应用报告
ZHCA581A–November 2013

使用 C2000™ 微控制器针对三相并网应用的软件锁相环设计

Manish Bhardwaj

摘要

并网应用需要对电网相角进行准确估算以将电能同步馈入电网。 这通过一个软件锁相环 (PLL) 实现。 这份应用报告讨论了为三相并网逆变器设计软件锁相环时的不同需要，并且提出一个使用 C2000 控制器来设计锁相环的方法。

为了举例说明，讨论了两个常见的 PLL 结构 [1]， [2]，并且使用 C2000 MCU 对其进行设计。

内容

1 简介 1
2 同步基准框锁相环 (PLL) 3
3 三相电网内的不均衡 12
4 去耦合双同步基准框 PLL 13
5 太阳能库和 ControlSuite 25
6 参考书目 26

附图目录

1 从三相电压到静态和旋转基准框的变换 2
2 三相到旋转基准框变换模拟 2
3 静态基准框上基于 3 相位的 SPLL 3
4 不同电网条件下的 PLL 响应 9
5 SRF SPLL 使用流程图 11
6 三相电网内的不均衡 12
7 不均衡三相系统的正和负序列 13
8 DDSRF PLL 结构 15
9 不同电网条件下的 PLL 响应 20
10 DDSRF SPLL 流程图 24
11 太阳能库和 ControlSuite 25

1 简介

此工具的相位角是诸如光伏 (PV) 逆变器等将电能馈入电网的功率器件运行所需的关键信息。 一个 PLL 是一个闭环系统，在这个系统中，内部振荡器被控制用来使用反馈环来保存一个外部周期信号的时间和相位。 此锁的质量直接影响并网应用中控制环路的性能。 在线路多级时，电压不均衡、线路骤降、相位损耗和频率变化是与电网对接设备的常见状况。 此 PLL 需要能够抑制这些错误源，并且保持一个到电网电压的洁净相位锁。

在三相系统中，通常将三相时间变化量转换为一个直流系统（在一个旋转基准框内）。

公式 1 显示电压的顺序为 $V_a \rightarrow V_c \rightarrow V_b$ 并且频率为 ω。

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = V \begin{bmatrix} \cos(\omega t) \\ \cos(\omega t - 2\pi / 3) \\ \cos(wt - 4\pi / 3) \end{bmatrix} \quad (1)$$

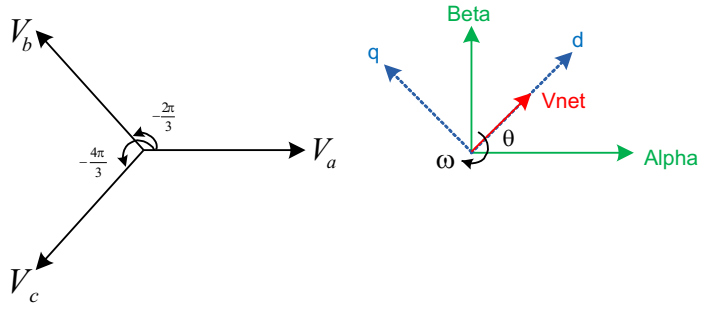


图 **1.** 从三相电压到静态和旋转基准框的变换

要把三相量变换为旋转基准框，第一步是将三相量变换为一个正交分量系统（alpha，beta 也被称为静态基准框），变换方法是将三相量投射到一个正交坐标轴上。 这被称为 Clarke 变换（请见 公式 2）。

$$V_{\alpha\beta 0} = T_{abc \rightarrow \alpha\beta 0} V_{abc}$$

$$\begin{bmatrix} V_\alpha \\ V_\beta \\ V_o \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & \cos(2\pi/3) & \cos(4\pi/3) \\ 0 & \sin(2\pi/3) & \sin(4\pi/3) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} X \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\omega t) \\ \sin(\omega t) \\ 0 \end{bmatrix} V \quad (2)$$

在静态基准框内，电压矢量净量与正交基准框之间形成一个角度 θ，并且以 ω 的频率旋转。 然后，通过获得静态基准框分量在旋转基准框上的投影来将此系统减至直流。 这被称为 Park 变换（请见 公式 3）。

$$V_{dq0} = T_{\alpha\beta 0 \rightarrow dq0} V_{\alpha\beta 0}$$

$$\begin{bmatrix} v_d \\ v_q \\ v_o \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} v_\alpha \\ v_\beta \\ v_o \end{bmatrix} \quad (3)$$

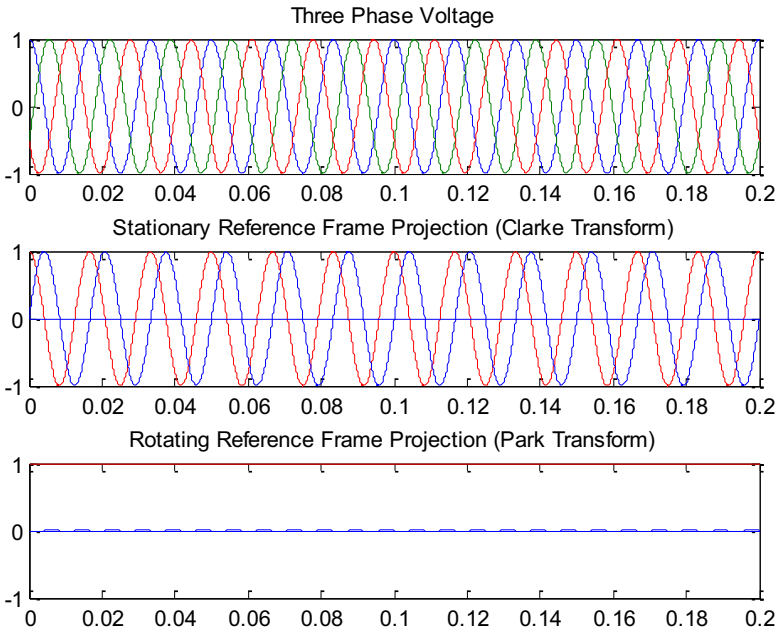


图 **2.** 三相到旋转基准框变换模拟

2 同步基准框锁相环 (PLL)

三相环境中 PLL 的作用是通过测量瞬时电压波形来准确估计电压矢量净量产生的角度。 假设 PLL 估算出的角度为 θ，而实际角度为 ω*t，可使用 公式 2 和 公式 3 写出 ABC->DQ0 变换：

$$\begin{bmatrix} v_d \\ v_q \\ v_o \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} \cos(\omega t) \\ \sin(\omega t) \\ 0 \end{bmatrix} V = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) * \cos(\omega t) + \sin(\theta)\sin(\omega t) \\ -\sin(\theta) * \cos(\omega t) + \cos(\theta)\sin(\omega t) \\ 0 \end{bmatrix} V \tag{4}$$

使用三角恒等式，公式 5 可被精简为：

$$\begin{bmatrix} v_d \\ v_q \\ v_o \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\omega t - \theta) \\ \sin(\omega t - \theta) \\ 0 \end{bmatrix} \tag{5}$$

当 PLL 角度接近于实际电压矢量角时，(ω*t-θ) 很小或接近于零，而此时 sin(ω*t-θ) ~ = (ω*t-θ)。 因此，可以认为，对于一个均衡三相系统，当 PLL 被锁定时，旋转基准框内的 q 轴分量减少为零，而当 PLL 未被锁定时，或者误差很小时，q 轴与误差成正比线性关系。

$V_q \approx (\omega t - \theta)$ (6)

这一属性在同步基准框 PLL 中用于三相并网应用。 此三相量被变换为旋转基准框，并且 q 分量被用作相位检测值。 然后，一个低通滤波器 / PI 被用来消除稳定状态误差以及被馈入压控振荡器 (VCO) 的输出，此输出产生角度和正弦值。

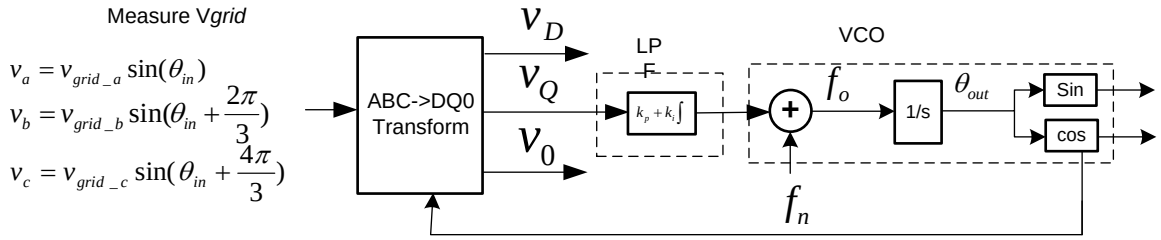


图 3. 静态基准框上基于 3 相位的 SPLL

从 公式 6 中可知，角度锁定中的任何误差将出现在 q 数据项上，并且小值误差间的关系是线性的，请见 公式 7：

$err \propto V_{grid} (\theta_{grid} - \theta_{PLL})$ (7)

使用网络理论来完成小信号分析，在这里，反馈环路被打破，以获得开环传输等式，然后，闭环传输函数给出如下：

闭环 TF = 开环 TF / (1+ OpenLoopTF) (8)

PLL 传输函数可写为如下：

$$\text{Closed loop Phase TF}: \quad H_o(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{LF(s)}{s + LF(s)} = \frac{v_{grid}(k_p s + \frac{k_p}{T_i})}{s^2 + v_{grid} k_p s + v_{grid} \frac{k_p}{T_i}}$$

$$\text{Closed loop error transfer function}: \quad E_o(s) \frac{V_d(s)}{\theta_{in}(s)} = 1 - H_o(s) = \frac{s}{s + LF(s)} = \frac{s^2}{s^2 + k_p s + \frac{k_p}{T_i}} \tag{9}$$

