**STPM33 register driver code explanation**

1. Stpm33 register code is divided into two parts. One is the c file (2) stpm33\_register\_code.c), in which main code is written. Second is the header file(1) stpm33\_all\_register\_check\_code.c) in which, all constants to be used, are defined.
2. In the header file, following register’s bits along with their possible values are defined:
3. 12 Digital Signal Processor control register, ranging from DSP\_CR1 to DSP\_CR12
4. 2 Digital Front End control register, DFE\_CR1 and DFE\_CR2
5. 2 Digital Signal Processor Status register, DSP\_SR1 and DSP\_SR2
6. Bit assignment and register configuration unsigned integers are all initialized in different section
7. “metrology.h” file’s constants are also copied in this header file for easy access to macros.
8. LSB pin position of each calibrator of each calibration register is defined in another section.
9. For some bits in a particular register, can have multiple values in the order of 210. For those, particular bits, only default values are defined in the header file.   
   These bits, which can have many values, can be configured in c file (2)stpm33\_register\_code.c)   
   e.g., CHV1. Its default value i.e. 0x800 is defined in header file.   
   But since it is 12 bit register. It can 212 possible values.

According to our requirement, we can set whatever value we want in the c file, in the unsigned int CHV1 = “input”  
For CHV1 correct value, we need to consider Vav and Xv.  
uint8\_t Xv = (Vn\*GAIN\_I\_STAGE\_0\*CAL\_V\*pow(2,15))/(V\_REF\*((1+R1\_V1)/R2\_V1)));  
  
Since, Xv depends on Vn, which is assumed to be 230V, we only need to consider Vav. Vav is the average values of 20 sampled of rms voltage in 5 line cycles, which is to be read via DSP\_REG14 and DSP\_REG\_16 and then calculated accordingly.

1. In the c file, each register’s bits are “OR”-ed and equated to the corresponding register.  
   e.g., DSP\_CR1 = LCS1\_primary | LPS1\_active | LPW1\_4 | ROC1\_bypassed | BHPFC1\_enabled | BHPFV1\_enabled | APM1\_Fundamental | AEM1\_RMS | TC1\_30 | ENVREF1\_enabled | CLRSS1 | CLR\_SS\_TIME\_OUT1\_15;
2. Next in the c file, we have section for voltage sag and swell parameters, where we can directly set the voltage sag and swell rms values and current sag and swell rms values.
3. Similarly, “Amplitude calibration OR Voltage and current calibration”, “Phase calibration”, “Power offset calibration”, “Neutral missing OR Tamper parameters” are defined.
4. Whatever values we give input as, needs to be converted using conversion formula and then it needs to be converted to hexadecimal.  
   e.g., SWV\_THR1\_value = 253;//rms Volts  
   Swell threshold value input is 253 V . It needs to be multiplied by its LSB, which is done using SWV\_THR1\_conversion. And then it is stored in the required bit of required register. Nevertheless, before storing this value, any pre-stored-value must be cleared. And then new value must be pasted. This is done with the help of AND , OR.  
   DSP\_CR5 = DSP\_CR5 & ~((0x3FF)<<SWV\_THR1\_pin\_position); //clear the respective swell bits

DSP\_CR5 = DSP\_CR5 | (SWV\_THR1\_conversion<<SWV\_THR1\_pin\_position); //writing the new swell bits

1. Then we are printing the corresponding 32 bit register values in hexadecimal.
2. The last section, we are reading the stpm33\_register value and comparing it with required value. If both, are same , then successful written operation on register. Else, there was some error in writing those register values.