ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

BÁO CÁO TỔNG KẾT
ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP ĐẠI HỌC

NGHIÊN CỨU THIẾT KẾ VÀ CHẾ TẠO HỆ THỐNG XÁC ĐỊNH CHÍNH XÁC VỊ TRÍ CỦA ĐỘNG CƠ TUYẾN TÍNH TRONG CÁC HỆ THỐNG CHUYỂN ĐỘNG THẲNG

Mã số: ĐH2016 -TN02 – 04

Chủ nhiệm đề tài: TS. Cao Xuân Tuyển

Thái Nguyên, 02/2019

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

BÁO CÁO TỔNG KẾT
ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP ĐẠI HỌC

NGHIÊN CỨU THIẾT KẾ VÀ CHẾ TẠO HỆ THỐNG XÁC ĐỊNH CHÍNH XÁC VỊ TRÍ CỦA ĐỘNG CƠ TUYẾN TÍNH TRONG CÁC HỆ THỐNG CHUYỂN ĐỘNG THẲNG

Mã số: ĐH2016 -TN02 – 04

Xác nhận của tổ chức chủ trì	**Chủ nhiệm đề tài**
KT. HIỆU TRƯỞNG	
PHÓ HIỆU TRƯỞNG	
PGS.TS. Vũ Ngọc Pi	**TS. Cao Xuân Tuyển**

Thái Nguyên, 02/2019

NHỮNG NGƯỜI THAM GIA THỰC HIỆN ĐỀ TÀI

1.TS. Vũ Quốc Đông - Khoa Quốc tế – Trường ĐHKT Công nghiệp.

2. TS. Vũ Ngọc Kiên – Khoa Điện – Trường ĐHKT Công nghiệp.

3. ThS. Nguyễn Tiến Dũng – Khoa Điện – Trường ĐHKT Công nghiệp.

4. ThS. Vũ Xuân Tùng – Khoa Điện – Trường ĐHKT Công nghiệp.

ĐƠN VỊ PHỐI HỢP CHÍNH

1.Viện Nghiên cứu Phát triển Công nghệ cao

2.Trung tâm Thí nghiệm – Trường ĐHKT Công nghiệp.

MỤC LỤC

DANH MỤC HÌNH VẼ - BẢNG BIỂU

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

STT	KÝ HIỆU	ĐƠN VỊ	Ý NGHĨA
1	L_{rd}, L_{rq}	H	Điện cảm phần động dọc trục và ngang trục
2	i_{rd} , i_{rq}	A	Thành phần dòng điện phần động trên trục d và q của hệ toạ độ (d,q) chuyển động với vận tốc của phần động (tương đương với hệ toạ độ quay với tốc độ rotor)
3	V	m/s	Vận tốc cơ phần động
4	S	m	Quãng đường dịch chuyển của phần động động cơ
5	τ	m	Bước cực của động cơ
6	ψ_p	Wb	Từ thông của một cực từ
7	u_{rd}, u_{rq}	V	Các thành phần điện áp phần động trên trục d,q
8	m	kg	Khối lượng phần động
9	F, F_c	N	Lực điện từ và lực cản động cơ
10	R_r	Ω	Điện trở cuộn dây pha phần động

STT	CHỮ VIẾT TẮT	Ý NGHĨA
1	(d,jq)	Hệ tọa độ tựa theo cực từ chuyển động tịnh tiến theo phần động [2]
2	REC	Bộ chỉnh lưu
3	FLC	Bộ điều khiển mờ
4	NLNA	nghịch lưu nguồn áp
5	SVPWM	Điều chế vec tơ không gian
6	BĐK	Bộ điều khiển
7	FLC	Bộ điều khiển logic mờ
8	DSP	Bộ xử lý tín hiệu số
9	SLVT(ΔS)	Sai lệch vị trí
10	VT(Δe)	Đạo hàm sai lệch vị trí
11	CPU	Khối xử lý trung tâm
12	JTAG	Joint Tool Action Group
13	PID	Tỉ lệ, tích phân, đạo hàm
14	IGBT	Transistor có cực điều khiển cách ly
15	CCS	Code Composer Studio
16	PC	Máy tính cá nhân
17	ĐB-KTVC	Động cơ đồng bộ 3 pha kích thích vĩnh cửu
18	ĐCTT ĐB-KTVC	động cơ tuyến tính 3 pha kiểu đồng bộ kích thích vĩnh cửu
19	(a,b)	Hệ toạ độ cố định với phần động the tài liệu [2]
20	CNC	Computer Numerical Control
21	ĐCTT	Động cơ tuyến tính

22	DC	Dòng một chiều
23	μC, DSP	Vi điều khiển/vi xử lý tín hiệu số
24	IPM	Intelligent Power Modules
25	LV ASIC	Low voltage Application Specific Integrated Circuit
26	HV ASIC	High voltage Application Specific Integrated Circuit
27	ASIPM	Modul công suất thông minh chuyên dụng (Application Specific IPM)
28	HVIC	High Voltage Integrated Circuit
29	PWM	Điều chế độ rộng xung

ĐẠI HỌC THÁI NGUYÊN
Trường Đại học Kỹ thuật Công nghiệp

THÔNG TIN KẾT QUẢ NGHIÊN CỨU

1.Thông tin chung

- Tên đề tài: Nghiên cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng

- Mã số: ĐH2016 -TN02 – 04

- Chủ nhiệm đề tài: TS. Cao Xuân Tuyển.

- Tổ chức chủ trì: Trường Đại học Kỹ thuật Công nghiệp - Đại học Thái Nguyên.

- Thời gian thực hiện: từ 01/8/2016 đến 28/2/2019.

2. Mục tiêu

Nghiên cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng

3. Tính mới, tính sáng tạo

Tính mới, tính sáng tạo của đề tài được thể hiện ở các khía cạnh sau:

- Thứ nhất, cho đến nay, ở Việt nam chưa có nhà chế tạo nào chế tạo ra thiết bị điều khiển vị trí cho động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu nói riêng và động cơ tuyến tính nói chung. Đề tài này là đề tài đầu tiên ở Việt Nam thực hiện chế tạo loại thiết bị này, tuy ở mức công suất nhỏ, nhưng đây là cơ sở để tiến tới chế tạo thiết bị ở các mức công suất lớn hơn.

- Thứ hai, đề tài đã áp dụng phương pháp điều khiển hiện đại là phương pháp điều khiển vector với bộ điều khiển dòng điện phi tuyến và bộ điều khiển mờ PID cho mạch vòng vị trí để nâng cao tính chính xác trong điều khiển vị trí của hệ thống.

4. Kết quả nghiên cứu

- Tài liệu báo cáo về tính toán thiết kế, xây dựng mạch động lực của hệ thống.

- Tài liệu báo cáo về tính toán thiết kế, xây dựng mạch điều khiển của hệ thống.

- Tài liệu báo cáo về viết thuật toán, viết phần mềm điều khiển hệ thống.

- Sơ đồ mô phỏng trên matlab/Simulink dùng để tạo code chương trình nạp vào DSP TMS320F2812.

- Chế tạo hoàn chỉnh hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng với các kết quả kiểm nghiệm hệ thống đạt yêu cầu như thuyết minh.

5. Sản phẩm

5.1.Sản phẩm khoa học:

1. Cao Xuân Tuyển, Vũ Quốc Đông, Vũ Ngọc Kiên, Nguyễn Tiến Dũng, Vũ Xuân Tùng (2019), *Nghiên cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng*, tài liệu tham khảo Khoa Điện, trường Đại học Kỹ thuật Công nghiệp – Đại học Thái Nguyên.

2. Cao Xuân Tuyển, Nguyễn Thị Hương (2018), "Áp dụng phương pháp điều khiển Backstepping và bộ điều khiển PID mờ để điều khiển vị trí động cơ chạy thẳng (tuyến tính) xoay chiều ba pha kích thích nam châm vĩnh cửu", *Tạp chí Khoa học & Công nghệ- Đại học Thái Nguyên*, 178(02), tr. 55-60.

5.2.Sản phẩm đào tạo:

1. Nguyễn Văn Quyết (2016), *Nghiên cứu điều khiển tốc độ, vị trí và đảo chiều động cơ tuyến tính theo phương pháp điều chế vector không gian ứng dụng trong hệ chuyển động thẳng,* Luận văn thạc sĩ Khoa học Kỹ thuật chuyên ngành Kỹ thuật điều khiển & tự động hóa, trường Đại học Kỹ thuật Công nghiệp – Đại họcThái Nguyên.

2. Nguyễn Ngọc Quyết (2016), *Nghiên cứu điều khiển tốc độ, vị trí và đảo chiều động cơ tuyến tính theo phương pháp điều chế độ rộng xung ứng dụng trong hệ thống chuyển động thẳng*, Luận văn thạc sĩ Khoa học Kỹ thuật chuyên ngành Kỹ thuật điều khiển & tự động hóa, trường Đại học Kỹ thuật Công nghiệp – Đại học Thái Nguyên.

5.3.Sản phẩm ứng dụng:

Stt	Tên sản phẩm	Số lượng	Yêu cầu khoa học	Địa chỉ ứng dụng
1	Thiết bị và tài liệu báo cáo về hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng	01	Điện áp nguồn 220V, một pha; Công suất: 0,25kW; Độ chính xác vị trí: 10-4 m	Hệ thống mà modul cơ sở dùng để chế tạo các robot, máy gia công CNC, các tàu điện cao tốc, … trong các hệ thống gia công chính xác chuyển động thẳng.

6. Phương thức chuyển giao, địa chỉ ứng dụng, tác động và lợi ích mang lại của kết quả nghiên cứu

-Sản phẩm thiết bị hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng là sản phẩm hữu ích có thể được áp dụng

trong các lĩnh vực công nghiệp hiện đại yêu cầu điều khiển vị trí với độ chính xác cao, như các máy CNC, robot,…

-Tài liệu tham khảo dưới dạng báo cáo tổng kết là nguồn tham khảo trên cơ sở đó để tiến tới chế tạo thiết bị ở các mức công suất lớn hơn.

- Kết quả nghiên cứu, phương pháp nghiên cứu, hướng nghiên cứu của đề tài làm cơ sở cho việc phát triển nghiên cứu, đào tạo trình độ đại học và sau đại học tại trường Đại học Kỹ thuật Công nghiệp – Đại học Thái Nguyên hiện tại và trong tương lai.

Ngày 11 tháng 02 năm 2019

Tổ chức chủ trì	**Chủ nhiệm đề tài**
KT. HIỆU TRƯỞNG	
PHÓ HIỆU TRƯỞNG	
	TS. Cao Xuân Tuyển
PGS.TS. Vũ Ngọc Pi	

INFORMATION ON RESEARCH RESULTS

1. General information

- Project title: Research, design and manufacture a accurately position determining system using the linear motor in linear motion systems.
- Code number: ĐH2016 -TN02 – 04
- Project Director: Dr. Cao Xuan Tuyen
- Organization: TNU - Thai Nguyen University of Technology
- Duration: From August 2015 to February 2019

2. Objective: Research, design and manufacture a accurately position determining system using the linear motor in linear motion systems

3. Creativeness and innovativeness: The creativeness and innovativeness of the topic are expressed in the following aspects:

- Firstly, until now, no manufacturer in Vietnam has produced a position control device for permanent magnetic three-phase synchronous linear motor in particular and linear motors in general. This project is the first project in Vietnam to implement this type of device, although at a small capacity, but this is the basis for making equipment at larger capacity levels.

- Secondly, the project has applied modern control method which is vector control method with nonlinear current controller and PID fuzzy controller for position loop circuit to improve the control of position control of system.

4. Research results: Reporting material on design, building the power circuit of the system, reporting material on design, building the control circuit of the system, reports on writing algorithms, writing system control software, simulation model on matlab / Simulink used to create program code loaded into TMS320F2812 DSP, design and manufacture a accurately position determining system using the linear motor in linear motion systems with system test results meeting the requirements on the project.

5. Products

5.1. Scientific products:

1.Cao Xuan Tuyen, Vu Quoc Đong, Vu Ngoc Kien, Nguyen Tien Dung, Vu Xuan Tung (2019), *Research, design and manufacture a accurately position determining system using the linear motor in linear motion systems.*

2. Cao Xuan Tuyen, Nguyen Thi Huong (2018),"Applying the Backstepping control method and fuzzy PID controller to control the position of permanent magnet three phase AC linear motors", *Journal of science and technology – Thai Nguyen University*, 178(02), pp. 55-60.

5.3. Training products:

1. Nguyen Van Quyet (2016), *Study the control of speed, position and reverse of linear motors using the space vetor modulation method in linear motion systems*, Science master thesis in Control & Automation, TNU - Thai Nguyen University of Technology.
2. Nguyen Ngoc Quyet (2016), *Study the control of speed, position and reverse of linear motors using the pulse width modulation method in linear motion systems*, Science master thesis in Control & Automation, TNU - Thai Nguyen University of Technology.

5.2. Application product:

Numerical order	Name of product	quantity	Science requirement	Application address
1	A accurately position determining system using the linear motor in linear motion systems	01	Source voltage : Single phase AC 220 V, 0.25 kW, Position accuracy: 10-4 m	Basic module in robots. CNC, high-speed trams, ... in linear motion precision machining systems .

6. Transfer alternatives, application institutions, impacts and benefits of research results:

- The device of accurately position determining system using the linear motor in linear motion systems is a useful product that can be applied in modern industrial fields that require position control with high precision, such as CNC machines, robots, ...

- References in the form of summarized reports are reference sources on that basis to proceed to manufacture equipment at larger capacity levels.

- Research results, research methods, research directions of the project serve as a basis for the development of research and training at the undergraduate and postgraduate levels in TNU - Thai Nguyen University of Technology now and in the future.

MỞ ĐẦU

Trong thực tế sản xuất hiện nay, chuyển động thẳng là dạng chuyển động phổ biến, xuất hiện nhiều, đặc biệt trong lĩnh vực cơ khí. Xuất phát từ công nghiệp chế tạo máy với những dịch chuyển của bàn gá, mũi khoan,… trong các máy gia công cho đến sự ra đời của máy CNC đã dẫn đến nhu cầu đòi hỏi tạo ra chuyển động thẳng có chất lượng cao. Ngoài ra những chuyển động thẳng này còn tồn tại nhiều trong các thiết bị khác như Robot công nghiệp hay máy móc phục vụ ngành công nghiệp bán dẫn,… và nó còn xuất hiện ở cả những lĩnh vực tưởng chừng xa lạ như ngành giao thông vận tải với tàu đệm từ trường ở các nước phát triển (Đức, Nhật,..).

Cho đến nay việc tạo ra các chuyển động thẳng hầu hết được thực hiện một cách gián tiếp thông qua các động cơ quay tròn với những ưu thế như bền vững, không nhạy với nhiễu, độ tin cậy cao,… Tuy nhiên đối với những hệ thống này do phải bổ sung các cơ cấu chuyển đổi trung gian như hộp số, trục vít,… nên dẫn đến sự phức tạp về kết cấu cơ khí, tiềm ẩn bên trong nó những dao động riêng, tổn hao năng lượng cũng như ảnh hưởng đến chất lượng chuyển động của hệ thống. Việc sử dụng loại động cơ có khả năng tạo chuyển động thẳng trực tiếp (động cơ tuyến tính) cho phép loại bỏ những nhược điểm nói trên và những nghiên cứu về loại động cơ này hy vọng sẽ phần nào khắc phục được những đặc điểm đó.