比较闭环相位传输函数与普通二阶系统传输函数：

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{10}$$

现在，将这个函数与闭环相位传输函数相比较，线性化 PLL 的固有频率和阻尼率给出如下：

$$\omega_n = \sqrt{\frac{v_{grid} K_p}{T_i}}$$

$$\zeta = \sqrt{\frac{v_{grid} T_i K_p}{4}} \tag{11}$$

2.1 PI 控制器的离散执行

如 公式 12 中所示，环路滤波器或 PI 被执行为一个数字控制器：

$$ylf[n] = ylf[n-1] * A1 + ynotch[n] * B0 + ynotch[n-1] * B1 \tag{12}$$

通过使用 z 变换，公式 13 可被重写为：

$$\frac{ylf(z)}{ynotch(z)} = \frac{B0 + B1 * z^{-1}}{1 - z^{-1}} \tag{13}$$

环路滤波器（PI 控制器）的 Laplace 变换已知可由以下等式给出：

$$\frac{ylf(s)}{ynotch(s)} = K_p + \frac{K_i}{s} \tag{14}$$

现在，使用双线性变换，替代 $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$，在这里，T = 采样时间。

$$\frac{ylf(z)}{ynotch(z)} = \frac{\left(\frac{2 * K_p + K_i * T}{2}\right) - \left(\frac{2 * K_p - K_i * T}{2}\right) z^{-1}}{1 - z^{-1}} \tag{15}$$

可比较 公式 2 和 公式 3 来将 PI 控制器的比例增益和积分增益映射到数字域。 下一个问题是为比例和积分增益选择一个合适的值。

已知通用二阶等式的阶跃响应为：

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{16}$$

给出如下：

$$y(t) = 1 - ce^{-\sigma t}\sin(\omega_d t + \varphi) \tag{17}$$

忽略 公式 17 中的 LHP 零值，稳定时间被给出为一个误差带之间响应到稳定所花费的时间，假定这个值为 ∂，那么：

$$1 - \partial = 1 - ce^{-\sigma t} \Rightarrow \partial = ce^{-\sigma t} \Rightarrow t_s = \frac{1}{\sigma} * \ln\left(\frac{c}{\sigma}\right)$$
$$\text{Where } \sigma = \varsigma\omega_n \text{ and } c = \frac{\omega_n}{\omega_d} \text{ and } \omega_d = \sqrt{1 - \varsigma^2}\,\omega_n \tag{18}$$

使用 30ms 的稳定时间，5% 的误差带以及 0.7 的阻尼比，可获得 158.6859 的固有频率；然后，通过回代，您可得到 K_p = 222.1603 和 K_1 = 25181.22。

将这些值回代至数字环路滤波器系数中，您可以得到：

$$B0 = \left(\frac{2 * K_p + K_i * T}{2}\right) \text{ and } B1 = -\left(\frac{2 * K_p - K_i * T}{2}\right) \tag{19}$$

对于 10Khz 的 PLL 运行速度来说，B0 = 223.4194 并且 B1 = -220.901。

2.2 模拟不同条件下的锁相环

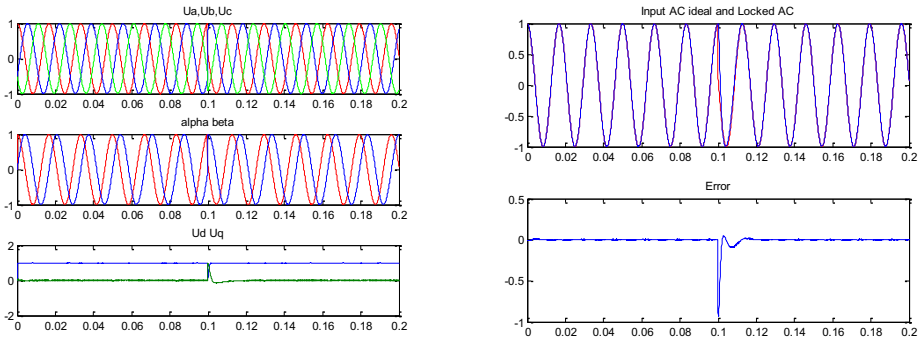
在编写 SPLL 结构代码时，对于电网上的不同条件，模拟 PLL 的运行方式很重要。 定点处理器被用来降低很多并网转换器的成本。 IQ Math 是一个便捷的途径来观察具有小数点的定点数。 C2000 IQ math 库提供内置函数，此函数可简化编程人员对小数点的处理。 第一个 MATLAB 被用来模拟和识别此算法需要运行的 Q 点。 下面是使用定点工具箱的 MATLAB 脚本，此脚本测试不同电网条件下的 PLL 算法。

```
%%%%%%%%%%%%%%%%%%%% PLL 3ph Modeling %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Texas Instruments
% Digital Control Systems Group, Houston, TX
% Manish Bhardwaj
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This software is licensed for use with Texas Instruments C28x
%  family DSCs.  This license was provided to you prior to installing
%  the software.
% ------------------------------------------------------------------------
%          Copyright (C) 2010-2012 Texas Instruments, Incorporated.
%                          All Rights Reserved.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

%Select numeric type,
T=numerictype('WordLength',32,'FractionLength',22);

%Specify math attributes to the fimath object
F=fimath('RoundMode','floor','OverflowMode','wrap');
F.ProductMode='SpecifyPrecision';
F.ProductWordLength=32;
F.ProductFractionLength=22;
F.SumMode='SpecifyPrecision';
F.SumWordLength=32;
F.SumFractionLength=22;

%specify fipref object, to display warning in cases of overflow and
%underflow
P=fipref;
P.LoggingMode='on';
P.NumericTypeDisplay='none';
P.FimathDisplay='none';

%PLL Modelling starts from here
Fs=10000;
Ts=1/Fs;            %Sampling Time = (1/fs) , Note Ts is related to the frequency
                    % the ISR would run in the solar application as well, assuming
                    % 10Khz control loop for the inverter

Tfinal=0.2;         % Total time the simulation is run for
t=0:Ts:Tfinal;      % time vector
GridFreq=60;        % nominal grid frequency
wn=2*pi*GridFreq;   % nominal frequency in radians
fn=GridFreq;
wu=2*pi*GridFreq;
theta=rem((t*2*pi*GridFreq),2*pi);
L=length(t);

%generate input signal and create a fi object of it

% CASE 1 : Phase Jump at the Mid Point
%
% for n=1:floor(L/2)
```

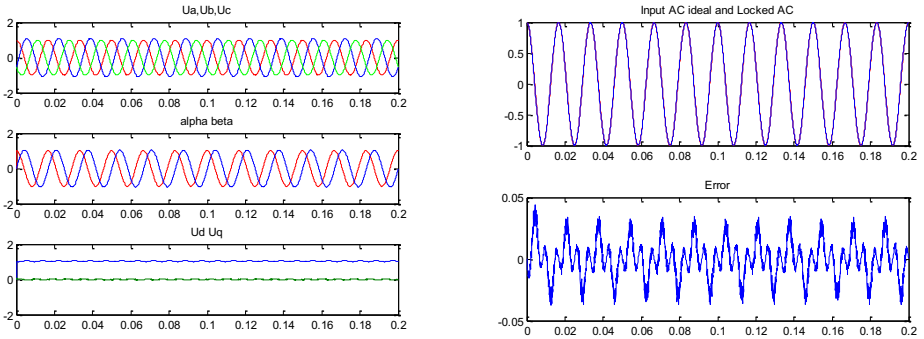
```
%     Ua(n)=cos(theta(n));
%     Ub(n)=cos(theta(n)-2*pi/3);
%     Uc(n)=cos(theta(n)-4*pi/3);
% end
% for n=floor(L/2):L
%     Ua(n)=cos(theta(n)+1.5);
%     Ub(n)=cos(theta(n)-2*pi/3+1.5);
%     Uc(n)=cos(theta(n)-4*pi/3+1.5);
% end
% Ua_ideal=Ua;

%CASE 2: Unbalanced Grid
Ua_ideal=cos(theta);


Ua=fi(Ua,T,F);
Ub=fi(Ub,T,F);
Uc=fi(Uc,T,F);
Ua_ideal=fi(Ua_ideal,T,F);

%declare arrays used by the PLL process
ylf =fi([0,0],T,F);
u_q =fi([0,0],T,F);
Theta=fi([0,0,0],T,F);
fo=fi(0,T,F);
Ualpha=fi([0,0],T,F);
Ubeta=fi([0,0],T,F);
Ud_plus=fi([0,0],T,F);
Uq_plus=fi([0,0],T,F);
fn=fi(fn,T,F);
wo=fi(0,T,F);

% simulate the PLL process
for n=3:L    % No of iteration of the PLL process in the simulation time

    Ualpha(n) =fi((2.0/3.0),T,F)*(Ua(n)-fi(0.5,T,F)*(Ub(n)+Uc(n)));
    Ubeta(n)  =fi((2.0/3.0)*0.866,T,F)*(Ub(n)-Uc(n));

    Ud_plus(n)=Ualpha(n)*cos(Theta(n))+Ubeta(n)*sin(Theta(n));
    Uq_plus(n)=-Ualpha(n)*sin(Theta(n))+Ubeta(n)*cos(Theta(n));

    u_q(1)=Uq_plus(n);

    %Loop Filter
    ylf(1)=ylf(2)+fi(167.9877775,T,F)*u_q(1)-fi(165.2122225,T,F)*u_q(2);
    %ylf(1)=ylf(2)+fi(1000.00005,T,F)*u_q(1)-fi(999.99995,T,F)*u_q(2);

    ylf(1)=min([ylf(1) fi(200.0,T,F)]);

    %update u_q for future use
    u_q(2)=u_q(1);

    ylf(2)=ylf(1);

    %update output frequency
    fo=fn+ylf(1);

    Theta(n+1)=Theta(n)+fi(2*pi,T,F)*(fo*Ts);

    if(Theta(n+1)>=(fi(2*pi,T,F)-fi(2*pi,T,F)*(fo*Ts)))
        Theta(n+1)=0;
    end

end
```

```
figure,subplot(3,1,1),plot(t,Ua,'r',t,Ub,'b',t,Uc,'g'),title('Ua,Ub,Uc');
subplot(3,1,2),plot(t,Ualpha,'r',t,Ubeta),title('alpha beta');
subplot(3,1,3),plot(t,Ud_plus(1:L),t,Uq_plus(1:L)), title('Ud Uq') ;

figure,subplot(2,1,1),plot(t,Ua_ideal,'r',t,cos(Theta(1:L)),'b'),title('Input AC ideal and Locked
AC');
subplot(2,1,2),plot(t,Ua_ideal-cos(Theta(1:L))),title('Error');
% for n=1:floor(L)
% Ua(n)=cos(theta(n));
% Ub(n)=1.1*cos(theta(n)-2*pi/3);
% Uc(n)=cos(theta(n)-4*pi/3);
% end
% Ua_ideal=cos(theta);

%CASE 3: voltage harmonics
% for n=1:floor(L)
% Ua(n)=cos(theta(n))+0.05*cos(5*theta(n));
% Ub(n)=cos(theta(n)-2*pi/3)+0.05*(cos(5*(theta(n)-2*pi/3)));
% Uc(n)=cos(theta(n)-4*pi/3)+0.05*(cos(5*(theta(n)-4*pi/3)));
% end
% Ua_ideal=cos(theta);

% CASE 4: voltage dips and swells

for n=1:floor(L/2)
    Ua(n)=cos(theta(n));
    Ub(n)=cos(theta(n)-2*pi/3);
    Uc(n)=cos(theta(n)-4*pi/3);
end
for n=floor(L/2):L
    Ua(n)=0.7*cos(theta(n));
    Ub(n)=0.7*cos(theta(n)-2*pi/3);
    Uc(n)=0.7*cos(theta(n)-4*pi/3);
end
```

