ACI1-1

AC Induction Motor Control
(Single phase)

Open Loop V/Hz Control

Closed Loop Speed Control

DC Bus Ripple Compensation

System Document

Digital Control Systems (DCS) Group

Texas Instruments

April 30, 2003

ACI1-1 – System overview		
Modules	BOX_CAR CAP_EVENT_DRV COMPWM DATA_LOG PID_ID FC_PWM_O_DRV ADC04U_DRV	RMP_CNTL SINCOSPH SINTB360 SPEED_PRD SYS_INIT V_HZ_PROFILE
System S/W	**ASM**	**"C"**
Program memory	1402 words	1915 words[1]
Data memory	205 words	495 words[1]
Development / Emulation	Code Composer Studio v2.2 (or above) with Real Time debug	
Target controller H/W	Spectrum Digital – F243 / F2407 EVMs , F2407 eZdsp*	
Power Inverter H/W	Spectrum Digital – DMC1500	
Motor Type	Single Phase Induction Motor	
PWM frequency	15 KHz (Timer T1 based)	
PWM mode	Symmetrical with Dead band	
Interrupts	1 (Timer 1 underflow)	
Main sampling loop	15KHz	
Peripheral Resources Used	Timer 1 & 2, PWM 1/2/3/4, 2 ADC Channels	

* Default target HW is F2407 eZdsp. If different target is used, the software may need to be modified.

[1] Note: The C version of the software at this time excludes the modules COMPWM, FC_PWM_O_DRV, and ADC04U_DRV. Instead, the module FC_PWM_DRV has been used. Also, the incremental build level 4 is not implemented for the C system.

System Overview

Figure 1 shows the complete system configuration for a single-phase ACI motor drive. A full H-bridge is utilized to control two phase voltages of the motor. Four PWM channels from the DSP control four power devices of the inverter. Two analog to digital channels are utilized to measure DC bus voltages. One capture input is needed to measure the speed of the drive.

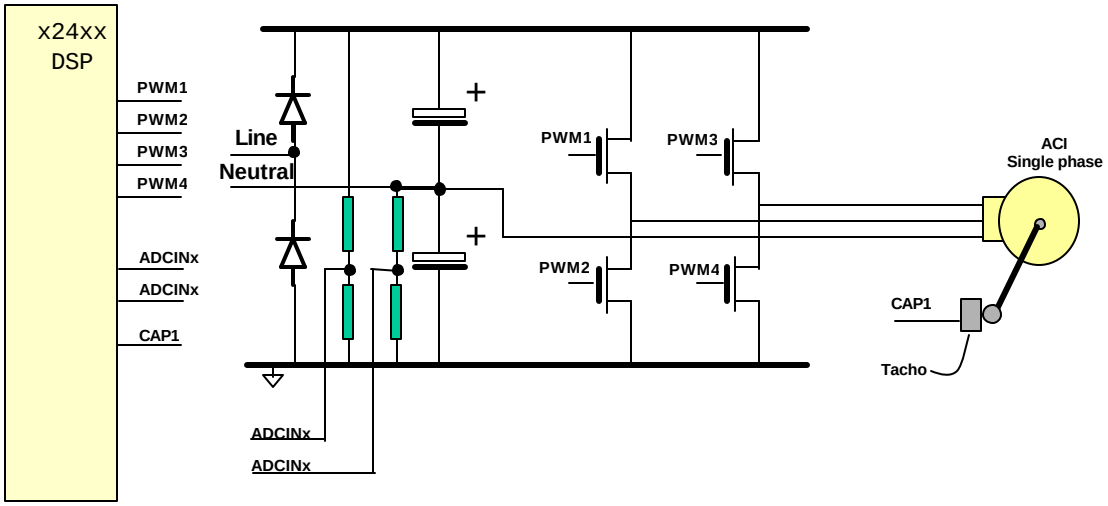
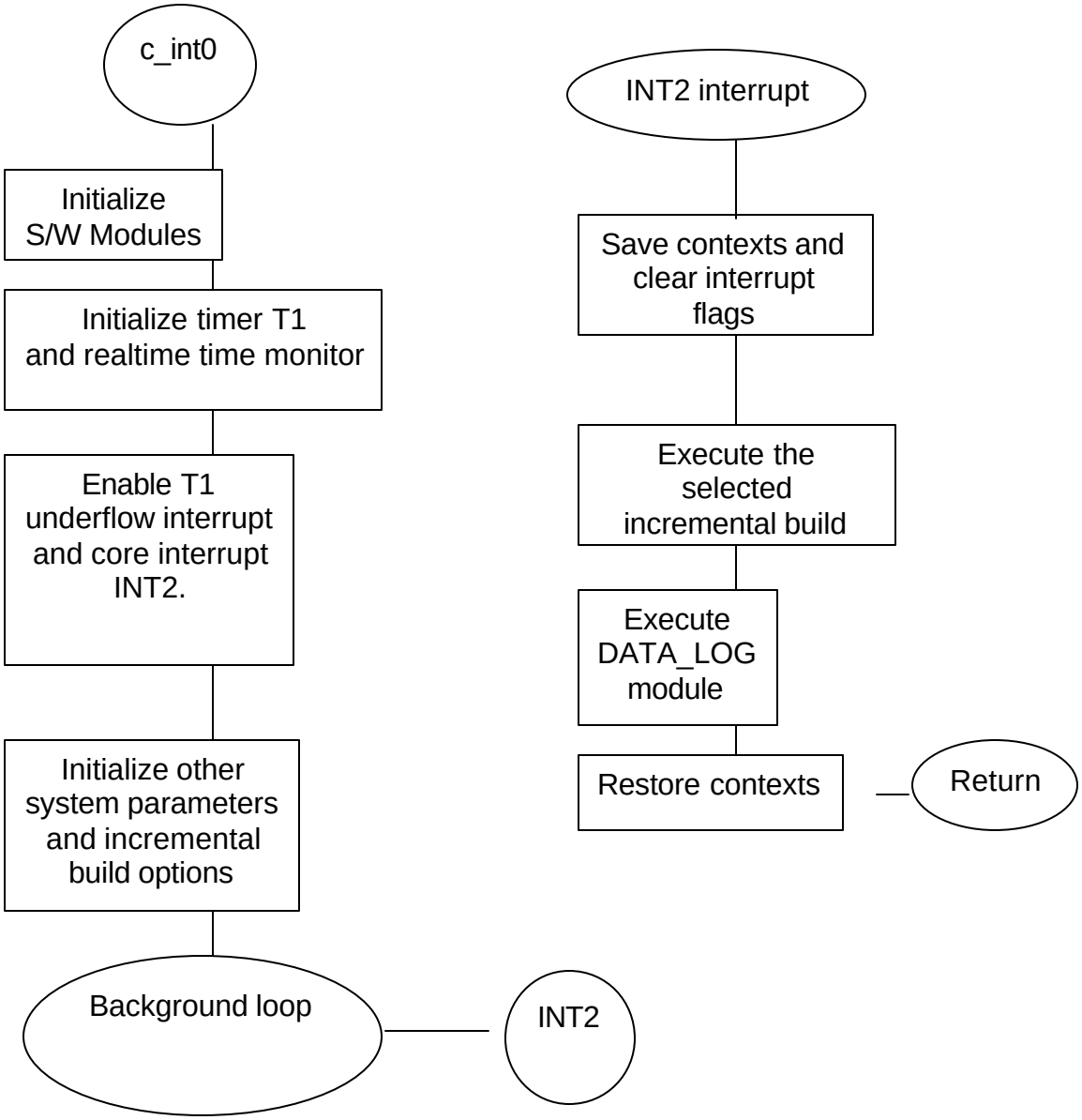