Xuất phát từ thực tế nêu trên, nhóm tác giả đã đề suất và thực hiện đề tài "**Nghiên *cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng*".**

1.Tính khoa học và cấp thiết của đề tài

Động cơ tuyến tính là một giải pháp công nghệ mới hiện được ứng dụng nhiều trong sản xuất công nghiệp, ứng dụng trong gia công cắt gọt, tạo hình bề mặt phức tạp như máy gia công CNC và cả giao thông vận tải (dùng ở loại tàu đệm từ cao tốc). So với truyền động cơ khí kiểu truyền thống, giải pháp công nghệ động cơ tuyến tính có nhiều ưu điểm hơn, chẳng hạn như có thể đạt được mức dịch chuyển với tốc độ cao; giảm ma sát trong truyền động; có tuổi thọ rất dài… Hiện nay, tại các nước phát triển, giải pháp công nghệ động cơ tuyến tính được sử dụng rất phổ biến. Mặc dù có nhiều ưu điểm về tính năng, hiệu quả khi sử dụng nhưng do giá

thành còn khá cao nên giải pháp công nghệ động cơ tuyến tính vẫn chưa có mặt ở Việt Nam.

Một lĩnh vực ứng dụng quan trọng khác của hệ truyền động động cơ tuyến tính là sự tham gia của nó trong các máy Robot gia công các bề mặt phức tạp của các miếng vá sọ não, các bề mặt khớp gối, khớp vai… trong y học.

Các hướng nghiên cứu nêu trên đề thuộc các định hướng nghiên cứu trọng điểm quốc gia của nhà nước.

Với giải pháp công nghệ động cơ tuyến tính ứng dụng trong các máy gia công cắt gọt, tạo hình bề mặt phức tạp như máy gia công CNC, robot, nhằm nâng cao độ chính xác trong gia công, thì yêu cầu về điều khiển chính xác vị trí của động cơ tuyến tính là vấn đề mấu chốt. Vấn đề này đã được các hãng sản xuất nước ngoài thực hiện tốt, nhưng ở Việt Nam thì chưa có hang sản xuất nào snar xuất động cơ tuyến tính cũng như thiết bị điều khiển nó.

Vì vậy, đề tài được đặt ra với mục đích tạo ra sản phẩm trong nước, là thiết bị "cơ sở" sử dụng để chế tạo các máy gia công cắt gọt, tạo hình bề mặt phức tạp như máy gia công CNC, robot, nhằm nâng cao độ chính xác trong gia công cơ khí công nghệ cao.

2. Mục tiêu của đề tài

Nghiên cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng

3. Đối tượng, phạm vi nghiên cứu

- Đối tượng nghiên cứu

Động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu và hệ thống xác định chính xác vị trí của động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu trong các hệ thống chuyển động thẳng.

-Phạm vi nghiên cứu

Mạch động lực, mạch điều khiển và thuật toán điều khiển cho hệ thống xác định chính xác vị trí động cơ tuyến tính trong các hệ thống chuyển động thẳng

4. Cách tiếp cận và phương pháp nghiên cứu

- Cách tiếp cận

Tính toán thiết kế xây dựng mô hình mô phỏng trên máy tính và mô hình thí nghiệm; thiết kế chế tạo hệ thống thực và thử nghiệm trên hệ thống thực.

- Phương pháp nghiên cứu

Sử dụng phương pháp phân tích, tổng hợp, so sánh và đánh giá :Nghiên cứu tài liệu về thiết kế, chế tạo các động cơ tuyến tính, về các phương pháp điều khiển động cơ tuyến tính; tìm hiểu, phân tích, đánh giá một số hệ thống điều khiển vị trí động cơ tuyến tính đã có trên thế giới để thiết kế hệ thống xác định vị trí phù hợp với điều kiện trong nước về sản xuất và ứng dụng.

Mô hình hóa và mô phỏng để kiểm tra khả năng làm việc và chỉ tiêu kỹ thuật của thiết bị (hệ thống) đã thiết kế;

Chế tạo, thử nghiệm thiết bị thực đã chế tạo.

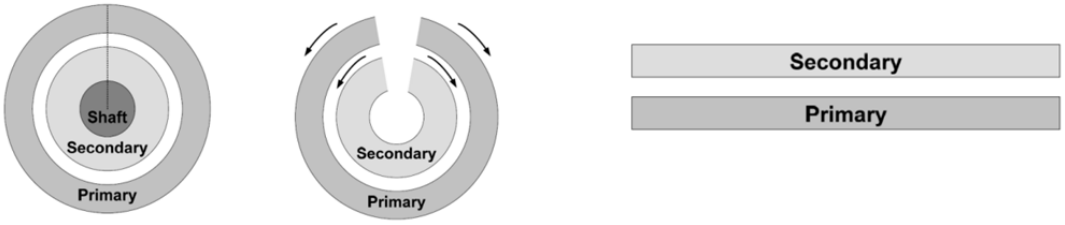
CHƯƠNG 1. ĐỘNG CƠ TUYẾN TÍNH VÀ ĐIỀU KHIỂN VỊ TRÍ HỆ TRUYỀN ĐỘNG SỬ DỤNG ĐỘNG CƠ TUYẾN TÍNH

1.1. Lịch sử phát triển của động cơ tuyến tính

Từ năm 1840, Charles Wheastone đã mô tả động cơ điện tuyến tính (còn gọi là động cơ truyền động thẳng) ở Viện Hoàng Gia London, tuy nhiên động cơ này chưa được triển khai trong thực tế. Năm 1905 Alfred Zehden ở Frankfurt-am-Main đã mô tả động cơ điện tuyến tính trong truyền động tàu thủy, thang máy. Năm 1935 kỹ sư người Đức Hermann Kemper đã xây dựng mô hình động cơ tuyến tính. Mãi đến năm 1947, Eric Laithwaite, một kỹ sư điện người Anh, đã sử dụng động cơ điện tuyến tính trong hệ thống truyền động máy dệt công nghiệp. Nghiên cứu của Laithwaite đã nhận được sự quan tâm của các nhà khoa học. Công trình này được Viện nghiên cứu Hoàng Gia Anh công nhận vào những năm 60 của thế kỷ XX với tên gọi: Máy điện của tương lai.

1.2. Cấu tạo và phân loại động cơ tuyến tính

Để hiểu rõ hơn về động cơ tuyến tính ta có thể hình dung ra một động cơ quay tròn bất kỳ nào, khi tăng bán kính của động cơ đến vô cùng, sẽ thu được hình ảnh rotor và stator song song với nhau (hình 1.1). Trong chuyển động tương đối khi chọn gốc tọa độ gắn với hệ quy chiếu nào ta sẽ suy ra được chuyển động tương đối của thành phần còn lại so với gốc tọa độ. Với quan điểm như vậy động cơ tuyến tính sẽ gồm hai thành phần: Thành phần thứ nhất nhận dòng năng lượng điện đi tới (phần sơ cấp), thành phần thứ hai là dòng năng lượng đưa ra dưới dạng cơ năng (phía thứ cấp). Từ quan điểm trên ta có thể thấy với động cơ tuyến tính phần tạo chuyển động thẳng có thể là phần stator hay phần rotor của máy điện quay truyền thống, từ đó tạo ra những động cơ tuyến tính tương ứng.



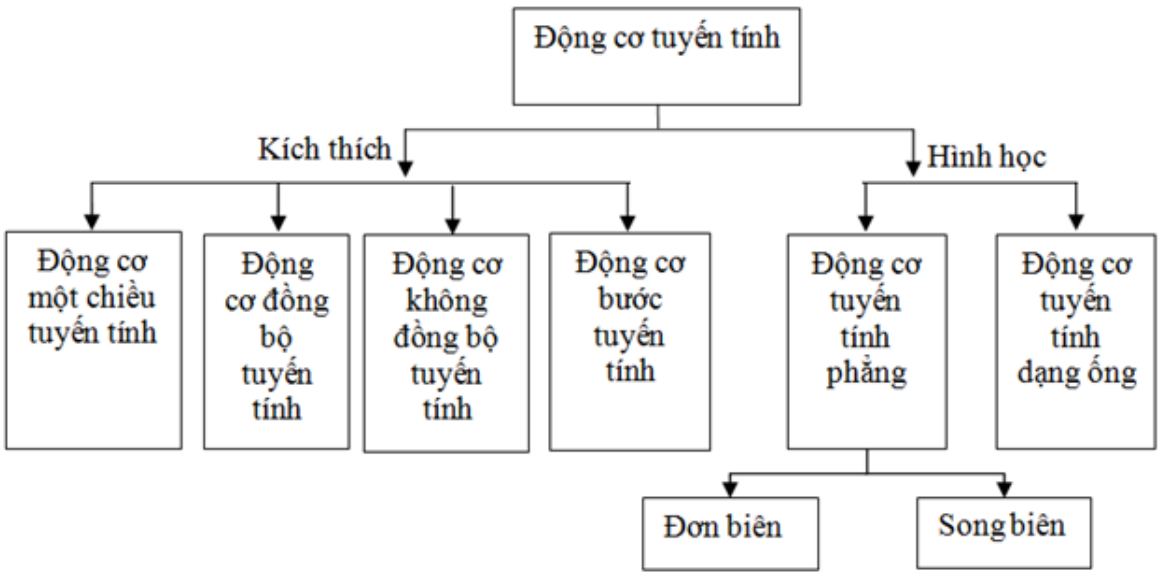
Hình 7.1. Nguyên lý chuyển đổi từ động cơ quay sang động cơ tuyến tính

Từ nguyên lý cơ bản trên, động cơ tuyến tính được phát triển với cấu tạo khác nhau tương ứng dựa vào mục đích sử dụng.

Ban đầu động cơ tuyến tính chủ yếu được sử dụng cho hệ thống giao thông vận tải. Hiện nay động cơ tuyến tính được sử dụng để thay thế một hệ thống sử dụng động cơ quay và các thiết bị cơ khí để tạo ra một chuyển động tuyến tính (thẳng) trực tiếp.

Theo cấu trúc hình học, động cơ tuyến tính được chia thành 2 loại chính: dạng phẳng và dạng ống.

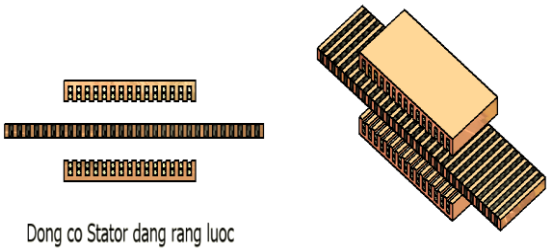
Theo nguồn kích thích, động cơ tuyến tính có thể chia thành 4 loại chính: Động cơ một chiều tuyến tính, động cơ đồng bộ tuyến tính, động cơ không đồng bộ tuyến tính, động cơ bước tuyến tính.



Hình 1.8. Phân loại động cơ tuyến tính

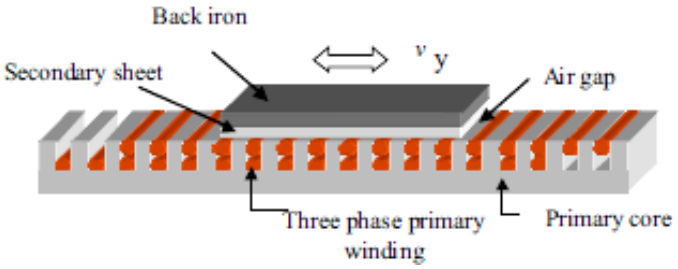
Ngoài ra, thực tế cho thấy tùy theo những ứng dụng cụ thể mà động cơ tuyến tính còn được phân loại như sau:

- Động cơ có stator dạng răng lược.



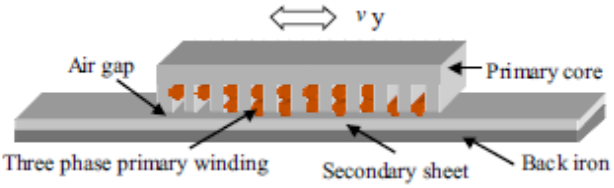
Hình 1.9. Động cơ tuyến tính có stator dạng răng lược

- Động cơ cơ stator dạng dài: Chiều dài của phần cung cấp thường lớn hơn nhiều lần phần kích thích (cảm ứng), đa số trong các trường hợp thì phần kích thích chính là phần chuyển động.



Hình 1.10. Động cơ tuyến tính có stator dài

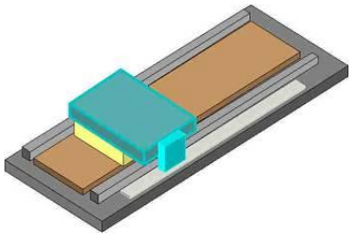
- Động cơ có stator dạng ngắn: Chiều dài của phần cung cấp ngắn hơn (hoặc bằng) phần kích thích (cảm ứng), đa số trong các trường hợp thì phần cung cấp chính là phần chuyển động.



Hình 1.11. Động cơ tuyến tính có stator ngắn

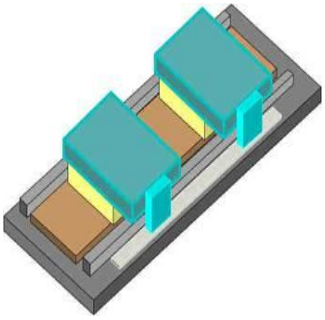
Dựa vào số trục di chuyển, động cơ tuyến tính có hai loại:

- Loại 1 trục.



Hình 1.12. Động cơ tuyến tính 1 trục

- Loại 2 trục.



Hình 1.7. Động cơ tuyến tính 2 trục

Trong số các động cơ tuyến tính ở trên, theo tài liệu [32], lực đẩy của loại động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu là lớn nhất, mặt khác, trong các loại động cơ tuyến tính đồng bộ kích thích vĩnh cửu : dạng phẳng đơn, dạng phẳng có kết cấu răng lược và dạng ống thì dạng phẳng đơn dễ chế tạo hơn, giá thành rẻ hơn, do đó đề tài sẽ chọn đối tượng là động cơ tuyến tính đồng bộ kích thích vĩnh cửu dạng phẳng đơn.

Để hiểu rõ hơn về cấu tạo của động cơ tuyến tính đồng bộ kích thích vĩnh cửu dạng phẳng đơn, bảng 1.1 đưa ra so sánh giữa các phần chuyển động và cố định của động cơ đồng bộ ba pha kích thích vĩnh cửu quay tròn và động cơ tuyến tính đồng bộ kích thích vĩnh cửu dạng phẳng đơn.