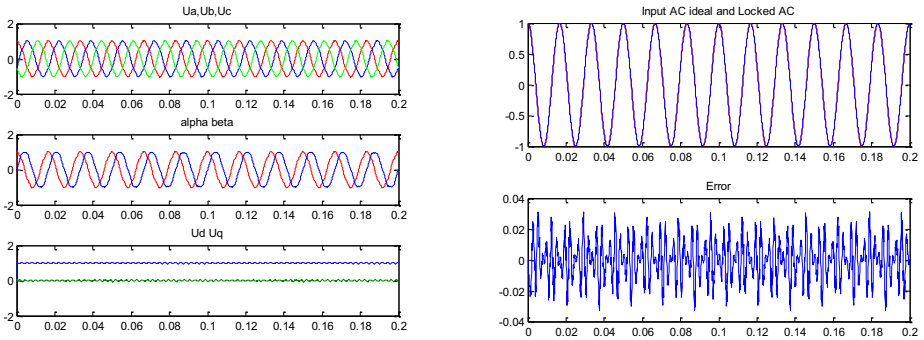
下面是 PLL 上不同电网条件的结果：



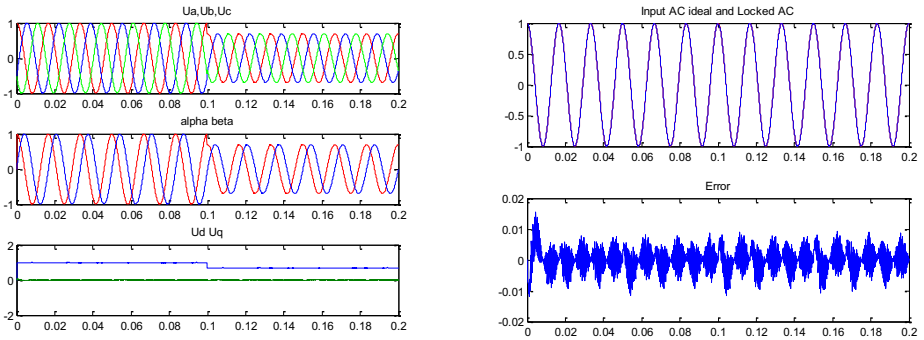
(a)



(b)



(c)



(d)

A 90°的相位跳变

B 一个相位上 10% 的电压不均衡

C 5% 5th 谐波含量

D 振幅变化（电压暂降和骤降）

图 **4.** 不同电网条件下的 **PLL** 响应

2.3 在使用 IQ Math 的 C2000 控制器上执行 PLL

如上面部分显示，由于普通逆变器通常使用 IQ24，为 PLL 选择的 Q 点为 IQ21，PLL 的代码可写出如下。

SPLL_3ph_SRF_IQ.h 头文件

:

```
#ifndef SPLL_3ph_SRF_IQ_H_
#define SPLL_3ph_SRF_IQ_H_

#define SPLL_SRF_Q _IQ21
#define SPLL_SRF_Qmpy _IQ21mpy

//*********** Structure Definition ********//
typedef struct{
           int32      B1_lf;
           int32      B0_lf;
           int32      A1_lf;
}SPLL_3ph_SRF_IQ_LPF_COEFF;

typedef struct{
           int32 v_q[2];
           int32 ylf[2];
           int32 fo; // output frequency of PLL
           int32 fn; //nominal frequency
           int32 theta[2];
           int32 delta_T;
           SPLL_3ph_SRF_IQ_LPF_COEFF lpf_coeff;
}SPLL_3ph_SRF_IQ;

//*********** Function Declarations *******//
void SPLL_3ph_SRF_IQ_init(int Grid_freq, long DELTA_T, SPLL_3ph_SRF_IQ *spll);
void SPLL_3ph_SRF_IQ_FUNC(SPLL_3ph_SRF_IQ *spll_obj);

//*********** Macro Definition ***********//
#define SPLL_3ph_SRF_IQ_MACRO(spll_obj)\
           /*update v_q[0] before calling the routine*/ \
           /* Loop Filter                      */ \

spll_obj.ylf[0]=spll_obj.ylf[1]+SPLL_SRF_Qmpy(spll_obj.lpf_coeff.B0_lf,spll_obj.v_q[0])+SPLL_SRF_Q
mpy(spll_obj.lpf_coeff.B1_lf,spll_obj.v_q[1]);             \
           spll_obj.ylf[1]=spll_obj.ylf[0];  \
           spll_obj.v_q[1]=spll_obj.v_q[0];  \
           spll_obj.ylf[0]=(spll_obj.ylf[0]>SPLL_SRF_Q(200.0))?SPLL_SRF_Q(200.0):spll_obj.ylf[0];\
           /* VCO                              */ \
           spll_obj.fo=spll_obj.fn+spll_obj.ylf[0]; \

spll_obj.theta[0]=spll_obj.theta[1]+SPLL_SRF_Qmpy(SPLL_SRF_Qmpy(spll_obj.fo,spll_obj.delta_T),SPLL
_SRF_Q(2*3.1415926));                                                  \
           if(spll_obj.theta[0]>SPLL_SRF_Q(2*3.1415926)) \
                      spll_obj.theta[0]=spll_obj.theta[0]-SPLL_SRF_Q(2*3.1415926); \
           spll_obj.theta[1]=spll_obj.theta[0];\
#endif /* SPLL_3ph_SRF_IQ_H_ */
```

SPLL_3ph_SRF_IQ.c 源文件：

```
#include "Solar_IQ.h"

//*********** Structure Init Function ****//
void SPLL_3ph_SRF_IQ_init(int Grid_freq, long DELTA_T, SPLL_3ph_SRF_IQ *spll_obj)
{
           spll_obj->v_q[0]=SPLL_SRF_Q(0.0);
           spll_obj->v_q[1]=SPLL_SRF_Q(0.0);

           spll_obj->ylf[0]=SPLL_SRF_Q(0.0);
           spll_obj->ylf[1]=SPLL_SRF_Q(0.0);
```

```
            spll_obj->fo=SPLL_SRF_Q(0.0);
            spll_obj->fn=SPLL_SRF_Q(Grid_freq);

            spll_obj->theta[0]=SPLL_SRF_Q(0.0);
            spll_obj->theta[1]=SPLL_SRF_Q(0.0);

            // loop filter coefficients for 20kHz
            spll_obj->lpf_coeff.B0_lf=_IQ21(166.9743);
            spll_obj->lpf_coeff.B1_lf=_IQ21(-166.266);
            spll_obj->lpf_coeff.A1_lf=_IQ21(-1.0);

            spll_obj->delta_T=DELTA_T;
}

//*********** Function Definition ********//
void SPLL_3ph_SRF_IQ_FUNC(SPLL_3ph_SRF_IQ *spll_obj)
{
            //update v_q[0] before calling the routine
            //---------------------------------//
            // Loop Filter                     //
            //---------------------------------//
            spll_obj->ylf[0]=spll_obj->ylf[1]+SPLL_SRF_Qmpy(spll_obj->lpf_coeff.B0_lf,spll_obj-
>v_q[0])+SPLL_SRF_Qmpy(spll_obj->lpf_coeff.B1_lf,spll_obj->v_q[1]);
            spll_obj->ylf[1]=spll_obj->ylf[0];
            spll_obj->v_q[1]=spll_obj->v_q[0];

            spll_obj->ylf[0]=(spll_obj->ylf[0]>SPLL_SRF_Q(200.0))?SPLL_SRF_Q(200.0):spll_obj-
>ylf[0];
            //---------------------------------//
            // VCO                             //
            //---------------------------------//
            spll_obj->fo=spll_obj->fn+spll_obj->ylf[0];

            spll_obj->theta[0]=spll_obj->theta[1]+SPLL_SRF_Qmpy(SPLL_SRF_Qmpy(spll_obj-
>fo,spll_obj->delta_T),SPLL_SRF_Q(2*3.1415926));
            if(spll_obj->theta[0]>SPLL_SRF_Q(2*3.1415926))
                        spll_obj->theta[0]=spll_obj->theta[0]-SPLL_SRF_Q(2*3.1415926);

            spll_obj->theta[1]=spll_obj->theta[0];
}
```