Figure 1: Single-phase ACI motor drive system configuration

Software Flowchart



1.0 Hardware and Software Configuration

1.1 Hardware

The experimental system consists of the following hardware components:

1. Spectrum Digital DMC1500 drive platform;
2. TMS320F243, TMS320F2407 EVM, or TMS320F2407 eZdsp platform (default target);
3. Single Phase AC Induction(ACI) motor;
4. IBM compatible development environment including an IBM compatible PC with Code Composer Studio (CCS) v2.20 installed;

Refer to the User's Guides and or Manuals for configuration of items 2 and 4 and their connection to the system. The following are some of the other system level choices that must be made:

1.1.1 Configuration for DMC1500

Jumper Number	Jumper Setting	Comments
P21-P22	Install jumper for ac input neutral connection	
P15-P16	Install jumper for voltage doubler front-end configuration only	**Donot use** DMC1500 boards in voltage doubler mode with 220VAC input.
JP1, JP2, JP3	Install all jumpers	Use 3-Phase power inverter configuration
JP4	Remove jumper	Use DC bus current sense resistor R6
JP6, JP9	Remove jumpers	Use sense amplifier circuits to condition voltages before applying to ADC inputs
JP20	Install jumper on pins 1-2	Use sense amplifier circuit to condition dc bus voltage
JP19	Install jumper on pins 1-2	Use sense amplifier circuit to condition dc bus cap voltage
JP24	Install jumper on pins 27-28, 33-34 and remove jumper on pins 23-24.	Apply conditioned voltage signals to ADC input
JP17	Install jumper on pins 1-2	Disable PFC circuit
JP27	Install jumper on pins 1-2	Use s/w to enable/disable PWM signals on DMC1500

1.1.2 Configuration for DMC to EVM Interface Board

Jumper Number	Jumper Setting	Comments
JP1	Remove jumper	
JP2	Install jumper	

1.2 Loading and building CC Project for Assembly case

In order to use the CC workspace file in "**C:\TIDCS\DMC\C24\SYS\ACI1_1\ASM**" system directory, all necessary assembly modules in the modular library located in "**C:\TIDCS\DMC\C24\LIB\DMCLIB\ALIB**" and **"C:\TIDCS\DMC\C24\LIB\DRVLIB\ALIB"** directories are linked into the system project file. The assembly files needed for this system are – Box_car, Cap_drv, Data_log, PID_ID, ADC4UDRV, Rmp_cntl, Sincosph, Sintb360, Speed_pr, Pwmodrv, Compwm. All the assembly modules used as described above are linked into the project file as well as real-time monitor related files, which are located in "**C:\TIDCS\DMC\C24\RTMON**" directory. The other files in this system directory are:

- The ACI system assembly file aci1_10.asm and sys_init.asm;
- The linker command file, aci1_10.cmd, that defines memory map and specifies memory allocation;
- One header file x24x_app.h;
- CC project file aci11.pjt;
- CC workspace file (aci1_1.wks) which contains the setup information for the whole project and debugging environment.

It is also assumed that the emulator used is XDS510pp. Once the directory **"C:\TIDCS\DMC\C24\SYS\ACI1_1\ASM"** contains all the necessary files (as mentioned above), the next step is to provide the supply voltage (+5V DC) to the x2407eZdsp and RESET the emulator. Then start the Code Composer and open the project aci11.pjt. Now load the workspace file aci1_1.wks.

Loading the workspace will automatically open up the project file aci11.pjt and show all the files relevant to the project. The following shows the expanded project view as part of CC environment aci1_1.wks is loaded:

Note that the same project can be built from scratch easily if the workspace file, aci1_1.wks, can't be loaded directly because of differences in CC setup and or emulator used. Refer to CC tutorial for information on a building project.

The variables in the Watch Window can be added manually according to CC tutorial.

From the 'PROGRAM LOAD OPTIONS' in CC select 'Load Program After Build' for automatic loading of the program to the target once the program is compiled.

1.3 Loading and building CC Project for "C" case

The process for loading and building the C version of the ACI1-1 system is same as the assembly case with the following exceptions:

1. Modules: The modules COMPWM, FC_PWM_O_DRV, and ADC04U_DRV are excluded at this time. The module FC_PWM_DRV is used instead of FC_PWM_O_DRV. The DATA_LOG module has pointer variables that need proper configuration. These pointers should point to the variables whose values are desired to be saved in the memory. For example, to save the values of a variable 'sine_a1' in the memory, the following code is used to set the pointer:

   dlog1.dlog_iptr1=&vhz.sc.sine_a1;

   This outputs 'sine_a1' to the DATA_LOG Channel 1,

2. Incremental Build Control: The build level is set in the header file, **build.h** in the C infrastructure. This header file is in the **C:\TIDCS\DMC\C24\SYS\ACI1_1\C\INCLUDE** directory. The incremental build level 4 is not implemented for the C system.