Bảng 1.1. So sánh về cấu tạo giữa động cơ đồng bộ kích thích vĩnh cửu (ĐB-KTVC) và động cơ tuyến tính kiểu đồng bộ kích thích vĩnh cửu (ĐCTT ĐB-KTVC)

Loại ĐC / **Bộ phận**	**ĐB-KTVC**	**ĐCTT ĐB-KTVC**
Phần chuyển động	Nam châm vĩnh cửu cực ẩn hoặc cực lồi gắn trên lõi thép có dạng khối trụ tròn chuyển động quay quanh một trục.	Mạch từ và dây quấn 3 pha trải phẳng, chuyển động tịnh tiến (chuyển động thẳng)
Phần cố định	Mạch từ có kết cấu hình vành trụ tròn, trong có xẻ rãnh đặt dây quấn 3 pha.	Nam châm vĩnh cửu gồm nhiều cực từ đặt liên tiếp nhau, cực tính luân phiên nhau.

1.3. Nguyên lý làm việc của động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu dạng phẳng đơn.

Nguyên lý làm việc tương tự như động cơ 3 pha kích thích vĩnh cửu quay tròn thông thường, nhưng stator và rotor được trải phẳng và thay cho momen là lực điện từ đẩy động chuyển động thẳng .

Để hiểu rõ hơn về nguyên lý làm việc của động cơ tuyến tính ba pha đồng bộ kích thích vĩnh cửu dạng phẳng đơn, bảng 1.2 đưa ra so sánh về nguyên lý làm việc của động cơ đồng bộ ba pha kích thích vĩnh cửu quay tròn và động cơ tuyến tính đồng bộ kích thích vĩnh cửu dạng phẳng đơn.

Bảng 1.2 So sánh về nguyên lý làm việc của động cơ đồng bộ ba pha kích thích vĩnh cửu quay tròn và động cơ tuyến tính đồng bộ kích thích vĩnh cửu dạng phẳng đơn.

Loại ĐC / **Tiêu chí So sánh**	**ĐB-KTVC**	**ĐCTT ĐB-KTVC**
Dạng từ từ trong máy	Từ trường ba pha quay tròn	Từ trường ba pha chuyển động tịnh tiến (chuyển động thẳng)
Tác nhân gây chuyển động	Mô men do tương tác giữa từ trường quay tròn stator với từ trường nam châm vĩnh cử phía rotor.	Lực đẩy phần động do tương tác giữa từ trường ba pha chuyển động tịnh tiến (chuyển động thẳng) với từ trường nam châm vĩnh cửu phía stator.

1.4. Nguyên tắc chung điều khiển chung độngcơ tuyến tính ba pha kích thích vĩnh cửu

Cũng như những phương pháp đã được thực hiện đối với động cơ quay, lúc này phương pháp điều khiển cho ĐCTT vẫn dựa trên hai hướng chính dựa vào nguyên lý điều khiển vector và nguyên lý điều khiển vô hướng.

1.4.1. Nguyên lý điều khiển vô hướng

Tài liệu [29] đã chỉ ra các phương pháp đại diện cho hướng nghiên cứu sử dụng nguyên lý điều khiển vô hướng: U/f không đổi (với mục đích duy trì từ thông khe hở không đổi giúp tạo ra khả năng sinh mômen mong muốn), điều khiển độ trượt,… Tuy nhiên việc tạo ra từ thông khe hở không đổi sẽ gặp khó khăn khi phụ tải thay đổi vì sụt áp trên Stator phụ thuộc vào dòng chảy qua nó [29] và điều này được khắc phục bằng cách điều khiển U/ f sao cho từ thông khe hở là hàm của mômen tải. Các phương pháp dựa trên nguyên lý điều kiển vô hướng có ưu điểm dễ thực hiện nhưng chúng đều gặp khó khăn trong việc nâng cao chất lượng của hệ truyền động (đặc biệt ở vùng tốc độ thấp).

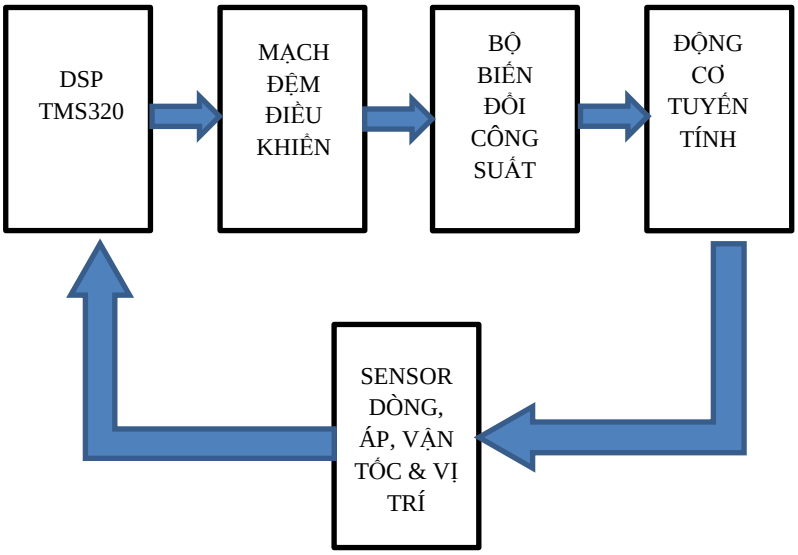
1.4.2. Nguyên lý điều khiển vector

Nguyên lý điều khiển vector đã được trình bày trong hệ thống tài liệu [7,8,9,10]. Đối với ĐCTT, việc vận dụng nguyên lý này cần dựa trên một hệ thống các vector mô tả một cách tường minh các đại lượng vật lý (dòng điện, điện áp, từ thông,…) được trình bày tài liệu [2].

Các phương pháp điều khiển dựa trên nguyên tắc điều khiển vector có ưu điểm là nâng cao được chất lượng của hệ truyền động so với phương pháp điều khiển vo hứng, do đó đề tài chọn phương pháp điều khiển vector để thực hiện, trong đó, đề tài sử dụng phương pháp điều khiển phi tuyến backstepping để thiết kế cho bộ điều khiển dòng điện và sử dụng bộ điều khiển PID mờ cho mạch vòng điều khiển vị trí.

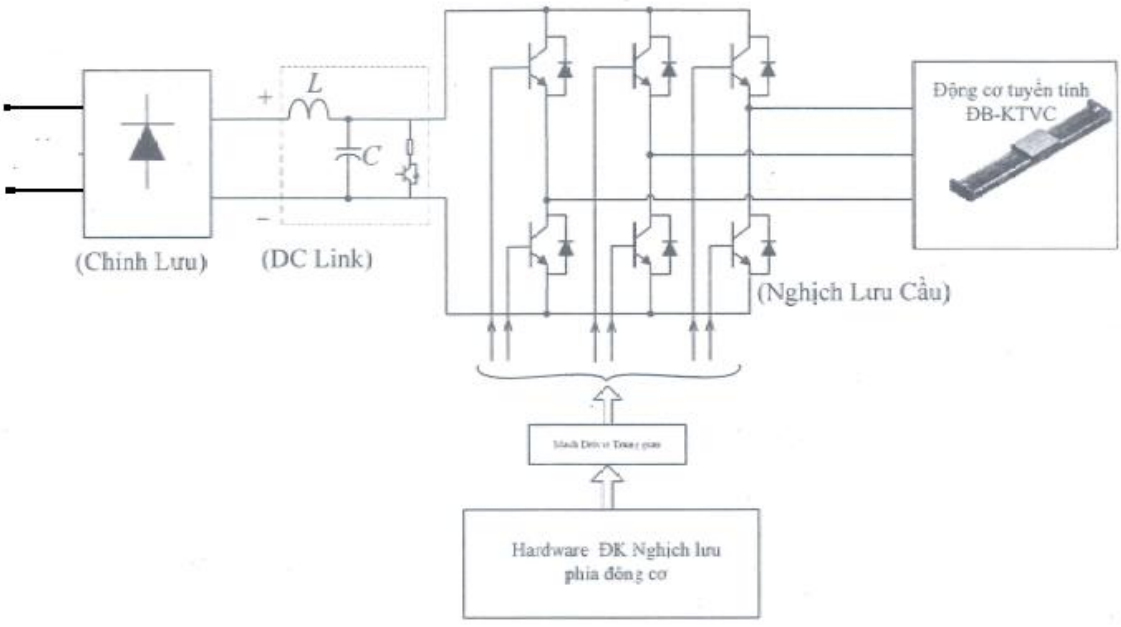
CHƯƠNG 2. XÂY DỰNG PHẦN CỨNG

2.1. Sơ đồ khối hệ thống



Hình 2.1. Sơ đồ khối hệ thống

2.2. Phần cứng mạch động lực



Hình 2.2. Sơ đồ cấu trúc phần cứng mạch lực cho động cơ tuyến tính đồng bộ ba pha kích thích vĩnh cửu

2.2.1. Lựa chọn thiết bị chuyển mạch

Để lựa chọn cho thiết bị chuyển mạch công suất, trong đề tài chọn transistor IGBT, lý do IGBT có những ưu điểm nổi trội như cho phép đóng cắt dễ dàng, chức năng điều khiển nhanh, chịu áp lớn, thường 600V tới 1.5kV, tải dòng lớn, cỡ xấp xỉ 1kA. Sụt áp bé và điều khiển dễ dàng bằng áp. Những ưu điểm này phù hợp với yêu cầu của hệ thống điều khiển vị trí sử dụng động cơ tuyến tính loại đồng bộ cũng như không đồng bộ ba pha.

2.2.2. Lựa chọn loại IGBT

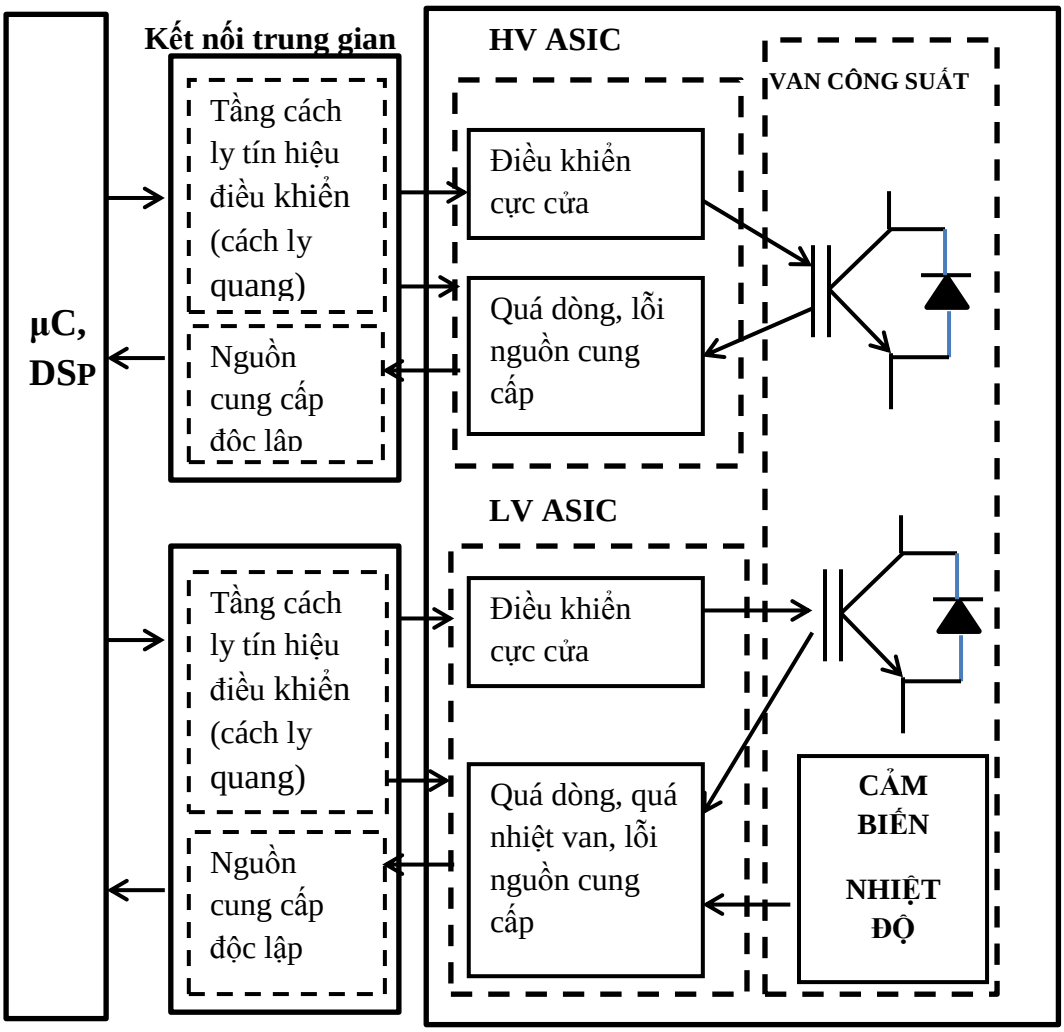
Căn cứ vào số lượng IGBT và mạch điều khiển các van IGBT hiện có trên thị trường, các bộ chuyển mạch IGBT có trên thị trường được phân thành ba loại sau:

- IGBT đơn.
- IGBT module.
- IGBT Integrated Motor Solution.

Hai loại sau gồm nhiều IGBT đơn được tích hợp trong một vi mạch. Số lượng IGBT đơn được tích hợp trong vi mạch phụ thuộc vào nhà chế tạo. Điểm khác giữa IGBT module và IGBT Integrated Motor Solution là ở IGBT module không có mạch driver cho IGBT, còn ở IGBT Integrated Motor Solution có mạch Driver cho các IGBT. Một điểm khác nữa là IGBT module được chế tạo với công suất lớn , còn IGBT Integrated Motor Solution được chế tạo với công suất trung bình.

Các bộ nghịch lưu truyền thống gồm sáu IGBT đơn và có mạch driver cho các IGBT. Việc thiết kế một bộ nghịch lưu sử dụng sáu IGBT đơn mất nhiều thời gian và giá thành cao, hơn nữa lại làm việc không tin cây. Vì vậy trong đề tài này, loại nghịch lưu IGBT Integrated Motor Solution được sử dụng. IGBT Integrated Motor Solution có sáu IGBT đơn, sáu diode mắc song song và được tích hợp với mạch driver. Đây là loại phù hợp với nghịch lưu nguồn áp.