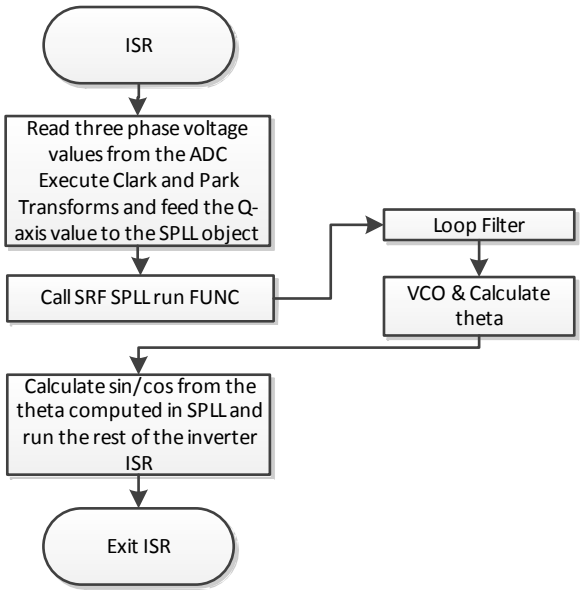


图 **5. SRF SPLL** 使用流程图

为了在终端应用中使用这个块，为 SPLL 结构、环路滤波器系数和陷波滤波器系数声明对象。

```
// ------------- Software PLL for Grid Tie Applications ----------
SPLL_3ph_SRF_IQ spll1;
```

以 ISR 的频率调用 SPLL_3ph_SRF_init 例程，从 SPLL 将以参数和电网频率执行，然后调用陷波滤波器更新系数更新例程。

```
SPLL_3ph_SRF_init(GRID_FREQ,_IQ21((float)(1.0/ISR_FREQUENCY)),&spll1;
```

在 ISR 中，从 ADC 中读取正弦输入电压，并且将其馈入 SPLL 块；写入具有当前电网角正弦值的反正弦 (invsine) 值。 然后，这可被用于控制操作。

```
abc_dq0_pos1.a = _IQmpy(GridMeas1,_IQ(0.5));
abc_dq0_pos1.b = _IQmpy(GridMeas2,_IQ(0.5));
abc_dq0_pos1.c = _IQmpy(GridMeas3,_IQ(0.5));
abc_dq0_pos1.sin=_IQsin((spll1.theta[1])<<3);// Q24 to Q21
abc_dq0_pos1.cos=_IQcos((spll1.theta[1])<<3);// Q24 to Q21
ABC_DQ0_POS_IQ_MACRO(abc_dq0_pos1);

// Q24 to Q21
spll1.v_q[0] = (int32)(_IQtoIQ21(abc_dq0_pos1.q));

// SPLL call
SPLL_3ph_SRF_IQ_FUNC(&spll1;
```

3 三相电网内的不均衡

此电网会受到不同条件的影响，这导致相位电压的不均衡。 根据对称分量理论，已知任何不均衡三相系统可被精简为两个对称系统和零分量：一个正向旋转，被称为正序列，另外一个负向旋转，被称为负序列（请见图 6）。 在下面部分中分析 Park 和 Clarke 变换上不均衡电压的运行方式。

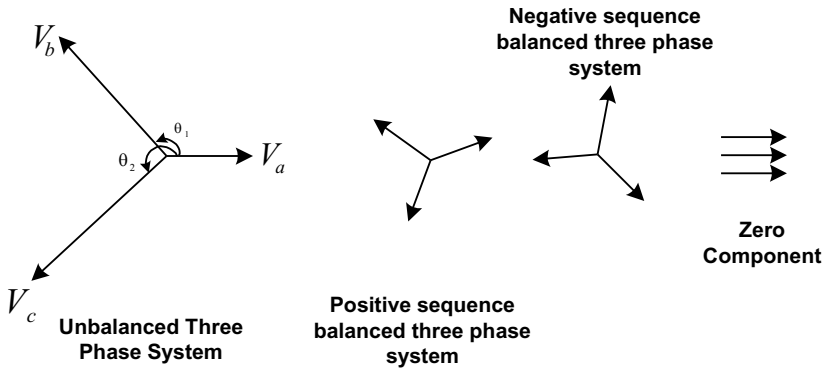


图 6. 三相电网内的不均衡

.

$$v = V^{+1}\begin{bmatrix}\cos(wt)\\ \cos(wt-2\pi/3)\\ \cos(wt-4\pi/3)\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(wt)\\ \cos(wt-4\pi/3)\\ \cos(wt-2\pi/3)\end{bmatrix} + V^{0}\begin{bmatrix}1\\1\\1\end{bmatrix} \quad (20)$$

通过使用 Clarke 变换，ω.v.t 正序列。

$$v_{\alpha\beta} = T_{abc->\alpha\beta} * v = \frac{2}{3}\begin{bmatrix}1 & -\frac{1}{2} & -\frac{1}{2}\\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2}\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix} * v = V^{+1}\begin{bmatrix}\cos(wt)\\ \sin(wt)\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(-wt)\\ \sin(-wt)\end{bmatrix} \quad (21)$$

此外，通过获取旋转基准框上的 Park 变换和投影，可以观察到任何负向序列分量在正序列旋转框坐标轴以两倍的频率出现，反之亦然。

$$v_{dq+} = T_{abc->dq0+} * v_{\alpha\beta} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix} * \left(V^{+1}\begin{bmatrix} \cos(wt) \\ \sin(wt) \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(-wt) \\ \sin(-wt) \end{bmatrix} \right) = V^{+1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(-2wt) \\ \sin(-2wt) \end{bmatrix}$$

$$v_{dq-} = T_{abc->dq0-} * v_{\alpha\beta} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{bmatrix} * \left(V^{+1}\begin{bmatrix} \cos(wt) \\ \sin(wt) \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(-wt) \\ \sin(-wt) \end{bmatrix} \right) = V^{+1}\begin{bmatrix} \cos(-2wt) \\ \sin(-2wt) \end{bmatrix} + V^{-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (22)$$

这导致电网角的估算误差，并且需要在为三相并网应用设计一个锁相环的同时将其考虑在内。

4 去耦合双同步基准框 PLL

如 3 节 中所示，不均衡三相系统可被换算为两个均衡三相系统：一个以正序列旋转，另外一个以负序列旋转 [1]。 在 PLL 已经被锁定前，正和负矢量可被写为 公式 23：

$$v = V^{+1}\begin{bmatrix} \cos(wt+\varphi_1) \\ \cos(wt-2\pi/3+\varphi_1) \\ \cos(wt-4\pi/3+\varphi_1) \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(wt+\varphi_{-1}) \\ \cos(wt-4\pi/3+\varphi_{-1}) \\ \cos(wt-2\pi/3+\varphi_{-1}) \end{bmatrix} + V^{0}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (23)$$

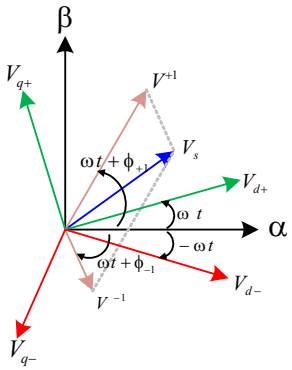


图 7. 不均衡三相系统的正和负序列

使用 Park 变换：

$$v_{\alpha\beta} = V^{+1}\begin{bmatrix} \cos(\omega t+\varphi_{+1}) \\ \sin(\omega t+\varphi_{+1}) \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(-\omega t+\varphi_{-1}) \\ \sin(-\omega t+\varphi_{-1}) \end{bmatrix} \quad (24)$$

正交系统在正循环的旋转基准框上的投影可写为 公式 25：

$$v_{dq+} = \left(V^{+1}\begin{bmatrix} \cos(\omega t+\varphi_{+1}) \\ \sin(\omega t+\varphi_{+1}) \end{bmatrix} + V^{-1}\begin{bmatrix} \cos(-\omega t+\varphi_{-1}) \\ \sin(-\omega t+\varphi_{-1}) \end{bmatrix} \right) * \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix} \quad (25)$$