   To set the build level, define the symbol BUILDLEVEL to one of the available levels.

```
/*-----------------------------------------------------------------------------------------------------
Following is the list of the Build Level choices.
-----------------------------------------------------------------------------------------------------*/
#define LEVEL1  1
#define LEVEL2  2
#define LEVEL3  3
#define LEVEL3  4                       /* This LEVEL4 is not implemented yet */

/*-----------------------------------------------------------------------------------------------------
This line sets the BUILDLEVEL to one of the available choices.
-----------------------------------------------------------------------------------------------------*/
#define   BUILDLEVEL LEVEL3
```

3. The target processor selection is in the header file, **target.h**. This file is included in the code composer project and can also be found in the directory

   C:\TIDCS\DMC\C24\SYS\ACI1_1\C\INCLUDE

   To change the processor set the TARGET macro to the desired one.

```
/*-----------------------------------------------------------------------------------------------------
Following is the selection list of the target choices.
Note that the F241 is represented by the F243 and the
LF2407 represents the LF2406 and the LF2402.
-----------------------------------------------------------------------------------------------------*/
#define F240   1
#define F243   2
```

```
#define F2407  4
#define UNKNOWN 8

/*----------------------------------------------------------------------------------------------------
This line sets the target to one of the available choices.
----------------------------------------------------------------------------------------------------*/
#define TARGET  F2407
```

4. The object VHZ11_TI is defined in the header file, **vhz11_ti.h** and it is implemented as vhz in **aci11.c**. This has many sub-objects such as PIDREG1, SINCOSPH, RMPCNTL, etc. To modify variables such as pid coefficients, etc. watch:
   i. The entire object vhz, and observe/modify vhz.pid.Kp_reg1.
   ii. An alternative is to watch vhz.pid and modify pid.Kp_reg1.

   Watching smaller objects in the watch window has the advantage of lowering the amount of data that must be up-loaded to the debug tool.

2.0 Incremental System Build – Phase 1

Assuming section 1.0 is completed successfully, this section describes the steps for a "minimum" system check-out which confirms operation of system interrupts, some peripheral & target independent modules and one peripheral dependent module.

In the SYSTEM OPTIONS section of the code, select phase 1 incremental build option by setting the constant *phase1_commissioning* to 1. Save the program, compile it and load it to the target.

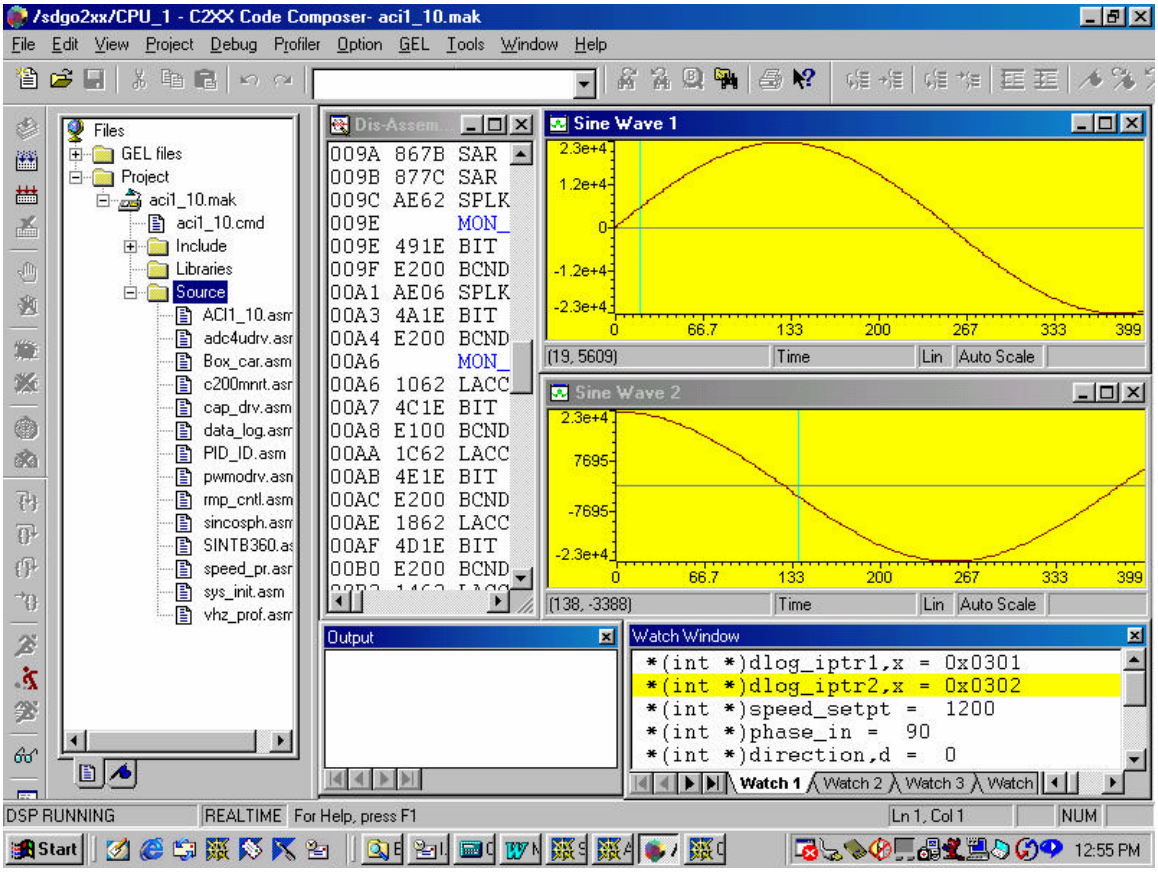
2.1 Phase 1

- In this phase, two sine waves are generated based on the user speed input as shown in Figure-1. Three software modules are utilized in this phase to generate two sine wave forms.
- Set *phase1_commissioning = 1* in *ACI1_10.asm* as shown below-

```
phase1_commissioning    .set    1
phase2_commissioning    .set    0
phase3_commissioning    .set    0
phase4_commissioning    .set    0
```

- The software modules utilized in this phase are – RAMP_CNTL, V_Hz_PROFILE, SINCOSPH
- RAMP_CNTL is utilized to avoid sudden change in speed command.
- V_Hz_PROFILE generates a voltage command "v_out". V_Hz_PROFILE also maintains a fixed ratio between the input frequency and voltage command.
- The module SINCOSPH generates two sine waves. The phase angle between these waves can be controlled by modifying the variable "phase_in".
- The generated wave forms can be watched using the software module "DATA_LOG".

