvTotalSamplesShort and FPIInvTotalSamplesLong for short and long averaging times, respectively. The resulting average forward (or load) power is written to the LCD display using the subroutine DisplayAvgPower. Then, the program transfers control to the instruction location StartSeries, which starts the next measurement series.

Calibrate Program

The next part of the program, starting at the bottom of page 12 of the Appendix, is assembled only if the constant Calibrate is given the value 1 at the beginning of the program. This function of this program is to write to the LCD display the average of 1024 ADC measurements of the signals entering the forward and reflected power input pins of the processor

In the remainder of this document, I will give notes about some of the subroutines used by my SWR/Power Meter program.

Subroutine CalculateSWRandSetMeter

Standing wave ratio (SWR) is defined as

$$\text{SWR} = V_{\max} / V_{\min} ,$$

where V_{\max} and V_{\min} are maximum and minimum voltages, respectively, present on the transmission line connecting the transmitter to load. Let P_F and P_R be the forward and reflected powers on the transmission line, respectively. The voltages, V_F and V_R , of the forward and reflected waves, respectively, are related to P_F and P_R as follows:

$$V_F = \sqrt{P_F Z_0}$$

$$V_R = \sqrt{P_R Z_0}$$

where Z_0 is the characteristic impedance of the transmission line. The maximum voltage, V_{\max} , will occur where the two waves are in phase, in which case $V_{\max} = V_F + V_R$. Similarly, the minimum voltages on the line will occur where the forward and reflected waves are out of phase, so $V_{\min} = V_F - V_R$. Thus,

$$\text{SWR} = \frac{V_F + V_R}{V_F - V_R} = \frac{\sqrt{P_F Z_0} + \sqrt{P_R Z_0}}{\sqrt{P_F Z_0} - \sqrt{P_R Z_0}}$$

$$= \frac{\sqrt{P_F} + \sqrt{P_R}}{\sqrt{P_F} - \sqrt{P_R}} = \frac{1 + \sqrt{P_R/P_F}}{1 - \sqrt{P_R/P_F}} .$$

The subroutine, however, uses a different form of this formula to reduce the number of calculations. This modification is

$$\text{SWR} = \frac{-2}{\sqrt{P_R/P_F} - 1} - 1 .$$

To calculate the square root, I use a subroutine sqrt that I wrote. This subroutine was written to emphasize speed over accuracy. The accuracy (0.2% or better) is sufficient for our purposes. The routine begins by using a linear approximation to estimate the square root of a number between 1 and 4. (All other numbers can be scaled to values between 1 and 4 by multiplying or dividing by 2^N or where N is an even integer.) This linear approximation is further improved as follows. Let x be the number whose square root we wish to obtain, and let s_1 be the estimate obtained from the linear approximation. Our goal is to obtain a better estimate, s_2 . I use the following formula to accomplish this:

$$s_2 = \frac{s_1 + x/s_1}{2}$$

Once the SWR is calculated, it is displayed on the front panel analog meter. As discussed earlier, this display is facilitated using a pulse-width-modulated output signal from the processor. The pulse width can be varied from 0 (i.e., the output signal is always 0 V) to 255 (the output signal is always high, at about 5 V). The circuitry driving the meter must be adjusted so that an output of 255 will cause a full scale deflection of the front panel analog meter.

Actually, the SWR is not directly displayed. Rather, the quantity $255 \times (1 - 1/\text{SWR})$ is displayed. I have found that this provides a superior meter indication to use when adjusting an antenna tuner for lowest SWR.

Subroutine DisplayPeakPowerAndSWR

This routine writes the peak forward or load power (depending on the setting of the front panel power switch shown in Figure 1) to the LCD display. First, the routine determines if the power is large enough to write to the screen. If not, it writes the word “low”. It next determines if the forward or load power is to be written. If needed, load power, P_L , is calculated from the formula

$$P_L = (1 - |\rho|^2) P_F,$$

where $|\rho|$ is the reflection coefficient and P_F is the measured peak forward power. $|\rho| = \sqrt{P_R/P_F}$ is calculated during the SWR calculation and its square is stored in the data register LoadFactor. Once the peak power is calculated, the subroutine FormatWatts is called. This

routine decides whether the power is to be displayed in units of watts (W) or milliwatts (mW). The power is next displayed with the appropriate units.

Finally, this subroutine formats the SWR for LCD display, by calling the subroutine FormatSWR, and then displays the formatted SWR.

Subroutine DisplayAvgPower

This subroutine first checks to make sure the peak power was large enough to display. If so, it then calculates the load power if needed, calls FormatWatts to format the power for display, and then writes the average forward (or load) power to the front panel LCD display.

Subroutine LCD_DisplayASCII Num

This subroutine writes a number to the LCD display. The decimal digits (in ASCII format) of the number to be written must already be in the data registers ASCII_Num through ASCII_Num+4. The sign of the number (i.e., positive or negative) must be indicated by the presence of 0 (for positive) or 1 in the register ASCII_Sign. The location of the decimal point is contained in ASCII_Decimal: The number in this register gives the number of digits to the right of the decimal point. Finally, ASCII_MinField gives the minimum number of spaces to allocate to the number.

Subroutine Int16ToASCII Num

This subroutine converts a 16-bit integer, initially located in aargb0 and aargb1, to ASCII form. The result is placed in the registers ASCII_Num through ASCII_Num+4.

The largest number that can be contained in 16 binary bits is $2^{16} - 1 = 65,535$. The routine thus first repetitively subtracts 1000 from the integer to be converted until the result goes negative. The total number of subtractions before the remainder went negative is the first decimal digit. It is converted to ASCII form by adding 0x30. Next, 100 is repetitively subtracted from the remainder, thus determining the second digit. This process continues until all five decimal digits are converted.

Subroutine GetForwardWatts

As discussed previously, this subroutine is called with a 10-bit value from the forward power ADC and returns with the corresponding forward power in watts. A system of lookup tables is used to accomplish this task.

The listing for this program starts on page 27 of Appendix 1. The first instruction sets `rp0` in status so that bank 1 of data memory can be accessed. The program then calls `_GetForwardLowByte`. The purpose of this program is to obtain the low (least significant) byte of the forward power. The subroutine is also on page 27. It first sets bits in `PCLATH` to access page 2 of program memory, where the forward power lookup tables are located. It then checks bits 6 and 7 of `bForwardNew + 1`. `bForwardNew` holds the 10-bit digitized value of forward power. The most significant 8 bits are located in `bForwardNew` and the least significant 2 bits are bits 7 and 6 of `bForwardNew + 1`. Depending on the contents of these latter two bits, bits in `PCLATH` are set appropriately so one of four 256-word-long lookup tables can be accessed. Finally, the contents of `bForwardNew` (i.e., the most significant 8 bits) are loaded into `PCL`, the lower byte of the program counter. As discussed earlier, the effect of this final operation is to cause the processor to next execute the instruction pointed to by `PCLATH` and `PCL`. The lookup tables begin on page 51 of Appendix 1 and consist of nothing more than a whole string of `retlw` instructions. The value of `PCLATH` determines which of four tables will be accessed, and the value of `PCL` will determine which `retlw` instruction within the table will be executed. Whichever `retlw` is executed will cause the processor to return to the instruction following the last subroutine call with the value of the low byte of the 16-bit floating-point power in watts loaded in the `w` register.

Finally, let me describe how I construct these tables. As mentioned earlier, the response of the forward (or reflected) power channel can be characterized by the linear relation

$$dBm = \alpha \times ADC + \beta$$

where dBm is the power input to the forward channel, measured in units of dBm and ADC is the resulting digitized value of this power. The constants, α and β , must be determined experimentally. The power in units of dBm can be converted to power, P , in units of watts using the following formula:

$$P = (0.001 \text{ W}) \times 10^{dBm/10}.$$

Combining these formulae:

$$P = (0.001 \text{ W}) \times 10^{(\alpha \times \text{ADC} + \beta)/10}.$$

The lookup tables included in Appendix 1 are the ones I used for my power meter based on the calibration data acquired with it. Unfortunately, they are not terribly accurate because I do not have a good way to accurately measure rf power in watts. I estimate their uncertainty to be about $\pm 10\%$. One of my goals is to acquire the equipment to more accurately measure rf power. Consequently, it would be good if anyone who duplicates my power meter would do their own calibration. This will, of course, require that the lookup tables in Appendix 1 be changed to match the new calibration data.

The best way to do the calibration would be feed known powers through the directional coupler and record the resulting ADC readings. (Doing this way calibrates the directional coupler, the 20-dB attenuators, and the AD8307 chips.) Then, the powers in watts should be converted to units of dBm. Next a straight line should be fit to the dBm data as a function of the ADC data. If more than two power points are obtained, this line could be fit using the least-squares method. The slope and intercept of the resulting line are the constants α and β , respectively, defined above.

The new forward-power and reflected-power lookup tables can then be constructed by hand using these formulae. This would, however, be very tedious. As an alternative, I wrote a computer program to do this. This program, written in Microsoft Visual Basic 6.0, enables the user to enter values for α and β (named the “slope” and “intercept” in the program), tell the program whether the forward or reflected power channel lookup tables are being constructed, and then construct them. The tables are shown in a box. These tables can then be saved to a file. The file can be later opened using a text editor such as Notepad, the contents copied to the clipboard, and then pasted into the assembly language listing.

The lookup table program is a Windows program and is installed in the same way normal windows programs are installed. There are three files involved in this process: Power Tables.Cab, setup.exe, and SETUP.LST. They are included on the ARRL website. Follow this procedure to install this program on your computer:

1. Download the three files listed in the preceding paragraph from the ARRL website into a convenient temporary file.

2. Using Windows Explorer (or equivalent) create a subdirectory to the Program Files directory on your main hard drive named "PowerMeterLookup Tables".
3. Start (double click) setup.exe
4. Follow the onscreen prompts. At one point, the installation program will ask where you want to place the program files. Using the Browse button, select \Program Files\PowerMeterLookupTables.
5. Finish the installation process.
6. To run the program, hit the Start button, select programs, then select the "Power Meter Lookup Tables" folder, and double click on the program "Power Meter Lookup Tables". On my computer, the first time you do this, the system will ask you to help find the program. Hit the "Browse" button, and select the program directory where you placed the files. Apparently, you only have to do this once.

Appendix 1. List of Power/SWR Meter Software

```

; **POWER METER, Hardware Version 2, Software Version 2*****
include <p16f876a.inc>
;*****Constants*****

; To facilitate simulation, use the Simulate variable. It has the following
; values:
;
;      Value      Function
;      0          No simulation
;      1          Simulate

Simulate          equ 0

ADRES_STEP        equ 0

; Switch to determine whether software runs in calibrate (Calibrate = 1) or normal (Calibrate = 0) mode.
; In calibrate mode, the processor will read the forward and reflected power channels

Calibrate         equ 0

; The following constants are only used if Simulate = 1

ForwardOrLoadWatts equ 1 ; Set 1 for Forward, 0 for Load Watts
LongOrShortAvgTime equ 1 ; Set 1 for long averaging time

; This constant defines the binary low-power threshold, where the display shows
; "Low power".
LowPowerThreshold equ 404/4 ; Corresponds to forward powers < about 1 mW

; Time (microseconds) between consecutive samples
SamplePeriod      equ 90
TMR1_Init         equ 65536 - 5*SamplePeriod + 16

if Calibrate == 1
NumSamples        equ 1024 ; This value must not be changed.
else
if Simulate > 0
MaxNumPeakWaits   equ 30 ; Number of waits in peak mode
MaxNumSWRMeterWaits equ 16 ; Number of consecutive low power readings to set SWR meter to 0
NumSamples        equ 100000/SamplePeriod ; Number of average-mode samples in one cycle
NumCyclesShort    equ 16 ; Number of cycles in a series for "short" averaging
NumCyclesLong     equ 48 ; Number of cycles in a series for "long" averaging
else
MaxNumPeakWaits   equ 30 ; Number of waits in peak mode
MaxNumSWRMeterWaits equ 16 ; Number of consecutive low power readings to set SWR meter to 0
NumSamples        equ 100000/SamplePeriod ; Number of average-mode samples in one cycle
NumCyclesShort    equ 16 ; Number of cycles in a series for "short" averaging
NumCyclesLong     equ 48 ; Number of cycles in a series for "long" averaging
endif
endif

;*****Packed 16 bit floating point values*****

; This routine uses a packed 16-bit floating point value that I developed to reduce
; memory usage in a look-up table relating power in watts to measured adc values for
; forward and reflected powers.
;
; The packed values contain a 10-bit "reduced" mantissa (m) and a 6 bit shifted exponent (s).
; All values are assumed positive so there is no need for a sign bit. Also, the first bit
; in the mantissa, which is always 1 except when the number is 0, is suppressed.
; The offset value for the exponent is 27. A shifted exponent of 0 means the number is 0.
;
; For example, the number 0 has a packed 16 bit value of 0x0000.

; Next, consider the number 1. This can be written as 1.0 * 2^0. The reduced mantissa
; is 0 and the shifted exponent is 27 + 0 = 27 = 0x2B = 0b011011. Thus, the packed 16-bit
; value is 0b0110110000000000 = 0x6C00

```

```

;
; Next take the number 1107 = 0x453. In binary, this can be written
; 10001010011 = 1.0001010011 * 2^10. The reduced mantissa is 00 0101 0011 and the exponent
; is 27 + 10 = 37 = 0x25 = 100101. Thus the packed value is 1001 0100 0101 0011
; = 0x9453
;
; The largest value that can be represented is in binary 1.11 1111 1111 & 2^(63 - 27)
; = 111 1111 1111 * 2^26 = 0x7FF * 2^26 = 137,371,844,608 = 1.37371844608E+11
; and the smallest value is 1.00 0000 0000 * 2^(1 - 27) = 1 * 2^-26 = 1.49011611938E-8.

; This 16-bit format is used only in the interrupt service routine and for looking up
; powers give a particular ADC reading. All other arithmetic is done using a 24-bit format
; developed by Microchip.

```

```

;*****CONSTANTS & MEMORY ALLOCATIONS*****
;

```

```

; First make general register assignments

```

```

    cblock        0x20
; Next group of variables used by MicroChip floating point routines.
    exp:0, aexp:0
    aarg,aargb0, aargb1, aargb2
    aargb3, aargb4, aargb7
    sign
    fpflags
    bexp:0, barg,bargb0,bargb1,bargb2

```

```

; Following variables used by my floating point routines and by
; Watts lookup table routines

```

```

    isign, intPart
    temp:0, tempb0, tempb1
    t1:3, t2:3, t3:3

```

```

; Variables for construction of and display of ASCII numbers
    ASCII_num:5, ASCII_Sign, ASCII_Decimal, ASCII_MinField

```

```

; More temporaray variables
    Timer, Timer2, Timer3
    LCDTemp, LCDChar, LCDTextPtr

```

```

; Floating point variables for power meter data
    SWR:3, PeakForwardWatts:3

```

```

; This variable denotes various statuses associated with power levels

```

```

;      Bit  Value  Means
;      -----
;      0      0      PeakForwardWatts >= LowPowerThreshold
;      0      1      Peak ForwardWatts <  LowPowerThreshold
;
;      2      0      Display Mode = Forward - Reflected Power (Watts)
;      2      1      Display Mode = Forward Power (Watts)
;
;      4      0      Short averaging time
;      4      1      Long averaging time
;
;      5      0      Displays power in units of W
;      5      1      Displays power in units of mW

```

```

    ModeStatusFlags

```

```

; Register to count # of cycles.
    CycleCounter

```

```

; This variable holds the total number of samples used to calculate average power
    FPInvTotalSamplesShort:3
    FPInvTotalSamplesLong:3

```

```

; Counter for number of waits, after peak is found
    PeakWaitCounter

```

```

; Counter for number of waits, while power is low, before SWR meter set to 0

```

```

        SWRMeterWaitCounter

; More temporary variables
    temp1, temp2, temp3, temp4, TableTemp

; Reflection Coefficient squared
    LoadFactor:3

; Debug Registers
    Debug1, Debug2

    endc

; The definitions used to access bits in ModeStatusFlags

LowPeakWatts    equ 0
ForwardWatts    equ 2
LongAvgTime     equ 4
MilliWattsFlag  equ 5

; Temporary storage for critical variables during interrupt service

    cblock      0x7D

    w_temp, status_temp, PCLATH_temp

    endc

; Variables located in Block 1

    cblock      0xA0

; Accumulator for forward power running sum, and FP packed value.
    Ch0Avg:0, FPaccum:4, Ch1Avg:0, FP:4
    B1Temp1, B1Temp2, B1Temp3

; Variables used to hold digital values1 produced by ADC
    Ch0Max:0, bForward:2
    Ch0Min:0, bForwardPeak:2
    Ch1Max:0, bForwardNew:2
    Ch1Min:0, bReflected:2

; Status flags
    Flags

; Loop counter for average and peak modes
    SampleCounter:2

; Temporary holders for ADC values
    Ch0Value:2, Ch1Value:2

; Temp registers used only in simulate mode
    S_ADRESL, S_ADRESH

; Bank1 debug registers
    B1_Debug1, B1_Debug2

    endc

; Definition of bits of Flags

NewPeak equ 0
SampleDoneFlag equ 1
LZflag equ 2
BusyFlag equ 3

; Constants used for selecting pages

psel0 equ 3
psel1 equ 4

bit0 equ 0x0
bit1 equ 0x1
bit2 equ 0x2
bit3 equ 0x3
bit4 equ 0x4

```

```

bit5 equ 0x5
bit6 equ 0x6
bit7 equ 0x7

B0 equ 0
B1 equ 1
B2 equ 2
B3 equ 3
B4 equ 4
B5 equ 5
B6 equ 6
B7 equ 7

MSB equ 7
LSB equ 0

#define _C STATUS,0
#define _Z STATUS,2

;
; FLOATING POINT SPECIFIC DEFINITIONS
;
; literal constants
;
EXPBIAS equ D'127'
;
; floating point library exception flags
;

IOV equ 0 ; bit0 = integer overflow flag
FOV equ 1 ; bit1 = floating point overflow flag
FUN equ 2 ; bit2 = floating point underflow flag
FDZ equ 3 ; bit3 = floating point divide by zero flag
NAN equ 4 ; bit4 = not-a-number exception flag
DOM equ 5 ; bit5 = domain error exception flag
RND equ 6 ; bit6 = floating point rounding flag, 0 = truncation
SAT equ 7 ; bit7 = floating point saturate flag, 0 = terminate on
; exception without saturation, 1 = terminate on
; exception with saturation to appropriate value

;*****MACRO DEFINITIONS*****

MoveBytes macro AA, BB, nbytes ; Macro that moves nbytes bytes from

    if (0x80 & AA) == (0x80 & BB)
        if (0x80 & AA) != 0
            bsf status,rp0
        else
            bcf status,rp0
        endif
    endif

    local i = 0 ; a through a + nbytes - 1 to
    while i < nbytes ; b through b + nbytes - 1
        if (0x80 & AA) != (0x80 & BB)
            if (0x80 & AA) != 0
                bsf status,rp0
            else
                bcf status,rp0
            endif
        endif

        movf 0x7F & (AA + i),w

        if (0x80 & AA) != (0x80 & BB)
            if (0x80 & BB) != 0
                bsf status,rp0
            else
                bcf status,rp0
            endif
        endif

        movwf (0x7f & (BB + i))
        i set i + 1

    endw
endm

; This macro stores a 24-bit floating point literal in three consecutive
; memory locations starting at Address in Bank 0.

```

```

StoreFPLiteral macro Literal,Address
    movlw    (0xff0000 & Literal) >> 16    ; Get top 8 bits
    movwf    Address
    movlw    (0xff00 & Literal) >> 8        ; Get middle 8 bits
    movwf    Address + 1
    movlw    (0xff & Literal)               ; Get lower 8 bits
    movwf    Address + 2
endm

;-----LCD_DisplayText-----
; This macro sends text to the LCD display, starting at the current
; cursor position. The first argument of the macro is the text to be displayed. The
; second is a unique address which is used internally within the routine.
;
; For example, suppose one wishes to output the string "Test output" to the
; LCD display, starting at the current cursor position.
; The following code would do this:
;
;   LCD_DisplayText  "Test output",Adr8898

    if Simulate > 0
LCD_DisplayText macro TextString, UniqueAddress
    endm

    else
LCD_DisplayText macro TextString, UniqueAddress
    bcf     status,rp0                ; Select bank 0
    bcf     PCLATH,psel0              ; Select page 0
    bcf     PCLATH,psell

    clrf    LCDTextPtr                ; Start with first character in string, i.e., with 0 offset

    movf    LCDTextPtr,w              ; Load offset
    call    ($+7)                    ; Get next character from table
    andlw   0xff                      ; Test if character is 0
    btfs    status,z                 ; Skip if not zero
    goto    UniqueAddress             ; Finished-go to end
    call    LCD_SendChar
    incf    LCDTextPtr,f              ; Increment to point to next character
    goto    $-7                      ; Do next character

; The next 8 instruction and the lookup table that follows it comprise a subroutine.

    movwf   TableTemp                ;Store offset temporarily
    movlw   HIGH ($+7)                ;High part of address of table
    movwf   PCLATH                    ;stored in PCLATH
    movf    TableTemp,w               ;Load offset back into w
    addlw   LOW ($+4)                 ;Offset into table
    btfs    status,c                  ;Carry on add?
    incf    PCLATH,f                  ;Yes
    movwf   PCL                       ;Enter address of table entry
    dt      TextString,0              ;Defines table
UniqueAddress
    endm
    endif

; Macros for ADC use

; Initialize the ADC unit.
ADC_Initialize macro
    bsf     status,rp0
    movlw   0x05                      ; Select AN0, AN1, Vref as analog inputs, left justified result
    movwf   0x7f & ADCON1
    bcf     status,rp0

    movlw   0x81                      ; Set up for AN0 and turn ADC on
    movwf   ADCON0

    bcf     PCLATH,psel0 ; Select Page 0
    bcf     PCLATH,psell

    movlw   7                          ; Wait 7 microseconds for system
    call    WaitMicroSeconds ; to stabilize
    endm

```

```

ADC_StartRead    macro
    bsf          ADCON0,2
    if Simulate==1
        movlw    LOW(ADRES_STEP)                ; Subtract ADRES_STEP from S_ADRES-
        bsf      status,rp0
        subwf    0x7F & S_ADRESL,f
        btfss    status,c
        decf     0x7F & S_ADRESH,f
        movlw    HIGH(ADRES_STEP)
        subwf    0x7F & S_ADRESH,f
        bcf      status,rp0
    endif
endm

ADC_ReadFinished macro
    if Simulate==0
        btfsc    ADCON0,2
        goto     $-1
    else
        bsf      status,rp0
        movf     0x7F & S_ADRESL,w
        movwf    0x7F & ADRESL
        movf     0x7F & S_ADRESH,w
        bcf      status,rp0
        movwf    ADRESH
    endif
endm

ADC_SetCh0       macro
    bcf          ADCON0,3
endm

ADC_SetCh1       macro
    bsf          ADCON0,3
endm

if Calibrate == 0
;***** CODE FOR NORMAL MODE *****
    org         0x0
    goto        StartUp

; This interrupt driven routine is the inner sampling loop.

    org         0x04
ISR
    movwf       w_temp
    swapf       status,w
    movwf       status_temp

    bcf         status,rp0
    bcf         PORTA,2                ; S/H converters -> hold
    bsf         status,rp0
    bcf         0x7F & Flags, NewPeak
    bcf         status,rp0

    bcf         T1CON,TMR1ON           ; Turn Timer1 off and initialize it

    movlw       LOW(TMR1_Init)
    movwf       TMR1L
    movlw       HIGH(TMR1_Init)
    movwf       TMR1H
    bsf         T1CON,TMR1ON           ; Turn Timer 1 on

    bcf         PIR1,TMR1IF           ; Turn off Timer1 interrupt flag

    bsf         status,rp0
    btfsc       0x7F & Flags,BusyFlag   ; Are we already processing a measurement? If so, do not start another one

    goto        ISR_040

    bsf         0x7F & Flags,BusyFlag   ; Set busy flag, indicating a new measurement is being processed.
    bcf         status,rp0

    movlw       16
    call        WaitMicroSeconds        ; Wait a total of 12 microseconds for S/H to settle of held value

    ADC_StartRead

```

```

ADC_ReadFinished
ADC_SetCh1                ; Set ADC to Channel 1 to read reflected power.

movf      ADRESH,w        ; Transfer result of ADC to bForwardNew
bsf       status,rp0
movwf     0x7F & bForwardNew
movf      0x7F & ADRESL,w
movwf     0x7F & bForwardNew + 1

; Is the newly measured value of Forward power greater than the previous maximum?
movf      0x7F & bForwardNew,w
subwf     0x7F & bForward,w    ; 256 + (bForward - bForwardNew): c = 0 means bForwardNew > bForward
btfss     status,c
goto      ISR_012           ; bForwardNew < bForward, so new maximum found
btfss     status,z
goto      ISR_019           ; bForward < bForwardNew so no new maximum

movf      0x7F & bForwardNew+1,w    ; MSB's equal so check LSB's
subwf     0x7F & bForward+1,w    ; 256 + ((bForward+1) - (bForwardNew+1))
btfsc     status,c
goto      ISR_019           ; (bForwardNew+1) > (bForward+1)

ISR_012                ; New maximum found so replace bForward with bForwardNew
movf      0x7F & bForwardNew,w
movwf     0x7F & bForward
movf      0x7F & bForwardNew+1,w
movwf     0x7F & bForward+1
bsf       0x7F & Flags,NewPeak    ; Set flag for latter use

ISR_019
call      GetForwardWatts
bcf       status,rp0

ADC_StartRead          ; Start Read of reflected power

bcf       PCLATH,psel1    ; Select Page 1
bsf       PCLATH,psel0

call      0x7FF & SumP16    ; Update accumulated value
bcf       PCLATH,psel0

bcf       status,rp0

ADC_ReadFinished

bsf       PORTA,2        ; S/H to sample mode
ADC_SetCh0            ; ADC to Channel 0 (Forward power)
bsf       status, rp0
btfss     (0x7F & Flags),NewPeak    ; Was a new maximum forward power measured?
goto      ISR_021
bcf       status,rp0
movf      ADRESH,w        ; Yes, so update bReflected with new value
bsf       status,rp0
movwf     0x7F & bReflected
movf      0x7F & ADRESL,w
movwf     0x7F & bReflected+1

ISR_021                ; Decrement two-byte sample counter
decfsz    0x7F & SampleCounter+1,f ; Decrement LSByte
goto      ISR_031        ; Byte not zero
bsf       0x7F & Flags,LZflag
movf      0x7F & SampleCounter,f
btfsc     status,z
bsf       0x7F & Flags,SampleDoneFlag
goto      ISR_040

ISR_031
btfss     0x7F & Flags,LZflag
goto      ISR_040
decf      0x7F & SampleCounter,f
bcf       0x7F & Flags,LZflag

ISR_040
swapf     status_temp,w
movwf     status
swapf     w_temp,f
swapf     w_temp,w
retfie

```



```

StartUp

; Initialize I/O ports
bsf      status,rp0
movlw    0xFB      ; PORTA: all 5 bits inputs except for bit 2
movwf    0x7F & TRISA

movlw    0x00      ; PORTB: all 8 bits inputs normally
movwf    0x7F & TRISB

movlw    0x8B      ; PORTC: Bits 0, 1 unused. Bit 2 CCP1 output. Bit 3 S1 input. Bits 4-6 LCD control pins (outputs)
movwf    0x7F & TRISC      ; Bit 7 S2 input

; Initialize LCD module and display introductory screen
call     LCD_Init
LCD_DisplayText " W7IEQ Power Meter", uAdr10
movlw    Line2 + 1
call     LCD_SetLineAndColumn
LCD_DisplayText "      Version 2", uAdr237
movlw    30
call     WaitTenthSeconds

; Initialize the ADC unit.
ADC_Initialize

bcf      status,rp0
bsf PORTA,2      ; Put S/H chips into sample mode

call     PWM_Initialize      ; Initialize pulse width modulated output

; Setup Timer 1 interrupts for sample control

bsf      status,rp0
bsf      0x7F & PIE1,TMR1IE      ; Arm Timer 1 interrupt
bsf      INTCON,PEIE

; Initialize floating point routines by selecting "rounding" and "saturation"

bcf      status,rp0
movlw    0xC0
movwf    fpflags

; Convert NumSamples*NumCyclesShort to floating point, and invert

movlw    LOW(NumSamples)
movwf    aargb1
movlw    HIGH(NumSamples)
movwf    aargb0
bsf      PCLATH,psel0
call     0x7ff & FLO1624
MoveBytes aarg,t1,3
MoveBytes aarg,barg,3
movlw    NumCyclesShort
movwf    aargb1
clrf     aargb0
call     0x7ff & FLO1624
call     0x7ff & FPM24
MoveBytes aarg,barg,3
movlw    0x7f
movwf    aarg
clrf     aarg+1
clrf     aarg+2
call     0x7ff & FPD24
MoveBytes aarg,FPIInvTotalSamplesShort,3

; Convert NumSamples*NumCyclesLong to floating point, and invert

movlw    NumCyclesLong
movwf    aargb1
clrf     aargb0
call     0x7ff & FLO1624
MoveBytes t1,barg,3
call     0x7ff & FPM24
MoveBytes aarg,barg,3
movlw    0x7f
movwf    aarg
clrf     aarg+1

```

```

        clrf          aarg+2
        call          0x7ff & FPD24
        MoveBytes aarg,FPInvTotalSamplesLong,3
        bcf           PCLATH,psel0

; Initialize for normal operation

        clrf          ModeStatusFlags

        if Simulate==1
; Initialize the simulated ADCRES registers
        bsf           status,rp0
        movlw         0x40
        movwf         0x7F & S_ADRESL
        movlw         0x97
        movwf         0x7F & S_ADRESH
        bcf           status,rp0
        endif

;Initialize for Peak Hold mode

        bsf           status,rp0
        clrf          0x7F & bForwardPeak      ; Start with 0 watts
        clrf          0x7F & bForwardPeak+1
        bcf           status,rp0
        movlw         MaxNumPeakWaits
        movwf         PeakWaitCounter

; Initialize Meter wait counter

        movlw         MaxNumSWRMeterWaits
        movwf         SWRMeterWaitCounter

;Initialize the display.
        bcf           PCLATH,psel0
        call          LCD_ClearDisplay
        LCD_DisplayText "          SWR", uAdr397

StartSeries
        bcf           PCLATH,psel0
        bcf           status,rp0
; Determine display mode: Forward or Load Power.

        if Simulate == 0
        bcf ModeStatusFlags,ForwardWatts
        btfscc        PORTC,3          ; Test setting of mode switch
        bsf ModeStatusFlags,ForwardWatts
        else
        if ForwardOrLoadWatts == 0
        bcf          ModeStatusFlags,ForwardWatts
        else
        bsf          ModeStatusFlags,ForwardWatts
        endif
        endif

; Finish Initializing the Display
        btfscc        ModeStatusFlags,ForwardWatts
        goto          Main_401
        movlw         Line1+1
        call          LCD_SetLineAndColumn
        LCD_DisplayText "PEP-F=",uadr8133
        movlw         Line2+1
        call          LCD_SetLineAndColumn
        LCD_DisplayText "AEP-F=",uadr8134
        goto          Main_402
Main_401
        movlw         Line1+1
        call          LCD_SetLineAndColumn
        LCD_DisplayText "PEP-L=",uadr8135
        movlw         Line2+1
        call          LCD_SetLineAndColumn
        LCD_DisplayText "AEP-L=",uadr8136
Main_402

        if Simulate==0
        btfscc        PORTC,7          ; Is Averaging time switch set?
        goto          $+4
        movlw         NumCyclesLong

```

```

        bsf          ModeStatusFlags,LongAvgTime
        goto         $+3
        movlw        NumCyclesShort
        bcf ModeStatusFlags,LongAvgTime
    else
        if LongOrShortAvgTime == 1
            movlw      NumCyclesLong
            bsf        ModeStatusFlags,LongAvgTime
        else
            movlw      NumCyclesShort
            bcf        ModeStatusFlags,LongAvgTime
        endif
    endif

    movwf            CycleCounter

; Clear the forward power accumulator

    bsf          status,rp0
    clrf         0x7F & FPaccum
    clrf         0x7F & FPaccum + 1
    clrf         0x7F & FPaccum + 2
    clrf         0x7F & FPaccum + 3

NextCycle
    bsf          status,rp0

    movlw        HIGH(NumSamples)      ; Initialize inner loop counter
    movwf        0x7F & SampleCounter
    movlw        LOW(NumSamples)
    movwf        0x7F & SampleCounter+1

    clrf         0x7F & Flags
    if LOW(NumSamples) == 0
        bsf      0x7F & Flags,LZflag
    endif

    clrf         0x7F & bForward
    clrf         0x7F & bForward+1

; Start a sampling cycle

    call         ISR                      ; Call interrupt routine to start sampling loop

; Now loop until SampleCounter has been decremented to 0
    bsf          status,rp0

ZeroCheck
    bcf          0x7F & Flags,BusyFlag
    btfss        0x7F & Flags,SampleDoneFlag
    goto         $-2

; Sample counter is 0. Disable interrupt and continue with
; processing at end of one cycle.

    bcf          INTCON,GIE

; Is this peak greater than the previous peak?

    movf         0x7F & bForward,w
    subwf        0x7F & bForwardPeak,w ; 256 + (bForwardPeak - bForward): c = 0 means new peak
    btfss        status,c
    goto         hh001                  ; New peak value found
    btfss        status,z                ; Are MSB's equal. If so, check lower bytes
    goto         hh003                  ; Not a new peak
    movf         0x7F & bForward+1,w
    subwf        0x7F & bForwardPeak+1,w ; 256 + ((bForwardPeak+1) - (bForward+1)): c = 0 means new peak
    btfss        status,c
    goto         hh001                  ; New peak value found

; There is no new peak value. Keep the previous peak for a maximum
; of MaxNumPeakWaits sampling cycles.
hh003
    bcf          status,rp0
    decfsz       PeakWaitCounter,f

```

```

        goto          hh002          ; Keep previous peak but update SWR meter

; Come here if a new peak has been found, or the old peak value has existed for
; MaxNumPeakWaits sampling cycles.

hh001
MoveBytes    bForward,bForwardPeak,2
MoveBytes    bForward,bForwardNew,2

call         GetForwardWatts

bcf          PCLATH,psel1           ; Set to page 1
bsf          PCLATH,psel0
clrf         0x7F & FP+2
call         0x7FF & P16ToFP24      ; Convert P16 forward power to FP24 format in aarg

MoveBytes    aarg,PeakForwardWatts,3

bcf          PCLATH,psel0

call         GetReflectedWatts
clrf         0x7F & FP+2
bcf          PCLATH,psel1
call         0x7FF & P16ToFP24

MoveBytes    PeakForwardWatts,barg,3
bcf          PCLATH,psel0

call         CalculateSWRandSetMeter
bcf          PCLATH,psel0
call         DisplayPeakPowerAndSWR

movlw        MaxNumPeakWaits
movwf        PeakWaitCounter
goto         c002

; Come here if not a new peak.

hh002
MoveBytes    bForward,bForwardNew,2

call         GetForwardWatts
clrf         0x7F & FP+2
bcf          PCLATH,psel1           ; Set to page 1
bsf          PCLATH,psel0

call         0x7FF & P16ToFP24      ; Convert P16 forward power to FP24 format in aarg

MoveBytes    aarg,PeakForwardWatts,3

bcf          PCLATH,psel0

call         GetReflectedWatts
clrf         0x7F & FP+2
bcf          PCLATH,psel1
call         0x7FF & P16ToFP24

MoveBytes    PeakForwardWatts,barg,3
bcf          PCLATH,psel0

call         CalculateSWRandSetMeter

c002
decfsz       CycleCounter,f
goto         NextCycle

MoveBytes    FPaccum,FP,3
bsf          PCLATH,psel0
call         0x7FF & P16ToFP24
bcf          PCLATH,psel0

btfss        ModeStatusFlags,LongAvgTime ; Is averaging time long?
goto         cc031                  ; No
MoveBytes    FPInvTotalSamplesLong,barg,3
goto         cc032

cc031
MoveBytes    FPInvTotalSamplesShort,barg,3

cc032
bsf          PCLATH,psel0
call         0x7ff & FPM24

```

```

        bcf          PCLATH,psel0
        call         DisplayAvgPower

        goto         StartSeries
endif

if Calibrate == 1
;***** CODE for CALIBRATE MODE *****
        org 0x0
        goto         StartUp

        org 0x5
CalMode
StartUp

        bsf          status,rp0
        bcf          0x7F & TRISA,2
        bcf          status,rp0

        movlw        10
        call WaitTenthSeconds

; General initialization

; Initialize I/O ports
        bsf          status,rp0
        movlw        0xFB      ; PORTA: all 5 bits inputs except for bit 2
        movwf        0x7F & TRISA

        movlw        0x00      ; PORTB: all 8 bits outputs normally
        movwf        0x7F & TRISB

        movlw        0x8B      ; PORTC: Bits 0, 1 unused. Bit 2 CCP1 output. Bit 3 S1 input. Bits 4-6 LCD control pins (outputs)
        movwf        0x7F & TRISC ; Bit 7 S2 input
        bcf          status,rp0

; Initialize LCD module and display introductory screen
        call         LCD_Init

        LCD_DisplayText " W7IEQ Power Meter", uAdr10
        movlw        Line2 + 1
        call LCD_SetLineAndColumn
        LCD_DisplayText " Calibrate Mode", uAdr237
        movlw        50
        call WaitTenthSeconds

; Initialize the ADC unit.
        ADC_Initialize

        bcf          status,rp0
        bsf PORTA,2      ; Put S/H chips into sample mode

; Initialize for normal operation
        bsf          status,rp0
        clrf         0x7F & Flags

;Initialize the display.
        call         LCD_ClearDisplay
        LCD_DisplayText "CH0=", uAdr397
        movlw        Line2 + 1
        call LCD_SetLineAndColumn
        LCD_DisplayText "CH1=", uAdr137

; Initialize the simulated ADRES registers
        bsf          status,rp0
        clrf         0x7F & S_ADRESL
        clrf         0x7F & S_ADRESH

; Initialize the PWM unit and set duty cycle to 100%

        bcf          status,rp0
        call         PWM_Initialize
        movlw        255

```

```

        call        PWM_SetDutyCycle

StartSeries
        bcf         PCLATH,psel0
        bsf         status,rp0

        movlw       HIGH(NumSamples)
        movwf       0x7F & SampleCounter
        movlw       LOW(NumSamples)
        movwf       0x7F & SampleCounter+1

; Clear the accumulators for Channel 1 and Channel 2

        clrf        0x7F & Ch0Avg
        clrf        0x7F & Ch0Avg + 1
        clrf        0x7F & Ch0Avg + 2
        clrf        0x7F & Ch0Avg + 3
        clrf        0x7F & Ch1Avg
        clrf        0x7F & Ch1Avg + 1
        clrf        0x7F & Ch1Avg + 2
        clrf        0x7F & Ch1Avg + 3
        clrf        0x7F & Ch0Max
        clrf        0x7F & Ch0Max+1
        clrf        0x7F & Ch1Max
        clrf        0x7F & Ch1Max+1
        movlw       0xFF
        movwf       0x7F & Ch0Min
        movwf       0x7F & Ch1Min
        movlw       0xC0
        movwf       0x7F & Ch0Min+1
        movwf       0x7F & Ch1Min+1

NextSample

        bcf         status,rp0
        bcf         PORTA,2                ; Set S/H to hold
        ADC_SetCh0                ; Select ADC channel 0
        movlw       30
        call        WaitMicroSeconds
        ADC_StartRead                ; Start AD conversion Channel 0
        ADC_ReadFinished            ; Wait for conversion to finish

        movf        ADRESH,w
        bsf         status,rp0
        movwf       0x7F & Ch0Value
        movf        0x7F & ADRESL,w
        movwf       0x7F & Ch0Value+1
        bcf         status,rp0

        ADC_SetCh1                ; Select ADC channel 0
        movlw       30
        call        WaitMicroSeconds
        ADC_StartRead                ; Start AD conversion Channel 0
        ADC_ReadFinished            ; Wait for conversion to finish

        movf        ADRESH,w
        bsf         status,rp0
        movwf       0x7F & Ch1Value
        movf        0x7F & ADRESL,w
        movwf       0x7F & Ch1Value+1

        bcf         status,rp0
        bsf         PORTA,2                ; S/H to sample
        bsf         status,rp0

        movf        0x7F & Ch0Value+1,w    ; Add new value to Ch0Avg
        addwf       0x7F & Ch0Avg+3,f
        btfs        status,c
        goto        CalMode_101
        incf        0x7F & Ch0Avg+2,f
        btfs        status,z
        goto        CalMode_101
        incf        0x7F & Ch0Avg+1,f
        btfs        status,z

```

```

    incf      0x7F & Ch0Avg,f
CalMode_101
    movf      0x7F & Ch0Value,w
    addwf     0x7F & Ch0Avg+2,f
    btfss     status,c
    goto      CalMode_102
    incf      0x7F & Ch0Avg+1,f
    btfsc     status,z
    incf      0x7F & Ch0Avg,f
CalMode_102

    movf      0x7F & Ch1Value+1,w      ; Add new value to Ch1Avg
    addwf     0x7F & Ch1Avg+3,f
    btfss     status,c
    goto      CalMode_103
    incf      0x7F & Ch1Avg+2,f
    btfss     status,z
    goto      CalMode_103
    incf      0x7F & Ch1Avg+1,f
    btfsc     status,z
    incf      0x7F & Ch1Avg,f
CalMode_103
    movf      0x7F & Ch1Value,w
    addwf     0x7F & Ch1Avg+2,f
    btfss     status,c
    goto      CalMode_104
    incf      0x7F & Ch1Avg+1,f
    btfsc     status,z
    incf      0x7F & Ch1Avg,f
CalMode_104

; Now see if new maximum values found

    movf      0x7F & Ch0Value,w
    subwf     0x7F & Ch0Max,w          ; 256 + (Ch0Max - Ch0Value); c = 0 means new maximum found
    btfss     status,c
    goto      CalMode_111            ; New maximum found
    btfss     status,z
    goto      CalMode_120            ; Not a new maximum
    movf      0x7F & Ch0Value+1,w     ; MSB are equal so check LSB
    subwf     0x7F & Ch0Max+1,w      ; 256 + ((Ch0Max+1) - (Ch0Value+1)); c = 0 means new maximum found
    btfsc     status,c
    goto      CalMode_120            ; Not a new maximum
CalMode_111                          ; New maximum found
    movf      0x7F & Ch0Value,w
    movwf     0x7F & Ch0Max
    movf      0x7F & Ch0Value+1,w
    movwf     0x7F & Ch0Max+1

CalMode_120
    movf      0x7F & Ch1Value,w
    subwf     0x7F & Ch1Max,w          ; 256 + (Ch1Max - Ch1Value); c = 0 means new maximum found
    btfss     status,c
    goto      CalMode_121            ; New maximum found
    btfss     status,z
    goto      CalMode_130            ; Not a new maximum
    movf      0x7F & Ch1Value+1,w     ; MSB are equal so check LSB
    subwf     0x7F & Ch1Max+1,w      ; 256 + ((Ch1Max+1) - (Ch1Value+1)); c = 0 means new maximum found
    btfsc     status,c
    goto      CalMode_130            ; Not a new maximum
CalMode_121                          ; New maximum found
    movf      0x7F & Ch1Value,w
    movwf     0x7F & Ch1Max
    movf      0x7F & Ch1Value+1,w
    movwf     0x7F & Ch1Max+1

CalMode_130
; Now see if new minimum values found

    movf      0x7F & Ch0Min,w
    subwf     0x7F & Ch0Value,w        ; 256 + (Ch0Value - Ch0Min); c = 0 means new Minimum found
    btfss     status,c
    goto      CalMode_131            ; New Minimum found
    btfss     status,z
    goto      CalMode_140            ; Not a new Minimum
    movf      0x7F & Ch0Min+1,w        ; MSB are equal so check LSB
    subwf     0x7F & Ch0Value+1,w      ; 256 + ((Ch0Value+1) - (Ch0Min+1)); c = 0 means new Minimum found
    btfsc     status,c
    goto      CalMode_140            ; Not a new Minimum
CalMode_131                          ; New Minimum found

```

```

    movf      0x7F & Ch0Value,w
    movwf    0x7F & Ch0Min
    movf      0x7F & Ch0Value+1,w
    movwf    0x7F & Ch0Min+1

CalMode_140
; Now see if new minimum values found

    movf      0x7F & Ch1Min,w
    subwf    0x7F & Ch1Value,w           ; 256 + (Ch1Value - Ch1Min); c = 0 means new Minimum found
    btfss    status,c
    goto     CalMode_141                 ; New Minimum found
    btfss    status,z
    goto     CalMode_150                 ; Not a new Minimum
    movf      0x7F & Ch1Min+1,w           ; MSB are equal so check LSB
    subwf    0x7F & Ch1Value+1,w         ; 256 + ((Ch1Value+1) - (Ch1Min+1)); c = 0 means new Minimum found
    btfsc    status,c
    goto     CalMode_150                 ; Not a new Minimum
CalMode_141
    movf      0x7F & Ch1Value,w
    movwf    0x7F & Ch1Min
    movf      0x7F & Ch1Value+1,w
    movwf    0x7F & Ch1Min+1

CalMode_150
; Have we taken enough samples?
    movlw    1
    subwf    0x7F & SampleCounter+1,f
    btfss    status,c
    decf     0x7F & SampleCounter,f
    movf     0x7F & SampleCounter+1,f
    btfss    status,z
    goto     NextSample
    movf     0x7F & SampleCounter,f
    btfss    status,z
    goto     NextSample
; All samples taken. Prepare data for display

; Divide Ch0Avg and Ch1Avg by 1024
; First Ch0Avg

    movlw    0x80
    subwf    0x7F & Ch0Avg+2,w           ; 256 + ((Ch0Avg+2) - 0x80) Ch0Avg + 2 first 8 bits of remainder
    btfss    status,c
    goto     CalMode_210                 ; Skip if Ch0Avg+2 >= 0x80
    btfsc    status,c
    goto     CalMode_203                 ; Round down
    btfsc    status,c
    goto     CalMode_203                 ; Is difference 0.
    btfsc    status,c
    goto     CalMode_203                 ; Yes--Round so as to make result even number

CalMode_201
    incf     0x7F & Ch0Avg+1,f
    btfsc    status,z
    incf     0x7F & Ch0Avg,f
    goto     CalMode_210                 ; Round up

CalMode_203
    btfsc    0x7F & Ch0Avg,0
    goto     CalMode_201                 ; Come here to determine whether result odd or even
    btfsc    0x7F & Ch0Avg,0
    goto     CalMode_201                 ; Result is odd so round up

CalMode_210
; Next Ch1Avg

    movlw    0x80
    subwf    0x7F & Ch1Avg+2,w           ; 256 + ((Ch0Avg+2) - 0x80) Ch0Avg + 2 first 8 bits of remainder
    btfss    status,c
    goto     CalMode_220                 ; Skip if Ch0Avg+2 >= 0x80
    btfsc    status,c
    goto     CalMode_211                 ; Round down
    btfsc    status,c
    goto     CalMode_213                 ; Is difference 0.
    btfsc    status,c
    goto     CalMode_213                 ; Yes--Round so as to make result even number

CalMode_211
    incf     0x7F & Ch1Avg+1,f
    btfsc    status,z
    incf     0x7F & Ch1Avg,f
    goto     CalMode_220                 ; Round up

CalMode_213
    btfsc    0x7F & Ch1Avg,0
    goto     CalMode_211                 ; Come here to determine whether result odd or even
    btfsc    0x7F & Ch1Avg,0
    goto     CalMode_211                 ; Result is odd so round up

CalMode_220

```



```

; Now display result on LCD. First Channel0
    bcf          status,rp0
    movlw       Line1+5
    call LCD_SetLineAndColumn
    MoveBytes    Ch0Avg,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    clrf         ASCII_Decimal
    clrf         ASCII_Sign
    movlw       4
    movwf        ASCII_MinField
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " (" ,uadr8771

    bsf          status,rp0
    movlw       6
    movwf        0x7F & B1Temp1
    bcf          status,c
CalMode_230
    rrf          0x7F & Ch0Min,f
    rrf          0x7F & Ch0Min+1,f
    decfsz       0x7F & B1Temp1,f
    goto         CalMode_230

    MoveBytes    Ch0Min,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " ," ,uadr8772

    bsf          status,rp0
    movlw       6
    movwf        0x7F & B1Temp1
    bcf          status,c
CalMode_231
    rrf          0x7F & Ch0Max,f
    rrf          0x7F & Ch0Max+1,f
    decfsz       0x7F & B1Temp1,f
    goto         CalMode_231

    MoveBytes    Ch0Max,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " )" ,uadr8773

; Then Channel 1
    movlw       Line2+5
    call LCD_SetLineAndColumn

    MoveBytes    Ch1Avg,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    clrf         ASCII_Decimal
    clrf         ASCII_Sign
    movlw       4
    movwf        ASCII_MinField
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " (" ,uadr8774

    bsf          status,rp0
    movlw       6
    movwf        0x7F & B1Temp1
    bcf          status,c
CalMode_232
    rrf          0x7F & Ch1Min,f
    rrf          0x7F & Ch1Min+1,f
    decfsz       0x7F & B1Temp1,f
    goto         CalMode_232

    MoveBytes    Ch1Min,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " ," ,uadr8775

    bsf          status,rp0
    movlw       6
    movwf        0x7F & B1Temp1
    bcf          status,c

```

```

CalMode_233
    rrf          0x7F & ChlMax,f
    rrf          0x7F & ChlMax+1,f
    decfsz       0x7F & B1Temp1,f
    goto         CalMode_233

    MoveBytes    ChlMax,aarg+1,2
    bcf          status,rp0
    call         Int16ToASCII_Num
    call         LCD_DisplayASCII_Num
    LCD_DisplayText " )",uadr8776

    goto         StartSeries

endif

if Calibrate == 0

;*****
; This subroutine calculates the SWR from forward and reflected powers
; (or these powers multiplied by a constant) according to the formula

;   SWR = (1 + sqrt(Pr/Pf))/(1 - sqrt(Pr/Pf)).

; This will actually be calculated as follows:
;
;   SWR = (-2/(sqrt(Pr/Pf) - 1) -1
;
; The subroutine is entered with Pr in aarg and Pf in barg.
; The SWR is returned in aarg

CalculateSWRandSetMeter
; Is SWR based on values of peak power too low to display?
; First check forward power
    bsf          status,rp0
    movlw        LowPowerThreshold
    subwf        0x7F & bForward,w    ; bForward - LowPowerThreshold
    bcf          status,rp0
    btfsc        status,c    ; Is <0?
    goto         mm03
    bsf          ModeStatusFlags,LowPeakWatts    ; Power too low
    decfsz       SWRMeterWaitCounter,f    ; Have we waited enough times when power is low?
    return       ; No
    clrw         ; We have waited enough. Set SWR meter to 0
    call         PWM_SetDutyCycle
    return

mm03
    bcf          ModeStatusFlags,LowPeakWatts    ; Power not too low
    movlw        MaxNumSWRMeterWaits    ; Initialize meter wait counter
    movwf        SWRMeterWaitCounter

    bsf          PCLATH,psel0
    call         0x7ff & FPD24    ; Yes, calculate sqrt(Pr/Pf) = rho
    MoveBytes    aarg,LoadFactor,3    ; LoadFactor = rho^2
    call         0x7ff & sqrt    ; Calculate rho
    StoreFPLiteral 0x7f0000,barg    ; Load 1.0
    call         0x7ff & FPS24    ; rho - 1
    MoveBytes    aarg,barg,3
    StoreFPLiteral 0x808000,aarg    ; Store -2
    call         0x7ff & FPD24    ; Calculate -2/(rho - 1)
    StoreFPLiteral 0x7f8000,barg    ; Load -1
    call         0x7ff & FPA24    ; This is the SWR
    bcf          PCLATH,psel0
    andlw        0xff    ; is w = 0 ?
    btfss        status,z
    goto         mm01    ; w != 0: Divide by 0 so SWR large
    btfss        aarg+1,7    ; Is SWR negative? If so, SWR large
    goto         mm02

mm01
    StoreFPLiteral 0x887A00,aarg    ; Set SWR to 1000
mm02

```

```

MoveBytes aarg,SWR,3

; This next section takes a 24-bit floating point SWR value, in aarg,
; and displays it on the panel meter.
; The meter reading will be 256*(1 - 1/SWR), where
; 0 is at the bottom of the scale and 256 is full scale

MoveBytes aarg,barg,3
movlw 0x7f ; Load 1.0
movwf aarg
clrf aarg+1
clrf aarg+2
bsf PCLATH,psel0
call 0x7ff & FPD24 ; Result =1/SWR
movlw 0x7f ; Load 1.0
movwf barg
clrf barg+1
clrf barg+2
call 0x7ff & FPS24 ; This is 1/SWR - 1. It should always be negative. If it is positive, then SWR = 1
bcf PCLATH,psel0
btfsc aarg+1,7 ; Skip if value positive.
goto mm10
clrw ; Set meter to 0, indicating 1:1 SWR
call PWM_SetDutyCycle
return
mm10
bcf aarg+1,7 ; Result = 1 - 1/SWR
movlw 8 ; Multiply by 256
addwf aarg,f
bsf PCLATH,psel0
call 0x7ff & INT2416
bcf PCLATH,psel0
movlw 0xff
andwf aargb0,f ; Test to see if result > 255
btfsc status,z ; Skip if too large
movf aargb1,w ; Otherwise load actual value
call PWM_SetDutyCycle
return

;*****
; This subroutine writes the peak forward power and SWR to the LCD

DisplayPeakPowerAndSWR
; First determine if power is large enough to display
btfss ModeStatusFlags,LowPeakWatts ; Is power too low?
goto t001
; Power too low
movlw Line1 + 8
call LCD_SetLineAndColumn
LCD_DisplayText " Low ", uAdr5
movlw Line2 + 16
call LCD_SetLineAndColumn
LCD_DisplayText " ", uAdr5a
return

; Power large enough to display

t001
MoveBytes PeakForwardWatts,aarg,3
btfsc ModeStatusFlags,ForwardWatts ;Display Forward or Load Watts
goto t002
; Calculate watts delivered to load
StoreFPLiteral 0x7F0000,aarg ; Load 1.0
MoveBytes LoadFactor,barg,3 ; Load rho^2
bsf PCLATH,psel0
call 0x7ff & FPS24 ; = 1 - rho^2
MoveBytes PeakForwardWatts,barg,3
call 0x7ff & FPM24 ; Load power = Pf*(1 - rho^2)
bcf PCLATH,psel0

; Display power in watts
t002
call FormatWatts
movlw 4
movwf ASCII_MinField
movlw Line1 + 8
call LCD_SetLineAndColumn
call LCD_DisplayASCII_Num
btfsc ModeStatusFlags,MilliWattsFlag
goto t003 ; Units are milliwatts

```

```

        LCD_DisplayText " W ", uAdr57          ; or units are watts
        goto          t004
t003
        nop
        LCD_DisplayText " mW",uAdr7891
t004
; Display SWR
        call          FormatSWR
        movlw         5
        movwf         ASCII_MinField
        movlw         Line2 + 16
        call          LCD_SetLineAndColumn
        call          LCD_DisplayASCII_Num
        return

;*****
; This subroutine writes the average forward or load power to the LCD. Enter with
; the forward power in aarg.

DisplayAvgPower
; First determine if power is large enough to display
        bcf           status,rp0
        btfs          ModeStatusFlags,LowPeakWatts
        goto          t101
; Power too low
        movlw         Line2 + 8
        call          LCD_SetLineAndColumn
        LCD_DisplayText " Low      ", uAdr67
        return

; Power large enough to display

t101
        btfs          ModeStatusFlags,ForwardWatts      ;Display forward or load power
        goto          t102

; Calculate load power
        MoveBytes     aarg,t1,3                        ; Temporary storage of average power
        StoreFPLiteral 0x7F0000,aarg                  ; Load 1.0
        MoveBytes     LoadFactor,barg,3; Load rho^2
        bsf           PCLATH,psel0
        call          0x7ff & FPS24                    ; = 1 - rho^2
        MoveBytes     t1,barg,3
        call          0x7ff & FPM24                    ; Load power = Pf*(1 - rho^2)
        bcf           PCLATH,psel0

t102
        call          FormatWatts
        movlw         4
        movwf         ASCII_MinField
        movlw         Line2 + 8
        call          LCD_SetLineAndColumn
        call          LCD_DisplayASCII_Num
        btfs          ModeStatusFlags,MilliWattsFlag    ; Units are W or mW?
        goto          t103
        LCD_DisplayText " W ", uAdr76
        return
t103
        LCD_DisplayText " mW", uAdr761145
        return

;*****
; This subroutine sets up Watts data for output

FormatWatts
; Determine scale of power.
        clrf          ASCII_Decimal
        movlw         5
        movwf         temp2
        bcf           ModeStatusFlags,MilliWattsFlag
        StoreFPLiteral 0x844800,barg      ; Load value 20

FW_02
        bsf           PCLATH,psel0
        call          0x7ff & TAGEB24      ; Is 10^(4 - count)*watts >= 20?
        bcf           PCLATH,psel0
        andlw         0xff

```

```

    btfss    status,z
    goto     FW_01
    bsf      PCLATH,psel0          ; No, < 20
    call     0x7ff & AargTimes10
    bcf      PCLATH,psel0
    incf     ASCII_Decimal,f
    movlw    3
    subwf    ASCII_Decimal,w
    btfss    status,z
    goto     FW_03
    clrf     ASCII_Decimal
    bsf      ModeStatusFlags,MilliWattsFlag
FW_03
    decfsz   temp2,f
    goto     FW_02

FW_01
    clrf     ASCII_Sign
    bsf      PCLATH,psel0
    StoreFPLiteral 0x7E0000,barg
    call     0x7ff & FPA24
    call     0x7ff & INT2416          ; Integer floor
    pagesel  Int16ToASCII_Num
    call     Int16ToASCII_Num
    return

;*****
; This routine formats SWR data for output

FormatSWR
    MoveBytes SWR,aarg,3
    clrf     ASCII_Sign
    StoreFPLiteral 0x832000,barg          ; Load 20
    pagesel  TAGEB24
    call     0x7ff & TAGEB24
    pagesel  SWRge20
    andlw    0xff          ; Determine if w = 0
    btfss    status,z      ; Skip if w != 0, i.e., SWR <20
    goto     SWRge20       ; Comes here if SWR >= 20
    StoreFPLiteral 0x822000,barg          ; Load 10
    pagesel  TAGEB24
    call     0x7ff & TAGEB24
    pagesel  SWRge10
    andlw    0xff          ; Determine if w = 0
    btfss    status,z      ; Skip if w != 0, i.e., SWR <20
    goto     SWRge10       ; Comes here if SWR >= 20

SWRlt10
    pagesel  AargTimes10
    call     0x7ff & AargTimes10
    call     0x7ff & AargTimes10
    movlw    2
    movwf    ASCII_Decimal
    pagesel  ss001
    goto     ss001

SWRge10
    pagesel  AargTimes10
    call     0x7ff & AargTimes10
    movlw    1
    movwf    ASCII_Decimal

ss001
    StoreFPLiteral 0x7e0000,barg          ; Load 0.5
    pagesel  FPA24
    call     0x7ff & FPA24
    call     0x7ff & INT2416
    pagesel  Int16ToASCII_Num
    call     Int16ToASCII_Num
    return

SWRge20
    movlw    0x30
    movwf    ASCII_Num
    movwf    ASCII_Num + 1
    movwf    ASCII_Num + 4
    movlw    0x3e
    movwf    ASCII_Num + 2
    movlw    0x32
    movwf    ASCII_Num + 3
    clrf     ASCII_Decimal

```

```

    return

endif

;*****
; This routine Displays an ASCII Number, in ASCII_Num through ASCII_Num+5
; on the LCD.
; ASCII_Sign = 1 if number negative, 0 if positive
; ASCII_Digits gives the location of the decimal point (i.e., #
; of digits to right of decimal point)
; ASCII_MinField gives the minimum field width in which the number will
; be displayed. The number will always be right justified in this field.

LCD_DisplayASCII_Num
    bcf status,rp0
    clrf        Temp1        ; Flag for leading decimal point
    movlw       5            ; 5 characters in ASCII_Num
    movwf       Temp2
    movlw       ASCII_Num    ; Pointer to first character
    movwf       FSR
s001 movf       Temp2,w      ; Is next character decimal?
    subwf       ASCII_Decimal,w
    btfsc       status,z ; Skip if it is decimal point
    goto        s002
    movlw       0x30        ; Otherwise check to see if leading 0
    subwf       INDF,w
    btfss       status,z ; Skip if it is leading 0
    goto        s002+2
    incf        FSR,f
    decfsz      Temp2,f      ; Temp2 should never be 0. If so
    goto        s001        ; nothing to print so simply print out a 0

    decf        ASCII_MinField,w
    movwf       Temp1
s007 movlw       0x20        ; Print out leading blanks
    call        LCD_SendChar
    decfsz      Temp1,f
    goto        s007
    movlw       0x30        ; Print out single 0
    call        LCD_SendChar
    return

s002 bsf Temp1,1            ; Yes, decimal point is leading
    incf        Temp2,f
    movlw       0xff        ; Is there a decimal point at all?
    andwf       ASCII_Decimal,w
    btfss       status,z ; Skip if there is no decimal point
    incf        Temp2,f      ; Provide space for decimal
    movlw       0xff        ; Is there a minus sign?
    andwf       ASCII_Sign,w
    btfss       status,z ; Skip if no minus sign
    incf        Temp2,f      ; Provide space for minus sign

; Now output actual characters
; First output spaces, if any, needed to fill up field
    movf        Temp2,w      ; Load total actual field size
    subwf       ASCII_MinField,w ; Subtract from minimum field size
    btfss       status,c
    goto s006
    btfsc       status,z
    goto s006
    movwf       Temp2        ; Output spaces to fill field
    movlw       0x20
    call        LCD_SendChar
    decfsz      Temp2,f
    goto        $-3

; Determine # of characters excluding sign and decimal point
s006 movf       FSR,w
    sublw       ASCII_Num+5
    movwf       Temp2
    btfss       ASCII_Sign,0 ; Skip if number negative
    goto        $+3
    movlw       0x2d        ; Display minus sign
    call        LCD_SendChar
    btfss       Temp1,1      ; Place 0 in front of leading decimal?
    goto        $+3
    movlw       0x30