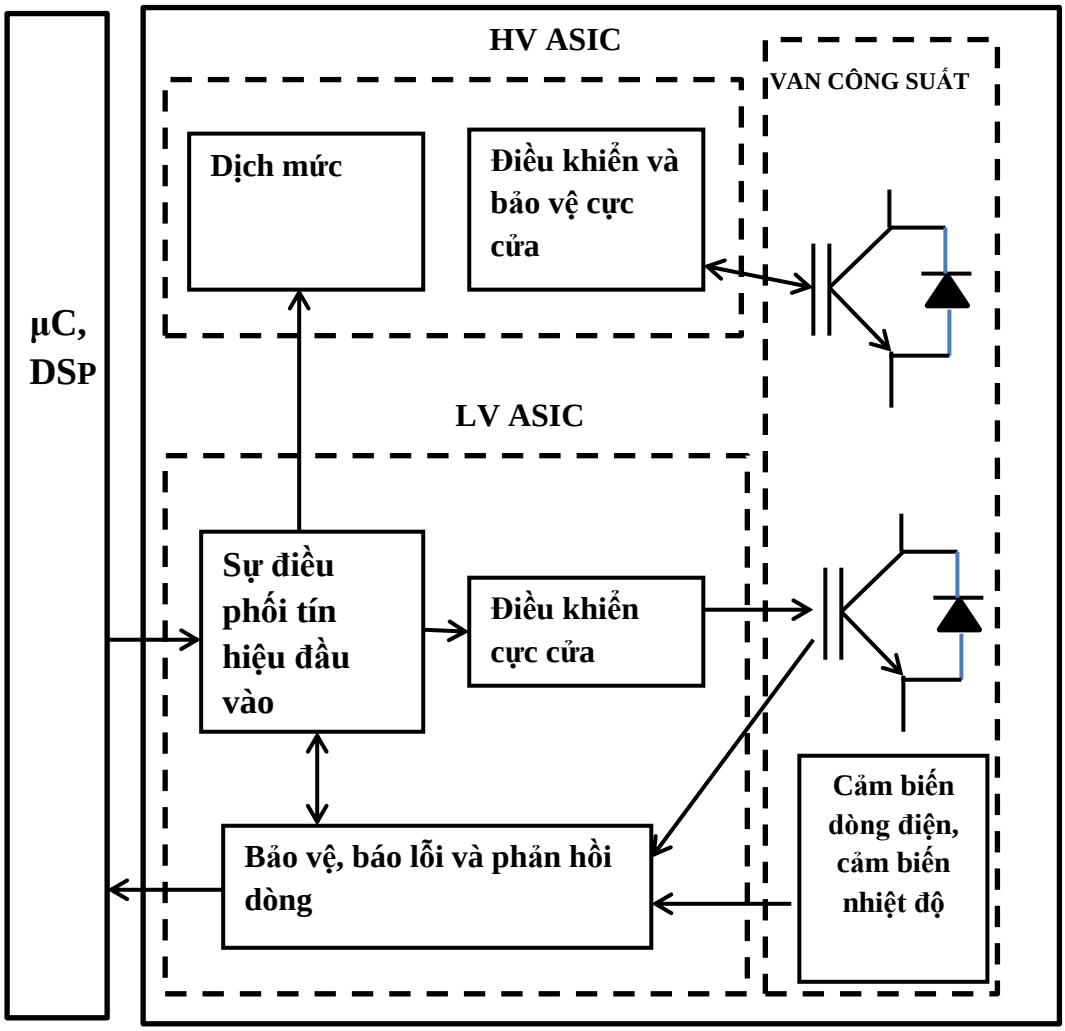
Với loại nghịch lưu IGBT Integrated Motor Solution, hiện nay ta có thể lựa chọn các modul công suất thông thường (Conventional Intelligent Power Modules).



Hình 2.3. Sơ đồ khối IPM thông thường

Hình 2.3 chỉ ra sơ đồ khối cấu tạo của Conventional IPM. Theo đó, Conventional IPM được tích hợp các van công suất với các mạch tích hợp chuyên dụng phía thấp áp (LV ASIC - Low voltage Application Specific Integrated Circuit) để kích mở van và cung cấp một số chức năng bảo vệ đã được sử dụng rộng rãi trong hệ truyền động xoay chiều. Modul loại này có một vài ưu điểm như : Giảm thời gian thiết kế, nâng cao độ tin cậy; giảm tổn hao công suất bằng việc tối ưu hóa đồng thời các van công suất và chức năng bảo vệ trên cùng một modul; cải thiện khả năng chế tạo do giảm số lượng các linh kiện phụ trợ. Tuy nhiên Conventional IPM vẫn còn có hạn chế nhất định. Theo đó, để có thể ghép nối giữa *µC / DSP* (vi điều khiển/vi xử lý tín hiệu số) và Convention IPM cần có tầng kết nối trung gian

(hình 2.3) bao gồm các mạch cách ly quang đảm bảo cách ly μC / DSP với phía cao áp (các van trên). Điều này dẫn tới số lượng nguồn điều khiển sử dụng để kích mở van tăng lên. Đối với các hệ truyền động công suất nhỏ, hạn chế nêu trên sẽ làm gia tăng thêm chi phí và kích thước của thiết bị. Việc modul công suất thông minh chuyên dụng ASIPM (Application Specific IPM) ra đời đã khắc phục được mặt hạn chế của Convention IPM.



Hình 2.4. Sơ đồ khối ASIPM

Hình 2.4 chỉ ra sơ đồ khối cấu tạo của ASIPM. Bằng việc tích hợp thêm công nghệ HVIC (High Voltage Integrated Circuit) có chức năng dịch mức và điều khiển kích mở van, ASIPM cho phép kết nối trực tiếp 6 tín hiệu PWM từ µC/DSP tới đầu vào của nó mà không cần cách ly quang, chỉ cần một nguồn điều khiển duy

nhất để kích mở van. Ngoài ra, một số ASIPM còn tích hợp sẵn cả cảm biến đo dòng bên trong giúp cho việc thiết kế khâu đo lường trở nên thuận tiện hơn. Xuất phát từ các phân tích trên, modul công suất thông minh chuyên dụng được lựa chọn để thiết kế mạch động lực cho hệ thống.

Hiện nay có các hãng sản xuất ASIPM sau: Fairchild, International Rectifier, ST Semicondoctor. Sau khi phân tích các đặc tính kỹ thuật, tính phổ dụng và giá cả, đề tài chọn modul công suất thông minh chuyên dụng ASIPM loại FSBB30CH60C của hãng Fairchild

2.2.3. Thực hiện bộ nghịch lưu nguồn áp ba nhánh sử dụng modul công suất thông minh chuyên dụng ASIPM loại FSBB30CH60C.

2.2.3.1. Các đặc tính kỹ thuật của FSBB30CH60C

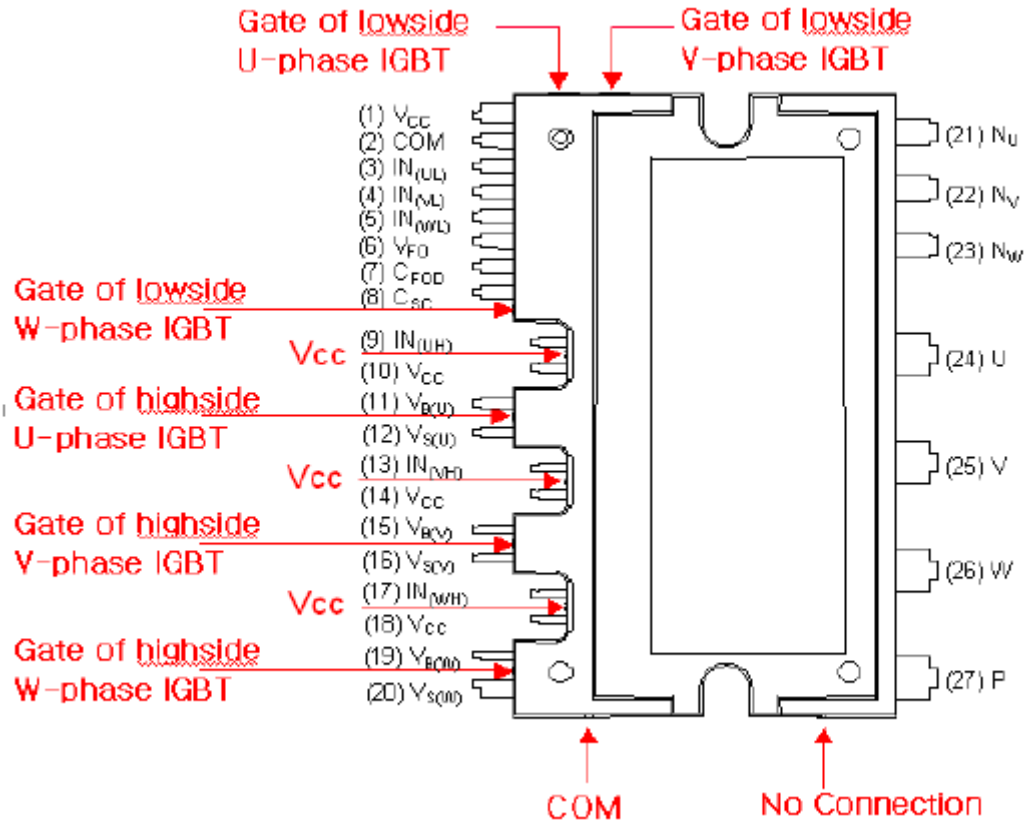
- Là bộ nghịch lưu sử dụng 6 IGBT ba pha ba nhánh, điện áp định mức 600V, dòng định mức là 20A tích hợp với mạch driver và mạch bảo vệ.

- Giảm nhiễu, hiệu suất cao, nhỏ, nhẹ, độ chính xác trong điều khiển được nâng cao.

- Tích hợp chức năng bảo vệ giảm thấp điện áp, bảo vệ quá nhiệt, bảo vệ ngắn mạch.

- Cho phép kết nối trực tiếp 6 tín hiệu PWM từ μC/DSP tới đầu vào của nó mà không cần cách ly quang và cac mạch phụ khác.

2.2.3.2. Các chân vào và ra của FSBB30CH60C



Hình 2.5. Sơ đồ chân của FSBB30CH60C

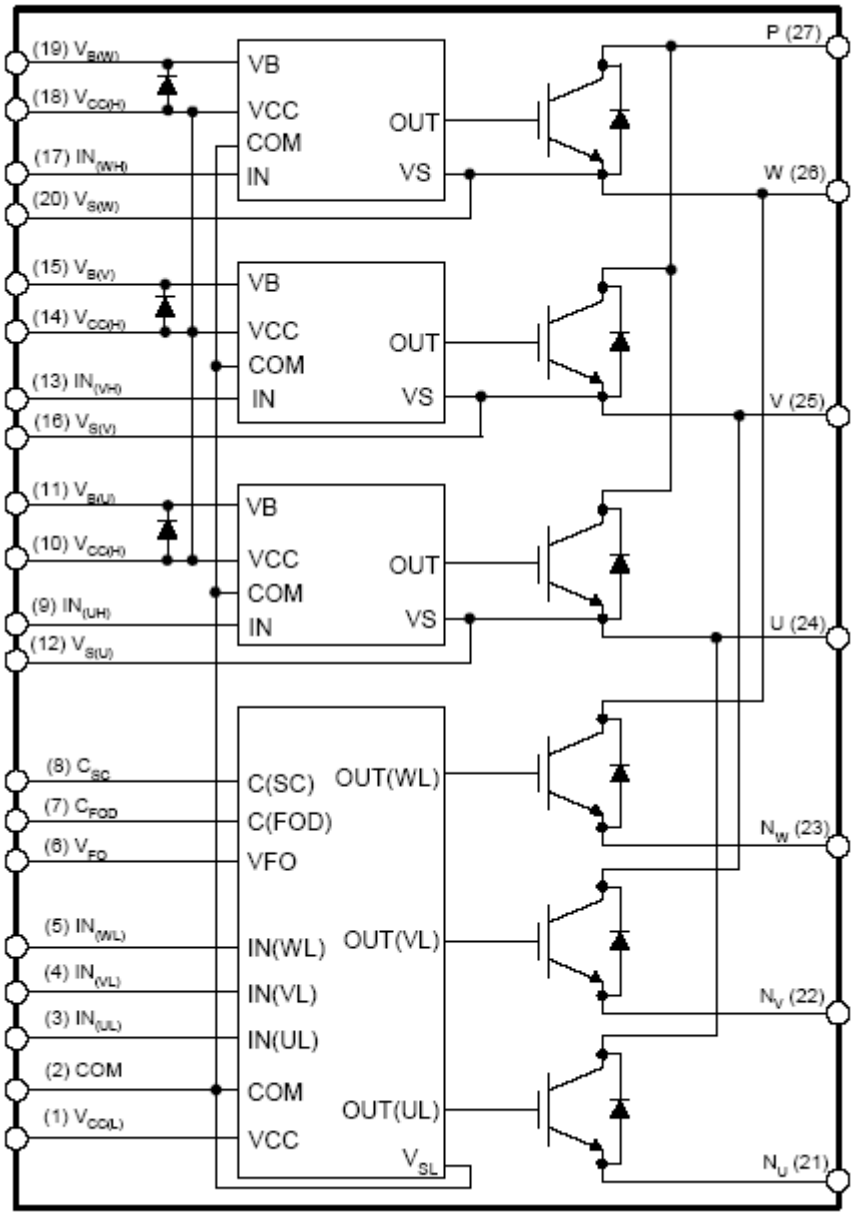
Chức năng của cac chân vào ra của FSBB30CH60C được mô tả ở bảng 2.1.

Bảng 2.1. Chức năng của cac chân vào ra của FSBB30CH60C

Số chân	Tên chân	Mô tả chức năng
1	VCC(L)	Chân nối với nguồn dương chung
2	COM	Chân nối đất chung
3	$IN_{(UL)}$	Đầu vào tín hiệu cho pha U phía thấp
4	$IN_{(VL)}$	Đầu vào tín hiệu cho pha V phía thấp
5	$IN_{(WL)}$	Đầu vào tín hiệu cho pha W phía thấp
6	V_{FO}	Chân ra báo lỗi
7	C_{FOD}	Chân nối tụ để tạo thời gian báo lỗi
8	C_{SC}	Tụ lọc thông thấp cho đầu vào bảo vệ ngắn mạch