- In order to watch the waveforms in real time initialize *dlog_iptr1 = sine_a1* & *dlog_iptr2 = sine_a2*.
- Figure 2 shows the CC window with phase 1 running.



- Appendix-1 shows the complete software block diagram.

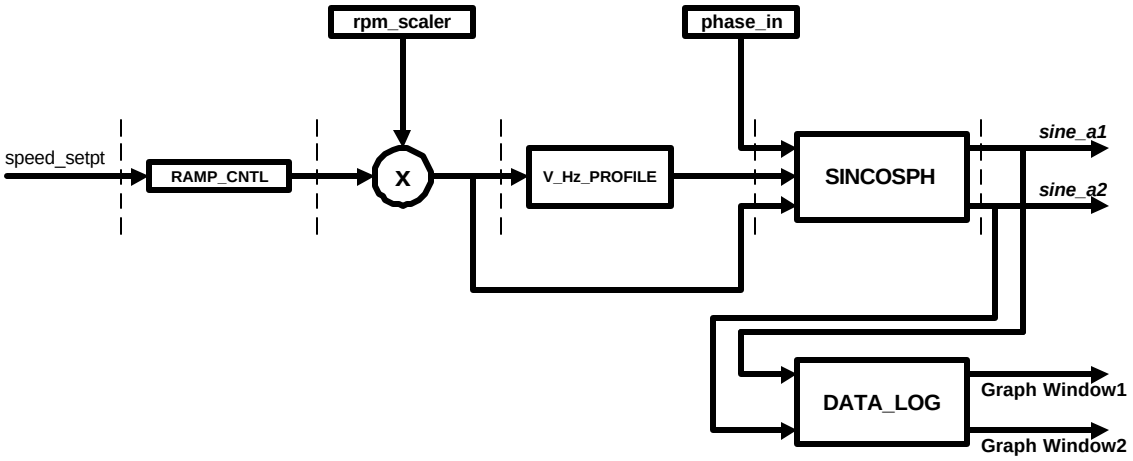


Figure 1 Phase 1 incremental build for ACI1-1

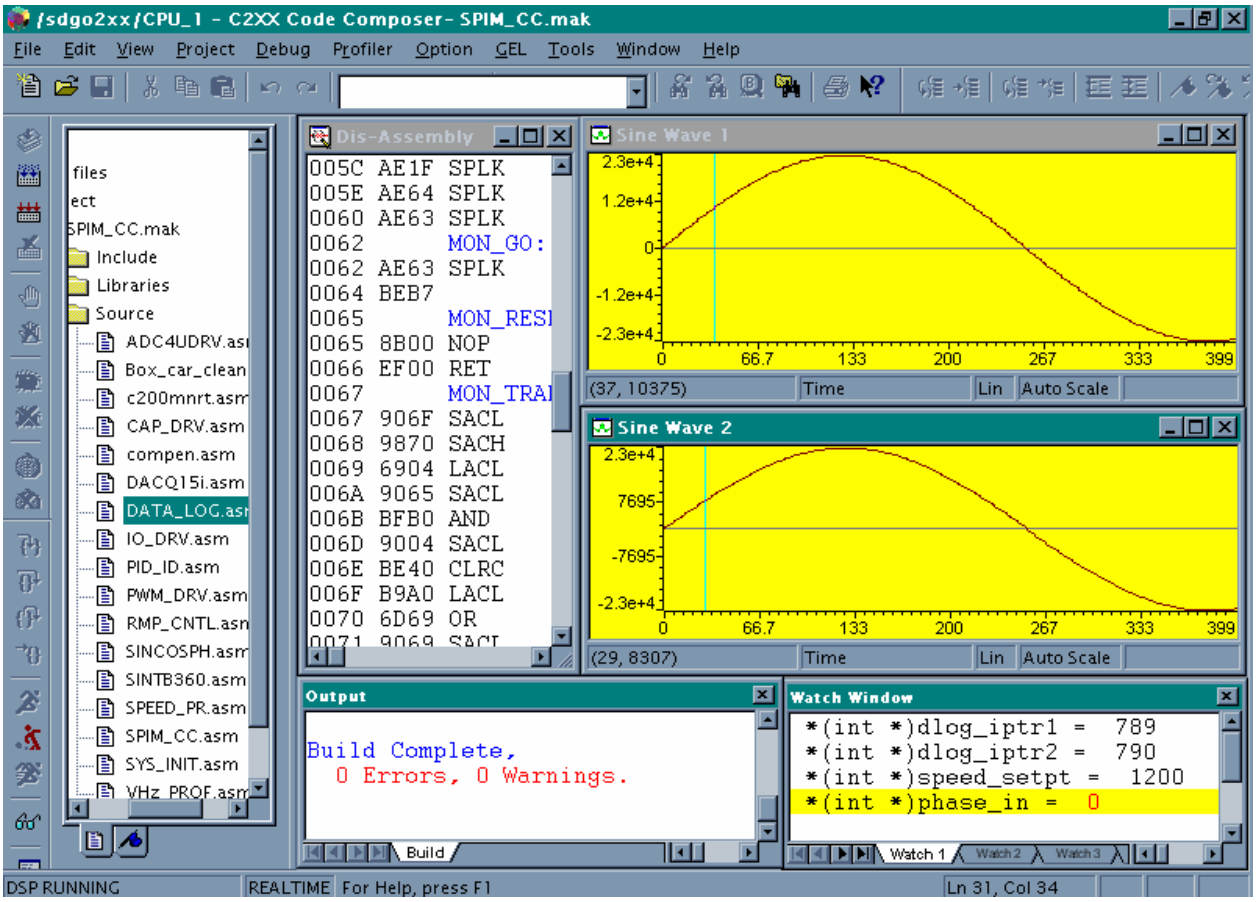


Figure 2 CC window for Phase 1 incremental build of ACI1-1

3.0 Incremental System Build – Phase 2

Successful completion of this phase will enable one to operate a single phase AC induction motor with open loop speed control. Closed loop speed control is not implemented in Phase 2, however, software block is utilized to measure the speed. This will verify the proper operation of speed measuring hardware (sprocket & hall effect sensor).