```

```

    call    LCD_SendChar
s05 movf    ASCII_Decimal,w ; Display decimal point?
    subwf   Temp2,w
    btfss   status,z ; Skip if place for decimal point
    goto    s03 ;
    movlw   0x2e ; Load decimal point
    call    LCD_SendChar
s03 movf    INDF,w ; Load character
    call    LCD_SendChar ; Display character
s04 incf    FSR,f ; Increment pointer to next character
    decfsz  temp2,f ; Decrement character counter
    goto    s05
    return

;*****
;*****
; These routines control the PWM1 module.
;
; PWM_Initialize initializes the PWM1 module for this purpose
; PWM_SetDutyCycle sets the duty cycle for the PWM1 module.
;

; Setup PWM to have a pulse frequency of 4.88 kHz and a duty cycle resolution of 1 part in 256

PWM_Initialize
    movlw   0x3f ; Will count up to 64
    bsf     status,rp0
    movwf   PR2 & 0x7f
    bcf     status,rp0
    clrf    CCP1RL ; Start with duty cycle = 0
    bsf     status,rp0
    bcf     (TRISC & 0x7f),2 ; Set RC2 as output
    bcf     status,rp0
    movlw   0x06 ; 1:16 prescale, 1:1 postscale, turn counter on
    movwf   T2CON
    movlw   0x0C ; Set PWM mode.
    movwf   CCP1CON
    return

;*****
; This subroutine sets the 8-bit duty cycle of PWM1.
; Enter with the 8-bit duty cycle in the w register

PWM_SetDutyCycle
    bcf     status,rp0
    movwf   temp ; Store duty cycle in temp
r111
    movf    TMR2,w ; Determine how close to the end of the
    bsf     status,rp0
    subwf   PR2 & 0x7f,w ; Form PR2 - TMR2
    bcf     status,rp0
    andlw   0xf0 ; This will be nonzero if PR2 - TMR2 > 15
    btfsc   status,z ; If zero, too close, wait for overflow
    goto    r111
    bcf     CCP1CON,ccplx ; Update LSB's in CCP1CON. First 0 them
    bcf     CCP1CON,ccply
    rrf     temp,f ; Rotate LSB of duty cycle into C
    btfsc   status,c ; If 1, set the ccply bit
    bsf     CCP1CON,ccply
    rrf     temp,f ; Rotate next LSB into C
    btfsc   status,c ; If 1, set the ccplx bit
    bsf     CCP1CON,ccplx
    movf    temp,w
    andlw   0x3f
    movwf   ccpr1l ; Place in ccpr1l register
    return

;*****
;*****
; These routines implement an 8 bit interface to a Hitachi
; LCD module, busy flag used when valid. The data lines
; are on port B, E is on port_c bit6,
; R/W is on port_c bit 5, RS is on port_c bit 4.

LCDDATAPORT EQU PORTB ;LCD data port

```

```

LCDDataTRIS EQU    TRISB    ;LCD data TRIS register
LCDControlPORT EQU    PORTC    ;LCD control port
LCDControlTRIS EQU    TRISC    ;LCD Control TRIS register

E EQU    6    ;LCD enable signal
RW EQU    5    ;LCD R/W signal
RS EQU    4    ;LCD register select

;* LCD_SendChar - Sends character contained in register W to LCD *

LCD_SendChar
    bcf        status,rp0
    movwf     LCDChar    ;Character to be sent is in W
    call      LCD_BusyCheck    ;Wait for LCD to be ready
    movf      LCDChar,w
    movwf     LCDDataPORT    ;Send data to LCD
    bcf       LCDControlPORT,RW    ;Set LCD in read mode
    bsf       LCDControlPORT,RS    ;Set LCD in data mode
    nop
    bsf       LCDControlPORT,E    ;toggle E for LCD
    nop
    bcf       LCDControlPORT,E
    return

;*****
;* LCD_SendCmd - Sends command contained in register W to LCD *

LCD_SendCmd
    bcf        status,rp0
    movwf     LCDChar    ;Command to be sent is in W
    call      LCD_BusyCheck    ;Wait for LCD to be ready
    movf      LCDChar,w
    movwf     LCDDataPORT    ;Send data to LCD
    bcf       LCDControlPORT,RW    ;Set LCD in read mode
    bcf       LCDControlPORT,RS    ;Set LCD in command mode
    nop
    bsf       LCDControlPORT,E    ;toggle E for LCD
    nop
    bcf       LCDControlPORT,E
    return

;*****
;* This routine sets the DDRAM address to the line and column # contained in w.
; Suppose it is desired to place the "cursor" at Line 2, column 7. The following
; code will do this:
;
;   movlw     Line2 + 7
;   call      LCD_SetLineAndColumn

Line1 equ    0x7f
Line2 equ    0xbf

LCD_SetLineAndColumn
    call LCD_SendCmd
    return

;*****
; This routine clears display, moves cursor to beginning of line 1

LCD_ClearDisplay
    movlw     0x01
    call      LCD_SendCmd
    return

;*****
;* This routine checks the busy flag, returns when not busy *
;* Affects: *
;* LCDTemp - Returned with busy/address *

LCD_BusyCheck
    if Simulate==0
        ;Set Data for input
        bsf     STATUS,RP0
        movlw   0xFF

```



```

    movwf    0x7f&LCDDDataTRIS
    bcf      STATUS,RP0
    bcf      LCDControlPORT,RS      ;Set LCD for command mode
    bsf      LCDControlPORT,RW      ;Setup to read busy flag
LCDBS001
    bsf      LCDControlPORT,E      ;Set E high
    nop
    movf     LCDDDataPORT,W        ;Read busy flag, DDram address
    bcf      LCDControlPORT,E      ;Set E low
    andlw    0x80                  ; Test Bit 7, the busy flag
    btfss    status,z
    goto     LCDBS001

    bcf      LCDControlPORT,RW
    bsf      STATUS,RP0            ;Set Data for output
    clrf     0x7f & LCDDDataTRIS
    bcf      STATUS,RP0
    return
else
    return
endif

;*****
;* This routine initializes the LCD module          *
;* Affects:                                         *
;*   LCDTemp - Returned with busy/address          *
;*****

LCD_Init
    bcf      status,rp0
    movlw    20
    call     WaitMilliseconds      ; Wait for 20 ms

    bsf      STATUS,RP0            ;---Bank 1
    clrf     0x7f&LCDDDataTRIS    ; Set all 8 data pins as outputs
    bcf      0x7f&LCDControlTRIS,E ; Set three pins as outputs--Others inputs
    bcf      0x7f&LCDControlTRIS,RW
    bcf      0x7f&LCDControlTRIS,RS
    bcf      STATUS,RP0            ;---Bank 0

    clrf     LCDDDataPORT
    bcf      LCDControlPORT,E
    bcf      LCDControlPORT,RW
    bcf      LCDControlPORT,RS

    movlw    0x38                  ;Set LCD to 8 bit interface
    movwf    LCDDDataPORT
    bsf      LCDControlPORT,E      ;toggle E for LCD
    nop
    bcf      LCDControlPORT,E
    movlw    6
    call     WaitMilliseconds      ; Wait for 6 ms

    movlw    0x38                  ;Function set to 2 lines
    movwf    LCDDDataPORT          ;of 5x7 bit chars
    bsf      LCDControlPORT,E      ;toggle E for LCD
    nop
    bcf      LCDControlPORT,E
    movlw    1
    call     WaitMilliseconds      ; Wait for 1 msec

;Busy flag should be valid after this point
    movlw    0x38
    call     LCD_SendCmd

    movlw    0x06                  ;Increment cursor, no shift
    call     LCD_SendCmd

    movlw    0x0C                  ;Turn display, cursor off
    call     LCD_SendCmd

    movlw    0x01                  ;Clear display, cursor to begin
    call     LCD_SendCmd          ; of line 1

    return

;*****
;*****

```

```

; This routine takes a 16 bit integer (always positive) located in
; aargb0 (most significant 8 bits) and aargb1 and produces a six character
; ASCII string that expresses the number in decimal characters. The result is placed
; in ASCII_Num, ASCII_Num+1, ... , ASCII_Num+4.
;

```

```

Int16ToASCII_Num
    bcf status,rp0      ; Data in bank 0
    movlw 0x27          ; Load integer 10000 into bargb0, 1
    movwf bargb0
    movlw 0x10
    movwf bargb1
    call DivideDown     ; Subtract 10,000 repeatedly until
    movwf ASCII_Num     ; total less than 10,000. # of subtractions
                        ; is 10000's digit. STore in ASCII_num

    movlw 0x03          ; load 1000
    movwf bargb0
    movlw 0xe8
    movwf bargb1
    call DivideDown
    movwf ASCII_Num + 1

    movlw 0x00          ; load 100
    movwf bargb0
    movlw 0x64
    movwf bargb1
    call DivideDown
    movwf ASCII_Num + 2

    movlw 0x00          ; load 10
    movwf bargb0
    movlw 0x0a
    movwf bargb1
    call DivideDown
    movwf ASCII_Num + 3

    movf aargb1,w       ; Remainder in aargb1 is units digit
    addlw 0x30          ; Add 0x30 to convert to ASCII
    movwf ASCII_Num + 4
    return

```

```

;*****
; This routine repeatedly subtracts bargb0:b1 from aargb0:b1 until result is no longer positive

```

```

DivideDown
    clrf temp1          ; Counts # of succesful subtractions
r11 call Int16Sub       ; subtract bargb0:b1 from aargb0:b1
    andlw 0xff          ; Check is result < 0
    btfss status,z
    goto dl             ; Yes, result was negative
    incf temp1,f        ; No, result still positive
    goto r11            ; Subtract again
dl  movf temp1,w        ; Last subtraction resulted in - result
                        ; aargb0:b1 restored to last + value
    addlw 0x30          ; Add 0x30 to count to convert to ASCII
    return

```

```

;*****
; This routine does a 16 bit integer subtraction if the result is positive

```

```

Int16Sub
    MoveBytes aargb0,t1,2 ; Save aargb0:b1 in case result of subtraction is -
    movf bargb1,w        ; Subtract least significant 8 bits
    subwf aargb1,f        ; Result goes in aargb1
    btfsc status,c       ; Was a borrow necessary?
    goto r1              ; No, jump to r1
    movlw 1               ; Yes. Subtract 1 from aargb0
    subwf aargb0,f        ; Place result back in aargb0
    btfsc status,c       ; If borrow necessary, overall result negative.
    goto r1              ; No borrow
    MoveBytes t1,aargb0,2 ; Borrow necessary-result negative-restore aargb0:b1
    retlw 0xff           ; Return with w=0xff: result was negative
r1  movf bargb0,w        ; Subtract most significant 8 bits
    subwf aargb0,f        ; Place result back in aargb0
    btfsc status,c       ; Was borrow necessary?
    retlw 0x00           ; No: result was positive, return
    MoveBytes t1,aargb0,2 ; Yes, restore aargb0:b1 to previous value
    retlw 0xff           ; RReturn with w=0xff; result was negative

```

```

;*****
;
; This routine sets a time delay in units of microseconds.
; Enter with the number of units to wait in the w register.
; The mininum wait is 3 microseconds.

```

```

WaitMicroSeconds
if Simulate>0
    return
endif
    banksel    0
    movwf     Timer
    decf      Timer,f
    decf      Timer,f
    nop
    nop
    nop
    nop
    decfsz    Timer,f
    goto      $-3
    return

```

```

;*****
; This routine waits for units of milliseconds.
; Enter with the number of units to wait in w register.
; The minimum wait is 1 millisecond

```

```

WaitMilliSeconds
if Simulate>0
    return
endif
    banksel    0
    movwf     Timer2
    decfsz    Timer2,f
    goto      WMS1
    goto      WMS2
WMS1
    movlw     250
    Call      WaitMicroSeconds
    movlw     250
    Call      WaitMicroSeconds
    movlw     250
    Call      WaitMicroSeconds
    movlw     248
    Call      WaitMicroSeconds
    nop
    nop
    nop
    decfsz    Timer2,f
    goto      WMS1
    nop
    nop
WMS2
    movlw     250
    Call      WaitMicroSeconds
    movlw     250
    Call      WaitMicroSeconds
    movlw     250
    Call      WaitMicroSeconds
    movlw     247
    Call      WaitMicroSeconds
    return

```

```

;*****
; This routine waits for units of tenths of seconds.
; Enter with the number of units to wait in w register.
; The minimum wait is 0.1 second.

```

```

WaitTenthSeconds
if Simulate>0
    return
endif
    banksel    0

```

```

        movwf    Timer3
        decfsz   Timer3,f
        goto     WTS1
        goto     WTS2
WTS1:
        movlw    99
        call     WaitMilliSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    248
        call     WaitMicroSeconds
        nop
        nop
        decfsz   Timer3,f
        goto     WTS1
        nop
        nop
WTS2:
        movlw    99
        call     WaitMilliSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    250
        call     WaitMicroSeconds
        movlw    246
        call     WaitMicroSeconds
        nop
        nop
        nop
        nop
        return

;***** GetForwardWatts *****
; This routine uses a lookup table to translate 10-bit binary ADC values, located in bForwardNew, into packed-16 values
; of the forward power. The result is returned in FP and FP+1
; The actually lookup table takes all of page 2 of memory.

GetForwardWatts
    bsf        status,rp0
    call       _GetForwardLowByte
    movwf     (0x7F & FP)+1
    bcf        PCLATH,psell
    call       _GetForwardHighByte
    movwf     (0x7F & FP)
    return

_GetForwardLowByte
    movlw     0x10                ; Lookup table begins at beginning of page 2.
    movwf     PCLATH
    btfsc     0x7F & bForwardNew+1,6
    bsf       PCLATH,0
    btfsc     0x7F & bForwardNew+1,7
    bsf       PCLATH,1
    movf      0x7F & bForwardNew,w
    movwf     PCL

_GetForwardHighByte
    movlw     0x14                ; Lookup table begins at beginning of page 2.
    movwf     PCLATH
    btfsc     0x7F & bForwardNew+1,6
    bsf       PCLATH,0
    btfsc     0x7F & bForwardNew+1,7
    bsf       PCLATH,1
    movf      0x7F & bForwardNew,w
    movwf     PCL

;***** GetReflectedWatts *****
; This routine uses a lookup table to translate 10-bit binary ADC values into packed-16 values of the reflected power.
; The actual lookup table takes all of page 3 of memory.

```

```

GetReflectedWatts
    bsf        status,rp0
    call       _GetReflectedLowByte
    movwf     (0x7F & FP)+1
    bcf        PCLATH,psel0
    bcf        PCLATH,psel1
    call       _GetReflectedHighByte
    movwf     (0x7F & FP)
    return

_GetReflectedLowByte
    movlw     0x18                ; Lookup table for reflected power begins at beginning of page 3.
    movwf     PCLATH
    btfsc     0x7F & bReflected+1,6
    bsf        PCLATH,0
    btfsc     0x7F & bReflected+1,7
    bsf        PCLATH,1
    movf      0x7F & bReflected,w
    movwf     PCL

_GetReflectedHighByte
    movlw     0x1C                ; Lookup table for reflected power begins at beginning of page 3.
    movwf     PCLATH
    btfsc     0x7F & bReflected+1,6
    bsf        PCLATH,0
    btfsc     0x7F & bReflected+1,7
    bsf        PCLATH,1
    movf      0x7F & bReflected,w
    movwf     PCL

if Calibrate != 1

;*****
;   Locate floating point routines in page 1

    org      0x0800

; ***** SumP16 *****
;
; This routine adds the packed 16-bit value in FP (Bank 1) to the packed 32-bit running sum in FPaccum (Bank 1). The result is
; returned
; in FPaccum.
; My packed format works as follows: All numbers are positive. The first six bits are the exponent, which is offset by
; 27. An exponent of 0 means the value is zero. The mantissa is the remaining 10 bits, where the leading 1 is suppressed.

SumP16
    bsf        status,rp0
    movf      (0x7F & FPaccum),w                ; Calculate difference of exponents and
    andlw     0xFC                                ; determine which exponent is larger
    movwf     0x7F & B1Temp1
    movf      0x7F & FP,w
    andlw     0xFC
    subwf     0x7F & B1Temp1,w                ; 4*(Exp(FPaccum) - Exp(FP))
    btfsc     status,c
    goto      SumP16_001                    ; FPaccum has larger exponent.

; Come here if Exp(FP) > Exp(FPaccum). We always want the opposite, so switch the two values.

    sublw     0                                ; Calculate negative of exponent difference
    movwf     0x7F & B1Temp3                ; B1Temp1 = 4*(Exp(FPaccum) - Exp(FP))
    movf      0x7F & FP,w                    ; Now interchange FP and FPaccum
    movwf     0x7F & B1Temp1
    movf      0x7F & FP+1,w
    movwf     0x7F & B1Temp2
    movf      0x7F & FPaccum,w
    movwf     0x7F & FP
    movf      0x7F & FPaccum+1,w
    movwf     0x7F & FP+1
    movf      0x7F & FPaccum+2,w

```

```

movwf    0x7F & FP+2
movf     0x7F & FPaccum+3,w
movwf    0x7F & FP+3
movf     0x7F & B1Temp1,w
movwf    0x7F & FPaccum
movf     0x7F & B1Temp2,w
movwf    0x7F & FPaccum+1
clrf     0x7F & FPaccum+2
clrf     0x7F & FPaccum+3
movf     0x7F & FPaccum,w          ; Now put larger exponent * 4 into B1Temp1
andlw    0xFC
movwf    0x7F & B1Temp1
goto     SumP16_001A

SumP16_001
movwf    (0x7F & B1Temp3)          ; Store exponent difference.
clrf     (0x7F & FP)+2
clrf     (0x7F & FP)+3

SumP16_001A
movf     0x7F & FP,w              ; Is FP 0? If so return with FPaccum unchanged
andlw    0xFC
btfsc    status,z
return

bcf       status,c                ; B1Temp3 = 4(Exp(FPaccum) - Exp(FP)). Divide by 4
rrf      (0x7F & B1Temp3),f
rrf      (0x7F & B1Temp3),f          ; Two right shifts and B1Temp3 = Exp(FPaccum) - Exp(FP)

movlw    28                      ; If a shift of 28 or more bits required, FP is too small to affect FPaccum
subwf    (0x7F & B1Temp3),w
btfsc    status,c
return

movlw    0x3                    ; Extract 10-bit mantissa and add leading implied 1's to FPaccum and FP
andwf    (0x7F & FP),f
bsf      (0x7F & FP),2
andwf    (0x7F & FPaccum),f
bsf      (0x7F & FPaccum),2

movlw    24                      ; If a shift of 24 or more bits required, shift bytes three places to right
subwf    (0x7F & B1Temp3),w
btfss    status,c
goto     SumP16_002
movwf    0x7F & B1Temp3
rlf      (0x7F & FP)+1,f          ; This bit becomes the round-off bit
movf     (0x7F & FP),w          ; Shift three bytes over
movwf    (0x7F & FP)+3
clrf     (0x7F & FP)+2
clrf     (0x7F & FP)+1
clrf     (0x7F & FP)
goto     SumP16_005

SumP16_002
btfss    (0x7F & B1Temp3),4      ; If a shift of 16-23 bits required, just move bytes
goto     SumP16_003
rlf      0x7F & FP+2,f          ; This bit becomes round-off bit
movf     (0x7F & FP)+1,w        ; Shift two bytes over
movwf    (0x7F & FP)+3
movf     (0x7F & FP),w
movwf    (0x7F & FP)+2
clrf     (0x7F & FP)+1
clrf     (0x7F & FP)
bcf      (0x7F & B1Temp3),4
goto     SumP16_005

SumP16_003
btfss    (0x7F & B1Temp3),3      ; If a shift of 8-15 bits required, just move bytes
goto     SumP16_005
rlf      0x7F & FP+3,f
movf     0x7F & FP+2,w
movwf    0x7F & FP+3
movf     (0x7F & FP)+1,w
movwf    (0x7F & FP)+2
movf     (0x7F & FP),w
movwf    (0x7F & FP)+1
clrf     (0x7F & FP)
bcf      (0x7F & B1Temp3),3

SumP16_005

```

```

    movf      (0x7F & B1Temp3),w          ; Any more bits to shift?
    btfsc     status,z
    goto      SumP16_007                  ; no
SumP16_006
    bcf       status,c                    ; Shift FP to right
    rrf       (0x7F & FP),f
    rrf       (0x7F & FP)+1,f
    rrf       (0x7F & FP)+2,f
    rrf       (0x7F & FP)+3,f
    decfsz    (0x7F & B1Temp3),f
    goto      SumP16_006

SumP16_007
    btfss     status,c                    ; Do we round up or down?
    goto      SumP16_100                  ; Round down
    incf      (0x7F & FP)+3,f
    btfss     status,z
    goto      SumP16_100
    incf      (0x7F & FP)+2,f
    btfss     status,z
    goto      SumP16_100
    incf      (0x7F & FP)+1,f
    btfsc     status,z
    incf      (0x7F & FP),f

SumP16_100
    movf      (0x7F & FP)+3,w              ; Two values now have same exponent so can be directly added.
    addwf     (0x7F & FPaccum)+3,f         ; Add fourth bytes
    btfss     status,c                    ; Is there a carry? If so, ripple carry through remaining bytes.
    goto      SumP16_101
    incf      (0x7F & FPaccum)+2,f
    btfss     status,z
    goto      SumP16_101
    incf      (0x7F & FPaccum)+1,f
    btfsc     status,z
    incf      (0x7F & FPaccum),f

SumP16_101
    movf      (0x7F & FP)+2,w              ; Add Third bytes, and handle carry if necessary.
    addwf     (0x7F & FPaccum)+2,f
    btfss     status,c
    goto      SumP16_102
    incf      (0x7F & FPaccum)+1,f
    btfsc     status,z
    incf      (0x7F & FPaccum),f

SumP16_102
    movf      (0x7F & FP)+1,w              ; Add second bytes, and handle carry if necessary.
    addwf     (0x7F & FPaccum)+1,f
    btfsc     status,c
    incf      (0x7F & FPaccum),f

    movf      (0x7F & FP),w                ; Add first (most significant) bytes
    addwf     (0x7F & FPaccum),f

    btfss     (0x7F & FPaccum),3           ; Has there been a carry to the fourth bit of the most significant byte?
    goto      SumP16_200

SumP16_105
    movlw     0x4                          ; Yes there has been a carry. Renormalize result by shifting to right
    addwf     (0x7F & B1Temp1),f          ; and incrementing the exponent by 1. First do exponent.
    bcf       status,c
    rrf       (0x7F & FPaccum),f
    rrf       (0x7F & FPaccum)+1,f
    rrf       (0x7F & FPaccum)+2,f
    rrf       (0x7F & FPaccum)+3,f
    btfss     status,c                    ; If carry, must round result up.
    goto      SumP16_200
    incf      (0x7F & FPaccum)+3,f
    btfss     status,z
    goto      SumP16_200
    incf      (0x7F & FPaccum)+2,f
    btfss     status,z
    goto      SumP16_200
    incf      (0x7F & FPaccum)+1,f
    btfsc     status,z
    incf      (0x7F & FPaccum),f
    btfsc     (0x7F & FPaccum),3           ; Has there been a carry to the fourth bit of the most significant byte?
    goto      SumP16_105                  ; Yes, so normalize again.

```

```

SumP16_200
    bcf      (0x7F & FPaccum),2      ; Clear the leading bit which is always 1.
    movf     (0x7F & B1Temp1),w      ; Add exponent to leading byte
    addwf    (0x7F & FPaccum),f
    return

;***** P16ToFP24 *****
; This routine converts a packed-16 floating point value in FP (Bank1) to a FP24 value in AARG

P16ToFP24
    bsf      status,rp0
    movf     0x7F & FP,w             ; Get exponent of packed value
    andlw    0xFC
    btfss    status,z               ; Is exponent, and value, 0?
    goto     $+6
    bcf      status,rp0
    clrf     AARG                   ; Set FP24 value to 0
    clrf     AARG+1
    clrf     AARG+2
    return
    bcf      status,rp0
    movwf    AARG
    bcf      status,c
    rrf      AARG,f                 ; Shift value two bits to right to occupy all of AARG
    rrf      AARG,f
    movlw    100                    ; Now change offset from 27 to 127
    addwf    AARG,f
    bsf      status,rp0             ; Shift packed mantissa 3 bits to right and place in AARG
    rrf      0x7F & FP,f
    rrf      0x7F & FP+1,f
    rrf      0x7F & FP+2,f
    rrf      0x7F & FP,f
    rrf      0x7F & FP+1,f
    rrf      0x7F & FP+2,f
    bcf      status,c
    rrf      0x7F & FP+1,w
    bcf      status,rp0
    movwf    AARG+1
    bsf      status,rp0
    rrf      0x7F & FP+2,w
    bcf      status,rp0
    movwf    AARG+2
    btfss    status,c               ; Round up or down?
    return
    incf     AARG+2,f               ; Round up
    btfss    status,z
    return
    incf     AARG+1,f
    btfss    AARG+1,7
    return
    incf     AARG,f
    clrf     AARG+1
    return

;*****
; This routine estimates the square root of the argument x. The argument
; is placed in aarg, and the result is returned in the same register. If the
; result is valid, w will be returned with 0 in it. Otherwise (i.e., argument
; is negative), w is returned containing 0xff.
; The accuracy of the routine is better than 0.2%.

sqrt
    btfsc    aarg+1,7 ; Is argument positive?
    retlw    0xff      ; No--return with 0xff in w
    movlw    0x7f      ; Subtract offset from exponent
    subwf    aarg,w

```



```

movwf      templ
bcf status,c ; Prepare to do arithmetic right shift of templ
btfsc      templ,7 ; temp 1 positive?
bsf status,c ; No--right shift in 1
rrf templ,f ; Divide exponent by 2
bcf status,c
rlf templ,w ; Multiply by 2--Result always even
subwf      aarg,f ; Normalize argument to be >=1 and <4
MoveBytes aarg,t1,3 ; Temporary save of normalized argument
; Next estimate square root using linear approximation
StoreFPLiteral 0x7d37f6,barg ; Load 0.35930, slope of line
call        FPM24 ; Multiplay by normalized x
StoreFPLiteral 0x7e2405,barg ; Load 0.64070, intercept
call        FPA24
; Now improve estimate as 0.5*(sqrt + x/sqrt)
MoveBytes aarg,t2,3 ; Save estimate
MoveBytes aarg,barg,3
MoveBytes t1,aarg,3 ; Load normalized x
call        FPD24
MoveBytes t2,barg,3
call        FPA24
decf        aarg,f ; Divide by 2.
movf        templ,w ; Now multiply by power of 2
addwf       aarg,f
retlw       0

;*****
;*****

; PIC16 24 BIT FLOATING POINT LIBRARY
;
; Place library at beginning of Page 1 of memory

;*****
;*****

; Integer to float conversion
; Input: 16 bit 2's complement integer right justified in AARGB0, AARGB1
; Use: CALL FLO1624 or CALL FLO24
; Output: 24 bit floating point number in AEXP, AARGB0, AARGB1
; Result: AARG <-- FLOAT( AARG )
; Max Timing: 11+72 = 83 clks SAT = 0
;              11+77 = 88 clks SAT = 1

; Min Timing: 7+14 = 21 clks AARG = 0
;              7+18 = 25 clks

; PM: 11+26 = 37 DM: 6

;-----
FLO1624

FLO24      MOVLW      D'15'+EXPBIAS ; initialize exponent and add bias
           MOVWF      EXP
           MOVF        AARGB0,W
           MOVWF      SIGN
           BTFSS       AARGB0,MSB ; test sign
           GOTO        NRM2424
           COMF        AARGB1,F ; if < 0, negate and set MSB in SIGN
           COMF        AARGB0,F
           INCF        AARGB1,F
           BTFSC       _Z
           INCF        AARGB0,F

;*****
;*****

; Normalization routine

; Input: 24 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
; with sign in SIGN,MSB and other bits zero.

; Use: CALL NRM2424 or CALL NRM24

; Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

; Result: AARG <-- NORMALIZE( AARG )

```

```

;      Max Timing:      10+6+7*7+7 = 72 clks          SAT = 0
;                      10+6+7*7+1+11 = 77 clks SAT = 1

;      Min Timing:      14 clks                      AARG = 0
;                      5+9+4 = 18 clks

;      PM: 26                      DM: 6

;-----

NRM2424
NRM24
    CLRF      TEMP          ; clear exponent decrement
    MOVF      AARGB0,W      ; test if highbyte=0
    BTFSS     _Z
    GOTO      NORM2424
    MOVF      AARGB1,W      ; if so, shift 8 bits by move
    MOVWF     AARGB0
    BTFSC     _Z            ; if highbyte=0, result=0
    GOTO      RES024
    CLRF      AARGB1
    BSF       TEMP,3

NORM2424
    MOVF      TEMP,W
    SUBWF     EXP,F
    BTFSS     _Z
    BTFSS     _C
    GOTO      SETFUN24

    BCF       _C            ; clear carry bit

NORM2424A
    BTFSC     AARGB0,MSB    ; if MSB=1, normalization done
    GOTO      FIXSIGN24
    RLF       AARGB1,F      ; otherwise, shift left and
    RLF       AARGB0,F      ; decrement EXP
    DECFSZ    EXP,F
    GOTO      NORM2424A

    GOTO      SETFUN24      ; underflow if EXP=0

FIXSIGN24
    BTFSS     SIGN,MSB
    BCF       AARGB0,MSB    ; clear explicit MSB if positive
    RETLW     0

RES024
    CLRF      AARGB0        ; result equals zero
    CLRF      AARGB1
    CLRF      AARGB2        ; clear extended byte
    CLRF      EXP
    RETLW     0

;*****
;*****

;      Integer to float conversion

;      Input:  24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;      Use:    CALL    FLO2424

;      Output: 24 bit floating point number in AEXP, AARGB0, AARGB1

;      Result: AARG <--  FLOAT( AARG )

;      Max Timing:      14+94 = 108 clks          RND = 0
;                      14+103 = 117 clks         RND = 1, SAT = 0
;                      14+109 = 123 clks         RND = 1, SAT = 1

;      Min Timing:      6+28 = 34 clks          AARG = 0
;                      6+22 = 28 clks

;      PM: 14+51 = 65          DM: 7

;-----

FLO2424
    MOVLW     D'23'+EXPBIAS ; initialize exponent and add bias
    MOVWF     EXP
    CLRF      SIGN
    BTFSS     AARGB0,MSB    ; test sign
    GOTO      NRM3224
    COMF      AARGB2,F      ; if < 0, negate and set MSB in SIGN

```

```

        COMF          AARGB1,F
        COMF          AARGB0,F
        INCF          AARGB2,F
        BTFSC         _Z
        INCF          AARGB1,F
        BTFSC         _Z
        INCF          AARGB0,F
        BSF           SIGN,MSB

;*****

;      Normalization routine

;      Input:  32 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;              AARGB2, with sign in SIGN,MSB

;      Use:    CALL    NRM3224

;      Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

;      Result: AARG <-- NORMALIZE( AARG )

;      Max Timing:  21+6+7*8+7+4 = 94 clks  RND = 0
;                  21+6+7*8+20+4 = 103 clks   RND = 1, SAT = 0
;                  21+6+7*8+19+11 = 109 clks  RND = 1, SAT = 1

;      Min Timing:  22+6 = 28 clks           AARG = 0
;                  5+9+4+4 = 22 clks

;      PM: 51                                DM: 7

;-----

NRM3224      CLRF          TEMP                ; clear exponent decrement
            MOVF          AARGB0,W            ; test if highbyte=0
            BTFSS         _Z
            GOTO          NORM3224
            MOVF          AARGB1,W            ; if so, shift 8 bits by move
            MOVWF         AARGB0
            MOVF          AARGB2,W
            MOVWF         AARGB1
            CLRF          AARGB2
            BSF           TEMP,3              ; increase decrement by 8

            MOVF          AARGB0,W            ; test if highbyte=0
            BTFSS         _Z
            GOTO          NORM3224
            MOVF          AARGB1,W            ; if so, shift 8 bits by move
            MOVWF         AARGB0
            CLRF          AARGB1
            BCF           TEMP,3              ; increase decrement by 8
            BSF           TEMP,4

            MOVF          AARGB0,W            ; if highbyte=0, result=0
            BTFSC         _Z
            GOTO          RES024

NORM3224      MOVF          TEMP,W
            SUBWF          EXP,F
            BTFSS         _Z
            BTFSS         _C
            GOTO          SETFUN24

            BCF           _C                  ; clear carry bit

NORM3224A     BTFSC         AARGB0,MSB        ; if MSB=1, normalization done
            GOTO          NMRMRND3224
            RLF           AARGB2,F            ; otherwise, shift left and
            RLF           AARGB1,F            ; decrement EXP
            RLF           AARGB0,F
            DECFSZ        EXP,F
            GOTO          NORM3224A
            GOTO          SETFUN24            ; underflow if EXP=0

NMRMRND3224   BTFSC         FPFLAGS,RND
            BTFSS         AARGB1,LSB
            GOTO          FIXSIGN24
            BTFSS         AARGB2,MSB        ; round if next bit is set
            GOTO          FIXSIGN24
            INCF          AARGB1,F
            BTFSC         _Z

```

```

        INCF          AARGB0,F

        BTFSS        _Z                      ; has rounding caused carryout?
        GOTO         FIXSIGN24
        RRF          AARGB0,F                ; if so, right shift
        RRF          AARGB1,F
        INCF          EXP,F
        BTFSC        _Z                      ; check for overflow
        GOTO         SETFOV24
        GOTO         FIXSIGN24

;*****
;*****
;
;   Float to integer conversion
;
;   Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;
;   Use:    CALL    INT2416          or    CALL    INT24
;
;   Output: 16 bit 2's complement integer right justified in AARGB0, AARGB1
;
;   Result: AARG <-- INT( AARG )
;
;   Max Timing:  29+6*6+5+13 = 83 clks          RND = 0
;               29+6*6+5+19 = 89 clks          RND = 1, SAT = 0
;               29+6*6+5+22 = 92 clks          RND = 1, SAT = 1
;
;   Min Timing:  18+5+7 = 30 clks
;
;   PM: 63                                DM: 6
;-----
INT2416
INT24
    MOVF          EXP,W                    ; test for zero argument
    BTFSC         _Z
    RETLW         0x00

    MOVF          AARGB0,W                ; save sign in SIGN
    MOVWF         SIGN
    BSF           AARGB0,MSB              ; make MSB explicit

    MOVLW         EXPBIAS+D'15'           ; remove bias from EXP
    SUBWF         EXP,F
    BTFSS         EXP,MSB
    GOTO          SETIOV16
    COMF          EXP,F
    INCF          EXP,F

    MOVLW         8                       ; do byte shift if EXP >= 8
    SUBWF         EXP,W
    BTFSS         _C
    GOTO          TSHIFT2416
    MOVWF         EXP
    RLF           AARGB1,F                ; rotate next bit for rounding
    MOVF          AARGB0,W
    MOVWF         AARGB1
    CLRF          AARGB0

    MOVLW         8                       ; do byte shift if EXP >= 8
    SUBWF         EXP,W
    BTFSS         _C
    GOTO          TSHIFT2416
    MOVWF         EXP
    RLF           AARGB1,F                ; rotate next bit for rounding
    CLRF          AARGB1

    MOVF          EXP,W
    BTFSS         _Z
    BCF           _C
    GOTO          SHIFT2416OK

TSHIFT2416
    MOVF          EXP,W                    ; shift completed if EXP = 0
    BTFSC         _Z
    GOTO          SHIFT2416OK

SHIFT2416
    BCF           _C
    RRF          AARGB0,F                ; right shift by EXP
    RRF          AARGB1,F
    DECFSZ       EXP,F

```

```

        GOTO          SHIFT2416

SHIFT2416OK    BTFSC          FPFLAGS,RND
               BTFSS          AARGB1,LSB
               GOTO          INT2416OK
               ; round if next bit is set
               BTFSS          _C          INT2416OK
               GOTO          INT2416OK
               INCF          AARGB1,F
               BTFSC          _Z
               INCF          AARGB0,F

               BTFSC          AARGB0,MSB      ; test for overflow
               GOTO          SETIOV16

INT2416OK      BTFSS          SIGN,MSB          ; if sign bit set, negate
               RETLW          0
               COMF          AARGB1,F
               COMF          AARGB0,F
               INCF          AARGB1,F
               BTFSC          _Z
               INCF          AARGB0,F
               RETLW          0

SETIOV16       BSF          FPFLAGS,IOV          ; set integer overflow flag
               BTFSS          FPFLAGS,SAT        ; test for saturation
               RETLW          0xFF              ; return error code in WREG

               CLRF          AARGB0              ; saturate to largest two's
               BTFSS          SIGN,MSB          ; complement 16 bit integer
               MOVLW          0xFF
               MOVWF          AARGB0              ; SIGN = 0, 0x 7F FF
               MOVWF          AARGB1              ; SIGN = 1, 0x 80 00
               RLF          SIGN,F
               RRF          AARGB0,F
               RETLW          0xFF              ; return error code in WREG

```

```

;*****
;*****

```

```

;      Float to integer conversion

;      Input:  24 bit floating point number in AEXP, AARGB0, AARGB1

;      Use:    CALL    INT2424

;      Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;      Result: AARG <-- INT( AARG )

;      Max Timing:  41+6*7+6+16 = 105 clks      RND = 0
;                  41+6*7+6+24 = 113 clks      RND = 1, SAT = 0
;                  41+6*7+6+26 = 115 clks      RND = 1, SAT = 1

;      Min Timing:   5 clks

;      PM: 82                      DM: 6

```

```

;-----

```

```

INT2424        CLRF          AARGB2
               MOVF          EXP,W              ; test for zero argument
               BTFSC          _Z
               RETLW          0x00

               MOVF          AARGB0,W          ; save sign in SIGN
               MOVWF          SIGN
               BSF          AARGB0,MSB        ; make MSB explicit

               MOVLW          EXPBIAS+D'23'    ; remove bias from EXP
               SUBWF          EXP,F
               BTFSS          EXP,MSB
               GOTO          SETIOV24

               COMF          EXP,F
               INCF          EXP,F

               MOVLW          8                  ; do byte shift if EXP >= 8
               SUBWF          EXP,W
               BTFSS          _C

```

```

        GOTO          TSHIFT2424
        MOVWF         EXP
        RLF           AARGB2,F      ; rotate next bit for rounding
        MOVF          AARGB1,W
        MOVWF         AARGB2
        MOVF          AARGB0,W
        MOVWF         AARGB1
        CLRF          AARGB0

        MOVLW         8              ; do another byte shift if EXP >= 8
        SUBWF         EXP,W
        BTFSS         _C
        GOTO          TSHIFT2424
        MOVWF         EXP
        RLF           AARGB2,F      ; rotate next bit for rounding
        MOVF          AARGB1,W
        MOVWF         AARGB2
        CLRF          AARGB1

        MOVLW         8              ; do another byte shift if EXP >= 8
        SUBWF         EXP,W
        BTFSS         _C
        GOTO          TSHIFT2424
        MOVWF         EXP
        RLF           AARGB2,F      ; rotate next bit for rounding
        CLRF          AARGB2

        MOVF          EXP,W
        BTFSS         _Z
        BCF           _C
        GOTO          SHIFT2424OK

TSHIFT2424  MOVF          EXP,W      ; shift completed if EXP = 0
            BTFSC         _Z
            GOTO          SHIFT2424OK

SHIFT2424   BCF           _C
            RRF           AARGB0,F   ; right shift by EXP
            RRF           AARGB1,F
            RRF           AARGB2,F
            DECF          EXP,F
            GOTO          SHIFT2424

SHIFT2424OK BTFSC         FPFLAGS,RND
            BTFSS         AARGB2,LSB
            GOTO          INT2424OK
            BTFSS         _C
            GOTO          INT2424OK
            INCF          AARGB2,F
            BTFSC         _Z
            INCF          AARGB1,F
            BTFSC         _Z
            INCF          AARGB0,F
            BTFSC         AARGB0,MSB ; test for overflow
            GOTO          SETIOV24

INT2424OK   BTFSS         SIGN,MSB   ; if sign bit set, negate
            RETLW         0
            COMF          AARGB0,F
            COMF          AARGB1,F
            COMF          AARGB2,F
            INCF          AARGB2,F
            BTFSC         _Z
            INCF          AARGB1,F
            BTFSC         _Z
            INCF          AARGB0,F
            RETLW         0

IRES024     CLRF          AARGB0      ; integer result equals zero
            CLRF          AARGB1
            CLRF          AARGB2
            RETLW         0

SETIOV24    BSF           FPFLAGS,IOV ; set integer overflow flag
            BTFSS         FPFLAGS,SAT ; test for saturation
            RETLW         0xFF        ; return error code in WREG

            CLRF          AARGB0      ; saturate to largest two's
            BTFSS         SIGN,MSB   ; complement 24 bit integer
            MOVLW         0xFF
            MOVWF         AARGB0      ; SIGN = 0, 0x 7F FF FF
            MOVWF         AARGB1      ; SIGN = 1, 0x 80 00 00

```

```

MOVWF      AARGB2
RLF        SIGN,F
RRF        AARGB0,F
RETLW      0xFF          ; return error code in WREG

;*****

;   Multiply by 10
;
;   This routine multiplies the 24-bit floating point number in
;   aarg by 10.

AargTimes10
    movlw   3
    bcf status,rp0
    addwf   aexp,f          ; Multiply aarg by 3
    clrf    Temp1           ; Set 1 if aarg is negative
    btfsc   aarg+1,7
    incf    Temp1,f
    bsf aarg+1,7           ; Make number positive & insert
                          ; implied MSB
    MoveBytes aarg+1,barg+1,2 ; Copy fraction to barg
    rrf barg+1,f           ; Shift barg+1, barg+2 right 2 times
    rrf barg+2,f
    bcf barg+1,7
    rrf barg+1,f
    rrf barg+2,w
    bcf barg+1,7
    addwf   aarg+2,f        ; add aarg+1,2 to barg+1,2
    btfsc   status,c
    incf    barg+1,f
    movf    barg+1,w
    addwf   aarg+1,f
    btfss   status,c        ; Skip if carry
    goto    $+4            ; No carry
    rrf aarg+1,f           ; Shift right 1 place
    rrf aarg+2,f           ; and add 1 to exponent
    incf    aexp,f
    bcf aarg+1,7
    btfsc   Temp1,0         ; Is number negative?
    bsf aarg+1,7           ; Yes, set sign bit
    return

;*****
;*****

;   Floating Point Multiply
;
;   Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;           24 bit floating point number in BEXP, BARGB0, BARGB1
;
;   Use:    CALL    FPM24
;
;   Output: 24 bit floating point product in AEXP, AARGB0, AARGB1
;
;   Result: AARG <-- AARG * BARG
;
;   Max Timing:  25+15*16+15+18 = 298 clks      RND = 0
;               25+15*16+15+29 = 309 clks      RND = 1, SAT = 0
;               25+15*16+15+33 = 313 clks      RND = 1, SAT = 1
;
;   Min Timing:  6+5 = 11 clks                  AARG * BARG = 0
;               24+15*11+14+15 = 218 clks
;
;   PM: 80                      DM: 11
;-----

FPM24      MOVF      AEXP,W          ; test for zero arguments
           BTFSS     _Z
           MOVF      BEXP,W
           BTFSC     _Z
           GOTO      RES024

M24BNE0    MOVF      AARGB0,W
           XORWF     BARGB0,W
           MOVWF     SIGN          ; save sign in SIGN
           MOVF      BEXP,W

```

	ADDWF	EXP,F	
	MOVLW	EXPBIAS-1	
	BTFSS	_C	
	GOTO	MTUN24	
	SUBWF	EXP,F	
	BTFSC	_C	
	GOTO	SETFOV24	; set multiply overflow flag
	GOTO	MOK24	
MTUN24	SUBWF	EXP,F	
	BTFSS	_C	
	GOTO	SETFUN24	
MOK24	MOVF	AARGB0,W	
	MOVWF	AARGB2	; move result to AARG
	MOVF	AARGB1,W	
	MOVWF	AARGB3	
BSF	AARGB2,MSB		; make argument MSB's explicit
	BSF	BARGB0,MSB	
	BCF	_C	
	CLRF	AARGB0	; clear initial partial product
	CLRF	AARGB1	
	MOVLW	D'16'	
	MOVWF	TEMP	; initialize counter
MLOOP24	BTFSS	AARGB3,LSB	; test next bit
	GOTO	MNOADD24	
MADD24	MOVF	BARGB1,W	
	ADDWF	AARGB1,F	
	MOVF	BARGB0,W	
	BTFSC	_C	
	INCFSZ	BARGB0,W	
	ADDWF	AARGB0,F	
MNOADD24	RRF	AARGB0,F	
	RRF	AARGB1,F	
	RRF	AARGB2,F	
	RRF	AARGB3,F	
	BCF	_C	
	DECFSZ	TEMP,F	
	GOTO	MLOOP24	
	BTFSC	AARGB0,MSB	; check for postnormalization
	GOTO	MROUND24	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	EXP,F	
MROUND24	BTFSC	FPFLAGS,RND	
	BTFSS	AARGB1,LSB	
	GOTO	MUL24OK	
BTFSS	AARGB2,MSB		; round if next bit is set
	GOTO	MUL24OK	
INCF	AARGB1,F		
	BTFSC	_Z	
	INCF	AARGB0,F	
	BTFSS	_Z	; has rounding caused carryout?
	GOTO	MUL24OK	
	RRF	AARGB0,F	; if so, right shift
	RRF	AARGB1,F	
	INCF	EXP,F	
	BTFSC	_Z	; check for overflow
	GOTO	SETFOV24	
MUL24OK	BTFSS	SIGN,MSB	
	BCF	AARGB0,MSB	; clear explicit MSB if positive
	RETLW	0	
SETFOV24	BSF	FPFLAGS,FOV	; set floating point underflag
	BTFSS	FPFLAGS,SAT	; test for saturation
	RETLW	0xFF	; return error code in WREG
	MOVLW	0xFF	
	MOVWF	AEXP	; saturate to largest floating
	MOVWF	AARGB0	; point number = 0x FF 7F FF


```

        MOVWF      AARGB1      ; modulo the appropriate sign bit
        RLF        SIGN,F
        RRF        AARGB0,F
        RETLW      0xFF      ; return error code in WREG

;*****
;*****
;
;   Floating Point Divide
;
;   Input:  24 bit floating point dividend in AEXP, AARGB0, AARGB1
;           24 bit floating point divisor in BEXP, BARGB0, BARGB1
;
;   Use:    CALL    FPD24
;
;   Output: 24 bit floating point quotient in AEXP, AARGB0, AARGB1
;
;   Result: AARG <-- AARG / BARG
;
;   Max Timing:  32+13+15*26+25+12 = 472 clks   RND = 0
;               32+13+15*26+25+34 = 494 clks   RND = 1, SAT = 0
;               32+13+15*26+25+38 = 498 clks   RND = 1, SAT = 1
;
;   Min Timing:  7+5 = 12 clks
;
;   PM: 120                      DM: 11
;-----
FPD24      MOVF      BEXP,W      ; test for divide by zero
          BTFSC     _Z
          GOTO      SETFDZ24

          MOVF      AEXP,W
          BTFSC     _Z
          GOTO      RES024

D24BNE0    MOVF      AARGB0,W
          XORWF     BARGB0,W
          MOVWF     SIGN      ; save sign in SIGN
          BSF       AARGB0,MSB ; make argument MSB's explicit
          BSF       BARGB0,MSB

TALIGN24   CLRF      TEMP      ; clear align increment
          MOVF      AARGB0,W
          MOVWF     AARGB2
          MOVF      AARGB1,W   ; test for alignment
          MOVWF     AARGB3

          MOVF      BARGB1,W
          SUBWF     AARGB3, f
          MOVF      BARGB0,W
          BTFSS     _C
          INCF      BARGB0,W
          SUBWF     AARGB2, f

          CLRF      AARGB2
          CLRF      AARGB3

          BTFSS     _C
          GOTO      DALIGN24OK

          BCF       _C      ; align if necessary
          RRF       AARGB0,F
          RRF       AARGB1,F
          RRF       AARGB2,F
          MOVLW     0x01
          MOVWF     TEMP      ; save align increment

DALIGN24OK MOVF      BEXP,W      ; compare AEXP and BEXP
          SUBWF     EXP,F
          BTFSS     _C
          GOTO      ALTB24

AGEB24     MOVLW     EXPBIAS-1
          ADDWF     TEMP,W
          ADDWF     EXP,F
          BTFSC     _C
          GOTO      SETFOV24
          GOTO      DARGOK24      ; set overflow flag

```



```

        RETLW        0

SETFUN24    BSF        FPFLAGS,FUN        ; set floating point underflag
            BTFSS      FPFLAGS,SAT        ; test for saturation
            RETLW      0xFF                ; return error code in WREG

            MOVLW      0x01                ; saturate to smallest floating
            MOVWF      AEXP                ; point number = 0x 01 00 00
            CLRF       AARGB0              ; modulo the appropriate sign bit
            CLRF       AARGB1
            RLF        SIGN,F
            RRF        AARGB0,F
            RETLW      0xFF                ; return error code in WREG

SETFDZ24    BSF        FPFLAGS,FDZ        ; set divide by zero flag
            RETLW      0xFF

;*****
;*****

;      Floating Point Subtract

;      Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;              24 bit floating point number in BEXP, BARGB0, BARGB1

;      Use:    CALL FPS24

;      Output: 24 bit floating point sum in AEXP, AARGB0, AARGB1

;      Result: AARG <-- AARG - BARG

;      Max Timing:  2+197 = 199 clks          RND = 0
;                  2+208 = 210 clks          RND = 1, SAT = 0
;                  2+213 = 215 clks          RND = 1, SAT = 1

;      Min Timing:  2+12 = 14 clks

;      PM: 2+112 = 114                      DM: 11

;-----

FPS24       MOVLW      0x80
            XORWF      BARGB0,F

;*****
;*****

;      Floating Point Add

;      Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;              24 bit floating point number in BEXP, BARGB0, BARGB1

;      Use:    CALL FPA24

;      Output: 24 bit floating point sum in AEXP, AARGB0, AARGB1

;      Result: AARG <-- AARG + BARG

;      Max Timing:  25+28+6*6+5+31+72 = 197 clks          RND = 0
;                  25+28+6*6+5+42+72 = 208 clks          RND = 1, SAT = 0
;                  25+28+6*6+5+42+77 = 213 clks          RND = 1, SAT = 1

;      Min Timing:  8+4 = 12 clks

;      PM: 112                                DM: 11

;-----

FPA24       MOVF       AARGB0,W            ; exclusive or of signs in TEMP
            XORWF      BARGB0,W
            MOVWF      TEMP

            CLRF       AARGB2              ; clear extended byte
            CLRF       BARGB2

            MOVF       AEXP,W              ; use AARG if AEXP >= BEXP
            SUBWF      BEXP,W
            BTFSS      _C
            GOTO       USEA24

            MOVF       BEXP,W              ; use BARG if AEXP < BEXP
            MOVWF      AARGB4              ; therefore, swap AARG and BARG

```

```

        MOVF      AEXP,W
        MOVWF     BEXP
        MOVF      AARGB4,W
        MOVWF     AEXP

        MOVF      BARGB0,W
        MOVWF     AARGB4
        MOVF      AARGB0,W
        MOVWF     BARGB0
        MOVF      AARGB4,W
        MOVWF     AARGB0

        MOVF      BARGB1,W
        MOVWF     AARGB4
        MOVF      AARGB1,W
        MOVWF     BARGB1
        MOVF      AARGB4,W
        MOVWF     AARGB1

USEA24   MOVF      BEXP,W           ; return AARG if BARG = 0
        BTFSC     _Z
        RETLW     0x00

        MOVF      AARGB0,W
        MOVWF     SIGN           ; save sign in SIGN
        BSF       AARGB0,MSB     ; make MSB's explicit
        BSF       BARGB0,MSB

        MOVF      BEXP,W           ; compute shift count in BEXP
        SUBWF     AEXP,W
        MOVWF     BEXP
        BTFSC     _Z
        GOTO      ALIGNED24

        MOVLW     8
        SUBWF     BEXP,W
        BTFSS     _C             ; if BEXP >= 8, do byte shift
        GOTO      ALIGNB24
        MOVWF     BEXP
        MOVF      BARGB1,W       ; keep for postnormalization

MOVWF    BARGB2
MOVWF    BARGB1
        CLRF      BARGB0

        MOVLW     8
        SUBWF     BEXP,W
        BTFSS     _C             ; if BEXP >= 8, BARG = 0 relative to AARG
        GOTO      ALIGNB24
        MOVF      SIGN,W
        MOVWF     AARGB0
        RETLW     0x00

ALIGNB24 MOVF      BEXP,W           ; already aligned if BEXP = 0
        BTFSC     _Z
        GOTO      ALIGNED24

ALOOPB24 BCF       _C             ; right shift by BEXP
        RRF       BARGB0,F
        RRF       BARGB1,F

        RRF       BARGB2,F
        DECFSZ    BEXP,F
        GOTO      ALOOPB24

ALIGNED24 BTFSS     TEMP,MSB       ; negate if signs opposite
        GOTO      AOK24
        COMF      BARGB2,F
        COMF      BARGB1,F
        COMF      BARGB0,F
        INCF      BARGB2,F
        BTFSC     _Z
        INCF      BARGB1,F

        BTFSC     _Z
        INCF      BARGB0,F

AOK24   MOVF      BARGB2,W
        ADDWF     AARGB2,F
        MOVF      BARGB1,W
        BTFSC     _C
        INCF      BARGB1,W

```

```

        ADDWF      AARGB1,F
        MOVF      BARGB0,W
        BTFSC     _C
        INCFSZ    BARGB0,W
        ADDWF     AARGB0,F

        BTFSC     TEMP,MSB
        GOTO      ACOMP24
        BTFSS     _C
        GOTO      NMRMRND3224

        RRF       AARGB0,F      ; shift right and increment EXP
        RRF       AARGB1,F
        RRF       AARGB2,F
        INCFSZ    AEXP,F
        GOTO      NMRMRND3224
        GOTO      SETFOV24

ACOMP24      BTFSC     _C
        GOTO      NRM3224      ; normalize and fix sign

        COMF      AARGB2,F
        COMF      AARGB1,F      ; negate, toggle sign bit and
        COMF      AARGB0,F      ; then normalize
        INCF      AARGB2,F
        BTFSC     _Z
        INCF      AARGB1,F
        BTFSC     _Z
        INCF      AARGB0,F

        MOVLW     0x80
        XORWF     SIGN,F
        GOTO      NRM24

;*****
;*****
; Evaluate floor(x)

; Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
; Use:        CALL FLOOR24
; Output:     24 bit floating point number in AEXP, AARGB0, AARGB1
; Result:     AARG <-- FLOOR( AARG )
; Testing on [-MAXNUM,MAXNUM] from 100000 trials:

; min      max mean
; Timing:   37  55  42.7 clks

; min      max meanrms
; Error:    0x00 0x00 0.0 0.0 nsb

;-----

; floor(x) evaluates the largest integer, as a float, not greater than x.

FLOOR24
        CLRF      AARGB2      ; test for zero argument
        MOVF      AEXP,W
        BTFSC     _Z
        RETLW     0x00

        MOVF      AARGB0,W
        MOVWF     AARGB3      ; save mantissa
        MOVF      AARGB1,W
        MOVWF     AARGB4

        MOVLW     EXPBIAS      ; computed unbiased exponent
        SUBWF     AEXP,W
        MOVWF     TEMPB1
        BTFSC     TEMPB1,MSB
        GOTO      FLOOR24ZERO

        SUBLW     0x10-1
        MOVWF     TEMPB0      ; save number of zero bits in TEMPB0
        MOVWF     TEMPB1

        BTFSC     TEMPB1,LSB+3 ; divide by eight

```

```

        GOTO          FLOOR24MASKH

FLOOR24MASKL
    MOVLW            0x07          ; get remainder for mask pointer
    ANDWF            TEMPB0,F
    MOVLW            LOW FLOOR24MASKTABLE
    ADDWF            TEMPB0,F
    MOVLW            HIGH FLOOR24MASKTABLE
    BTFSC            _C
    ADDLW            0x01
    MOVWF            PCLATH
    INCF             TEMPB0,W

    CALL             FLOOR24MASKTABLE ; access table for mask

    ANDWF            AARGB1,F
    BTFSS            AARGB0,MSB      ; if negative, round down
    RETLW            0x00

    MOVWF            AARGB7
    MOVF             AARGB4,W
    SUBWF            AARGB1,W
    BTFSS            _Z
    GOTO             FLOOR24RNDL
    RETLW            0x00

FLOOR24RNDL
    COMF             AARGB7,W
    MOVWF            TEMPB1
    INCF             TEMPB1,W
    ADDWF            AARGB1,F
    BTFSC            _Z
    INCF             AARGB0, F      ; has rounding caused carryout?
    BTFSS            _Z
    RETLW            0x00
    RRF             AARGB0,F
    RRF             AARGB1,F
    INCFSZ           AEXP,F        ; check for overflow
    RETLW            0x00
    GOTO             SETFOV24

FLOOR24MASKH
    MOVLW            0x07          ; get remainder for mask pointer
    ANDWF            TEMPB0,F
    MOVLW            LOW FLOOR24MASKTABLE
    ADDWF            TEMPB0,F
    MOVLW            HIGH FLOOR24MASKTABLE
    BTFSC            _C
    ADDLW            0x01
    MOVWF            PCLATH
    INCF             TEMPB0,W

    CALL             FLOOR24MASKTABLE ; access table for mask

    ANDWF            AARGB0,F
    CLRF             AARGB1
    BTFSS            AARGB0,MSB      ; if negative, round down
    RETLW            0x00

    MOVWF            AARGB7
    MOVF             AARGB4,W
    SUBWF            AARGB1,W
    BTFSS            _Z
    GOTO             FLOOR24RNDH
    MOVF             AARGB3,W
    SUBWF            AARGB0,W
    BTFSS            _Z
    GOTO             FLOOR24RNDH
    RETLW            0x00

FLOOR24RNDH
    COMF             AARGB7,W
    MOVWF            TEMPB1
    INCF             TEMPB1,W
    ADDWF            AARGB0,F
    BTFSC            _C
    RETLW            0x00          ; has rounding caused carryout?
    RRF             AARGB0,F
    RRF             AARGB1,F
    INCFSZ           AEXP,F
    RETLW            0x00

```

```

        GOTO          SETFOV24      ; check for overflow

FLOOR24ZERO
    BTFSC            AARGB0,MSB
    GOTO            FLOOR24MINUSONE
    CLRF            AEXP
    CLRF            AARGB0
    CLRF            AARGB1
    RETLW           0x00

FLOOR24MINUSONE
    MOVLW           0x7F
    MOVWF           AEXP
    MOVLW           0x80
    MOVWF           AARGB0
    CLRF            AARGB1
    RETLW           0x00

;-----

;   table for least significant byte requiring masking, using pointer from
;   the remainder of the number of zero bits divided by eight.

FLOOR24MASKTABLE
    MOVWF           PCL
    RETLW           0xFF
    RETLW           0xFE
    RETLW           0xFC
    RETLW           0xF8
    RETLW           0xF0
    RETLW           0xE0
    RETLW           0xC0
    RETLW           0x80
    RETLW           0x00

;*****
;*****

;   Floating Point Relation   A < B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;   24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALL TALTB24

;   Output:     logical result in W

;   Result:     if A < B TRUE,      W = 0x01
;               if A < B FALSE, W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min      max  mean
;   Timing:   9   28  14.6 clks

TALTB24      MOVF      AARGB0,W      ; test if signs opposite
             XORWF     BARGB0,W
             MOVWF     TEMPB0
             BTFSC     TEMPB0,MSB
             GOTO      TALTB24O

             BTFSC     AARGB0,MSB
             GOTO      TALTB24N

TALTB24P     MOVF      AEXP,W        ; compare positive arguments
             SUBWF     BEXP,W
             BTFSS     _C
             RETLW     0x00
             BTFSS     _Z
             RETLW     0x01

             MOVF      AARGB0,W
             SUBWF     BARGB0,W
             BTFSS     _C
             RETLW     0x00
             BTFSS     _Z
             RETLW     0x01

             MOVF      AARGB1,W
             SUBWF     BARGB1,W
             BTFSS     _C

```

```

        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01
        RETLW        0x00

TALTB24N        MOVF        BEXP,W            ; compare negative arguments
        SUBWF        AEXP,W
        BTFSS        _C
        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01

        MOVF        BARGB0,W
        SUBWF        AARGB0,W
        BTFSS        _C
        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01

        MOVF        BARGB1,W
        SUBWF        AARGB1,W
        BTFSS        _C
        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01
        RETLW        0x00

TALTB24O        BTFSS        BARGB0,MSB
        RETLW        0x01
        RETLW        0x00

;*****
;*****

;   Floating Point Relation   A <= B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;   24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALL TALEB24

;   Output:     logical result in W

;   Result:     if A <= B TRUE,   W = 0x01
;               if A <= B FALSE,  W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min         max mean
;   Timing:     9   26  14.4 clks

TALEB24        MOVF        AARGB0,W            ; test if signs opposite
        XORWF        BARGB0,W
        MOVWF        TEMPB0
        BTFSC        TEMPB0,MSB
        GOTO         TALEB24O

        BTFSC        AARGB0,MSB
        GOTO         TALEB24N

TALEB24P        MOVF        AEXP,W            ; compare positive arguments
        SUBWF        BEXP,W
        BTFSS        _C
        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01

        MOVF        AARGB0,W
        SUBWF        BARGB0,W
        BTFSS        _C
        RETLW        0x00
        BTFSS        _Z
        RETLW        0x01

        MOVF        AARGB1,W
        SUBWF        BARGB1,W
        BTFSS        _C
        RETLW        0x00
        RETLW        0x01

TALEB24N        MOVF        BEXP,W            ; compare negative arguments