精简化，

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(-\omega t+\varphi_{-1})\\ \sin(-\omega t+\varphi_{-1})\end{bmatrix} * \begin{bmatrix}\cos(\omega t) & \sin(\omega t)\\ -\sin(\omega t) & \cos(\omega t)\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(-\omega t+\varphi_{-1})\\ \sin(-\omega t+\varphi_{-1})\end{bmatrix} * \begin{bmatrix}\cos(\omega t) & \sin(\omega t)\\ -\sin(\omega t) & \cos(\omega t)\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(-\omega t+\varphi_{-1})\cos(\omega t)+\sin(-\omega t+\varphi_{-1})\sin(\omega t)\\ -\cos(-\omega t+\varphi_{-1})\sin(\omega t)+\sin(-\omega t+\varphi_{-1})\cos(\omega t)\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(-2\omega t+\varphi_{-1})\\ \sin(-2\omega t+\varphi_{-1})\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\begin{bmatrix}\cos(\varphi_{-1})\cos(2\omega t)+\sin(\varphi_{-1})\sin(2\omega t)\\ \sin(\varphi_{-1})\cos(2\omega t)-\cos(\varphi_{-1})\sin(2\omega t)\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + V^{-1}\cos(\varphi_{-1})\begin{bmatrix}\cos(2\omega t)\\ -\sin(2\omega t)\end{bmatrix} + V^{-1}\sin(\varphi_{-1})\begin{bmatrix}\sin(2\omega t)\\ \cos(2\omega t)\end{bmatrix}\right)$$

$$\Rightarrow v_{dq+} = \left(V^{+1}\begin{bmatrix}\cos(\varphi_{+1})\\ \sin(\varphi_{+1})\end{bmatrix} + \bar{v}_{d-}\begin{bmatrix}\cos(2\omega t)\\ -\sin(2\omega t)\end{bmatrix} + \bar{v}_{q-}\begin{bmatrix}\sin(2\omega t)\\ \cos(2\omega t)\end{bmatrix}\right) \quad (26)$$

d 和 q 坐标轴分量可写为：

$$v_{d+\_decoupled} = V^{+1}\cos(\varphi_{+1}) = v_{d+} - \bar{v}_{d-}\cos(2\omega t) - \bar{v}_{q-}\sin(2\omega t)$$

$$v_{q+\_decoupled} = V^{+1}\sin(\varphi_{+1}) = v_{q+} + \bar{v}_{d-}\sin(2\omega t) - \bar{v}_{q-}\cos(2\omega t) \quad (27)$$

一个去耦合网络被用来消除这些值 [1]。

一旦去耦合，可使用设计 SRF PLL 时所用到的 q 分量的同样属性（请见 2.1 节），在这里，正序列 q 坐标轴分量，当 PLL 被锁定时，它为零，或者是与角度误差成线性关系的值。 在 图 8 中显示了使用去耦合网络的 PLL。

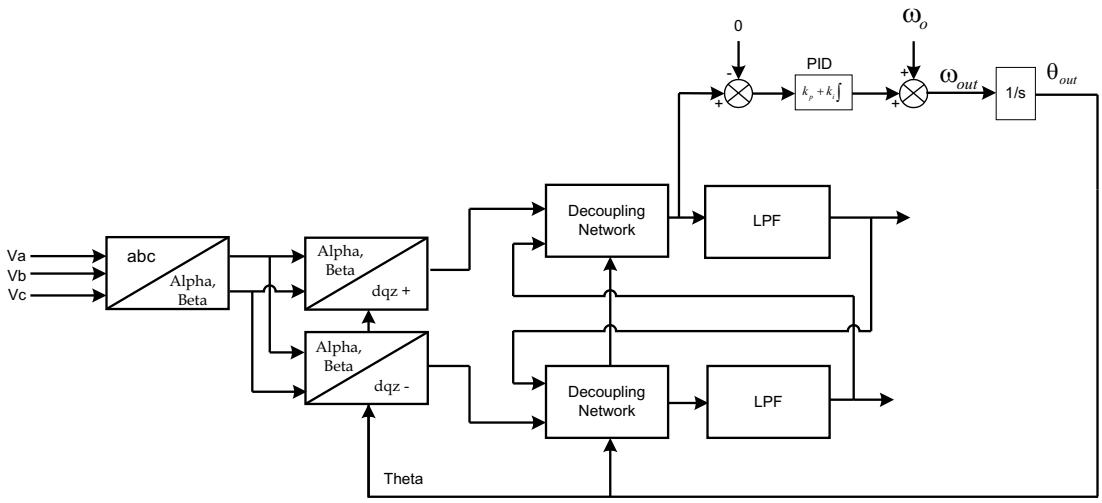


图 8. DDSRF PLL 结构

4.1 低通滤波器的离散执行

典型低通滤波器转换函数给出如下：

$$LPF(s) = \frac{\omega_f}{(s + \omega_f)} \tag{28}$$

在模拟域中，现在使用双线性变换：

$$LPF(z) = \frac{\omega_f}{(\frac{2}{T}\frac{(z-1)}{(z+1)} + \omega_f)} = \frac{\frac{\omega_f T}{(2 + T\omega_f)}(z+1)}{(z + \frac{(\omega_f T - 2)}{(\omega_f T + 2)})} = \frac{k_1(z+1)}{(z + k_2)} \tag{29}$$

在这里，T 是数字低通滤波器的运行采样周期。 使用 T = 1/(10Khz)=0.0001 和 [1] 中的讨论，可知 $\frac{\omega_f}{\omega} < \frac{1}{\sqrt{2}}$ 用于 PLL 的稳定响应，因此，选择：

$\omega_f = 30$

得出：

$k_1 = 0.00933678$，$k_2 = -0.9813264$

4.2 针对不同的条件，模拟锁相环

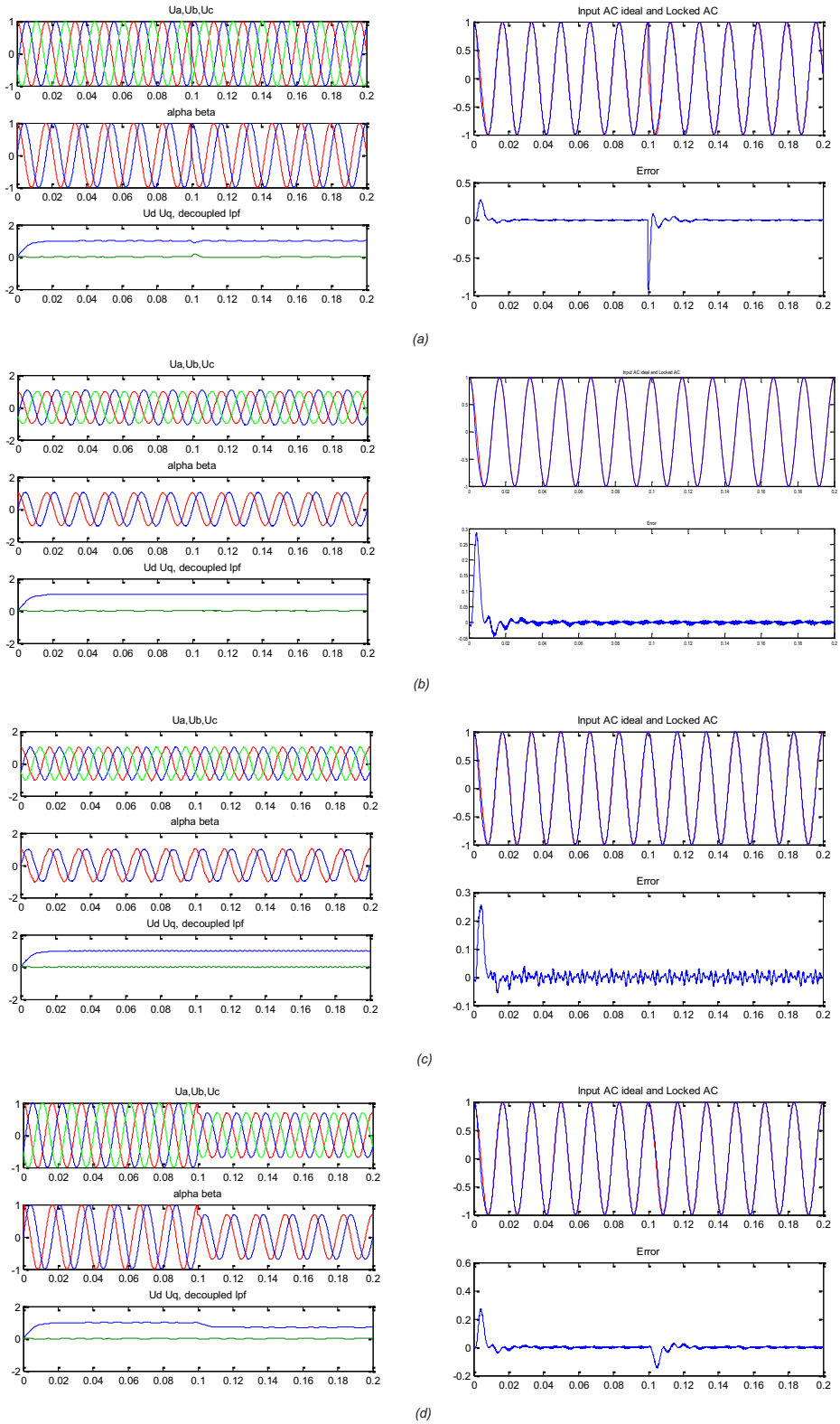
在编写 SPLL 结构之前，针对电网上的不同条件，模拟 PLL 的运行方式很重要。 定点处理器被用来降低很多并网转换器的成本。 IQ Math 是一个便捷的途径来观察具有小数点的定点数。 C2000 IQ math 库提供内置函数，此函数可简化编程人员对小数点的处理。 首先，MATLAB 被用来模拟和识别此算法需要运行的 Q 点。 下面是使用定点工具箱的 MATLAB 脚本，此脚本测试不同电网条件下的 PLL 算法。

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This software is licensed for use with Texas Instruments C28x
%  family DSCs.  This license was provided to you prior to installing
%  the software.
% ------------------------------------------------------------------------
%          Copyright (C) 2010-2012 Texas Instruments, Incorporated.
%                          All Rights Reserved.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

%Select numeric type,
T=numerictype('WordLength',32,'FractionLength',22);

