HELP Si5351 Library

The Si5351 (below for 3 output A version) library is an advanced frequency setting library with the ability to automatically calculate the best PLL multiplier and Multisynth divider to get the required output frequency.

(see https://github.com/etherkit/Si5351Arduino). Uses freq in cHz.

```
Si5351 dds;
// create object

void dds.init(uint8_t xtal_load_c, uint32_t ref_osc_freq);
// setup comms, I2C addr 0x60
// xtal_load_c = SI5351_CRYSTAL_LOAD_*PF, * = 6, 8, 10
// ref_osc_freq = 0 (for default 25MHz) or actual XTAL freq in Hz

void dds.set_freq(uint64_t freq, uint64_t pll_freq, enum si5351_clock_clk);
// set output
// freq (Hz)
// pll_freq input to multisynth SI5351_PLL_FIXED (= 90000000000cHz), or 0 for
// function to chose automatically
// si5351_clock_clk = SI5351_CLK*, * = 0, 1, 2

void dds.set_pll(uint64_t pll_freq, enum si5351_pll target_pll);
// pll_freq (cHz)
// target_pll = SI5351_PLL*, * = A,B

void dds.output_enable(enum si5351_clock_clk, uint8_t enable);
// si5351_clock_clk = SI5351_CLK*, * = 0, 1, 2
// enable = 1, disable = 0

void dds.drive_strength(enum si5351_clock_clk, enum si5351_drive);
// si5351_clock_clk = SI5351_CLK*, * = 0, 1, 2
// si5351_drive = SI5351_DRIVE_*MA, MA = 2, 4, 6, 8 (8mA give +14dBm out or 25mW)

void set_correction(int32_t corr);
// set osc correction in EEPROM
// corr = part-per-10M freq that deviates from actual, +/-Hz x 10
// do once as EEPROM value read by init

uint32_t get_correction(void);
// reads stored EEPROM correction

void dds.set_phase(enum si5351_clock_clk, uint8_t phase)
// si5351_clock_clk = SI5351_CLK*, * = 0, 1, 2
// Set CLK (MHz)to be even multiple of PLL (600-900MHz). Then 90deg shift needs
PLL/CLK int.

void pll_reset(enum si5351_pll)
// si5351_pll = SI5351_PLL*, * = A, B
```

```
void dds.set_ms_source(enum si5351_clock_clk, enum si5351_pll)
// si5351_clock_clk = SI5351_CLK*, * = 0, 1, 2
// si5351_pll = SI5351_PLL*, * = A, B

int32_t corr_factor = dds.get_correction()
// reads correction from EEPROM
// corr is (actual - nominal) freq x10 => make integer
// if 0 then set it once as below
// library is set for nominal 25MHz xtal, set in si5351.h as SI5351_XTAL_DEFINE

void dds.set_correction(int32_t corr);
// e.g. corr = ppb value that actual oscillator deviates from specified
// measure freq_difference (Hz) ppb = diff x 10
// value stored in EEPROM, so do only once. Result called up by init()
```

Tokens

```
    SI5351_CRYSTAL_LOAD_6PF
    SI5351_CRYSTAL_LOAD_8PF
    SI5351_CRYSTAL_LOAD_10PF

    enum si5351_clock {SI5351_CLK0, SI5351_CLK1, SI5351_CLK2, SI5351_CLK3,
                       SI5351_CLK4, SI5351_CLK5, SI5351_CLK6, SI5351_CLK7};

    enum si5351_pll {SI5351_PLLA, SI5351_PLLB};

    enum si5351_drive {SI5351_DRIVE_2MA, SI5351_DRIVE_4MA, SI5351_DRIVE_6MA,
                       SI5351_DRIVE_8MA};

    enum si5351_clock_source {SI5351_CLK_SRC_XTAL, SI5351_CLK_SRC_CLKIN,
                              SI5351_CLK_SRC_MS0, SI5351_CLK_SRC_MS};


The library by default uses PLLA for CLK0, PLLB for CLK1 and PLLB again for CLK2
```