```



```

        SUBWF      AEXP,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01

        MOVF       BARGB0,W
        SUBWF      AARGB0,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01

        MOVF       BARGB1,W
        SUBWF      AARGB1,W
        BTFSS      _C
        RETLW      0x00
        RETLW      0x01

TALB240      BTFSS      BARGB0,MSB
        RETLW      0x01
        RETLW      0x00

;*****
;*****

;   Floating Point Relation   A > B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;   24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALLTAGTB24

;   Output:     logical result in W

;   Result:     if A > B TRUE,      W = 0x01
;               if A > B FALSE, W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min      max mean
;   Timing:   9   28 14.6clks

TAGTB24      MOVF      BARGB0,W      ; test if signs opposite
        XORWF      AARGB0,W
        MOVWF      TEMPB0
        BTFSC      TEMPB0,MSB
        GOTO       TAGTB240

        BTFSC      BARGB0,MSB
        GOTO       TAGTB24N

TAGTB24P      MOVF      BEXP,W      ; compare positive arguments
        SUBWF      AEXP,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01

        MOVF       BARGB0,W
        SUBWF      AARGB0,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01

        MOVF       BARGB1,W
        SUBWF      AARGB1,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01
        RETLW      0x00

TAGTB24N      MOVF      AEXP,W      ; compare negative arguments
        SUBWF      BEXP,W
        BTFSS      _C
        RETLW      0x00
        BTFSS      _Z
        RETLW      0x01

```

```

        MOVF      AARGB0,W
        SUBWF     BARGB0,W
        BTFSS     _C
        RETLW     0x00
        BTFSS     _Z
        RETLW     0x01

        MOVF      AARGB1,W
        SUBWF     BARGB1,W
        BTFSS     _C
        RETLW     0x00
        BTFSS     _Z
        RETLW     0x01
        RETLW     0x00

TAGTB240      BTFSS      AARGB0,MSB
               RETLW     0x01
               RETLW     0x00

;*****
;*****

;   Floating Point Relation   A >= B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALL TAGEB24

;   Output:     logical result in W

;   Result:     if A >= B TRUE,   W = 0x01
;               if A >= B FALSE,  W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min      max mean
;   Timing:   9   26  14.4 clks

TAGEB24      MOVF      BARGB0,W      ; test if signs opposite
               XORWF     AARGB0,W
               MOVWF     TEMPB0
               BTFSC     TEMPB0,MSB
               GOTO      TAGEB240

               BTFSC     BARGB0,MSB
               GOTO      TAGEB24N

TAGEB24P     MOVF      BEXP,W        ; compare positive arguments
               SUBWF     AEXP,W
               BTFSS     _C
               RETLW     0x00
               BTFSS     _Z
               RETLW     0x01

               MOVF      BARGB0,W
               SUBWF     AARGB0,W
               BTFSS     _C
               RETLW     0x00
               BTFSS     _Z
               RETLW     0x01

               MOVF      BARGB1,W
               SUBWF     AARGB1,W
               BTFSS     _C
               RETLW     0x00
               RETLW     0x01

TAGEB24N     MOVF      AEXP,W        ; compare negative arguments
               SUBWF     BEXP,W
               BTFSS     _C
               RETLW     0x00
               BTFSS     _Z
               RETLW     0x01

               MOVF      AARGB0,W
               SUBWF     BARGB0,W
               BTFSS     _C
               RETLW     0x00
               BTFSS     _Z
               RETLW     0x01

```

```

        MOVF      AARGB1,W
        SUBWF     BARGB1,W
        BTFSS     _C
        RETLW     0x00
        RETLW     0x01

TAGEB240      BTFSS      AARGB0,MSB
        RETLW     0x01
        RETLW     0x00

;*****
;*****

;   Floating Point Relation   A == B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALLTAEQB24

;   Output:     logical result in W

;   Result:     if A == B TRUE,   W = 0x01
;               if A == B FALSE,  W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min      max mean
;   Timing:   5   14  6.9  clks

TAEQB24      MOVF      BEXP,W
        SUBWF     AEXP,W
        BTFSS     _Z
        RETLW     0x00

        MOVF      BARGB0,W
        SUBWF     AARGB0,W
        BTFSS     _Z
        RETLW     0x00

        MOVF      BARGB1,W
        SUBWF     AARGB1,W
        BTFSS     _Z
        RETLW     0x00
        RETLW     0x01

;*****
;*****

;   Floating Point Relation   A != B

;   Input:      24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;   Use:        CALLTANEB24

;   Output:     logical result in W

;   Result:     if A != B TRUE,   W = 0x01
;               if A != B FALSE,  W = 0x00

;   Testing on [-MAXNUM,MAXNUM] from 100000 trials:

;   min      max mean
;   Timing:   5   14  6.9  clks

TANEB24      MOVF      BEXP,W
        SUBWF     AEXP,W
        BTFSS     _Z
        RETLW     0x01

        MOVF      BARGB0,W
        SUBWF     AARGB0,W
        BTFSS     _Z
        RETLW     0x01

        MOVF      BARGB1,W
        SUBWF     AARGB1,W

```

```

BTFS    _Z
RETLW   0x01
RETLW   0x00

```

```

org      0x1000

```

```

;*****
; Lookup Table for Forward (Channel 0) Power
; Calibration dBm = .10063*ADC + -40.83

```

```

FLowByteTable0

```

retlw	0x8B	; Low Byte, ADRES = 0, Forward Power = 82.60 nW
retlw	0x15	; Low Byte, ADRES = 4, Forward Power = 90.63 nW
retlw	0xAC	; Low Byte, ADRES = 8, Forward Power = 99.43 nW
retlw	0x52	; Low Byte, ADRES = 12, Forward Power = 109.1 nW
retlw	0x04	; Low Byte, ADRES = 16, Forward Power = 119.7 nW
retlw	0x68	; Low Byte, ADRES = 20, Forward Power = 131.3 nW
retlw	0xD5	; Low Byte, ADRES = 24, Forward Power = 144.0 nW
retlw	0x4E	; Low Byte, ADRES = 28, Forward Power = 158.0 nW
retlw	0xD1	; Low Byte, ADRES = 32, Forward Power = 173.4 nW
retlw	0x62	; Low Byte, ADRES = 36, Forward Power = 190.2 nW
retlw	0x01	; Low Byte, ADRES = 40, Forward Power = 208.7 nW
retlw	0xAF	; Low Byte, ADRES = 44, Forward Power = 229.0 nW
retlw	0x37	; Low Byte, ADRES = 48, Forward Power = 251.2 nW
retlw	0xA0	; Low Byte, ADRES = 52, Forward Power = 275.6 nW
retlw	0x13	; Low Byte, ADRES = 56, Forward Power = 302.4 nW
retlw	0x91	; Low Byte, ADRES = 60, Forward Power = 331.7 nW
retlw	0x1B	; Low Byte, ADRES = 64, Forward Power = 363.9 nW
retlw	0xB3	; Low Byte, ADRES = 68, Forward Power = 399.3 nW
retlw	0x59	; Low Byte, ADRES = 72, Forward Power = 438.1 nW
retlw	0x08	; Low Byte, ADRES = 76, Forward Power = 480.6 nW
retlw	0x6C	; Low Byte, ADRES = 80, Forward Power = 527.3 nW
retlw	0xDA	; Low Byte, ADRES = 84, Forward Power = 578.5 nW
retlw	0x53	; Low Byte, ADRES = 88, Forward Power = 634.7 nW
retlw	0xD7	; Low Byte, ADRES = 92, Forward Power = 696.3 nW
retlw	0x69	; Low Byte, ADRES = 96, Forward Power = 763.9 nW
retlw	0x08	; Low Byte, ADRES = 100, Forward Power = 838.1 nW
retlw	0xB7	; Low Byte, ADRES = 104, Forward Power = 919.5 nW
retlw	0x3B	; Low Byte, ADRES = 108, Forward Power = 1.009 uW
retlw	0xA4	; Low Byte, ADRES = 112, Forward Power = 1.107 uW
retlw	0x18	; Low Byte, ADRES = 116, Forward Power = 1.214 uW
retlw	0x96	; Low Byte, ADRES = 120, Forward Power = 1.332 uW
retlw	0x21	; Low Byte, ADRES = 124, Forward Power = 1.462 uW
retlw	0xBA	; Low Byte, ADRES = 128, Forward Power = 1.603 uW
retlw	0x61	; Low Byte, ADRES = 132, Forward Power = 1.759 uW
retlw	0x0C	; Low Byte, ADRES = 136, Forward Power = 1.930 uW
retlw	0x71	; Low Byte, ADRES = 140, Forward Power = 2.117 uW
retlw	0xDF	; Low Byte, ADRES = 144, Forward Power = 2.323 uW
retlw	0x58	; Low Byte, ADRES = 148, Forward Power = 2.549 uW
retlw	0xDD	; Low Byte, ADRES = 152, Forward Power = 2.796 uW
retlw	0x6F	; Low Byte, ADRES = 156, Forward Power = 3.068 uW
retlw	0x0F	; Low Byte, ADRES = 160, Forward Power = 3.366 uW
retlw	0xBE	; Low Byte, ADRES = 164, Forward Power = 3.693 uW
retlw	0x40	; Low Byte, ADRES = 168, Forward Power = 4.051 uW
retlw	0xA9	; Low Byte, ADRES = 172, Forward Power = 4.445 uW
retlw	0x1D	; Low Byte, ADRES = 176, Forward Power = 4.876 uW
retlw	0x9C	; Low Byte, ADRES = 180, Forward Power = 5.350 uW
retlw	0x28	; Low Byte, ADRES = 184, Forward Power = 5.869 uW
retlw	0xC1	; Low Byte, ADRES = 188, Forward Power = 6.439 uW
retlw	0x68	; Low Byte, ADRES = 192, Forward Power = 7.065 uW
retlw	0x10	; Low Byte, ADRES = 196, Forward Power = 7.751 uW
retlw	0x75	; Low Byte, ADRES = 200, Forward Power = 8.504 uW
retlw	0xE4	; Low Byte, ADRES = 204, Forward Power = 9.329 uW
retlw	0x5E	; Low Byte, ADRES = 208, Forward Power = 10.24 uW
retlw	0xE3	; Low Byte, ADRES = 212, Forward Power = 11.23 uW
retlw	0x76	; Low Byte, ADRES = 216, Forward Power = 12.32 uW
retlw	0x16	; Low Byte, ADRES = 220, Forward Power = 13.52 uW
retlw	0xC6	; Low Byte, ADRES = 224, Forward Power = 14.83 uW
retlw	0x44	; Low Byte, ADRES = 228, Forward Power = 16.27 uW
retlw	0xAE	; Low Byte, ADRES = 232, Forward Power = 17.85 uW
retlw	0x22	; Low Byte, ADRES = 236, Forward Power = 19.58 uW
retlw	0xA2	; Low Byte, ADRES = 240, Forward Power = 21.48 uW
retlw	0x2E	; Low Byte, ADRES = 244, Forward Power = 23.57 uW
retlw	0xC7	; Low Byte, ADRES = 248, Forward Power = 25.86 uW
retlw	0x70	; Low Byte, ADRES = 252, Forward Power = 28.37 uW

retlw	0x14	; Low Byte, ADRES = 256, Forward Power = 31.13 uW
retlw	0x7A	; Low Byte, ADRES = 260, Forward Power = 34.15 uW
retlw	0xE9	; Low Byte, ADRES = 264, Forward Power = 37.47 uW
retlw	0x63	; Low Byte, ADRES = 268, Forward Power = 41.10 uW
retlw	0xE9	; Low Byte, ADRES = 272, Forward Power = 45.10 uW
retlw	0x7C	; Low Byte, ADRES = 276, Forward Power = 49.48 uW
retlw	0x1D	; Low Byte, ADRES = 280, Forward Power = 54.28 uW
retlw	0xCE	; Low Byte, ADRES = 284, Forward Power = 59.55 uW
retlw	0x48	; Low Byte, ADRES = 288, Forward Power = 65.33 uW
retlw	0xB3	; Low Byte, ADRES = 292, Forward Power = 71.68 uW
retlw	0x27	; Low Byte, ADRES = 296, Forward Power = 78.64 uW
retlw	0xA8	; Low Byte, ADRES = 300, Forward Power = 86.28 uW
retlw	0x34	; Low Byte, ADRES = 304, Forward Power = 94.66 uW
retlw	0xCE	; Low Byte, ADRES = 308, Forward Power = 103.8 uW
retlw	0x78	; Low Byte, ADRES = 312, Forward Power = 113.9 uW
retlw	0x19	; Low Byte, ADRES = 316, Forward Power = 125.0 uW
retlw	0x7E	; Low Byte, ADRES = 320, Forward Power = 137.1 uW
retlw	0xEE	; Low Byte, ADRES = 324, Forward Power = 150.5 uW
retlw	0x69	; Low Byte, ADRES = 328, Forward Power = 165.1 uW
retlw	0xEF	; Low Byte, ADRES = 332, Forward Power = 181.1 uW
retlw	0x83	; Low Byte, ADRES = 336, Forward Power = 198.7 uW
retlw	0x25	; Low Byte, ADRES = 340, Forward Power = 218.0 uW
retlw	0xD6	; Low Byte, ADRES = 344, Forward Power = 239.2 uW
retlw	0x4D	; Low Byte, ADRES = 348, Forward Power = 262.4 uW
retlw	0xB7	; Low Byte, ADRES = 352, Forward Power = 287.9 uW
retlw	0x2D	; Low Byte, ADRES = 356, Forward Power = 315.8 uW
retlw	0xAD	; Low Byte, ADRES = 360, Forward Power = 346.5 uW
retlw	0x3A	; Low Byte, ADRES = 364, Forward Power = 380.1 uW
retlw	0xD5	; Low Byte, ADRES = 368, Forward Power = 417.0 uW
retlw	0x7F	; Low Byte, ADRES = 372, Forward Power = 457.5 uW
retlw	0x1D	; Low Byte, ADRES = 376, Forward Power = 502.0 uW
retlw	0x83	; Low Byte, ADRES = 380, Forward Power = 550.7 uW
retlw	0xF3	; Low Byte, ADRES = 384, Forward Power = 604.2 uW
retlw	0x6E	; Low Byte, ADRES = 388, Forward Power = 662.9 uW
retlw	0xF5	; Low Byte, ADRES = 392, Forward Power = 727.3 uW
retlw	0x89	; Low Byte, ADRES = 396, Forward Power = 797.9 uW
retlw	0x2C	; Low Byte, ADRES = 400, Forward Power = 875.4 uW
retlw	0xDE	; Low Byte, ADRES = 404, Forward Power = 960.4 uW
retlw	0x51	; Low Byte, ADRES = 408, Forward Power = 1.054 mW
retlw	0xBC	; Low Byte, ADRES = 412, Forward Power = 1.156 mW
retlw	0x32	; Low Byte, ADRES = 416, Forward Power = 1.268 mW
retlw	0xB3	; Low Byte, ADRES = 420, Forward Power = 1.391 mW
retlw	0x41	; Low Byte, ADRES = 424, Forward Power = 1.527 mW
retlw	0xDC	; Low Byte, ADRES = 428, Forward Power = 1.675 mW
retlw	0x87	; Low Byte, ADRES = 432, Forward Power = 1.837 mW
retlw	0x21	; Low Byte, ADRES = 436, Forward Power = 2.016 mW
retlw	0x88	; Low Byte, ADRES = 440, Forward Power = 2.212 mW
retlw	0xF8	; Low Byte, ADRES = 444, Forward Power = 2.426 mW
retlw	0x74	; Low Byte, ADRES = 448, Forward Power = 2.662 mW
retlw	0xFB	; Low Byte, ADRES = 452, Forward Power = 2.921 mW
retlw	0x90	; Low Byte, ADRES = 456, Forward Power = 3.204 mW
retlw	0x33	; Low Byte, ADRES = 460, Forward Power = 3.515 mW
retlw	0xE6	; Low Byte, ADRES = 464, Forward Power = 3.857 mW
retlw	0x55	; Low Byte, ADRES = 468, Forward Power = 4.231 mW
retlw	0xC1	; Low Byte, ADRES = 472, Forward Power = 4.642 mW
retlw	0x37	; Low Byte, ADRES = 476, Forward Power = 5.093 mW
retlw	0xB9	; Low Byte, ADRES = 480, Forward Power = 5.588 mW
retlw	0x47	; Low Byte, ADRES = 484, Forward Power = 6.130 mW
retlw	0xE3	; Low Byte, ADRES = 488, Forward Power = 6.726 mW
retlw	0x8E	; Low Byte, ADRES = 492, Forward Power = 7.379 mW
retlw	0x25	; Low Byte, ADRES = 496, Forward Power = 8.096 mW
retlw	0x8C	; Low Byte, ADRES = 500, Forward Power = 8.882 mW
retlw	0xFD	; Low Byte, ADRES = 504, Forward Power = 9.744 mW
retlw	0x79	; Low Byte, ADRES = 508, Forward Power = 10.69 mW
retlw	0x01	; Low Byte, ADRES = 512, Forward Power = 11.73 mW
retlw	0x97	; Low Byte, ADRES = 516, Forward Power = 12.87 mW
retlw	0x3A	; Low Byte, ADRES = 520, Forward Power = 14.12 mW
retlw	0xEE	; Low Byte, ADRES = 524, Forward Power = 15.49 mW
retlw	0x5A	; Low Byte, ADRES = 528, Forward Power = 16.99 mW
retlw	0xC6	; Low Byte, ADRES = 532, Forward Power = 18.64 mW
retlw	0x3C	; Low Byte, ADRES = 536, Forward Power = 20.45 mW
retlw	0xBF	; Low Byte, ADRES = 540, Forward Power = 22.44 mW
retlw	0x4D	; Low Byte, ADRES = 544, Forward Power = 24.62 mW
retlw	0xEA	; Low Byte, ADRES = 548, Forward Power = 27.01 mW
retlw	0x96	; Low Byte, ADRES = 552, Forward Power = 29.63 mW
retlw	0x29	; Low Byte, ADRES = 556, Forward Power = 32.51 mW
retlw	0x91	; Low Byte, ADRES = 560, Forward Power = 35.67 mW
retlw	0x02	; Low Byte, ADRES = 564, Forward Power = 39.13 mW
retlw	0x7F	; Low Byte, ADRES = 568, Forward Power = 42.93 mW
retlw	0x07	; Low Byte, ADRES = 572, Forward Power = 47.10 mW
retlw	0x9D	; Low Byte, ADRES = 576, Forward Power = 51.68 mW

retlw	0x42	; Low Byte, ADRES = 580, Forward Power = 56.69 mW
retlw	0xF6	; Low Byte, ADRES = 584, Forward Power = 62.20 mW
retlw	0x5E	; Low Byte, ADRES = 588, Forward Power = 68.24 mW
retlw	0xCB	; Low Byte, ADRES = 592, Forward Power = 74.87 mW
retlw	0x42	; Low Byte, ADRES = 596, Forward Power = 82.14 mW
retlw	0xC4	; Low Byte, ADRES = 600, Forward Power = 90.12 mW
retlw	0x54	; Low Byte, ADRES = 604, Forward Power = 98.87 mW
retlw	0xF1	; Low Byte, ADRES = 608, Forward Power = 108.5 mW
retlw	0x9E	; Low Byte, ADRES = 612, Forward Power = 119.0 mW
retlw	0x2E	; Low Byte, ADRES = 616, Forward Power = 130.6 mW
retlw	0x95	; Low Byte, ADRES = 620, Forward Power = 143.2 mW
retlw	0x07	; Low Byte, ADRES = 624, Forward Power = 157.1 mW
retlw	0x84	; Low Byte, ADRES = 628, Forward Power = 172.4 mW
retlw	0x0E	; Low Byte, ADRES = 632, Forward Power = 189.2 mW
retlw	0xA4	; Low Byte, ADRES = 636, Forward Power = 207.5 mW
retlw	0x49	; Low Byte, ADRES = 640, Forward Power = 227.7 mW
retlw	0xFE	; Low Byte, ADRES = 644, Forward Power = 249.8 mW
retlw	0x63	; Low Byte, ADRES = 648, Forward Power = 274.0 mW
retlw	0xD0	; Low Byte, ADRES = 652, Forward Power = 300.7 mW
retlw	0x47	; Low Byte, ADRES = 656, Forward Power = 329.9 mW
retlw	0xCA	; Low Byte, ADRES = 660, Forward Power = 361.9 mW
retlw	0x5A	; Low Byte, ADRES = 664, Forward Power = 397.0 mW
retlw	0xF8	; Low Byte, ADRES = 668, Forward Power = 435.6 mW
retlw	0xA5	; Low Byte, ADRES = 672, Forward Power = 477.9 mW
retlw	0x32	; Low Byte, ADRES = 676, Forward Power = 524.3 mW
retlw	0x9A	; Low Byte, ADRES = 680, Forward Power = 575.2 mW
retlw	0x0C	; Low Byte, ADRES = 684, Forward Power = 631.1 mW
retlw	0x8A	; Low Byte, ADRES = 688, Forward Power = 692.4 mW
retlw	0x14	; Low Byte, ADRES = 692, Forward Power = 759.6 mW
retlw	0xAB	; Low Byte, ADRES = 696, Forward Power = 833.4 mW
retlw	0x51	; Low Byte, ADRES = 700, Forward Power = 914.3 mW
retlw	0x03	; Low Byte, ADRES = 704, Forward Power = 1.003 W
retlw	0x67	; Low Byte, ADRES = 708, Forward Power = 1.101 W
retlw	0xD4	; Low Byte, ADRES = 712, Forward Power = 1.207 W
retlw	0x4C	; Low Byte, ADRES = 716, Forward Power = 1.325 W
retlw	0xD0	; Low Byte, ADRES = 720, Forward Power = 1.453 W
retlw	0x61	; Low Byte, ADRES = 724, Forward Power = 1.594 W
retlw	0xFF	; Low Byte, ADRES = 728, Forward Power = 1.749 W
retlw	0xAD	; Low Byte, ADRES = 732, Forward Power = 1.919 W
retlw	0x36	; Low Byte, ADRES = 736, Forward Power = 2.106 W
retlw	0x9F	; Low Byte, ADRES = 740, Forward Power = 2.310 W
retlw	0x12	; Low Byte, ADRES = 744, Forward Power = 2.534 W
retlw	0x90	; Low Byte, ADRES = 748, Forward Power = 2.781 W
retlw	0x1A	; Low Byte, ADRES = 752, Forward Power = 3.051 W
retlw	0xB2	; Low Byte, ADRES = 756, Forward Power = 3.347 W
retlw	0x58	; Low Byte, ADRES = 760, Forward Power = 3.672 W
retlw	0x07	; Low Byte, ADRES = 764, Forward Power = 4.028 W
retlw	0x6B	; Low Byte, ADRES = 768, Forward Power = 4.420 W
retlw	0xD9	; Low Byte, ADRES = 772, Forward Power = 4.849 W
retlw	0x52	; Low Byte, ADRES = 776, Forward Power = 5.320 W
retlw	0xD6	; Low Byte, ADRES = 780, Forward Power = 5.836 W
retlw	0x67	; Low Byte, ADRES = 784, Forward Power = 6.403 W
retlw	0x06	; Low Byte, ADRES = 788, Forward Power = 7.025 W
retlw	0xB5	; Low Byte, ADRES = 792, Forward Power = 7.707 W
retlw	0x3A	; Low Byte, ADRES = 796, Forward Power = 8.456 W
retlw	0xA3	; Low Byte, ADRES = 800, Forward Power = 9.277 W
retlw	0x17	; Low Byte, ADRES = 804, Forward Power = 10.18 W
retlw	0x95	; Low Byte, ADRES = 808, Forward Power = 11.17 W
retlw	0x20	; Low Byte, ADRES = 812, Forward Power = 12.25 W
retlw	0xB8	; Low Byte, ADRES = 816, Forward Power = 13.44 W
retlw	0x5F	; Low Byte, ADRES = 820, Forward Power = 14.75 W
retlw	0x0B	; Low Byte, ADRES = 824, Forward Power = 16.18 W
retlw	0x70	; Low Byte, ADRES = 828, Forward Power = 17.75 W
retlw	0xDE	; Low Byte, ADRES = 832, Forward Power = 19.47 W
retlw	0x57	; Low Byte, ADRES = 836, Forward Power = 21.36 W
retlw	0xDC	; Low Byte, ADRES = 840, Forward Power = 23.44 W
retlw	0x6E	; Low Byte, ADRES = 844, Forward Power = 25.71 W
retlw	0x0E	; Low Byte, ADRES = 848, Forward Power = 28.21 W
retlw	0xBD	; Low Byte, ADRES = 852, Forward Power = 30.95 W
retlw	0x3F	; Low Byte, ADRES = 856, Forward Power = 33.96 W
retlw	0xA8	; Low Byte, ADRES = 860, Forward Power = 37.25 W
retlw	0x1C	; Low Byte, ADRES = 864, Forward Power = 40.87 W
retlw	0x9B	; Low Byte, ADRES = 868, Forward Power = 44.84 W
retlw	0x26	; Low Byte, ADRES = 872, Forward Power = 49.20 W
retlw	0xBF	; Low Byte, ADRES = 876, Forward Power = 53.97 W
retlw	0x67	; Low Byte, ADRES = 880, Forward Power = 59.22 W
retlw	0x0F	; Low Byte, ADRES = 884, Forward Power = 64.97 W
retlw	0x74	; Low Byte, ADRES = 888, Forward Power = 71.28 W
retlw	0xE3	; Low Byte, ADRES = 892, Forward Power = 78.20 W
retlw	0x5D	; Low Byte, ADRES = 896, Forward Power = 85.79 W
retlw	0xE2	; Low Byte, ADRES = 900, Forward Power = 94.12 W

```

retlw    0x74    ; Low Byte, ADRES = 904, Forward Power = 103.3 W
retlw    0x15    ; Low Byte, ADRES = 908, Forward Power = 113.3 W
retlw    0xC5    ; Low Byte, ADRES = 912, Forward Power = 124.3 W
retlw    0x43    ; Low Byte, ADRES = 916, Forward Power = 136.4 W
retlw    0xAD    ; Low Byte, ADRES = 920, Forward Power = 149.6 W
retlw    0x21    ; Low Byte, ADRES = 924, Forward Power = 164.1 W
retlw    0xA1    ; Low Byte, ADRES = 928, Forward Power = 180.1 W
retlw    0x2D    ; Low Byte, ADRES = 932, Forward Power = 197.6 W
retlw    0xC6    ; Low Byte, ADRES = 936, Forward Power = 216.8 W
retlw    0x6E    ; Low Byte, ADRES = 940, Forward Power = 237.8 W
retlw    0x14    ; Low Byte, ADRES = 944, Forward Power = 260.9 W
retlw    0x79    ; Low Byte, ADRES = 948, Forward Power = 286.2 W
retlw    0xE8    ; Low Byte, ADRES = 952, Forward Power = 314.0 W
retlw    0x62    ; Low Byte, ADRES = 956, Forward Power = 344.5 W
retlw    0xE8    ; Low Byte, ADRES = 960, Forward Power = 378.0 W
retlw    0x7B    ; Low Byte, ADRES = 964, Forward Power = 414.7 W
retlw    0x1C    ; Low Byte, ADRES = 968, Forward Power = 455.0 W
retlw    0xCD    ; Low Byte, ADRES = 972, Forward Power = 499.2 W
retlw    0x47    ; Low Byte, ADRES = 976, Forward Power = 547.6 W
retlw    0xB2    ; Low Byte, ADRES = 980, Forward Power = 600.8 W
retlw    0x26    ; Low Byte, ADRES = 984, Forward Power = 659.2 W
retlw    0xA6    ; Low Byte, ADRES = 988, Forward Power = 723.2 W
retlw    0x33    ; Low Byte, ADRES = 992, Forward Power = 793.4 W
retlw    0xCD    ; Low Byte, ADRES = 996, Forward Power = 870.5 W
retlw    0x76    ; Low Byte, ADRES = 1000, Forward Power = 955.0 W
retlw    0x18    ; Low Byte, ADRES = 1004, Forward Power = 1,047.7 W
retlw    0x7E    ; Low Byte, ADRES = 1008, Forward Power = 1,149.5 W
retlw    0xED    ; Low Byte, ADRES = 1012, Forward Power = 1,261.1 W
retlw    0x68    ; Low Byte, ADRES = 1016, Forward Power = 1,383.6 W
retlw    0xEE    ; Low Byte, ADRES = 1020, Forward Power = 1,518.0 W

```

FLOWByteTable1

```

retlw    0xAC    ; Low Byte, ADRES = 1, Forward Power = 84.54 nW
retlw    0x39    ; Low Byte, ADRES = 5, Forward Power = 92.75 nW
retlw    0xD4    ; Low Byte, ADRES = 9, Forward Power = 101.8 nW
retlw    0x7E    ; Low Byte, ADRES = 13, Forward Power = 111.6 nW
retlw    0x1C    ; Low Byte, ADRES = 17, Forward Power = 122.5 nW
retlw    0x82    ; Low Byte, ADRES = 21, Forward Power = 134.4 nW
retlw    0xF2    ; Low Byte, ADRES = 25, Forward Power = 147.4 nW
retlw    0x6D    ; Low Byte, ADRES = 29, Forward Power = 161.7 nW
retlw    0xF4    ; Low Byte, ADRES = 33, Forward Power = 177.5 nW
retlw    0x88    ; Low Byte, ADRES = 37, Forward Power = 194.7 nW
retlw    0x2B    ; Low Byte, ADRES = 41, Forward Power = 213.6 nW
retlw    0xDD    ; Low Byte, ADRES = 45, Forward Power = 234.3 nW
retlw    0x50    ; Low Byte, ADRES = 49, Forward Power = 257.1 nW
retlw    0xBB    ; Low Byte, ADRES = 53, Forward Power = 282.1 nW
retlw    0x31    ; Low Byte, ADRES = 57, Forward Power = 309.5 nW
retlw    0xB2    ; Low Byte, ADRES = 61, Forward Power = 339.5 nW
retlw    0x40    ; Low Byte, ADRES = 65, Forward Power = 372.5 nW
retlw    0xDB    ; Low Byte, ADRES = 69, Forward Power = 408.6 nW
retlw    0x86    ; Low Byte, ADRES = 73, Forward Power = 448.3 nW
retlw    0x20    ; Low Byte, ADRES = 77, Forward Power = 491.9 nW
retlw    0x87    ; Low Byte, ADRES = 81, Forward Power = 539.6 nW
retlw    0xF7    ; Low Byte, ADRES = 85, Forward Power = 592.0 nW
retlw    0x73    ; Low Byte, ADRES = 89, Forward Power = 649.5 nW
retlw    0xFA    ; Low Byte, ADRES = 93, Forward Power = 712.6 nW
retlw    0x8F    ; Low Byte, ADRES = 97, Forward Power = 781.8 nW
retlw    0x32    ; Low Byte, ADRES = 101, Forward Power = 857.8 nW
retlw    0xE5    ; Low Byte, ADRES = 105, Forward Power = 941.1 nW
retlw    0x55    ; Low Byte, ADRES = 109, Forward Power = 1.032 uW
retlw    0xC0    ; Low Byte, ADRES = 113, Forward Power = 1.133 uW
retlw    0x36    ; Low Byte, ADRES = 117, Forward Power = 1.243 uW
retlw    0xB8    ; Low Byte, ADRES = 121, Forward Power = 1.363 uW
retlw    0x46    ; Low Byte, ADRES = 125, Forward Power = 1.496 uW
retlw    0xE2    ; Low Byte, ADRES = 129, Forward Power = 1.641 uW
retlw    0x8D    ; Low Byte, ADRES = 133, Forward Power = 1.800 uW
retlw    0x24    ; Low Byte, ADRES = 137, Forward Power = 1.975 uW
retlw    0x8B    ; Low Byte, ADRES = 141, Forward Power = 2.167 uW
retlw    0xFC    ; Low Byte, ADRES = 145, Forward Power = 2.378 uW
retlw    0x78    ; Low Byte, ADRES = 149, Forward Power = 2.608 uW
retlw    0x00    ; Low Byte, ADRES = 153, Forward Power = 2.862 uW
retlw    0x96    ; Low Byte, ADRES = 157, Forward Power = 3.140 uW
retlw    0x39    ; Low Byte, ADRES = 161, Forward Power = 3.445 uW
retlw    0xED    ; Low Byte, ADRES = 165, Forward Power = 3.779 uW
retlw    0x59    ; Low Byte, ADRES = 169, Forward Power = 4.146 uW
retlw    0xC5    ; Low Byte, ADRES = 173, Forward Power = 4.549 uW
retlw    0x3C    ; Low Byte, ADRES = 177, Forward Power = 4.991 uW
retlw    0xBE    ; Low Byte, ADRES = 181, Forward Power = 5.475 uW
retlw    0x4C    ; Low Byte, ADRES = 185, Forward Power = 6.007 uW
retlw    0xE9    ; Low Byte, ADRES = 189, Forward Power = 6.590 uW
retlw    0x95    ; Low Byte, ADRES = 193, Forward Power = 7.230 uW

```

retlw	0x29	; Low Byte, ADRES = 197, Forward Power = 7.933 uW
retlw	0x90	; Low Byte, ADRES = 201, Forward Power = 8.703 uW
retlw	0x02	; Low Byte, ADRES = 205, Forward Power = 9.548 uW
retlw	0x7E	; Low Byte, ADRES = 209, Forward Power = 10.48 uW
retlw	0x07	; Low Byte, ADRES = 213, Forward Power = 11.49 uW
retlw	0x9C	; Low Byte, ADRES = 217, Forward Power = 12.61 uW
retlw	0x41	; Low Byte, ADRES = 221, Forward Power = 13.83 uW
retlw	0xF5	; Low Byte, ADRES = 225, Forward Power = 15.18 uW
retlw	0x5D	; Low Byte, ADRES = 229, Forward Power = 16.65 uW
retlw	0xCA	; Low Byte, ADRES = 233, Forward Power = 18.27 uW
retlw	0x41	; Low Byte, ADRES = 237, Forward Power = 20.04 uW
retlw	0xC4	; Low Byte, ADRES = 241, Forward Power = 21.99 uW
retlw	0x53	; Low Byte, ADRES = 245, Forward Power = 24.12 uW
retlw	0xF0	; Low Byte, ADRES = 249, Forward Power = 26.47 uW
retlw	0x9D	; Low Byte, ADRES = 253, Forward Power = 29.04 uW
retlw	0x2D	; Low Byte, ADRES = 257, Forward Power = 31.86 uW
retlw	0x95	; Low Byte, ADRES = 261, Forward Power = 34.95 uW
retlw	0x07	; Low Byte, ADRES = 265, Forward Power = 38.34 uW
retlw	0x84	; Low Byte, ADRES = 269, Forward Power = 42.07 uW
retlw	0x0D	; Low Byte, ADRES = 273, Forward Power = 46.15 uW
retlw	0xA3	; Low Byte, ADRES = 277, Forward Power = 50.64 uW
retlw	0x48	; Low Byte, ADRES = 281, Forward Power = 55.55 uW
retlw	0xFD	; Low Byte, ADRES = 285, Forward Power = 60.95 uW
retlw	0x62	; Low Byte, ADRES = 289, Forward Power = 66.87 uW
retlw	0xCF	; Low Byte, ADRES = 293, Forward Power = 73.36 uW
retlw	0x46	; Low Byte, ADRES = 297, Forward Power = 80.48 uW
retlw	0xC9	; Low Byte, ADRES = 301, Forward Power = 88.30 uW
retlw	0x59	; Low Byte, ADRES = 305, Forward Power = 96.88 uW
retlw	0xF7	; Low Byte, ADRES = 309, Forward Power = 106.3 uW
retlw	0xA4	; Low Byte, ADRES = 313, Forward Power = 116.6 uW
retlw	0x31	; Low Byte, ADRES = 317, Forward Power = 127.9 uW
retlw	0x99	; Low Byte, ADRES = 321, Forward Power = 140.4 uW
retlw	0x0C	; Low Byte, ADRES = 325, Forward Power = 154.0 uW
retlw	0x89	; Low Byte, ADRES = 329, Forward Power = 168.9 uW
retlw	0x13	; Low Byte, ADRES = 333, Forward Power = 185.3 uW
retlw	0xAA	; Low Byte, ADRES = 337, Forward Power = 203.3 uW
retlw	0x4F	; Low Byte, ADRES = 341, Forward Power = 223.1 uW
retlw	0x03	; Low Byte, ADRES = 345, Forward Power = 244.8 uW
retlw	0x66	; Low Byte, ADRES = 349, Forward Power = 268.5 uW
retlw	0xD4	; Low Byte, ADRES = 353, Forward Power = 294.6 uW
retlw	0x4C	; Low Byte, ADRES = 357, Forward Power = 323.2 uW
retlw	0xCF	; Low Byte, ADRES = 361, Forward Power = 354.6 uW
retlw	0x60	; Low Byte, ADRES = 365, Forward Power = 389.0 uW
retlw	0xFE	; Low Byte, ADRES = 369, Forward Power = 426.8 uW
retlw	0xAC	; Low Byte, ADRES = 373, Forward Power = 468.3 uW
retlw	0x35	; Low Byte, ADRES = 377, Forward Power = 513.7 uW
retlw	0x9E	; Low Byte, ADRES = 381, Forward Power = 563.6 uW
retlw	0x11	; Low Byte, ADRES = 385, Forward Power = 618.4 uW
retlw	0x8F	; Low Byte, ADRES = 389, Forward Power = 678.4 uW
retlw	0x19	; Low Byte, ADRES = 393, Forward Power = 744.3 uW
retlw	0xB1	; Low Byte, ADRES = 397, Forward Power = 816.6 uW
retlw	0x57	; Low Byte, ADRES = 401, Forward Power = 895.9 uW
retlw	0x07	; Low Byte, ADRES = 405, Forward Power = 982.9 uW
retlw	0x6B	; Low Byte, ADRES = 409, Forward Power = 1.078 mW
retlw	0xD9	; Low Byte, ADRES = 413, Forward Power = 1.183 mW
retlw	0x51	; Low Byte, ADRES = 417, Forward Power = 1.298 mW
retlw	0xD5	; Low Byte, ADRES = 421, Forward Power = 1.424 mW
retlw	0x66	; Low Byte, ADRES = 425, Forward Power = 1.562 mW
retlw	0x05	; Low Byte, ADRES = 429, Forward Power = 1.714 mW
retlw	0xB4	; Low Byte, ADRES = 433, Forward Power = 1.881 mW
retlw	0x3A	; Low Byte, ADRES = 437, Forward Power = 2.063 mW
retlw	0xA3	; Low Byte, ADRES = 441, Forward Power = 2.264 mW
retlw	0x16	; Low Byte, ADRES = 445, Forward Power = 2.483 mW
retlw	0x94	; Low Byte, ADRES = 449, Forward Power = 2.725 mW
retlw	0x1F	; Low Byte, ADRES = 453, Forward Power = 2.989 mW
retlw	0xB7	; Low Byte, ADRES = 457, Forward Power = 3.279 mW
retlw	0x5E	; Low Byte, ADRES = 461, Forward Power = 3.598 mW
retlw	0x0B	; Low Byte, ADRES = 465, Forward Power = 3.947 mW
retlw	0x6F	; Low Byte, ADRES = 469, Forward Power = 4.331 mW
retlw	0xDE	; Low Byte, ADRES = 473, Forward Power = 4.751 mW
retlw	0x56	; Low Byte, ADRES = 477, Forward Power = 5.213 mW
retlw	0xDB	; Low Byte, ADRES = 481, Forward Power = 5.719 mW
retlw	0x6D	; Low Byte, ADRES = 485, Forward Power = 6.274 mW
retlw	0x0C	; Low Byte, ADRES = 489, Forward Power = 6.883 mW
retlw	0xBC	; Low Byte, ADRES = 493, Forward Power = 7.552 mW
retlw	0x3E	; Low Byte, ADRES = 497, Forward Power = 8.285 mW
retlw	0xA7	; Low Byte, ADRES = 501, Forward Power = 9.090 mW
retlw	0x1B	; Low Byte, ADRES = 505, Forward Power = 9.973 mW
retlw	0x9A	; Low Byte, ADRES = 509, Forward Power = 10.94 mW
retlw	0x25	; Low Byte, ADRES = 513, Forward Power = 12.00 mW
retlw	0xBE	; Low Byte, ADRES = 517, Forward Power = 13.17 mW

retlw	0x66	; Low Byte, ADRES = 521, Forward Power = 14.45 mW
retlw	0x0F	; Low Byte, ADRES = 525, Forward Power = 15.85 mW
retlw	0x74	; Low Byte, ADRES = 529, Forward Power = 17.39 mW
retlw	0xE2	; Low Byte, ADRES = 533, Forward Power = 19.08 mW
retlw	0x5C	; Low Byte, ADRES = 537, Forward Power = 20.93 mW
retlw	0xE1	; Low Byte, ADRES = 541, Forward Power = 22.97 mW
retlw	0x73	; Low Byte, ADRES = 545, Forward Power = 25.20 mW
retlw	0x14	; Low Byte, ADRES = 549, Forward Power = 27.64 mW
retlw	0xC4	; Low Byte, ADRES = 553, Forward Power = 30.33 mW
retlw	0x42	; Low Byte, ADRES = 557, Forward Power = 33.27 mW
retlw	0xAC	; Low Byte, ADRES = 561, Forward Power = 36.50 mW
retlw	0x20	; Low Byte, ADRES = 565, Forward Power = 40.05 mW
retlw	0xA0	; Low Byte, ADRES = 569, Forward Power = 43.94 mW
retlw	0x2C	; Low Byte, ADRES = 573, Forward Power = 48.21 mW
retlw	0xC5	; Low Byte, ADRES = 577, Forward Power = 52.89 mW
retlw	0x6D	; Low Byte, ADRES = 581, Forward Power = 58.02 mW
retlw	0x13	; Low Byte, ADRES = 585, Forward Power = 63.66 mW
retlw	0x78	; Low Byte, ADRES = 589, Forward Power = 69.84 mW
retlw	0xE7	; Low Byte, ADRES = 593, Forward Power = 76.62 mW
retlw	0x61	; Low Byte, ADRES = 597, Forward Power = 84.06 mW
retlw	0xE7	; Low Byte, ADRES = 601, Forward Power = 92.23 mW
retlw	0x7A	; Low Byte, ADRES = 605, Forward Power = 101.2 mW
retlw	0x1B	; Low Byte, ADRES = 609, Forward Power = 111.0 mW
retlw	0xCB	; Low Byte, ADRES = 613, Forward Power = 121.8 mW
retlw	0x47	; Low Byte, ADRES = 617, Forward Power = 133.6 mW
retlw	0xB1	; Low Byte, ADRES = 621, Forward Power = 146.6 mW
retlw	0x26	; Low Byte, ADRES = 625, Forward Power = 160.8 mW
retlw	0xA6	; Low Byte, ADRES = 629, Forward Power = 176.5 mW
retlw	0x32	; Low Byte, ADRES = 633, Forward Power = 193.6 mW
retlw	0xCC	; Low Byte, ADRES = 637, Forward Power = 212.4 mW
retlw	0x75	; Low Byte, ADRES = 641, Forward Power = 233.0 mW
retlw	0x17	; Low Byte, ADRES = 645, Forward Power = 255.6 mW
retlw	0x7D	; Low Byte, ADRES = 649, Forward Power = 280.5 mW
retlw	0xEC	; Low Byte, ADRES = 653, Forward Power = 307.7 mW
retlw	0x67	; Low Byte, ADRES = 657, Forward Power = 337.6 mW
retlw	0xED	; Low Byte, ADRES = 661, Forward Power = 370.4 mW
retlw	0x80	; Low Byte, ADRES = 665, Forward Power = 406.3 mW
retlw	0x22	; Low Byte, ADRES = 669, Forward Power = 445.8 mW
retlw	0xD3	; Low Byte, ADRES = 673, Forward Power = 489.1 mW
retlw	0x4B	; Low Byte, ADRES = 677, Forward Power = 536.6 mW
retlw	0xB6	; Low Byte, ADRES = 681, Forward Power = 588.7 mW
retlw	0x2B	; Low Byte, ADRES = 685, Forward Power = 645.9 mW
retlw	0xAB	; Low Byte, ADRES = 689, Forward Power = 708.6 mW
retlw	0x38	; Low Byte, ADRES = 693, Forward Power = 777.4 mW
retlw	0xD3	; Low Byte, ADRES = 697, Forward Power = 852.9 mW
retlw	0x7C	; Low Byte, ADRES = 701, Forward Power = 935.8 mW
retlw	0x1B	; Low Byte, ADRES = 705, Forward Power = 1.027 W
retlw	0x81	; Low Byte, ADRES = 709, Forward Power = 1.126 W
retlw	0xF1	; Low Byte, ADRES = 713, Forward Power = 1.236 W
retlw	0x6C	; Low Byte, ADRES = 717, Forward Power = 1.356 W
retlw	0xF3	; Low Byte, ADRES = 721, Forward Power = 1.487 W
retlw	0x87	; Low Byte, ADRES = 725, Forward Power = 1.632 W
retlw	0x29	; Low Byte, ADRES = 729, Forward Power = 1.790 W
retlw	0xDB	; Low Byte, ADRES = 733, Forward Power = 1.964 W
retlw	0x4F	; Low Byte, ADRES = 737, Forward Power = 2.155 W
retlw	0xBA	; Low Byte, ADRES = 741, Forward Power = 2.364 W
retlw	0x30	; Low Byte, ADRES = 745, Forward Power = 2.594 W
retlw	0xB1	; Low Byte, ADRES = 749, Forward Power = 2.846 W
retlw	0x3F	; Low Byte, ADRES = 753, Forward Power = 3.122 W
retlw	0xDA	; Low Byte, ADRES = 757, Forward Power = 3.425 W
retlw	0x84	; Low Byte, ADRES = 761, Forward Power = 3.758 W
retlw	0x1F	; Low Byte, ADRES = 765, Forward Power = 4.123 W
retlw	0x86	; Low Byte, ADRES = 769, Forward Power = 4.523 W
retlw	0xF6	; Low Byte, ADRES = 773, Forward Power = 4.962 W
retlw	0x72	; Low Byte, ADRES = 777, Forward Power = 5.444 W
retlw	0xF9	; Low Byte, ADRES = 781, Forward Power = 5.973 W
retlw	0x8E	; Low Byte, ADRES = 785, Forward Power = 6.553 W
retlw	0x31	; Low Byte, ADRES = 789, Forward Power = 7.190 W
retlw	0xE3	; Low Byte, ADRES = 793, Forward Power = 7.888 W
retlw	0x54	; Low Byte, ADRES = 797, Forward Power = 8.654 W
retlw	0xBF	; Low Byte, ADRES = 801, Forward Power = 9.494 W
retlw	0x35	; Low Byte, ADRES = 805, Forward Power = 10.42 W
retlw	0xB7	; Low Byte, ADRES = 809, Forward Power = 11.43 W
retlw	0x45	; Low Byte, ADRES = 813, Forward Power = 12.54 W
retlw	0xE1	; Low Byte, ADRES = 817, Forward Power = 13.76 W
retlw	0x8C	; Low Byte, ADRES = 821, Forward Power = 15.09 W
retlw	0x24	; Low Byte, ADRES = 825, Forward Power = 16.56 W
retlw	0x8B	; Low Byte, ADRES = 829, Forward Power = 18.16 W
retlw	0xFB	; Low Byte, ADRES = 833, Forward Power = 19.93 W
retlw	0x77	; Low Byte, ADRES = 837, Forward Power = 21.86 W
retlw	0xFF	; Low Byte, ADRES = 841, Forward Power = 23.99 W

retlw	0x94	; Low Byte, ADRES = 845, Forward Power = 26.32 W
retlw	0x38	; Low Byte, ADRES = 849, Forward Power = 28.87 W
retlw	0xEB	; Low Byte, ADRES = 853, Forward Power = 31.68 W
retlw	0x58	; Low Byte, ADRES = 857, Forward Power = 34.75 W
retlw	0xC4	; Low Byte, ADRES = 861, Forward Power = 38.13 W
retlw	0x3B	; Low Byte, ADRES = 865, Forward Power = 41.83 W
retlw	0xBD	; Low Byte, ADRES = 869, Forward Power = 45.89 W
retlw	0x4B	; Low Byte, ADRES = 873, Forward Power = 50.35 W
retlw	0xE8	; Low Byte, ADRES = 877, Forward Power = 55.24 W
retlw	0x93	; Low Byte, ADRES = 881, Forward Power = 60.60 W
retlw	0x28	; Low Byte, ADRES = 885, Forward Power = 66.49 W
retlw	0x8F	; Low Byte, ADRES = 889, Forward Power = 72.95 W
retlw	0x01	; Low Byte, ADRES = 893, Forward Power = 80.03 W
retlw	0x7D	; Low Byte, ADRES = 897, Forward Power = 87.80 W
retlw	0x05	; Low Byte, ADRES = 901, Forward Power = 96.33 W
retlw	0x9B	; Low Byte, ADRES = 905, Forward Power = 105.7 W
retlw	0x3F	; Low Byte, ADRES = 909, Forward Power = 115.9 W
retlw	0xF3	; Low Byte, ADRES = 913, Forward Power = 127.2 W
retlw	0x5D	; Low Byte, ADRES = 917, Forward Power = 139.6 W
retlw	0xC9	; Low Byte, ADRES = 921, Forward Power = 153.1 W
retlw	0x40	; Low Byte, ADRES = 925, Forward Power = 168.0 W
retlw	0xC2	; Low Byte, ADRES = 929, Forward Power = 184.3 W
retlw	0x52	; Low Byte, ADRES = 933, Forward Power = 202.2 W
retlw	0xEF	; Low Byte, ADRES = 937, Forward Power = 221.8 W
retlw	0x9B	; Low Byte, ADRES = 941, Forward Power = 243.4 W
retlw	0x2C	; Low Byte, ADRES = 945, Forward Power = 267.0 W
retlw	0x94	; Low Byte, ADRES = 949, Forward Power = 292.9 W
retlw	0x06	; Low Byte, ADRES = 953, Forward Power = 321.4 W
retlw	0x82	; Low Byte, ADRES = 957, Forward Power = 352.6 W
retlw	0x0B	; Low Byte, ADRES = 961, Forward Power = 386.9 W
retlw	0xA2	; Low Byte, ADRES = 965, Forward Power = 424.4 W
retlw	0x47	; Low Byte, ADRES = 969, Forward Power = 465.6 W
retlw	0xFB	; Low Byte, ADRES = 973, Forward Power = 510.9 W
retlw	0x61	; Low Byte, ADRES = 977, Forward Power = 560.5 W
retlw	0xCE	; Low Byte, ADRES = 981, Forward Power = 614.9 W
retlw	0x45	; Low Byte, ADRES = 985, Forward Power = 674.6 W
retlw	0xC8	; Low Byte, ADRES = 989, Forward Power = 740.1 W
retlw	0x58	; Low Byte, ADRES = 993, Forward Power = 812.0 W
retlw	0xF6	; Low Byte, ADRES = 997, Forward Power = 890.9 W
retlw	0xA3	; Low Byte, ADRES = 1001, Forward Power = 977.4 W
retlw	0x30	; Low Byte, ADRES = 1005, Forward Power = 1,072.3 W
retlw	0x98	; Low Byte, ADRES = 1009, Forward Power = 1,176.4 W
retlw	0x0B	; Low Byte, ADRES = 1013, Forward Power = 1,290.7 W
retlw	0x88	; Low Byte, ADRES = 1017, Forward Power = 1,416.0 W
retlw	0x12	; Low Byte, ADRES = 1021, Forward Power = 1,553.5 W

FLOWByteTable2

retlw	0xCE	; Low Byte, ADRES = 2, Forward Power = 86.52 nW
retlw	0x5F	; Low Byte, ADRES = 6, Forward Power = 94.92 nW
retlw	0xFD	; Low Byte, ADRES = 10, Forward Power = 104.1 nW
retlw	0xAB	; Low Byte, ADRES = 14, Forward Power = 114.3 nW
retlw	0x35	; Low Byte, ADRES = 18, Forward Power = 125.4 nW
retlw	0x9D	; Low Byte, ADRES = 22, Forward Power = 137.5 nW
retlw	0x10	; Low Byte, ADRES = 26, Forward Power = 150.9 nW
retlw	0x8E	; Low Byte, ADRES = 30, Forward Power = 165.5 nW
retlw	0x18	; Low Byte, ADRES = 34, Forward Power = 181.6 nW
retlw	0xB0	; Low Byte, ADRES = 38, Forward Power = 199.2 nW
retlw	0x56	; Low Byte, ADRES = 42, Forward Power = 218.6 nW
retlw	0x06	; Low Byte, ADRES = 46, Forward Power = 239.8 nW
retlw	0x6A	; Low Byte, ADRES = 50, Forward Power = 263.1 nW
retlw	0xD8	; Low Byte, ADRES = 54, Forward Power = 288.7 nW
retlw	0x50	; Low Byte, ADRES = 58, Forward Power = 316.7 nW
retlw	0xD4	; Low Byte, ADRES = 62, Forward Power = 347.5 nW
retlw	0x65	; Low Byte, ADRES = 66, Forward Power = 381.2 nW
retlw	0x04	; Low Byte, ADRES = 70, Forward Power = 418.2 nW
retlw	0xB3	; Low Byte, ADRES = 74, Forward Power = 458.8 nW
retlw	0x39	; Low Byte, ADRES = 78, Forward Power = 503.4 nW
retlw	0xA2	; Low Byte, ADRES = 82, Forward Power = 552.3 nW
retlw	0x15	; Low Byte, ADRES = 86, Forward Power = 605.9 nW
retlw	0x94	; Low Byte, ADRES = 90, Forward Power = 664.8 nW
retlw	0x1E	; Low Byte, ADRES = 94, Forward Power = 729.3 nW
retlw	0xB6	; Low Byte, ADRES = 98, Forward Power = 800.2 nW
retlw	0x5D	; Low Byte, ADRES = 102, Forward Power = 877.9 nW
retlw	0x0A	; Low Byte, ADRES = 106, Forward Power = 963.1 nW
retlw	0x6F	; Low Byte, ADRES = 110, Forward Power = 1.057 uW
retlw	0xDD	; Low Byte, ADRES = 114, Forward Power = 1.159 uW
retlw	0x56	; Low Byte, ADRES = 118, Forward Power = 1.272 uW
retlw	0xDA	; Low Byte, ADRES = 122, Forward Power = 1.395 uW
retlw	0x6C	; Low Byte, ADRES = 126, Forward Power = 1.531 uW
retlw	0x0B	; Low Byte, ADRES = 130, Forward Power = 1.680 uW
retlw	0xBB	; Low Byte, ADRES = 134, Forward Power = 1.843 uW

retlw	0x3D	; Low Byte, ADRES = 138, Forward Power = 2.022 uW
retlw	0xA7	; Low Byte, ADRES = 142, Forward Power = 2.218 uW
retlw	0x1A	; Low Byte, ADRES = 146, Forward Power = 2.433 uW
retlw	0x99	; Low Byte, ADRES = 150, Forward Power = 2.670 uW
retlw	0x24	; Low Byte, ADRES = 154, Forward Power = 2.929 uW
retlw	0xBD	; Low Byte, ADRES = 158, Forward Power = 3.213 uW
retlw	0x65	; Low Byte, ADRES = 162, Forward Power = 3.525 uW
retlw	0x0E	; Low Byte, ADRES = 166, Forward Power = 3.868 uW
retlw	0x73	; Low Byte, ADRES = 170, Forward Power = 4.243 uW
retlw	0xE2	; Low Byte, ADRES = 174, Forward Power = 4.655 uW
retlw	0x5B	; Low Byte, ADRES = 178, Forward Power = 5.108 uW
retlw	0xE0	; Low Byte, ADRES = 182, Forward Power = 5.604 uW
retlw	0x72	; Low Byte, ADRES = 186, Forward Power = 6.148 uW
retlw	0x13	; Low Byte, ADRES = 190, Forward Power = 6.745 uW
retlw	0xC2	; Low Byte, ADRES = 194, Forward Power = 7.400 uW
retlw	0x42	; Low Byte, ADRES = 198, Forward Power = 8.118 uW
retlw	0xAB	; Low Byte, ADRES = 202, Forward Power = 8.907 uW
retlw	0x20	; Low Byte, ADRES = 206, Forward Power = 9.772 uW
retlw	0x9F	; Low Byte, ADRES = 210, Forward Power = 10.72 uW
retlw	0x2B	; Low Byte, ADRES = 214, Forward Power = 11.76 uW
retlw	0xC4	; Low Byte, ADRES = 218, Forward Power = 12.90 uW
retlw	0x6C	; Low Byte, ADRES = 222, Forward Power = 14.16 uW
retlw	0x12	; Low Byte, ADRES = 226, Forward Power = 15.53 uW
retlw	0x78	; Low Byte, ADRES = 230, Forward Power = 17.04 uW
retlw	0xE7	; Low Byte, ADRES = 234, Forward Power = 18.70 uW
retlw	0x61	; Low Byte, ADRES = 238, Forward Power = 20.51 uW
retlw	0xE6	; Low Byte, ADRES = 242, Forward Power = 22.50 uW
retlw	0x79	; Low Byte, ADRES = 246, Forward Power = 24.69 uW
retlw	0x1A	; Low Byte, ADRES = 250, Forward Power = 27.09 uW
retlw	0xCA	; Low Byte, ADRES = 254, Forward Power = 29.72 uW
retlw	0x46	; Low Byte, ADRES = 258, Forward Power = 32.60 uW
retlw	0xB0	; Low Byte, ADRES = 262, Forward Power = 35.77 uW
retlw	0x25	; Low Byte, ADRES = 266, Forward Power = 39.24 uW
retlw	0xA5	; Low Byte, ADRES = 270, Forward Power = 43.05 uW
retlw	0x31	; Low Byte, ADRES = 274, Forward Power = 47.23 uW
retlw	0xCB	; Low Byte, ADRES = 278, Forward Power = 51.82 uW
retlw	0x74	; Low Byte, ADRES = 282, Forward Power = 56.85 uW
retlw	0x17	; Low Byte, ADRES = 286, Forward Power = 62.38 uW
retlw	0x7C	; Low Byte, ADRES = 290, Forward Power = 68.43 uW
retlw	0xEC	; Low Byte, ADRES = 294, Forward Power = 75.08 uW
retlw	0x66	; Low Byte, ADRES = 298, Forward Power = 82.37 uW
retlw	0xEC	; Low Byte, ADRES = 302, Forward Power = 90.37 uW
retlw	0x7F	; Low Byte, ADRES = 306, Forward Power = 99.15 uW
retlw	0x21	; Low Byte, ADRES = 310, Forward Power = 108.8 uW
retlw	0xD2	; Low Byte, ADRES = 314, Forward Power = 119.3 uW
retlw	0x4A	; Low Byte, ADRES = 318, Forward Power = 130.9 uW
retlw	0xB5	; Low Byte, ADRES = 322, Forward Power = 143.6 uW
retlw	0x2A	; Low Byte, ADRES = 326, Forward Power = 157.6 uW
retlw	0xAA	; Low Byte, ADRES = 330, Forward Power = 172.9 uW
retlw	0x37	; Low Byte, ADRES = 334, Forward Power = 189.7 uW
retlw	0xD2	; Low Byte, ADRES = 338, Forward Power = 208.1 uW
retlw	0x7B	; Low Byte, ADRES = 342, Forward Power = 228.3 uW
retlw	0x1B	; Low Byte, ADRES = 346, Forward Power = 250.5 uW
retlw	0x81	; Low Byte, ADRES = 350, Forward Power = 274.8 uW
retlw	0xF1	; Low Byte, ADRES = 354, Forward Power = 301.5 uW
retlw	0x6B	; Low Byte, ADRES = 358, Forward Power = 330.8 uW
retlw	0xF2	; Low Byte, ADRES = 362, Forward Power = 362.9 uW
retlw	0x86	; Low Byte, ADRES = 366, Forward Power = 398.2 uW
retlw	0x28	; Low Byte, ADRES = 370, Forward Power = 436.8 uW
retlw	0xDA	; Low Byte, ADRES = 374, Forward Power = 479.2 uW
retlw	0x4F	; Low Byte, ADRES = 378, Forward Power = 525.8 uW
retlw	0xBA	; Low Byte, ADRES = 382, Forward Power = 576.9 uW
retlw	0x2F	; Low Byte, ADRES = 386, Forward Power = 632.9 uW
retlw	0xB0	; Low Byte, ADRES = 390, Forward Power = 694.3 uW
retlw	0x3E	; Low Byte, ADRES = 394, Forward Power = 761.8 uW
retlw	0xD9	; Low Byte, ADRES = 398, Forward Power = 835.7 uW
retlw	0x83	; Low Byte, ADRES = 402, Forward Power = 916.9 uW
retlw	0x1F	; Low Byte, ADRES = 406, Forward Power = 1.006 mW
retlw	0x85	; Low Byte, ADRES = 410, Forward Power = 1.104 mW
retlw	0xF6	; Low Byte, ADRES = 414, Forward Power = 1.211 mW
retlw	0x71	; Low Byte, ADRES = 418, Forward Power = 1.328 mW
retlw	0xF8	; Low Byte, ADRES = 422, Forward Power = 1.457 mW
retlw	0x8D	; Low Byte, ADRES = 426, Forward Power = 1.599 mW
retlw	0x2F	; Low Byte, ADRES = 430, Forward Power = 1.754 mW
retlw	0xE2	; Low Byte, ADRES = 434, Forward Power = 1.925 mW
retlw	0x53	; Low Byte, ADRES = 438, Forward Power = 2.112 mW
retlw	0xBF	; Low Byte, ADRES = 442, Forward Power = 2.317 mW
retlw	0x35	; Low Byte, ADRES = 446, Forward Power = 2.542 mW
retlw	0xB6	; Low Byte, ADRES = 450, Forward Power = 2.788 mW
retlw	0x44	; Low Byte, ADRES = 454, Forward Power = 3.059 mW
retlw	0xE0	; Low Byte, ADRES = 458, Forward Power = 3.356 mW