%Specify math attributes to the fimath object
F=fimath('RoundMode','floor','OverflowMode','wrap');
F.ProductMode='SpecifyPrecision';
F.ProductWordLength=32;
F.ProductFractionLength=22;
F.SumMode='SpecifyPrecision';
F.SumWordLength=32;
F.SumFractionLength=22;
%specify fipref object, to display warning in cases of overflow and
%underflow
P=fipref;
P.LoggingMode='on';
P.NumericTypeDisplay='none';
P.FimathDisplay='none';

%PLL Modelling starts from here
Fs=10000;
Ts=1/Fs;            %Sampling Time = (1/fs) , Note Ts is related to the frequency
                    % the ISR would run in the solar application as well, assuming
                    % 10Khz control loop for the inverter

Tfinal=0.2;         % Total time the simulation is run for
t=0:Ts:Tfinal;      % time vector
GridFreq=60;        % nominal grid frequency
wn=2*pi*GridFreq;   % nominal frequency in radians
fn=GridFreq;

%generate input signal
wu=2*pi*GridFreq;
theta=rem((t*2*pi*GridFreq),2*pi);
L=length(t);

%generate input signal and create a fi object of it

%CASE 1 : Phase Jump at the Mid Point

for n=1:floor(L/2)
    Ua(n)=cos(theta(n));
    Ub(n)=cos(theta(n)-2*pi/3);
    Uc(n)=cos(theta(n)-4*pi/3);
end
for n=floor(L/2):L
    Ua(n)=cos(theta(n)+1.5);
```

```
    Ub(n)=cos(theta(n)-2*pi/3+1.5);
    Uc(n)=cos(theta(n)-4*pi/3+1.5);
end
Ua_ideal=Ua;

%CASE 2: Unbalanced Grid

% for n=1:floor(L)
% Ua(n)=cos(theta(n));
% Ub(n)=1.1*cos(theta(n)-2*pi/3);
% Uc(n)=cos(theta(n)-4*pi/3);
% end
% Ua_ideal=cos(theta);

%CASE 3: voltage harmonics
% for n=1:floor(L)
% Ua(n)=cos(theta(n))+0.05*cos(5*theta(n));
% Ub(n)=cos(theta(n)-2*pi/3)+0.05*(cos(5*(theta(n)-2*pi/3)));
% Uc(n)=cos(theta(n)-4*pi/3)+0.05*(cos(5*(theta(n)-4*pi/3)));
% end
% Ua_ideal=cos(theta);

% CASE 4: voltage dips and swells
% %
% for n=1:floor(L/2)
%     Ua(n)=cos(theta(n));
%     Ub(n)=cos(theta(n)-2*pi/3);
%     Uc(n)=cos(theta(n)-4*pi/3);
% end
% for n=floor(L/2):L
%     Ua(n)=0.7*cos(theta(n));
%     Ub(n)=0.7*cos(theta(n)-2*pi/3);
%     Uc(n)=0.7*cos(theta(n)-4*pi/3);
% end
% Ua_ideal=cos(theta);

Ualpha=[0,0];
Ubeta=[0,0];

Ud_plus=[0,0];
Uq_plus=[0,0];

Ud_minus=[0,0];
Uq_minus=[0,0];

Ud_plus_decoupl=[0,0];
Uq_plus_decoupl=[0,0];

Ud_minus_decoupl=[0,0];
Uq_minus_decoupl=[0,0];

Ud_plus_decoupl_lpf=[0,0];
Uq_plus_decoupl_lpf=[0,0];

Ud_minus_decoupl_lpf=[0,0];
Uq_minus_decoupl_lpf=[0,0];

y=[0,0];
x=[0,0];
w=[0,0];
z=[0,0];

ylf =[0,0];
u_q =[0,0];
Theta=[0,0,0];
fo=0;
```

```
Ua=fi(Ua,T,F);
Ub=fi(Ub,T,F);
Uc=fi(Uc,T,F);
Ua_ideal=fi(Ua_ideal,T,F);

%declare arrays used by the PLL process
Ualpha  =fi(Ualpha,T,F);
Ubeta  =fi(Ubeta,T,F);
Ud_plus  =fi(Ud_plus,T,F);
Uq_plus  =fi(Uq_plus,T,F);
Ud_minus  =fi(Ud_minus,T,F);
Uq_minus  =fi(Uq_minus,T,F);
Ud_plus_decoupl  =fi(Ud_plus_decoupl,T,F);
Uq_plus_decoupl  =fi(Uq_plus_decoupl,T,F);
Ud_minus_decoupl =fi(Ud_minus_decoupl,T,F);
Uq_minus_decoupl =fi(Uq_minus_decoupl,T,F);
Ud_plus_decoupl_lpf  =fi(Ud_plus_decoupl_lpf,T,F);
Uq_plus_decoupl_lpf  =fi(Uq_plus_decoupl_lpf,T,F);
Ud_minus_decoupl_lpf =fi(Ud_minus_decoupl_lpf,T,F);
Uq_minus_decoupl_lpf =fi(Uq_minus_decoupl_lpf,T,F);
y=fi(y,T,F);
x=fi(x,T,F);
w=fi(w,T,F);
z=fi(z,T,F);
ylf=fi(ylf,T,F);
u_q=fi(u_q,T,F);
Theta=fi(Theta,T,F);
fo=fi(fo,T,F);

% simulate the PLL process
for n=3:L    % No of iteration of the PLL process in the simulation time


    Ualpha(n) =fi((2.0/3.0),T,F)*(Ua(n)-fi(0.5,T,F)*(Ub(n)+Uc(n)));
    Ubeta(n)  =fi((2.0/3.0)*0.866,T,F)*(Ub(n)-Uc(n));

    s1=sin(Theta(n));
    c1=cos(Theta(n));
    s2=sin(fi(2,T,F)*Theta(n));
    c2=cos(fi(2,T,F)*Theta(n));

    Ud_plus(n)=Ualpha(n)*c1+Ubeta(n)*s1;
    Uq_plus(n)=-Ualpha(n)*s1+Ubeta(n)*c1;

    Ud_minus(n)=Ualpha(n)*c1-Ubeta(n)*s1;
    Uq_minus(n)=Ualpha(n)*s1+Ubeta(n)*c1;

    Ud_plus_decoupl(n)=Ud_plus(n)-Ud_minus_decoupl_lpf(n-1)*c2-Uq_minus_decoupl_lpf(n-1)*s2;
    Uq_plus_decoupl(n)=Uq_plus(n)+Ud_minus_decoupl_lpf(n-1)*s2-Uq_minus_decoupl_lpf(n-1)*c2;

    Ud_minus_decoupl(n)=Ud_minus(n)-Ud_plus_decoupl_lpf(n-1)*c2+Uq_plus_decoupl_lpf(n-1)*s2;
    Uq_minus_decoupl(n)=Uq_minus(n)-Ud_plus_decoupl_lpf(n-1)*s2-Uq_plus_decoupl_lpf(n-1)*c2;

k1=fi(0.00933678,T,F);
k2=fi(-0.9813264,T,F);

    y(n)=Ud_plus_decoupl(n)*k1-k2*y(n-1);
    Ud_plus_decoupl_lpf(n) = y(n)+y(n-1);

    x(n)=Uq_plus_decoupl(n)*k1-k2*x(n-1);
    Uq_plus_decoupl_lpf(n) = x(n)+x(n-1);

    w(n)=Ud_minus_decoupl(n)*k1-k2*w(n-1);
    Ud_minus_decoupl_lpf(n) = w(n)+w(n-1);
```

```
        z(n)=Uq_minus_decoupl(n)*k1-k2*z(n-1);
        Uq_minus_decoupl_lpf(n) = z(n)+z(n-1);


        u_q(1)=Uq_plus_decoupl(n);

        %Loop Filter
        ylf(1)=ylf(2)+fi(167.9877775,T,F)*u_q(1)-fi(165.2122225,T,F)*u_q(2);

        %update u_q for future use
        u_q(2)=u_q(1);

        ylf(2)=ylf(1);

        %update output frequency
        fo=fn+ylf(1);

        Theta(n+1)=Theta(n)+fi(2*pi,T,F)*(fo*Ts);

        Theta(n+1)=Theta(n)+fi(2*pi,T,F)*(fo*Ts);

        if(Theta(n+1)>=(fi(2*pi,T,F)))
            Theta(n+1)=Theta(n+1)-fi(2*pi,T,F);
        end

    end

    figure,subplot(3,1,1),plot(t,Ua,'r',t,Ub,'b',t,Uc,'g'),title('Ua,Ub,Uc');
    subplot(3,1,2),plot(t,Ualpha,'r',t,Ubeta),title('alpha beta');
    subplot(3,1,3),plot(t,Ud_plus_decoupl_lpf(1:L),t,Uq_plus_decoupl_lpf(1:L)), title('Ud Uq,
    decoupled lpf') ;

    figure,subplot(2,1,1),plot(t,Ua_ideal,'r',t,cos(Theta(1:L)),'b'),title('Input AC ideal and Locked
    AC');
    subplot(2,1,2),plot(t,Ua_ideal-cos(Theta(1:L))),title('Error');
```



A 90 度的相位跳变

B 一个相位上 10% 的电压不均衡

C 5% 5th 谐波含量

D 振幅变化（电压暂降和骤降）

图 **9.** 不同电网条件下的 **PLL** 响应

4.3 在使用 IQ Math 的 C2000 控制器上执行 PLL

如上面的部分所示，由于普通逆变器通常使用 IQ24，为 PLL 选择的 Q 点为 IQ22，PLL 的代码可被表示如下。

SPLL_3ph_DDSRF_IQ.h 头文件：