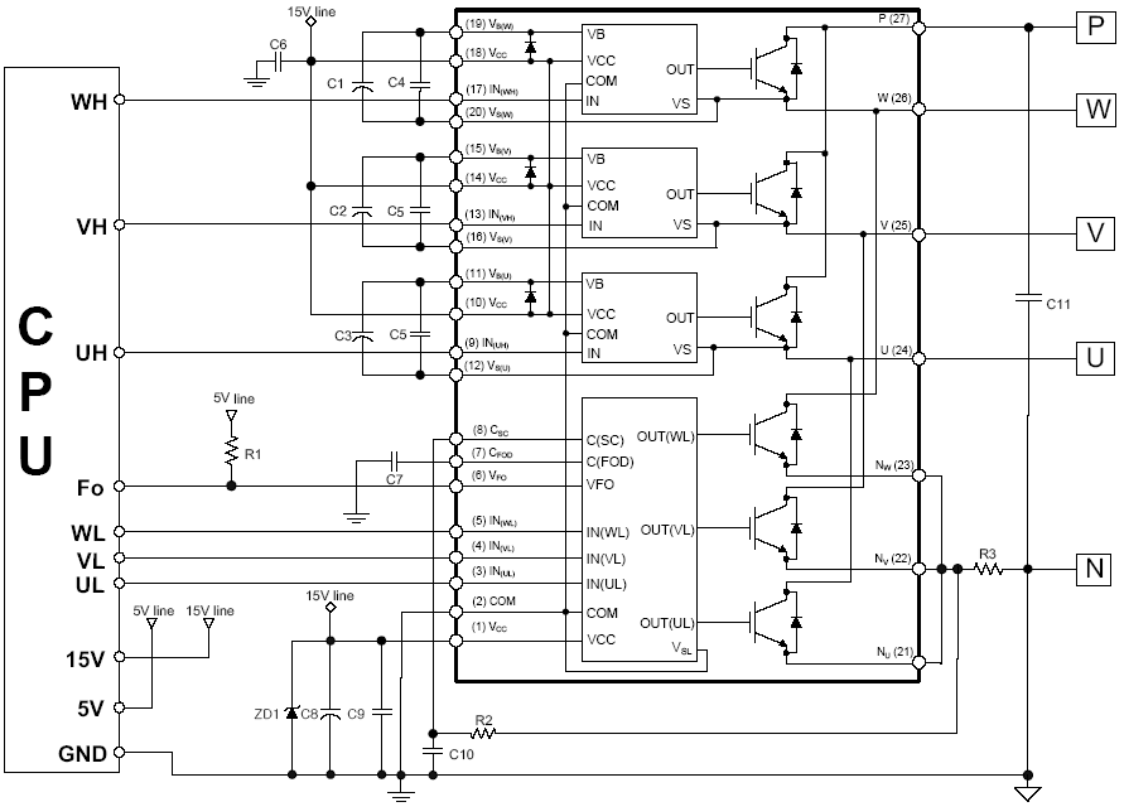
9	$IN_{(UH)}$	Đầu vào tín hiệu cho pha U phía cao
10	$V_{CC(H)}$	Chân nối với điện áp nguồn chung phía cao
11	$V_{B(U)}$	Chân điện áp phía cao điều khiển IGBT pha U
12	$V_{S(U)}$	Chân nối đất phía cao điều khiển IGBT pha U
13	$IN_{(VH)}$	Đầu vào tín hiệu cho pha V phía cao
14	$V_{CC(H)}$	Chân nối với điện áp nguồn chung phía cao
15	$V_{B(V)}$	Chân điện áp phía cao điều khiển IGBT pha V
16	$V_{S(V)}$	Chân nối đất phía cao điều khiển IGBT pha V
17	$IN_{(WH)}$	Đầu vào tín hiệu cho pha V phía cao
18	$V_{CC(H)}$	Chân nối với điện áp nguồn chung phía cao
19	$V_{B(W)}$	Chân điện áp phía cao điều khiển IGBT pha W
20	$V_{S(W)}$	Chân nối đất phía cao điều khiển IGBT pha W
21	N_U	Chân nối tụ kết nối với âm nguồn một chiều cho pha U
22	N_V	Chân nối tụ kết nối với âm nguồn một chiều cho pha V
23	N_W	Chân nối tụ kết nối với âm nguồn một chiều cho pha W
24	U	Đầu ra pha U
25	V	Đầu ra pha V
26	W	Đầu ra pha W
27	P	Chân nối với dương nguồn một chiều đầu vào cho bộ nghịch lưu

2.2.3.3. Sơ đồ mạch bên trong của FSBB30CH60C



Hình 2.6. Sơ đồ mạch bên trong của FSBB30CH60C

2.2.3.4. Sơ đồ ghép nối với CPU (DSP hay vi xử lý)



Hình 2.7. Sơ đồ ghép nối FSBB30CH60C với CPU (DSP hay Vi xử lý)

2.2.4. Lựa chọn thiết bị cho bộ chỉnh lưu

Để lựa chọn thiết bị cho bộ chỉnh lưu, đề tài chọn loại bộ chỉnh lưu cầu tích hợp 4 Diod trong một IC chỉnh lưu cầu tích hợp. yêu cầu về thông số:

- Điện áp vào là nguồn xoay chiều một pha, có trị hiệu dụng U=220V, tần số 50Hz, công suất 0.25 kW.

- Điện áp một chiều đầu ra: $U_{do} = \frac{2\sqrt{2}}{\pi} U = 0,9U = 0,9.220 = 198\,V$

- Dòng điện tải đầu ra: $I_d = \frac{P_t}{U_{do}} = \frac{250}{198} = 1,26\,A$

- Điện áp ngược cực đại đặt lên Diode chỉnh lưu:

$$U_{dng\max} = \sqrt{2}U = \sqrt{2}.220 = 311V$$

Căn cứ vào thông số của các bộ chỉnh lưu cầu một pha tích hợp hiện có trên thị trường của các hãng SHANGHAI SUNRISE ELECTRONICS CO., LTD, đề tài chọn bộ chỉnh lưu cầu một pha tích hợp có thông số sau:

- Loại bộ chỉnh lưu cầu một pha tích hợp: KBPC 2504

- Điện áp ngược cực đại cho phép đặt lên Diode chỉnh lưu: 400 V.

- Trị hiệu dụng điện áp vào cực đại cho phép: 280 V.

- Điện áp một chiều cực đại cho phép ở đầu ra: 400 V.

- Dòng điện trung bình cực đại sau bộ chỉnh lưu: 25 A ở nhiệt độ 55^0C.

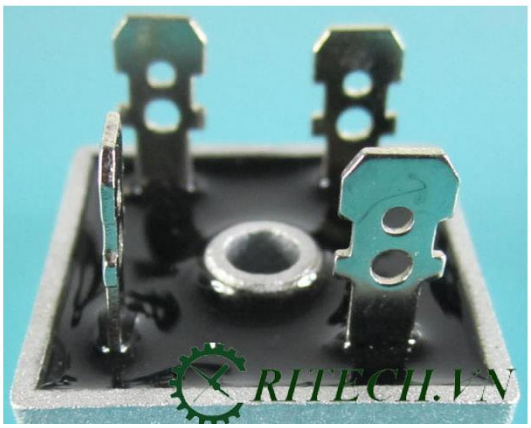
- Dòng điện xung cực đại cho phép trong khoảng thời gian 8,3ms: 300 A.

- Điện áp rơi cực đại trên bộ chỉnh lưu cầu một pha tích hợp ở tải 12,5 A DC: 1,1 V.

- Dòng điện ngược một chiều cực đại ở nhiệt độ 35^0C : 10 μA

- Dòng điện ngược một chiều cực đại ở nhiệt độ 100^0C : 500 μA.

- Nhiệt độ làm việc cho phép: -55^0C đến 150^0C.



Hình 2.8. IC chỉnh lưu cầu một pha tích hợp: KBPC 2504

2.3. Phần cứng mạch điều khiển

2.3.1. Giới thiệu DSP TMS320F2812

Thông số kỹ thuật:

+ Tần số xung nhịp 150MHz(chu kì lệnh 6,67ns)

+ Điện áp lập trình 3,3V

+ Hỗ trợ chuẩn JTAG

+ Đơn vị xử lí trung tâm CPU 32 bit

+ Quản lí theo địa chỉ tuyến tính 4M nhớ chương trình.

+ Quản lí theo địa chỉ tuyến tính 4M nhớ dữ liệu.

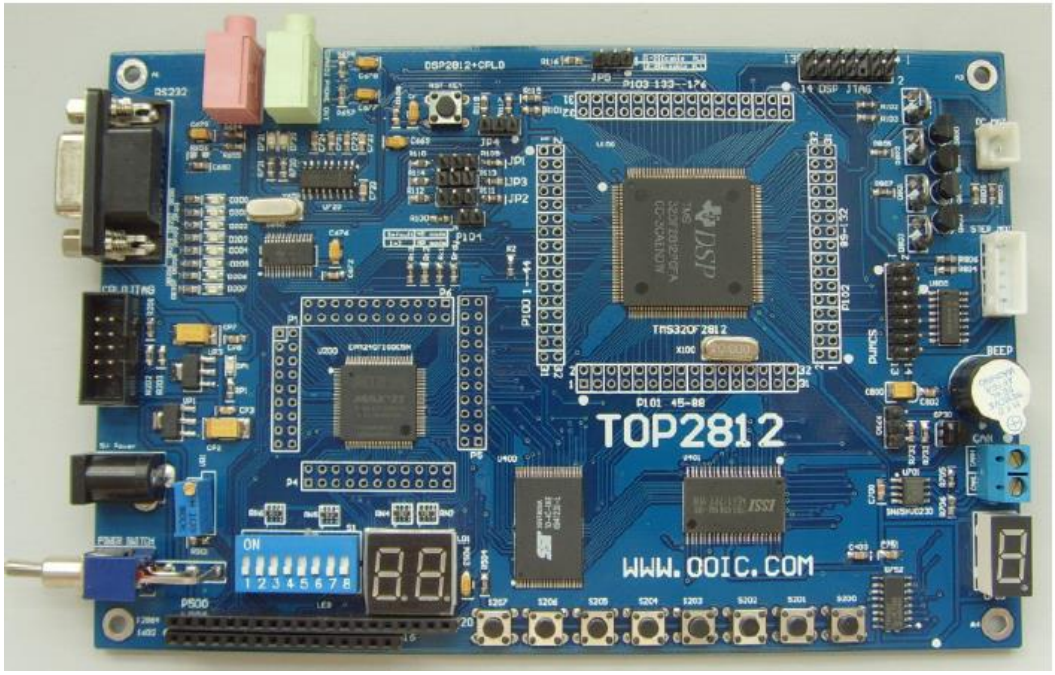
+ Có 128Kx16 Flash(4 mảng 8Kx16 và sáu mảng 6Kx16)

+ 1Kx16 loại OTP ROM.

+ Boot ROM(4Kx16)

+ 16 kênh ADC 12-Bit.