retlw	0x8B	; Low Byte, ADRES = 462, Forward Power = 3.682 mW
retlw	0x23	; Low Byte, ADRES = 466, Forward Power = 4.040 mW
retlw	0x8A	; Low Byte, ADRES = 470, Forward Power = 4.432 mW
retlw	0xFB	; Low Byte, ADRES = 474, Forward Power = 4.863 mW
retlw	0x76	; Low Byte, ADRES = 478, Forward Power = 5.335 mW
retlw	0xFE	; Low Byte, ADRES = 482, Forward Power = 5.853 mW
retlw	0x93	; Low Byte, ADRES = 486, Forward Power = 6.421 mW
retlw	0x37	; Low Byte, ADRES = 490, Forward Power = 7.045 mW
retlw	0xEA	; Low Byte, ADRES = 494, Forward Power = 7.729 mW
retlw	0x57	; Low Byte, ADRES = 498, Forward Power = 8.480 mW
retlw	0xC3	; Low Byte, ADRES = 502, Forward Power = 9.303 mW
retlw	0x3A	; Low Byte, ADRES = 506, Forward Power = 10.21 mW
retlw	0xBC	; Low Byte, ADRES = 510, Forward Power = 11.20 mW
retlw	0x4A	; Low Byte, ADRES = 514, Forward Power = 12.29 mW
retlw	0xE7	; Low Byte, ADRES = 518, Forward Power = 13.48 mW
retlw	0x92	; Low Byte, ADRES = 522, Forward Power = 14.79 mW
retlw	0x27	; Low Byte, ADRES = 526, Forward Power = 16.22 mW
retlw	0x8E	; Low Byte, ADRES = 530, Forward Power = 17.80 mW
retlw	0x00	; Low Byte, ADRES = 534, Forward Power = 19.53 mW
retlw	0x7C	; Low Byte, ADRES = 538, Forward Power = 21.42 mW
retlw	0x04	; Low Byte, ADRES = 542, Forward Power = 23.50 mW
retlw	0x9A	; Low Byte, ADRES = 546, Forward Power = 25.79 mW
retlw	0x3E	; Low Byte, ADRES = 550, Forward Power = 28.29 mW
retlw	0xF2	; Low Byte, ADRES = 554, Forward Power = 31.04 mW
retlw	0x5C	; Low Byte, ADRES = 558, Forward Power = 34.05 mW
retlw	0xC8	; Low Byte, ADRES = 562, Forward Power = 37.36 mW
retlw	0x3F	; Low Byte, ADRES = 566, Forward Power = 40.99 mW
retlw	0xC2	; Low Byte, ADRES = 570, Forward Power = 44.97 mW
retlw	0x51	; Low Byte, ADRES = 574, Forward Power = 49.34 mW
retlw	0xEE	; Low Byte, ADRES = 578, Forward Power = 54.13 mW
retlw	0x9A	; Low Byte, ADRES = 582, Forward Power = 59.38 mW
retlw	0x2B	; Low Byte, ADRES = 586, Forward Power = 65.15 mW
retlw	0x93	; Low Byte, ADRES = 590, Forward Power = 71.48 mW
retlw	0x05	; Low Byte, ADRES = 594, Forward Power = 78.42 mW
retlw	0x82	; Low Byte, ADRES = 598, Forward Power = 86.03 mW
retlw	0x0B	; Low Byte, ADRES = 602, Forward Power = 94.39 mW
retlw	0xA1	; Low Byte, ADRES = 606, Forward Power = 103.6 mW
retlw	0x45	; Low Byte, ADRES = 610, Forward Power = 113.6 mW
retlw	0xFA	; Low Byte, ADRES = 614, Forward Power = 124.6 mW
retlw	0x60	; Low Byte, ADRES = 618, Forward Power = 136.8 mW
retlw	0xCD	; Low Byte, ADRES = 622, Forward Power = 150.0 mW
retlw	0x44	; Low Byte, ADRES = 626, Forward Power = 164.6 mW
retlw	0xC7	; Low Byte, ADRES = 630, Forward Power = 180.6 mW
retlw	0x57	; Low Byte, ADRES = 634, Forward Power = 198.1 mW
retlw	0xF5	; Low Byte, ADRES = 638, Forward Power = 217.4 mW
retlw	0xA2	; Low Byte, ADRES = 642, Forward Power = 238.5 mW
retlw	0x30	; Low Byte, ADRES = 646, Forward Power = 261.6 mW
retlw	0x98	; Low Byte, ADRES = 650, Forward Power = 287.0 mW
retlw	0x0A	; Low Byte, ADRES = 654, Forward Power = 314.9 mW
retlw	0x87	; Low Byte, ADRES = 658, Forward Power = 345.5 mW
retlw	0x11	; Low Byte, ADRES = 662, Forward Power = 379.1 mW
retlw	0xA7	; Low Byte, ADRES = 666, Forward Power = 415.9 mW
retlw	0x4D	; Low Byte, ADRES = 670, Forward Power = 456.3 mW
retlw	0x01	; Low Byte, ADRES = 674, Forward Power = 500.6 mW
retlw	0x65	; Low Byte, ADRES = 678, Forward Power = 549.2 mW
retlw	0xD2	; Low Byte, ADRES = 682, Forward Power = 602.5 mW
retlw	0x4A	; Low Byte, ADRES = 686, Forward Power = 661.0 mW
retlw	0xCD	; Low Byte, ADRES = 690, Forward Power = 725.2 mW
retlw	0x5E	; Low Byte, ADRES = 694, Forward Power = 795.6 mW
retlw	0xFC	; Low Byte, ADRES = 698, Forward Power = 872.9 mW
retlw	0xA9	; Low Byte, ADRES = 702, Forward Power = 957.7 mW
retlw	0x34	; Low Byte, ADRES = 706, Forward Power = 1.051 W
retlw	0x9C	; Low Byte, ADRES = 710, Forward Power = 1.153 W
retlw	0x0F	; Low Byte, ADRES = 714, Forward Power = 1.265 W
retlw	0x8D	; Low Byte, ADRES = 718, Forward Power = 1.388 W
retlw	0x17	; Low Byte, ADRES = 722, Forward Power = 1.522 W
retlw	0xAE	; Low Byte, ADRES = 726, Forward Power = 1.670 W
retlw	0x54	; Low Byte, ADRES = 730, Forward Power = 1.832 W
retlw	0x05	; Low Byte, ADRES = 734, Forward Power = 2.010 W
retlw	0x69	; Low Byte, ADRES = 738, Forward Power = 2.205 W
retlw	0xD7	; Low Byte, ADRES = 742, Forward Power = 2.420 W
retlw	0x4F	; Low Byte, ADRES = 746, Forward Power = 2.655 W
retlw	0xD3	; Low Byte, ADRES = 750, Forward Power = 2.912 W
retlw	0x64	; Low Byte, ADRES = 754, Forward Power = 3.195 W
retlw	0x03	; Low Byte, ADRES = 758, Forward Power = 3.506 W
retlw	0xB1	; Low Byte, ADRES = 762, Forward Power = 3.846 W
retlw	0x38	; Low Byte, ADRES = 766, Forward Power = 4.219 W
retlw	0xA1	; Low Byte, ADRES = 770, Forward Power = 4.629 W
retlw	0x14	; Low Byte, ADRES = 774, Forward Power = 5.079 W
retlw	0x92	; Low Byte, ADRES = 778, Forward Power = 5.572 W
retlw	0x1D	; Low Byte, ADRES = 782, Forward Power = 6.113 W

retlw	0xB5	; Low Byte, ADRES = 786, Forward Power = 6.707 W
retlw	0x5C	; Low Byte, ADRES = 790, Forward Power = 7.358 W
retlw	0x09	; Low Byte, ADRES = 794, Forward Power = 8.073 W
retlw	0x6E	; Low Byte, ADRES = 798, Forward Power = 8.857 W
retlw	0xDC	; Low Byte, ADRES = 802, Forward Power = 9.717 W
retlw	0x55	; Low Byte, ADRES = 806, Forward Power = 10.66 W
retlw	0xD9	; Low Byte, ADRES = 810, Forward Power = 11.70 W
retlw	0x6A	; Low Byte, ADRES = 814, Forward Power = 12.83 W
retlw	0x0A	; Low Byte, ADRES = 818, Forward Power = 14.08 W
retlw	0xB9	; Low Byte, ADRES = 822, Forward Power = 15.44 W
retlw	0x3C	; Low Byte, ADRES = 826, Forward Power = 16.94 W
retlw	0xA6	; Low Byte, ADRES = 830, Forward Power = 18.59 W
retlw	0x19	; Low Byte, ADRES = 834, Forward Power = 20.40 W
retlw	0x98	; Low Byte, ADRES = 838, Forward Power = 22.38 W
retlw	0x23	; Low Byte, ADRES = 842, Forward Power = 24.55 W
retlw	0xBC	; Low Byte, ADRES = 846, Forward Power = 26.93 W
retlw	0x63	; Low Byte, ADRES = 850, Forward Power = 29.55 W
retlw	0x0D	; Low Byte, ADRES = 854, Forward Power = 32.42 W
retlw	0x72	; Low Byte, ADRES = 858, Forward Power = 35.57 W
retlw	0xE1	; Low Byte, ADRES = 862, Forward Power = 39.02 W
retlw	0x5A	; Low Byte, ADRES = 866, Forward Power = 42.81 W
retlw	0xDF	; Low Byte, ADRES = 870, Forward Power = 46.97 W
retlw	0x71	; Low Byte, ADRES = 874, Forward Power = 51.53 W
retlw	0x11	; Low Byte, ADRES = 878, Forward Power = 56.53 W
retlw	0xC1	; Low Byte, ADRES = 882, Forward Power = 62.02 W
retlw	0x41	; Low Byte, ADRES = 886, Forward Power = 68.05 W
retlw	0xAB	; Low Byte, ADRES = 890, Forward Power = 74.66 W
retlw	0x1F	; Low Byte, ADRES = 894, Forward Power = 81.91 W
retlw	0x9E	; Low Byte, ADRES = 898, Forward Power = 89.86 W
retlw	0x29	; Low Byte, ADRES = 902, Forward Power = 98.59 W
retlw	0xC3	; Low Byte, ADRES = 906, Forward Power = 108.2 W
retlw	0x6B	; Low Byte, ADRES = 910, Forward Power = 118.7 W
retlw	0x12	; Low Byte, ADRES = 914, Forward Power = 130.2 W
retlw	0x77	; Low Byte, ADRES = 918, Forward Power = 142.8 W
retlw	0xE6	; Low Byte, ADRES = 922, Forward Power = 156.7 W
retlw	0x5F	; Low Byte, ADRES = 926, Forward Power = 171.9 W
retlw	0xE5	; Low Byte, ADRES = 930, Forward Power = 188.6 W
retlw	0x78	; Low Byte, ADRES = 934, Forward Power = 206.9 W
retlw	0x18	; Low Byte, ADRES = 938, Forward Power = 227.0 W
retlw	0xC9	; Low Byte, ADRES = 942, Forward Power = 249.1 W
retlw	0x45	; Low Byte, ADRES = 946, Forward Power = 273.3 W
retlw	0xAF	; Low Byte, ADRES = 950, Forward Power = 299.8 W
retlw	0x24	; Low Byte, ADRES = 954, Forward Power = 328.9 W
retlw	0xA4	; Low Byte, ADRES = 958, Forward Power = 360.9 W
retlw	0x30	; Low Byte, ADRES = 962, Forward Power = 395.9 W
retlw	0xC9	; Low Byte, ADRES = 966, Forward Power = 434.4 W
retlw	0x72	; Low Byte, ADRES = 970, Forward Power = 476.6 W
retlw	0x16	; Low Byte, ADRES = 974, Forward Power = 522.8 W
retlw	0x7B	; Low Byte, ADRES = 978, Forward Power = 573.6 W
retlw	0xEB	; Low Byte, ADRES = 982, Forward Power = 629.3 W
retlw	0x65	; Low Byte, ADRES = 986, Forward Power = 690.4 W
retlw	0xEB	; Low Byte, ADRES = 990, Forward Power = 757.5 W
retlw	0x7E	; Low Byte, ADRES = 994, Forward Power = 831.0 W
retlw	0x20	; Low Byte, ADRES = 998, Forward Power = 911.7 W
retlw	0xD1	; Low Byte, ADRES = 1002, Forward Power = 1,000.3 W
retlw	0x49	; Low Byte, ADRES = 1006, Forward Power = 1,097.4 W
retlw	0xB4	; Low Byte, ADRES = 1010, Forward Power = 1,204.0 W
retlw	0x29	; Low Byte, ADRES = 1014, Forward Power = 1,320.9 W
retlw	0xA9	; Low Byte, ADRES = 1018, Forward Power = 1,449.2 W
retlw	0x36	; Low Byte, ADRES = 1022, Forward Power = 1,590.0 W

FLowByteTable3

retlw	0xF1	; Low Byte, ADRES = 3, Forward Power = 88.55 nW
retlw	0x85	; Low Byte, ADRES = 7, Forward Power = 97.15 nW
retlw	0x27	; Low Byte, ADRES = 11, Forward Power = 106.6 nW
retlw	0xD9	; Low Byte, ADRES = 15, Forward Power = 116.9 nW
retlw	0x4E	; Low Byte, ADRES = 19, Forward Power = 128.3 nW
retlw	0xB9	; Low Byte, ADRES = 23, Forward Power = 140.8 nW
retlw	0x2E	; Low Byte, ADRES = 27, Forward Power = 154.4 nW
retlw	0xAF	; Low Byte, ADRES = 31, Forward Power = 169.4 nW
retlw	0x3D	; Low Byte, ADRES = 35, Forward Power = 185.9 nW
retlw	0xD8	; Low Byte, ADRES = 39, Forward Power = 203.9 nW
retlw	0x82	; Low Byte, ADRES = 43, Forward Power = 223.7 nW
retlw	0x1E	; Low Byte, ADRES = 47, Forward Power = 245.4 nW
retlw	0x85	; Low Byte, ADRES = 51, Forward Power = 269.3 nW
retlw	0xF5	; Low Byte, ADRES = 55, Forward Power = 295.4 nW
retlw	0x70	; Low Byte, ADRES = 59, Forward Power = 324.1 nW
retlw	0xF7	; Low Byte, ADRES = 63, Forward Power = 355.6 nW
retlw	0x8C	; Low Byte, ADRES = 67, Forward Power = 390.1 nW
retlw	0x2E	; Low Byte, ADRES = 71, Forward Power = 428.0 nW
retlw	0xE1	; Low Byte, ADRES = 75, Forward Power = 469.6 nW

retlw	0x52	; Low Byte, ADRES = 79, Forward Power = 515.2 nW
retlw	0xBE	; Low Byte, ADRES = 83, Forward Power = 565.2 nW
retlw	0x34	; Low Byte, ADRES = 87, Forward Power = 620.1 nW
retlw	0xB5	; Low Byte, ADRES = 91, Forward Power = 680.4 nW
retlw	0x43	; Low Byte, ADRES = 95, Forward Power = 746.4 nW
retlw	0xDF	; Low Byte, ADRES = 99, Forward Power = 818.9 nW
retlw	0x89	; Low Byte, ADRES = 103, Forward Power = 898.4 nW
retlw	0x22	; Low Byte, ADRES = 107, Forward Power = 985.7 nW
retlw	0x89	; Low Byte, ADRES = 111, Forward Power = 1.081 uW
retlw	0xFA	; Low Byte, ADRES = 115, Forward Power = 1.186 uW
retlw	0x76	; Low Byte, ADRES = 119, Forward Power = 1.302 uW
retlw	0xFD	; Low Byte, ADRES = 123, Forward Power = 1.428 uW
retlw	0x92	; Low Byte, ADRES = 127, Forward Power = 1.567 uW
retlw	0x36	; Low Byte, ADRES = 131, Forward Power = 1.719 uW
retlw	0xE9	; Low Byte, ADRES = 135, Forward Power = 1.886 uW
retlw	0x57	; Low Byte, ADRES = 139, Forward Power = 2.069 uW
retlw	0xC3	; Low Byte, ADRES = 143, Forward Power = 2.270 uW
retlw	0x39	; Low Byte, ADRES = 147, Forward Power = 2.490 uW
retlw	0xBB	; Low Byte, ADRES = 151, Forward Power = 2.732 uW
retlw	0x49	; Low Byte, ADRES = 155, Forward Power = 2.998 uW
retlw	0xE6	; Low Byte, ADRES = 159, Forward Power = 3.289 uW
retlw	0x91	; Low Byte, ADRES = 163, Forward Power = 3.608 uW
retlw	0x27	; Low Byte, ADRES = 167, Forward Power = 3.958 uW
retlw	0x8E	; Low Byte, ADRES = 171, Forward Power = 4.343 uW
retlw	0xFF	; Low Byte, ADRES = 175, Forward Power = 4.765 uW
retlw	0x7B	; Low Byte, ADRES = 179, Forward Power = 5.227 uW
retlw	0x03	; Low Byte, ADRES = 183, Forward Power = 5.735 uW
retlw	0x99	; Low Byte, ADRES = 187, Forward Power = 6.292 uW
retlw	0x3D	; Low Byte, ADRES = 191, Forward Power = 6.903 uW
retlw	0xF1	; Low Byte, ADRES = 195, Forward Power = 7.573 uW
retlw	0x5B	; Low Byte, ADRES = 199, Forward Power = 8.309 uW
retlw	0xC8	; Low Byte, ADRES = 203, Forward Power = 9.116 uW
retlw	0x3E	; Low Byte, ADRES = 207, Forward Power = 10.00 uW
retlw	0xC1	; Low Byte, ADRES = 211, Forward Power = 10.97 uW
retlw	0x50	; Low Byte, ADRES = 215, Forward Power = 12.04 uW
retlw	0xED	; Low Byte, ADRES = 219, Forward Power = 13.21 uW
retlw	0x99	; Low Byte, ADRES = 223, Forward Power = 14.49 uW
retlw	0x2B	; Low Byte, ADRES = 227, Forward Power = 15.90 uW
retlw	0x92	; Low Byte, ADRES = 231, Forward Power = 17.44 uW
retlw	0x04	; Low Byte, ADRES = 235, Forward Power = 19.13 uW
retlw	0x81	; Low Byte, ADRES = 239, Forward Power = 20.99 uW
retlw	0x0A	; Low Byte, ADRES = 243, Forward Power = 23.03 uW
retlw	0xA0	; Low Byte, ADRES = 247, Forward Power = 25.27 uW
retlw	0x44	; Low Byte, ADRES = 251, Forward Power = 27.72 uW
retlw	0xF9	; Low Byte, ADRES = 255, Forward Power = 30.41 uW
retlw	0x60	; Low Byte, ADRES = 259, Forward Power = 33.37 uW
retlw	0xCC	; Low Byte, ADRES = 263, Forward Power = 36.61 uW
retlw	0x44	; Low Byte, ADRES = 267, Forward Power = 40.16 uW
retlw	0xC7	; Low Byte, ADRES = 271, Forward Power = 44.06 uW
retlw	0x56	; Low Byte, ADRES = 275, Forward Power = 48.34 uW
retlw	0xF4	; Low Byte, ADRES = 279, Forward Power = 53.04 uW
retlw	0xA0	; Low Byte, ADRES = 283, Forward Power = 58.19 uW
retlw	0x2F	; Low Byte, ADRES = 287, Forward Power = 63.84 uW
retlw	0x97	; Low Byte, ADRES = 291, Forward Power = 70.04 uW
retlw	0x09	; Low Byte, ADRES = 295, Forward Power = 76.84 uW
retlw	0x86	; Low Byte, ADRES = 299, Forward Power = 84.30 uW
retlw	0x10	; Low Byte, ADRES = 303, Forward Power = 92.49 uW
retlw	0xA6	; Low Byte, ADRES = 307, Forward Power = 101.5 uW
retlw	0x4C	; Low Byte, ADRES = 311, Forward Power = 111.3 uW
retlw	0x01	; Low Byte, ADRES = 315, Forward Power = 122.1 uW
retlw	0x64	; Low Byte, ADRES = 319, Forward Power = 134.0 uW
retlw	0xD1	; Low Byte, ADRES = 323, Forward Power = 147.0 uW
retlw	0x49	; Low Byte, ADRES = 327, Forward Power = 161.3 uW
retlw	0xCC	; Low Byte, ADRES = 331, Forward Power = 177.0 uW
retlw	0x5D	; Low Byte, ADRES = 335, Forward Power = 194.1 uW
retlw	0xFB	; Low Byte, ADRES = 339, Forward Power = 213.0 uW
retlw	0xA8	; Low Byte, ADRES = 343, Forward Power = 233.7 uW
retlw	0x33	; Low Byte, ADRES = 347, Forward Power = 256.4 uW
retlw	0x9C	; Low Byte, ADRES = 351, Forward Power = 281.3 uW
retlw	0x0E	; Low Byte, ADRES = 355, Forward Power = 308.6 uW
retlw	0x8C	; Low Byte, ADRES = 359, Forward Power = 338.5 uW
retlw	0x16	; Low Byte, ADRES = 363, Forward Power = 371.4 uW
retlw	0xAD	; Low Byte, ADRES = 367, Forward Power = 407.5 uW
retlw	0x53	; Low Byte, ADRES = 371, Forward Power = 447.1 uW
retlw	0x05	; Low Byte, ADRES = 375, Forward Power = 490.5 uW
retlw	0x69	; Low Byte, ADRES = 379, Forward Power = 538.1 uW
retlw	0xD6	; Low Byte, ADRES = 383, Forward Power = 590.4 uW
retlw	0x4E	; Low Byte, ADRES = 387, Forward Power = 647.7 uW
retlw	0xD2	; Low Byte, ADRES = 391, Forward Power = 710.6 uW
retlw	0x63	; Low Byte, ADRES = 395, Forward Power = 779.6 uW
retlw	0x02	; Low Byte, ADRES = 399, Forward Power = 855.3 uW

retlw	0xB0	; Low Byte, ADRES = 403, Forward Power = 938.4 uW
retlw	0x38	; Low Byte, ADRES = 407, Forward Power = 1.030 mW
retlw	0xA0	; Low Byte, ADRES = 411, Forward Power = 1.130 mW
retlw	0x13	; Low Byte, ADRES = 415, Forward Power = 1.239 mW
retlw	0x92	; Low Byte, ADRES = 419, Forward Power = 1.360 mW
retlw	0x1C	; Low Byte, ADRES = 423, Forward Power = 1.492 mW
retlw	0xB4	; Low Byte, ADRES = 427, Forward Power = 1.636 mW
retlw	0x5B	; Low Byte, ADRES = 431, Forward Power = 1.795 mW
retlw	0x09	; Low Byte, ADRES = 435, Forward Power = 1.970 mW
retlw	0x6D	; Low Byte, ADRES = 439, Forward Power = 2.161 mW
retlw	0xDB	; Low Byte, ADRES = 443, Forward Power = 2.371 mW
retlw	0x54	; Low Byte, ADRES = 447, Forward Power = 2.601 mW
retlw	0xD8	; Low Byte, ADRES = 451, Forward Power = 2.854 mW
retlw	0x69	; Low Byte, ADRES = 455, Forward Power = 3.131 mW
retlw	0x09	; Low Byte, ADRES = 459, Forward Power = 3.435 mW
retlw	0xB8	; Low Byte, ADRES = 463, Forward Power = 3.769 mW
retlw	0x3C	; Low Byte, ADRES = 467, Forward Power = 4.134 mW
retlw	0xA5	; Low Byte, ADRES = 471, Forward Power = 4.536 mW
retlw	0x19	; Low Byte, ADRES = 475, Forward Power = 4.977 mW
retlw	0x97	; Low Byte, ADRES = 479, Forward Power = 5.460 mW
retlw	0x22	; Low Byte, ADRES = 483, Forward Power = 5.990 mW
retlw	0xBB	; Low Byte, ADRES = 487, Forward Power = 6.572 mW
retlw	0x62	; Low Byte, ADRES = 491, Forward Power = 7.210 mW
retlw	0x0D	; Low Byte, ADRES = 495, Forward Power = 7.910 mW
retlw	0x72	; Low Byte, ADRES = 499, Forward Power = 8.678 mW
retlw	0xE0	; Low Byte, ADRES = 503, Forward Power = 9.521 mW
retlw	0x59	; Low Byte, ADRES = 507, Forward Power = 10.45 mW
retlw	0xDE	; Low Byte, ADRES = 511, Forward Power = 11.46 mW
retlw	0x70	; Low Byte, ADRES = 515, Forward Power = 12.57 mW
retlw	0x10	; Low Byte, ADRES = 519, Forward Power = 13.79 mW
retlw	0xC0	; Low Byte, ADRES = 523, Forward Power = 15.13 mW
retlw	0x40	; Low Byte, ADRES = 527, Forward Power = 16.60 mW
retlw	0xAA	; Low Byte, ADRES = 531, Forward Power = 18.22 mW
retlw	0x1E	; Low Byte, ADRES = 535, Forward Power = 19.99 mW
retlw	0x9D	; Low Byte, ADRES = 539, Forward Power = 21.93 mW
retlw	0x28	; Low Byte, ADRES = 543, Forward Power = 24.06 mW
retlw	0xC2	; Low Byte, ADRES = 547, Forward Power = 26.39 mW
retlw	0x6A	; Low Byte, ADRES = 551, Forward Power = 28.95 mW
retlw	0x11	; Low Byte, ADRES = 555, Forward Power = 31.77 mW
retlw	0x76	; Low Byte, ADRES = 559, Forward Power = 34.85 mW
retlw	0xE5	; Low Byte, ADRES = 563, Forward Power = 38.24 mW
retlw	0x5F	; Low Byte, ADRES = 567, Forward Power = 41.95 mW
retlw	0xE4	; Low Byte, ADRES = 571, Forward Power = 46.02 mW
retlw	0x77	; Low Byte, ADRES = 575, Forward Power = 50.49 mW
retlw	0x17	; Low Byte, ADRES = 579, Forward Power = 55.40 mW
retlw	0xC8	; Low Byte, ADRES = 583, Forward Power = 60.78 mW
retlw	0x44	; Low Byte, ADRES = 587, Forward Power = 66.68 mW
retlw	0xAF	; Low Byte, ADRES = 591, Forward Power = 73.15 mW
retlw	0x23	; Low Byte, ADRES = 595, Forward Power = 80.26 mW
retlw	0xA3	; Low Byte, ADRES = 599, Forward Power = 88.05 mW
retlw	0x2F	; Low Byte, ADRES = 603, Forward Power = 96.60 mW
retlw	0xC8	; Low Byte, ADRES = 607, Forward Power = 106.0 mW
retlw	0x71	; Low Byte, ADRES = 611, Forward Power = 116.3 mW
retlw	0x15	; Low Byte, ADRES = 615, Forward Power = 127.6 mW
retlw	0x7B	; Low Byte, ADRES = 619, Forward Power = 140.0 mW
retlw	0xEA	; Low Byte, ADRES = 623, Forward Power = 153.5 mW
retlw	0x64	; Low Byte, ADRES = 627, Forward Power = 168.5 mW
retlw	0xEA	; Low Byte, ADRES = 631, Forward Power = 184.8 mW
retlw	0x7D	; Low Byte, ADRES = 635, Forward Power = 202.8 mW
retlw	0x1E	; Low Byte, ADRES = 639, Forward Power = 222.5 mW
retlw	0xCF	; Low Byte, ADRES = 643, Forward Power = 244.1 mW
retlw	0x49	; Low Byte, ADRES = 647, Forward Power = 267.8 mW
retlw	0xB3	; Low Byte, ADRES = 651, Forward Power = 293.8 mW
retlw	0x28	; Low Byte, ADRES = 655, Forward Power = 322.3 mW
retlw	0xA8	; Low Byte, ADRES = 659, Forward Power = 353.6 mW
retlw	0x35	; Low Byte, ADRES = 663, Forward Power = 387.9 mW
retlw	0xCF	; Low Byte, ADRES = 667, Forward Power = 425.6 mW
retlw	0x79	; Low Byte, ADRES = 671, Forward Power = 467.0 mW
retlw	0x19	; Low Byte, ADRES = 675, Forward Power = 512.3 mW
retlw	0x7F	; Low Byte, ADRES = 679, Forward Power = 562.1 mW
retlw	0xEF	; Low Byte, ADRES = 683, Forward Power = 616.6 mW
retlw	0x6A	; Low Byte, ADRES = 687, Forward Power = 676.5 mW
retlw	0xF0	; Low Byte, ADRES = 691, Forward Power = 742.2 mW
retlw	0x84	; Low Byte, ADRES = 695, Forward Power = 814.3 mW
retlw	0x26	; Low Byte, ADRES = 699, Forward Power = 893.4 mW
retlw	0xD7	; Low Byte, ADRES = 703, Forward Power = 980.1 mW
retlw	0x4D	; Low Byte, ADRES = 707, Forward Power = 1.075 W
retlw	0xB8	; Low Byte, ADRES = 711, Forward Power = 1.180 W
retlw	0x2D	; Low Byte, ADRES = 715, Forward Power = 1.294 W
retlw	0xAE	; Low Byte, ADRES = 719, Forward Power = 1.420 W
retlw	0x3B	; Low Byte, ADRES = 723, Forward Power = 1.558 W

```

retlw    0xD6      ; Low Byte, ADRES = 727, Forward Power = 1.709 W
retlw    0xD8      ; Low Byte, ADRES = 731, Forward Power = 1.875 W
retlw    0xD9      ; Low Byte, ADRES = 735, Forward Power = 2.057 W
retlw    0xDA      ; Low Byte, ADRES = 739, Forward Power = 2.257 W
retlw    0xDC      ; Low Byte, ADRES = 743, Forward Power = 2.476 W
retlw    0xDE      ; Low Byte, ADRES = 747, Forward Power = 2.717 W
retlw    0xDF      ; Low Byte, ADRES = 751, Forward Power = 2.981 W
retlw    0xE0      ; Low Byte, ADRES = 755, Forward Power = 3.270 W
retlw    0xE2      ; Low Byte, ADRES = 759, Forward Power = 3.588 W
retlw    0xE3      ; Low Byte, ADRES = 763, Forward Power = 3.936 W
retlw    0xE4      ; Low Byte, ADRES = 767, Forward Power = 4.318 W
retlw    0xE5      ; Low Byte, ADRES = 771, Forward Power = 4.738 W
retlw    0xE6      ; Low Byte, ADRES = 775, Forward Power = 5.198 W
retlw    0xE7      ; Low Byte, ADRES = 779, Forward Power = 5.703 W
retlw    0xE8      ; Low Byte, ADRES = 783, Forward Power = 6.256 W
retlw    0xEA      ; Low Byte, ADRES = 787, Forward Power = 6.864 W
retlw    0xEB      ; Low Byte, ADRES = 791, Forward Power = 7.531 W
retlw    0xEC      ; Low Byte, ADRES = 795, Forward Power = 8.262 W
retlw    0xEE      ; Low Byte, ADRES = 799, Forward Power = 9.064 W
retlw    0xEF      ; Low Byte, ADRES = 803, Forward Power = 9.945 W
retlw    0xF0      ; Low Byte, ADRES = 807, Forward Power = 10.91 W
retlw    0xF1      ; Low Byte, ADRES = 811, Forward Power = 11.97 W
retlw    0xF2      ; Low Byte, ADRES = 815, Forward Power = 13.13 W
retlw    0xF3      ; Low Byte, ADRES = 819, Forward Power = 14.41 W
retlw    0xF4      ; Low Byte, ADRES = 823, Forward Power = 15.81 W
retlw    0xF5      ; Low Byte, ADRES = 827, Forward Power = 17.34 W
retlw    0xF6      ; Low Byte, ADRES = 831, Forward Power = 19.03 W
retlw    0xF7      ; Low Byte, ADRES = 835, Forward Power = 20.87 W
retlw    0xF8      ; Low Byte, ADRES = 839, Forward Power = 22.90 W
retlw    0xF9      ; Low Byte, ADRES = 843, Forward Power = 25.13 W
retlw    0xFA      ; Low Byte, ADRES = 847, Forward Power = 27.57 W
retlw    0xFB      ; Low Byte, ADRES = 851, Forward Power = 30.24 W
retlw    0xFC      ; Low Byte, ADRES = 855, Forward Power = 33.18 W
retlw    0xFD      ; Low Byte, ADRES = 859, Forward Power = 36.40 W
retlw    0xFE      ; Low Byte, ADRES = 863, Forward Power = 39.94 W
retlw    0xFF      ; Low Byte, ADRES = 867, Forward Power = 43.81 W
retlw    0x00      ; Low Byte, ADRES = 871, Forward Power = 48.07 W
retlw    0x01      ; Low Byte, ADRES = 875, Forward Power = 52.74 W
retlw    0x02      ; Low Byte, ADRES = 879, Forward Power = 57.86 W
retlw    0x03      ; Low Byte, ADRES = 883, Forward Power = 63.48 W
retlw    0x04      ; Low Byte, ADRES = 887, Forward Power = 69.64 W
retlw    0x05      ; Low Byte, ADRES = 891, Forward Power = 76.41 W
retlw    0x06      ; Low Byte, ADRES = 895, Forward Power = 83.83 W
retlw    0x07      ; Low Byte, ADRES = 899, Forward Power = 91.97 W
retlw    0x08      ; Low Byte, ADRES = 903, Forward Power = 100.9 W
retlw    0x09      ; Low Byte, ADRES = 907, Forward Power = 110.7 W
retlw    0x0A      ; Low Byte, ADRES = 911, Forward Power = 121.4 W
retlw    0x0B      ; Low Byte, ADRES = 915, Forward Power = 133.2 W
retlw    0x0C      ; Low Byte, ADRES = 919, Forward Power = 146.2 W
retlw    0x0D      ; Low Byte, ADRES = 923, Forward Power = 160.4 W
retlw    0x0E      ; Low Byte, ADRES = 927, Forward Power = 176.0 W
retlw    0x0F      ; Low Byte, ADRES = 931, Forward Power = 193.0 W
retlw    0x10      ; Low Byte, ADRES = 935, Forward Power = 211.8 W
retlw    0x11      ; Low Byte, ADRES = 939, Forward Power = 232.4 W
retlw    0x12      ; Low Byte, ADRES = 943, Forward Power = 254.9 W
retlw    0x13      ; Low Byte, ADRES = 947, Forward Power = 279.7 W
retlw    0x14      ; Low Byte, ADRES = 951, Forward Power = 306.8 W
retlw    0x15      ; Low Byte, ADRES = 955, Forward Power = 336.6 W
retlw    0x16      ; Low Byte, ADRES = 959, Forward Power = 369.3 W
retlw    0x17      ; Low Byte, ADRES = 963, Forward Power = 405.2 W
retlw    0x18      ; Low Byte, ADRES = 967, Forward Power = 444.6 W
retlw    0x19      ; Low Byte, ADRES = 971, Forward Power = 487.7 W
retlw    0x1A      ; Low Byte, ADRES = 975, Forward Power = 535.1 W
retlw    0x1B      ; Low Byte, ADRES = 979, Forward Power = 587.1 W
retlw    0x1C      ; Low Byte, ADRES = 983, Forward Power = 644.1 W
retlw    0x1D      ; Low Byte, ADRES = 987, Forward Power = 706.6 W
retlw    0x1E      ; Low Byte, ADRES = 991, Forward Power = 775.2 W
retlw    0x1F      ; Low Byte, ADRES = 995, Forward Power = 850.5 W
retlw    0x20      ; Low Byte, ADRES = 999, Forward Power = 933.1 W
retlw    0x21      ; Low Byte, ADRES = 1003, Forward Power = 1,023.7 W
retlw    0x22      ; Low Byte, ADRES = 1007, Forward Power = 1,123.2 W
retlw    0x23      ; Low Byte, ADRES = 1011, Forward Power = 1,232.2 W
retlw    0x24      ; Low Byte, ADRES = 1015, Forward Power = 1,351.9 W
retlw    0x25      ; Low Byte, ADRES = 1019, Forward Power = 1,483.2 W
retlw    0x26      ; Low Byte, ADRES = 1023, Forward Power = 1,627.2 W

```

FHighByteTable0

```

retlw    0x0D      ; High Byte, ADRES = 0, Forward Power = 82.60 nW
retlw    0x0E      ; High Byte, ADRES = 4, Forward Power = 90.63 nW
retlw    0x0F      ; High Byte, ADRES = 8, Forward Power = 99.43 nW
retlw    0x10      ; High Byte, ADRES = 12, Forward Power = 109.1 nW

```


retlw	0x10	; High Byte, ADRES = 16, Forward Power = 119.7 nW
retlw	0x10	; High Byte, ADRES = 20, Forward Power = 131.3 nW
retlw	0x10	; High Byte, ADRES = 24, Forward Power = 144.0 nW
retlw	0x11	; High Byte, ADRES = 28, Forward Power = 158.0 nW
retlw	0x11	; High Byte, ADRES = 32, Forward Power = 173.4 nW
retlw	0x12	; High Byte, ADRES = 36, Forward Power = 190.2 nW
retlw	0x13	; High Byte, ADRES = 40, Forward Power = 208.7 nW
retlw	0x13	; High Byte, ADRES = 44, Forward Power = 229.0 nW
retlw	0x14	; High Byte, ADRES = 48, Forward Power = 251.2 nW
retlw	0x14	; High Byte, ADRES = 52, Forward Power = 275.6 nW
retlw	0x15	; High Byte, ADRES = 56, Forward Power = 302.4 nW
retlw	0x15	; High Byte, ADRES = 60, Forward Power = 331.7 nW
retlw	0x16	; High Byte, ADRES = 64, Forward Power = 363.9 nW
retlw	0x16	; High Byte, ADRES = 68, Forward Power = 399.3 nW
retlw	0x17	; High Byte, ADRES = 72, Forward Power = 438.1 nW
retlw	0x18	; High Byte, ADRES = 76, Forward Power = 480.6 nW
retlw	0x18	; High Byte, ADRES = 80, Forward Power = 527.3 nW
retlw	0x18	; High Byte, ADRES = 84, Forward Power = 578.5 nW
retlw	0x19	; High Byte, ADRES = 88, Forward Power = 634.7 nW
retlw	0x19	; High Byte, ADRES = 92, Forward Power = 696.3 nW
retlw	0x1A	; High Byte, ADRES = 96, Forward Power = 763.9 nW
retlw	0x1B	; High Byte, ADRES = 100, Forward Power = 838.1 nW
retlw	0x1B	; High Byte, ADRES = 104, Forward Power = 919.5 nW
retlw	0x1C	; High Byte, ADRES = 108, Forward Power = 1.009 uW
retlw	0x1C	; High Byte, ADRES = 112, Forward Power = 1.107 uW
retlw	0x1D	; High Byte, ADRES = 116, Forward Power = 1.214 uW
retlw	0x1D	; High Byte, ADRES = 120, Forward Power = 1.332 uW
retlw	0x1E	; High Byte, ADRES = 124, Forward Power = 1.462 uW
retlw	0x1E	; High Byte, ADRES = 128, Forward Power = 1.603 uW
retlw	0x1F	; High Byte, ADRES = 132, Forward Power = 1.759 uW
retlw	0x20	; High Byte, ADRES = 136, Forward Power = 1.930 uW
retlw	0x20	; High Byte, ADRES = 140, Forward Power = 2.117 uW
retlw	0x20	; High Byte, ADRES = 144, Forward Power = 2.323 uW
retlw	0x21	; High Byte, ADRES = 148, Forward Power = 2.549 uW
retlw	0x21	; High Byte, ADRES = 152, Forward Power = 2.796 uW
retlw	0x22	; High Byte, ADRES = 156, Forward Power = 3.068 uW
retlw	0x23	; High Byte, ADRES = 160, Forward Power = 3.366 uW
retlw	0x23	; High Byte, ADRES = 164, Forward Power = 3.693 uW
retlw	0x24	; High Byte, ADRES = 168, Forward Power = 4.051 uW
retlw	0x24	; High Byte, ADRES = 172, Forward Power = 4.445 uW
retlw	0x25	; High Byte, ADRES = 176, Forward Power = 4.876 uW
retlw	0x25	; High Byte, ADRES = 180, Forward Power = 5.350 uW
retlw	0x26	; High Byte, ADRES = 184, Forward Power = 5.869 uW
retlw	0x26	; High Byte, ADRES = 188, Forward Power = 6.439 uW
retlw	0x27	; High Byte, ADRES = 192, Forward Power = 7.065 uW
retlw	0x28	; High Byte, ADRES = 196, Forward Power = 7.751 uW
retlw	0x28	; High Byte, ADRES = 200, Forward Power = 8.504 uW
retlw	0x28	; High Byte, ADRES = 204, Forward Power = 9.329 uW
retlw	0x29	; High Byte, ADRES = 208, Forward Power = 10.24 uW
retlw	0x29	; High Byte, ADRES = 212, Forward Power = 11.23 uW
retlw	0x2A	; High Byte, ADRES = 216, Forward Power = 12.32 uW
retlw	0x2B	; High Byte, ADRES = 220, Forward Power = 13.52 uW
retlw	0x2B	; High Byte, ADRES = 224, Forward Power = 14.83 uW
retlw	0x2C	; High Byte, ADRES = 228, Forward Power = 16.27 uW
retlw	0x2C	; High Byte, ADRES = 232, Forward Power = 17.85 uW
retlw	0x2D	; High Byte, ADRES = 236, Forward Power = 19.58 uW
retlw	0x2D	; High Byte, ADRES = 240, Forward Power = 21.48 uW
retlw	0x2E	; High Byte, ADRES = 244, Forward Power = 23.57 uW
retlw	0x2E	; High Byte, ADRES = 248, Forward Power = 25.86 uW
retlw	0x2F	; High Byte, ADRES = 252, Forward Power = 28.37 uW
retlw	0x30	; High Byte, ADRES = 256, Forward Power = 31.13 uW
retlw	0x30	; High Byte, ADRES = 260, Forward Power = 34.15 uW
retlw	0x30	; High Byte, ADRES = 264, Forward Power = 37.47 uW
retlw	0x31	; High Byte, ADRES = 268, Forward Power = 41.10 uW
retlw	0x31	; High Byte, ADRES = 272, Forward Power = 45.10 uW
retlw	0x32	; High Byte, ADRES = 276, Forward Power = 49.48 uW
retlw	0x33	; High Byte, ADRES = 280, Forward Power = 54.28 uW
retlw	0x33	; High Byte, ADRES = 284, Forward Power = 59.55 uW
retlw	0x34	; High Byte, ADRES = 288, Forward Power = 65.33 uW
retlw	0x34	; High Byte, ADRES = 292, Forward Power = 71.68 uW
retlw	0x35	; High Byte, ADRES = 296, Forward Power = 78.64 uW
retlw	0x35	; High Byte, ADRES = 300, Forward Power = 86.28 uW
retlw	0x36	; High Byte, ADRES = 304, Forward Power = 94.66 uW
retlw	0x36	; High Byte, ADRES = 308, Forward Power = 103.8 uW
retlw	0x37	; High Byte, ADRES = 312, Forward Power = 113.9 uW
retlw	0x38	; High Byte, ADRES = 316, Forward Power = 125.0 uW
retlw	0x38	; High Byte, ADRES = 320, Forward Power = 137.1 uW
retlw	0x38	; High Byte, ADRES = 324, Forward Power = 150.5 uW
retlw	0x39	; High Byte, ADRES = 328, Forward Power = 165.1 uW
retlw	0x39	; High Byte, ADRES = 332, Forward Power = 181.1 uW
retlw	0x3A	; High Byte, ADRES = 336, Forward Power = 198.7 uW

retlw	0x3B	; High Byte, ADRES = 340, Forward Power = 218.0 uW
retlw	0x3B	; High Byte, ADRES = 344, Forward Power = 239.2 uW
retlw	0x3C	; High Byte, ADRES = 348, Forward Power = 262.4 uW
retlw	0x3C	; High Byte, ADRES = 352, Forward Power = 287.9 uW
retlw	0x3D	; High Byte, ADRES = 356, Forward Power = 315.8 uW
retlw	0x3D	; High Byte, ADRES = 360, Forward Power = 346.5 uW
retlw	0x3E	; High Byte, ADRES = 364, Forward Power = 380.1 uW
retlw	0x3E	; High Byte, ADRES = 368, Forward Power = 417.0 uW
retlw	0x3F	; High Byte, ADRES = 372, Forward Power = 457.5 uW
retlw	0x40	; High Byte, ADRES = 376, Forward Power = 502.0 uW
retlw	0x40	; High Byte, ADRES = 380, Forward Power = 550.7 uW
retlw	0x40	; High Byte, ADRES = 384, Forward Power = 604.2 uW
retlw	0x41	; High Byte, ADRES = 388, Forward Power = 662.9 uW
retlw	0x41	; High Byte, ADRES = 392, Forward Power = 727.3 uW
retlw	0x42	; High Byte, ADRES = 396, Forward Power = 797.9 uW
retlw	0x43	; High Byte, ADRES = 400, Forward Power = 875.4 uW
retlw	0x43	; High Byte, ADRES = 404, Forward Power = 960.4 uW
retlw	0x44	; High Byte, ADRES = 408, Forward Power = 1.054 mW
retlw	0x44	; High Byte, ADRES = 412, Forward Power = 1.156 mW
retlw	0x45	; High Byte, ADRES = 416, Forward Power = 1.268 mW
retlw	0x45	; High Byte, ADRES = 420, Forward Power = 1.391 mW
retlw	0x46	; High Byte, ADRES = 424, Forward Power = 1.527 mW
retlw	0x46	; High Byte, ADRES = 428, Forward Power = 1.675 mW
retlw	0x47	; High Byte, ADRES = 432, Forward Power = 1.837 mW
retlw	0x48	; High Byte, ADRES = 436, Forward Power = 2.016 mW
retlw	0x48	; High Byte, ADRES = 440, Forward Power = 2.212 mW
retlw	0x48	; High Byte, ADRES = 444, Forward Power = 2.426 mW
retlw	0x49	; High Byte, ADRES = 448, Forward Power = 2.662 mW
retlw	0x49	; High Byte, ADRES = 452, Forward Power = 2.921 mW
retlw	0x4A	; High Byte, ADRES = 456, Forward Power = 3.204 mW
retlw	0x4B	; High Byte, ADRES = 460, Forward Power = 3.515 mW
retlw	0x4B	; High Byte, ADRES = 464, Forward Power = 3.857 mW
retlw	0x4C	; High Byte, ADRES = 468, Forward Power = 4.231 mW
retlw	0x4C	; High Byte, ADRES = 472, Forward Power = 4.642 mW
retlw	0x4D	; High Byte, ADRES = 476, Forward Power = 5.093 mW
retlw	0x4D	; High Byte, ADRES = 480, Forward Power = 5.588 mW
retlw	0x4E	; High Byte, ADRES = 484, Forward Power = 6.130 mW
retlw	0x4E	; High Byte, ADRES = 488, Forward Power = 6.726 mW
retlw	0x4F	; High Byte, ADRES = 492, Forward Power = 7.379 mW
retlw	0x50	; High Byte, ADRES = 496, Forward Power = 8.096 mW
retlw	0x50	; High Byte, ADRES = 500, Forward Power = 8.882 mW
retlw	0x50	; High Byte, ADRES = 504, Forward Power = 9.744 mW
retlw	0x51	; High Byte, ADRES = 508, Forward Power = 10.69 mW
retlw	0x52	; High Byte, ADRES = 512, Forward Power = 11.73 mW
retlw	0x52	; High Byte, ADRES = 516, Forward Power = 12.87 mW
retlw	0x53	; High Byte, ADRES = 520, Forward Power = 14.12 mW
retlw	0x53	; High Byte, ADRES = 524, Forward Power = 15.49 mW
retlw	0x54	; High Byte, ADRES = 528, Forward Power = 16.99 mW
retlw	0x54	; High Byte, ADRES = 532, Forward Power = 18.64 mW
retlw	0x55	; High Byte, ADRES = 536, Forward Power = 20.45 mW
retlw	0x55	; High Byte, ADRES = 540, Forward Power = 22.44 mW
retlw	0x56	; High Byte, ADRES = 544, Forward Power = 24.62 mW
retlw	0x56	; High Byte, ADRES = 548, Forward Power = 27.01 mW
retlw	0x57	; High Byte, ADRES = 552, Forward Power = 29.63 mW
retlw	0x58	; High Byte, ADRES = 556, Forward Power = 32.51 mW
retlw	0x58	; High Byte, ADRES = 560, Forward Power = 35.67 mW
retlw	0x59	; High Byte, ADRES = 564, Forward Power = 39.13 mW
retlw	0x59	; High Byte, ADRES = 568, Forward Power = 42.93 mW
retlw	0x5A	; High Byte, ADRES = 572, Forward Power = 47.10 mW
retlw	0x5A	; High Byte, ADRES = 576, Forward Power = 51.68 mW
retlw	0x5B	; High Byte, ADRES = 580, Forward Power = 56.69 mW
retlw	0x5B	; High Byte, ADRES = 584, Forward Power = 62.20 mW
retlw	0x5C	; High Byte, ADRES = 588, Forward Power = 68.24 mW
retlw	0x5C	; High Byte, ADRES = 592, Forward Power = 74.87 mW
retlw	0x5D	; High Byte, ADRES = 596, Forward Power = 82.14 mW
retlw	0x5D	; High Byte, ADRES = 600, Forward Power = 90.12 mW
retlw	0x5E	; High Byte, ADRES = 604, Forward Power = 98.87 mW
retlw	0x5E	; High Byte, ADRES = 608, Forward Power = 108.5 mW
retlw	0x5F	; High Byte, ADRES = 612, Forward Power = 119.0 mW
retlw	0x60	; High Byte, ADRES = 616, Forward Power = 130.6 mW
retlw	0x60	; High Byte, ADRES = 620, Forward Power = 143.2 mW
retlw	0x61	; High Byte, ADRES = 624, Forward Power = 157.1 mW
retlw	0x61	; High Byte, ADRES = 628, Forward Power = 172.4 mW
retlw	0x62	; High Byte, ADRES = 632, Forward Power = 189.2 mW
retlw	0x62	; High Byte, ADRES = 636, Forward Power = 207.5 mW
retlw	0x63	; High Byte, ADRES = 640, Forward Power = 227.7 mW
retlw	0x63	; High Byte, ADRES = 644, Forward Power = 249.8 mW
retlw	0x64	; High Byte, ADRES = 648, Forward Power = 274.0 mW
retlw	0x64	; High Byte, ADRES = 652, Forward Power = 300.7 mW
retlw	0x65	; High Byte, ADRES = 656, Forward Power = 329.9 mW
retlw	0x65	; High Byte, ADRES = 660, Forward Power = 361.9 mW

retlw	0x66	; High Byte, ADRES = 664, Forward Power = 397.0 mW
retlw	0x66	; High Byte, ADRES = 668, Forward Power = 435.6 mW
retlw	0x67	; High Byte, ADRES = 672, Forward Power = 477.9 mW
retlw	0x68	; High Byte, ADRES = 676, Forward Power = 524.3 mW
retlw	0x68	; High Byte, ADRES = 680, Forward Power = 575.2 mW
retlw	0x69	; High Byte, ADRES = 684, Forward Power = 631.1 mW
retlw	0x69	; High Byte, ADRES = 688, Forward Power = 692.4 mW
retlw	0x6A	; High Byte, ADRES = 692, Forward Power = 759.6 mW
retlw	0x6A	; High Byte, ADRES = 696, Forward Power = 833.4 mW
retlw	0x6B	; High Byte, ADRES = 700, Forward Power = 914.3 mW
retlw	0x6C	; High Byte, ADRES = 704, Forward Power = 1.003 W
retlw	0x6C	; High Byte, ADRES = 708, Forward Power = 1.101 W
retlw	0x6C	; High Byte, ADRES = 712, Forward Power = 1.207 W
retlw	0x6D	; High Byte, ADRES = 716, Forward Power = 1.325 W
retlw	0x6D	; High Byte, ADRES = 720, Forward Power = 1.453 W
retlw	0x6E	; High Byte, ADRES = 724, Forward Power = 1.594 W
retlw	0x6E	; High Byte, ADRES = 728, Forward Power = 1.749 W
retlw	0x6F	; High Byte, ADRES = 732, Forward Power = 1.919 W
retlw	0x70	; High Byte, ADRES = 736, Forward Power = 2.106 W
retlw	0x70	; High Byte, ADRES = 740, Forward Power = 2.310 W
retlw	0x71	; High Byte, ADRES = 744, Forward Power = 2.534 W
retlw	0x71	; High Byte, ADRES = 748, Forward Power = 2.781 W
retlw	0x72	; High Byte, ADRES = 752, Forward Power = 3.051 W
retlw	0x72	; High Byte, ADRES = 756, Forward Power = 3.347 W
retlw	0x73	; High Byte, ADRES = 760, Forward Power = 3.672 W
retlw	0x74	; High Byte, ADRES = 764, Forward Power = 4.028 W
retlw	0x74	; High Byte, ADRES = 768, Forward Power = 4.420 W
retlw	0x74	; High Byte, ADRES = 772, Forward Power = 4.849 W
retlw	0x75	; High Byte, ADRES = 776, Forward Power = 5.320 W
retlw	0x75	; High Byte, ADRES = 780, Forward Power = 5.836 W
retlw	0x76	; High Byte, ADRES = 784, Forward Power = 6.403 W
retlw	0x77	; High Byte, ADRES = 788, Forward Power = 7.025 W
retlw	0x77	; High Byte, ADRES = 792, Forward Power = 7.707 W
retlw	0x78	; High Byte, ADRES = 796, Forward Power = 8.456 W
retlw	0x78	; High Byte, ADRES = 800, Forward Power = 9.277 W
retlw	0x79	; High Byte, ADRES = 804, Forward Power = 10.18 W
retlw	0x79	; High Byte, ADRES = 808, Forward Power = 11.17 W
retlw	0x7A	; High Byte, ADRES = 812, Forward Power = 12.25 W
retlw	0x7A	; High Byte, ADRES = 816, Forward Power = 13.44 W
retlw	0x7B	; High Byte, ADRES = 820, Forward Power = 14.75 W
retlw	0x7C	; High Byte, ADRES = 824, Forward Power = 16.18 W
retlw	0x7C	; High Byte, ADRES = 828, Forward Power = 17.75 W
retlw	0x7C	; High Byte, ADRES = 832, Forward Power = 19.47 W
retlw	0x7D	; High Byte, ADRES = 836, Forward Power = 21.36 W
retlw	0x7D	; High Byte, ADRES = 840, Forward Power = 23.44 W
retlw	0x7E	; High Byte, ADRES = 844, Forward Power = 25.71 W
retlw	0x7F	; High Byte, ADRES = 848, Forward Power = 28.21 W
retlw	0x7F	; High Byte, ADRES = 852, Forward Power = 30.95 W
retlw	0x80	; High Byte, ADRES = 856, Forward Power = 33.96 W
retlw	0x80	; High Byte, ADRES = 860, Forward Power = 37.25 W
retlw	0x81	; High Byte, ADRES = 864, Forward Power = 40.87 W
retlw	0x81	; High Byte, ADRES = 868, Forward Power = 44.84 W
retlw	0x82	; High Byte, ADRES = 872, Forward Power = 49.20 W
retlw	0x82	; High Byte, ADRES = 876, Forward Power = 53.97 W
retlw	0x83	; High Byte, ADRES = 880, Forward Power = 59.22 W
retlw	0x84	; High Byte, ADRES = 884, Forward Power = 64.97 W
retlw	0x84	; High Byte, ADRES = 888, Forward Power = 71.28 W
retlw	0x84	; High Byte, ADRES = 892, Forward Power = 78.20 W
retlw	0x85	; High Byte, ADRES = 896, Forward Power = 85.79 W
retlw	0x85	; High Byte, ADRES = 900, Forward Power = 94.12 W
retlw	0x86	; High Byte, ADRES = 904, Forward Power = 103.3 W
retlw	0x87	; High Byte, ADRES = 908, Forward Power = 113.3 W
retlw	0x87	; High Byte, ADRES = 912, Forward Power = 124.3 W
retlw	0x88	; High Byte, ADRES = 916, Forward Power = 136.4 W
retlw	0x88	; High Byte, ADRES = 920, Forward Power = 149.6 W
retlw	0x89	; High Byte, ADRES = 924, Forward Power = 164.1 W
retlw	0x89	; High Byte, ADRES = 928, Forward Power = 180.1 W
retlw	0x8A	; High Byte, ADRES = 932, Forward Power = 197.6 W
retlw	0x8A	; High Byte, ADRES = 936, Forward Power = 216.8 W
retlw	0x8B	; High Byte, ADRES = 940, Forward Power = 237.8 W
retlw	0x8C	; High Byte, ADRES = 944, Forward Power = 260.9 W
retlw	0x8C	; High Byte, ADRES = 948, Forward Power = 286.2 W
retlw	0x8C	; High Byte, ADRES = 952, Forward Power = 314.0 W
retlw	0x8D	; High Byte, ADRES = 956, Forward Power = 344.5 W
retlw	0x8D	; High Byte, ADRES = 960, Forward Power = 378.0 W
retlw	0x8E	; High Byte, ADRES = 964, Forward Power = 414.7 W
retlw	0x8F	; High Byte, ADRES = 968, Forward Power = 455.0 W
retlw	0x8F	; High Byte, ADRES = 972, Forward Power = 499.2 W
retlw	0x90	; High Byte, ADRES = 976, Forward Power = 547.6 W
retlw	0x90	; High Byte, ADRES = 980, Forward Power = 600.8 W
retlw	0x91	; High Byte, ADRES = 984, Forward Power = 659.2 W

retlw	0x91	; High Byte, ADRES = 988, Forward Power = 723.2 W
retlw	0x92	; High Byte, ADRES = 992, Forward Power = 793.4 W
retlw	0x92	; High Byte, ADRES = 996, Forward Power = 870.5 W
retlw	0x93	; High Byte, ADRES = 1000, Forward Power = 955.0 W
retlw	0x94	; High Byte, ADRES = 1004, Forward Power = 1,047.7 W
retlw	0x94	; High Byte, ADRES = 1008, Forward Power = 1,149.5 W
retlw	0x94	; High Byte, ADRES = 1012, Forward Power = 1,261.1 W
retlw	0x95	; High Byte, ADRES = 1016, Forward Power = 1,383.6 W
retlw	0x95	; High Byte, ADRES = 1020, Forward Power = 1,518.0 W