```
#ifndef SPLL_3ph_DDSRF_IQ_H_
#define SPLL_3ph_DDSRF_IQ_H_

#define SPLL_DDSRF_Q _IQ22
#define SPLL_DDSRF_Qmpy _IQ22mpy

//*********** Structure Definition ********//
typedef struct{
            int32    B1_lf;
            int32    B0_lf;
            int32    A1_lf;
}SPLL_3ph_DDSRF_IQ_LPF_COEFF;

typedef struct{
            int32 d_p;
            int32 d_n;
            int32 q_p;
            int32 q_n;

            int32 d_p_decoupl;
            int32 d_n_decoupl;
            int32 q_p_decoupl;
            int32 q_n_decoupl;

            int32 cos_2theta;
            int32 sin_2theta;

            int32 y[2];
            int32 x[2];
            int32 w[2];
            int32 z[2];
            int32 k1;
            int32 k2;
            int32 d_p_decoupl_lpf;
            int32 d_n_decoupl_lpf;
            int32 q_p_decoupl_lpf;
            int32 q_n_decoupl_lpf;

            int32 v_q[2];
            int32 theta[2];
            int32 ylf[2];
            int32 fo;
            int32 fn;
            int32 delta_T;
            SPLL_3ph_DDSRF_IQ_LPF_COEFF lpf_coeff;
}SPLL_3ph_DDSRF_IQ;

//*********** Function Declarations *******//
void SPLL_3ph_DDSRF_IQ_init(int Grid_freq, long DELTA_T, long k1, long k2, SPLL_3ph_DDSRF_IQ
*spll);
void SPLL_3ph_DDSRF_IQ_FUNC(SPLL_3ph_DDSRF_IQ *spll_obj);

//*********** Macro Definition **********//
#define SPLL_3ph_DDSRF_IQ_MACRO(spll_obj)
                                                                                    \
            spll_obj.d_p_decoupl=spll_obj.d_p -
 SPLL_DDSRF_Qmpy(spll_obj.d_n_decoupl_lpf,spll_obj.cos_2theta) -
 SPLL_DDSRF_Qmpy(spll_obj.q_n_decoupl,spll_obj.sin_2theta);     \
            spll_obj.q_p_decoupl=spll_obj.q_p +
```

```
SPLL_DDSRF_Qmpy(spll_obj.d_n_decoupl_lpf,spll_obj.sin_2theta) -
 SPLL_DDSRF_Qmpy(spll_obj.q_n_decoupl,spll_obj.cos_2theta);     \
            spll_obj.d_n_decoupl=spll_obj.d_n -
 SPLL_DDSRF_Qmpy(spll_obj.d_p_decoupl_lpf,spll_obj.cos_2theta) +
SPLL_DDSRF_Qmpy(spll_obj.q_p_decoupl,spll_obj.sin_2theta);      \
            spll_obj.q_n_decoupl=spll_obj.q_n -
 SPLL_DDSRF_Qmpy(spll_obj.d_p_decoupl_lpf,spll_obj.sin_2theta) -
 SPLL_DDSRF_Qmpy(spll_obj.q_p_decoupl,spll_obj.cos_2theta);      \
            spll_obj.y[1]=SPLL_DDSRF_Qmpy(spll_obj.d_p_decoupl,spll_obj.k1)-
SPLL_DDSRF_Qmpy(spll_obj.y[0],spll_obj.k2); \
            spll_obj.d_p_decoupl_lpf=spll_obj.y[1]+spll_obj.y[0];  \
            spll_obj.y[0]=spll_obj.y[1]; \
            spll_obj.x[1]=SPLL_DDSRF_Qmpy(spll_obj.q_p_decoupl,spll_obj.k1)-
SPLL_DDSRF_Qmpy(spll_obj.x[0],spll_obj.k2); \
            spll_obj.q_p_decoupl_lpf=spll_obj.x[1]+spll_obj.x[0]; \
            spll_obj.x[0]=spll_obj.x[1];  \
            spll_obj.w[1]=SPLL_DDSRF_Qmpy(spll_obj.d_n_decoupl,spll_obj.k1)-
SPLL_DDSRF_Qmpy(spll_obj.w[0],spll_obj.k2);  \
            spll_obj.d_n_decoupl_lpf=spll_obj.w[1]+spll_obj.w[0]; \
            spll_obj.w[0]=spll_obj.w[1]; \
            spll_obj.z[1]=SPLL_DDSRF_Qmpy(spll_obj.q_n_decoupl,spll_obj.k1)-
SPLL_DDSRF_Qmpy(spll_obj.z[0],spll_obj.k2);  \
            spll_obj.q_n_decoupl_lpf=spll_obj.z[1]+spll_obj.z[0];  \
            spll_obj.z[0]=spll_obj.z[1]; \
            spll_obj.v_q[0]=spll_obj.q_p_decoupl; \

spll_obj.ylf[0]=spll_obj.ylf[1]+SPLL_DDSRF_Qmpy(spll_obj.lpf_coeff.B0_lf,spll_obj.v_q[0])+SPLL_DDS
RF_Qmpy(spll_obj.lpf_coeff.B1_lf,spll_obj.v_q[1]);                                          \
            spll_obj.ylf[1]=spll_obj.ylf[0]; \
            spll_obj.v_q[1]=spll_obj.v_q[0]; \
            spll_obj.fo=spll_obj.fn+spll_obj.ylf[0];


            spll_obj.theta[0]=spll_obj.theta[1]+SPLL_DDSRF_Qmpy
            (SPLL_DDSRF_Qmpy(spll_obj.fo,spll_obj.delta_T),SPLL_DDSRF_Q(2*3.1415926));
            if(spll_obj.theta[0]>SPLL_DDSRF_Q(2*3.1415926)) \
                        spll_obj.theta[0]=spll_obj.theta[0]-SPLL_DDSRF_Q(2*3.1415926); \
            spll_obj.theta[1]=spll_obj.theta[0]; \
#endif /* SPLL_3ph_DDSRF_IQ_H_ */
```

SPLL_3ph_DDSRF_IQ.c 源文件：

```
#include "Solar_IQ.h"

//*********** Structure Init Function ****//
void SPLL_3ph_DDSRF_IQ_init(int Grid_freq, long DELTA_T, long k1, long k2, SPLL_3ph_DDSRF_IQ
*spll_obj)
{
            spll_obj->d_p=SPLL_DDSRF_Q(0.0);
            spll_obj->d_n=SPLL_DDSRF_Q(0.0);

            spll_obj->q_p=SPLL_DDSRF_Q(0.0);
            spll_obj->q_n=SPLL_DDSRF_Q(0.0);

            spll_obj->d_p_decoupl=SPLL_DDSRF_Q(0.0);
            spll_obj->d_n_decoupl=SPLL_DDSRF_Q(0.0);

            spll_obj->q_p_decoupl=SPLL_DDSRF_Q(0.0);
            spll_obj->q_n_decoupl=SPLL_DDSRF_Q(0.0);

            spll_obj->d_p_decoupl_lpf=SPLL_DDSRF_Q(0.0);
            spll_obj->d_n_decoupl_lpf=SPLL_DDSRF_Q(0.0);

            spll_obj->q_p_decoupl_lpf=SPLL_DDSRF_Q(0.0);
            spll_obj->q_n_decoupl_lpf=SPLL_DDSRF_Q(0.0);
```

```
            spll_obj->y[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->y[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->x[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->x[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->w[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->w[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->z[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->z[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->k1=k1;
            spll_obj->k2=k2;

            spll_obj->v_q[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->v_q[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->ylf[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->ylf[1]=SPLL_DDSRF_Q(0.0);

            spll_obj->fo=SPLL_DDSRF_Q(0.0);
            spll_obj->fn=SPLL_DDSRF_Q(Grid_freq);

            spll_obj->theta[0]=SPLL_DDSRF_Q(0.0);
            spll_obj->theta[1]=SPLL_DDSRF_Q(0.0);

            // loop filter coefficients for 20kHz
            spll_obj->lpf_coeff.B0_lf=_IQ22(166.9743);
            spll_obj->lpf_coeff.B1_lf=_IQ22(-166.266);
            spll_obj->lpf_coeff.A1_lf=_IQ22(-1.0);

            spll_obj->delta_T=DELTA_T;
}

//*********** Function Definition ********//
void SPLL_3ph_DDSRF_IQ_FUNC(SPLL_3ph_DDSRF_IQ *spll_obj)
{
            // before calling this routine run the ABC_DQ0_Pos_Neg and update the values for
d_p,d_n,q_p,q_n
            // and update the cos_2theta and sin_2theta values with the prev angle

            //-------------------------//
            // Decoupling Network      //
            //-------------------------//
            spll_obj->d_p_decoupl=spll_obj->d_p - SPLL_DDSRF_Qmpy(spll_obj-
>d_n_decoupl_lpf,spll_obj->cos_2theta) - SPLL_DDSRF_Qmpy(spll_obj->q_n_decoupl,spll_obj-
>sin_2theta);
            spll_obj->q_p_decoupl=spll_obj->q_p + SPLL_DDSRF_Qmpy(spll_obj-
>d_n_decoupl_lpf,spll_obj->sin_2theta) - SPLL_DDSRF_Qmpy(spll_obj->q_n_decoupl,spll_obj-
>cos_2theta);

            spll_obj->d_n_decoupl=spll_obj->d_n - SPLL_DDSRF_Qmpy(spll_obj-
>d_p_decoupl_lpf,spll_obj->cos_2theta) + SPLL_DDSRF_Qmpy(spll_obj->q_p_decoupl,spll_obj-
>sin_2theta);
            spll_obj->q_n_decoupl=spll_obj->q_n - SPLL_DDSRF_Qmpy(spll_obj-
>d_p_decoupl_lpf,spll_obj->sin_2theta) - SPLL_DDSRF_Qmpy(spll_obj->q_p_decoupl,spll_obj-
>cos_2theta);

