```
#ifndef __FOC_tests_fixed_sim_hpp__
#definde __FOC_tests_fixed_sim_hpp__


#include "controlSuite_fixed_sim.hpp"



void FOC_tests_init(void)
{
        //One time initialization.

    init_cm1_fix();
    init_pi_spd_pi_id_pi_iq_fix();

}


void FOC_tests(void)
{
    // This function called by the simulator every 50 uSec.




    //From /home/maiello/Desktop/Development-Simulation/InductionMotor_DTC_SVM/
controlSUITE_ref/HVACI_Sensored/HVACI_Sensored.c

    // Using the the test labeled "LEVEL 5" as a reference...

    // ================================ LEVEL 5
====================================
    //  Level 5 verifies verifies the speed regulator performed by PI module.
    //  The system speed loop is closed by using the measured speed from capture
    //  signal as a feedback.
    //
================================================================================







    // Our equivilent connections to the simulator...

    // Inputs from the simulator:  is[0], is[1], is[2], omegad, omega_r
    // Outputs to the simulator:   vdd, vdq, theta_ctrl

    //
--------------------------------------------------------------------------------
    //  Connect inputs of the CLARKE module and call the clarke transformation
macro
    //
--------------------------------------------------------------------------------
        clarke1_fix.As = is[0];       // Phase A curr.
        clarke1_fix.Bs = is[1];       // Phase B curr.
        clarke1_fix.Cs = is[2];       // Phase C curr.
        CLARKE1_MACRO_fix(clarke1_fix)
```

```
    //
-------------------------------------------------------------------------------
    //  Connect inputs of the PARK module and call the park trans. macro
    //
-------------------------------------------------------------------------------
        park1_fix.Alpha = clarke1_fix.Alpha;
        park1_fix.Beta  = clarke1_fix.Beta;
        park1_fix.Angle = cm1_fix.Theta;
        park1_fix.Sine  = sin(fpt2fl(park1_fix.Angle) * 6.283185307);  // NOTE: In
the actual C2000 implementation the RG_MACRO was used in place with fpt2fl().
        park1_fix.Cosine= cos(fpt2fl(park1_fix.Angle) * 6.283185307);  //       The
code here is just to maintain equivalence with the double precision model.
        PARK_MACRO_fix(park1_fix)


        //
-------------------------------------------------------------------------------
        //  Connect inputs of the PI module and call the PID IQ controller macro
        //
-------------------------------------------------------------------------------

        pi_spd_fix.Ref = fl2fpt((float) omegad);
        pi_spd_fix.Fbk = fl2fpt((float) omega_r);
        PI_MACRO_fix(pi_spd_fix)


        //
-------------------------------------------------------------------------------
        //  Connect inputs of the PI module and call the PID IQ controller macro
        //
-------------------------------------------------------------------------------
        pi_iq_fix.Ref = pi_spd_fix.Out;
        pi_iq_fix.Fbk = park1_fix.Qs;
        PI_MACRO_fix(pi_iq_fix)

            //For this simple test, we fix the field current as a constant for the
entire simulation.
        IdRef_fix = fl2fpt(20);

        //
-------------------------------------------------------------------------------
        //  Connect inputs of the PI module and call the PID ID controller macro
        //
-------------------------------------------------------------------------------
        pi_id_fix.Ref = IdRef_fix;
        pi_id_fix.Fbk = park1_fix.Ds;
        PI_MACRO_fix(pi_id_fix)

        //
-------------------------------------------------------------------------------
        //  Connect inputs of the INV_PARK module and call the inverse park trans.
macro
        //  (In our model, pi_id connects to vdd and pi_iq connects to vqd. We use
are own SV PWM algorithm which takes care of INV_PARK calculation)
        //
-------------------------------------------------------------------------------
        vdd_fix = (double) fpt2fl(pi_id_fix.Out);
        vqd_fix = (double) fpt2fl(pi_iq_fix.Out);
        theta_ctrl_fix = (double) (fpt2fl(cm1_fix.Theta) * 6.283185307);
```

```
        //
-------------------------------------------------------------------------------
        //    Connect inputs of the CUR_MOD module and call the current model
        //    calculation function.
        //
-------------------------------------------------------------------------------
        cm1_fix.IDs = park1_fix.Ds;
        cm1_fix.IQs = park1_fix.Qs;
        cm1_fix.Wr = omega_r;
        CUR_MOD_MACRO_fix(cm1_fix)


        vdd = vdd_fix;
        vqd = vqd_fix;
        theta_ctrl = theta_ctrl_fix;

}




#endif
```