FHighByteTable1

retlw	0x0D	; High Byte, ADRES = 1, Forward Power = 84.54 nW
retlw	0x0E	; High Byte, ADRES = 5, Forward Power = 92.75 nW
retlw	0x0E	; High Byte, ADRES = 9, Forward Power = 101.8 nW
retlw	0x0F	; High Byte, ADRES = 13, Forward Power = 111.6 nW
retlw	0x10	; High Byte, ADRES = 17, Forward Power = 122.5 nW
retlw	0x10	; High Byte, ADRES = 21, Forward Power = 134.4 nW
retlw	0x10	; High Byte, ADRES = 25, Forward Power = 147.4 nW
retlw	0x11	; High Byte, ADRES = 29, Forward Power = 161.7 nW
retlw	0x11	; High Byte, ADRES = 33, Forward Power = 177.5 nW
retlw	0x12	; High Byte, ADRES = 37, Forward Power = 194.7 nW
retlw	0x13	; High Byte, ADRES = 41, Forward Power = 213.6 nW
retlw	0x13	; High Byte, ADRES = 45, Forward Power = 234.3 nW
retlw	0x14	; High Byte, ADRES = 49, Forward Power = 257.1 nW
retlw	0x14	; High Byte, ADRES = 53, Forward Power = 282.1 nW
retlw	0x15	; High Byte, ADRES = 57, Forward Power = 309.5 nW
retlw	0x15	; High Byte, ADRES = 61, Forward Power = 339.5 nW
retlw	0x16	; High Byte, ADRES = 65, Forward Power = 372.5 nW
retlw	0x16	; High Byte, ADRES = 69, Forward Power = 408.6 nW
retlw	0x17	; High Byte, ADRES = 73, Forward Power = 448.3 nW
retlw	0x18	; High Byte, ADRES = 77, Forward Power = 491.9 nW
retlw	0x18	; High Byte, ADRES = 81, Forward Power = 539.6 nW
retlw	0x18	; High Byte, ADRES = 85, Forward Power = 592.0 nW
retlw	0x19	; High Byte, ADRES = 89, Forward Power = 649.5 nW
retlw	0x19	; High Byte, ADRES = 93, Forward Power = 712.6 nW
retlw	0x1A	; High Byte, ADRES = 97, Forward Power = 781.8 nW
retlw	0x1B	; High Byte, ADRES = 101, Forward Power = 857.8 nW
retlw	0x1B	; High Byte, ADRES = 105, Forward Power = 941.1 nW
retlw	0x1C	; High Byte, ADRES = 109, Forward Power = 1.032 uW
retlw	0x1C	; High Byte, ADRES = 113, Forward Power = 1.133 uW
retlw	0x1D	; High Byte, ADRES = 117, Forward Power = 1.243 uW
retlw	0x1D	; High Byte, ADRES = 121, Forward Power = 1.363 uW
retlw	0x1E	; High Byte, ADRES = 125, Forward Power = 1.496 uW
retlw	0x1E	; High Byte, ADRES = 129, Forward Power = 1.641 uW
retlw	0x1F	; High Byte, ADRES = 133, Forward Power = 1.800 uW
retlw	0x20	; High Byte, ADRES = 137, Forward Power = 1.975 uW
retlw	0x20	; High Byte, ADRES = 141, Forward Power = 2.167 uW
retlw	0x20	; High Byte, ADRES = 145, Forward Power = 2.378 uW
retlw	0x21	; High Byte, ADRES = 149, Forward Power = 2.608 uW
retlw	0x22	; High Byte, ADRES = 153, Forward Power = 2.862 uW
retlw	0x22	; High Byte, ADRES = 157, Forward Power = 3.140 uW
retlw	0x23	; High Byte, ADRES = 161, Forward Power = 3.445 uW
retlw	0x23	; High Byte, ADRES = 165, Forward Power = 3.779 uW
retlw	0x24	; High Byte, ADRES = 169, Forward Power = 4.146 uW
retlw	0x24	; High Byte, ADRES = 173, Forward Power = 4.549 uW
retlw	0x25	; High Byte, ADRES = 177, Forward Power = 4.991 uW
retlw	0x25	; High Byte, ADRES = 181, Forward Power = 5.475 uW
retlw	0x26	; High Byte, ADRES = 185, Forward Power = 6.007 uW
retlw	0x26	; High Byte, ADRES = 189, Forward Power = 6.590 uW
retlw	0x27	; High Byte, ADRES = 193, Forward Power = 7.230 uW
retlw	0x28	; High Byte, ADRES = 197, Forward Power = 7.933 uW
retlw	0x28	; High Byte, ADRES = 201, Forward Power = 8.703 uW
retlw	0x29	; High Byte, ADRES = 205, Forward Power = 9.548 uW
retlw	0x29	; High Byte, ADRES = 209, Forward Power = 10.48 uW
retlw	0x2A	; High Byte, ADRES = 213, Forward Power = 11.49 uW
retlw	0x2A	; High Byte, ADRES = 217, Forward Power = 12.61 uW
retlw	0x2B	; High Byte, ADRES = 221, Forward Power = 13.83 uW
retlw	0x2B	; High Byte, ADRES = 225, Forward Power = 15.18 uW
retlw	0x2C	; High Byte, ADRES = 229, Forward Power = 16.65 uW
retlw	0x2C	; High Byte, ADRES = 233, Forward Power = 18.27 uW
retlw	0x2D	; High Byte, ADRES = 237, Forward Power = 20.04 uW
retlw	0x2D	; High Byte, ADRES = 241, Forward Power = 21.99 uW
retlw	0x2E	; High Byte, ADRES = 245, Forward Power = 24.12 uW
retlw	0x2E	; High Byte, ADRES = 249, Forward Power = 26.47 uW
retlw	0x2F	; High Byte, ADRES = 253, Forward Power = 29.04 uW
retlw	0x30	; High Byte, ADRES = 257, Forward Power = 31.86 uW
retlw	0x30	; High Byte, ADRES = 261, Forward Power = 34.95 uW
retlw	0x31	; High Byte, ADRES = 265, Forward Power = 38.34 uW
retlw	0x31	; High Byte, ADRES = 269, Forward Power = 42.07 uW
retlw	0x32	; High Byte, ADRES = 273, Forward Power = 46.15 uW
retlw	0x32	; High Byte, ADRES = 277, Forward Power = 50.64 uW

retlw	0x33	; High Byte, ADRES = 281, Forward Power = 55.55 uW
retlw	0x33	; High Byte, ADRES = 285, Forward Power = 60.95 uW
retlw	0x34	; High Byte, ADRES = 289, Forward Power = 66.87 uW
retlw	0x34	; High Byte, ADRES = 293, Forward Power = 73.36 uW
retlw	0x35	; High Byte, ADRES = 297, Forward Power = 80.48 uW
retlw	0x35	; High Byte, ADRES = 301, Forward Power = 88.30 uW
retlw	0x36	; High Byte, ADRES = 305, Forward Power = 96.88 uW
retlw	0x36	; High Byte, ADRES = 309, Forward Power = 106.3 uW
retlw	0x37	; High Byte, ADRES = 313, Forward Power = 116.6 uW
retlw	0x38	; High Byte, ADRES = 317, Forward Power = 127.9 uW
retlw	0x38	; High Byte, ADRES = 321, Forward Power = 140.4 uW
retlw	0x39	; High Byte, ADRES = 325, Forward Power = 154.0 uW
retlw	0x39	; High Byte, ADRES = 329, Forward Power = 168.9 uW
retlw	0x3A	; High Byte, ADRES = 333, Forward Power = 185.3 uW
retlw	0x3A	; High Byte, ADRES = 337, Forward Power = 203.3 uW
retlw	0x3B	; High Byte, ADRES = 341, Forward Power = 223.1 uW
retlw	0x3C	; High Byte, ADRES = 345, Forward Power = 244.8 uW
retlw	0x3C	; High Byte, ADRES = 349, Forward Power = 268.5 uW
retlw	0x3C	; High Byte, ADRES = 353, Forward Power = 294.6 uW
retlw	0x3D	; High Byte, ADRES = 357, Forward Power = 323.2 uW
retlw	0x3D	; High Byte, ADRES = 361, Forward Power = 354.6 uW
retlw	0x3E	; High Byte, ADRES = 365, Forward Power = 389.0 uW
retlw	0x3E	; High Byte, ADRES = 369, Forward Power = 426.8 uW
retlw	0x3F	; High Byte, ADRES = 373, Forward Power = 468.3 uW
retlw	0x40	; High Byte, ADRES = 377, Forward Power = 513.7 uW
retlw	0x40	; High Byte, ADRES = 381, Forward Power = 563.6 uW
retlw	0x41	; High Byte, ADRES = 385, Forward Power = 618.4 uW
retlw	0x41	; High Byte, ADRES = 389, Forward Power = 678.4 uW
retlw	0x42	; High Byte, ADRES = 393, Forward Power = 744.3 uW
retlw	0x42	; High Byte, ADRES = 397, Forward Power = 816.6 uW
retlw	0x43	; High Byte, ADRES = 401, Forward Power = 895.9 uW
retlw	0x44	; High Byte, ADRES = 405, Forward Power = 982.9 uW
retlw	0x44	; High Byte, ADRES = 409, Forward Power = 1.078 mW
retlw	0x44	; High Byte, ADRES = 413, Forward Power = 1.183 mW
retlw	0x45	; High Byte, ADRES = 417, Forward Power = 1.298 mW
retlw	0x45	; High Byte, ADRES = 421, Forward Power = 1.424 mW
retlw	0x46	; High Byte, ADRES = 425, Forward Power = 1.562 mW
retlw	0x47	; High Byte, ADRES = 429, Forward Power = 1.714 mW
retlw	0x47	; High Byte, ADRES = 433, Forward Power = 1.881 mW
retlw	0x48	; High Byte, ADRES = 437, Forward Power = 2.063 mW
retlw	0x48	; High Byte, ADRES = 441, Forward Power = 2.264 mW
retlw	0x49	; High Byte, ADRES = 445, Forward Power = 2.483 mW
retlw	0x49	; High Byte, ADRES = 449, Forward Power = 2.725 mW
retlw	0x4A	; High Byte, ADRES = 453, Forward Power = 2.989 mW
retlw	0x4A	; High Byte, ADRES = 457, Forward Power = 3.279 mW
retlw	0x4B	; High Byte, ADRES = 461, Forward Power = 3.598 mW
retlw	0x4C	; High Byte, ADRES = 465, Forward Power = 3.947 mW
retlw	0x4C	; High Byte, ADRES = 469, Forward Power = 4.331 mW
retlw	0x4C	; High Byte, ADRES = 473, Forward Power = 4.751 mW
retlw	0x4D	; High Byte, ADRES = 477, Forward Power = 5.213 mW
retlw	0x4D	; High Byte, ADRES = 481, Forward Power = 5.719 mW
retlw	0x4E	; High Byte, ADRES = 485, Forward Power = 6.274 mW
retlw	0x4F	; High Byte, ADRES = 489, Forward Power = 6.883 mW
retlw	0x4F	; High Byte, ADRES = 493, Forward Power = 7.552 mW
retlw	0x50	; High Byte, ADRES = 497, Forward Power = 8.285 mW
retlw	0x50	; High Byte, ADRES = 501, Forward Power = 9.090 mW
retlw	0x51	; High Byte, ADRES = 505, Forward Power = 9.973 mW
retlw	0x51	; High Byte, ADRES = 509, Forward Power = 10.94 mW
retlw	0x52	; High Byte, ADRES = 513, Forward Power = 12.00 mW
retlw	0x52	; High Byte, ADRES = 517, Forward Power = 13.17 mW
retlw	0x53	; High Byte, ADRES = 521, Forward Power = 14.45 mW
retlw	0x54	; High Byte, ADRES = 525, Forward Power = 15.85 mW
retlw	0x54	; High Byte, ADRES = 529, Forward Power = 17.39 mW
retlw	0x54	; High Byte, ADRES = 533, Forward Power = 19.08 mW
retlw	0x55	; High Byte, ADRES = 537, Forward Power = 20.93 mW
retlw	0x55	; High Byte, ADRES = 541, Forward Power = 22.97 mW
retlw	0x56	; High Byte, ADRES = 545, Forward Power = 25.20 mW
retlw	0x57	; High Byte, ADRES = 549, Forward Power = 27.64 mW
retlw	0x57	; High Byte, ADRES = 553, Forward Power = 30.33 mW
retlw	0x58	; High Byte, ADRES = 557, Forward Power = 33.27 mW
retlw	0x58	; High Byte, ADRES = 561, Forward Power = 36.50 mW
retlw	0x59	; High Byte, ADRES = 565, Forward Power = 40.05 mW
retlw	0x59	; High Byte, ADRES = 569, Forward Power = 43.94 mW
retlw	0x5A	; High Byte, ADRES = 573, Forward Power = 48.21 mW
retlw	0x5A	; High Byte, ADRES = 577, Forward Power = 52.89 mW
retlw	0x5B	; High Byte, ADRES = 581, Forward Power = 58.02 mW
retlw	0x5C	; High Byte, ADRES = 585, Forward Power = 63.66 mW
retlw	0x5C	; High Byte, ADRES = 589, Forward Power = 69.84 mW
retlw	0x5C	; High Byte, ADRES = 593, Forward Power = 76.62 mW
retlw	0x5D	; High Byte, ADRES = 597, Forward Power = 84.06 mW
retlw	0x5D	; High Byte, ADRES = 601, Forward Power = 92.23 mW

retlw	0x5E	; High Byte, ADRES = 605, Forward Power = 101.2 mW
retlw	0x5F	; High Byte, ADRES = 609, Forward Power = 111.0 mW
retlw	0x5F	; High Byte, ADRES = 613, Forward Power = 121.8 mW
retlw	0x60	; High Byte, ADRES = 617, Forward Power = 133.6 mW
retlw	0x60	; High Byte, ADRES = 621, Forward Power = 146.6 mW
retlw	0x61	; High Byte, ADRES = 625, Forward Power = 160.8 mW
retlw	0x61	; High Byte, ADRES = 629, Forward Power = 176.5 mW
retlw	0x62	; High Byte, ADRES = 633, Forward Power = 193.6 mW
retlw	0x62	; High Byte, ADRES = 637, Forward Power = 212.4 mW
retlw	0x63	; High Byte, ADRES = 641, Forward Power = 233.0 mW
retlw	0x64	; High Byte, ADRES = 645, Forward Power = 255.6 mW
retlw	0x64	; High Byte, ADRES = 649, Forward Power = 280.5 mW
retlw	0x64	; High Byte, ADRES = 653, Forward Power = 307.7 mW
retlw	0x65	; High Byte, ADRES = 657, Forward Power = 337.6 mW
retlw	0x65	; High Byte, ADRES = 661, Forward Power = 370.4 mW
retlw	0x66	; High Byte, ADRES = 665, Forward Power = 406.3 mW
retlw	0x67	; High Byte, ADRES = 669, Forward Power = 445.8 mW
retlw	0x67	; High Byte, ADRES = 673, Forward Power = 489.1 mW
retlw	0x68	; High Byte, ADRES = 677, Forward Power = 536.6 mW
retlw	0x68	; High Byte, ADRES = 681, Forward Power = 588.7 mW
retlw	0x69	; High Byte, ADRES = 685, Forward Power = 645.9 mW
retlw	0x69	; High Byte, ADRES = 689, Forward Power = 708.6 mW
retlw	0x6A	; High Byte, ADRES = 693, Forward Power = 777.4 mW
retlw	0x6A	; High Byte, ADRES = 697, Forward Power = 852.9 mW
retlw	0x6B	; High Byte, ADRES = 701, Forward Power = 935.8 mW
retlw	0x6C	; High Byte, ADRES = 705, Forward Power = 1.027 W
retlw	0x6C	; High Byte, ADRES = 709, Forward Power = 1.126 W
retlw	0x6C	; High Byte, ADRES = 713, Forward Power = 1.236 W
retlw	0x6D	; High Byte, ADRES = 717, Forward Power = 1.356 W
retlw	0x6D	; High Byte, ADRES = 721, Forward Power = 1.487 W
retlw	0x6E	; High Byte, ADRES = 725, Forward Power = 1.632 W
retlw	0x6F	; High Byte, ADRES = 729, Forward Power = 1.790 W
retlw	0x6F	; High Byte, ADRES = 733, Forward Power = 1.964 W
retlw	0x70	; High Byte, ADRES = 737, Forward Power = 2.155 W
retlw	0x70	; High Byte, ADRES = 741, Forward Power = 2.364 W
retlw	0x71	; High Byte, ADRES = 745, Forward Power = 2.594 W
retlw	0x71	; High Byte, ADRES = 749, Forward Power = 2.846 W
retlw	0x72	; High Byte, ADRES = 753, Forward Power = 3.122 W
retlw	0x72	; High Byte, ADRES = 757, Forward Power = 3.425 W
retlw	0x73	; High Byte, ADRES = 761, Forward Power = 3.758 W
retlw	0x74	; High Byte, ADRES = 765, Forward Power = 4.123 W
retlw	0x74	; High Byte, ADRES = 769, Forward Power = 4.523 W
retlw	0x74	; High Byte, ADRES = 773, Forward Power = 4.962 W
retlw	0x75	; High Byte, ADRES = 777, Forward Power = 5.444 W
retlw	0x75	; High Byte, ADRES = 781, Forward Power = 5.973 W
retlw	0x76	; High Byte, ADRES = 785, Forward Power = 6.553 W
retlw	0x77	; High Byte, ADRES = 789, Forward Power = 7.190 W
retlw	0x77	; High Byte, ADRES = 793, Forward Power = 7.888 W
retlw	0x78	; High Byte, ADRES = 797, Forward Power = 8.654 W
retlw	0x78	; High Byte, ADRES = 801, Forward Power = 9.494 W
retlw	0x79	; High Byte, ADRES = 805, Forward Power = 10.42 W
retlw	0x79	; High Byte, ADRES = 809, Forward Power = 11.43 W
retlw	0x7A	; High Byte, ADRES = 813, Forward Power = 12.54 W
retlw	0x7A	; High Byte, ADRES = 817, Forward Power = 13.76 W
retlw	0x7B	; High Byte, ADRES = 821, Forward Power = 15.09 W
retlw	0x7C	; High Byte, ADRES = 825, Forward Power = 16.56 W
retlw	0x7C	; High Byte, ADRES = 829, Forward Power = 18.16 W
retlw	0x7C	; High Byte, ADRES = 833, Forward Power = 19.93 W
retlw	0x7D	; High Byte, ADRES = 837, Forward Power = 21.86 W
retlw	0x7D	; High Byte, ADRES = 841, Forward Power = 23.99 W
retlw	0x7E	; High Byte, ADRES = 845, Forward Power = 26.32 W
retlw	0x7F	; High Byte, ADRES = 849, Forward Power = 28.87 W
retlw	0x7F	; High Byte, ADRES = 853, Forward Power = 31.68 W
retlw	0x80	; High Byte, ADRES = 857, Forward Power = 34.75 W
retlw	0x80	; High Byte, ADRES = 861, Forward Power = 38.13 W
retlw	0x81	; High Byte, ADRES = 865, Forward Power = 41.83 W
retlw	0x81	; High Byte, ADRES = 869, Forward Power = 45.89 W
retlw	0x82	; High Byte, ADRES = 873, Forward Power = 50.35 W
retlw	0x82	; High Byte, ADRES = 877, Forward Power = 55.24 W
retlw	0x83	; High Byte, ADRES = 881, Forward Power = 60.60 W
retlw	0x84	; High Byte, ADRES = 885, Forward Power = 66.49 W
retlw	0x84	; High Byte, ADRES = 889, Forward Power = 72.95 W
retlw	0x85	; High Byte, ADRES = 893, Forward Power = 80.03 W
retlw	0x85	; High Byte, ADRES = 897, Forward Power = 87.80 W
retlw	0x86	; High Byte, ADRES = 901, Forward Power = 96.33 W
retlw	0x86	; High Byte, ADRES = 905, Forward Power = 105.7 W
retlw	0x87	; High Byte, ADRES = 909, Forward Power = 115.9 W
retlw	0x87	; High Byte, ADRES = 913, Forward Power = 127.2 W
retlw	0x88	; High Byte, ADRES = 917, Forward Power = 139.6 W
retlw	0x88	; High Byte, ADRES = 921, Forward Power = 153.1 W
retlw	0x89	; High Byte, ADRES = 925, Forward Power = 168.0 W

retlw	0x89	; High Byte, ADRES = 929, Forward Power = 184.3 W
retlw	0x8A	; High Byte, ADRES = 933, Forward Power = 202.2 W
retlw	0x8A	; High Byte, ADRES = 937, Forward Power = 221.8 W
retlw	0x8B	; High Byte, ADRES = 941, Forward Power = 243.4 W
retlw	0x8C	; High Byte, ADRES = 945, Forward Power = 267.0 W
retlw	0x8C	; High Byte, ADRES = 949, Forward Power = 292.9 W
retlw	0x8D	; High Byte, ADRES = 953, Forward Power = 321.4 W
retlw	0x8D	; High Byte, ADRES = 957, Forward Power = 352.6 W
retlw	0x8E	; High Byte, ADRES = 961, Forward Power = 386.9 W
retlw	0x8E	; High Byte, ADRES = 965, Forward Power = 424.4 W
retlw	0x8F	; High Byte, ADRES = 969, Forward Power = 465.6 W
retlw	0x8F	; High Byte, ADRES = 973, Forward Power = 510.9 W
retlw	0x90	; High Byte, ADRES = 977, Forward Power = 560.5 W
retlw	0x90	; High Byte, ADRES = 981, Forward Power = 614.9 W
retlw	0x91	; High Byte, ADRES = 985, Forward Power = 674.6 W
retlw	0x91	; High Byte, ADRES = 989, Forward Power = 740.1 W
retlw	0x92	; High Byte, ADRES = 993, Forward Power = 812.0 W
retlw	0x92	; High Byte, ADRES = 997, Forward Power = 890.9 W
retlw	0x93	; High Byte, ADRES = 1001, Forward Power = 977.4 W
retlw	0x94	; High Byte, ADRES = 1005, Forward Power = 1,072.3 W
retlw	0x94	; High Byte, ADRES = 1009, Forward Power = 1,176.4 W
retlw	0x95	; High Byte, ADRES = 1013, Forward Power = 1,290.7 W
retlw	0x95	; High Byte, ADRES = 1017, Forward Power = 1,416.0 W
retlw	0x96	; High Byte, ADRES = 1021, Forward Power = 1,553.5 W

FHighByteTable2

retlw	0x0D	; High Byte, ADRES = 2, Forward Power = 86.52 nW
retlw	0x0E	; High Byte, ADRES = 6, Forward Power = 94.92 nW
retlw	0x0E	; High Byte, ADRES = 10, Forward Power = 104.1 nW
retlw	0x0F	; High Byte, ADRES = 14, Forward Power = 114.3 nW
retlw	0x10	; High Byte, ADRES = 18, Forward Power = 125.4 nW
retlw	0x10	; High Byte, ADRES = 22, Forward Power = 137.5 nW
retlw	0x11	; High Byte, ADRES = 26, Forward Power = 150.9 nW
retlw	0x11	; High Byte, ADRES = 30, Forward Power = 165.5 nW
retlw	0x12	; High Byte, ADRES = 34, Forward Power = 181.6 nW
retlw	0x12	; High Byte, ADRES = 38, Forward Power = 199.2 nW
retlw	0x13	; High Byte, ADRES = 42, Forward Power = 218.6 nW
retlw	0x14	; High Byte, ADRES = 46, Forward Power = 239.8 nW
retlw	0x14	; High Byte, ADRES = 50, Forward Power = 263.1 nW
retlw	0x14	; High Byte, ADRES = 54, Forward Power = 288.7 nW
retlw	0x15	; High Byte, ADRES = 58, Forward Power = 316.7 nW
retlw	0x15	; High Byte, ADRES = 62, Forward Power = 347.5 nW
retlw	0x16	; High Byte, ADRES = 66, Forward Power = 381.2 nW
retlw	0x17	; High Byte, ADRES = 70, Forward Power = 418.2 nW
retlw	0x17	; High Byte, ADRES = 74, Forward Power = 458.8 nW
retlw	0x18	; High Byte, ADRES = 78, Forward Power = 503.4 nW
retlw	0x18	; High Byte, ADRES = 82, Forward Power = 552.3 nW
retlw	0x19	; High Byte, ADRES = 86, Forward Power = 605.9 nW
retlw	0x19	; High Byte, ADRES = 90, Forward Power = 664.8 nW
retlw	0x1A	; High Byte, ADRES = 94, Forward Power = 729.3 nW
retlw	0x1A	; High Byte, ADRES = 98, Forward Power = 800.2 nW
retlw	0x1B	; High Byte, ADRES = 102, Forward Power = 877.9 nW
retlw	0x1C	; High Byte, ADRES = 106, Forward Power = 963.1 nW
retlw	0x1C	; High Byte, ADRES = 110, Forward Power = 1.057 uW
retlw	0x1C	; High Byte, ADRES = 114, Forward Power = 1.159 uW
retlw	0x1D	; High Byte, ADRES = 118, Forward Power = 1.272 uW
retlw	0x1D	; High Byte, ADRES = 122, Forward Power = 1.395 uW
retlw	0x1E	; High Byte, ADRES = 126, Forward Power = 1.531 uW
retlw	0x1F	; High Byte, ADRES = 130, Forward Power = 1.680 uW
retlw	0x1F	; High Byte, ADRES = 134, Forward Power = 1.843 uW
retlw	0x20	; High Byte, ADRES = 138, Forward Power = 2.022 uW
retlw	0x20	; High Byte, ADRES = 142, Forward Power = 2.218 uW
retlw	0x21	; High Byte, ADRES = 146, Forward Power = 2.433 uW
retlw	0x21	; High Byte, ADRES = 150, Forward Power = 2.670 uW
retlw	0x22	; High Byte, ADRES = 154, Forward Power = 2.929 uW
retlw	0x22	; High Byte, ADRES = 158, Forward Power = 3.213 uW
retlw	0x23	; High Byte, ADRES = 162, Forward Power = 3.525 uW
retlw	0x24	; High Byte, ADRES = 166, Forward Power = 3.868 uW
retlw	0x24	; High Byte, ADRES = 170, Forward Power = 4.243 uW
retlw	0x24	; High Byte, ADRES = 174, Forward Power = 4.655 uW
retlw	0x25	; High Byte, ADRES = 178, Forward Power = 5.108 uW
retlw	0x25	; High Byte, ADRES = 182, Forward Power = 5.604 uW
retlw	0x26	; High Byte, ADRES = 186, Forward Power = 6.148 uW
retlw	0x27	; High Byte, ADRES = 190, Forward Power = 6.745 uW
retlw	0x27	; High Byte, ADRES = 194, Forward Power = 7.400 uW
retlw	0x28	; High Byte, ADRES = 198, Forward Power = 8.118 uW
retlw	0x28	; High Byte, ADRES = 202, Forward Power = 8.907 uW
retlw	0x29	; High Byte, ADRES = 206, Forward Power = 9.772 uW
retlw	0x29	; High Byte, ADRES = 210, Forward Power = 10.72 uW
retlw	0x2A	; High Byte, ADRES = 214, Forward Power = 11.76 uW
retlw	0x2A	; High Byte, ADRES = 218, Forward Power = 12.90 uW

retlw	0x2B	; High Byte, ADRES = 222, Forward Power = 14.16 uW
retlw	0x2C	; High Byte, ADRES = 226, Forward Power = 15.53 uW
retlw	0x2C	; High Byte, ADRES = 230, Forward Power = 17.04 uW
retlw	0x2C	; High Byte, ADRES = 234, Forward Power = 18.70 uW
retlw	0x2D	; High Byte, ADRES = 238, Forward Power = 20.51 uW
retlw	0x2D	; High Byte, ADRES = 242, Forward Power = 22.50 uW
retlw	0x2E	; High Byte, ADRES = 246, Forward Power = 24.69 uW
retlw	0x2F	; High Byte, ADRES = 250, Forward Power = 27.09 uW
retlw	0x2F	; High Byte, ADRES = 254, Forward Power = 29.72 uW
retlw	0x30	; High Byte, ADRES = 258, Forward Power = 32.60 uW
retlw	0x30	; High Byte, ADRES = 262, Forward Power = 35.77 uW
retlw	0x31	; High Byte, ADRES = 266, Forward Power = 39.24 uW
retlw	0x31	; High Byte, ADRES = 270, Forward Power = 43.05 uW
retlw	0x32	; High Byte, ADRES = 274, Forward Power = 47.23 uW
retlw	0x32	; High Byte, ADRES = 278, Forward Power = 51.82 uW
retlw	0x33	; High Byte, ADRES = 282, Forward Power = 56.85 uW
retlw	0x34	; High Byte, ADRES = 286, Forward Power = 62.38 uW
retlw	0x34	; High Byte, ADRES = 290, Forward Power = 68.43 uW
retlw	0x34	; High Byte, ADRES = 294, Forward Power = 75.08 uW
retlw	0x35	; High Byte, ADRES = 298, Forward Power = 82.37 uW
retlw	0x35	; High Byte, ADRES = 302, Forward Power = 90.37 uW
retlw	0x36	; High Byte, ADRES = 306, Forward Power = 99.15 uW
retlw	0x37	; High Byte, ADRES = 310, Forward Power = 108.8 uW
retlw	0x37	; High Byte, ADRES = 314, Forward Power = 119.3 uW
retlw	0x38	; High Byte, ADRES = 318, Forward Power = 130.9 uW
retlw	0x38	; High Byte, ADRES = 322, Forward Power = 143.6 uW
retlw	0x39	; High Byte, ADRES = 326, Forward Power = 157.6 uW
retlw	0x39	; High Byte, ADRES = 330, Forward Power = 172.9 uW
retlw	0x3A	; High Byte, ADRES = 334, Forward Power = 189.7 uW
retlw	0x3A	; High Byte, ADRES = 338, Forward Power = 208.1 uW
retlw	0x3B	; High Byte, ADRES = 342, Forward Power = 228.3 uW
retlw	0x3C	; High Byte, ADRES = 346, Forward Power = 250.5 uW
retlw	0x3C	; High Byte, ADRES = 350, Forward Power = 274.8 uW
retlw	0x3C	; High Byte, ADRES = 354, Forward Power = 301.5 uW
retlw	0x3D	; High Byte, ADRES = 358, Forward Power = 330.8 uW
retlw	0x3D	; High Byte, ADRES = 362, Forward Power = 362.9 uW
retlw	0x3E	; High Byte, ADRES = 366, Forward Power = 398.2 uW
retlw	0x3F	; High Byte, ADRES = 370, Forward Power = 436.8 uW
retlw	0x3F	; High Byte, ADRES = 374, Forward Power = 479.2 uW
retlw	0x40	; High Byte, ADRES = 378, Forward Power = 525.8 uW
retlw	0x40	; High Byte, ADRES = 382, Forward Power = 576.9 uW
retlw	0x41	; High Byte, ADRES = 386, Forward Power = 632.9 uW
retlw	0x41	; High Byte, ADRES = 390, Forward Power = 694.3 uW
retlw	0x42	; High Byte, ADRES = 394, Forward Power = 761.8 uW
retlw	0x42	; High Byte, ADRES = 398, Forward Power = 835.7 uW
retlw	0x43	; High Byte, ADRES = 402, Forward Power = 916.9 uW
retlw	0x44	; High Byte, ADRES = 406, Forward Power = 1.006 mW
retlw	0x44	; High Byte, ADRES = 410, Forward Power = 1.104 mW
retlw	0x44	; High Byte, ADRES = 414, Forward Power = 1.211 mW
retlw	0x45	; High Byte, ADRES = 418, Forward Power = 1.328 mW
retlw	0x45	; High Byte, ADRES = 422, Forward Power = 1.457 mW
retlw	0x46	; High Byte, ADRES = 426, Forward Power = 1.599 mW
retlw	0x47	; High Byte, ADRES = 430, Forward Power = 1.754 mW
retlw	0x47	; High Byte, ADRES = 434, Forward Power = 1.925 mW
retlw	0x48	; High Byte, ADRES = 438, Forward Power = 2.112 mW
retlw	0x48	; High Byte, ADRES = 442, Forward Power = 2.317 mW
retlw	0x49	; High Byte, ADRES = 446, Forward Power = 2.542 mW
retlw	0x49	; High Byte, ADRES = 450, Forward Power = 2.788 mW
retlw	0x4A	; High Byte, ADRES = 454, Forward Power = 3.059 mW
retlw	0x4A	; High Byte, ADRES = 458, Forward Power = 3.356 mW
retlw	0x4B	; High Byte, ADRES = 462, Forward Power = 3.682 mW
retlw	0x4C	; High Byte, ADRES = 466, Forward Power = 4.040 mW
retlw	0x4C	; High Byte, ADRES = 470, Forward Power = 4.432 mW
retlw	0x4C	; High Byte, ADRES = 474, Forward Power = 4.863 mW
retlw	0x4D	; High Byte, ADRES = 478, Forward Power = 5.335 mW
retlw	0x4D	; High Byte, ADRES = 482, Forward Power = 5.853 mW
retlw	0x4E	; High Byte, ADRES = 486, Forward Power = 6.421 mW
retlw	0x4F	; High Byte, ADRES = 490, Forward Power = 7.045 mW
retlw	0x4F	; High Byte, ADRES = 494, Forward Power = 7.729 mW
retlw	0x50	; High Byte, ADRES = 498, Forward Power = 8.480 mW
retlw	0x50	; High Byte, ADRES = 502, Forward Power = 9.303 mW
retlw	0x51	; High Byte, ADRES = 506, Forward Power = 10.21 mW
retlw	0x51	; High Byte, ADRES = 510, Forward Power = 11.20 mW
retlw	0x52	; High Byte, ADRES = 514, Forward Power = 12.29 mW
retlw	0x52	; High Byte, ADRES = 518, Forward Power = 13.48 mW
retlw	0x53	; High Byte, ADRES = 522, Forward Power = 14.79 mW
retlw	0x54	; High Byte, ADRES = 526, Forward Power = 16.22 mW
retlw	0x54	; High Byte, ADRES = 530, Forward Power = 17.80 mW
retlw	0x55	; High Byte, ADRES = 534, Forward Power = 19.53 mW
retlw	0x55	; High Byte, ADRES = 538, Forward Power = 21.42 mW
retlw	0x56	; High Byte, ADRES = 542, Forward Power = 23.50 mW

retlw	0x56	; High Byte, ADRES = 546, Forward Power = 25.79 mW
retlw	0x57	; High Byte, ADRES = 550, Forward Power = 28.29 mW
retlw	0x57	; High Byte, ADRES = 554, Forward Power = 31.04 mW
retlw	0x58	; High Byte, ADRES = 558, Forward Power = 34.05 mW
retlw	0x58	; High Byte, ADRES = 562, Forward Power = 37.36 mW
retlw	0x59	; High Byte, ADRES = 566, Forward Power = 40.99 mW
retlw	0x59	; High Byte, ADRES = 570, Forward Power = 44.97 mW
retlw	0x5A	; High Byte, ADRES = 574, Forward Power = 49.34 mW
retlw	0x5A	; High Byte, ADRES = 578, Forward Power = 54.13 mW
retlw	0x5B	; High Byte, ADRES = 582, Forward Power = 59.38 mW
retlw	0x5C	; High Byte, ADRES = 586, Forward Power = 65.15 mW
retlw	0x5C	; High Byte, ADRES = 590, Forward Power = 71.48 mW
retlw	0x5D	; High Byte, ADRES = 594, Forward Power = 78.42 mW
retlw	0x5D	; High Byte, ADRES = 598, Forward Power = 86.03 mW
retlw	0x5E	; High Byte, ADRES = 602, Forward Power = 94.39 mW
retlw	0x5E	; High Byte, ADRES = 606, Forward Power = 103.6 mW
retlw	0x5F	; High Byte, ADRES = 610, Forward Power = 113.6 mW
retlw	0x5F	; High Byte, ADRES = 614, Forward Power = 124.6 mW
retlw	0x60	; High Byte, ADRES = 618, Forward Power = 136.8 mW
retlw	0x60	; High Byte, ADRES = 622, Forward Power = 150.0 mW
retlw	0x61	; High Byte, ADRES = 626, Forward Power = 164.6 mW
retlw	0x61	; High Byte, ADRES = 630, Forward Power = 180.6 mW
retlw	0x62	; High Byte, ADRES = 634, Forward Power = 198.1 mW
retlw	0x62	; High Byte, ADRES = 638, Forward Power = 217.4 mW
retlw	0x63	; High Byte, ADRES = 642, Forward Power = 238.5 mW
retlw	0x64	; High Byte, ADRES = 646, Forward Power = 261.6 mW
retlw	0x64	; High Byte, ADRES = 650, Forward Power = 287.0 mW
retlw	0x65	; High Byte, ADRES = 654, Forward Power = 314.9 mW
retlw	0x65	; High Byte, ADRES = 658, Forward Power = 345.5 mW
retlw	0x66	; High Byte, ADRES = 662, Forward Power = 379.1 mW
retlw	0x66	; High Byte, ADRES = 666, Forward Power = 415.9 mW
retlw	0x67	; High Byte, ADRES = 670, Forward Power = 456.3 mW
retlw	0x68	; High Byte, ADRES = 674, Forward Power = 500.6 mW
retlw	0x68	; High Byte, ADRES = 678, Forward Power = 549.2 mW
retlw	0x68	; High Byte, ADRES = 682, Forward Power = 602.5 mW
retlw	0x69	; High Byte, ADRES = 686, Forward Power = 661.0 mW
retlw	0x69	; High Byte, ADRES = 690, Forward Power = 725.2 mW
retlw	0x6A	; High Byte, ADRES = 694, Forward Power = 795.6 mW
retlw	0x6A	; High Byte, ADRES = 698, Forward Power = 872.9 mW
retlw	0x6B	; High Byte, ADRES = 702, Forward Power = 957.7 mW
retlw	0x6C	; High Byte, ADRES = 706, Forward Power = 1.051 W
retlw	0x6C	; High Byte, ADRES = 710, Forward Power = 1.153 W
retlw	0x6D	; High Byte, ADRES = 714, Forward Power = 1.265 W
retlw	0x6D	; High Byte, ADRES = 718, Forward Power = 1.388 W
retlw	0x6E	; High Byte, ADRES = 722, Forward Power = 1.522 W
retlw	0x6E	; High Byte, ADRES = 726, Forward Power = 1.670 W
retlw	0x6F	; High Byte, ADRES = 730, Forward Power = 1.832 W
retlw	0x70	; High Byte, ADRES = 734, Forward Power = 2.010 W
retlw	0x70	; High Byte, ADRES = 738, Forward Power = 2.205 W
retlw	0x70	; High Byte, ADRES = 742, Forward Power = 2.420 W
retlw	0x71	; High Byte, ADRES = 746, Forward Power = 2.655 W
retlw	0x71	; High Byte, ADRES = 750, Forward Power = 2.912 W
retlw	0x72	; High Byte, ADRES = 754, Forward Power = 3.195 W
retlw	0x73	; High Byte, ADRES = 758, Forward Power = 3.506 W
retlw	0x73	; High Byte, ADRES = 762, Forward Power = 3.846 W
retlw	0x74	; High Byte, ADRES = 766, Forward Power = 4.219 W
retlw	0x74	; High Byte, ADRES = 770, Forward Power = 4.629 W
retlw	0x75	; High Byte, ADRES = 774, Forward Power = 5.079 W
retlw	0x75	; High Byte, ADRES = 778, Forward Power = 5.572 W
retlw	0x76	; High Byte, ADRES = 782, Forward Power = 6.113 W
retlw	0x76	; High Byte, ADRES = 786, Forward Power = 6.707 W
retlw	0x77	; High Byte, ADRES = 790, Forward Power = 7.358 W
retlw	0x78	; High Byte, ADRES = 794, Forward Power = 8.073 W
retlw	0x78	; High Byte, ADRES = 798, Forward Power = 8.857 W
retlw	0x78	; High Byte, ADRES = 802, Forward Power = 9.717 W
retlw	0x79	; High Byte, ADRES = 806, Forward Power = 10.66 W
retlw	0x79	; High Byte, ADRES = 810, Forward Power = 11.70 W
retlw	0x7A	; High Byte, ADRES = 814, Forward Power = 12.83 W
retlw	0x7B	; High Byte, ADRES = 818, Forward Power = 14.08 W
retlw	0x7B	; High Byte, ADRES = 822, Forward Power = 15.44 W
retlw	0x7C	; High Byte, ADRES = 826, Forward Power = 16.94 W
retlw	0x7C	; High Byte, ADRES = 830, Forward Power = 18.59 W
retlw	0x7D	; High Byte, ADRES = 834, Forward Power = 20.40 W
retlw	0x7D	; High Byte, ADRES = 838, Forward Power = 22.38 W
retlw	0x7E	; High Byte, ADRES = 842, Forward Power = 24.55 W
retlw	0x7E	; High Byte, ADRES = 846, Forward Power = 26.93 W
retlw	0x7F	; High Byte, ADRES = 850, Forward Power = 29.55 W
retlw	0x80	; High Byte, ADRES = 854, Forward Power = 32.42 W
retlw	0x80	; High Byte, ADRES = 858, Forward Power = 35.57 W
retlw	0x80	; High Byte, ADRES = 862, Forward Power = 39.02 W
retlw	0x81	; High Byte, ADRES = 866, Forward Power = 42.81 W

retlw	0x81	; High Byte, ADRES = 870, Forward Power = 46.97 W
retlw	0x82	; High Byte, ADRES = 874, Forward Power = 51.53 W
retlw	0x83	; High Byte, ADRES = 878, Forward Power = 56.53 W
retlw	0x83	; High Byte, ADRES = 882, Forward Power = 62.02 W
retlw	0x84	; High Byte, ADRES = 886, Forward Power = 68.05 W
retlw	0x84	; High Byte, ADRES = 890, Forward Power = 74.66 W
retlw	0x85	; High Byte, ADRES = 894, Forward Power = 81.91 W
retlw	0x85	; High Byte, ADRES = 898, Forward Power = 89.86 W
retlw	0x86	; High Byte, ADRES = 902, Forward Power = 98.59 W
retlw	0x86	; High Byte, ADRES = 906, Forward Power = 108.2 W
retlw	0x87	; High Byte, ADRES = 910, Forward Power = 118.7 W
retlw	0x88	; High Byte, ADRES = 914, Forward Power = 130.2 W
retlw	0x88	; High Byte, ADRES = 918, Forward Power = 142.8 W
retlw	0x88	; High Byte, ADRES = 922, Forward Power = 156.7 W
retlw	0x89	; High Byte, ADRES = 926, Forward Power = 171.9 W
retlw	0x89	; High Byte, ADRES = 930, Forward Power = 188.6 W
retlw	0x8A	; High Byte, ADRES = 934, Forward Power = 206.9 W
retlw	0x8B	; High Byte, ADRES = 938, Forward Power = 227.0 W
retlw	0x8B	; High Byte, ADRES = 942, Forward Power = 249.1 W
retlw	0x8C	; High Byte, ADRES = 946, Forward Power = 273.3 W
retlw	0x8C	; High Byte, ADRES = 950, Forward Power = 299.8 W
retlw	0x8D	; High Byte, ADRES = 954, Forward Power = 328.9 W
retlw	0x8D	; High Byte, ADRES = 958, Forward Power = 360.9 W
retlw	0x8E	; High Byte, ADRES = 962, Forward Power = 395.9 W
retlw	0x8E	; High Byte, ADRES = 966, Forward Power = 434.4 W
retlw	0x8F	; High Byte, ADRES = 970, Forward Power = 476.6 W
retlw	0x90	; High Byte, ADRES = 974, Forward Power = 522.8 W
retlw	0x90	; High Byte, ADRES = 978, Forward Power = 573.6 W
retlw	0x90	; High Byte, ADRES = 982, Forward Power = 629.3 W
retlw	0x91	; High Byte, ADRES = 986, Forward Power = 690.4 W
retlw	0x91	; High Byte, ADRES = 990, Forward Power = 757.5 W
retlw	0x92	; High Byte, ADRES = 994, Forward Power = 831.0 W
retlw	0x93	; High Byte, ADRES = 998, Forward Power = 911.7 W
retlw	0x93	; High Byte, ADRES = 1002, Forward Power = 1,000.3 W
retlw	0x94	; High Byte, ADRES = 1006, Forward Power = 1,097.4 W
retlw	0x94	; High Byte, ADRES = 1010, Forward Power = 1,204.0 W
retlw	0x95	; High Byte, ADRES = 1014, Forward Power = 1,320.9 W
retlw	0x95	; High Byte, ADRES = 1018, Forward Power = 1,449.2 W
retlw	0x96	; High Byte, ADRES = 1022, Forward Power = 1,590.0 W

FHighByteTable3

retlw	0x0D	; High Byte, ADRES = 3, Forward Power = 88.55 nW
retlw	0x0E	; High Byte, ADRES = 7, Forward Power = 97.15 nW
retlw	0x0F	; High Byte, ADRES = 11, Forward Power = 106.6 nW
retlw	0x0F	; High Byte, ADRES = 15, Forward Power = 116.9 nW
retlw	0x10	; High Byte, ADRES = 19, Forward Power = 128.3 nW
retlw	0x10	; High Byte, ADRES = 23, Forward Power = 140.8 nW
retlw	0x11	; High Byte, ADRES = 27, Forward Power = 154.4 nW
retlw	0x11	; High Byte, ADRES = 31, Forward Power = 169.4 nW
retlw	0x12	; High Byte, ADRES = 35, Forward Power = 185.9 nW
retlw	0x12	; High Byte, ADRES = 39, Forward Power = 203.9 nW
retlw	0x13	; High Byte, ADRES = 43, Forward Power = 223.7 nW
retlw	0x14	; High Byte, ADRES = 47, Forward Power = 245.4 nW
retlw	0x14	; High Byte, ADRES = 51, Forward Power = 269.3 nW
retlw	0x14	; High Byte, ADRES = 55, Forward Power = 295.4 nW
retlw	0x15	; High Byte, ADRES = 59, Forward Power = 324.1 nW
retlw	0x15	; High Byte, ADRES = 63, Forward Power = 355.6 nW
retlw	0x16	; High Byte, ADRES = 67, Forward Power = 390.1 nW
retlw	0x17	; High Byte, ADRES = 71, Forward Power = 428.0 nW
retlw	0x17	; High Byte, ADRES = 75, Forward Power = 469.6 nW
retlw	0x18	; High Byte, ADRES = 79, Forward Power = 515.2 nW
retlw	0x18	; High Byte, ADRES = 83, Forward Power = 565.2 nW
retlw	0x19	; High Byte, ADRES = 87, Forward Power = 620.1 nW
retlw	0x19	; High Byte, ADRES = 91, Forward Power = 680.4 nW
retlw	0x1A	; High Byte, ADRES = 95, Forward Power = 746.4 nW
retlw	0x1A	; High Byte, ADRES = 99, Forward Power = 818.9 nW
retlw	0x1B	; High Byte, ADRES = 103, Forward Power = 898.4 nW
retlw	0x1C	; High Byte, ADRES = 107, Forward Power = 985.7 nW
retlw	0x1C	; High Byte, ADRES = 111, Forward Power = 1.081 uW
retlw	0x1C	; High Byte, ADRES = 115, Forward Power = 1.186 uW
retlw	0x1D	; High Byte, ADRES = 119, Forward Power = 1.302 uW
retlw	0x1D	; High Byte, ADRES = 123, Forward Power = 1.428 uW
retlw	0x1E	; High Byte, ADRES = 127, Forward Power = 1.567 uW
retlw	0x1F	; High Byte, ADRES = 131, Forward Power = 1.719 uW
retlw	0x1F	; High Byte, ADRES = 135, Forward Power = 1.886 uW
retlw	0x20	; High Byte, ADRES = 139, Forward Power = 2.069 uW
retlw	0x20	; High Byte, ADRES = 143, Forward Power = 2.270 uW
retlw	0x21	; High Byte, ADRES = 147, Forward Power = 2.490 uW
retlw	0x21	; High Byte, ADRES = 151, Forward Power = 2.732 uW
retlw	0x22	; High Byte, ADRES = 155, Forward Power = 2.998 uW
retlw	0x22	; High Byte, ADRES = 159, Forward Power = 3.289 uW

retlw	0x23	; High Byte, ADRES = 163, Forward Power = 3.608 uW
retlw	0x24	; High Byte, ADRES = 167, Forward Power = 3.958 uW
retlw	0x24	; High Byte, ADRES = 171, Forward Power = 4.343 uW
retlw	0x24	; High Byte, ADRES = 175, Forward Power = 4.765 uW
retlw	0x25	; High Byte, ADRES = 179, Forward Power = 5.227 uW
retlw	0x26	; High Byte, ADRES = 183, Forward Power = 5.735 uW
retlw	0x26	; High Byte, ADRES = 187, Forward Power = 6.292 uW
retlw	0x27	; High Byte, ADRES = 191, Forward Power = 6.903 uW
retlw	0x27	; High Byte, ADRES = 195, Forward Power = 7.573 uW
retlw	0x28	; High Byte, ADRES = 199, Forward Power = 8.309 uW
retlw	0x28	; High Byte, ADRES = 203, Forward Power = 9.116 uW
retlw	0x29	; High Byte, ADRES = 207, Forward Power = 10.00 uW
retlw	0x29	; High Byte, ADRES = 211, Forward Power = 10.97 uW
retlw	0x2A	; High Byte, ADRES = 215, Forward Power = 12.04 uW
retlw	0x2A	; High Byte, ADRES = 219, Forward Power = 13.21 uW
retlw	0x2B	; High Byte, ADRES = 223, Forward Power = 14.49 uW
retlw	0x2C	; High Byte, ADRES = 227, Forward Power = 15.90 uW
retlw	0x2C	; High Byte, ADRES = 231, Forward Power = 17.44 uW
retlw	0x2D	; High Byte, ADRES = 235, Forward Power = 19.13 uW
retlw	0x2D	; High Byte, ADRES = 239, Forward Power = 20.99 uW
retlw	0x2E	; High Byte, ADRES = 243, Forward Power = 23.03 uW
retlw	0x2E	; High Byte, ADRES = 247, Forward Power = 25.27 uW
retlw	0x2F	; High Byte, ADRES = 251, Forward Power = 27.72 uW
retlw	0x2F	; High Byte, ADRES = 255, Forward Power = 30.41 uW
retlw	0x30	; High Byte, ADRES = 259, Forward Power = 33.37 uW
retlw	0x30	; High Byte, ADRES = 263, Forward Power = 36.61 uW
retlw	0x31	; High Byte, ADRES = 267, Forward Power = 40.16 uW
retlw	0x31	; High Byte, ADRES = 271, Forward Power = 44.06 uW
retlw	0x32	; High Byte, ADRES = 275, Forward Power = 48.34 uW
retlw	0x32	; High Byte, ADRES = 279, Forward Power = 53.04 uW
retlw	0x33	; High Byte, ADRES = 283, Forward Power = 58.19 uW
retlw	0x34	; High Byte, ADRES = 287, Forward Power = 63.84 uW
retlw	0x34	; High Byte, ADRES = 291, Forward Power = 70.04 uW
retlw	0x35	; High Byte, ADRES = 295, Forward Power = 76.84 uW
retlw	0x35	; High Byte, ADRES = 299, Forward Power = 84.30 uW
retlw	0x36	; High Byte, ADRES = 303, Forward Power = 92.49 uW
retlw	0x36	; High Byte, ADRES = 307, Forward Power = 101.5 uW
retlw	0x37	; High Byte, ADRES = 311, Forward Power = 111.3 uW
retlw	0x38	; High Byte, ADRES = 315, Forward Power = 122.1 uW
retlw	0x38	; High Byte, ADRES = 319, Forward Power = 134.0 uW
retlw	0x38	; High Byte, ADRES = 323, Forward Power = 147.0 uW
retlw	0x39	; High Byte, ADRES = 327, Forward Power = 161.3 uW
retlw	0x39	; High Byte, ADRES = 331, Forward Power = 177.0 uW
retlw	0x3A	; High Byte, ADRES = 335, Forward Power = 194.1 uW
retlw	0x3A	; High Byte, ADRES = 339, Forward Power = 213.0 uW
retlw	0x3B	; High Byte, ADRES = 343, Forward Power = 233.7 uW
retlw	0x3C	; High Byte, ADRES = 347, Forward Power = 256.4 uW
retlw	0x3C	; High Byte, ADRES = 351, Forward Power = 281.3 uW
retlw	0x3D	; High Byte, ADRES = 355, Forward Power = 308.6 uW
retlw	0x3D	; High Byte, ADRES = 359, Forward Power = 338.5 uW
retlw	0x3E	; High Byte, ADRES = 363, Forward Power = 371.4 uW
retlw	0x3E	; High Byte, ADRES = 367, Forward Power = 407.5 uW
retlw	0x3F	; High Byte, ADRES = 371, Forward Power = 447.1 uW
retlw	0x40	; High Byte, ADRES = 375, Forward Power = 490.5 uW
retlw	0x40	; High Byte, ADRES = 379, Forward Power = 538.1 uW
retlw	0x40	; High Byte, ADRES = 383, Forward Power = 590.4 uW
retlw	0x41	; High Byte, ADRES = 387, Forward Power = 647.7 uW
retlw	0x41	; High Byte, ADRES = 391, Forward Power = 710.6 uW
retlw	0x42	; High Byte, ADRES = 395, Forward Power = 779.6 uW
retlw	0x43	; High Byte, ADRES = 399, Forward Power = 855.3 uW
retlw	0x43	; High Byte, ADRES = 403, Forward Power = 938.4 uW
retlw	0x44	; High Byte, ADRES = 407, Forward Power = 1.030 mW
retlw	0x44	; High Byte, ADRES = 411, Forward Power = 1.130 mW
retlw	0x45	; High Byte, ADRES = 415, Forward Power = 1.239 mW
retlw	0x45	; High Byte, ADRES = 419, Forward Power = 1.360 mW
retlw	0x46	; High Byte, ADRES = 423, Forward Power = 1.492 mW
retlw	0x46	; High Byte, ADRES = 427, Forward Power = 1.636 mW
retlw	0x47	; High Byte, ADRES = 431, Forward Power = 1.795 mW
retlw	0x48	; High Byte, ADRES = 435, Forward Power = 1.970 mW
retlw	0x48	; High Byte, ADRES = 439, Forward Power = 2.161 mW
retlw	0x48	; High Byte, ADRES = 443, Forward Power = 2.371 mW
retlw	0x49	; High Byte, ADRES = 447, Forward Power = 2.601 mW
retlw	0x49	; High Byte, ADRES = 451, Forward Power = 2.854 mW
retlw	0x4A	; High Byte, ADRES = 455, Forward Power = 3.131 mW
retlw	0x4B	; High Byte, ADRES = 459, Forward Power = 3.435 mW
retlw	0x4B	; High Byte, ADRES = 463, Forward Power = 3.769 mW
retlw	0x4C	; High Byte, ADRES = 467, Forward Power = 4.134 mW
retlw	0x4C	; High Byte, ADRES = 471, Forward Power = 4.536 mW
retlw	0x4D	; High Byte, ADRES = 475, Forward Power = 4.977 mW
retlw	0x4D	; High Byte, ADRES = 479, Forward Power = 5.460 mW
retlw	0x4E	; High Byte, ADRES = 483, Forward Power = 5.990 mW

retlw	0x4E	; High Byte, ADRES = 487, Forward Power = 6.572 mW
retlw	0x4F	; High Byte, ADRES = 491, Forward Power = 7.210 mW
retlw	0x50	; High Byte, ADRES = 495, Forward Power = 7.910 mW
retlw	0x50	; High Byte, ADRES = 499, Forward Power = 8.678 mW
retlw	0x50	; High Byte, ADRES = 503, Forward Power = 9.521 mW
retlw	0x51	; High Byte, ADRES = 507, Forward Power = 10.45 mW
retlw	0x51	; High Byte, ADRES = 511, Forward Power = 11.46 mW
retlw	0x52	; High Byte, ADRES = 515, Forward Power = 12.57 mW
retlw	0x53	; High Byte, ADRES = 519, Forward Power = 13.79 mW
retlw	0x53	; High Byte, ADRES = 523, Forward Power = 15.13 mW
retlw	0x54	; High Byte, ADRES = 527, Forward Power = 16.60 mW
retlw	0x54	; High Byte, ADRES = 531, Forward Power = 18.22 mW
retlw	0x55	; High Byte, ADRES = 535, Forward Power = 19.99 mW
retlw	0x55	; High Byte, ADRES = 539, Forward Power = 21.93 mW
retlw	0x56	; High Byte, ADRES = 543, Forward Power = 24.06 mW
retlw	0x56	; High Byte, ADRES = 547, Forward Power = 26.39 mW
retlw	0x57	; High Byte, ADRES = 551, Forward Power = 28.95 mW
retlw	0x58	; High Byte, ADRES = 555, Forward Power = 31.77 mW
retlw	0x58	; High Byte, ADRES = 559, Forward Power = 34.85 mW
retlw	0x58	; High Byte, ADRES = 563, Forward Power = 38.24 mW
retlw	0x59	; High Byte, ADRES = 567, Forward Power = 41.95 mW
retlw	0x59	; High Byte, ADRES = 571, Forward Power = 46.02 mW
retlw	0x5A	; High Byte, ADRES = 575, Forward Power = 50.49 mW
retlw	0x5B	; High Byte, ADRES = 579, Forward Power = 55.40 mW
retlw	0x5B	; High Byte, ADRES = 583, Forward Power = 60.78 mW
retlw	0x5C	; High Byte, ADRES = 587, Forward Power = 66.68 mW
retlw	0x5C	; High Byte, ADRES = 591, Forward Power = 73.15 mW
retlw	0x5D	; High Byte, ADRES = 595, Forward Power = 80.26 mW
retlw	0x5D	; High Byte, ADRES = 599, Forward Power = 88.05 mW
retlw	0x5E	; High Byte, ADRES = 603, Forward Power = 96.60 mW
retlw	0x5E	; High Byte, ADRES = 607, Forward Power = 106.0 mW
retlw	0x5F	; High Byte, ADRES = 611, Forward Power = 116.3 mW
retlw	0x60	; High Byte, ADRES = 615, Forward Power = 127.6 mW
retlw	0x60	; High Byte, ADRES = 619, Forward Power = 140.0 mW
retlw	0x60	; High Byte, ADRES = 623, Forward Power = 153.5 mW
retlw	0x61	; High Byte, ADRES = 627, Forward Power = 168.5 mW
retlw	0x61	; High Byte, ADRES = 631, Forward Power = 184.8 mW
retlw	0x62	; High Byte, ADRES = 635, Forward Power = 202.8 mW
retlw	0x63	; High Byte, ADRES = 639, Forward Power = 222.5 mW
retlw	0x63	; High Byte, ADRES = 643, Forward Power = 244.1 mW
retlw	0x64	; High Byte, ADRES = 647, Forward Power = 267.8 mW
retlw	0x64	; High Byte, ADRES = 651, Forward Power = 293.8 mW
retlw	0x65	; High Byte, ADRES = 655, Forward Power = 322.3 mW
retlw	0x65	; High Byte, ADRES = 659, Forward Power = 353.6 mW
retlw	0x66	; High Byte, ADRES = 663, Forward Power = 387.9 mW
retlw	0x66	; High Byte, ADRES = 667, Forward Power = 425.6 mW
retlw	0x67	; High Byte, ADRES = 671, Forward Power = 467.0 mW
retlw	0x68	; High Byte, ADRES = 675, Forward Power = 512.3 mW
retlw	0x68	; High Byte, ADRES = 679, Forward Power = 562.1 mW
retlw	0x68	; High Byte, ADRES = 683, Forward Power = 616.6 mW
retlw	0x69	; High Byte, ADRES = 687, Forward Power = 676.5 mW
retlw	0x69	; High Byte, ADRES = 691, Forward Power = 742.2 mW
retlw	0x6A	; High Byte, ADRES = 695, Forward Power = 814.3 mW
retlw	0x6B	; High Byte, ADRES = 699, Forward Power = 893.4 mW
retlw	0x6B	; High Byte, ADRES = 703, Forward Power = 980.1 mW
retlw	0x6C	; High Byte, ADRES = 707, Forward Power = 1.075 W
retlw	0x6C	; High Byte, ADRES = 711, Forward Power = 1.180 W
retlw	0x6D	; High Byte, ADRES = 715, Forward Power = 1.294 W
retlw	0x6D	; High Byte, ADRES = 719, Forward Power = 1.420 W
retlw	0x6E	; High Byte, ADRES = 723, Forward Power = 1.558 W
retlw	0x6E	; High Byte, ADRES = 727, Forward Power = 1.709 W
retlw	0x6F	; High Byte, ADRES = 731, Forward Power = 1.875 W
retlw	0x70	; High Byte, ADRES = 735, Forward Power = 2.057 W
retlw	0x70	; High Byte, ADRES = 739, Forward Power = 2.257 W
retlw	0x70	; High Byte, ADRES = 743, Forward Power = 2.476 W
retlw	0x71	; High Byte, ADRES = 747, Forward Power = 2.717 W
retlw	0x71	; High Byte, ADRES = 751, Forward Power = 2.981 W
retlw	0x72	; High Byte, ADRES = 755, Forward Power = 3.270 W
retlw	0x73	; High Byte, ADRES = 759, Forward Power = 3.588 W
retlw	0x73	; High Byte, ADRES = 763, Forward Power = 3.936 W
retlw	0x74	; High Byte, ADRES = 767, Forward Power = 4.318 W
retlw	0x74	; High Byte, ADRES = 771, Forward Power = 4.738 W
retlw	0x75	; High Byte, ADRES = 775, Forward Power = 5.198 W
retlw	0x75	; High Byte, ADRES = 779, Forward Power = 5.703 W
retlw	0x76	; High Byte, ADRES = 783, Forward Power = 6.256 W
retlw	0x76	; High Byte, ADRES = 787, Forward Power = 6.864 W
retlw	0x77	; High Byte, ADRES = 791, Forward Power = 7.531 W
retlw	0x78	; High Byte, ADRES = 795, Forward Power = 8.262 W
retlw	0x78	; High Byte, ADRES = 799, Forward Power = 9.064 W
retlw	0x78	; High Byte, ADRES = 803, Forward Power = 9.945 W
retlw	0x79	; High Byte, ADRES = 807, Forward Power = 10.91 W

```

retlw    0x79      ; High Byte, ADRES = 811, Forward Power = 11.97 W
retlw    0x7A      ; High Byte, ADRES = 815, Forward Power = 13.13 W
retlw    0x7B      ; High Byte, ADRES = 819, Forward Power = 14.41 W
retlw    0x7B      ; High Byte, ADRES = 823, Forward Power = 15.81 W
retlw    0x7C      ; High Byte, ADRES = 827, Forward Power = 17.34 W
retlw    0x7C      ; High Byte, ADRES = 831, Forward Power = 19.03 W
retlw    0x7D      ; High Byte, ADRES = 835, Forward Power = 20.87 W
retlw    0x7D      ; High Byte, ADRES = 839, Forward Power = 22.90 W
retlw    0x7E      ; High Byte, ADRES = 843, Forward Power = 25.13 W
retlw    0x7E      ; High Byte, ADRES = 847, Forward Power = 27.57 W
retlw    0x7F      ; High Byte, ADRES = 851, Forward Power = 30.24 W
retlw    0x80      ; High Byte, ADRES = 855, Forward Power = 33.18 W
retlw    0x80      ; High Byte, ADRES = 859, Forward Power = 36.40 W
retlw    0x80      ; High Byte, ADRES = 863, Forward Power = 39.94 W
retlw    0x81      ; High Byte, ADRES = 867, Forward Power = 43.81 W
retlw    0x82      ; High Byte, ADRES = 871, Forward Power = 48.07 W
retlw    0x82      ; High Byte, ADRES = 875, Forward Power = 52.74 W
retlw    0x83      ; High Byte, ADRES = 879, Forward Power = 57.86 W
retlw    0x83      ; High Byte, ADRES = 883, Forward Power = 63.48 W
retlw    0x84      ; High Byte, ADRES = 887, Forward Power = 69.64 W
retlw    0x84      ; High Byte, ADRES = 891, Forward Power = 76.41 W
retlw    0x85      ; High Byte, ADRES = 895, Forward Power = 83.83 W
retlw    0x85      ; High Byte, ADRES = 899, Forward Power = 91.97 W
retlw    0x86      ; High Byte, ADRES = 903, Forward Power = 100.9 W
retlw    0x86      ; High Byte, ADRES = 907, Forward Power = 110.7 W
retlw    0x87      ; High Byte, ADRES = 911, Forward Power = 121.4 W
retlw    0x88      ; High Byte, ADRES = 915, Forward Power = 133.2 W
retlw    0x88      ; High Byte, ADRES = 919, Forward Power = 146.2 W
retlw    0x89      ; High Byte, ADRES = 923, Forward Power = 160.4 W
retlw    0x89      ; High Byte, ADRES = 927, Forward Power = 176.0 W
retlw    0x8A      ; High Byte, ADRES = 931, Forward Power = 193.0 W
retlw    0x8A      ; High Byte, ADRES = 935, Forward Power = 211.8 W
retlw    0x8B      ; High Byte, ADRES = 939, Forward Power = 232.4 W
retlw    0x8B      ; High Byte, ADRES = 943, Forward Power = 254.9 W
retlw    0x8C      ; High Byte, ADRES = 947, Forward Power = 279.7 W
retlw    0x8C      ; High Byte, ADRES = 951, Forward Power = 306.8 W
retlw    0x8D      ; High Byte, ADRES = 955, Forward Power = 336.6 W
retlw    0x8D      ; High Byte, ADRES = 959, Forward Power = 369.3 W
retlw    0x8E      ; High Byte, ADRES = 963, Forward Power = 405.2 W
retlw    0x8E      ; High Byte, ADRES = 967, Forward Power = 444.6 W
retlw    0x8F      ; High Byte, ADRES = 971, Forward Power = 487.7 W
retlw    0x90      ; High Byte, ADRES = 975, Forward Power = 535.1 W
retlw    0x90      ; High Byte, ADRES = 979, Forward Power = 587.1 W
retlw    0x91      ; High Byte, ADRES = 983, Forward Power = 644.1 W
retlw    0x91      ; High Byte, ADRES = 987, Forward Power = 706.6 W
retlw    0x92      ; High Byte, ADRES = 991, Forward Power = 775.2 W
retlw    0x92      ; High Byte, ADRES = 995, Forward Power = 850.5 W
retlw    0x93      ; High Byte, ADRES = 999, Forward Power = 933.1 W
retlw    0x93      ; High Byte, ADRES = 1003, Forward Power = 1,023.7 W
retlw    0x94      ; High Byte, ADRES = 1007, Forward Power = 1,123.2 W
retlw    0x94      ; High Byte, ADRES = 1011, Forward Power = 1,232.2 W
retlw    0x95      ; High Byte, ADRES = 1015, Forward Power = 1,351.9 W
retlw    0x95      ; High Byte, ADRES = 1019, Forward Power = 1,483.2 W
retlw    0x96      ; High Byte, ADRES = 1023, Forward Power = 1,627.2 W

```

```

;*****
; Lookup Table for Reflected (Channel 1) Power
; Calibration dBm = .09965*ADC + -41.50