In Phase-2 of incremental system build, a single phase AC induction motor will be operated using DMC1500 board (www.spectrumdigital.com) and x2407eZdsp board from TI. In Phase-2a few software checks are made and then in Phase-2b an inverter and motor is used to operate a single phase ac induction motor with variable speed control.

3.1 Incremental System Build – Phase 2a

- Set *phase2_commissioning = 1* in *ACI1_10.asm* as shown below-

```
phase1_commissioning    .set    0
phase2_commissioning    .set    1
phase3_commissioning    .set    0
phase4_commissioning    .set    0
```

- The software block diagram of Phase-2a is shown in Figure 3.
- The additional software modules added in this Phase-2a are FC_PWM_DRV and SPEED_PRD.
- FC_PWM_DRV configures PWM module of the DSP controller. It also provides the required dead band through avoid shoot thorough faults. The third input, Mfunc_c3, FC_PWM_DRV is set to zero. This is necessary because a single-phase AC induction drive utilizes only four PWM channels.
- SPEED_PRD calculates motor speed from the capture events. The capture module of the device is configured at the very beginning by calling CAP_EVENT_DRV_INIT module.
- The variable “direction” can be set to “1” or “0” to change the direction of rotation of

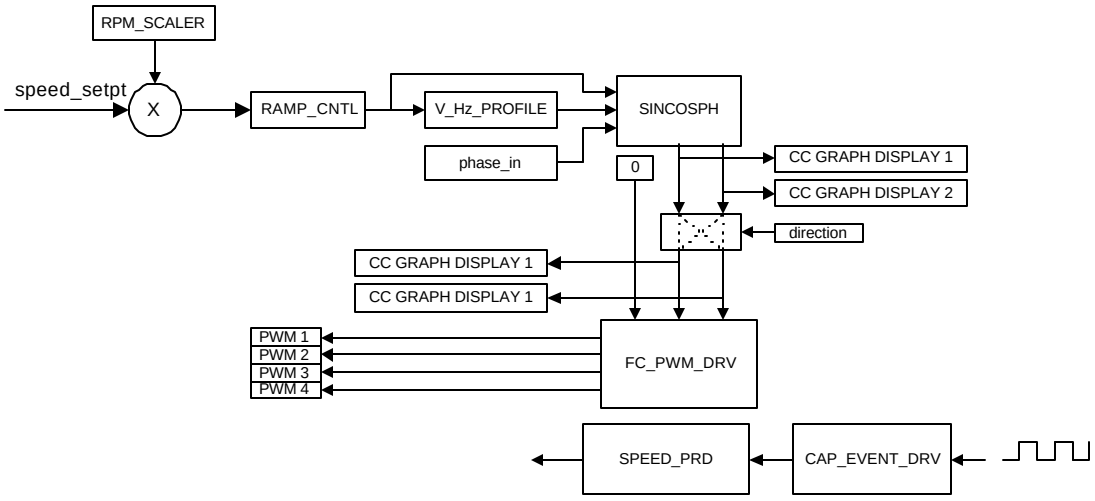


Figure 3 Phase 2a software block diagram for ACI1-1 system.

the motor. When direction is equal to “1’, sine wave-1 will lead sine wave-2. If direction is equal to “0” than the opposite will happen. The available graph display in CC can be utilized to observe this. Put dlog_iptr1 = Mfunc_c1 & dlog_iptr2 = Mfunc_c2. Figure 4 & Figure 5 shows how two sine waves lead or lag each other depending on the value of “direction”.

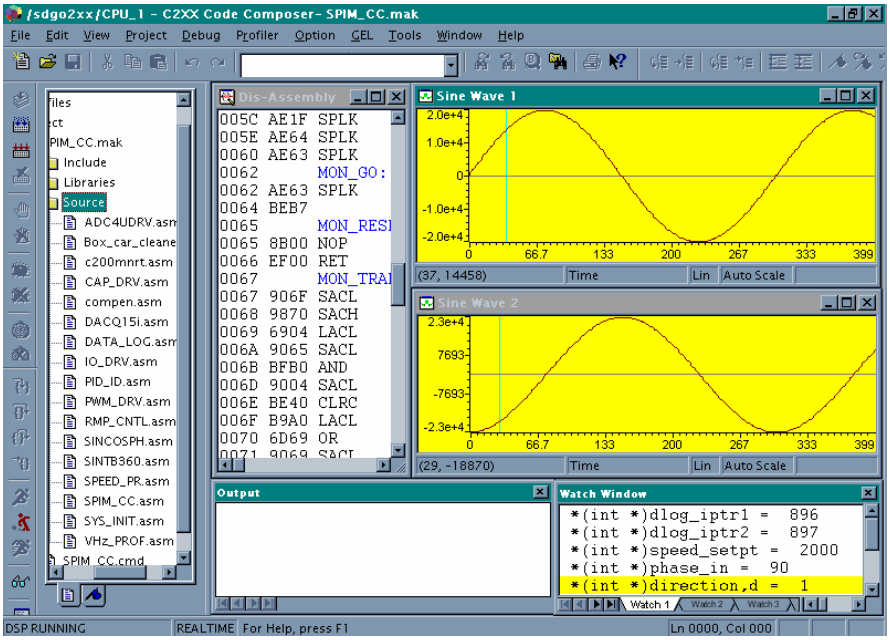


Figure 4 CC Display window with dlog_iptr1=Mfunc_c1, dlog_iptr2=Mfunc_c2, phase=90, direction=1 (wave 1 leads wave 2)