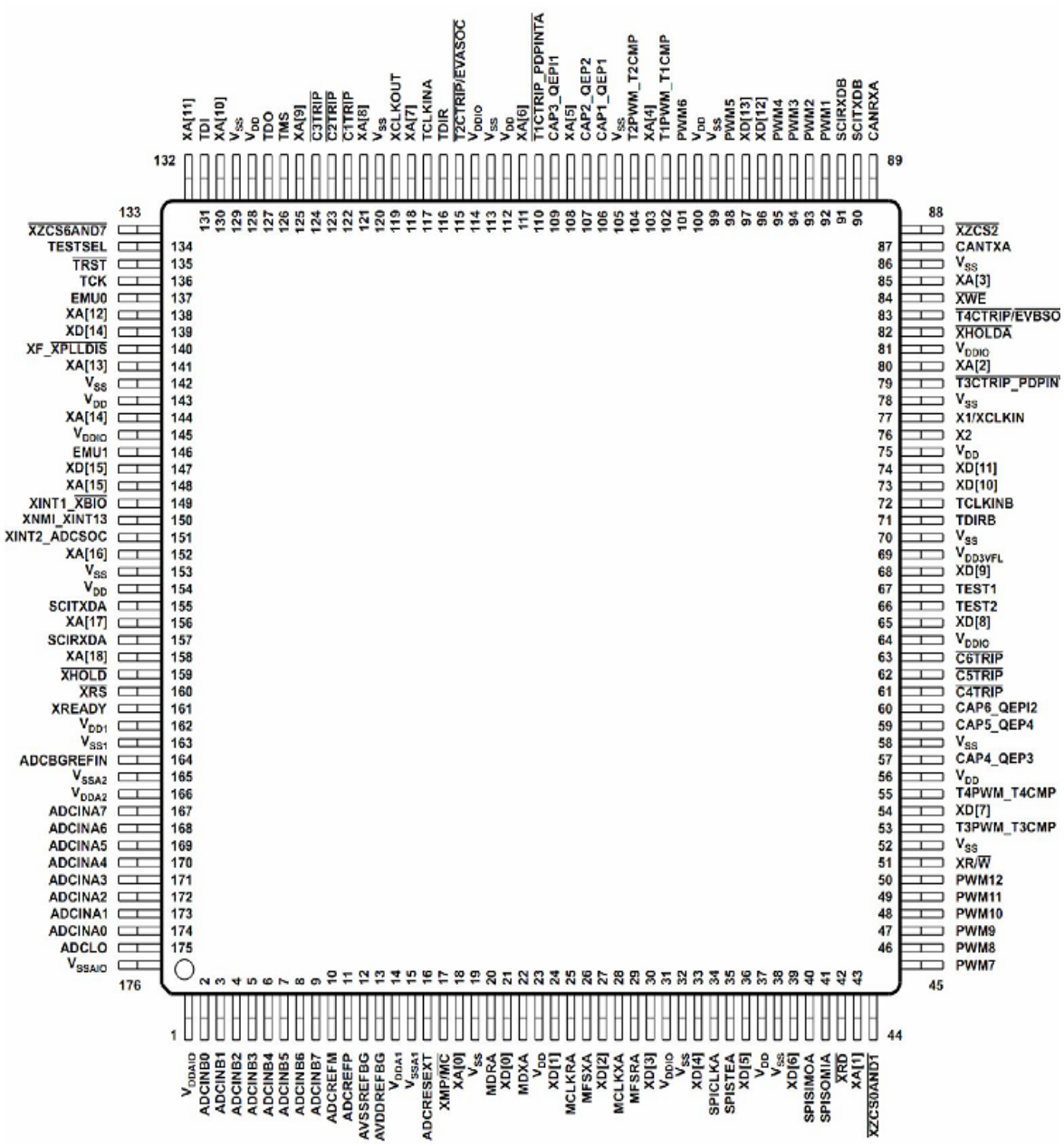
Xuất xứ: Mỹ



Hình 2.9. DSP TMS320F2812

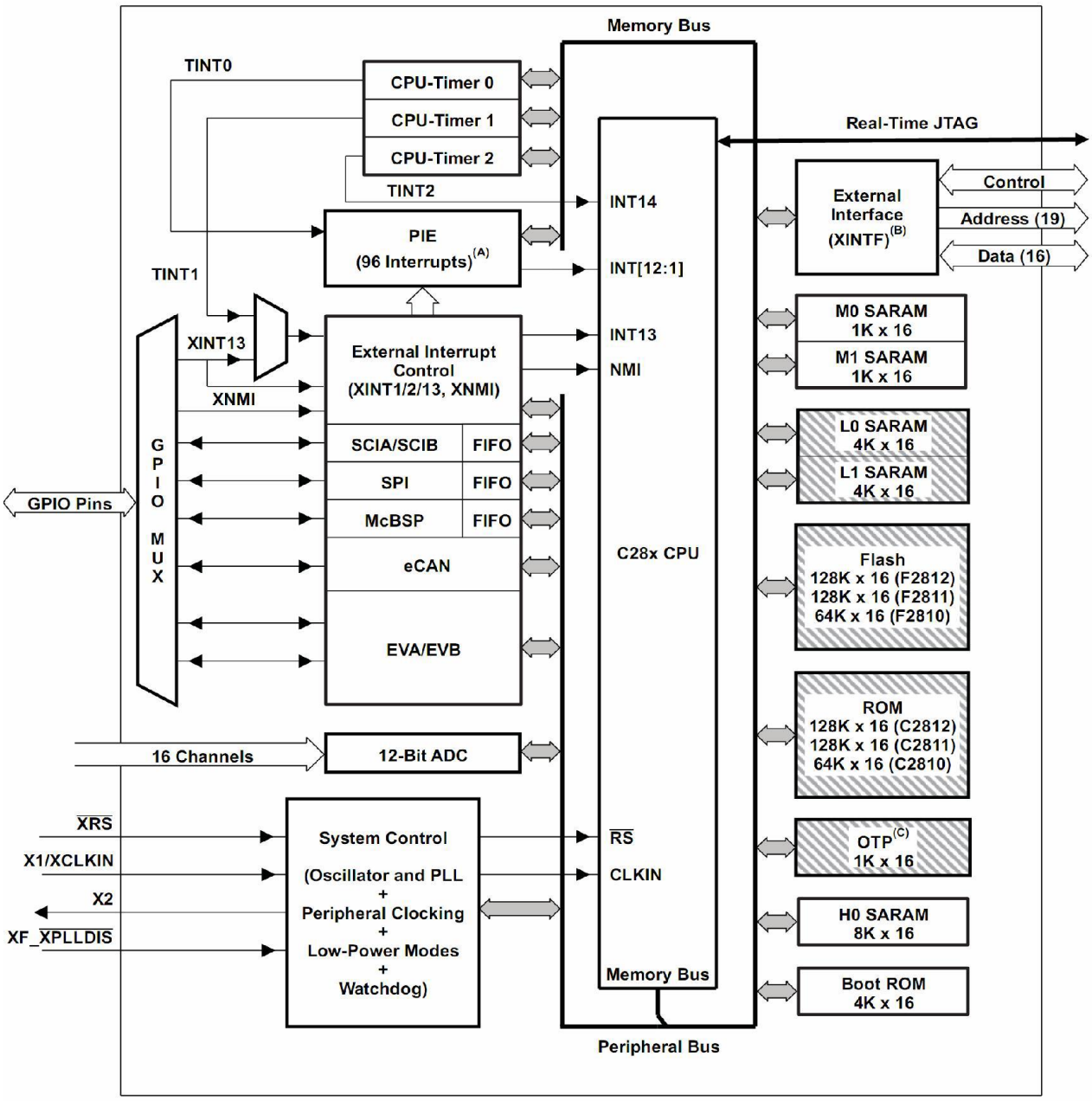
2.3.1.1. Phần cứng của vi xử lý F2812

- Sơ đồ chân của F2312



Hình 2.10. Sơ đồ 176 chân của vi xử lý TMS320F2812

2.3.1.2. Sơ đồ chức năng của vi xử lý TMS320F2812



Hình 2.11. Sơ đồ cấu trúc của vi xử lý TMS320F2812

Đặc điểm các khối trong sơ đồ cấu trúc của TMS320F2812 như sau

* Đơn vị xử lí trung tâm (Central Processing Unit: CPU) của TMS320C24x.

Dòng vi xử lí tín hiệu C28x là thành viên mới của họ TMS320C2000 mã nguồn C28x hoàn toàn tương thích với các chip loại 24x/240x, tạo điều kiện cho người sử dụng 240x có thể sử dụng các loại mảng mã quan trọng của họ. Thêm vào đó, dòng C28x có khả năng sử dụng hữu hiệu C/C++, cho phép người sử dụng

không chỉ cài đặt phần mềm điều khiển phức hợp bằng ngôn ngữ bậc cao, còn có thể phát triển các thuật toán học. Dòng C28x thực hiện có hiệu quả các phép tính cũng như các bài toán điều khiển thời gian thực thường thấy trong các hệ thống thiết bị điều khiển tự động. Khả năng cao đó đã cho phép loại bỏ yêu cầu sử dụng thêm vi xử lí thứ hai, thường thấy trong nhiều hệ thống. phép nhân MAC 32x32 bit MAC của C28x và các khả năng xử lí 64 bit đã cho phép C28x xử lí hữu hiệu các vấn đề tính số bậc cao, mà trong nhiều trường hợp đòi hỏi phải dùng tới giải pháp vi xử lí dấu phẩy động đắt tiền. Thêm vào đó, tốc độ đáp ứng ngắt nhanh kèm theo khả năng cất giữ nội dung các thanh ghi quan trọng, đã cho phép F2812 xử lí các sự kiện bất thường với thời gian trễ bé nhất.

* Hệ thống BUS ghép nối với bộ nhớ (Memory bus, cấu trúc Harvard)

Cũng như những chip DSP khác, hệ thống nhiều bus được dùng để trao đổi số liệu giữa CPU, bộ nhớ và ngoại vi. Kiến trúc bộ nhớ của C28x chứa những bus đọc chương trình, bus đọc số liệu và ghi số liệu. Bus đọc chương trình có 22 lines địa chỉ và 32 lines dữ liệu. 32 lines dữ liệu cho phép truy cập 32 bit chức năng trong 1 nhịp duy nhất. Kiến trúc nhiều bus, còn gọi là Harvard Bus, cho phép C28x lấy lệnh, đọc và ghi dữ liệu trong vòng 1 chu kì máy. Tất cả những ngoại vi và bộ nhớ được gắn vào bus bộ nhớ có thể được phân cấp ưu tiên truy cập bộ nhớ. Nói chung, ưu tiên của bus bộ nhớ truy cập có thể được tóm tắt như sau:

- *Cao nhất:*

+ Ghi dữ liệu (ghi dữ liệu và chương trình không thể xảy ra đồng thời trên bus bộ nhớ.)

+ Viết chương trình (ghi dữ liệu và chương trình không thể xảy ra đồng thời trên bus bộ nhớ.)

+ Đọc dữ liệu

+ Đọc chương trình

- Thấp nhất:

Nạp (Đọc chương trình và nạp chương trình không đồng thời xảy ra trên bus bộ nhớ.)

* Peripheral Bus

Để cho phép di chuyển thiết bị ngoại vi giữa họ Texas Instruments (TI ™) DSP khác nhau của thiết bị, F281x và C281x áp dụng một tiêu chuẩn cho kết nối tín hiệu ngoại vi.

Hai phiên bản của bus ngoại vi được hỗ trợ trên F281x và C281x. Một phiên bản chỉ hỗ trợ truy cập 16-bit (được gọi là thiết bị ngoại vi khung 2) và điều này vẫn giữ tính tương thích với thiết bị ngoại vi tương thích C240x. Phiên bản khác hỗ trợ cả hai đường dẫn truy cập 16 và 32 bit (được gọi là thiết bị ngoại vi khung 1)

* Chuẩn JTAG thời gian thực (Real- timeJTAG) và khả năng phân tích

Họ F281x và C281x được tích hợp chuẩn JTAG IEEE 1149.1. Hơn nữa, họ F281x và C281x hỗ trợ:

- Mode thời gian thực cho các lệnh thực hiện giữa nôi dung các ô nhớ, các ngoại vi và các thanh ghi trong khi nhân xử lý đang chạy, vẫn đang thực hiện mã lệnh hay đang phục vụ ngắt.

Nhờ đó, người sử dụng có thể thực hiện các lệnh đơn của nhiệm vụ không có đòi hỏi khắc nghiệt về thời gian, trong khi vẫn đang liên tục phục vụ các ngắt có đòi hỏi khắc nghiệt về thời gian mà không hề ảnh hưởng lẫn nhau. Dòng C28x có cài đặt sẵn Model thời gian thực trong Hardware cùng với CPU. Đây là đặc điểm chung của dòng C28x mà không cần đến phần mềm Monitor. Thêm vào đó một cơ chế phân tích đặc biệt bằng Hardware sẽ cho phép người sử dụng đặt điểm dừng hoặc điểm quan sát số liệu/ địa chỉ và tạo nên các sự kiện ngắt khác nhau.

Chuẩn JTAG gồm 5 chân:

TID (Kiểm tra dữ liệu trong)

TDO (kiểm tra dữ liệu ngoài)

TCK (kiểm tra đồng hồ)

TMS (kiểm tra chế độ chọn)

TRST(kiểm tra thiết lập tùy chọn)

* Giao diện ngoài (External Interface, XINTF)

Giao tiếp bất đồng bộ bao gồm 19 dây địa chỉ, 16 dây số liệu và 3 dây chọn chip. Các dây chọn chip được tổ chức liên kết với 5 vùng ngoài chip Zones 0, 1, 2, 6 và 7. Vùng 0 và 1 chia nhau sử dụng chung 1 dây chọn chip share a single chip-select, vùng 6 và 7 chung nhau sử dụng 1 dây khác. Từng vùng trong số 5 vùng trên có thể được lập trình với các số lượng nhịp đợi (wait states) khác nhau, tín hiệu đòi hỏi thời gian giữ của từng vùng đều có thể được lập trình với thời gian đợi hoặc không. Trạng thái đợi, dây chọn chip và tín hiệu đòi là khả trình đã làm cho việc giao tiếp với bộ nhớ ngoài và với ngoại vi trở nên rất linh hoạt.

* Flash

Chip F2812 có:

- 126Kx 16 bộ nhớ Flash nhúng trên phiến.

- 1Kx16 loại OTP

Bộ nhớ Flash được tổ chức thành

- 4 mảng 8Kx16 và

- 6 mảng 16Kx16

- Người sử dụng có thể xóa, lập trình hay kiểm tra các mảng hoàn toàn độc lập. Tuy nhiên không thể đồng thời chạy một mảng và xóa, ghi một mảng khác. Kĩ thuật chồng kênh khi(pipelining) khi truy cập bộ nhớ đã làm tăng tính năng của bộ nhớ Flash. Bộ nhớ Flash/OTP được tổ chức xên ở cả hai không gian nhớ chương trình và số liệu, phục vụ cả hai nhu cầu chạy chương trình hoặc cất giữ số liệu.

*Các khối nhớ M0, M1 SARAMs.

Mọi chip C28x đều có hai khối nhớ truy cậ đơn với kích cỡ 1K x 16 mỗi khối. Con trỏ ngăn xếp sẽ chỉ vào M1 sau khi reset. Khối M0 trùng địa chỉ với các khối RAM B0, B1, B2 của dòng 240x, và khi đó tổ chức cất dữ liệu của 240x cần phải lưu ý. Hai khối M0 và M1, cũng như mọi khối nhớ khác của dòng C28x, đều được tổ chức ở hai không gin nhớ chương trình và số liệu. Vì vậy, người sử dụng có thể dùng M0 hoặc M1 vào hai mục đích: Chạy mã chương trình hay xử lí số liệu.

Việc phân vùng sử dụng sẽ được quyết định khi ghép các Module software. Dòng C28x có tổ chức bộ nhớ thống nhất trong toàn bộ dòng. Điều này sẽ làm đơn giản hơn công tác lập trình bậc cao.

*Các khối nhớ L0, L1, H0 SARAMs.

Chip F2812 còn có thêm 16Kx 16 RAM loại truy cập đơn, chia thành 3 khối (4K+ 4K + 8K). Mỗi khối có thể được truy cập độc lập và do đó làm giảm nguy cơ mâu thuẫn khi chồng kênh. Mỗi khối đều được tổ chức ở cả hai không gian nhớ chương trình và số liệu.

*Boot ROM.

Bộ nhớ Boot Rom đã được nạp sẵn chương trình boot từ trước khi xuất xưởng. Các tín hiệu chọn chế độ boot cho phép khai báo với phần mềm gọi chương trình boot chế độ boot cần chọn khi khởi động hệ thống. Người sử dụng có thể lựa chọn:

- Chế độ boot thông thường.

- Hay chế độ nạp

- Hoặc chọn chương trình boot cất tại bộ nhớ Flash trên phiến.

Bộ nhớ Boot Rom còn có các bảng được chuẩn bị sẵn như dạng song sin, cos phục vụ cho các ứng dụng tính toán.

* Bảo mật(Security)

F281x và C281x hỗ trợ các mức độ cao về bảo vệ. Chức năng bảo vệ cho phép sử dụng:

- Một từ khóa 128 bit, mã hóa cứng với 16 nhịp chờ.

Người sử dụng tự nạp từ khóa này vào Flash. Một Module mã bảo mật (code security module, CSM) được dùng để bảo vệ các khối Flash/OTP và L0/L1 SARAM. Khả năng bảo mật cho phép hòng chống đọc trộm hoặc nội dung bộ nhớ qua cổng JTAG, hoặc mã chương trình từ các bộ nhớ ngoài, hoặc tìm cách boot nạp phần mềm lạ với khả năng xuất ra ngoài nội dung các khối nhớ đã được khóa mã

bảo mật. Để truy cập các khối nhớ đó, người sử dụng phải nhấp đúng giá trị khóa 128 bit, trùng đúng với giá trị cất trong Flash.

* Khối mở rộng ngắt ngoại vi (Peripheral Interupt Exansion, PIE)

Khối PIE phục vụ xen kênh nhiều nguồ ngắt khác nhau và qui chúng về thành một tập nhỏ các đầu ngắt. Khối PIE có thể hỗ trợ 96 ngắt ngoại vi. Trên chip F2812

- Ngoại vi sử dụng tất cả 45 trong tổng số 96 ngắt.

- 96 ngắt được phân thành các nhóm, mỗi nhóm là 8 ngắt và được gán cho 1 trong số 12 dây ngắt của CPU (INT1 tới INT12).

- Mỗi ngắt đều có vector ngắt riêng cất trong RAM mà người sử dụng có thể ghi, xóa.

Khi phục vụ ngắt CPU sẽ tự động đọc vector ngắt. Thao tác đọc đó cần 9 nhịp xung đồng hồ của CPU và sẽ cất nội dung của các thanh ghi quan trọng. Nhờ vậy, CPU có thể phản ứng nhanh với các sự kiện ngắt. Việc phân cấp ưu tiên của ngắt sẽ do Hardware và software điều khiển. Từng ngắt riêng rẽ có thể bị cấm, hay cho phép trong khối PIE.

* Các ngắt ngoài (External Interrupts, XINT1, 2, 13, XNMI)

Chip F2812 hỗ trợ 3 ngắt ngoài là loại có thể che chắn (masked) được (XINT1, 2, 13). XINT 13 được liên kết với một ngắt ngoài không che được (XNMI) . Tín hiệu liên kết có tên là XNMI_ XINT13. Mỗi ngắt đều có thể được chọn bởi sườn âm hoặc sườn dương của tín hiệu ngắt và như vậy cũng có thể cấm hoặc cho phép kích hoạt (kể cả XNMI). Mọi ngắt loại che chắn được đều có một bộ đếm tiến 16 bit, có nội dung bị đưa về không khi hát hiện sườn ngắt hợp thức. Bộ đếm đó có thể được sử dụng để đo chính xác thời gian phục vụ ngắt.

*Mạch dao động(Oscillator) và PLL.

F2812 được điều khiển bởi xung nhịp do mạch dao động ngoài hay mạch dao động trong (bằng thạch anh gắn vào chip) cung cấp. Một mạch khóa pha PLL có sẵn hỗ trợ tới 10 hệ số chia tần đầu vào khác nhau. Các hệ số PLL có thể thay đổi

động bằng software, cho phép người sử dụng có thể thay đổi tần số đồng hồ ngay khi đang vận hành, một tính năng quan trọng khi xuất hiện đòi hỏi giảm công suất tổn hao khi đang chạy. Khối PLL có thể được chuyển sang chế độ nghỉ.

* Cơ chế (watchdog)

F281x có một đồng hồ watchdog. Phần mềm ứng dụng sẽ phải reset bộ đếm watchdog trong một khung thời gian đã định, nếu không cơ chế watchdog sẽ tự reset vi xử lí. Có thể cấm cơ chế watchdog khi cần thiết.