```

```

RLowByteTable0
retlw    0xC0      ; Low Byte, ADRES = 0, Reflected Power = 70.79 nW
retlw    0x35      ; Low Byte, ADRES = 4, Reflected Power = 77.60 nW
retlw    0xB5      ; Low Byte, ADRES = 8, Reflected Power = 85.06 nW
retlw    0x42      ; Low Byte, ADRES = 12, Reflected Power = 93.24 nW
retlw    0xDC      ; Low Byte, ADRES = 16, Reflected Power = 102.2 nW
retlw    0x85      ; Low Byte, ADRES = 20, Reflected Power = 112.0 nW
retlw    0x1F      ; Low Byte, ADRES = 24, Reflected Power = 122.8 nW
retlw    0x84      ; Low Byte, ADRES = 28, Reflected Power = 134.6 nW
retlw    0xF3      ; Low Byte, ADRES = 32, Reflected Power = 147.5 nW
retlw    0x6D      ; Low Byte, ADRES = 36, Reflected Power = 161.7 nW
retlw    0xF3      ; Low Byte, ADRES = 40, Reflected Power = 177.3 nW
retlw    0x85      ; Low Byte, ADRES = 44, Reflected Power = 194.3 nW
retlw    0x25      ; Low Byte, ADRES = 48, Reflected Power = 213.0 nW
retlw    0xD5      ; Low Byte, ADRES = 52, Reflected Power = 233.4 nW
retlw    0x4B      ; Low Byte, ADRES = 56, Reflected Power = 255.9 nW
retlw    0xB5      ; Low Byte, ADRES = 60, Reflected Power = 280.5 nW
retlw    0x28      ; Low Byte, ADRES = 64, Reflected Power = 307.4 nW
retlw    0xA7      ; Low Byte, ADRES = 68, Reflected Power = 337.0 nW

```

retlw	0x33	; Low Byte, ADRES = 72, Reflected Power = 369.4 nW
retlw	0xCB	; Low Byte, ADRES = 76, Reflected Power = 404.9 nW
retlw	0x72	; Low Byte, ADRES = 80, Reflected Power = 443.8 nW
retlw	0x15	; Low Byte, ADRES = 84, Reflected Power = 486.5 nW
retlw	0x79	; Low Byte, ADRES = 88, Reflected Power = 533.2 nW
retlw	0xE7	; Low Byte, ADRES = 92, Reflected Power = 584.5 nW
retlw	0x60	; Low Byte, ADRES = 96, Reflected Power = 640.7 nW
retlw	0xE4	; Low Byte, ADRES = 100, Reflected Power = 702.3 nW
retlw	0x75	; Low Byte, ADRES = 104, Reflected Power = 769.8 nW
retlw	0x14	; Low Byte, ADRES = 108, Reflected Power = 843.8 nW
retlw	0xC2	; Low Byte, ADRES = 112, Reflected Power = 924.9 nW
retlw	0x41	; Low Byte, ADRES = 116, Reflected Power = 1.014 uW
retlw	0xA9	; Low Byte, ADRES = 120, Reflected Power = 1.111 uW
retlw	0x1C	; Low Byte, ADRES = 124, Reflected Power = 1.218 uW
retlw	0x9A	; Low Byte, ADRES = 128, Reflected Power = 1.335 uW
retlw	0x23	; Low Byte, ADRES = 132, Reflected Power = 1.463 uW
retlw	0xBA	; Low Byte, ADRES = 136, Reflected Power = 1.604 uW
retlw	0x60	; Low Byte, ADRES = 140, Reflected Power = 1.758 uW
retlw	0x0B	; Low Byte, ADRES = 144, Reflected Power = 1.927 uW
retlw	0x6E	; Low Byte, ADRES = 148, Reflected Power = 2.113 uW
retlw	0xDB	; Low Byte, ADRES = 152, Reflected Power = 2.316 uW
retlw	0x53	; Low Byte, ADRES = 156, Reflected Power = 2.538 uW
retlw	0xD6	; Low Byte, ADRES = 160, Reflected Power = 2.782 uW
retlw	0x65	; Low Byte, ADRES = 164, Reflected Power = 3.050 uW
retlw	0x03	; Low Byte, ADRES = 168, Reflected Power = 3.343 uW
retlw	0xAF	; Low Byte, ADRES = 172, Reflected Power = 3.664 uW
retlw	0x36	; Low Byte, ADRES = 176, Reflected Power = 4.016 uW
retlw	0x9E	; Low Byte, ADRES = 180, Reflected Power = 4.403 uW
retlw	0x0F	; Low Byte, ADRES = 184, Reflected Power = 4.826 uW
retlw	0x8C	; Low Byte, ADRES = 188, Reflected Power = 5.290 uW
retlw	0x14	; Low Byte, ADRES = 192, Reflected Power = 5.798 uW
retlw	0xAA	; Low Byte, ADRES = 196, Reflected Power = 6.355 uW
retlw	0x4E	; Low Byte, ADRES = 200, Reflected Power = 6.966 uW
retlw	0x01	; Low Byte, ADRES = 204, Reflected Power = 7.636 uW
retlw	0x63	; Low Byte, ADRES = 208, Reflected Power = 8.370 uW
retlw	0xCF	; Low Byte, ADRES = 212, Reflected Power = 9.174 uW
retlw	0x46	; Low Byte, ADRES = 216, Reflected Power = 10.06 uW
retlw	0xC7	; Low Byte, ADRES = 220, Reflected Power = 11.02 uW
retlw	0x56	; Low Byte, ADRES = 224, Reflected Power = 12.08 uW
retlw	0xF2	; Low Byte, ADRES = 228, Reflected Power = 13.24 uW
retlw	0x9C	; Low Byte, ADRES = 232, Reflected Power = 14.52 uW
retlw	0x2C	; Low Byte, ADRES = 236, Reflected Power = 15.91 uW
retlw	0x93	; Low Byte, ADRES = 240, Reflected Power = 17.44 uW
retlw	0x03	; Low Byte, ADRES = 244, Reflected Power = 19.12 uW
retlw	0x7E	; Low Byte, ADRES = 248, Reflected Power = 20.96 uW
retlw	0x06	; Low Byte, ADRES = 252, Reflected Power = 22.97 uW
retlw	0x9A	; Low Byte, ADRES = 256, Reflected Power = 25.18 uW
retlw	0x3C	; Low Byte, ADRES = 260, Reflected Power = 27.60 uW
retlw	0xEE	; Low Byte, ADRES = 264, Reflected Power = 30.25 uW
retlw	0x59	; Low Byte, ADRES = 268, Reflected Power = 33.16 uW
retlw	0xC4	; Low Byte, ADRES = 272, Reflected Power = 36.35 uW
retlw	0x39	; Low Byte, ADRES = 276, Reflected Power = 39.84 uW
retlw	0xB9	; Low Byte, ADRES = 280, Reflected Power = 43.67 uW
retlw	0x46	; Low Byte, ADRES = 284, Reflected Power = 47.87 uW
retlw	0xE1	; Low Byte, ADRES = 288, Reflected Power = 52.47 uW
retlw	0x8A	; Low Byte, ADRES = 292, Reflected Power = 57.51 uW
retlw	0x22	; Low Byte, ADRES = 296, Reflected Power = 63.04 uW
retlw	0x87	; Low Byte, ADRES = 300, Reflected Power = 69.10 uW
retlw	0xF7	; Low Byte, ADRES = 304, Reflected Power = 75.75 uW
retlw	0x71	; Low Byte, ADRES = 308, Reflected Power = 83.03 uW
retlw	0xF7	; Low Byte, ADRES = 312, Reflected Power = 91.01 uW
retlw	0x8A	; Low Byte, ADRES = 316, Reflected Power = 99.76 uW
retlw	0x2B	; Low Byte, ADRES = 320, Reflected Power = 109.3 uW
retlw	0xDB	; Low Byte, ADRES = 324, Reflected Power = 119.9 uW
retlw	0x4E	; Low Byte, ADRES = 328, Reflected Power = 131.4 uW
retlw	0xB8	; Low Byte, ADRES = 332, Reflected Power = 144.0 uW
retlw	0x2C	; Low Byte, ADRES = 336, Reflected Power = 157.8 uW
retlw	0xAB	; Low Byte, ADRES = 340, Reflected Power = 173.0 uW
retlw	0x37	; Low Byte, ADRES = 344, Reflected Power = 189.7 uW
retlw	0xD0	; Low Byte, ADRES = 348, Reflected Power = 207.9 uW
retlw	0x77	; Low Byte, ADRES = 352, Reflected Power = 227.9 uW
retlw	0x18	; Low Byte, ADRES = 356, Reflected Power = 249.8 uW
retlw	0x7C	; Low Byte, ADRES = 360, Reflected Power = 273.8 uW
retlw	0xEB	; Low Byte, ADRES = 364, Reflected Power = 300.1 uW
retlw	0x64	; Low Byte, ADRES = 368, Reflected Power = 328.9 uW
retlw	0xE8	; Low Byte, ADRES = 372, Reflected Power = 360.6 uW
retlw	0x7A	; Low Byte, ADRES = 376, Reflected Power = 395.2 uW
retlw	0x19	; Low Byte, ADRES = 380, Reflected Power = 433.2 uW
retlw	0xC8	; Low Byte, ADRES = 384, Reflected Power = 474.9 uW
retlw	0x44	; Low Byte, ADRES = 388, Reflected Power = 520.5 uW
retlw	0xAD	; Low Byte, ADRES = 392, Reflected Power = 570.5 uW

retlw	0x20	; Low Byte, ADRES = 396, Reflected Power = 625.4 uW
retlw	0x9E	; Low Byte, ADRES = 400, Reflected Power = 685.5 uW
retlw	0x28	; Low Byte, ADRES = 404, Reflected Power = 751.4 uW
retlw	0xBF	; Low Byte, ADRES = 408, Reflected Power = 823.6 uW
retlw	0x65	; Low Byte, ADRES = 412, Reflected Power = 902.8 uW
retlw	0x0E	; Low Byte, ADRES = 416, Reflected Power = 989.6 uW
retlw	0x71	; Low Byte, ADRES = 420, Reflected Power = 1.085 mW
retlw	0xDF	; Low Byte, ADRES = 424, Reflected Power = 1.189 mW
retlw	0x57	; Low Byte, ADRES = 428, Reflected Power = 1.303 mW
retlw	0xDA	; Low Byte, ADRES = 432, Reflected Power = 1.428 mW
retlw	0x6A	; Low Byte, ADRES = 436, Reflected Power = 1.566 mW
retlw	0x08	; Low Byte, ADRES = 440, Reflected Power = 1.716 mW
retlw	0xB5	; Low Byte, ADRES = 444, Reflected Power = 1.881 mW
retlw	0x39	; Low Byte, ADRES = 448, Reflected Power = 2.062 mW
retlw	0xA1	; Low Byte, ADRES = 452, Reflected Power = 2.260 mW
retlw	0x13	; Low Byte, ADRES = 456, Reflected Power = 2.478 mW
retlw	0x90	; Low Byte, ADRES = 460, Reflected Power = 2.716 mW
retlw	0x19	; Low Byte, ADRES = 464, Reflected Power = 2.977 mW
retlw	0xAF	; Low Byte, ADRES = 468, Reflected Power = 3.263 mW
retlw	0x53	; Low Byte, ADRES = 472, Reflected Power = 3.577 mW
retlw	0x04	; Low Byte, ADRES = 476, Reflected Power = 3.920 mW
retlw	0x67	; Low Byte, ADRES = 480, Reflected Power = 4.297 mW
retlw	0xD3	; Low Byte, ADRES = 484, Reflected Power = 4.710 mW
retlw	0x4A	; Low Byte, ADRES = 488, Reflected Power = 5.163 mW
retlw	0xCC	; Low Byte, ADRES = 492, Reflected Power = 5.660 mW
retlw	0x5A	; Low Byte, ADRES = 496, Reflected Power = 6.204 mW
retlw	0xF7	; Low Byte, ADRES = 500, Reflected Power = 6.800 mW
retlw	0xA2	; Low Byte, ADRES = 504, Reflected Power = 7.453 mW
retlw	0x2F	; Low Byte, ADRES = 508, Reflected Power = 8.170 mW
retlw	0x96	; Low Byte, ADRES = 512, Reflected Power = 8.955 mW
retlw	0x07	; Low Byte, ADRES = 516, Reflected Power = 9.816 mW
retlw	0x82	; Low Byte, ADRES = 520, Reflected Power = 10.76 mW
retlw	0x0A	; Low Byte, ADRES = 524, Reflected Power = 11.79 mW
retlw	0x9E	; Low Byte, ADRES = 528, Reflected Power = 12.93 mW
retlw	0x41	; Low Byte, ADRES = 532, Reflected Power = 14.17 mW
retlw	0xF4	; Low Byte, ADRES = 536, Reflected Power = 15.53 mW
retlw	0x5C	; Low Byte, ADRES = 540, Reflected Power = 17.03 mW
retlw	0xC7	; Low Byte, ADRES = 544, Reflected Power = 18.66 mW
retlw	0x3D	; Low Byte, ADRES = 548, Reflected Power = 20.46 mW
retlw	0xBD	; Low Byte, ADRES = 552, Reflected Power = 22.42 mW
retlw	0x4B	; Low Byte, ADRES = 556, Reflected Power = 24.58 mW
retlw	0xE6	; Low Byte, ADRES = 560, Reflected Power = 26.94 mW
retlw	0x8F	; Low Byte, ADRES = 564, Reflected Power = 29.53 mW
retlw	0x25	; Low Byte, ADRES = 568, Reflected Power = 32.37 mW
retlw	0x8B	; Low Byte, ADRES = 572, Reflected Power = 35.48 mW
retlw	0xFA	; Low Byte, ADRES = 576, Reflected Power = 38.89 mW
retlw	0x75	; Low Byte, ADRES = 580, Reflected Power = 42.63 mW
retlw	0xFB	; Low Byte, ADRES = 584, Reflected Power = 46.73 mW
retlw	0x8E	; Low Byte, ADRES = 588, Reflected Power = 51.22 mW
retlw	0x30	; Low Byte, ADRES = 592, Reflected Power = 56.14 mW
retlw	0xE0	; Low Byte, ADRES = 596, Reflected Power = 61.54 mW
retlw	0x51	; Low Byte, ADRES = 600, Reflected Power = 67.45 mW
retlw	0xBB	; Low Byte, ADRES = 604, Reflected Power = 73.94 mW
retlw	0x30	; Low Byte, ADRES = 608, Reflected Power = 81.04 mW
retlw	0xAF	; Low Byte, ADRES = 612, Reflected Power = 88.83 mW
retlw	0x3B	; Low Byte, ADRES = 616, Reflected Power = 97.37 mW
retlw	0xD5	; Low Byte, ADRES = 620, Reflected Power = 106.7 mW
retlw	0x7D	; Low Byte, ADRES = 624, Reflected Power = 117.0 mW
retlw	0x1B	; Low Byte, ADRES = 628, Reflected Power = 128.2 mW
retlw	0x80	; Low Byte, ADRES = 632, Reflected Power = 140.6 mW
retlw	0xEE	; Low Byte, ADRES = 636, Reflected Power = 154.1 mW
retlw	0x68	; Low Byte, ADRES = 640, Reflected Power = 168.9 mW
retlw	0xED	; Low Byte, ADRES = 644, Reflected Power = 185.1 mW
retlw	0x7E	; Low Byte, ADRES = 648, Reflected Power = 202.9 mW
retlw	0x1E	; Low Byte, ADRES = 652, Reflected Power = 222.4 mW
retlw	0xCD	; Low Byte, ADRES = 656, Reflected Power = 243.8 mW
retlw	0x47	; Low Byte, ADRES = 660, Reflected Power = 267.2 mW
retlw	0xB0	; Low Byte, ADRES = 664, Reflected Power = 292.9 mW
retlw	0x23	; Low Byte, ADRES = 668, Reflected Power = 321.1 mW
retlw	0xA2	; Low Byte, ADRES = 672, Reflected Power = 351.9 mW
retlw	0x2C	; Low Byte, ADRES = 676, Reflected Power = 385.8 mW
retlw	0xC4	; Low Byte, ADRES = 680, Reflected Power = 422.9 mW
retlw	0x6B	; Low Byte, ADRES = 684, Reflected Power = 463.5 mW
retlw	0x11	; Low Byte, ADRES = 688, Reflected Power = 508.1 mW
retlw	0x75	; Low Byte, ADRES = 692, Reflected Power = 556.9 mW
retlw	0xE2	; Low Byte, ADRES = 696, Reflected Power = 610.4 mW
retlw	0x5A	; Low Byte, ADRES = 700, Reflected Power = 669.1 mW
retlw	0xDE	; Low Byte, ADRES = 704, Reflected Power = 733.4 mW
retlw	0x6E	; Low Byte, ADRES = 708, Reflected Power = 803.9 mW
retlw	0x0D	; Low Byte, ADRES = 712, Reflected Power = 881.2 mW
retlw	0xBA	; Low Byte, ADRES = 716, Reflected Power = 965.9 mW

retlw	0x3C	; Low Byte, ADRES = 720, Reflected Power = 1.059 W
retlw	0xA4	; Low Byte, ADRES = 724, Reflected Power = 1.161 W
retlw	0x17	; Low Byte, ADRES = 728, Reflected Power = 1.272 W
retlw	0x94	; Low Byte, ADRES = 732, Reflected Power = 1.394 W
retlw	0x1D	; Low Byte, ADRES = 736, Reflected Power = 1.528 W
retlw	0xB4	; Low Byte, ADRES = 740, Reflected Power = 1.675 W
retlw	0x58	; Low Byte, ADRES = 744, Reflected Power = 1.836 W
retlw	0x07	; Low Byte, ADRES = 748, Reflected Power = 2.013 W
retlw	0x6A	; Low Byte, ADRES = 752, Reflected Power = 2.206 W
retlw	0xD6	; Low Byte, ADRES = 756, Reflected Power = 2.418 W
retlw	0x4D	; Low Byte, ADRES = 760, Reflected Power = 2.651 W
retlw	0xD0	; Low Byte, ADRES = 764, Reflected Power = 2.906 W
retlw	0x5F	; Low Byte, ADRES = 768, Reflected Power = 3.185 W
retlw	0xFC	; Low Byte, ADRES = 772, Reflected Power = 3.491 W
retlw	0xA7	; Low Byte, ADRES = 776, Reflected Power = 3.827 W
retlw	0x32	; Low Byte, ADRES = 780, Reflected Power = 4.195 W
retlw	0x99	; Low Byte, ADRES = 784, Reflected Power = 4.598 W
retlw	0x0A	; Low Byte, ADRES = 788, Reflected Power = 5.040 W
retlw	0x86	; Low Byte, ADRES = 792, Reflected Power = 5.524 W
retlw	0x0E	; Low Byte, ADRES = 796, Reflected Power = 6.055 W
retlw	0xA3	; Low Byte, ADRES = 800, Reflected Power = 6.637 W
retlw	0x47	; Low Byte, ADRES = 804, Reflected Power = 7.275 W
retlw	0xFA	; Low Byte, ADRES = 808, Reflected Power = 7.975 W
retlw	0x5F	; Low Byte, ADRES = 812, Reflected Power = 8.741 W
retlw	0xCA	; Low Byte, ADRES = 816, Reflected Power = 9.582 W
retlw	0x40	; Low Byte, ADRES = 820, Reflected Power = 10.50 W
retlw	0xC2	; Low Byte, ADRES = 824, Reflected Power = 11.51 W
retlw	0x4F	; Low Byte, ADRES = 828, Reflected Power = 12.62 W
retlw	0xEA	; Low Byte, ADRES = 832, Reflected Power = 13.83 W
retlw	0x95	; Low Byte, ADRES = 836, Reflected Power = 15.16 W
retlw	0x28	; Low Byte, ADRES = 840, Reflected Power = 16.62 W
retlw	0x8E	; Low Byte, ADRES = 844, Reflected Power = 18.22 W
retlw	0xFE	; Low Byte, ADRES = 848, Reflected Power = 19.97 W
retlw	0x79	; Low Byte, ADRES = 852, Reflected Power = 21.89 W
retlw	0xFF	; Low Byte, ADRES = 856, Reflected Power = 23.99 W
retlw	0x93	; Low Byte, ADRES = 860, Reflected Power = 26.30 W
retlw	0x35	; Low Byte, ADRES = 864, Reflected Power = 28.82 W
retlw	0xE6	; Low Byte, ADRES = 868, Reflected Power = 31.60 W
retlw	0x54	; Low Byte, ADRES = 872, Reflected Power = 34.63 W
retlw	0xBF	; Low Byte, ADRES = 876, Reflected Power = 37.96 W
retlw	0x34	; Low Byte, ADRES = 880, Reflected Power = 41.61 W
retlw	0xB4	; Low Byte, ADRES = 884, Reflected Power = 45.61 W
retlw	0x40	; Low Byte, ADRES = 888, Reflected Power = 49.99 W
retlw	0xDA	; Low Byte, ADRES = 892, Reflected Power = 54.80 W
retlw	0x82	; Low Byte, ADRES = 896, Reflected Power = 60.07 W
retlw	0x1D	; Low Byte, ADRES = 900, Reflected Power = 65.84 W
retlw	0x83	; Low Byte, ADRES = 904, Reflected Power = 72.17 W
retlw	0xF2	; Low Byte, ADRES = 908, Reflected Power = 79.11 W
retlw	0x6B	; Low Byte, ADRES = 912, Reflected Power = 86.71 W
retlw	0xF1	; Low Byte, ADRES = 916, Reflected Power = 95.05 W
retlw	0x83	; Low Byte, ADRES = 920, Reflected Power = 104.2 W
retlw	0x23	; Low Byte, ADRES = 924, Reflected Power = 114.2 W
retlw	0xD3	; Low Byte, ADRES = 928, Reflected Power = 125.2 W
retlw	0x4A	; Low Byte, ADRES = 932, Reflected Power = 137.2 W
retlw	0xB3	; Low Byte, ADRES = 936, Reflected Power = 150.4 W
retlw	0x27	; Low Byte, ADRES = 940, Reflected Power = 164.9 W
retlw	0xA6	; Low Byte, ADRES = 944, Reflected Power = 180.7 W
retlw	0x31	; Low Byte, ADRES = 948, Reflected Power = 198.1 W
retlw	0xC9	; Low Byte, ADRES = 952, Reflected Power = 217.1 W
retlw	0x70	; Low Byte, ADRES = 956, Reflected Power = 238.0 W
retlw	0x13	; Low Byte, ADRES = 960, Reflected Power = 260.9 W
retlw	0x78	; Low Byte, ADRES = 964, Reflected Power = 285.9 W
retlw	0xE6	; Low Byte, ADRES = 968, Reflected Power = 313.4 W
retlw	0x5E	; Low Byte, ADRES = 972, Reflected Power = 343.5 W
retlw	0xE2	; Low Byte, ADRES = 976, Reflected Power = 376.6 W
retlw	0x73	; Low Byte, ADRES = 980, Reflected Power = 412.8 W
retlw	0x12	; Low Byte, ADRES = 984, Reflected Power = 452.4 W
retlw	0xC0	; Low Byte, ADRES = 988, Reflected Power = 495.9 W
retlw	0x3F	; Low Byte, ADRES = 992, Reflected Power = 543.6 W
retlw	0xA8	; Low Byte, ADRES = 996, Reflected Power = 595.9 W
retlw	0x1A	; Low Byte, ADRES = 1000, Reflected Power = 653.1 W
retlw	0x98	; Low Byte, ADRES = 1004, Reflected Power = 715.9 W
retlw	0x21	; Low Byte, ADRES = 1008, Reflected Power = 784.7 W
retlw	0xB8	; Low Byte, ADRES = 1012, Reflected Power = 860.2 W
retlw	0x5E	; Low Byte, ADRES = 1016, Reflected Power = 942.8 W
retlw	0x09	; Low Byte, ADRES = 1020, Reflected Power = 1,033.5 W

RLowByteTable1

retlw	0xDC	; Low Byte, ADRES = 1, Reflected Power = 72.44 nW
retlw	0x54	; Low Byte, ADRES = 5, Reflected Power = 79.40 nW
retlw	0xD7	; Low Byte, ADRES = 9, Reflected Power = 87.03 nW

retlw	0x67	; Low Byte, ADRES = 13, Reflected Power = 95.40 nW
retlw	0x05	; Low Byte, ADRES = 17, Reflected Power = 104.6 nW
retlw	0xB1	; Low Byte, ADRES = 21, Reflected Power = 114.6 nW
retlw	0x37	; Low Byte, ADRES = 25, Reflected Power = 125.6 nW
retlw	0x9F	; Low Byte, ADRES = 29, Reflected Power = 137.7 nW
retlw	0x11	; Low Byte, ADRES = 33, Reflected Power = 151.0 nW
retlw	0x8D	; Low Byte, ADRES = 37, Reflected Power = 165.5 nW
retlw	0x16	; Low Byte, ADRES = 41, Reflected Power = 181.4 nW
retlw	0xAC	; Low Byte, ADRES = 45, Reflected Power = 198.8 nW
retlw	0x50	; Low Byte, ADRES = 49, Reflected Power = 217.9 nW
retlw	0x02	; Low Byte, ADRES = 53, Reflected Power = 238.9 nW
retlw	0x65	; Low Byte, ADRES = 57, Reflected Power = 261.8 nW
retlw	0xD1	; Low Byte, ADRES = 61, Reflected Power = 287.0 nW
retlw	0x47	; Low Byte, ADRES = 65, Reflected Power = 314.6 nW
retlw	0xC9	; Low Byte, ADRES = 69, Reflected Power = 344.8 nW
retlw	0x57	; Low Byte, ADRES = 73, Reflected Power = 378.0 nW
retlw	0xF3	; Low Byte, ADRES = 77, Reflected Power = 414.3 nW
retlw	0x9E	; Low Byte, ADRES = 81, Reflected Power = 454.1 nW
retlw	0x2D	; Low Byte, ADRES = 85, Reflected Power = 497.8 nW
retlw	0x94	; Low Byte, ADRES = 89, Reflected Power = 545.6 nW
retlw	0x04	; Low Byte, ADRES = 93, Reflected Power = 598.1 nW
retlw	0x80	; Low Byte, ADRES = 97, Reflected Power = 655.5 nW
retlw	0x07	; Low Byte, ADRES = 101, Reflected Power = 718.6 nW
retlw	0x9B	; Low Byte, ADRES = 105, Reflected Power = 787.6 nW
retlw	0x3E	; Low Byte, ADRES = 109, Reflected Power = 863.3 nW
retlw	0xF0	; Low Byte, ADRES = 113, Reflected Power = 946.3 nW
retlw	0x5A	; Low Byte, ADRES = 117, Reflected Power = 1.037 uW
retlw	0xC5	; Low Byte, ADRES = 121, Reflected Power = 1.137 uW
retlw	0x3A	; Low Byte, ADRES = 125, Reflected Power = 1.246 uW
retlw	0xBB	; Low Byte, ADRES = 129, Reflected Power = 1.366 uW
retlw	0x48	; Low Byte, ADRES = 133, Reflected Power = 1.497 uW
retlw	0xE2	; Low Byte, ADRES = 137, Reflected Power = 1.641 uW
retlw	0x8C	; Low Byte, ADRES = 141, Reflected Power = 1.799 uW
retlw	0x23	; Low Byte, ADRES = 145, Reflected Power = 1.972 uW
retlw	0x89	; Low Byte, ADRES = 149, Reflected Power = 2.162 uW
retlw	0xF8	; Low Byte, ADRES = 153, Reflected Power = 2.369 uW
retlw	0x72	; Low Byte, ADRES = 157, Reflected Power = 2.597 uW
retlw	0xF8	; Low Byte, ADRES = 161, Reflected Power = 2.847 uW
retlw	0x8B	; Low Byte, ADRES = 165, Reflected Power = 3.121 uW
retlw	0x2C	; Low Byte, ADRES = 169, Reflected Power = 3.420 uW
retlw	0xDD	; Low Byte, ADRES = 173, Reflected Power = 3.749 uW
retlw	0x4F	; Low Byte, ADRES = 177, Reflected Power = 4.110 uW
retlw	0xB9	; Low Byte, ADRES = 181, Reflected Power = 4.505 uW
retlw	0x2D	; Low Byte, ADRES = 185, Reflected Power = 4.938 uW
retlw	0xAD	; Low Byte, ADRES = 189, Reflected Power = 5.412 uW
retlw	0x39	; Low Byte, ADRES = 193, Reflected Power = 5.933 uW
retlw	0xD2	; Low Byte, ADRES = 197, Reflected Power = 6.503 uW
retlw	0x79	; Low Byte, ADRES = 201, Reflected Power = 7.128 uW
retlw	0x19	; Low Byte, ADRES = 205, Reflected Power = 7.813 uW
retlw	0x7D	; Low Byte, ADRES = 209, Reflected Power = 8.564 uW
retlw	0xEC	; Low Byte, ADRES = 213, Reflected Power = 9.387 uW
retlw	0x65	; Low Byte, ADRES = 217, Reflected Power = 10.29 uW
retlw	0xEA	; Low Byte, ADRES = 221, Reflected Power = 11.28 uW
retlw	0x7B	; Low Byte, ADRES = 225, Reflected Power = 12.36 uW
retlw	0x1B	; Low Byte, ADRES = 229, Reflected Power = 13.55 uW
retlw	0xCA	; Low Byte, ADRES = 233, Reflected Power = 14.85 uW
retlw	0x45	; Low Byte, ADRES = 237, Reflected Power = 16.28 uW
retlw	0xAE	; Low Byte, ADRES = 241, Reflected Power = 17.85 uW
retlw	0x21	; Low Byte, ADRES = 245, Reflected Power = 19.56 uW
retlw	0x9F	; Low Byte, ADRES = 249, Reflected Power = 21.44 uW
retlw	0x29	; Low Byte, ADRES = 253, Reflected Power = 23.50 uW
retlw	0xC1	; Low Byte, ADRES = 257, Reflected Power = 25.76 uW
retlw	0x67	; Low Byte, ADRES = 261, Reflected Power = 28.24 uW
retlw	0x0F	; Low Byte, ADRES = 265, Reflected Power = 30.95 uW
retlw	0x73	; Low Byte, ADRES = 269, Reflected Power = 33.93 uW
retlw	0xE0	; Low Byte, ADRES = 273, Reflected Power = 37.19 uW
retlw	0x58	; Low Byte, ADRES = 277, Reflected Power = 40.77 uW
retlw	0xDB	; Low Byte, ADRES = 281, Reflected Power = 44.69 uW
retlw	0x6C	; Low Byte, ADRES = 285, Reflected Power = 48.98 uW
retlw	0x0A	; Low Byte, ADRES = 289, Reflected Power = 53.69 uW
retlw	0xB7	; Low Byte, ADRES = 293, Reflected Power = 58.85 uW
retlw	0x3A	; Low Byte, ADRES = 297, Reflected Power = 64.51 uW
retlw	0xA2	; Low Byte, ADRES = 301, Reflected Power = 70.71 uW
retlw	0x14	; Low Byte, ADRES = 305, Reflected Power = 77.50 uW
retlw	0x91	; Low Byte, ADRES = 309, Reflected Power = 84.95 uW
retlw	0x1A	; Low Byte, ADRES = 313, Reflected Power = 93.12 uW
retlw	0xB0	; Low Byte, ADRES = 317, Reflected Power = 102.1 uW
retlw	0x55	; Low Byte, ADRES = 321, Reflected Power = 111.9 uW
retlw	0x05	; Low Byte, ADRES = 325, Reflected Power = 122.6 uW
retlw	0x68	; Low Byte, ADRES = 329, Reflected Power = 134.4 uW
retlw	0xD4	; Low Byte, ADRES = 333, Reflected Power = 147.3 uW

retlw	0x4B	; Low Byte, ADRES = 337, Reflected Power = 161.5 uW
retlw	0xCD	; Low Byte, ADRES = 341, Reflected Power = 177.0 uW
retlw	0x5C	; Low Byte, ADRES = 345, Reflected Power = 194.1 uW
retlw	0xF8	; Low Byte, ADRES = 349, Reflected Power = 212.7 uW
retlw	0xA4	; Low Byte, ADRES = 353, Reflected Power = 233.2 uW
retlw	0x30	; Low Byte, ADRES = 357, Reflected Power = 255.6 uW
retlw	0x97	; Low Byte, ADRES = 361, Reflected Power = 280.1 uW
retlw	0x08	; Low Byte, ADRES = 365, Reflected Power = 307.1 uW
retlw	0x84	; Low Byte, ADRES = 369, Reflected Power = 336.6 uW
retlw	0x0B	; Low Byte, ADRES = 373, Reflected Power = 368.9 uW
retlw	0xA0	; Low Byte, ADRES = 377, Reflected Power = 404.4 uW
retlw	0x43	; Low Byte, ADRES = 381, Reflected Power = 443.3 uW
retlw	0xF6	; Low Byte, ADRES = 385, Reflected Power = 485.9 uW
retlw	0x5D	; Low Byte, ADRES = 389, Reflected Power = 532.6 uW
retlw	0xC8	; Low Byte, ADRES = 393, Reflected Power = 583.8 uW
retlw	0x3E	; Low Byte, ADRES = 397, Reflected Power = 639.9 uW
retlw	0xBF	; Low Byte, ADRES = 401, Reflected Power = 701.4 uW
retlw	0x4C	; Low Byte, ADRES = 405, Reflected Power = 768.8 uW
retlw	0xE7	; Low Byte, ADRES = 409, Reflected Power = 842.7 uW
retlw	0x91	; Low Byte, ADRES = 413, Reflected Power = 923.7 uW
retlw	0x26	; Low Byte, ADRES = 417, Reflected Power = 1.013 mW
retlw	0x8C	; Low Byte, ADRES = 421, Reflected Power = 1.110 mW
retlw	0xFC	; Low Byte, ADRES = 425, Reflected Power = 1.217 mW
retlw	0x76	; Low Byte, ADRES = 429, Reflected Power = 1.333 mW
retlw	0xFD	; Low Byte, ADRES = 433, Reflected Power = 1.462 mW
retlw	0x90	; Low Byte, ADRES = 437, Reflected Power = 1.602 mW
retlw	0x31	; Low Byte, ADRES = 441, Reflected Power = 1.756 mW
retlw	0xE2	; Low Byte, ADRES = 445, Reflected Power = 1.925 mW
retlw	0x52	; Low Byte, ADRES = 449, Reflected Power = 2.110 mW
retlw	0xBD	; Low Byte, ADRES = 453, Reflected Power = 2.313 mW
retlw	0x31	; Low Byte, ADRES = 457, Reflected Power = 2.535 mW
retlw	0xB1	; Low Byte, ADRES = 461, Reflected Power = 2.779 mW
retlw	0x3D	; Low Byte, ADRES = 465, Reflected Power = 3.046 mW
retlw	0xD6	; Low Byte, ADRES = 469, Reflected Power = 3.339 mW
retlw	0x7F	; Low Byte, ADRES = 473, Reflected Power = 3.660 mW
retlw	0x1C	; Low Byte, ADRES = 477, Reflected Power = 4.011 mW
retlw	0x81	; Low Byte, ADRES = 481, Reflected Power = 4.397 mW
retlw	0xEF	; Low Byte, ADRES = 485, Reflected Power = 4.820 mW
retlw	0x69	; Low Byte, ADRES = 489, Reflected Power = 5.283 mW
retlw	0xEE	; Low Byte, ADRES = 493, Reflected Power = 5.791 mW
retlw	0x80	; Low Byte, ADRES = 497, Reflected Power = 6.348 mW
retlw	0x20	; Low Byte, ADRES = 501, Reflected Power = 6.958 mW
retlw	0xCF	; Low Byte, ADRES = 505, Reflected Power = 7.626 mW
retlw	0x48	; Low Byte, ADRES = 509, Reflected Power = 8.360 mW
retlw	0xB1	; Low Byte, ADRES = 513, Reflected Power = 9.163 mW
retlw	0x24	; Low Byte, ADRES = 517, Reflected Power = 10.04 mW
retlw	0xA3	; Low Byte, ADRES = 521, Reflected Power = 11.01 mW
retlw	0x2E	; Low Byte, ADRES = 525, Reflected Power = 12.07 mW
retlw	0xC6	; Low Byte, ADRES = 529, Reflected Power = 13.23 mW
retlw	0x6C	; Low Byte, ADRES = 533, Reflected Power = 14.50 mW
retlw	0x12	; Low Byte, ADRES = 537, Reflected Power = 15.89 mW
retlw	0x76	; Low Byte, ADRES = 541, Reflected Power = 17.42 mW
retlw	0xE3	; Low Byte, ADRES = 545, Reflected Power = 19.10 mW
retlw	0x5C	; Low Byte, ADRES = 549, Reflected Power = 20.93 mW
retlw	0xE0	; Low Byte, ADRES = 553, Reflected Power = 22.94 mW
retlw	0x70	; Low Byte, ADRES = 557, Reflected Power = 25.15 mW
retlw	0x0F	; Low Byte, ADRES = 561, Reflected Power = 27.57 mW
retlw	0xBC	; Low Byte, ADRES = 565, Reflected Power = 30.22 mW
retlw	0x3D	; Low Byte, ADRES = 569, Reflected Power = 33.12 mW
retlw	0xA6	; Low Byte, ADRES = 573, Reflected Power = 36.30 mW
retlw	0x18	; Low Byte, ADRES = 577, Reflected Power = 39.79 mW
retlw	0x95	; Low Byte, ADRES = 581, Reflected Power = 43.62 mW
retlw	0x1F	; Low Byte, ADRES = 585, Reflected Power = 47.81 mW
retlw	0xB5	; Low Byte, ADRES = 589, Reflected Power = 52.41 mW
retlw	0x5A	; Low Byte, ADRES = 593, Reflected Power = 57.44 mW
retlw	0x08	; Low Byte, ADRES = 597, Reflected Power = 62.97 mW
retlw	0x6B	; Low Byte, ADRES = 601, Reflected Power = 69.02 mW
retlw	0xD8	; Low Byte, ADRES = 605, Reflected Power = 75.65 mW
retlw	0x4F	; Low Byte, ADRES = 609, Reflected Power = 82.92 mW
retlw	0xD1	; Low Byte, ADRES = 613, Reflected Power = 90.90 mW
retlw	0x60	; Low Byte, ADRES = 617, Reflected Power = 99.63 mW
retlw	0xFD	; Low Byte, ADRES = 621, Reflected Power = 109.2 mW
retlw	0xA9	; Low Byte, ADRES = 625, Reflected Power = 119.7 mW
retlw	0x33	; Low Byte, ADRES = 629, Reflected Power = 131.2 mW
retlw	0x9A	; Low Byte, ADRES = 633, Reflected Power = 143.8 mW
retlw	0x0C	; Low Byte, ADRES = 637, Reflected Power = 157.7 mW
retlw	0x88	; Low Byte, ADRES = 641, Reflected Power = 172.8 mW
retlw	0x10	; Low Byte, ADRES = 645, Reflected Power = 189.4 mW
retlw	0xA5	; Low Byte, ADRES = 649, Reflected Power = 207.6 mW
retlw	0x48	; Low Byte, ADRES = 653, Reflected Power = 227.6 mW
retlw	0xFC	; Low Byte, ADRES = 657, Reflected Power = 249.5 mW

retlw	0x60	; Low Byte, ADRES = 661, Reflected Power = 273.4 mW
retlw	0xCC	; Low Byte, ADRES = 665, Reflected Power = 299.7 mW
retlw	0x42	; Low Byte, ADRES = 669, Reflected Power = 328.5 mW
retlw	0xC3	; Low Byte, ADRES = 673, Reflected Power = 360.1 mW
retlw	0x51	; Low Byte, ADRES = 677, Reflected Power = 394.7 mW
retlw	0xEC	; Low Byte, ADRES = 681, Reflected Power = 432.7 mW
retlw	0x97	; Low Byte, ADRES = 685, Reflected Power = 474.3 mW
retlw	0x29	; Low Byte, ADRES = 689, Reflected Power = 519.9 mW
retlw	0x8F	; Low Byte, ADRES = 693, Reflected Power = 569.8 mW
retlw	0xFF	; Low Byte, ADRES = 697, Reflected Power = 624.6 mW
retlw	0x7A	; Low Byte, ADRES = 701, Reflected Power = 684.6 mW
retlw	0x01	; Low Byte, ADRES = 705, Reflected Power = 750.5 mW
retlw	0x95	; Low Byte, ADRES = 709, Reflected Power = 822.6 mW
retlw	0x37	; Low Byte, ADRES = 713, Reflected Power = 901.7 mW
retlw	0xE8	; Low Byte, ADRES = 717, Reflected Power = 988.3 mW
retlw	0x55	; Low Byte, ADRES = 721, Reflected Power = 1.083 W
retlw	0xC0	; Low Byte, ADRES = 725, Reflected Power = 1.187 W
retlw	0x35	; Low Byte, ADRES = 729, Reflected Power = 1.302 W
retlw	0xB5	; Low Byte, ADRES = 733, Reflected Power = 1.427 W
retlw	0x41	; Low Byte, ADRES = 737, Reflected Power = 1.564 W
retlw	0xDB	; Low Byte, ADRES = 741, Reflected Power = 1.714 W
retlw	0x84	; Low Byte, ADRES = 745, Reflected Power = 1.879 W
retlw	0x1F	; Low Byte, ADRES = 749, Reflected Power = 2.060 W
retlw	0x84	; Low Byte, ADRES = 753, Reflected Power = 2.258 W
retlw	0xF3	; Low Byte, ADRES = 757, Reflected Power = 2.475 W
retlw	0x6D	; Low Byte, ADRES = 761, Reflected Power = 2.712 W
retlw	0xF2	; Low Byte, ADRES = 765, Reflected Power = 2.973 W
retlw	0x85	; Low Byte, ADRES = 769, Reflected Power = 3.259 W
retlw	0x25	; Low Byte, ADRES = 773, Reflected Power = 3.572 W
retlw	0xD5	; Low Byte, ADRES = 777, Reflected Power = 3.916 W
retlw	0x4B	; Low Byte, ADRES = 781, Reflected Power = 4.292 W
retlw	0xB4	; Low Byte, ADRES = 785, Reflected Power = 4.705 W
retlw	0x28	; Low Byte, ADRES = 789, Reflected Power = 5.157 W
retlw	0xA7	; Low Byte, ADRES = 793, Reflected Power = 5.653 W
retlw	0x32	; Low Byte, ADRES = 797, Reflected Power = 6.196 W
retlw	0xCB	; Low Byte, ADRES = 801, Reflected Power = 6.791 W
retlw	0x72	; Low Byte, ADRES = 805, Reflected Power = 7.444 W
retlw	0x14	; Low Byte, ADRES = 809, Reflected Power = 8.160 W
retlw	0x79	; Low Byte, ADRES = 813, Reflected Power = 8.944 W
retlw	0xE7	; Low Byte, ADRES = 817, Reflected Power = 9.804 W
retlw	0x60	; Low Byte, ADRES = 821, Reflected Power = 10.75 W
retlw	0xE4	; Low Byte, ADRES = 825, Reflected Power = 11.78 W
retlw	0x75	; Low Byte, ADRES = 829, Reflected Power = 12.91 W
retlw	0x14	; Low Byte, ADRES = 833, Reflected Power = 14.15 W
retlw	0xC2	; Low Byte, ADRES = 837, Reflected Power = 15.51 W
retlw	0x40	; Low Byte, ADRES = 841, Reflected Power = 17.00 W
retlw	0xA9	; Low Byte, ADRES = 845, Reflected Power = 18.64 W
retlw	0x1C	; Low Byte, ADRES = 849, Reflected Power = 20.43 W
retlw	0x99	; Low Byte, ADRES = 853, Reflected Power = 22.39 W
retlw	0x23	; Low Byte, ADRES = 857, Reflected Power = 24.55 W
retlw	0xBA	; Low Byte, ADRES = 861, Reflected Power = 26.91 W
retlw	0x60	; Low Byte, ADRES = 865, Reflected Power = 29.49 W
retlw	0x0B	; Low Byte, ADRES = 869, Reflected Power = 32.33 W
retlw	0x6E	; Low Byte, ADRES = 873, Reflected Power = 35.44 W
retlw	0xDB	; Low Byte, ADRES = 877, Reflected Power = 38.84 W
retlw	0x52	; Low Byte, ADRES = 881, Reflected Power = 42.58 W
retlw	0xD5	; Low Byte, ADRES = 885, Reflected Power = 46.67 W
retlw	0x65	; Low Byte, ADRES = 889, Reflected Power = 51.15 W
retlw	0x02	; Low Byte, ADRES = 893, Reflected Power = 56.07 W
retlw	0xAF	; Low Byte, ADRES = 897, Reflected Power = 61.46 W
retlw	0x36	; Low Byte, ADRES = 901, Reflected Power = 67.37 W
retlw	0x9E	; Low Byte, ADRES = 905, Reflected Power = 73.85 W
retlw	0x0F	; Low Byte, ADRES = 909, Reflected Power = 80.94 W
retlw	0x8C	; Low Byte, ADRES = 913, Reflected Power = 88.72 W
retlw	0x14	; Low Byte, ADRES = 917, Reflected Power = 97.25 W
retlw	0xAA	; Low Byte, ADRES = 921, Reflected Power = 106.6 W
retlw	0x4E	; Low Byte, ADRES = 925, Reflected Power = 116.8 W
retlw	0x01	; Low Byte, ADRES = 929, Reflected Power = 128.1 W
retlw	0x63	; Low Byte, ADRES = 933, Reflected Power = 140.4 W
retlw	0xCF	; Low Byte, ADRES = 937, Reflected Power = 153.9 W
retlw	0x45	; Low Byte, ADRES = 941, Reflected Power = 168.7 W
retlw	0xC7	; Low Byte, ADRES = 945, Reflected Power = 184.9 W
retlw	0x55	; Low Byte, ADRES = 949, Reflected Power = 202.7 W
retlw	0xF1	; Low Byte, ADRES = 953, Reflected Power = 222.1 W
retlw	0x9C	; Low Byte, ADRES = 957, Reflected Power = 243.5 W
retlw	0x2C	; Low Byte, ADRES = 961, Reflected Power = 266.9 W
retlw	0x92	; Low Byte, ADRES = 965, Reflected Power = 292.6 W
retlw	0x03	; Low Byte, ADRES = 969, Reflected Power = 320.7 W
retlw	0x7E	; Low Byte, ADRES = 973, Reflected Power = 351.5 W
retlw	0x05	; Low Byte, ADRES = 977, Reflected Power = 385.3 W
retlw	0x99	; Low Byte, ADRES = 981, Reflected Power = 422.3 W

```

retlw    0x3C      ; Low Byte, ADRES = 985, Reflected Power = 462.9 W
retlw    0xEE      ; Low Byte, ADRES = 989, Reflected Power = 507.4 W
retlw    0x58      ; Low Byte, ADRES = 993, Reflected Power = 556.2 W
retlw    0xC3      ; Low Byte, ADRES = 997, Reflected Power = 609.7 W
retlw    0x39      ; Low Byte, ADRES = 1001, Reflected Power = 668.3 W
retlw    0xB9      ; Low Byte, ADRES = 1005, Reflected Power = 732.5 W
retlw    0x46      ; Low Byte, ADRES = 1009, Reflected Power = 802.9 W
retlw    0xE0      ; Low Byte, ADRES = 1013, Reflected Power = 880.1 W
retlw    0x89      ; Low Byte, ADRES = 1017, Reflected Power = 964.7 W
retlw    0x21      ; Low Byte, ADRES = 1021, Reflected Power = 1,057.5 W

```

RLowByteTable2

```

retlw    0xF9      ; Low Byte, ADRES = 2, Reflected Power = 74.12 nW
retlw    0x74      ; Low Byte, ADRES = 6, Reflected Power = 81.24 nW
retlw    0xFA      ; Low Byte, ADRES = 10, Reflected Power = 89.05 nW
retlw    0x8D      ; Low Byte, ADRES = 14, Reflected Power = 97.61 nW
retlw    0x2E      ; Low Byte, ADRES = 18, Reflected Power = 107.0 nW
retlw    0xDF      ; Low Byte, ADRES = 22, Reflected Power = 117.3 nW
retlw    0x50      ; Low Byte, ADRES = 26, Reflected Power = 128.6 nW
retlw    0xBA      ; Low Byte, ADRES = 30, Reflected Power = 140.9 nW
retlw    0x2F      ; Low Byte, ADRES = 34, Reflected Power = 154.5 nW
retlw    0xAE      ; Low Byte, ADRES = 38, Reflected Power = 169.3 nW
retlw    0x3A      ; Low Byte, ADRES = 42, Reflected Power = 185.6 nW
retlw    0xD3      ; Low Byte, ADRES = 46, Reflected Power = 203.4 nW
retlw    0x7B      ; Low Byte, ADRES = 50, Reflected Power = 223.0 nW
retlw    0x1A      ; Low Byte, ADRES = 54, Reflected Power = 244.4 nW
retlw    0x7F      ; Low Byte, ADRES = 58, Reflected Power = 267.9 nW
retlw    0xED      ; Low Byte, ADRES = 62, Reflected Power = 293.7 nW
retlw    0x66      ; Low Byte, ADRES = 66, Reflected Power = 321.9 nW
retlw    0xEB      ; Low Byte, ADRES = 70, Reflected Power = 352.8 nW
retlw    0x7D      ; Low Byte, ADRES = 74, Reflected Power = 386.7 nW
retlw    0x1D      ; Low Byte, ADRES = 78, Reflected Power = 423.9 nW
retlw    0xCC      ; Low Byte, ADRES = 82, Reflected Power = 464.7 nW
retlw    0x46      ; Low Byte, ADRES = 86, Reflected Power = 509.3 nW
retlw    0xAF      ; Low Byte, ADRES = 90, Reflected Power = 558.3 nW
retlw    0x22      ; Low Byte, ADRES = 94, Reflected Power = 611.9 nW
retlw    0xA0      ; Low Byte, ADRES = 98, Reflected Power = 670.8 nW
retlw    0x2B      ; Low Byte, ADRES = 102, Reflected Power = 735.2 nW
retlw    0xC3      ; Low Byte, ADRES = 106, Reflected Power = 805.9 nW
retlw    0x69      ; Low Byte, ADRES = 110, Reflected Power = 883.4 nW
retlw    0x10      ; Low Byte, ADRES = 114, Reflected Power = 968.3 nW
retlw    0x74      ; Low Byte, ADRES = 118, Reflected Power = 1.061 uW
retlw    0xE1      ; Low Byte, ADRES = 122, Reflected Power = 1.163 uW
retlw    0x59      ; Low Byte, ADRES = 126, Reflected Power = 1.275 uW
retlw    0xDD      ; Low Byte, ADRES = 130, Reflected Power = 1.398 uW
retlw    0x6D      ; Low Byte, ADRES = 134, Reflected Power = 1.532 uW
retlw    0x0B      ; Low Byte, ADRES = 138, Reflected Power = 1.679 uW
retlw    0xB9      ; Low Byte, ADRES = 142, Reflected Power = 1.841 uW
retlw    0x3B      ; Low Byte, ADRES = 146, Reflected Power = 2.018 uW
retlw    0xA3      ; Low Byte, ADRES = 150, Reflected Power = 2.212 uW
retlw    0x16      ; Low Byte, ADRES = 154, Reflected Power = 2.424 uW
retlw    0x93      ; Low Byte, ADRES = 158, Reflected Power = 2.657 uW
retlw    0x1C      ; Low Byte, ADRES = 162, Reflected Power = 2.913 uW
retlw    0xB2      ; Low Byte, ADRES = 166, Reflected Power = 3.193 uW
retlw    0x57      ; Low Byte, ADRES = 170, Reflected Power = 3.500 uW
retlw    0x06      ; Low Byte, ADRES = 174, Reflected Power = 3.836 uW
retlw    0x69      ; Low Byte, ADRES = 178, Reflected Power = 4.205 uW
retlw    0xD5      ; Low Byte, ADRES = 182, Reflected Power = 4.609 uW
retlw    0x4C      ; Low Byte, ADRES = 186, Reflected Power = 5.052 uW
retlw    0xCF      ; Low Byte, ADRES = 190, Reflected Power = 5.538 uW
retlw    0x5D      ; Low Byte, ADRES = 194, Reflected Power = 6.070 uW
retlw    0xFA      ; Low Byte, ADRES = 198, Reflected Power = 6.654 uW
retlw    0xA6      ; Low Byte, ADRES = 202, Reflected Power = 7.293 uW
retlw    0x31      ; Low Byte, ADRES = 206, Reflected Power = 7.994 uW
retlw    0x98      ; Low Byte, ADRES = 210, Reflected Power = 8.763 uW
retlw    0x09      ; Low Byte, ADRES = 214, Reflected Power = 9.605 uW
retlw    0x85      ; Low Byte, ADRES = 218, Reflected Power = 10.53 uW
retlw    0x0D      ; Low Byte, ADRES = 222, Reflected Power = 11.54 uW
retlw    0xA2      ; Low Byte, ADRES = 226, Reflected Power = 12.65 uW
retlw    0x45      ; Low Byte, ADRES = 230, Reflected Power = 13.87 uW
retlw    0xF8      ; Low Byte, ADRES = 234, Reflected Power = 15.20 uW
retlw    0x5E      ; Low Byte, ADRES = 238, Reflected Power = 16.66 uW
retlw    0xCA      ; Low Byte, ADRES = 242, Reflected Power = 18.26 uW
retlw    0x3F      ; Low Byte, ADRES = 246, Reflected Power = 20.02 uW
retlw    0xC0      ; Low Byte, ADRES = 250, Reflected Power = 21.94 uW
retlw    0x4E      ; Low Byte, ADRES = 254, Reflected Power = 24.05 uW
retlw    0xE9      ; Low Byte, ADRES = 258, Reflected Power = 26.36 uW
retlw    0x93      ; Low Byte, ADRES = 262, Reflected Power = 28.90 uW
retlw    0x27      ; Low Byte, ADRES = 266, Reflected Power = 31.67 uW
retlw    0x8D      ; Low Byte, ADRES = 270, Reflected Power = 34.72 uW
retlw    0xFD      ; Low Byte, ADRES = 274, Reflected Power = 38.05 uW