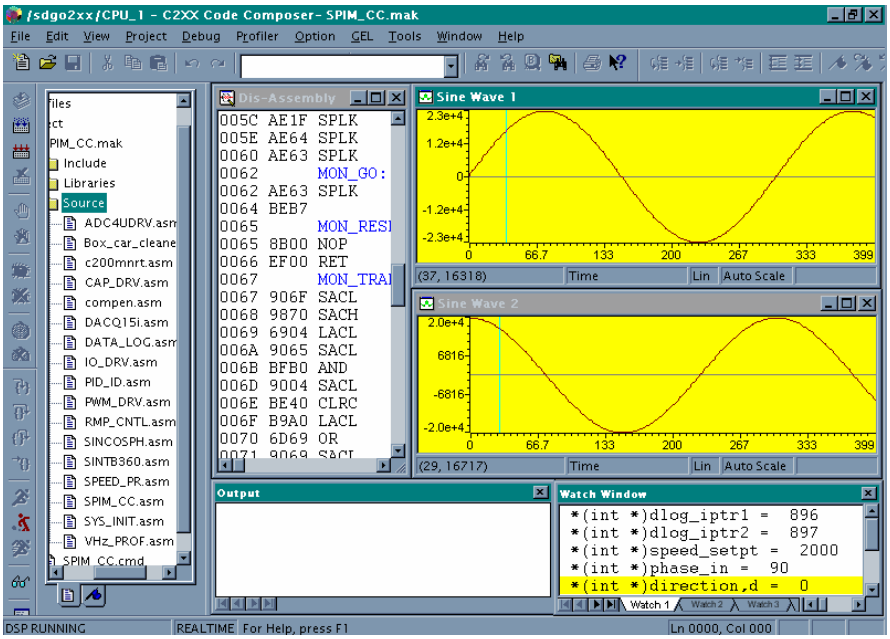


Figure 5 CC Display window with dlog_iptr1=Mfunc_c1, dlog_iptr2=Mfunc_c2, phase=90, direction=0 (wave 1 lags wave 2). Motor rotational direction in Figure 5 will be opposite of rotational direction in Figure 4.

3.2 Incremental System Build – Phase 2b

In this phase a single phase AC induction motor is connected with an voltage source inverter and the above mentioned software is utilized to implement a variable speed drive. A x2407eZdsp is interfaced with DMC1500 board to setup the controller and power stage of the drive. Figure 6 shows the connection between induction motor and DMC1500. The following steps are needed for proper setup of operation –

- Turn the variac all the CCW to zero voltage position.
- Connect the variac to 120V wall socket.
- Connect a current probe with any motor phase.
- Power ON x2407eZdsp and DMC1500 boards.
- Start CC and load the software.
- Once the software is running, apply DC bus voltage by rotating the variac. At this time the current probe should show sinusoidal phase current and after sufficient DC bus voltage, the motor will start rotating.
- The direction of rotation will depend on two variables – First the motor phase connections and second the value of the variable "direction".
- Changing the value of "direction" as mentioned in the previous section (Phase 2a) can change the rotational direction.
- Modifying the variable "speed_setpt" can vary the motor speed. The maximum speed is fixed at 2500RPM. However, user can change that by modifying V_Hz_PROFILE software module.
- Figure 7 shows motor phase currents.
- Appendix-2 shows the complete software block diagram of phase-2.

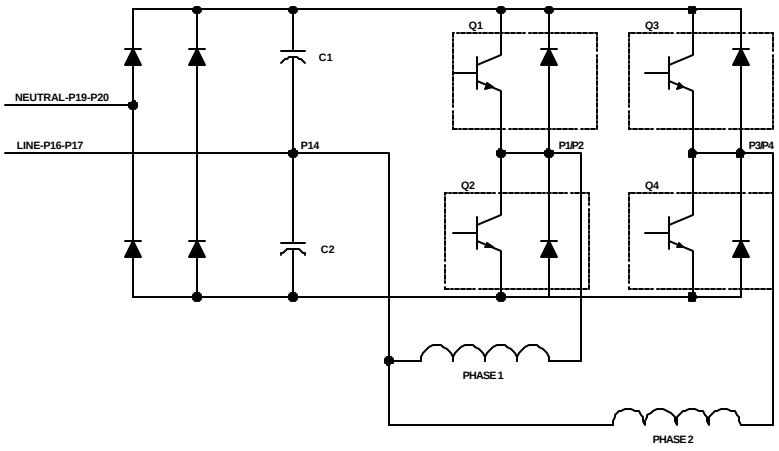


Figure 6 Hardware connection between a single phase AC induction motor and DMC1500 board.

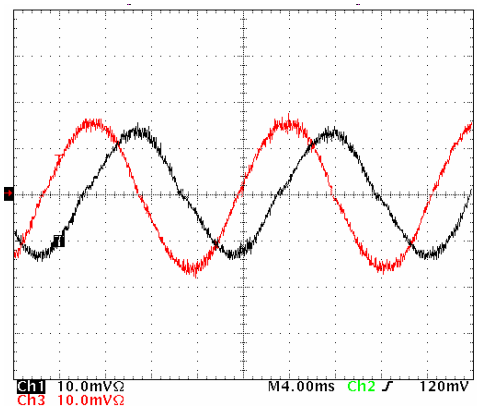


Figure 7 Motor phase currents, 2A/div (phase 1 current is leading phase 2 current by 90 degrees) .

4.1 Incremental System Build – Phase 3

In this phase, the same single phase AC induction motor is operated in variable speed mode. However, in this phase, the speed of the motor is measured using a hall effect sensor,and hardware capture module of the DSP controller in order to close the speed loop. The following asm files are utilized to measure the motor speed–

a. CAP_DRV.asm : configures the capture module
b. SPEED_PR.asm : calculates speed from captured values
c. Box_car : calculates the average of the calculated speed.
: This eliminates jitters in speed sensing hardware.

The following steps will ensure proper operation of Phase-3 –

- Set *phase3_commissioning = 1* in *ACI1_10.asm* as shown below-

```
phase1_commissioning    .set    0
phase2_commissioning    .set    0
phase3_commissioning    .set    1
phase4_commissioning    .set    0
```

- The Hall effect speed sensor output is connected to P30 of DMC1500. P30 has ground, power and capture input. The available ground and power from P30 can be utilized as sensor ground and power. The voltage output at P30 can be set at various level to match the sensor requirement.
- The remaining hardware setup is similar to Phase-2b.
- After starting the software, the motor will start rotating as the DC bus voltage is applied by rotating the variac from zero to higher voltages.
- The measured speed can be watched by adding the variable “speed_rpm” in the watch window.
- The speed loop is closed by putting *closed_loop_flag=1*
- Appendix-3 shows the complete software block diagram of Phase-3.