* Xung đồng hồ cho ngoại vi (Peripheral Clocking)

Người lập trình có thể cho phép hoặc cấm cấp xung đồng hồ riêng rẽ cho từng ngoại vi, nhằm mục đích tiết kiệm năng lượng khi ngoại vi đó không hoạt động. Thêm vào đó, tần số xung đồng hồ hệ thống đưa tới các cổng tuần tự trừ eCan và các bộ xử lí sự kiện, tới các khối CAP và QEP có thể được giảm so với xung đồng hồ của CPU. Khả năng này cho phép cách li nhịp của ngoại vi với nhịp đồng hồ của CPU khi tần số rất cao.

* Các chế độ tiết kiệm năng lượng(Low- Power modes).

Chip F2812 thuộc loại chip thuần túy “static CMOS” có tất cả ba chế độ tiết kiệm năng lượng:

- IDLE: Đưa CPU về chế độ tiết kiệm năng lượng. Có thể ngừng xung đồng hồ của ngoại vi, chỉ giữ lại đồng hồ của những ngoại vi có nhu cầu hoạt động ở chế độ IDLE. Một tín hiệu ngắt của ngoại vi đang hoạt động sẽ đánh thức đưa vi xử lí ra khỏi trạng thái IDLE.

- STANDBY: Ngừng xung đồng hồ của CPU và ngoại vi. Chế độ này ngừng cả bộ dao động và chức năng PLL. Một sự kiện ngắt ngoài sẽ đánh thức vi xử lí và ngoại vi. Trạng thái hoạt động sẽ quay trở lại ở nhịp tiếp ngay sau khi phát hiện được sự kiện ngắt.

- HALT: Ngừng bộ dao động. Chế độ này ngừng hoạt động toàn chip về trạng thái tiêu thụ ít năng lượng. Chỉ tín hiệu reset hoặc XNMI có thể đánh thức chip ra khỏi chế độ này.

* Peripheral Frames 0, 1, 2 (PFn)

F281x và C281x chia thiết bị ngoại vi thành 3 mảng. Các ngoại vi được tổ chức như sau:

- PF0: + XINTF Thanh ghi lập cấu hình giao diện ngoài.

+ PIE Thanh ghi cho phép kiểm tr ngắt PIE và bảng vecto PIE

+ Flash Thanh gh điều khiển Flash, lập trình, xóa, kiểm tra

+ Timers Thanh gh đồng hồ CPU 0,1,2.

+ CSM thanh gh KeY bảo mật

- PF1: + Ecan Thanh ghi hộp thoại và kiểm tra eCan

- PF2: + SYS Thanh ghi kiểm tra hệ thống

+ GPIO thanh ghi lập cấu hình xen kênh và kiểm tra vào/ raGPIO

+ EV thanh ghi điều khiển của bộ xử lí sự kiện (EVA/ EVB)

+ McBSP thanh gh điều khiển McBSP và TX/ RX

+ SCI Thanh ghi điều khiển giao diện tuần tự truyền thông (SCI) và RX/ TX

+SPI thanh ghi điều khiển giao diện tuần tự của ngoại vi (SPI) và RX/TX

+ ADC thanhn ghi điều khiển bộ ADC 12 bit

* Bộ xen kênh vào/ ra đa năng (General-Purpose Input/Output GPIO Multiplexer)

Hầu hết các tín hiệu ngoại vi đều được xen kênh với tín hiệu I/O đa năng. Điều này cho phép người sử dụng dùng một chân (pin) dưới dạng vào/ ra đa năng khi tín hiệu hay chức năng ngoại vi không được dùng đến. Sau reset, mọi chân GPIO đều được đặt ở dạng đầu vào.

*Các đồng hồ 32 bit của CPU – timer (0, 1, 2)

Các đồng hồ CPU 0,1 và 2 là 3 đồng hồ 32 bit giống hệt nhau với chu kì khả trình và khả năng chọn chia tần số đồng hồ tới 16 bit. Mỗi đồng hồ có một thanh ghi đếm lùi 32bit, phát tín hiệu ngắt khi bộ đếm về tới không. Bộ đếm lùi hoạt động với tần số đồng hồ của CPU chia cho hệ thống đã đặt ra. Khi bộ đếm về tới không, nó

sẽ tự động được nạp mới giá trị chu kì 32 bit. Đồng hồ CPU 2 được dự trữ dành cho các ứng dụng OS (RTOS)/BIOS thời gian thực. Đồng hồ CPU 1 dành cho các chức năng hệ thống TI. Đồng hồ CPU2 được nối với INT14 của CPU. Đồng hồ CPU 1 có thể được nối với INT13 của CPU. Đồng hồ CU 0 được dùng để sử dụng tùy nhu cầu và được nối với khối PIE.

* Ngoại vi điều khiển động cơ (Motor Control Peripherals)

Chip F2812 hỗ trợ các ngoại vi sau đây, đáp ứng nhu cầu thực hiện các bộ điều khiển nhúng và truyền thông.

- EV: Module quản lí sự kiện bao gồm các thanh ghi đa năng, các bộ so sánh /PWM (full- compare/PWM), các đầu vào đo(Capture inputs,CAP) và các mạch thu nhập xung của encoder. Hai bộ quản lí sự kiện đó cho phép điều khiển đồng thời 2 động cơ xoay chiều ba pha hoặc 4 động cơ 2 pha. Các bộ quản lí sự kiện của F2812 tương thích với các bộ cùng chức năng của dòng 240x devices.

- ADC: Khối ADC là một bộ biến đổi 12 bit, 16 kênh. Bộ ADC có hai đơn vị trích mẫu / giữ chậm cho phép trích mẫu một cách liên tục.

* Ngoại vi cổng tuần tự (Serial Port Peripherals)

- Chip F2812 hỗ trợ các ngoại vi có truyền thông tuần tự sau đây:

- eCAN: Đây là phiên bản nâng cao của ngoại vi CAN. Phiên bản này hỗ trợ 32 hộp thoại, giữ bản tin, và tương thích với CAN 2.0B.

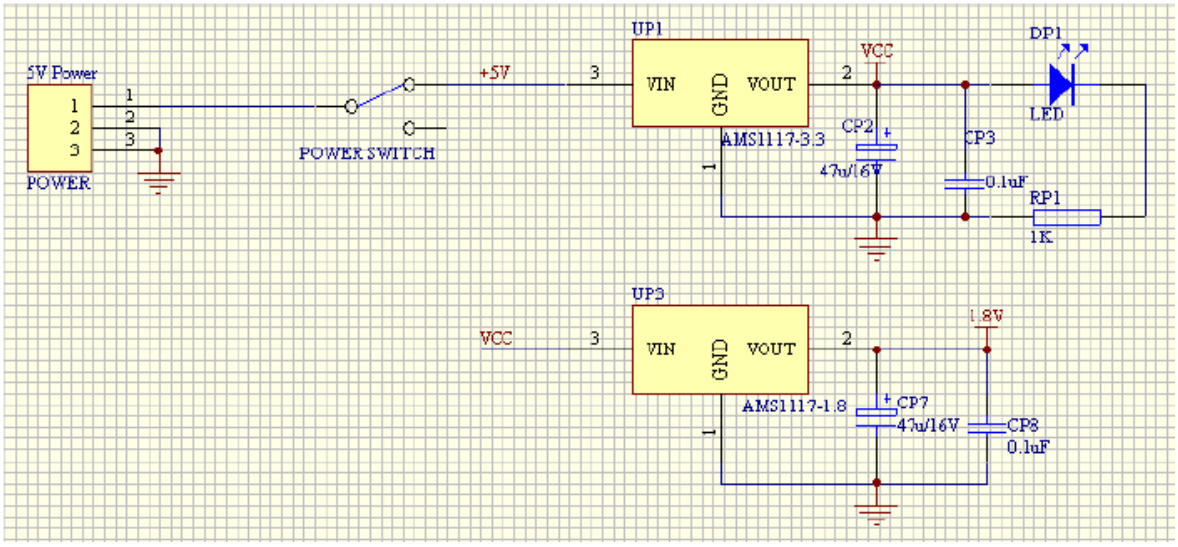
- McBSP: Đâylà cổng tuần tự có đệm đa kênh, được sử dụng để nối với các dây E1/T1, mã hóa với chất lượng phone để dùng trong các ứng dụng modem hoặc thiết bị Audio DAC loại stereo có chất lượng cao. Các thanh ghi thu và phát của McBSP hỗ trợ FIFO 16 mức. Điều này đx giảm đáng kể sự trùng lặp khi phục vụ ngoại vi loại này.

- SPI: Là một cổng I/O tuần tự tốc độ cao, cho phép dịch chuyển một chuỗi bit tuần tự với chiều dài khả trình từ 1 đến 16 bit vào hay ra khỏi chip chỉ trong 1 lần gửi bit do ta lập trình. Thông thường, SPI được sử dụng để truyền thông giữa chip DSP controller và ngoại vi ngoài, hoặc một vi xử lí khác. Các ứng dụng đặc trưng bao gồm I/O ngoài hoặc mở rộng ngoại vi bằng các thanh ghi dịch, các bộ

đệm hiển thị và các bộ ADCs. Việc truyền thông giữa nhiều đơn vị cũng được hỗ trợ nhờ chế độ hoạt động master / slave.

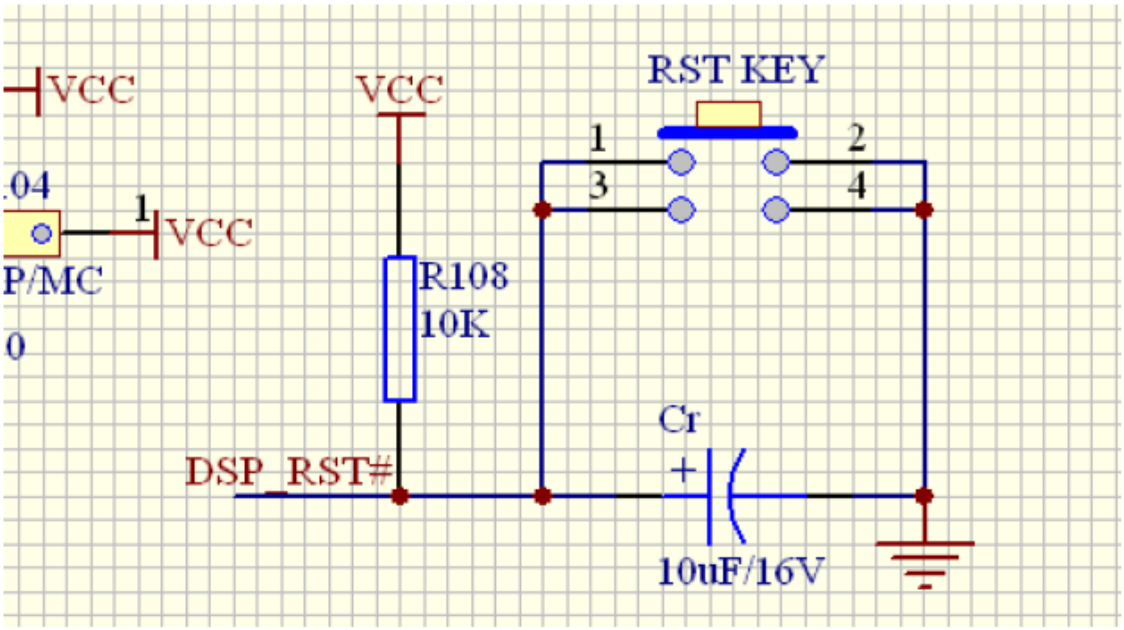
- SCI:Giao diện truyền thông tuần tự này là một cổng tuần tự không đồng bộ hai dây, thường được gọi là UART. Trên chip F2812 cổng này hỗ trợ FIFO thu phát 16 mức.