```

retlw	0x78	; Low Byte, ADRES = 278, Reflected Power = 41.71 uW
retlw	0xFE	; Low Byte, ADRES = 282, Reflected Power = 45.72 uW
retlw	0x92	; Low Byte, ADRES = 286, Reflected Power = 50.12 uW
retlw	0x33	; Low Byte, ADRES = 290, Reflected Power = 54.94 uW
retlw	0xE5	; Low Byte, ADRES = 294, Reflected Power = 60.22 uW
retlw	0x53	; Low Byte, ADRES = 298, Reflected Power = 66.00 uW
retlw	0xBE	; Low Byte, ADRES = 302, Reflected Power = 72.35 uW
retlw	0x33	; Low Byte, ADRES = 306, Reflected Power = 79.30 uW
retlw	0xB2	; Low Byte, ADRES = 310, Reflected Power = 86.93 uW
retlw	0x3F	; Low Byte, ADRES = 314, Reflected Power = 95.28 uW
retlw	0xD8	; Low Byte, ADRES = 318, Reflected Power = 104.4 uW
retlw	0x81	; Low Byte, ADRES = 322, Reflected Power = 114.5 uW
retlw	0x1D	; Low Byte, ADRES = 326, Reflected Power = 125.5 uW
retlw	0x82	; Low Byte, ADRES = 330, Reflected Power = 137.5 uW
retlw	0xF1	; Low Byte, ADRES = 334, Reflected Power = 150.8 uW
retlw	0x6A	; Low Byte, ADRES = 338, Reflected Power = 165.3 uW
retlw	0xF0	; Low Byte, ADRES = 342, Reflected Power = 181.1 uW
retlw	0x82	; Low Byte, ADRES = 346, Reflected Power = 198.6 uW
retlw	0x22	; Low Byte, ADRES = 350, Reflected Power = 217.6 uW
retlw	0xD1	; Low Byte, ADRES = 354, Reflected Power = 238.6 uW
retlw	0x49	; Low Byte, ADRES = 358, Reflected Power = 261.5 uW
retlw	0xB2	; Low Byte, ADRES = 362, Reflected Power = 286.6 uW
retlw	0x26	; Low Byte, ADRES = 366, Reflected Power = 314.2 uW
retlw	0xA4	; Low Byte, ADRES = 370, Reflected Power = 344.4 uW
retlw	0x2F	; Low Byte, ADRES = 374, Reflected Power = 377.5 uW
retlw	0xC8	; Low Byte, ADRES = 378, Reflected Power = 413.8 uW
retlw	0x6E	; Low Byte, ADRES = 382, Reflected Power = 453.6 uW
retlw	0x13	; Low Byte, ADRES = 386, Reflected Power = 497.2 uW
retlw	0x77	; Low Byte, ADRES = 390, Reflected Power = 544.9 uW
retlw	0xE5	; Low Byte, ADRES = 394, Reflected Power = 597.3 uW
retlw	0x5D	; Low Byte, ADRES = 398, Reflected Power = 654.7 uW
retlw	0xE1	; Low Byte, ADRES = 402, Reflected Power = 717.7 uW
retlw	0x72	; Low Byte, ADRES = 406, Reflected Power = 786.7 uW
retlw	0x10	; Low Byte, ADRES = 410, Reflected Power = 862.3 uW
retlw	0xBE	; Low Byte, ADRES = 414, Reflected Power = 945.2 uW
retlw	0x3E	; Low Byte, ADRES = 418, Reflected Power = 1.036 mW
retlw	0xA7	; Low Byte, ADRES = 422, Reflected Power = 1.136 mW
retlw	0x19	; Low Byte, ADRES = 426, Reflected Power = 1.245 mW
retlw	0x97	; Low Byte, ADRES = 430, Reflected Power = 1.364 mW
retlw	0x20	; Low Byte, ADRES = 434, Reflected Power = 1.496 mW
retlw	0xB7	; Low Byte, ADRES = 438, Reflected Power = 1.639 mW
retlw	0x5C	; Low Byte, ADRES = 442, Reflected Power = 1.797 mW
retlw	0x09	; Low Byte, ADRES = 446, Reflected Power = 1.970 mW
retlw	0x6C	; Low Byte, ADRES = 450, Reflected Power = 2.159 mW
retlw	0xD9	; Low Byte, ADRES = 454, Reflected Power = 2.367 mW
retlw	0x50	; Low Byte, ADRES = 458, Reflected Power = 2.594 mW
retlw	0xD3	; Low Byte, ADRES = 462, Reflected Power = 2.843 mW
retlw	0x62	; Low Byte, ADRES = 466, Reflected Power = 3.117 mW
retlw	0xFF	; Low Byte, ADRES = 470, Reflected Power = 3.416 mW
retlw	0xAB	; Low Byte, ADRES = 474, Reflected Power = 3.745 mW
retlw	0x34	; Low Byte, ADRES = 478, Reflected Power = 4.105 mW
retlw	0x9B	; Low Byte, ADRES = 482, Reflected Power = 4.499 mW
retlw	0x0D	; Low Byte, ADRES = 486, Reflected Power = 4.932 mW
retlw	0x89	; Low Byte, ADRES = 490, Reflected Power = 5.406 mW
retlw	0x11	; Low Byte, ADRES = 494, Reflected Power = 5.925 mW
retlw	0xA7	; Low Byte, ADRES = 498, Reflected Power = 6.495 mW
retlw	0x4A	; Low Byte, ADRES = 502, Reflected Power = 7.119 mW
retlw	0xFE	; Low Byte, ADRES = 506, Reflected Power = 7.804 mW
retlw	0x61	; Low Byte, ADRES = 510, Reflected Power = 8.554 mW
retlw	0xCD	; Low Byte, ADRES = 514, Reflected Power = 9.376 mW
retlw	0x43	; Low Byte, ADRES = 518, Reflected Power = 10.28 mW
retlw	0xC5	; Low Byte, ADRES = 522, Reflected Power = 11.26 mW
retlw	0x52	; Low Byte, ADRES = 526, Reflected Power = 12.35 mW
retlw	0xEE	; Low Byte, ADRES = 530, Reflected Power = 13.53 mW
retlw	0x99	; Low Byte, ADRES = 534, Reflected Power = 14.84 mW
retlw	0x2A	; Low Byte, ADRES = 538, Reflected Power = 16.26 mW
retlw	0x90	; Low Byte, ADRES = 542, Reflected Power = 17.83 mW
retlw	0x00	; Low Byte, ADRES = 546, Reflected Power = 19.54 mW
retlw	0x7C	; Low Byte, ADRES = 550, Reflected Power = 21.42 mW
retlw	0x02	; Low Byte, ADRES = 554, Reflected Power = 23.48 mW
retlw	0x96	; Low Byte, ADRES = 558, Reflected Power = 25.73 mW
retlw	0x38	; Low Byte, ADRES = 562, Reflected Power = 28.21 mW
retlw	0xEA	; Low Byte, ADRES = 566, Reflected Power = 30.92 mW
retlw	0x56	; Low Byte, ADRES = 570, Reflected Power = 33.89 mW
retlw	0xC1	; Low Byte, ADRES = 574, Reflected Power = 37.15 mW
retlw	0x36	; Low Byte, ADRES = 578, Reflected Power = 40.72 mW
retlw	0xB6	; Low Byte, ADRES = 582, Reflected Power = 44.63 mW
retlw	0x43	; Low Byte, ADRES = 586, Reflected Power = 48.92 mW
retlw	0xDD	; Low Byte, ADRES = 590, Reflected Power = 53.62 mW
retlw	0x86	; Low Byte, ADRES = 594, Reflected Power = 58.78 mW
retlw	0x20	; Low Byte, ADRES = 598, Reflected Power = 64.43 mW

retlw	0x85	; Low Byte, ADRES = 602, Reflected Power = 70.62 mW
retlw	0xF4	; Low Byte, ADRES = 606, Reflected Power = 77.41 mW
retlw	0x6E	; Low Byte, ADRES = 610, Reflected Power = 84.85 mW
retlw	0xF4	; Low Byte, ADRES = 614, Reflected Power = 93.01 mW
retlw	0x86	; Low Byte, ADRES = 618, Reflected Power = 101.9 mW
retlw	0x27	; Low Byte, ADRES = 622, Reflected Power = 111.7 mW
retlw	0xD7	; Low Byte, ADRES = 626, Reflected Power = 122.5 mW
retlw	0x4C	; Low Byte, ADRES = 630, Reflected Power = 134.3 mW
retlw	0xB6	; Low Byte, ADRES = 634, Reflected Power = 147.2 mW
retlw	0x29	; Low Byte, ADRES = 638, Reflected Power = 161.3 mW
retlw	0xA9	; Low Byte, ADRES = 642, Reflected Power = 176.8 mW
retlw	0x34	; Low Byte, ADRES = 646, Reflected Power = 193.8 mW
retlw	0xCC	; Low Byte, ADRES = 650, Reflected Power = 212.4 mW
retlw	0x74	; Low Byte, ADRES = 654, Reflected Power = 232.9 mW
retlw	0x16	; Low Byte, ADRES = 658, Reflected Power = 255.3 mW
retlw	0x7A	; Low Byte, ADRES = 662, Reflected Power = 279.8 mW
retlw	0xE8	; Low Byte, ADRES = 666, Reflected Power = 306.7 mW
retlw	0x61	; Low Byte, ADRES = 670, Reflected Power = 336.2 mW
retlw	0xE5	; Low Byte, ADRES = 674, Reflected Power = 368.5 mW
retlw	0x76	; Low Byte, ADRES = 678, Reflected Power = 403.9 mW
retlw	0x15	; Low Byte, ADRES = 682, Reflected Power = 442.7 mW
retlw	0xC4	; Low Byte, ADRES = 686, Reflected Power = 485.3 mW
retlw	0x41	; Low Byte, ADRES = 690, Reflected Power = 531.9 mW
retlw	0xAA	; Low Byte, ADRES = 694, Reflected Power = 583.1 mW
retlw	0x1D	; Low Byte, ADRES = 698, Reflected Power = 639.1 mW
retlw	0x9B	; Low Byte, ADRES = 702, Reflected Power = 700.5 mW
retlw	0x25	; Low Byte, ADRES = 706, Reflected Power = 767.9 mW
retlw	0xBC	; Low Byte, ADRES = 710, Reflected Power = 841.7 mW
retlw	0x61	; Low Byte, ADRES = 714, Reflected Power = 922.6 mW
retlw	0x0C	; Low Byte, ADRES = 718, Reflected Power = 1.011 W
retlw	0x6F	; Low Byte, ADRES = 722, Reflected Power = 1.108 W
retlw	0xDC	; Low Byte, ADRES = 726, Reflected Power = 1.215 W
retlw	0x54	; Low Byte, ADRES = 730, Reflected Power = 1.332 W
retlw	0xD7	; Low Byte, ADRES = 734, Reflected Power = 1.460 W
retlw	0x67	; Low Byte, ADRES = 738, Reflected Power = 1.600 W
retlw	0x04	; Low Byte, ADRES = 742, Reflected Power = 1.754 W
retlw	0xB1	; Low Byte, ADRES = 746, Reflected Power = 1.923 W
retlw	0x37	; Low Byte, ADRES = 750, Reflected Power = 2.107 W
retlw	0x9F	; Low Byte, ADRES = 754, Reflected Power = 2.310 W
retlw	0x10	; Low Byte, ADRES = 758, Reflected Power = 2.532 W
retlw	0x8D	; Low Byte, ADRES = 762, Reflected Power = 2.775 W
retlw	0x16	; Low Byte, ADRES = 766, Reflected Power = 3.042 W
retlw	0xAB	; Low Byte, ADRES = 770, Reflected Power = 3.335 W
retlw	0x4F	; Low Byte, ADRES = 774, Reflected Power = 3.655 W
retlw	0x02	; Low Byte, ADRES = 778, Reflected Power = 4.007 W
retlw	0x64	; Low Byte, ADRES = 782, Reflected Power = 4.392 W
retlw	0xD0	; Low Byte, ADRES = 786, Reflected Power = 4.814 W
retlw	0x47	; Low Byte, ADRES = 790, Reflected Power = 5.277 W
retlw	0xC9	; Low Byte, ADRES = 794, Reflected Power = 5.784 W
retlw	0x57	; Low Byte, ADRES = 798, Reflected Power = 6.340 W
retlw	0xF3	; Low Byte, ADRES = 802, Reflected Power = 6.949 W
retlw	0x9E	; Low Byte, ADRES = 806, Reflected Power = 7.617 W
retlw	0x2D	; Low Byte, ADRES = 810, Reflected Power = 8.349 W
retlw	0x93	; Low Byte, ADRES = 814, Reflected Power = 9.152 W
retlw	0x04	; Low Byte, ADRES = 818, Reflected Power = 10.03 W
retlw	0x7F	; Low Byte, ADRES = 822, Reflected Power = 11.00 W
retlw	0x07	; Low Byte, ADRES = 826, Reflected Power = 12.05 W
retlw	0x9B	; Low Byte, ADRES = 830, Reflected Power = 13.21 W
retlw	0x3E	; Low Byte, ADRES = 834, Reflected Power = 14.48 W
retlw	0xF0	; Low Byte, ADRES = 838, Reflected Power = 15.87 W
retlw	0x5A	; Low Byte, ADRES = 842, Reflected Power = 17.40 W
retlw	0xC5	; Low Byte, ADRES = 846, Reflected Power = 19.07 W
retlw	0x3A	; Low Byte, ADRES = 850, Reflected Power = 20.90 W
retlw	0xBB	; Low Byte, ADRES = 854, Reflected Power = 22.91 W
retlw	0x48	; Low Byte, ADRES = 858, Reflected Power = 25.12 W
retlw	0xE2	; Low Byte, ADRES = 862, Reflected Power = 27.53 W
retlw	0x8B	; Low Byte, ADRES = 866, Reflected Power = 30.18 W
retlw	0x23	; Low Byte, ADRES = 870, Reflected Power = 33.08 W
retlw	0x88	; Low Byte, ADRES = 874, Reflected Power = 36.26 W
retlw	0xF8	; Low Byte, ADRES = 878, Reflected Power = 39.74 W
retlw	0x72	; Low Byte, ADRES = 882, Reflected Power = 43.56 W
retlw	0xF8	; Low Byte, ADRES = 886, Reflected Power = 47.75 W
retlw	0x8B	; Low Byte, ADRES = 890, Reflected Power = 52.34 W
retlw	0x2C	; Low Byte, ADRES = 894, Reflected Power = 57.37 W
retlw	0xDC	; Low Byte, ADRES = 898, Reflected Power = 62.89 W
retlw	0x4F	; Low Byte, ADRES = 902, Reflected Power = 68.93 W
retlw	0xB9	; Low Byte, ADRES = 906, Reflected Power = 75.56 W
retlw	0x2D	; Low Byte, ADRES = 910, Reflected Power = 82.82 W
retlw	0xAD	; Low Byte, ADRES = 914, Reflected Power = 90.78 W
retlw	0x38	; Low Byte, ADRES = 918, Reflected Power = 99.51 W
retlw	0xD1	; Low Byte, ADRES = 922, Reflected Power = 109.1 W

retlw	0x79	; Low Byte, ADRES = 926, Reflected Power = 119.6 W
retlw	0x18	; Low Byte, ADRES = 930, Reflected Power = 131.1 W
retlw	0x7D	; Low Byte, ADRES = 934, Reflected Power = 143.7 W
retlw	0xEC	; Low Byte, ADRES = 938, Reflected Power = 157.5 W
retlw	0x65	; Low Byte, ADRES = 942, Reflected Power = 172.6 W
retlw	0xEA	; Low Byte, ADRES = 946, Reflected Power = 189.2 W
retlw	0x7B	; Low Byte, ADRES = 950, Reflected Power = 207.4 W
retlw	0x1A	; Low Byte, ADRES = 954, Reflected Power = 227.3 W
retlw	0xC9	; Low Byte, ADRES = 958, Reflected Power = 249.2 W
retlw	0x44	; Low Byte, ADRES = 962, Reflected Power = 273.1 W
retlw	0xAD	; Low Byte, ADRES = 966, Reflected Power = 299.4 W
retlw	0x21	; Low Byte, ADRES = 970, Reflected Power = 328.1 W
retlw	0x9F	; Low Byte, ADRES = 974, Reflected Power = 359.7 W
retlw	0x29	; Low Byte, ADRES = 978, Reflected Power = 394.2 W
retlw	0xC1	; Low Byte, ADRES = 982, Reflected Power = 432.1 W
retlw	0x67	; Low Byte, ADRES = 986, Reflected Power = 473.7 W
retlw	0x0E	; Low Byte, ADRES = 990, Reflected Power = 519.2 W
retlw	0x72	; Low Byte, ADRES = 994, Reflected Power = 569.1 W
retlw	0xE0	; Low Byte, ADRES = 998, Reflected Power = 623.8 W
retlw	0x58	; Low Byte, ADRES = 1002, Reflected Power = 683.8 W
retlw	0xDB	; Low Byte, ADRES = 1006, Reflected Power = 749.5 W
retlw	0x6B	; Low Byte, ADRES = 1010, Reflected Power = 821.6 W
retlw	0x09	; Low Byte, ADRES = 1014, Reflected Power = 900.6 W
retlw	0xB6	; Low Byte, ADRES = 1018, Reflected Power = 987.1 W
retlw	0x3A	; Low Byte, ADRES = 1022, Reflected Power = 1,082.0 W

RLowByteTable3

retlw	0x17	; Low Byte, ADRES = 3, Reflected Power = 75.84 nW
retlw	0x94	; Low Byte, ADRES = 7, Reflected Power = 83.13 nW
retlw	0x1D	; Low Byte, ADRES = 11, Reflected Power = 91.12 nW
retlw	0xB4	; Low Byte, ADRES = 15, Reflected Power = 99.88 nW
retlw	0x59	; Low Byte, ADRES = 19, Reflected Power = 109.5 nW
retlw	0x07	; Low Byte, ADRES = 23, Reflected Power = 120.0 nW
retlw	0x6A	; Low Byte, ADRES = 27, Reflected Power = 131.5 nW
retlw	0xD7	; Low Byte, ADRES = 31, Reflected Power = 144.2 nW
retlw	0x4E	; Low Byte, ADRES = 35, Reflected Power = 158.0 nW
retlw	0xD0	; Low Byte, ADRES = 39, Reflected Power = 173.2 nW
retlw	0x5F	; Low Byte, ADRES = 43, Reflected Power = 189.9 nW
retlw	0xFC	; Low Byte, ADRES = 47, Reflected Power = 208.1 nW
retlw	0xA8	; Low Byte, ADRES = 51, Reflected Power = 228.1 nW
retlw	0x32	; Low Byte, ADRES = 55, Reflected Power = 250.1 nW
retlw	0x99	; Low Byte, ADRES = 59, Reflected Power = 274.1 nW
retlw	0x0B	; Low Byte, ADRES = 63, Reflected Power = 300.5 nW
retlw	0x87	; Low Byte, ADRES = 67, Reflected Power = 329.3 nW
retlw	0x0F	; Low Byte, ADRES = 71, Reflected Power = 361.0 nW
retlw	0xA4	; Low Byte, ADRES = 75, Reflected Power = 395.7 nW
retlw	0x47	; Low Byte, ADRES = 79, Reflected Power = 433.7 nW
retlw	0xFA	; Low Byte, ADRES = 83, Reflected Power = 475.4 nW
retlw	0x5F	; Low Byte, ADRES = 87, Reflected Power = 521.1 nW
retlw	0xCB	; Low Byte, ADRES = 91, Reflected Power = 571.2 nW
retlw	0x41	; Low Byte, ADRES = 95, Reflected Power = 626.1 nW
retlw	0xC2	; Low Byte, ADRES = 99, Reflected Power = 686.3 nW
retlw	0x50	; Low Byte, ADRES = 103, Reflected Power = 752.3 nW
retlw	0xEB	; Low Byte, ADRES = 107, Reflected Power = 824.6 nW
retlw	0x95	; Low Byte, ADRES = 111, Reflected Power = 903.9 nW
retlw	0x28	; Low Byte, ADRES = 115, Reflected Power = 990.8 nW
retlw	0x8E	; Low Byte, ADRES = 119, Reflected Power = 1.086 uW
retlw	0xFE	; Low Byte, ADRES = 123, Reflected Power = 1.190 uW
retlw	0x79	; Low Byte, ADRES = 127, Reflected Power = 1.305 uW
retlw	0x00	; Low Byte, ADRES = 131, Reflected Power = 1.430 uW
retlw	0x93	; Low Byte, ADRES = 135, Reflected Power = 1.568 uW
retlw	0x35	; Low Byte, ADRES = 139, Reflected Power = 1.718 uW
retlw	0xE7	; Low Byte, ADRES = 143, Reflected Power = 1.884 uW
retlw	0x54	; Low Byte, ADRES = 147, Reflected Power = 2.065 uW
retlw	0xBF	; Low Byte, ADRES = 151, Reflected Power = 2.263 uW
retlw	0x34	; Low Byte, ADRES = 155, Reflected Power = 2.481 uW
retlw	0xB4	; Low Byte, ADRES = 159, Reflected Power = 2.719 uW
retlw	0x40	; Low Byte, ADRES = 163, Reflected Power = 2.981 uW
retlw	0xDA	; Low Byte, ADRES = 167, Reflected Power = 3.267 uW
retlw	0x83	; Low Byte, ADRES = 171, Reflected Power = 3.581 uW
retlw	0x1E	; Low Byte, ADRES = 175, Reflected Power = 3.925 uW
retlw	0x83	; Low Byte, ADRES = 179, Reflected Power = 4.303 uW
retlw	0xF2	; Low Byte, ADRES = 183, Reflected Power = 4.716 uW
retlw	0x6C	; Low Byte, ADRES = 187, Reflected Power = 5.170 uW
retlw	0xF1	; Low Byte, ADRES = 191, Reflected Power = 5.667 uW
retlw	0x83	; Low Byte, ADRES = 195, Reflected Power = 6.211 uW
retlw	0x24	; Low Byte, ADRES = 199, Reflected Power = 6.808 uW
retlw	0xD3	; Low Byte, ADRES = 203, Reflected Power = 7.463 uW
retlw	0x4A	; Low Byte, ADRES = 207, Reflected Power = 8.180 uW
retlw	0xB3	; Low Byte, ADRES = 211, Reflected Power = 8.966 uW
retlw	0x27	; Low Byte, ADRES = 215, Reflected Power = 9.828 uW

retlw	0xA6	; Low Byte, ADRES = 219, Reflected Power = 10.77 uW
retlw	0x31	; Low Byte, ADRES = 223, Reflected Power = 11.81 uW
retlw	0xC9	; Low Byte, ADRES = 227, Reflected Power = 12.94 uW
retlw	0x70	; Low Byte, ADRES = 231, Reflected Power = 14.19 uW
retlw	0x14	; Low Byte, ADRES = 235, Reflected Power = 15.55 uW
retlw	0x78	; Low Byte, ADRES = 239, Reflected Power = 17.05 uW
retlw	0xE6	; Low Byte, ADRES = 243, Reflected Power = 18.69 uW
retlw	0x5E	; Low Byte, ADRES = 247, Reflected Power = 20.48 uW
retlw	0xE3	; Low Byte, ADRES = 251, Reflected Power = 22.45 uW
retlw	0x73	; Low Byte, ADRES = 255, Reflected Power = 24.61 uW
retlw	0x12	; Low Byte, ADRES = 259, Reflected Power = 26.97 uW
retlw	0xC0	; Low Byte, ADRES = 263, Reflected Power = 29.57 uW
retlw	0x3F	; Low Byte, ADRES = 267, Reflected Power = 32.41 uW
retlw	0xA8	; Low Byte, ADRES = 271, Reflected Power = 35.52 uW
retlw	0x1B	; Low Byte, ADRES = 275, Reflected Power = 38.94 uW
retlw	0x98	; Low Byte, ADRES = 279, Reflected Power = 42.68 uW
retlw	0x22	; Low Byte, ADRES = 283, Reflected Power = 46.78 uW
retlw	0xB9	; Low Byte, ADRES = 287, Reflected Power = 51.28 uW
retlw	0x5E	; Low Byte, ADRES = 291, Reflected Power = 56.21 uW
retlw	0x0A	; Low Byte, ADRES = 295, Reflected Power = 61.61 uW
retlw	0x6D	; Low Byte, ADRES = 299, Reflected Power = 67.54 uW
retlw	0xDA	; Low Byte, ADRES = 303, Reflected Power = 74.03 uW
retlw	0x51	; Low Byte, ADRES = 307, Reflected Power = 81.14 uW
retlw	0xD4	; Low Byte, ADRES = 311, Reflected Power = 88.94 uW
retlw	0x64	; Low Byte, ADRES = 315, Reflected Power = 97.49 uW
retlw	0x01	; Low Byte, ADRES = 319, Reflected Power = 106.9 uW
retlw	0xAD	; Low Byte, ADRES = 323, Reflected Power = 117.1 uW
retlw	0x35	; Low Byte, ADRES = 327, Reflected Power = 128.4 uW
retlw	0x9D	; Low Byte, ADRES = 331, Reflected Power = 140.7 uW
retlw	0x0E	; Low Byte, ADRES = 335, Reflected Power = 154.3 uW
retlw	0x8B	; Low Byte, ADRES = 339, Reflected Power = 169.1 uW
retlw	0x13	; Low Byte, ADRES = 343, Reflected Power = 185.4 uW
retlw	0xA8	; Low Byte, ADRES = 347, Reflected Power = 203.2 uW
retlw	0x4C	; Low Byte, ADRES = 351, Reflected Power = 222.7 uW
retlw	0x00	; Low Byte, ADRES = 355, Reflected Power = 244.1 uW
retlw	0x62	; Low Byte, ADRES = 359, Reflected Power = 267.6 uW
retlw	0xCE	; Low Byte, ADRES = 363, Reflected Power = 293.3 uW
retlw	0x44	; Low Byte, ADRES = 367, Reflected Power = 321.5 uW
retlw	0xC6	; Low Byte, ADRES = 371, Reflected Power = 352.4 uW
retlw	0x54	; Low Byte, ADRES = 375, Reflected Power = 386.3 uW
retlw	0xF0	; Low Byte, ADRES = 379, Reflected Power = 423.4 uW
retlw	0x9B	; Low Byte, ADRES = 383, Reflected Power = 464.1 uW
retlw	0x2B	; Low Byte, ADRES = 387, Reflected Power = 508.7 uW
retlw	0x91	; Low Byte, ADRES = 391, Reflected Power = 557.6 uW
retlw	0x02	; Low Byte, ADRES = 395, Reflected Power = 611.2 uW
retlw	0x7D	; Low Byte, ADRES = 399, Reflected Power = 669.9 uW
retlw	0x04	; Low Byte, ADRES = 403, Reflected Power = 734.3 uW
retlw	0x98	; Low Byte, ADRES = 407, Reflected Power = 804.9 uW
retlw	0x3A	; Low Byte, ADRES = 411, Reflected Power = 882.3 uW
retlw	0xEC	; Low Byte, ADRES = 415, Reflected Power = 967.1 uW
retlw	0x58	; Low Byte, ADRES = 419, Reflected Power = 1.060 mW
retlw	0xC2	; Low Byte, ADRES = 423, Reflected Power = 1.162 mW
retlw	0x38	; Low Byte, ADRES = 427, Reflected Power = 1.274 mW
retlw	0xB8	; Low Byte, ADRES = 431, Reflected Power = 1.396 mW
retlw	0x45	; Low Byte, ADRES = 435, Reflected Power = 1.530 mW
retlw	0xDF	; Low Byte, ADRES = 439, Reflected Power = 1.677 mW
retlw	0x88	; Low Byte, ADRES = 443, Reflected Power = 1.839 mW
retlw	0x21	; Low Byte, ADRES = 447, Reflected Power = 2.015 mW
retlw	0x86	; Low Byte, ADRES = 451, Reflected Power = 2.209 mW
retlw	0xF6	; Low Byte, ADRES = 455, Reflected Power = 2.421 mW
retlw	0x70	; Low Byte, ADRES = 459, Reflected Power = 2.654 mW
retlw	0xF5	; Low Byte, ADRES = 463, Reflected Power = 2.909 mW
retlw	0x88	; Low Byte, ADRES = 467, Reflected Power = 3.189 mW
retlw	0x29	; Low Byte, ADRES = 471, Reflected Power = 3.496 mW
retlw	0xD9	; Low Byte, ADRES = 475, Reflected Power = 3.832 mW
retlw	0x4D	; Low Byte, ADRES = 479, Reflected Power = 4.200 mW
retlw	0xB7	; Low Byte, ADRES = 483, Reflected Power = 4.604 mW
retlw	0x2B	; Low Byte, ADRES = 487, Reflected Power = 5.046 mW
retlw	0xAA	; Low Byte, ADRES = 491, Reflected Power = 5.531 mW
retlw	0x35	; Low Byte, ADRES = 495, Reflected Power = 6.063 mW
retlw	0xCE	; Low Byte, ADRES = 499, Reflected Power = 6.646 mW
retlw	0x76	; Low Byte, ADRES = 503, Reflected Power = 7.284 mW
retlw	0x17	; Low Byte, ADRES = 507, Reflected Power = 7.985 mW
retlw	0x7B	; Low Byte, ADRES = 511, Reflected Power = 8.752 mW
retlw	0xE9	; Low Byte, ADRES = 515, Reflected Power = 9.593 mW
retlw	0x62	; Low Byte, ADRES = 519, Reflected Power = 10.52 mW
retlw	0xE7	; Low Byte, ADRES = 523, Reflected Power = 11.53 mW
retlw	0x78	; Low Byte, ADRES = 527, Reflected Power = 12.63 mW
retlw	0x17	; Low Byte, ADRES = 531, Reflected Power = 13.85 mW
retlw	0xC6	; Low Byte, ADRES = 535, Reflected Power = 15.18 mW
retlw	0x42	; Low Byte, ADRES = 539, Reflected Power = 16.64 mW

retlw	0xAB	; Low Byte, ADRES = 543, Reflected Power = 18.24 mW
retlw	0x1E	; Low Byte, ADRES = 547, Reflected Power = 19.99 mW
retlw	0x9C	; Low Byte, ADRES = 551, Reflected Power = 21.91 mW
retlw	0x26	; Low Byte, ADRES = 555, Reflected Power = 24.02 mW
retlw	0xBE	; Low Byte, ADRES = 559, Reflected Power = 26.33 mW
retlw	0x63	; Low Byte, ADRES = 563, Reflected Power = 28.86 mW
retlw	0x0D	; Low Byte, ADRES = 567, Reflected Power = 31.63 mW
retlw	0x70	; Low Byte, ADRES = 571, Reflected Power = 34.67 mW
retlw	0xDD	; Low Byte, ADRES = 575, Reflected Power = 38.01 mW
retlw	0x55	; Low Byte, ADRES = 579, Reflected Power = 41.66 mW
retlw	0xD8	; Low Byte, ADRES = 583, Reflected Power = 45.67 mW
retlw	0x68	; Low Byte, ADRES = 587, Reflected Power = 50.06 mW
retlw	0x06	; Low Byte, ADRES = 591, Reflected Power = 54.87 mW
retlw	0xB3	; Low Byte, ADRES = 595, Reflected Power = 60.14 mW
retlw	0x38	; Low Byte, ADRES = 599, Reflected Power = 65.92 mW
retlw	0xA0	; Low Byte, ADRES = 603, Reflected Power = 72.26 mW
retlw	0x12	; Low Byte, ADRES = 607, Reflected Power = 79.21 mW
retlw	0x8E	; Low Byte, ADRES = 611, Reflected Power = 86.82 mW
retlw	0x17	; Low Byte, ADRES = 615, Reflected Power = 95.16 mW
retlw	0xAD	; Low Byte, ADRES = 619, Reflected Power = 104.3 mW
retlw	0x51	; Low Byte, ADRES = 623, Reflected Power = 114.3 mW
retlw	0x03	; Low Byte, ADRES = 627, Reflected Power = 125.3 mW
retlw	0x65	; Low Byte, ADRES = 631, Reflected Power = 137.4 mW
retlw	0xD2	; Low Byte, ADRES = 635, Reflected Power = 150.6 mW
retlw	0x48	; Low Byte, ADRES = 639, Reflected Power = 165.1 mW
retlw	0xCA	; Low Byte, ADRES = 643, Reflected Power = 180.9 mW
retlw	0x59	; Low Byte, ADRES = 647, Reflected Power = 198.3 mW
retlw	0xF5	; Low Byte, ADRES = 651, Reflected Power = 217.4 mW
retlw	0xA0	; Low Byte, ADRES = 655, Reflected Power = 238.3 mW
retlw	0x2E	; Low Byte, ADRES = 659, Reflected Power = 261.2 mW
retlw	0x95	; Low Byte, ADRES = 663, Reflected Power = 286.3 mW
retlw	0x05	; Low Byte, ADRES = 667, Reflected Power = 313.8 mW
retlw	0x81	; Low Byte, ADRES = 671, Reflected Power = 344.0 mW
retlw	0x08	; Low Byte, ADRES = 675, Reflected Power = 377.0 mW
retlw	0x9D	; Low Byte, ADRES = 679, Reflected Power = 413.3 mW
retlw	0x3F	; Low Byte, ADRES = 683, Reflected Power = 453.0 mW
retlw	0xF2	; Low Byte, ADRES = 687, Reflected Power = 496.5 mW
retlw	0x5B	; Low Byte, ADRES = 691, Reflected Power = 544.3 mW
retlw	0xC6	; Low Byte, ADRES = 695, Reflected Power = 596.6 mW
retlw	0x3B	; Low Byte, ADRES = 699, Reflected Power = 653.9 mW
retlw	0xBC	; Low Byte, ADRES = 703, Reflected Power = 716.8 mW
retlw	0x49	; Low Byte, ADRES = 707, Reflected Power = 785.7 mW
retlw	0xE4	; Low Byte, ADRES = 711, Reflected Power = 861.2 mW
retlw	0x8D	; Low Byte, ADRES = 715, Reflected Power = 944.0 mW
retlw	0x24	; Low Byte, ADRES = 719, Reflected Power = 1.035 W
retlw	0x89	; Low Byte, ADRES = 723, Reflected Power = 1.134 W
retlw	0xF9	; Low Byte, ADRES = 727, Reflected Power = 1.243 W
retlw	0x73	; Low Byte, ADRES = 731, Reflected Power = 1.363 W
retlw	0xFA	; Low Byte, ADRES = 735, Reflected Power = 1.494 W
retlw	0x8D	; Low Byte, ADRES = 739, Reflected Power = 1.637 W
retlw	0x2E	; Low Byte, ADRES = 743, Reflected Power = 1.795 W
retlw	0xDE	; Low Byte, ADRES = 747, Reflected Power = 1.967 W
retlw	0x50	; Low Byte, ADRES = 751, Reflected Power = 2.156 W
retlw	0xBA	; Low Byte, ADRES = 755, Reflected Power = 2.364 W
retlw	0x2F	; Low Byte, ADRES = 759, Reflected Power = 2.591 W
retlw	0xAE	; Low Byte, ADRES = 763, Reflected Power = 2.840 W
retlw	0x3A	; Low Byte, ADRES = 767, Reflected Power = 3.113 W
retlw	0xD3	; Low Byte, ADRES = 771, Reflected Power = 3.412 W
retlw	0x7B	; Low Byte, ADRES = 775, Reflected Power = 3.740 W
retlw	0x19	; Low Byte, ADRES = 779, Reflected Power = 4.100 W
retlw	0x7E	; Low Byte, ADRES = 783, Reflected Power = 4.494 W
retlw	0xED	; Low Byte, ADRES = 787, Reflected Power = 4.926 W
retlw	0x66	; Low Byte, ADRES = 791, Reflected Power = 5.399 W
retlw	0xEB	; Low Byte, ADRES = 795, Reflected Power = 5.918 W
retlw	0x7D	; Low Byte, ADRES = 799, Reflected Power = 6.487 W
retlw	0x1C	; Low Byte, ADRES = 803, Reflected Power = 7.110 W
retlw	0xCB	; Low Byte, ADRES = 807, Reflected Power = 7.794 W
retlw	0x46	; Low Byte, ADRES = 811, Reflected Power = 8.543 W
retlw	0xAF	; Low Byte, ADRES = 815, Reflected Power = 9.364 W
retlw	0x22	; Low Byte, ADRES = 819, Reflected Power = 10.26 W
retlw	0xA0	; Low Byte, ADRES = 823, Reflected Power = 11.25 W
retlw	0x2B	; Low Byte, ADRES = 827, Reflected Power = 12.33 W
retlw	0xC2	; Low Byte, ADRES = 831, Reflected Power = 13.52 W
retlw	0x69	; Low Byte, ADRES = 835, Reflected Power = 14.82 W
retlw	0x0F	; Low Byte, ADRES = 839, Reflected Power = 16.24 W
retlw	0x73	; Low Byte, ADRES = 843, Reflected Power = 17.80 W
retlw	0xE1	; Low Byte, ADRES = 847, Reflected Power = 19.51 W
retlw	0x59	; Low Byte, ADRES = 851, Reflected Power = 21.39 W
retlw	0xDD	; Low Byte, ADRES = 855, Reflected Power = 23.45 W
retlw	0x6D	; Low Byte, ADRES = 859, Reflected Power = 25.70 W
retlw	0x0B	; Low Byte, ADRES = 863, Reflected Power = 28.17 W

```

retlw 0xB8 ; Low Byte, ADRES = 867, Reflected Power = 30.88 W
retlw 0x3B ; Low Byte, ADRES = 871, Reflected Power = 33.85 W
retlw 0xA3 ; Low Byte, ADRES = 875, Reflected Power = 37.10 W
retlw 0x15 ; Low Byte, ADRES = 879, Reflected Power = 40.67 W
retlw 0x92 ; Low Byte, ADRES = 883, Reflected Power = 44.58 W
retlw 0x1C ; Low Byte, ADRES = 887, Reflected Power = 48.86 W
retlw 0xB2 ; Low Byte, ADRES = 891, Reflected Power = 53.56 W
retlw 0x57 ; Low Byte, ADRES = 895, Reflected Power = 58.70 W
retlw 0x06 ; Low Byte, ADRES = 899, Reflected Power = 64.35 W
retlw 0x69 ; Low Byte, ADRES = 903, Reflected Power = 70.53 W
retlw 0xD5 ; Low Byte, ADRES = 907, Reflected Power = 77.31 W
retlw 0x4C ; Low Byte, ADRES = 911, Reflected Power = 84.75 W
retlw 0xCE ; Low Byte, ADRES = 915, Reflected Power = 92.89 W
retlw 0x5D ; Low Byte, ADRES = 919, Reflected Power = 101.8 W
retlw 0xFA ; Low Byte, ADRES = 923, Reflected Power = 111.6 W
retlw 0xA5 ; Low Byte, ADRES = 927, Reflected Power = 122.3 W
retlw 0x31 ; Low Byte, ADRES = 931, Reflected Power = 134.1 W
retlw 0x98 ; Low Byte, ADRES = 935, Reflected Power = 147.0 W
retlw 0x09 ; Low Byte, ADRES = 939, Reflected Power = 161.1 W
retlw 0x85 ; Low Byte, ADRES = 943, Reflected Power = 176.6 W
retlw 0x0D ; Low Byte, ADRES = 947, Reflected Power = 193.6 W
retlw 0xA1 ; Low Byte, ADRES = 951, Reflected Power = 212.2 W
retlw 0x45 ; Low Byte, ADRES = 955, Reflected Power = 232.6 W
retlw 0xF8 ; Low Byte, ADRES = 959, Reflected Power = 254.9 W
retlw 0x5E ; Low Byte, ADRES = 963, Reflected Power = 279.4 W
retlw 0xC9 ; Low Byte, ADRES = 967, Reflected Power = 306.3 W
retlw 0x3F ; Low Byte, ADRES = 971, Reflected Power = 335.7 W
retlw 0xC0 ; Low Byte, ADRES = 975, Reflected Power = 368.0 W
retlw 0x4E ; Low Byte, ADRES = 979, Reflected Power = 403.4 W
retlw 0xE9 ; Low Byte, ADRES = 983, Reflected Power = 442.2 W
retlw 0x93 ; Low Byte, ADRES = 987, Reflected Power = 484.7 W
retlw 0x27 ; Low Byte, ADRES = 991, Reflected Power = 531.3 W
retlw 0x8D ; Low Byte, ADRES = 995, Reflected Power = 582.3 W
retlw 0xFD ; Low Byte, ADRES = 999, Reflected Power = 638.3 W
retlw 0x77 ; Low Byte, ADRES = 1003, Reflected Power = 699.7 W
retlw 0xFE ; Low Byte, ADRES = 1007, Reflected Power = 766.9 W
retlw 0x91 ; Low Byte, ADRES = 1011, Reflected Power = 840.6 W
retlw 0x33 ; Low Byte, ADRES = 1015, Reflected Power = 921.5 W
retlw 0xE4 ; Low Byte, ADRES = 1019, Reflected Power = 1,010.0 W
retlw 0x53 ; Low Byte, ADRES = 1023, Reflected Power = 1,107.1 W

```

RHighByteTable0

```

retlw 0x0C ; High Byte, ADRES = 0, Reflected Power = 70.79 nW
retlw 0x0D ; High Byte, ADRES = 4, Reflected Power = 77.60 nW
retlw 0x0D ; High Byte, ADRES = 8, Reflected Power = 85.06 nW
retlw 0x0E ; High Byte, ADRES = 12, Reflected Power = 93.24 nW
retlw 0x0E ; High Byte, ADRES = 16, Reflected Power = 102.2 nW
retlw 0x0F ; High Byte, ADRES = 20, Reflected Power = 112.0 nW
retlw 0x10 ; High Byte, ADRES = 24, Reflected Power = 122.8 nW
retlw 0x10 ; High Byte, ADRES = 28, Reflected Power = 134.6 nW
retlw 0x10 ; High Byte, ADRES = 32, Reflected Power = 147.5 nW
retlw 0x11 ; High Byte, ADRES = 36, Reflected Power = 161.7 nW
retlw 0x11 ; High Byte, ADRES = 40, Reflected Power = 177.3 nW
retlw 0x12 ; High Byte, ADRES = 44, Reflected Power = 194.3 nW
retlw 0x13 ; High Byte, ADRES = 48, Reflected Power = 213.0 nW
retlw 0x13 ; High Byte, ADRES = 52, Reflected Power = 233.4 nW
retlw 0x14 ; High Byte, ADRES = 56, Reflected Power = 255.9 nW
retlw 0x14 ; High Byte, ADRES = 60, Reflected Power = 280.5 nW
retlw 0x15 ; High Byte, ADRES = 64, Reflected Power = 307.4 nW
retlw 0x15 ; High Byte, ADRES = 68, Reflected Power = 337.0 nW
retlw 0x16 ; High Byte, ADRES = 72, Reflected Power = 369.4 nW
retlw 0x16 ; High Byte, ADRES = 76, Reflected Power = 404.9 nW
retlw 0x17 ; High Byte, ADRES = 80, Reflected Power = 443.8 nW
retlw 0x18 ; High Byte, ADRES = 84, Reflected Power = 486.5 nW
retlw 0x18 ; High Byte, ADRES = 88, Reflected Power = 533.2 nW
retlw 0x18 ; High Byte, ADRES = 92, Reflected Power = 584.5 nW
retlw 0x19 ; High Byte, ADRES = 96, Reflected Power = 640.7 nW
retlw 0x19 ; High Byte, ADRES = 100, Reflected Power = 702.3 nW
retlw 0x1A ; High Byte, ADRES = 104, Reflected Power = 769.8 nW
retlw 0x1B ; High Byte, ADRES = 108, Reflected Power = 843.8 nW
retlw 0x1B ; High Byte, ADRES = 112, Reflected Power = 924.9 nW
retlw 0x1C ; High Byte, ADRES = 116, Reflected Power = 1.014 uW
retlw 0x1C ; High Byte, ADRES = 120, Reflected Power = 1.111 uW
retlw 0x1D ; High Byte, ADRES = 124, Reflected Power = 1.218 uW
retlw 0x1D ; High Byte, ADRES = 128, Reflected Power = 1.335 uW
retlw 0x1E ; High Byte, ADRES = 132, Reflected Power = 1.463 uW
retlw 0x1E ; High Byte, ADRES = 136, Reflected Power = 1.604 uW
retlw 0x1F ; High Byte, ADRES = 140, Reflected Power = 1.758 uW
retlw 0x20 ; High Byte, ADRES = 144, Reflected Power = 1.927 uW
retlw 0x20 ; High Byte, ADRES = 148, Reflected Power = 2.113 uW
retlw 0x20 ; High Byte, ADRES = 152, Reflected Power = 2.316 uW

```

retlw	0x21	; High Byte, ADRES = 156, Reflected Power = 2.538 uW
retlw	0x21	; High Byte, ADRES = 160, Reflected Power = 2.782 uW
retlw	0x22	; High Byte, ADRES = 164, Reflected Power = 3.050 uW
retlw	0x23	; High Byte, ADRES = 168, Reflected Power = 3.343 uW
retlw	0x23	; High Byte, ADRES = 172, Reflected Power = 3.664 uW
retlw	0x24	; High Byte, ADRES = 176, Reflected Power = 4.016 uW
retlw	0x24	; High Byte, ADRES = 180, Reflected Power = 4.403 uW
retlw	0x25	; High Byte, ADRES = 184, Reflected Power = 4.826 uW
retlw	0x25	; High Byte, ADRES = 188, Reflected Power = 5.290 uW
retlw	0x26	; High Byte, ADRES = 192, Reflected Power = 5.798 uW
retlw	0x26	; High Byte, ADRES = 196, Reflected Power = 6.355 uW
retlw	0x27	; High Byte, ADRES = 200, Reflected Power = 6.966 uW
retlw	0x28	; High Byte, ADRES = 204, Reflected Power = 7.636 uW
retlw	0x28	; High Byte, ADRES = 208, Reflected Power = 8.370 uW
retlw	0x28	; High Byte, ADRES = 212, Reflected Power = 9.174 uW
retlw	0x29	; High Byte, ADRES = 216, Reflected Power = 10.06 uW
retlw	0x29	; High Byte, ADRES = 220, Reflected Power = 11.02 uW
retlw	0x2A	; High Byte, ADRES = 224, Reflected Power = 12.08 uW
retlw	0x2A	; High Byte, ADRES = 228, Reflected Power = 13.24 uW
retlw	0x2B	; High Byte, ADRES = 232, Reflected Power = 14.52 uW
retlw	0x2C	; High Byte, ADRES = 236, Reflected Power = 15.91 uW
retlw	0x2C	; High Byte, ADRES = 240, Reflected Power = 17.44 uW
retlw	0x2D	; High Byte, ADRES = 244, Reflected Power = 19.12 uW
retlw	0x2D	; High Byte, ADRES = 248, Reflected Power = 20.96 uW
retlw	0x2E	; High Byte, ADRES = 252, Reflected Power = 22.97 uW
retlw	0x2E	; High Byte, ADRES = 256, Reflected Power = 25.18 uW
retlw	0x2F	; High Byte, ADRES = 260, Reflected Power = 27.60 uW
retlw	0x2F	; High Byte, ADRES = 264, Reflected Power = 30.25 uW
retlw	0x30	; High Byte, ADRES = 268, Reflected Power = 33.16 uW
retlw	0x30	; High Byte, ADRES = 272, Reflected Power = 36.35 uW
retlw	0x31	; High Byte, ADRES = 276, Reflected Power = 39.84 uW
retlw	0x31	; High Byte, ADRES = 280, Reflected Power = 43.67 uW
retlw	0x32	; High Byte, ADRES = 284, Reflected Power = 47.87 uW
retlw	0x32	; High Byte, ADRES = 288, Reflected Power = 52.47 uW
retlw	0x33	; High Byte, ADRES = 292, Reflected Power = 57.51 uW
retlw	0x34	; High Byte, ADRES = 296, Reflected Power = 63.04 uW
retlw	0x34	; High Byte, ADRES = 300, Reflected Power = 69.10 uW
retlw	0x34	; High Byte, ADRES = 304, Reflected Power = 75.75 uW
retlw	0x35	; High Byte, ADRES = 308, Reflected Power = 83.03 uW
retlw	0x35	; High Byte, ADRES = 312, Reflected Power = 91.01 uW
retlw	0x36	; High Byte, ADRES = 316, Reflected Power = 99.76 uW
retlw	0x37	; High Byte, ADRES = 320, Reflected Power = 109.3 uW
retlw	0x37	; High Byte, ADRES = 324, Reflected Power = 119.9 uW
retlw	0x38	; High Byte, ADRES = 328, Reflected Power = 131.4 uW
retlw	0x38	; High Byte, ADRES = 332, Reflected Power = 144.0 uW
retlw	0x39	; High Byte, ADRES = 336, Reflected Power = 157.8 uW
retlw	0x39	; High Byte, ADRES = 340, Reflected Power = 173.0 uW
retlw	0x3A	; High Byte, ADRES = 344, Reflected Power = 189.7 uW
retlw	0x3A	; High Byte, ADRES = 348, Reflected Power = 207.9 uW
retlw	0x3B	; High Byte, ADRES = 352, Reflected Power = 227.9 uW
retlw	0x3C	; High Byte, ADRES = 356, Reflected Power = 249.8 uW
retlw	0x3C	; High Byte, ADRES = 360, Reflected Power = 273.8 uW
retlw	0x3C	; High Byte, ADRES = 364, Reflected Power = 300.1 uW
retlw	0x3D	; High Byte, ADRES = 368, Reflected Power = 328.9 uW
retlw	0x3D	; High Byte, ADRES = 372, Reflected Power = 360.6 uW
retlw	0x3E	; High Byte, ADRES = 376, Reflected Power = 395.2 uW
retlw	0x3F	; High Byte, ADRES = 380, Reflected Power = 433.2 uW
retlw	0x3F	; High Byte, ADRES = 384, Reflected Power = 474.9 uW
retlw	0x40	; High Byte, ADRES = 388, Reflected Power = 520.5 uW
retlw	0x40	; High Byte, ADRES = 392, Reflected Power = 570.5 uW
retlw	0x41	; High Byte, ADRES = 396, Reflected Power = 625.4 uW
retlw	0x41	; High Byte, ADRES = 400, Reflected Power = 685.5 uW
retlw	0x42	; High Byte, ADRES = 404, Reflected Power = 751.4 uW
retlw	0x42	; High Byte, ADRES = 408, Reflected Power = 823.6 uW
retlw	0x43	; High Byte, ADRES = 412, Reflected Power = 902.8 uW
retlw	0x44	; High Byte, ADRES = 416, Reflected Power = 989.6 uW
retlw	0x44	; High Byte, ADRES = 420, Reflected Power = 1.085 mW
retlw	0x44	; High Byte, ADRES = 424, Reflected Power = 1.189 mW
retlw	0x45	; High Byte, ADRES = 428, Reflected Power = 1.303 mW
retlw	0x45	; High Byte, ADRES = 432, Reflected Power = 1.428 mW
retlw	0x46	; High Byte, ADRES = 436, Reflected Power = 1.566 mW
retlw	0x47	; High Byte, ADRES = 440, Reflected Power = 1.716 mW
retlw	0x47	; High Byte, ADRES = 444, Reflected Power = 1.881 mW
retlw	0x48	; High Byte, ADRES = 448, Reflected Power = 2.062 mW
retlw	0x48	; High Byte, ADRES = 452, Reflected Power = 2.260 mW
retlw	0x49	; High Byte, ADRES = 456, Reflected Power = 2.478 mW
retlw	0x49	; High Byte, ADRES = 460, Reflected Power = 2.716 mW
retlw	0x4A	; High Byte, ADRES = 464, Reflected Power = 2.977 mW
retlw	0x4A	; High Byte, ADRES = 468, Reflected Power = 3.263 mW
retlw	0x4B	; High Byte, ADRES = 472, Reflected Power = 3.577 mW
retlw	0x4C	; High Byte, ADRES = 476, Reflected Power = 3.920 mW

retlw	0x4C	; High Byte, ADRES = 480, Reflected Power = 4.297 mW
retlw	0x4C	; High Byte, ADRES = 484, Reflected Power = 4.710 mW
retlw	0x4D	; High Byte, ADRES = 488, Reflected Power = 5.163 mW
retlw	0x4D	; High Byte, ADRES = 492, Reflected Power = 5.660 mW
retlw	0x4E	; High Byte, ADRES = 496, Reflected Power = 6.204 mW
retlw	0x4E	; High Byte, ADRES = 500, Reflected Power = 6.800 mW
retlw	0x4F	; High Byte, ADRES = 504, Reflected Power = 7.453 mW
retlw	0x50	; High Byte, ADRES = 508, Reflected Power = 8.170 mW
retlw	0x50	; High Byte, ADRES = 512, Reflected Power = 8.955 mW
retlw	0x51	; High Byte, ADRES = 516, Reflected Power = 9.816 mW
retlw	0x51	; High Byte, ADRES = 520, Reflected Power = 10.76 mW
retlw	0x52	; High Byte, ADRES = 524, Reflected Power = 11.79 mW
retlw	0x52	; High Byte, ADRES = 528, Reflected Power = 12.93 mW
retlw	0x53	; High Byte, ADRES = 532, Reflected Power = 14.17 mW
retlw	0x53	; High Byte, ADRES = 536, Reflected Power = 15.53 mW
retlw	0x54	; High Byte, ADRES = 540, Reflected Power = 17.03 mW
retlw	0x54	; High Byte, ADRES = 544, Reflected Power = 18.66 mW
retlw	0x55	; High Byte, ADRES = 548, Reflected Power = 20.46 mW
retlw	0x55	; High Byte, ADRES = 552, Reflected Power = 22.42 mW
retlw	0x56	; High Byte, ADRES = 556, Reflected Power = 24.58 mW
retlw	0x56	; High Byte, ADRES = 560, Reflected Power = 26.94 mW
retlw	0x57	; High Byte, ADRES = 564, Reflected Power = 29.53 mW
retlw	0x58	; High Byte, ADRES = 568, Reflected Power = 32.37 mW
retlw	0x58	; High Byte, ADRES = 572, Reflected Power = 35.48 mW
retlw	0x58	; High Byte, ADRES = 576, Reflected Power = 38.89 mW
retlw	0x59	; High Byte, ADRES = 580, Reflected Power = 42.63 mW
retlw	0x59	; High Byte, ADRES = 584, Reflected Power = 46.73 mW
retlw	0x5A	; High Byte, ADRES = 588, Reflected Power = 51.22 mW
retlw	0x5B	; High Byte, ADRES = 592, Reflected Power = 56.14 mW
retlw	0x5B	; High Byte, ADRES = 596, Reflected Power = 61.54 mW
retlw	0x5C	; High Byte, ADRES = 600, Reflected Power = 67.45 mW
retlw	0x5C	; High Byte, ADRES = 604, Reflected Power = 73.94 mW
retlw	0x5D	; High Byte, ADRES = 608, Reflected Power = 81.04 mW
retlw	0x5D	; High Byte, ADRES = 612, Reflected Power = 88.83 mW
retlw	0x5E	; High Byte, ADRES = 616, Reflected Power = 97.37 mW
retlw	0x5E	; High Byte, ADRES = 620, Reflected Power = 106.7 mW
retlw	0x5F	; High Byte, ADRES = 624, Reflected Power = 117.0 mW
retlw	0x60	; High Byte, ADRES = 628, Reflected Power = 128.2 mW
retlw	0x60	; High Byte, ADRES = 632, Reflected Power = 140.6 mW
retlw	0x60	; High Byte, ADRES = 636, Reflected Power = 154.1 mW
retlw	0x61	; High Byte, ADRES = 640, Reflected Power = 168.9 mW
retlw	0x61	; High Byte, ADRES = 644, Reflected Power = 185.1 mW
retlw	0x62	; High Byte, ADRES = 648, Reflected Power = 202.9 mW
retlw	0x63	; High Byte, ADRES = 652, Reflected Power = 222.4 mW
retlw	0x63	; High Byte, ADRES = 656, Reflected Power = 243.8 mW
retlw	0x64	; High Byte, ADRES = 660, Reflected Power = 267.2 mW
retlw	0x64	; High Byte, ADRES = 664, Reflected Power = 292.9 mW
retlw	0x65	; High Byte, ADRES = 668, Reflected Power = 321.1 mW
retlw	0x65	; High Byte, ADRES = 672, Reflected Power = 351.9 mW
retlw	0x66	; High Byte, ADRES = 676, Reflected Power = 385.8 mW
retlw	0x66	; High Byte, ADRES = 680, Reflected Power = 422.9 mW
retlw	0x67	; High Byte, ADRES = 684, Reflected Power = 463.5 mW
retlw	0x68	; High Byte, ADRES = 688, Reflected Power = 508.1 mW
retlw	0x68	; High Byte, ADRES = 692, Reflected Power = 556.9 mW
retlw	0x68	; High Byte, ADRES = 696, Reflected Power = 610.4 mW
retlw	0x69	; High Byte, ADRES = 700, Reflected Power = 669.1 mW
retlw	0x69	; High Byte, ADRES = 704, Reflected Power = 733.4 mW
retlw	0x6A	; High Byte, ADRES = 708, Reflected Power = 803.9 mW
retlw	0x6B	; High Byte, ADRES = 712, Reflected Power = 881.2 mW
retlw	0x6B	; High Byte, ADRES = 716, Reflected Power = 965.9 mW
retlw	0x6C	; High Byte, ADRES = 720, Reflected Power = 1.059 W
retlw	0x6C	; High Byte, ADRES = 724, Reflected Power = 1.161 W
retlw	0x6D	; High Byte, ADRES = 728, Reflected Power = 1.272 W
retlw	0x6D	; High Byte, ADRES = 732, Reflected Power = 1.394 W
retlw	0x6E	; High Byte, ADRES = 736, Reflected Power = 1.528 W
retlw	0x6E	; High Byte, ADRES = 740, Reflected Power = 1.675 W
retlw	0x6F	; High Byte, ADRES = 744, Reflected Power = 1.836 W
retlw	0x70	; High Byte, ADRES = 748, Reflected Power = 2.013 W
retlw	0x70	; High Byte, ADRES = 752, Reflected Power = 2.206 W
retlw	0x70	; High Byte, ADRES = 756, Reflected Power = 2.418 W
retlw	0x71	; High Byte, ADRES = 760, Reflected Power = 2.651 W
retlw	0x71	; High Byte, ADRES = 764, Reflected Power = 2.906 W
retlw	0x72	; High Byte, ADRES = 768, Reflected Power = 3.185 W
retlw	0x72	; High Byte, ADRES = 772, Reflected Power = 3.491 W
retlw	0x73	; High Byte, ADRES = 776, Reflected Power = 3.827 W
retlw	0x74	; High Byte, ADRES = 780, Reflected Power = 4.195 W
retlw	0x74	; High Byte, ADRES = 784, Reflected Power = 4.598 W
retlw	0x75	; High Byte, ADRES = 788, Reflected Power = 5.040 W
retlw	0x75	; High Byte, ADRES = 792, Reflected Power = 5.524 W
retlw	0x76	; High Byte, ADRES = 796, Reflected Power = 6.055 W
retlw	0x76	; High Byte, ADRES = 800, Reflected Power = 6.637 W

retlw	0x77	; High Byte, ADRES = 804, Reflected Power = 7.275 W
retlw	0x77	; High Byte, ADRES = 808, Reflected Power = 7.975 W
retlw	0x78	; High Byte, ADRES = 812, Reflected Power = 8.741 W
retlw	0x78	; High Byte, ADRES = 816, Reflected Power = 9.582 W
retlw	0x79	; High Byte, ADRES = 820, Reflected Power = 10.50 W
retlw	0x79	; High Byte, ADRES = 824, Reflected Power = 11.51 W
retlw	0x7A	; High Byte, ADRES = 828, Reflected Power = 12.62 W
retlw	0x7A	; High Byte, ADRES = 832, Reflected Power = 13.83 W
retlw	0x7B	; High Byte, ADRES = 836, Reflected Power = 15.16 W
retlw	0x7C	; High Byte, ADRES = 840, Reflected Power = 16.62 W
retlw	0x7C	; High Byte, ADRES = 844, Reflected Power = 18.22 W
retlw	0x7C	; High Byte, ADRES = 848, Reflected Power = 19.97 W
retlw	0x7D	; High Byte, ADRES = 852, Reflected Power = 21.89 W
retlw	0x7D	; High Byte, ADRES = 856, Reflected Power = 23.99 W
retlw	0x7E	; High Byte, ADRES = 860, Reflected Power = 26.30 W
retlw	0x7F	; High Byte, ADRES = 864, Reflected Power = 28.82 W
retlw	0x7F	; High Byte, ADRES = 868, Reflected Power = 31.60 W
retlw	0x80	; High Byte, ADRES = 872, Reflected Power = 34.63 W
retlw	0x80	; High Byte, ADRES = 876, Reflected Power = 37.96 W
retlw	0x81	; High Byte, ADRES = 880, Reflected Power = 41.61 W
retlw	0x81	; High Byte, ADRES = 884, Reflected Power = 45.61 W
retlw	0x82	; High Byte, ADRES = 888, Reflected Power = 49.99 W
retlw	0x82	; High Byte, ADRES = 892, Reflected Power = 54.80 W
retlw	0x83	; High Byte, ADRES = 896, Reflected Power = 60.07 W
retlw	0x84	; High Byte, ADRES = 900, Reflected Power = 65.84 W
retlw	0x84	; High Byte, ADRES = 904, Reflected Power = 72.17 W
retlw	0x84	; High Byte, ADRES = 908, Reflected Power = 79.11 W
retlw	0x85	; High Byte, ADRES = 912, Reflected Power = 86.71 W
retlw	0x85	; High Byte, ADRES = 916, Reflected Power = 95.05 W
retlw	0x86	; High Byte, ADRES = 920, Reflected Power = 104.2 W
retlw	0x87	; High Byte, ADRES = 924, Reflected Power = 114.2 W
retlw	0x87	; High Byte, ADRES = 928, Reflected Power = 125.2 W
retlw	0x88	; High Byte, ADRES = 932, Reflected Power = 137.2 W
retlw	0x88	; High Byte, ADRES = 936, Reflected Power = 150.4 W
retlw	0x89	; High Byte, ADRES = 940, Reflected Power = 164.9 W
retlw	0x89	; High Byte, ADRES = 944, Reflected Power = 180.7 W
retlw	0x8A	; High Byte, ADRES = 948, Reflected Power = 198.1 W
retlw	0x8A	; High Byte, ADRES = 952, Reflected Power = 217.1 W
retlw	0x8B	; High Byte, ADRES = 956, Reflected Power = 238.0 W
retlw	0x8C	; High Byte, ADRES = 960, Reflected Power = 260.9 W
retlw	0x8C	; High Byte, ADRES = 964, Reflected Power = 285.9 W
retlw	0x8C	; High Byte, ADRES = 968, Reflected Power = 313.4 W
retlw	0x8D	; High Byte, ADRES = 972, Reflected Power = 343.5 W
retlw	0x8D	; High Byte, ADRES = 976, Reflected Power = 376.6 W
retlw	0x8E	; High Byte, ADRES = 980, Reflected Power = 412.8 W
retlw	0x8F	; High Byte, ADRES = 984, Reflected Power = 452.4 W
retlw	0x8F	; High Byte, ADRES = 988, Reflected Power = 495.9 W
retlw	0x90	; High Byte, ADRES = 992, Reflected Power = 543.6 W
retlw	0x90	; High Byte, ADRES = 996, Reflected Power = 595.9 W
retlw	0x91	; High Byte, ADRES = 1000, Reflected Power = 653.1 W
retlw	0x91	; High Byte, ADRES = 1004, Reflected Power = 715.9 W
retlw	0x92	; High Byte, ADRES = 1008, Reflected Power = 784.7 W
retlw	0x92	; High Byte, ADRES = 1012, Reflected Power = 860.2 W
retlw	0x93	; High Byte, ADRES = 1016, Reflected Power = 942.8 W
retlw	0x94	; High Byte, ADRES = 1020, Reflected Power = 1,033.5 W