5.1 Incremental System Build – Phase 4

In this stage, the same single phase AC induction motor is operated with DC bus voltage ripple compensation. The configuration of single phase AC induction motor drive inverter creates significant DC bus voltage ripple. This DC bus voltage ripple distorts the inverter output voltage and consequently lowers the drive performance. The drive is made immune to ripple voltage my adjusting PWM outputs according to the DC bus voltage variation. In a single-phase inverter, voltage across both capacitors is measured to implement independent phase voltage compensation. The following asm files are utilized to implement ripple compensation –

Compwm.asm – Adjusts PWM waveforms according to DC bus voltage.
ADC04DRV.asm - Measures DC bus voltages across two capacitors.

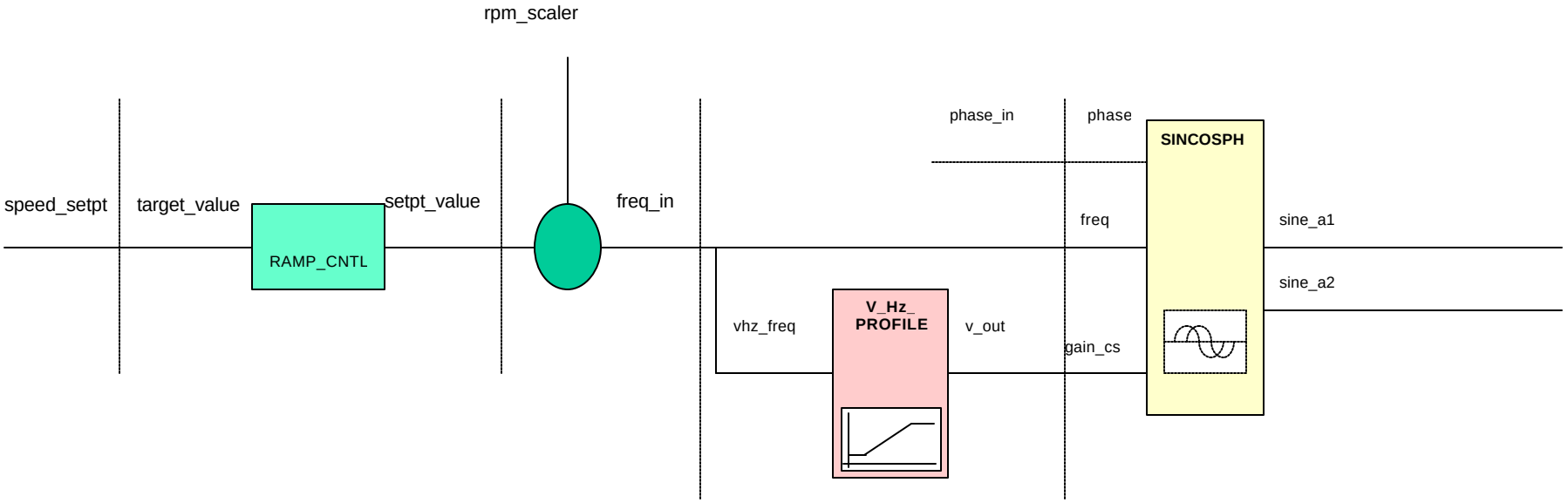
The following steps are necessary for Phase-4.

- Set *phase4_commissioning = 1* in *ACI1_10.asm* as shown below-

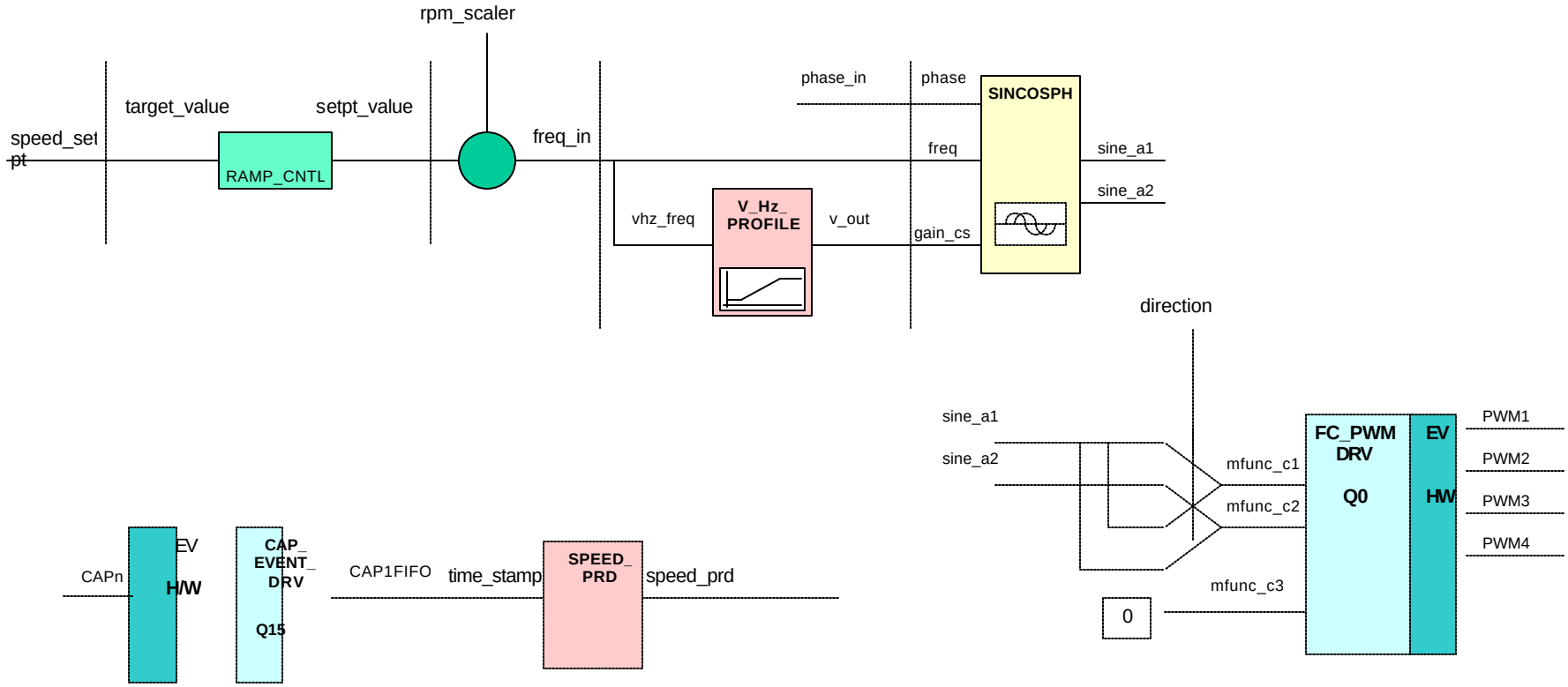
```
phase1_commissioning    .set    0
phase2_commissioning    .set    0
phase3_commissioning    .set    0
phase4_commissioning    .set    1
```