            //-------------------------//
            // Low pass filter         //
            //-------------------------//

            spll_obj->y[1]=SPLL_DDSRF_Qmpy(spll_obj->d_p_decoupl,spll_obj->k1)-
SPLL_DDSRF_Qmpy(spll_obj->y[0],spll_obj->k2);
            spll_obj->d_p_decoupl_lpf=spll_obj->y[1]+spll_obj->y[0];
```

```
            spll_obj->y[0]=spll_obj->y[1];

            spll_obj->x[1]=SPLL_DDSRF_Qmpy(spll_obj->q_p_decoupl,spll_obj->k1)-
SPLL_DDSRF_Qmpy(spll_obj->x[0],spll_obj->k2);
            spll_obj->q_p_decoupl_lpf=spll_obj->x[1]+spll_obj->x[0];
            spll_obj->x[0]=spll_obj->x[1];

            spll_obj->w[1]=SPLL_DDSRF_Qmpy(spll_obj->d_n_decoupl,spll_obj->k1)-
SPLL_DDSRF_Qmpy(spll_obj->w[0],spll_obj->k2);
            spll_obj->d_n_decoupl_lpf=spll_obj->w[1]+spll_obj->w[0];
            spll_obj->w[0]=spll_obj->w[1];

            spll_obj->z[1]=SPLL_DDSRF_Qmpy(spll_obj->q_n_decoupl,spll_obj->k1)-
SPLL_DDSRF_Qmpy(spll_obj->z[0],spll_obj->k2);
            spll_obj->q_n_decoupl_lpf=spll_obj->z[1]+spll_obj->z[0];
            spll_obj->z[0]=spll_obj->z[1];

            spll_obj->v_q[0]=spll_obj->q_p_decoupl;

            //---------------------------------//
            // Loop Filter                     //
            //---------------------------------//
            spll_obj->ylf[0]=spll_obj->ylf[1]+SPLL_DDSRF_Qmpy(spll_obj->lpf_coeff.B0_lf,spll_obj-
>v_q[0])+SPLL_DDSRF_Qmpy(spll_obj->lpf_coeff.B1_lf,spll_obj->v_q[1]);
            spll_obj->ylf[1]=spll_obj->ylf[0];
            spll_obj->v_q[1]=spll_obj->v_q[0];

            //---------------------------------//
            // VCO                             //
            //---------------------------------//
            spll_obj->fo=spll_obj->fn+spll_obj->ylf[0];

            spll_obj->theta[0]=spll_obj->theta[1]+SPLL_DDSRF_Qmpy(SPLL_DDSRF_Qmpy(spll_obj-
>fo,spll_obj->delta_T),SPLL_DDSRF_Q(2*3.1415926));
            if(spll_obj->theta[0]>SPLL_DDSRF_Q(2*3.1415926))
                        spll_obj->theta[0]=spll_obj->theta[0]-SPLL_DDSRF_Q(2*3.1415926);

vspll_obj->theta[1]=spll_obj->theta[0];
}
```

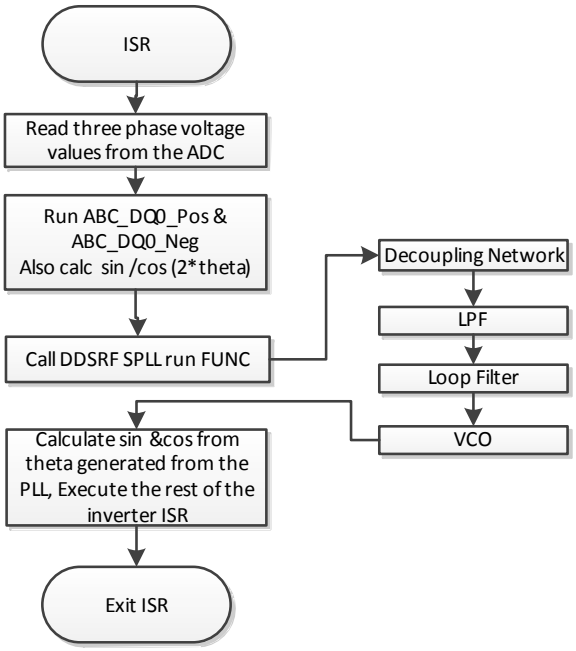


图 **10. DDSRF SPLL** 流程图

为了在终端应用中使用这个块，为 SPLL 结构、环路滤波器系数和陷波滤波器系数声明对象。

```
SPLL_3ph_DDSRF spll1;
```

以 ISR 的频率调用 SPLL_3ph_DDSRF_IQ_init 例程，在此例程中，SPLL 将以参数和电网频率执行，然后调用陷波滤波器更新系数更新例程。

```
SPLL_3ph_DDSRF_IQ_init(GRID_FREQ,_IQ22((float)(1.0/ISR_FREQUENCY)),SPLL_DDSRF_Q(0.00933678),
SPLL_DDSRF_Q(-0.9813264),&spll1);
```

在 ISR 中，从 ADC 中读取正弦输入电压，并且将其馈入 SPLL 块；写入具有当前电网角正弦值的 invsine 值。 然后，这可被用于控制操作。

```
spll1.d_p =abc_dq0_pos1.d>>2;          // Convert Q24 to Q22
spll1.q_p =abc_dq0_pos1.q<< 2;         // Convert Q24 to Q22
spll1.d_n =abc_dq0_neg1.d<<2;          // Convert Q24 to Q22
spll1.q_n =abc_dq0_neg1.q<<2;          // Convert Q24 to Q22
spll1.q_n =abc_dq0_neg1.q(&spll1);
spll1.cos_2theta = IQ22 cos (_IQ22mpy (_IQ22(2), Spll1.theta[0]));
Spll1.sin_2theta = IQ22 sin (_IQ22mpy (_IQ22(2), Spll1.theta[0]));
//Spll call
SPLL_3ph_DDSRF_IQ_FUNC (&Spll1);
```

5 太阳能库和 ControlSuite

C2000 为使用 C2000 器件的控制应用的开发提供了软件。 此软件可从 www.ti.com/controlSUITE 内下载。

诸如 IQ math 库和太阳能库等多种库文件简化了 C2000 器件上算法的设计和开发。 IQ math 库提供了一个将定点工具箱代码轻松转换为控制器代码的便捷途径。

此太阳能库提供在这个应用报告中讨论的软件块和更多信息。

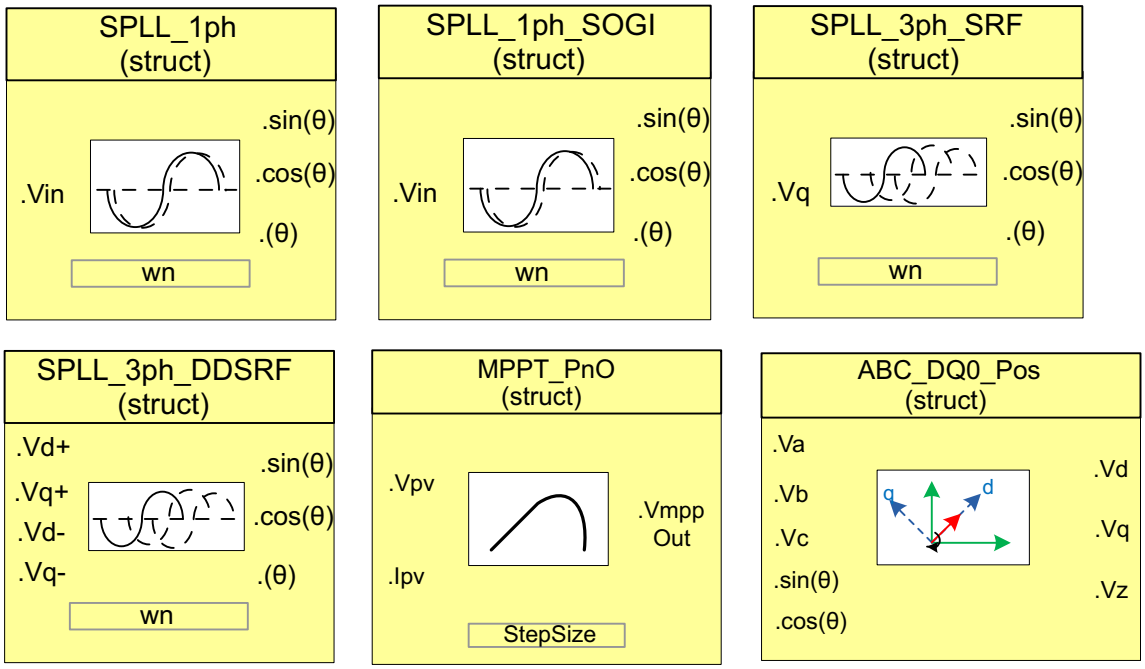


图 **11.** 太阳能库和 **ControlSuite**

6 参考书目

1. P. R. e. al，“针对功率转换器控制的去耦合双同步基准框 PLL，”电力电子学汇刊，vol. 22，no. 2，2007 年
2. Blaabjerg，F.，R. Teodorescu，M. Liserre 和 A. Timbus，“控制和同步配电生成系统概述，”工业用电子学汇刊，vol 53，no 5，2006 年 10 月
3. P. R. &. R. T. Marco Liserre，针对光伏和风能系统的电网逆变器，John Wiley & Sons Ltd.，2011 年
4. 《*TMS320F28032，TMS320F28033，TMS320F28034，TMS320F28035 Piccolo 微控制器数据手册*》（文献编号：SPRS584），可从：www.ti.com 内下载。

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122