RHighByteTable1

retlw	0x0C	; High Byte, ADRES = 1, Reflected Power = 72.44 nW
retlw	0x0D	; High Byte, ADRES = 5, Reflected Power = 79.40 nW
retlw	0x0D	; High Byte, ADRES = 9, Reflected Power = 87.03 nW
retlw	0x0E	; High Byte, ADRES = 13, Reflected Power = 95.40 nW
retlw	0x0F	; High Byte, ADRES = 17, Reflected Power = 104.6 nW
retlw	0x0F	; High Byte, ADRES = 21, Reflected Power = 114.6 nW
retlw	0x10	; High Byte, ADRES = 25, Reflected Power = 125.6 nW
retlw	0x10	; High Byte, ADRES = 29, Reflected Power = 137.7 nW
retlw	0x11	; High Byte, ADRES = 33, Reflected Power = 151.0 nW
retlw	0x11	; High Byte, ADRES = 37, Reflected Power = 165.5 nW
retlw	0x12	; High Byte, ADRES = 41, Reflected Power = 181.4 nW
retlw	0x12	; High Byte, ADRES = 45, Reflected Power = 198.8 nW
retlw	0x13	; High Byte, ADRES = 49, Reflected Power = 217.9 nW
retlw	0x14	; High Byte, ADRES = 53, Reflected Power = 238.9 nW
retlw	0x14	; High Byte, ADRES = 57, Reflected Power = 261.8 nW
retlw	0x14	; High Byte, ADRES = 61, Reflected Power = 287.0 nW
retlw	0x15	; High Byte, ADRES = 65, Reflected Power = 314.6 nW
retlw	0x15	; High Byte, ADRES = 69, Reflected Power = 344.8 nW
retlw	0x16	; High Byte, ADRES = 73, Reflected Power = 378.0 nW
retlw	0x16	; High Byte, ADRES = 77, Reflected Power = 414.3 nW
retlw	0x17	; High Byte, ADRES = 81, Reflected Power = 454.1 nW
retlw	0x18	; High Byte, ADRES = 85, Reflected Power = 497.8 nW
retlw	0x18	; High Byte, ADRES = 89, Reflected Power = 545.6 nW
retlw	0x19	; High Byte, ADRES = 93, Reflected Power = 598.1 nW

retlw	0x19	; High Byte, ADRES = 97, Reflected Power = 655.5 nW
retlw	0x1A	; High Byte, ADRES = 101, Reflected Power = 718.6 nW
retlw	0x1A	; High Byte, ADRES = 105, Reflected Power = 787.6 nW
retlw	0x1B	; High Byte, ADRES = 109, Reflected Power = 863.3 nW
retlw	0x1B	; High Byte, ADRES = 113, Reflected Power = 946.3 nW
retlw	0x1C	; High Byte, ADRES = 117, Reflected Power = 1.037 uW
retlw	0x1C	; High Byte, ADRES = 121, Reflected Power = 1.137 uW
retlw	0x1D	; High Byte, ADRES = 125, Reflected Power = 1.246 uW
retlw	0x1D	; High Byte, ADRES = 129, Reflected Power = 1.366 uW
retlw	0x1E	; High Byte, ADRES = 133, Reflected Power = 1.497 uW
retlw	0x1E	; High Byte, ADRES = 137, Reflected Power = 1.641 uW
retlw	0x1F	; High Byte, ADRES = 141, Reflected Power = 1.799 uW
retlw	0x20	; High Byte, ADRES = 145, Reflected Power = 1.972 uW
retlw	0x20	; High Byte, ADRES = 149, Reflected Power = 2.162 uW
retlw	0x20	; High Byte, ADRES = 153, Reflected Power = 2.369 uW
retlw	0x21	; High Byte, ADRES = 157, Reflected Power = 2.597 uW
retlw	0x21	; High Byte, ADRES = 161, Reflected Power = 2.847 uW
retlw	0x22	; High Byte, ADRES = 165, Reflected Power = 3.121 uW
retlw	0x23	; High Byte, ADRES = 169, Reflected Power = 3.420 uW
retlw	0x23	; High Byte, ADRES = 173, Reflected Power = 3.749 uW
retlw	0x24	; High Byte, ADRES = 177, Reflected Power = 4.110 uW
retlw	0x24	; High Byte, ADRES = 181, Reflected Power = 4.505 uW
retlw	0x25	; High Byte, ADRES = 185, Reflected Power = 4.938 uW
retlw	0x25	; High Byte, ADRES = 189, Reflected Power = 5.412 uW
retlw	0x26	; High Byte, ADRES = 193, Reflected Power = 5.933 uW
retlw	0x26	; High Byte, ADRES = 197, Reflected Power = 6.503 uW
retlw	0x27	; High Byte, ADRES = 201, Reflected Power = 7.128 uW
retlw	0x28	; High Byte, ADRES = 205, Reflected Power = 7.813 uW
retlw	0x28	; High Byte, ADRES = 209, Reflected Power = 8.564 uW
retlw	0x28	; High Byte, ADRES = 213, Reflected Power = 9.387 uW
retlw	0x29	; High Byte, ADRES = 217, Reflected Power = 10.29 uW
retlw	0x29	; High Byte, ADRES = 221, Reflected Power = 11.28 uW
retlw	0x2A	; High Byte, ADRES = 225, Reflected Power = 12.36 uW
retlw	0x2B	; High Byte, ADRES = 229, Reflected Power = 13.55 uW
retlw	0x2B	; High Byte, ADRES = 233, Reflected Power = 14.85 uW
retlw	0x2C	; High Byte, ADRES = 237, Reflected Power = 16.28 uW
retlw	0x2C	; High Byte, ADRES = 241, Reflected Power = 17.85 uW
retlw	0x2D	; High Byte, ADRES = 245, Reflected Power = 19.56 uW
retlw	0x2D	; High Byte, ADRES = 249, Reflected Power = 21.44 uW
retlw	0x2E	; High Byte, ADRES = 253, Reflected Power = 23.50 uW
retlw	0x2E	; High Byte, ADRES = 257, Reflected Power = 25.76 uW
retlw	0x2F	; High Byte, ADRES = 261, Reflected Power = 28.24 uW
retlw	0x30	; High Byte, ADRES = 265, Reflected Power = 30.95 uW
retlw	0x30	; High Byte, ADRES = 269, Reflected Power = 33.93 uW
retlw	0x30	; High Byte, ADRES = 273, Reflected Power = 37.19 uW
retlw	0x31	; High Byte, ADRES = 277, Reflected Power = 40.77 uW
retlw	0x31	; High Byte, ADRES = 281, Reflected Power = 44.69 uW
retlw	0x32	; High Byte, ADRES = 285, Reflected Power = 48.98 uW
retlw	0x33	; High Byte, ADRES = 289, Reflected Power = 53.69 uW
retlw	0x33	; High Byte, ADRES = 293, Reflected Power = 58.85 uW
retlw	0x34	; High Byte, ADRES = 297, Reflected Power = 64.51 uW
retlw	0x34	; High Byte, ADRES = 301, Reflected Power = 70.71 uW
retlw	0x35	; High Byte, ADRES = 305, Reflected Power = 77.50 uW
retlw	0x35	; High Byte, ADRES = 309, Reflected Power = 84.95 uW
retlw	0x36	; High Byte, ADRES = 313, Reflected Power = 93.12 uW
retlw	0x36	; High Byte, ADRES = 317, Reflected Power = 102.1 uW
retlw	0x37	; High Byte, ADRES = 321, Reflected Power = 111.9 uW
retlw	0x38	; High Byte, ADRES = 325, Reflected Power = 122.6 uW
retlw	0x38	; High Byte, ADRES = 329, Reflected Power = 134.4 uW
retlw	0x38	; High Byte, ADRES = 333, Reflected Power = 147.3 uW
retlw	0x39	; High Byte, ADRES = 337, Reflected Power = 161.5 uW
retlw	0x39	; High Byte, ADRES = 341, Reflected Power = 177.0 uW
retlw	0x3A	; High Byte, ADRES = 345, Reflected Power = 194.1 uW
retlw	0x3A	; High Byte, ADRES = 349, Reflected Power = 212.7 uW
retlw	0x3B	; High Byte, ADRES = 353, Reflected Power = 233.2 uW
retlw	0x3C	; High Byte, ADRES = 357, Reflected Power = 255.6 uW
retlw	0x3C	; High Byte, ADRES = 361, Reflected Power = 280.1 uW
retlw	0x3D	; High Byte, ADRES = 365, Reflected Power = 307.1 uW
retlw	0x3D	; High Byte, ADRES = 369, Reflected Power = 336.6 uW
retlw	0x3E	; High Byte, ADRES = 373, Reflected Power = 368.9 uW
retlw	0x3E	; High Byte, ADRES = 377, Reflected Power = 404.4 uW
retlw	0x3F	; High Byte, ADRES = 381, Reflected Power = 443.3 uW
retlw	0x3F	; High Byte, ADRES = 385, Reflected Power = 485.9 uW
retlw	0x40	; High Byte, ADRES = 389, Reflected Power = 532.6 uW
retlw	0x40	; High Byte, ADRES = 393, Reflected Power = 583.8 uW
retlw	0x41	; High Byte, ADRES = 397, Reflected Power = 639.9 uW
retlw	0x41	; High Byte, ADRES = 401, Reflected Power = 701.4 uW
retlw	0x42	; High Byte, ADRES = 405, Reflected Power = 768.8 uW
retlw	0x42	; High Byte, ADRES = 409, Reflected Power = 842.7 uW
retlw	0x43	; High Byte, ADRES = 413, Reflected Power = 923.7 uW
retlw	0x44	; High Byte, ADRES = 417, Reflected Power = 1.013 mW

retlw	0x44	; High Byte, ADRES = 421, Reflected Power = 1.110 mW
retlw	0x44	; High Byte, ADRES = 425, Reflected Power = 1.217 mW
retlw	0x45	; High Byte, ADRES = 429, Reflected Power = 1.333 mW
retlw	0x45	; High Byte, ADRES = 433, Reflected Power = 1.462 mW
retlw	0x46	; High Byte, ADRES = 437, Reflected Power = 1.602 mW
retlw	0x47	; High Byte, ADRES = 441, Reflected Power = 1.756 mW
retlw	0x47	; High Byte, ADRES = 445, Reflected Power = 1.925 mW
retlw	0x48	; High Byte, ADRES = 449, Reflected Power = 2.110 mW
retlw	0x48	; High Byte, ADRES = 453, Reflected Power = 2.313 mW
retlw	0x49	; High Byte, ADRES = 457, Reflected Power = 2.535 mW
retlw	0x49	; High Byte, ADRES = 461, Reflected Power = 2.779 mW
retlw	0x4A	; High Byte, ADRES = 465, Reflected Power = 3.046 mW
retlw	0x4A	; High Byte, ADRES = 469, Reflected Power = 3.339 mW
retlw	0x4B	; High Byte, ADRES = 473, Reflected Power = 3.660 mW
retlw	0x4C	; High Byte, ADRES = 477, Reflected Power = 4.011 mW
retlw	0x4C	; High Byte, ADRES = 481, Reflected Power = 4.397 mW
retlw	0x4C	; High Byte, ADRES = 485, Reflected Power = 4.820 mW
retlw	0x4D	; High Byte, ADRES = 489, Reflected Power = 5.283 mW
retlw	0x4D	; High Byte, ADRES = 493, Reflected Power = 5.791 mW
retlw	0x4E	; High Byte, ADRES = 497, Reflected Power = 6.348 mW
retlw	0x4F	; High Byte, ADRES = 501, Reflected Power = 6.958 mW
retlw	0x4F	; High Byte, ADRES = 505, Reflected Power = 7.626 mW
retlw	0x50	; High Byte, ADRES = 509, Reflected Power = 8.360 mW
retlw	0x50	; High Byte, ADRES = 513, Reflected Power = 9.163 mW
retlw	0x51	; High Byte, ADRES = 517, Reflected Power = 10.04 mW
retlw	0x51	; High Byte, ADRES = 521, Reflected Power = 11.01 mW
retlw	0x52	; High Byte, ADRES = 525, Reflected Power = 12.07 mW
retlw	0x52	; High Byte, ADRES = 529, Reflected Power = 13.23 mW
retlw	0x53	; High Byte, ADRES = 533, Reflected Power = 14.50 mW
retlw	0x54	; High Byte, ADRES = 537, Reflected Power = 15.89 mW
retlw	0x54	; High Byte, ADRES = 541, Reflected Power = 17.42 mW
retlw	0x54	; High Byte, ADRES = 545, Reflected Power = 19.10 mW
retlw	0x55	; High Byte, ADRES = 549, Reflected Power = 20.93 mW
retlw	0x55	; High Byte, ADRES = 553, Reflected Power = 22.94 mW
retlw	0x56	; High Byte, ADRES = 557, Reflected Power = 25.15 mW
retlw	0x57	; High Byte, ADRES = 561, Reflected Power = 27.57 mW
retlw	0x57	; High Byte, ADRES = 565, Reflected Power = 30.22 mW
retlw	0x58	; High Byte, ADRES = 569, Reflected Power = 33.12 mW
retlw	0x58	; High Byte, ADRES = 573, Reflected Power = 36.30 mW
retlw	0x59	; High Byte, ADRES = 577, Reflected Power = 39.79 mW
retlw	0x59	; High Byte, ADRES = 581, Reflected Power = 43.62 mW
retlw	0x5A	; High Byte, ADRES = 585, Reflected Power = 47.81 mW
retlw	0x5A	; High Byte, ADRES = 589, Reflected Power = 52.41 mW
retlw	0x5B	; High Byte, ADRES = 593, Reflected Power = 57.44 mW
retlw	0x5C	; High Byte, ADRES = 597, Reflected Power = 62.97 mW
retlw	0x5C	; High Byte, ADRES = 601, Reflected Power = 69.02 mW
retlw	0x5C	; High Byte, ADRES = 605, Reflected Power = 75.65 mW
retlw	0x5D	; High Byte, ADRES = 609, Reflected Power = 82.92 mW
retlw	0x5D	; High Byte, ADRES = 613, Reflected Power = 90.90 mW
retlw	0x5E	; High Byte, ADRES = 617, Reflected Power = 99.63 mW
retlw	0x5E	; High Byte, ADRES = 621, Reflected Power = 109.2 mW
retlw	0x5F	; High Byte, ADRES = 625, Reflected Power = 119.7 mW
retlw	0x60	; High Byte, ADRES = 629, Reflected Power = 131.2 mW
retlw	0x60	; High Byte, ADRES = 633, Reflected Power = 143.8 mW
retlw	0x61	; High Byte, ADRES = 637, Reflected Power = 157.7 mW
retlw	0x61	; High Byte, ADRES = 641, Reflected Power = 172.8 mW
retlw	0x62	; High Byte, ADRES = 645, Reflected Power = 189.4 mW
retlw	0x62	; High Byte, ADRES = 649, Reflected Power = 207.6 mW
retlw	0x63	; High Byte, ADRES = 653, Reflected Power = 227.6 mW
retlw	0x63	; High Byte, ADRES = 657, Reflected Power = 249.5 mW
retlw	0x64	; High Byte, ADRES = 661, Reflected Power = 273.4 mW
retlw	0x64	; High Byte, ADRES = 665, Reflected Power = 299.7 mW
retlw	0x65	; High Byte, ADRES = 669, Reflected Power = 328.5 mW
retlw	0x65	; High Byte, ADRES = 673, Reflected Power = 360.1 mW
retlw	0x66	; High Byte, ADRES = 677, Reflected Power = 394.7 mW
retlw	0x66	; High Byte, ADRES = 681, Reflected Power = 432.7 mW
retlw	0x67	; High Byte, ADRES = 685, Reflected Power = 474.3 mW
retlw	0x68	; High Byte, ADRES = 689, Reflected Power = 519.9 mW
retlw	0x68	; High Byte, ADRES = 693, Reflected Power = 569.8 mW
retlw	0x68	; High Byte, ADRES = 697, Reflected Power = 624.6 mW
retlw	0x69	; High Byte, ADRES = 701, Reflected Power = 684.6 mW
retlw	0x6A	; High Byte, ADRES = 705, Reflected Power = 750.5 mW
retlw	0x6A	; High Byte, ADRES = 709, Reflected Power = 822.6 mW
retlw	0x6B	; High Byte, ADRES = 713, Reflected Power = 901.7 mW
retlw	0x6B	; High Byte, ADRES = 717, Reflected Power = 988.3 mW
retlw	0x6C	; High Byte, ADRES = 721, Reflected Power = 1.083 W
retlw	0x6C	; High Byte, ADRES = 725, Reflected Power = 1.187 W
retlw	0x6D	; High Byte, ADRES = 729, Reflected Power = 1.302 W
retlw	0x6D	; High Byte, ADRES = 733, Reflected Power = 1.427 W
retlw	0x6E	; High Byte, ADRES = 737, Reflected Power = 1.564 W
retlw	0x6E	; High Byte, ADRES = 741, Reflected Power = 1.714 W

retlw	0x6F	; High Byte, ADRES = 745, Reflected Power = 1.879 W
retlw	0x70	; High Byte, ADRES = 749, Reflected Power = 2.060 W
retlw	0x70	; High Byte, ADRES = 753, Reflected Power = 2.258 W
retlw	0x70	; High Byte, ADRES = 757, Reflected Power = 2.475 W
retlw	0x71	; High Byte, ADRES = 761, Reflected Power = 2.712 W
retlw	0x71	; High Byte, ADRES = 765, Reflected Power = 2.973 W
retlw	0x72	; High Byte, ADRES = 769, Reflected Power = 3.259 W
retlw	0x73	; High Byte, ADRES = 773, Reflected Power = 3.572 W
retlw	0x73	; High Byte, ADRES = 777, Reflected Power = 3.916 W
retlw	0x74	; High Byte, ADRES = 781, Reflected Power = 4.292 W
retlw	0x74	; High Byte, ADRES = 785, Reflected Power = 4.705 W
retlw	0x75	; High Byte, ADRES = 789, Reflected Power = 5.157 W
retlw	0x75	; High Byte, ADRES = 793, Reflected Power = 5.653 W
retlw	0x76	; High Byte, ADRES = 797, Reflected Power = 6.196 W
retlw	0x76	; High Byte, ADRES = 801, Reflected Power = 6.791 W
retlw	0x77	; High Byte, ADRES = 805, Reflected Power = 7.444 W
retlw	0x78	; High Byte, ADRES = 809, Reflected Power = 8.160 W
retlw	0x78	; High Byte, ADRES = 813, Reflected Power = 8.944 W
retlw	0x78	; High Byte, ADRES = 817, Reflected Power = 9.804 W
retlw	0x79	; High Byte, ADRES = 821, Reflected Power = 10.75 W
retlw	0x79	; High Byte, ADRES = 825, Reflected Power = 11.78 W
retlw	0x7A	; High Byte, ADRES = 829, Reflected Power = 12.91 W
retlw	0x7B	; High Byte, ADRES = 833, Reflected Power = 14.15 W
retlw	0x7B	; High Byte, ADRES = 837, Reflected Power = 15.51 W
retlw	0x7C	; High Byte, ADRES = 841, Reflected Power = 17.00 W
retlw	0x7C	; High Byte, ADRES = 845, Reflected Power = 18.64 W
retlw	0x7D	; High Byte, ADRES = 849, Reflected Power = 20.43 W
retlw	0x7D	; High Byte, ADRES = 853, Reflected Power = 22.39 W
retlw	0x7E	; High Byte, ADRES = 857, Reflected Power = 24.55 W
retlw	0x7E	; High Byte, ADRES = 861, Reflected Power = 26.91 W
retlw	0x7F	; High Byte, ADRES = 865, Reflected Power = 29.49 W
retlw	0x80	; High Byte, ADRES = 869, Reflected Power = 32.33 W
retlw	0x80	; High Byte, ADRES = 873, Reflected Power = 35.44 W
retlw	0x80	; High Byte, ADRES = 877, Reflected Power = 38.84 W
retlw	0x81	; High Byte, ADRES = 881, Reflected Power = 42.58 W
retlw	0x81	; High Byte, ADRES = 885, Reflected Power = 46.67 W
retlw	0x82	; High Byte, ADRES = 889, Reflected Power = 51.15 W
retlw	0x83	; High Byte, ADRES = 893, Reflected Power = 56.07 W
retlw	0x83	; High Byte, ADRES = 897, Reflected Power = 61.46 W
retlw	0x84	; High Byte, ADRES = 901, Reflected Power = 67.37 W
retlw	0x84	; High Byte, ADRES = 905, Reflected Power = 73.85 W
retlw	0x85	; High Byte, ADRES = 909, Reflected Power = 80.94 W
retlw	0x85	; High Byte, ADRES = 913, Reflected Power = 88.72 W
retlw	0x86	; High Byte, ADRES = 917, Reflected Power = 97.25 W
retlw	0x86	; High Byte, ADRES = 921, Reflected Power = 106.6 W
retlw	0x87	; High Byte, ADRES = 925, Reflected Power = 116.8 W
retlw	0x88	; High Byte, ADRES = 929, Reflected Power = 128.1 W
retlw	0x88	; High Byte, ADRES = 933, Reflected Power = 140.4 W
retlw	0x88	; High Byte, ADRES = 937, Reflected Power = 153.9 W
retlw	0x89	; High Byte, ADRES = 941, Reflected Power = 168.7 W
retlw	0x89	; High Byte, ADRES = 945, Reflected Power = 184.9 W
retlw	0x8A	; High Byte, ADRES = 949, Reflected Power = 202.7 W
retlw	0x8A	; High Byte, ADRES = 953, Reflected Power = 222.1 W
retlw	0x8B	; High Byte, ADRES = 957, Reflected Power = 243.5 W
retlw	0x8C	; High Byte, ADRES = 961, Reflected Power = 266.9 W
retlw	0x8C	; High Byte, ADRES = 965, Reflected Power = 292.6 W
retlw	0x8D	; High Byte, ADRES = 969, Reflected Power = 320.7 W
retlw	0x8D	; High Byte, ADRES = 973, Reflected Power = 351.5 W
retlw	0x8E	; High Byte, ADRES = 977, Reflected Power = 385.3 W
retlw	0x8E	; High Byte, ADRES = 981, Reflected Power = 422.3 W
retlw	0x8F	; High Byte, ADRES = 985, Reflected Power = 462.9 W
retlw	0x8F	; High Byte, ADRES = 989, Reflected Power = 507.4 W
retlw	0x90	; High Byte, ADRES = 993, Reflected Power = 556.2 W
retlw	0x90	; High Byte, ADRES = 997, Reflected Power = 609.7 W
retlw	0x91	; High Byte, ADRES = 1001, Reflected Power = 668.3 W
retlw	0x91	; High Byte, ADRES = 1005, Reflected Power = 732.5 W
retlw	0x92	; High Byte, ADRES = 1009, Reflected Power = 802.9 W
retlw	0x92	; High Byte, ADRES = 1013, Reflected Power = 880.1 W
retlw	0x93	; High Byte, ADRES = 1017, Reflected Power = 964.7 W
retlw	0x94	; High Byte, ADRES = 1021, Reflected Power = 1,057.5 W

RHighByteTable2

retlw	0x0C	; High Byte, ADRES = 2, Reflected Power = 74.12 nW
retlw	0x0D	; High Byte, ADRES = 6, Reflected Power = 81.24 nW
retlw	0x0D	; High Byte, ADRES = 10, Reflected Power = 89.05 nW
retlw	0x0E	; High Byte, ADRES = 14, Reflected Power = 97.61 nW
retlw	0x0F	; High Byte, ADRES = 18, Reflected Power = 107.0 nW
retlw	0x0F	; High Byte, ADRES = 22, Reflected Power = 117.3 nW
retlw	0x10	; High Byte, ADRES = 26, Reflected Power = 128.6 nW
retlw	0x10	; High Byte, ADRES = 30, Reflected Power = 140.9 nW
retlw	0x11	; High Byte, ADRES = 34, Reflected Power = 154.5 nW

retlw	0x11	; High Byte, ADRES = 38, Reflected Power = 169.3 nW
retlw	0x12	; High Byte, ADRES = 42, Reflected Power = 185.6 nW
retlw	0x12	; High Byte, ADRES = 46, Reflected Power = 203.4 nW
retlw	0x13	; High Byte, ADRES = 50, Reflected Power = 223.0 nW
retlw	0x14	; High Byte, ADRES = 54, Reflected Power = 244.4 nW
retlw	0x14	; High Byte, ADRES = 58, Reflected Power = 267.9 nW
retlw	0x14	; High Byte, ADRES = 62, Reflected Power = 293.7 nW
retlw	0x15	; High Byte, ADRES = 66, Reflected Power = 321.9 nW
retlw	0x15	; High Byte, ADRES = 70, Reflected Power = 352.8 nW
retlw	0x16	; High Byte, ADRES = 74, Reflected Power = 386.7 nW
retlw	0x17	; High Byte, ADRES = 78, Reflected Power = 423.9 nW
retlw	0x17	; High Byte, ADRES = 82, Reflected Power = 464.7 nW
retlw	0x18	; High Byte, ADRES = 86, Reflected Power = 509.3 nW
retlw	0x18	; High Byte, ADRES = 90, Reflected Power = 558.3 nW
retlw	0x19	; High Byte, ADRES = 94, Reflected Power = 611.9 nW
retlw	0x19	; High Byte, ADRES = 98, Reflected Power = 670.8 nW
retlw	0x1A	; High Byte, ADRES = 102, Reflected Power = 735.2 nW
retlw	0x1A	; High Byte, ADRES = 106, Reflected Power = 805.9 nW
retlw	0x1B	; High Byte, ADRES = 110, Reflected Power = 883.4 nW
retlw	0x1C	; High Byte, ADRES = 114, Reflected Power = 968.3 nW
retlw	0x1C	; High Byte, ADRES = 118, Reflected Power = 1.061 uW
retlw	0x1C	; High Byte, ADRES = 122, Reflected Power = 1.163 uW
retlw	0x1D	; High Byte, ADRES = 126, Reflected Power = 1.275 uW
retlw	0x1D	; High Byte, ADRES = 130, Reflected Power = 1.398 uW
retlw	0x1E	; High Byte, ADRES = 134, Reflected Power = 1.532 uW
retlw	0x1F	; High Byte, ADRES = 138, Reflected Power = 1.679 uW
retlw	0x1F	; High Byte, ADRES = 142, Reflected Power = 1.841 uW
retlw	0x20	; High Byte, ADRES = 146, Reflected Power = 2.018 uW
retlw	0x20	; High Byte, ADRES = 150, Reflected Power = 2.212 uW
retlw	0x21	; High Byte, ADRES = 154, Reflected Power = 2.424 uW
retlw	0x21	; High Byte, ADRES = 158, Reflected Power = 2.657 uW
retlw	0x22	; High Byte, ADRES = 162, Reflected Power = 2.913 uW
retlw	0x22	; High Byte, ADRES = 166, Reflected Power = 3.193 uW
retlw	0x23	; High Byte, ADRES = 170, Reflected Power = 3.500 uW
retlw	0x24	; High Byte, ADRES = 174, Reflected Power = 3.836 uW
retlw	0x24	; High Byte, ADRES = 178, Reflected Power = 4.205 uW
retlw	0x24	; High Byte, ADRES = 182, Reflected Power = 4.609 uW
retlw	0x25	; High Byte, ADRES = 186, Reflected Power = 5.052 uW
retlw	0x25	; High Byte, ADRES = 190, Reflected Power = 5.538 uW
retlw	0x26	; High Byte, ADRES = 194, Reflected Power = 6.070 uW
retlw	0x26	; High Byte, ADRES = 198, Reflected Power = 6.654 uW
retlw	0x27	; High Byte, ADRES = 202, Reflected Power = 7.293 uW
retlw	0x28	; High Byte, ADRES = 206, Reflected Power = 7.994 uW
retlw	0x28	; High Byte, ADRES = 210, Reflected Power = 8.763 uW
retlw	0x29	; High Byte, ADRES = 214, Reflected Power = 9.605 uW
retlw	0x29	; High Byte, ADRES = 218, Reflected Power = 10.53 uW
retlw	0x2A	; High Byte, ADRES = 222, Reflected Power = 11.54 uW
retlw	0x2A	; High Byte, ADRES = 226, Reflected Power = 12.65 uW
retlw	0x2B	; High Byte, ADRES = 230, Reflected Power = 13.87 uW
retlw	0x2B	; High Byte, ADRES = 234, Reflected Power = 15.20 uW
retlw	0x2C	; High Byte, ADRES = 238, Reflected Power = 16.66 uW
retlw	0x2C	; High Byte, ADRES = 242, Reflected Power = 18.26 uW
retlw	0x2D	; High Byte, ADRES = 246, Reflected Power = 20.02 uW
retlw	0x2D	; High Byte, ADRES = 250, Reflected Power = 21.94 uW
retlw	0x2E	; High Byte, ADRES = 254, Reflected Power = 24.05 uW
retlw	0x2E	; High Byte, ADRES = 258, Reflected Power = 26.36 uW
retlw	0x2F	; High Byte, ADRES = 262, Reflected Power = 28.90 uW
retlw	0x30	; High Byte, ADRES = 266, Reflected Power = 31.67 uW
retlw	0x30	; High Byte, ADRES = 270, Reflected Power = 34.72 uW
retlw	0x30	; High Byte, ADRES = 274, Reflected Power = 38.05 uW
retlw	0x31	; High Byte, ADRES = 278, Reflected Power = 41.71 uW
retlw	0x31	; High Byte, ADRES = 282, Reflected Power = 45.72 uW
retlw	0x32	; High Byte, ADRES = 286, Reflected Power = 50.12 uW
retlw	0x33	; High Byte, ADRES = 290, Reflected Power = 54.94 uW
retlw	0x33	; High Byte, ADRES = 294, Reflected Power = 60.22 uW
retlw	0x34	; High Byte, ADRES = 298, Reflected Power = 66.00 uW
retlw	0x34	; High Byte, ADRES = 302, Reflected Power = 72.35 uW
retlw	0x35	; High Byte, ADRES = 306, Reflected Power = 79.30 uW
retlw	0x35	; High Byte, ADRES = 310, Reflected Power = 86.93 uW
retlw	0x36	; High Byte, ADRES = 314, Reflected Power = 95.28 uW
retlw	0x36	; High Byte, ADRES = 318, Reflected Power = 104.4 uW
retlw	0x37	; High Byte, ADRES = 322, Reflected Power = 114.5 uW
retlw	0x38	; High Byte, ADRES = 326, Reflected Power = 125.5 uW
retlw	0x38	; High Byte, ADRES = 330, Reflected Power = 137.5 uW
retlw	0x38	; High Byte, ADRES = 334, Reflected Power = 150.8 uW
retlw	0x39	; High Byte, ADRES = 338, Reflected Power = 165.3 uW
retlw	0x39	; High Byte, ADRES = 342, Reflected Power = 181.1 uW
retlw	0x3A	; High Byte, ADRES = 346, Reflected Power = 198.6 uW
retlw	0x3B	; High Byte, ADRES = 350, Reflected Power = 217.6 uW
retlw	0x3B	; High Byte, ADRES = 354, Reflected Power = 238.6 uW
retlw	0x3C	; High Byte, ADRES = 358, Reflected Power = 261.5 uW

retlw	0x3C	; High Byte, ADRES = 362, Reflected Power = 286.6 uW
retlw	0x3D	; High Byte, ADRES = 366, Reflected Power = 314.2 uW
retlw	0x3D	; High Byte, ADRES = 370, Reflected Power = 344.4 uW
retlw	0x3E	; High Byte, ADRES = 374, Reflected Power = 377.5 uW
retlw	0x3E	; High Byte, ADRES = 378, Reflected Power = 413.8 uW
retlw	0x3F	; High Byte, ADRES = 382, Reflected Power = 453.6 uW
retlw	0x40	; High Byte, ADRES = 386, Reflected Power = 497.2 uW
retlw	0x40	; High Byte, ADRES = 390, Reflected Power = 544.9 uW
retlw	0x40	; High Byte, ADRES = 394, Reflected Power = 597.3 uW
retlw	0x41	; High Byte, ADRES = 398, Reflected Power = 654.7 uW
retlw	0x41	; High Byte, ADRES = 402, Reflected Power = 717.7 uW
retlw	0x42	; High Byte, ADRES = 406, Reflected Power = 786.7 uW
retlw	0x43	; High Byte, ADRES = 410, Reflected Power = 862.3 uW
retlw	0x43	; High Byte, ADRES = 414, Reflected Power = 945.2 uW
retlw	0x44	; High Byte, ADRES = 418, Reflected Power = 1.036 mW
retlw	0x44	; High Byte, ADRES = 422, Reflected Power = 1.136 mW
retlw	0x45	; High Byte, ADRES = 426, Reflected Power = 1.245 mW
retlw	0x45	; High Byte, ADRES = 430, Reflected Power = 1.364 mW
retlw	0x46	; High Byte, ADRES = 434, Reflected Power = 1.496 mW
retlw	0x46	; High Byte, ADRES = 438, Reflected Power = 1.639 mW
retlw	0x47	; High Byte, ADRES = 442, Reflected Power = 1.797 mW
retlw	0x48	; High Byte, ADRES = 446, Reflected Power = 1.970 mW
retlw	0x48	; High Byte, ADRES = 450, Reflected Power = 2.159 mW
retlw	0x48	; High Byte, ADRES = 454, Reflected Power = 2.367 mW
retlw	0x49	; High Byte, ADRES = 458, Reflected Power = 2.594 mW
retlw	0x49	; High Byte, ADRES = 462, Reflected Power = 2.843 mW
retlw	0x4A	; High Byte, ADRES = 466, Reflected Power = 3.117 mW
retlw	0x4A	; High Byte, ADRES = 470, Reflected Power = 3.416 mW
retlw	0x4B	; High Byte, ADRES = 474, Reflected Power = 3.745 mW
retlw	0x4C	; High Byte, ADRES = 478, Reflected Power = 4.105 mW
retlw	0x4C	; High Byte, ADRES = 482, Reflected Power = 4.499 mW
retlw	0x4D	; High Byte, ADRES = 486, Reflected Power = 4.932 mW
retlw	0x4D	; High Byte, ADRES = 490, Reflected Power = 5.406 mW
retlw	0x4E	; High Byte, ADRES = 494, Reflected Power = 5.925 mW
retlw	0x4E	; High Byte, ADRES = 498, Reflected Power = 6.495 mW
retlw	0x4F	; High Byte, ADRES = 502, Reflected Power = 7.119 mW
retlw	0x4F	; High Byte, ADRES = 506, Reflected Power = 7.804 mW
retlw	0x50	; High Byte, ADRES = 510, Reflected Power = 8.554 mW
retlw	0x50	; High Byte, ADRES = 514, Reflected Power = 9.376 mW
retlw	0x51	; High Byte, ADRES = 518, Reflected Power = 10.28 mW
retlw	0x51	; High Byte, ADRES = 522, Reflected Power = 11.26 mW
retlw	0x52	; High Byte, ADRES = 526, Reflected Power = 12.35 mW
retlw	0x52	; High Byte, ADRES = 530, Reflected Power = 13.53 mW
retlw	0x53	; High Byte, ADRES = 534, Reflected Power = 14.84 mW
retlw	0x54	; High Byte, ADRES = 538, Reflected Power = 16.26 mW
retlw	0x54	; High Byte, ADRES = 542, Reflected Power = 17.83 mW
retlw	0x55	; High Byte, ADRES = 546, Reflected Power = 19.54 mW
retlw	0x55	; High Byte, ADRES = 550, Reflected Power = 21.42 mW
retlw	0x56	; High Byte, ADRES = 554, Reflected Power = 23.48 mW
retlw	0x56	; High Byte, ADRES = 558, Reflected Power = 25.73 mW
retlw	0x57	; High Byte, ADRES = 562, Reflected Power = 28.21 mW
retlw	0x57	; High Byte, ADRES = 566, Reflected Power = 30.92 mW
retlw	0x58	; High Byte, ADRES = 570, Reflected Power = 33.89 mW
retlw	0x58	; High Byte, ADRES = 574, Reflected Power = 37.15 mW
retlw	0x59	; High Byte, ADRES = 578, Reflected Power = 40.72 mW
retlw	0x59	; High Byte, ADRES = 582, Reflected Power = 44.63 mW
retlw	0x5A	; High Byte, ADRES = 586, Reflected Power = 48.92 mW
retlw	0x5A	; High Byte, ADRES = 590, Reflected Power = 53.62 mW
retlw	0x5B	; High Byte, ADRES = 594, Reflected Power = 58.78 mW
retlw	0x5C	; High Byte, ADRES = 598, Reflected Power = 64.43 mW
retlw	0x5C	; High Byte, ADRES = 602, Reflected Power = 70.62 mW
retlw	0x5C	; High Byte, ADRES = 606, Reflected Power = 77.41 mW
retlw	0x5D	; High Byte, ADRES = 610, Reflected Power = 84.85 mW
retlw	0x5D	; High Byte, ADRES = 614, Reflected Power = 93.01 mW
retlw	0x5E	; High Byte, ADRES = 618, Reflected Power = 101.9 mW
retlw	0x5F	; High Byte, ADRES = 622, Reflected Power = 111.7 mW
retlw	0x5F	; High Byte, ADRES = 626, Reflected Power = 122.5 mW
retlw	0x60	; High Byte, ADRES = 630, Reflected Power = 134.3 mW
retlw	0x60	; High Byte, ADRES = 634, Reflected Power = 147.2 mW
retlw	0x61	; High Byte, ADRES = 638, Reflected Power = 161.3 mW
retlw	0x61	; High Byte, ADRES = 642, Reflected Power = 176.8 mW
retlw	0x62	; High Byte, ADRES = 646, Reflected Power = 193.8 mW
retlw	0x62	; High Byte, ADRES = 650, Reflected Power = 212.4 mW
retlw	0x63	; High Byte, ADRES = 654, Reflected Power = 232.9 mW
retlw	0x64	; High Byte, ADRES = 658, Reflected Power = 255.3 mW
retlw	0x64	; High Byte, ADRES = 662, Reflected Power = 279.8 mW
retlw	0x64	; High Byte, ADRES = 666, Reflected Power = 306.7 mW
retlw	0x65	; High Byte, ADRES = 670, Reflected Power = 336.2 mW
retlw	0x65	; High Byte, ADRES = 674, Reflected Power = 368.5 mW
retlw	0x66	; High Byte, ADRES = 678, Reflected Power = 403.9 mW
retlw	0x67	; High Byte, ADRES = 682, Reflected Power = 442.7 mW

retlw	0x67	; High Byte, ADRES = 686, Reflected Power = 485.3 mW
retlw	0x68	; High Byte, ADRES = 690, Reflected Power = 531.9 mW
retlw	0x68	; High Byte, ADRES = 694, Reflected Power = 583.1 mW
retlw	0x69	; High Byte, ADRES = 698, Reflected Power = 639.1 mW
retlw	0x69	; High Byte, ADRES = 702, Reflected Power = 700.5 mW
retlw	0x6A	; High Byte, ADRES = 706, Reflected Power = 767.9 mW
retlw	0x6A	; High Byte, ADRES = 710, Reflected Power = 841.7 mW
retlw	0x6B	; High Byte, ADRES = 714, Reflected Power = 922.6 mW
retlw	0x6C	; High Byte, ADRES = 718, Reflected Power = 1.011 W
retlw	0x6C	; High Byte, ADRES = 722, Reflected Power = 1.108 W
retlw	0x6C	; High Byte, ADRES = 726, Reflected Power = 1.215 W
retlw	0x6D	; High Byte, ADRES = 730, Reflected Power = 1.332 W
retlw	0x6D	; High Byte, ADRES = 734, Reflected Power = 1.460 W
retlw	0x6E	; High Byte, ADRES = 738, Reflected Power = 1.600 W
retlw	0x6F	; High Byte, ADRES = 742, Reflected Power = 1.754 W
retlw	0x6F	; High Byte, ADRES = 746, Reflected Power = 1.923 W
retlw	0x70	; High Byte, ADRES = 750, Reflected Power = 2.107 W
retlw	0x70	; High Byte, ADRES = 754, Reflected Power = 2.310 W
retlw	0x71	; High Byte, ADRES = 758, Reflected Power = 2.532 W
retlw	0x71	; High Byte, ADRES = 762, Reflected Power = 2.775 W
retlw	0x72	; High Byte, ADRES = 766, Reflected Power = 3.042 W
retlw	0x72	; High Byte, ADRES = 770, Reflected Power = 3.335 W
retlw	0x73	; High Byte, ADRES = 774, Reflected Power = 3.655 W
retlw	0x74	; High Byte, ADRES = 778, Reflected Power = 4.007 W
retlw	0x74	; High Byte, ADRES = 782, Reflected Power = 4.392 W
retlw	0x74	; High Byte, ADRES = 786, Reflected Power = 4.814 W
retlw	0x75	; High Byte, ADRES = 790, Reflected Power = 5.277 W
retlw	0x75	; High Byte, ADRES = 794, Reflected Power = 5.784 W
retlw	0x76	; High Byte, ADRES = 798, Reflected Power = 6.340 W
retlw	0x76	; High Byte, ADRES = 802, Reflected Power = 6.949 W
retlw	0x77	; High Byte, ADRES = 806, Reflected Power = 7.617 W
retlw	0x78	; High Byte, ADRES = 810, Reflected Power = 8.349 W
retlw	0x78	; High Byte, ADRES = 814, Reflected Power = 9.152 W
retlw	0x79	; High Byte, ADRES = 818, Reflected Power = 10.03 W
retlw	0x79	; High Byte, ADRES = 822, Reflected Power = 11.00 W
retlw	0x7A	; High Byte, ADRES = 826, Reflected Power = 12.05 W
retlw	0x7A	; High Byte, ADRES = 830, Reflected Power = 13.21 W
retlw	0x7B	; High Byte, ADRES = 834, Reflected Power = 14.48 W
retlw	0x7B	; High Byte, ADRES = 838, Reflected Power = 15.87 W
retlw	0x7C	; High Byte, ADRES = 842, Reflected Power = 17.40 W
retlw	0x7C	; High Byte, ADRES = 846, Reflected Power = 19.07 W
retlw	0x7D	; High Byte, ADRES = 850, Reflected Power = 20.90 W
retlw	0x7D	; High Byte, ADRES = 854, Reflected Power = 22.91 W
retlw	0x7E	; High Byte, ADRES = 858, Reflected Power = 25.12 W
retlw	0x7E	; High Byte, ADRES = 862, Reflected Power = 27.53 W
retlw	0x7F	; High Byte, ADRES = 866, Reflected Power = 30.18 W
retlw	0x80	; High Byte, ADRES = 870, Reflected Power = 33.08 W
retlw	0x80	; High Byte, ADRES = 874, Reflected Power = 36.26 W
retlw	0x80	; High Byte, ADRES = 878, Reflected Power = 39.74 W
retlw	0x81	; High Byte, ADRES = 882, Reflected Power = 43.56 W
retlw	0x81	; High Byte, ADRES = 886, Reflected Power = 47.75 W
retlw	0x82	; High Byte, ADRES = 890, Reflected Power = 52.34 W
retlw	0x83	; High Byte, ADRES = 894, Reflected Power = 57.37 W
retlw	0x83	; High Byte, ADRES = 898, Reflected Power = 62.89 W
retlw	0x84	; High Byte, ADRES = 902, Reflected Power = 68.93 W
retlw	0x84	; High Byte, ADRES = 906, Reflected Power = 75.56 W
retlw	0x85	; High Byte, ADRES = 910, Reflected Power = 82.82 W
retlw	0x85	; High Byte, ADRES = 914, Reflected Power = 90.78 W
retlw	0x86	; High Byte, ADRES = 918, Reflected Power = 99.51 W
retlw	0x86	; High Byte, ADRES = 922, Reflected Power = 109.1 W
retlw	0x87	; High Byte, ADRES = 926, Reflected Power = 119.6 W
retlw	0x88	; High Byte, ADRES = 930, Reflected Power = 131.1 W
retlw	0x88	; High Byte, ADRES = 934, Reflected Power = 143.7 W
retlw	0x88	; High Byte, ADRES = 938, Reflected Power = 157.5 W
retlw	0x89	; High Byte, ADRES = 942, Reflected Power = 172.6 W
retlw	0x89	; High Byte, ADRES = 946, Reflected Power = 189.2 W
retlw	0x8A	; High Byte, ADRES = 950, Reflected Power = 207.4 W
retlw	0x8B	; High Byte, ADRES = 954, Reflected Power = 227.3 W
retlw	0x8B	; High Byte, ADRES = 958, Reflected Power = 249.2 W
retlw	0x8C	; High Byte, ADRES = 962, Reflected Power = 273.1 W
retlw	0x8C	; High Byte, ADRES = 966, Reflected Power = 299.4 W
retlw	0x8D	; High Byte, ADRES = 970, Reflected Power = 328.1 W
retlw	0x8D	; High Byte, ADRES = 974, Reflected Power = 359.7 W
retlw	0x8E	; High Byte, ADRES = 978, Reflected Power = 394.2 W
retlw	0x8E	; High Byte, ADRES = 982, Reflected Power = 432.1 W
retlw	0x8F	; High Byte, ADRES = 986, Reflected Power = 473.7 W
retlw	0x90	; High Byte, ADRES = 990, Reflected Power = 519.2 W
retlw	0x90	; High Byte, ADRES = 994, Reflected Power = 569.1 W
retlw	0x90	; High Byte, ADRES = 998, Reflected Power = 623.8 W
retlw	0x91	; High Byte, ADRES = 1002, Reflected Power = 683.8 W
retlw	0x91	; High Byte, ADRES = 1006, Reflected Power = 749.5 W

retlw	0x92	; High Byte, ADRES = 1010, Reflected Power = 821.6 W
retlw	0x93	; High Byte, ADRES = 1014, Reflected Power = 900.6 W
retlw	0x93	; High Byte, ADRES = 1018, Reflected Power = 987.1 W
retlw	0x94	; High Byte, ADRES = 1022, Reflected Power = 1,082.0 W

RHighByteTable3

retlw	0x0D	; High Byte, ADRES = 3, Reflected Power = 75.84 nW
retlw	0x0D	; High Byte, ADRES = 7, Reflected Power = 83.13 nW
retlw	0x0E	; High Byte, ADRES = 11, Reflected Power = 91.12 nW
retlw	0x0E	; High Byte, ADRES = 15, Reflected Power = 99.88 nW
retlw	0x0F	; High Byte, ADRES = 19, Reflected Power = 109.5 nW
retlw	0x10	; High Byte, ADRES = 23, Reflected Power = 120.0 nW
retlw	0x10	; High Byte, ADRES = 27, Reflected Power = 131.5 nW
retlw	0x10	; High Byte, ADRES = 31, Reflected Power = 144.2 nW
retlw	0x11	; High Byte, ADRES = 35, Reflected Power = 158.0 nW
retlw	0x11	; High Byte, ADRES = 39, Reflected Power = 173.2 nW
retlw	0x12	; High Byte, ADRES = 43, Reflected Power = 189.9 nW
retlw	0x12	; High Byte, ADRES = 47, Reflected Power = 208.1 nW
retlw	0x13	; High Byte, ADRES = 51, Reflected Power = 228.1 nW
retlw	0x14	; High Byte, ADRES = 55, Reflected Power = 250.1 nW
retlw	0x14	; High Byte, ADRES = 59, Reflected Power = 274.1 nW
retlw	0x15	; High Byte, ADRES = 63, Reflected Power = 300.5 nW
retlw	0x15	; High Byte, ADRES = 67, Reflected Power = 329.3 nW
retlw	0x16	; High Byte, ADRES = 71, Reflected Power = 361.0 nW
retlw	0x16	; High Byte, ADRES = 75, Reflected Power = 395.7 nW
retlw	0x17	; High Byte, ADRES = 79, Reflected Power = 433.7 nW
retlw	0x17	; High Byte, ADRES = 83, Reflected Power = 475.4 nW
retlw	0x18	; High Byte, ADRES = 87, Reflected Power = 521.1 nW
retlw	0x18	; High Byte, ADRES = 91, Reflected Power = 571.2 nW
retlw	0x19	; High Byte, ADRES = 95, Reflected Power = 626.1 nW
retlw	0x19	; High Byte, ADRES = 99, Reflected Power = 686.3 nW
retlw	0x1A	; High Byte, ADRES = 103, Reflected Power = 752.3 nW
retlw	0x1A	; High Byte, ADRES = 107, Reflected Power = 824.6 nW
retlw	0x1B	; High Byte, ADRES = 111, Reflected Power = 903.9 nW
retlw	0x1C	; High Byte, ADRES = 115, Reflected Power = 990.8 nW
retlw	0x1C	; High Byte, ADRES = 119, Reflected Power = 1.086 uW
retlw	0x1C	; High Byte, ADRES = 123, Reflected Power = 1.190 uW
retlw	0x1D	; High Byte, ADRES = 127, Reflected Power = 1.305 uW
retlw	0x1E	; High Byte, ADRES = 131, Reflected Power = 1.430 uW
retlw	0x1E	; High Byte, ADRES = 135, Reflected Power = 1.568 uW
retlw	0x1F	; High Byte, ADRES = 139, Reflected Power = 1.718 uW
retlw	0x1F	; High Byte, ADRES = 143, Reflected Power = 1.884 uW
retlw	0x20	; High Byte, ADRES = 147, Reflected Power = 2.065 uW
retlw	0x20	; High Byte, ADRES = 151, Reflected Power = 2.263 uW
retlw	0x21	; High Byte, ADRES = 155, Reflected Power = 2.481 uW
retlw	0x21	; High Byte, ADRES = 159, Reflected Power = 2.719 uW
retlw	0x22	; High Byte, ADRES = 163, Reflected Power = 2.981 uW
retlw	0x22	; High Byte, ADRES = 167, Reflected Power = 3.267 uW
retlw	0x23	; High Byte, ADRES = 171, Reflected Power = 3.581 uW
retlw	0x24	; High Byte, ADRES = 175, Reflected Power = 3.925 uW
retlw	0x24	; High Byte, ADRES = 179, Reflected Power = 4.303 uW
retlw	0x24	; High Byte, ADRES = 183, Reflected Power = 4.716 uW
retlw	0x25	; High Byte, ADRES = 187, Reflected Power = 5.170 uW
retlw	0x25	; High Byte, ADRES = 191, Reflected Power = 5.667 uW
retlw	0x26	; High Byte, ADRES = 195, Reflected Power = 6.211 uW
retlw	0x27	; High Byte, ADRES = 199, Reflected Power = 6.808 uW
retlw	0x27	; High Byte, ADRES = 203, Reflected Power = 7.463 uW
retlw	0x28	; High Byte, ADRES = 207, Reflected Power = 8.180 uW
retlw	0x28	; High Byte, ADRES = 211, Reflected Power = 8.966 uW
retlw	0x29	; High Byte, ADRES = 215, Reflected Power = 9.828 uW
retlw	0x29	; High Byte, ADRES = 219, Reflected Power = 10.77 uW
retlw	0x2A	; High Byte, ADRES = 223, Reflected Power = 11.81 uW
retlw	0x2A	; High Byte, ADRES = 227, Reflected Power = 12.94 uW
retlw	0x2B	; High Byte, ADRES = 231, Reflected Power = 14.19 uW
retlw	0x2C	; High Byte, ADRES = 235, Reflected Power = 15.55 uW
retlw	0x2C	; High Byte, ADRES = 239, Reflected Power = 17.05 uW
retlw	0x2C	; High Byte, ADRES = 243, Reflected Power = 18.69 uW
retlw	0x2D	; High Byte, ADRES = 247, Reflected Power = 20.48 uW
retlw	0x2D	; High Byte, ADRES = 251, Reflected Power = 22.45 uW
retlw	0x2E	; High Byte, ADRES = 255, Reflected Power = 24.61 uW
retlw	0x2F	; High Byte, ADRES = 259, Reflected Power = 26.97 uW
retlw	0x2F	; High Byte, ADRES = 263, Reflected Power = 29.57 uW
retlw	0x30	; High Byte, ADRES = 267, Reflected Power = 32.41 uW
retlw	0x30	; High Byte, ADRES = 271, Reflected Power = 35.52 uW
retlw	0x31	; High Byte, ADRES = 275, Reflected Power = 38.94 uW
retlw	0x31	; High Byte, ADRES = 279, Reflected Power = 42.68 uW
retlw	0x32	; High Byte, ADRES = 283, Reflected Power = 46.78 uW
retlw	0x32	; High Byte, ADRES = 287, Reflected Power = 51.28 uW
retlw	0x33	; High Byte, ADRES = 291, Reflected Power = 56.21 uW
retlw	0x34	; High Byte, ADRES = 295, Reflected Power = 61.61 uW
retlw	0x34	; High Byte, ADRES = 299, Reflected Power = 67.54 uW

retlw	0x34	; High Byte, ADRES = 303, Reflected Power = 74.03 uW
retlw	0x35	; High Byte, ADRES = 307, Reflected Power = 81.14 uW
retlw	0x35	; High Byte, ADRES = 311, Reflected Power = 88.94 uW
retlw	0x36	; High Byte, ADRES = 315, Reflected Power = 97.49 uW
retlw	0x37	; High Byte, ADRES = 319, Reflected Power = 106.9 uW
retlw	0x37	; High Byte, ADRES = 323, Reflected Power = 117.1 uW
retlw	0x38	; High Byte, ADRES = 327, Reflected Power = 128.4 uW
retlw	0x38	; High Byte, ADRES = 331, Reflected Power = 140.7 uW
retlw	0x39	; High Byte, ADRES = 335, Reflected Power = 154.3 uW
retlw	0x39	; High Byte, ADRES = 339, Reflected Power = 169.1 uW
retlw	0x3A	; High Byte, ADRES = 343, Reflected Power = 185.4 uW
retlw	0x3A	; High Byte, ADRES = 347, Reflected Power = 203.2 uW
retlw	0x3B	; High Byte, ADRES = 351, Reflected Power = 222.7 uW
retlw	0x3E	; High Byte, ADRES = 355, Reflected Power = 244.1 uW
retlw	0x3C	; High Byte, ADRES = 359, Reflected Power = 267.6 uW
retlw	0x3C	; High Byte, ADRES = 363, Reflected Power = 293.3 uW
retlw	0x3D	; High Byte, ADRES = 367, Reflected Power = 321.5 uW
retlw	0x3D	; High Byte, ADRES = 371, Reflected Power = 352.4 uW
retlw	0x3E	; High Byte, ADRES = 375, Reflected Power = 386.3 uW
retlw	0x3E	; High Byte, ADRES = 379, Reflected Power = 423.4 uW
retlw	0x3F	; High Byte, ADRES = 383, Reflected Power = 464.1 uW
retlw	0x40	; High Byte, ADRES = 387, Reflected Power = 508.7 uW
retlw	0x40	; High Byte, ADRES = 391, Reflected Power = 557.6 uW
retlw	0x41	; High Byte, ADRES = 395, Reflected Power = 611.2 uW
retlw	0x41	; High Byte, ADRES = 399, Reflected Power = 669.9 uW
retlw	0x42	; High Byte, ADRES = 403, Reflected Power = 734.3 uW
retlw	0x42	; High Byte, ADRES = 407, Reflected Power = 804.9 uW
retlw	0x43	; High Byte, ADRES = 411, Reflected Power = 882.3 uW
retlw	0x43	; High Byte, ADRES = 415, Reflected Power = 967.1 uW
retlw	0x44	; High Byte, ADRES = 419, Reflected Power = 1.060 mW
retlw	0x44	; High Byte, ADRES = 423, Reflected Power = 1.162 mW
retlw	0x45	; High Byte, ADRES = 427, Reflected Power = 1.274 mW
retlw	0x45	; High Byte, ADRES = 431, Reflected Power = 1.396 mW
retlw	0x46	; High Byte, ADRES = 435, Reflected Power = 1.530 mW
retlw	0x46	; High Byte, ADRES = 439, Reflected Power = 1.677 mW
retlw	0x47	; High Byte, ADRES = 443, Reflected Power = 1.839 mW
retlw	0x48	; High Byte, ADRES = 447, Reflected Power = 2.015 mW
retlw	0x48	; High Byte, ADRES = 451, Reflected Power = 2.209 mW
retlw	0x48	; High Byte, ADRES = 455, Reflected Power = 2.421 mW
retlw	0x49	; High Byte, ADRES = 459, Reflected Power = 2.654 mW
retlw	0x49	; High Byte, ADRES = 463, Reflected Power = 2.909 mW
retlw	0x4A	; High Byte, ADRES = 467, Reflected Power = 3.189 mW
retlw	0x4B	; High Byte, ADRES = 471, Reflected Power = 3.496 mW
retlw	0x4B	; High Byte, ADRES = 475, Reflected Power = 3.832 mW
retlw	0x4C	; High Byte, ADRES = 479, Reflected Power = 4.200 mW
retlw	0x4C	; High Byte, ADRES = 483, Reflected Power = 4.604 mW
retlw	0x4D	; High Byte, ADRES = 487, Reflected Power = 5.046 mW
retlw	0x4D	; High Byte, ADRES = 491, Reflected Power = 5.531 mW
retlw	0x4E	; High Byte, ADRES = 495, Reflected Power = 6.063 mW
retlw	0x4E	; High Byte, ADRES = 499, Reflected Power = 6.646 mW
retlw	0x4F	; High Byte, ADRES = 503, Reflected Power = 7.284 mW
retlw	0x50	; High Byte, ADRES = 507, Reflected Power = 7.985 mW
retlw	0x50	; High Byte, ADRES = 511, Reflected Power = 8.752 mW
retlw	0x50	; High Byte, ADRES = 515, Reflected Power = 9.593 mW
retlw	0x51	; High Byte, ADRES = 519, Reflected Power = 10.52 mW
retlw	0x51	; High Byte, ADRES = 523, Reflected Power = 11.53 mW
retlw	0x52	; High Byte, ADRES = 527, Reflected Power = 12.63 mW
retlw	0x53	; High Byte, ADRES = 531, Reflected Power = 13.85 mW
retlw	0x53	; High Byte, ADRES = 535, Reflected Power = 15.18 mW
retlw	0x54	; High Byte, ADRES = 539, Reflected Power = 16.64 mW
retlw	0x54	; High Byte, ADRES = 543, Reflected Power = 18.24 mW
retlw	0x55	; High Byte, ADRES = 547, Reflected Power = 19.99 mW
retlw	0x55	; High Byte, ADRES = 551, Reflected Power = 21.91 mW
retlw	0x56	; High Byte, ADRES = 555, Reflected Power = 24.02 mW
retlw	0x56	; High Byte, ADRES = 559, Reflected Power = 26.33 mW
retlw	0x57	; High Byte, ADRES = 563, Reflected Power = 28.86 mW
retlw	0x58	; High Byte, ADRES = 567, Reflected Power = 31.63 mW
retlw	0x58	; High Byte, ADRES = 571, Reflected Power = 34.67 mW
retlw	0x58	; High Byte, ADRES = 575, Reflected Power = 38.01 mW
retlw	0x59	; High Byte, ADRES = 579, Reflected Power = 41.66 mW
retlw	0x59	; High Byte, ADRES = 583, Reflected Power = 45.67 mW
retlw	0x5A	; High Byte, ADRES = 587, Reflected Power = 50.06 mW
retlw	0x5B	; High Byte, ADRES = 591, Reflected Power = 54.87 mW
retlw	0x5B	; High Byte, ADRES = 595, Reflected Power = 60.14 mW
retlw	0x5C	; High Byte, ADRES = 599, Reflected Power = 65.92 mW
retlw	0x5C	; High Byte, ADRES = 603, Reflected Power = 72.26 mW
retlw	0x5D	; High Byte, ADRES = 607, Reflected Power = 79.21 mW
retlw	0x5D	; High Byte, ADRES = 611, Reflected Power = 86.82 mW
retlw	0x5E	; High Byte, ADRES = 615, Reflected Power = 95.16 mW
retlw	0x5E	; High Byte, ADRES = 619, Reflected Power = 104.3 mW
retlw	0x5F	; High Byte, ADRES = 623, Reflected Power = 114.3 mW

retlw	0x60	; High Byte, ADRES = 627, Reflected Power = 125.3 mW
retlw	0x60	; High Byte, ADRES = 631, Reflected Power = 137.4 mW
retlw	0x60	; High Byte, ADRES = 635, Reflected Power = 150.6 mW
retlw	0x61	; High Byte, ADRES = 639, Reflected Power = 165.1 mW
retlw	0x61	; High Byte, ADRES = 643, Reflected Power = 180.9 mW
retlw	0x62	; High Byte, ADRES = 647, Reflected Power = 198.3 mW
retlw	0x62	; High Byte, ADRES = 651, Reflected Power = 217.4 mW
retlw	0x63	; High Byte, ADRES = 655, Reflected Power = 238.3 mW
retlw	0x64	; High Byte, ADRES = 659, Reflected Power = 261.2 mW
retlw	0x64	; High Byte, ADRES = 663, Reflected Power = 286.3 mW
retlw	0x65	; High Byte, ADRES = 667, Reflected Power = 313.8 mW
retlw	0x65	; High Byte, ADRES = 671, Reflected Power = 344.0 mW
retlw	0x66	; High Byte, ADRES = 675, Reflected Power = 377.0 mW
retlw	0x66	; High Byte, ADRES = 679, Reflected Power = 413.3 mW
retlw	0x67	; High Byte, ADRES = 683, Reflected Power = 453.0 mW
retlw	0x67	; High Byte, ADRES = 687, Reflected Power = 496.5 mW
retlw	0x68	; High Byte, ADRES = 691, Reflected Power = 544.3 mW
retlw	0x68	; High Byte, ADRES = 695, Reflected Power = 596.6 mW
retlw	0x69	; High Byte, ADRES = 699, Reflected Power = 653.9 mW
retlw	0x69	; High Byte, ADRES = 703, Reflected Power = 716.8 mW
retlw	0x6A	; High Byte, ADRES = 707, Reflected Power = 785.7 mW
retlw	0x6A	; High Byte, ADRES = 711, Reflected Power = 861.2 mW
retlw	0x6B	; High Byte, ADRES = 715, Reflected Power = 944.0 mW
retlw	0x6C	; High Byte, ADRES = 719, Reflected Power = 1.035 W
retlw	0x6C	; High Byte, ADRES = 723, Reflected Power = 1.134 W
retlw	0x6C	; High Byte, ADRES = 727, Reflected Power = 1.243 W
retlw	0x6D	; High Byte, ADRES = 731, Reflected Power = 1.363 W
retlw	0x6D	; High Byte, ADRES = 735, Reflected Power = 1.494 W
retlw	0x6E	; High Byte, ADRES = 739, Reflected Power = 1.637 W
retlw	0x6F	; High Byte, ADRES = 743, Reflected Power = 1.795 W
retlw	0x6F	; High Byte, ADRES = 747, Reflected Power = 1.967 W
retlw	0x70	; High Byte, ADRES = 751, Reflected Power = 2.156 W
retlw	0x70	; High Byte, ADRES = 755, Reflected Power = 2.364 W
retlw	0x71	; High Byte, ADRES = 759, Reflected Power = 2.591 W
retlw	0x71	; High Byte, ADRES = 763, Reflected Power = 2.840 W
retlw	0x72	; High Byte, ADRES = 767, Reflected Power = 3.113 W
retlw	0x72	; High Byte, ADRES = 771, Reflected Power = 3.412 W
retlw	0x73	; High Byte, ADRES = 775, Reflected Power = 3.740 W
retlw	0x74	; High Byte, ADRES = 779, Reflected Power = 4.100 W
retlw	0x74	; High Byte, ADRES = 783, Reflected Power = 4.494 W
retlw	0x74	; High Byte, ADRES = 787, Reflected Power = 4.926 W
retlw	0x75	; High Byte, ADRES = 791, Reflected Power = 5.399 W
retlw	0x75	; High Byte, ADRES = 795, Reflected Power = 5.918 W
retlw	0x76	; High Byte, ADRES = 799, Reflected Power = 6.487 W
retlw	0x77	; High Byte, ADRES = 803, Reflected Power = 7.110 W
retlw	0x77	; High Byte, ADRES = 807, Reflected Power = 7.794 W
retlw	0x78	; High Byte, ADRES = 811, Reflected Power = 8.543 W
retlw	0x78	; High Byte, ADRES = 815, Reflected Power = 9.364 W
retlw	0x79	; High Byte, ADRES = 819, Reflected Power = 10.26 W
retlw	0x79	; High Byte, ADRES = 823, Reflected Power = 11.25 W
retlw	0x7A	; High Byte, ADRES = 827, Reflected Power = 12.33 W
retlw	0x7A	; High Byte, ADRES = 831, Reflected Power = 13.52 W
retlw	0x7B	; High Byte, ADRES = 835, Reflected Power = 14.82 W
retlw	0x7C	; High Byte, ADRES = 839, Reflected Power = 16.24 W
retlw	0x7C	; High Byte, ADRES = 843, Reflected Power = 17.80 W
retlw	0x7C	; High Byte, ADRES = 847, Reflected Power = 19.51 W
retlw	0x7D	; High Byte, ADRES = 851, Reflected Power = 21.39 W
retlw	0x7D	; High Byte, ADRES = 855, Reflected Power = 23.45 W
retlw	0x7E	; High Byte, ADRES = 859, Reflected Power = 25.70 W
retlw	0x7F	; High Byte, ADRES = 863, Reflected Power = 28.17 W
retlw	0x7F	; High Byte, ADRES = 867, Reflected Power = 30.88 W
retlw	0x80	; High Byte, ADRES = 871, Reflected Power = 33.85 W
retlw	0x80	; High Byte, ADRES = 875, Reflected Power = 37.10 W
retlw	0x81	; High Byte, ADRES = 879, Reflected Power = 40.67 W
retlw	0x81	; High Byte, ADRES = 883, Reflected Power = 44.58 W
retlw	0x82	; High Byte, ADRES = 887, Reflected Power = 48.86 W
retlw	0x82	; High Byte, ADRES = 891, Reflected Power = 53.56 W
retlw	0x83	; High Byte, ADRES = 895, Reflected Power = 58.70 W
retlw	0x84	; High Byte, ADRES = 899, Reflected Power = 64.35 W
retlw	0x84	; High Byte, ADRES = 903, Reflected Power = 70.53 W
retlw	0x84	; High Byte, ADRES = 907, Reflected Power = 77.31 W
retlw	0x85	; High Byte, ADRES = 911, Reflected Power = 84.75 W
retlw	0x85	; High Byte, ADRES = 915, Reflected Power = 92.89 W
retlw	0x86	; High Byte, ADRES = 919, Reflected Power = 101.8 W
retlw	0x86	; High Byte, ADRES = 923, Reflected Power = 111.6 W
retlw	0x87	; High Byte, ADRES = 927, Reflected Power = 122.3 W
retlw	0x88	; High Byte, ADRES = 931, Reflected Power = 134.1 W
retlw	0x88	; High Byte, ADRES = 935, Reflected Power = 147.0 W
retlw	0x89	; High Byte, ADRES = 939, Reflected Power = 161.1 W
retlw	0x89	; High Byte, ADRES = 943, Reflected Power = 176.6 W
retlw	0x8A	; High Byte, ADRES = 947, Reflected Power = 193.6 W

```

retlw    0x8A        ; High Byte, ADRES = 951, Reflected Power = 212.2 W
retlw    0x8B        ; High Byte, ADRES = 955, Reflected Power = 232.6 W
retlw    0x8B        ; High Byte, ADRES = 959, Reflected Power = 254.9 W
retlw    0x8C        ; High Byte, ADRES = 963, Reflected Power = 279.4 W
retlw    0x8C        ; High Byte, ADRES = 967, Reflected Power = 306.3 W
retlw    0x8D        ; High Byte, ADRES = 971, Reflected Power = 335.7 W
retlw    0x8D        ; High Byte, ADRES = 975, Reflected Power = 368.0 W
retlw    0x8E        ; High Byte, ADRES = 979, Reflected Power = 403.4 W
retlw    0x8E        ; High Byte, ADRES = 983, Reflected Power = 442.2 W
retlw    0x8F        ; High Byte, ADRES = 987, Reflected Power = 484.7 W
retlw    0x90        ; High Byte, ADRES = 991, Reflected Power = 531.3 W
retlw    0x90        ; High Byte, ADRES = 995, Reflected Power = 582.3 W
retlw    0x90        ; High Byte, ADRES = 999, Reflected Power = 638.3 W
retlw    0x91        ; High Byte, ADRES = 1003, Reflected Power = 699.7 W
retlw    0x91        ; High Byte, ADRES = 1007, Reflected Power = 766.9 W
retlw    0x92        ; High Byte, ADRES = 1011, Reflected Power = 840.6 W
retlw    0x93        ; High Byte, ADRES = 1015, Reflected Power = 921.5 W
retlw    0x93        ; High Byte, ADRES = 1019, Reflected Power = 1,010.0 W
retlw    0x94        ; High Byte, ADRES = 1023, Reflected Power = 1,107.1 W

endif

end

```