Code - a simple example

```
// added comments
#include "si5351.h"
#include "Wire.h" // for I2C comms

Si5351 dds; // object

// frequencies in MHz
uint32_t f14 = 14000000;
uint32_t f20 = 20000000;
```

```
void setup()
{
  // Start serial
  Serial.begin(9600);

  // init the dds, uses I2C addr in '.h' file 0x06, 8pF dds xtal
  dds.init(SI5351_CRYSTAL_LOAD_8PF);

  // exmaple 1: Set CLK0 to output 14 MHz with a fixed PLL frequency
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA); // set PLLA to 900000000
  si5351.set_freq(f14 * SI5351_FREQ_MULT, SI5351_PLL_FIXED, SI5351_CLK0);
  si5351.clock_enable(SI5351_CLK0, 1); // enable clock 0 output

  // example 2: Set CLK1 to output 20 MHz
  si5351.set_freq(f20 * SI5351_FREQ_MULT, 0, SI5351_CLK1); // function choses PLL
frequency itself
  si5351.clock_enable(SI5351_CLK1, 1); // , clock 1, enable "1", disable "0"
}

void loop()
{
}
```

TAKE CARE this NEW library uses frequencies in 0.01Hz steps, so 14MHz is uint64_t = 14,000,000,00. To use 32 bit ints (frequency in Hz) define as uint32_t and multiply by SI5351_FREQ_MULT (= 100 in si5351.h) in set_freq function.

Second example - IQ generation

Two outputs can be set to be 90deg out of phase, i.e. in quadrature for IQ generation. This is done by shifting the phase of one output by PLL/CLK. PLL must be an even multiple of CLK and in the range 600-900MHz.

```
// IQ generator at 10MHz
// V0.5 M6KWH ganymedeham.blogspot.com
// Uses NEW "github.com/etherkit/Si5351Arduino" library

// ----- CONNECTIONS
// DDS I2C SI5351
// SCL = A5
// SDA = A4
// I2C address 0x60

// I2C, Si5351 libraries
#include "Wire.h"
#include "si5351.h"

// IQ output freq & PLLB freq
```

```
uint32_t iq_freq = 10000000;
uint32_t iq_pll_freq = 800000000;
uint8_t iq_phase = iq_pll_freq / iq_freq; // must be even number

// dds object
Si5351 dds;

void setup()
{
  dds.init(SI5351_CRYSTAL_LOAD_8PF, 0); // initialise

  dds.output_enable(SI5351_CLK0, 0); // enable CLK1 & 2
  dds.output_enable(SI5351_CLK1, 1);
  dds.output_enable(SI5351_CLK2, 1);

  IQOut(iq_freq, iq_pll_freq, iq_phase);
}

void loop()
{

}

// IQ out CLK1 & 2 at f (MHz), PLLB at pf (MHz), phase p
void IQOut(uint32_t f, uint32_t pf, uint8_t p)
{
 // set PLLB freq
  dds.set_pll(pf * SI5351_FREQ_MULT, SI5351_PLLB);

 // set MS for CLK1 & 2
  dds.set_ms_source(SI5351_CLK1, SI5351_PLLB);
  dds.set_ms_source(SI5351_CLK2, SI5351_PLLB);

  // set output CLK1 &2
  dds.set_freq(f * SI5351_FREQ_MULT, pf * SI5351_FREQ_MULT, SI5351_CLK1);
  dds.set_freq(f * SI5351_FREQ_MULT, pf * SI5351_FREQ_MULT, SI5351_CLK2);

 // set phases of CKL1 & 2
  dds.set_phase(SI5351_CLK1, 0);
  dds.set_phase(SI5351_CLK2, p);

 // restart PLLB to get in phase
  dds.pll_reset(SI5351_PLLB);
}
```