- Remove file PWMODRV.asm from the project and ADD COMPWM.asm to the project.
- Initialize the reference DC bus voltages ADC_ref1 and ADC_ref2. ADC_ref1 is upper capacitor voltage and ADC_ref2 is lower capacitor voltage. These numbers should be provided in Q15 format (see the COMPWM sofware module documentation).
- In this implementation 120VAC is supplied to the inverter.
- DMC1000 board is utilized to implement this phase. Two ADC channels are utilized to measure DC bus voltages. ADC07 measures the complete DC bus voltage (software variable is *total_bus*). ADC05 measures the lower capacitor voltage (software variable is *half_bus*). From these two measurements, individual capacitor voltages are calculated. For example, ADC1 is the voltage across the top capacitor and is equal to the difference between *total_bus* and *half_bus*.
- The software is initialized with the compensation turned off. In order to turn on the compensation the variable *DC_RIPPLE* should be changed from 0 to 1.
- Appendix-4 shows the complete software block diagram of Phase-3.

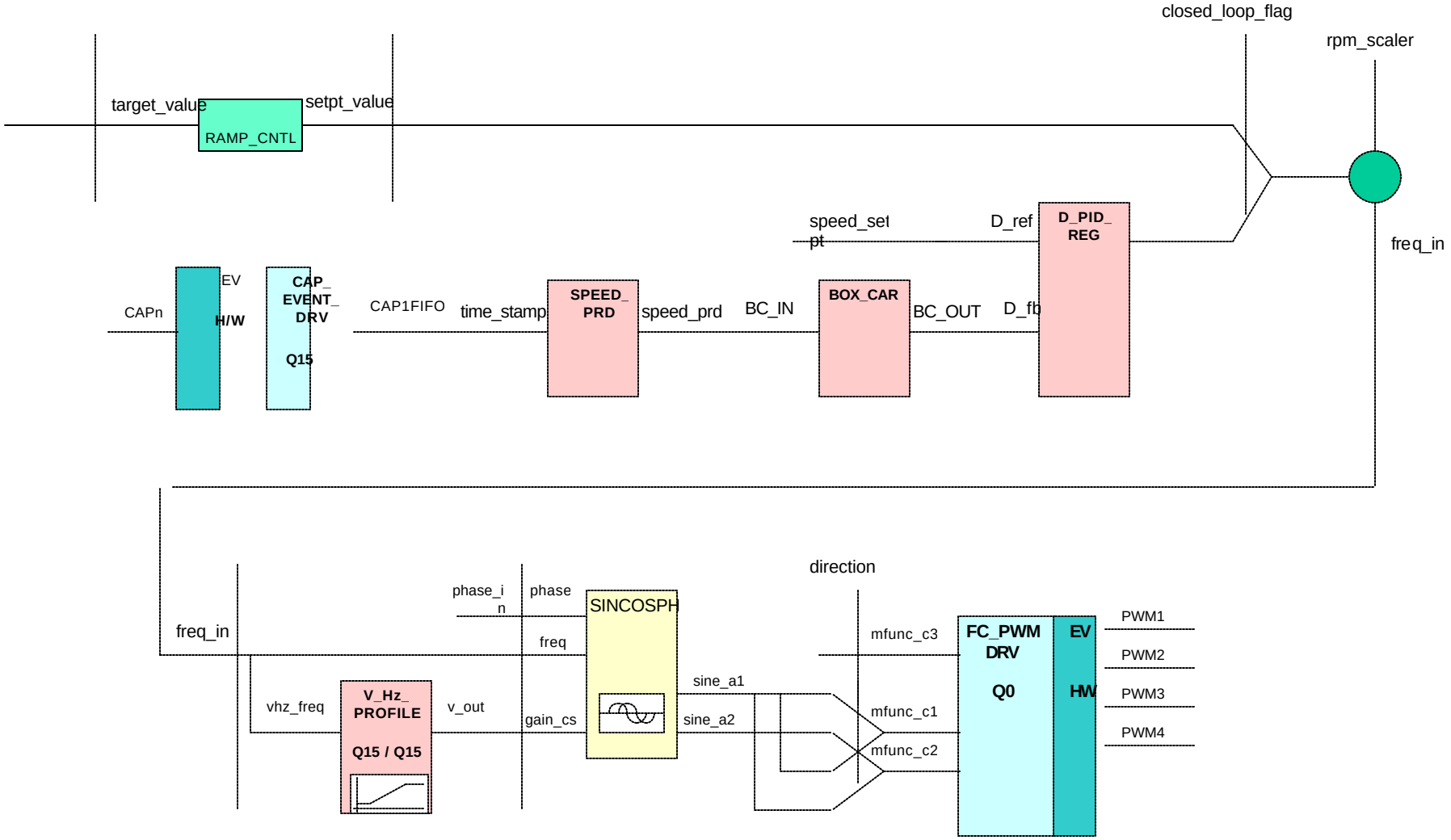
APPENDIX – 1 Phase 1



APPENDIX -2 Phase – 2



APPENDIX – III Phase –3



APPENDIX – IV Phase – 4