2.3.1.3. Khối nguồn mạch điều khiển



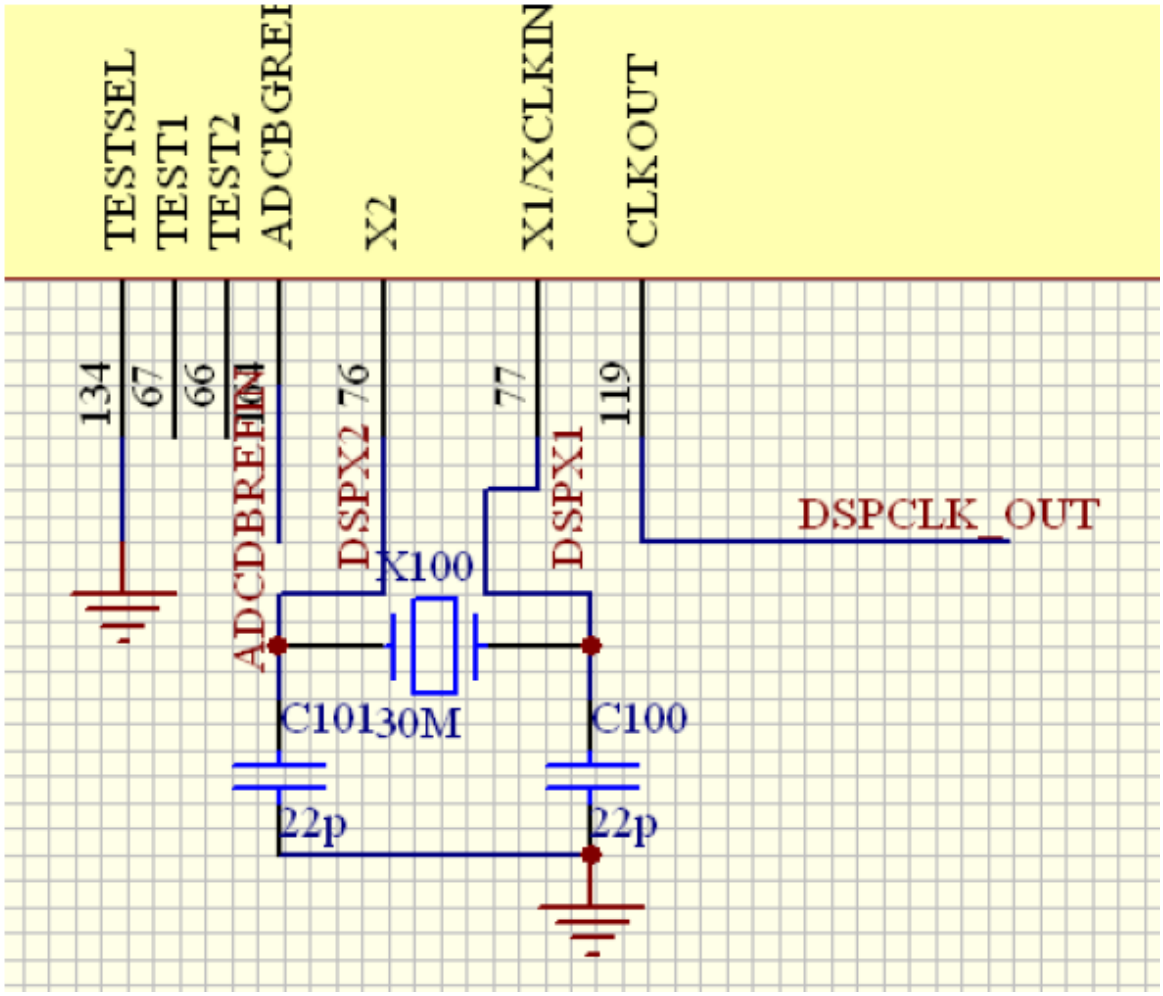
Hình 2.12. Khối nguồn mạch điều khiển

2.3.1.4. Mạch Reset



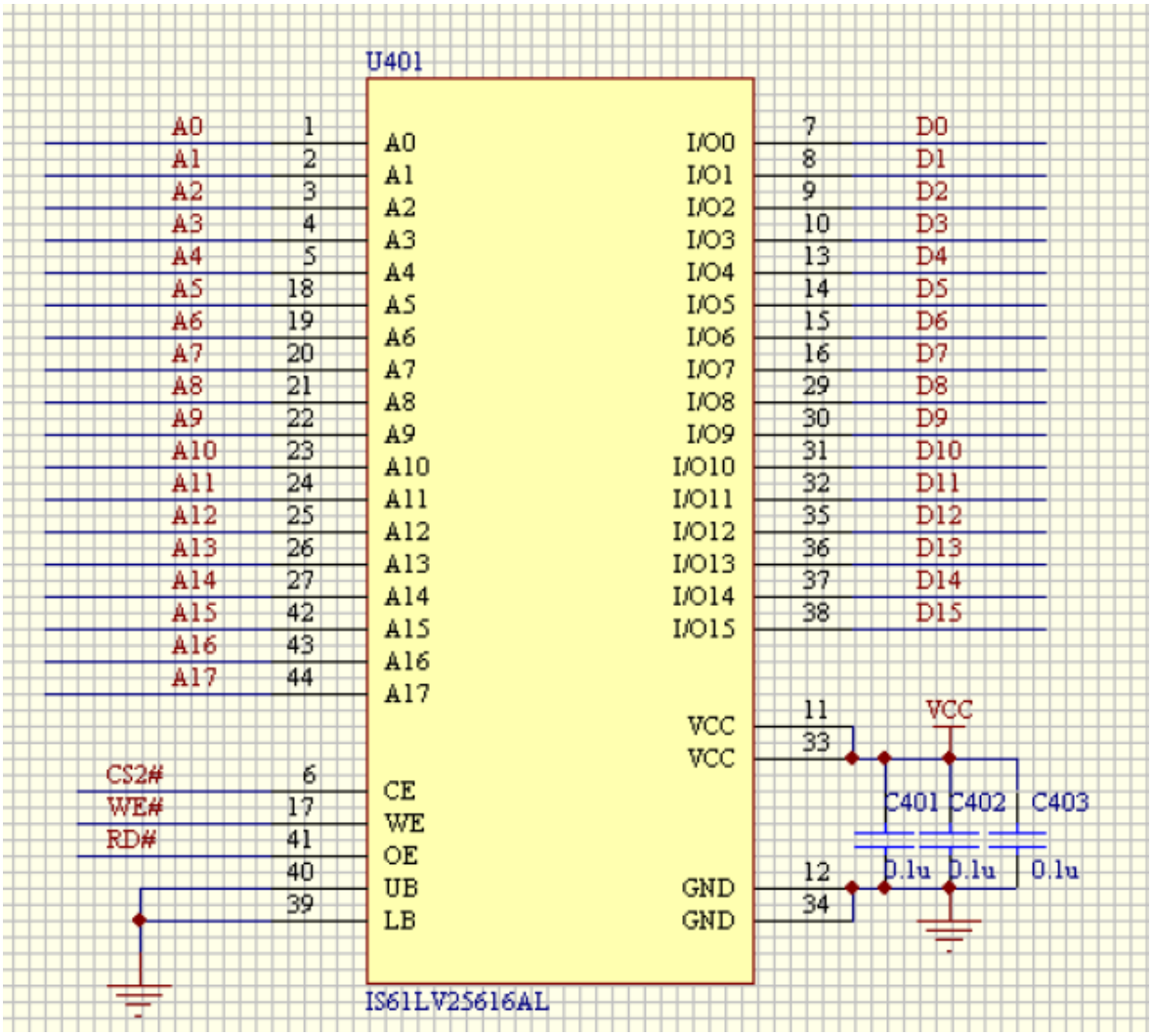
Hình 2.13. Mạch Reset

2.3.1.5. Mạch tạo dao động



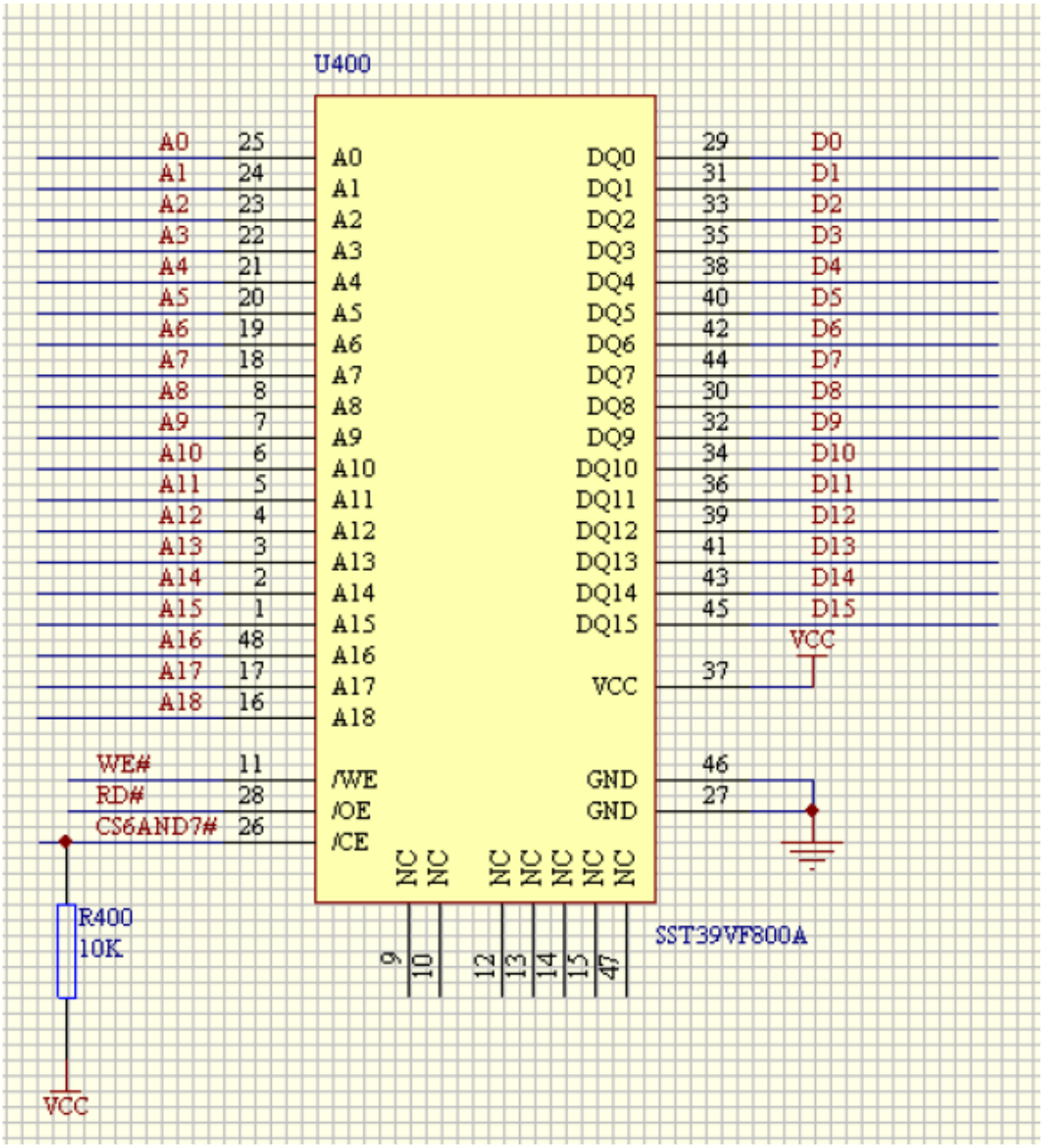
Hình 2.14. Mạch tạo dao động

2.3.1.6. Mạch ghép nối bộ nhớ bộ nhớ RAM IS61LV 25616AL, 256Kx16 bit



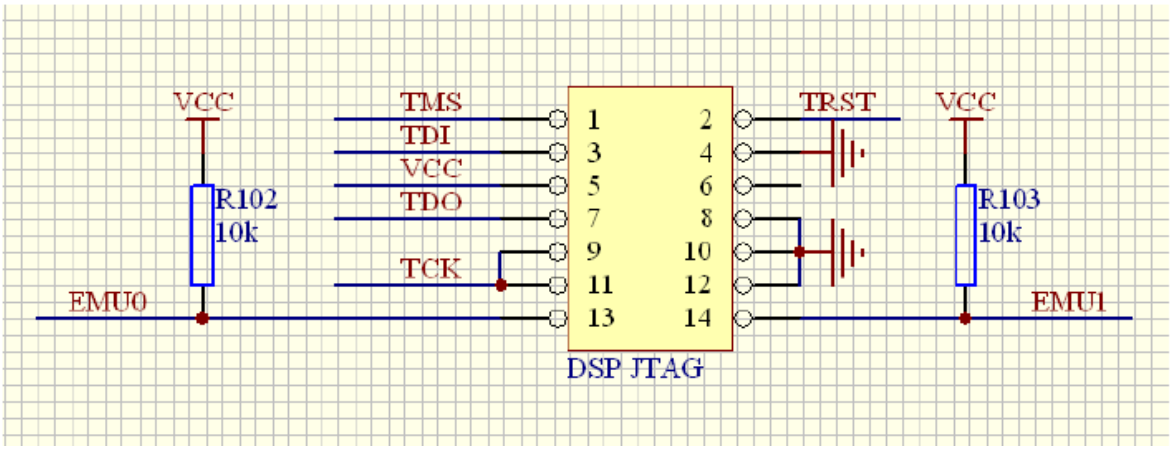
Hình 2.15. Mạch ghép nối bộ nhớ bộ nhớ RAM IS61LV 25616AL, 256Kx16 bit

2.3.1.7. Mạch ghép nối bộ nhớ bộ nhớ FLASH SST39VF800, 512Kx16bit



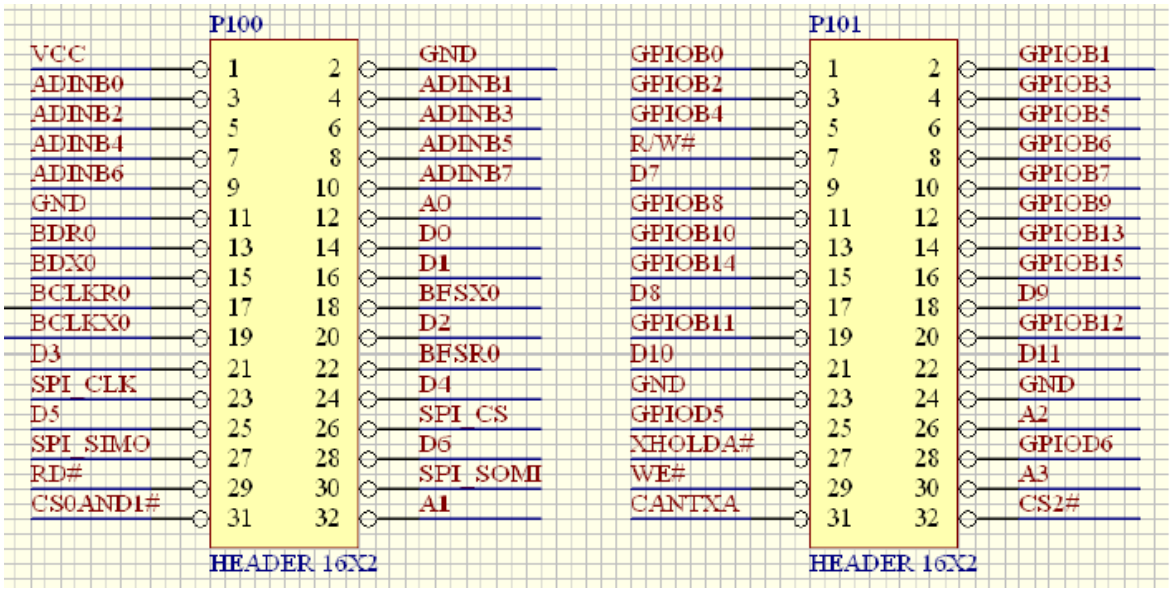
Hình 2.16. Mạch ghép nối bộ nhớ bộ nhớ FLASH SST39VF800, 512Kx16bit

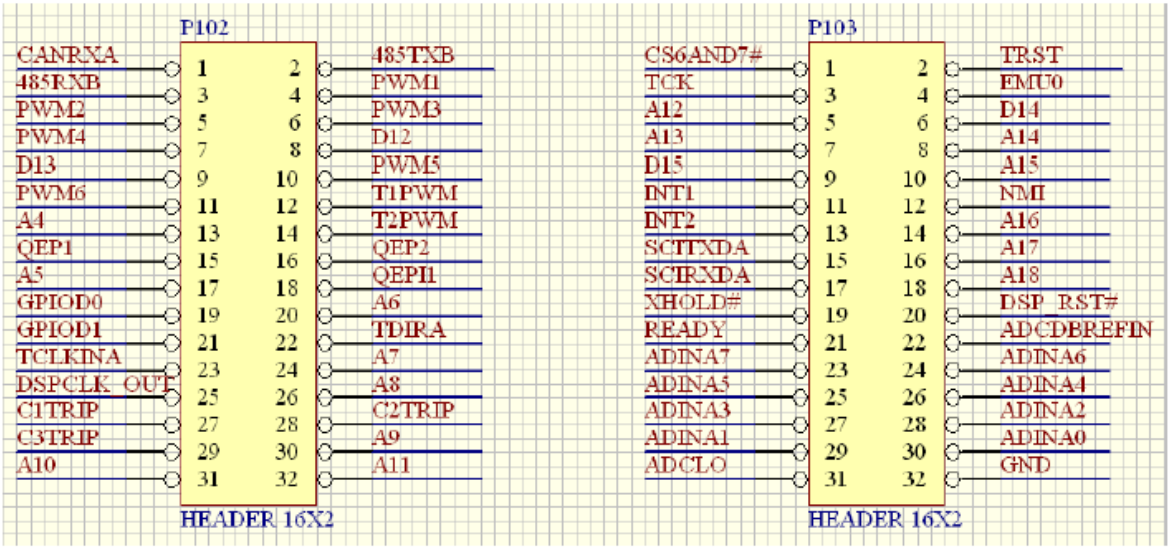
2.3.1.8. Mạch DSP JTAG



Hình 2.17. Mạch DSP JTAG

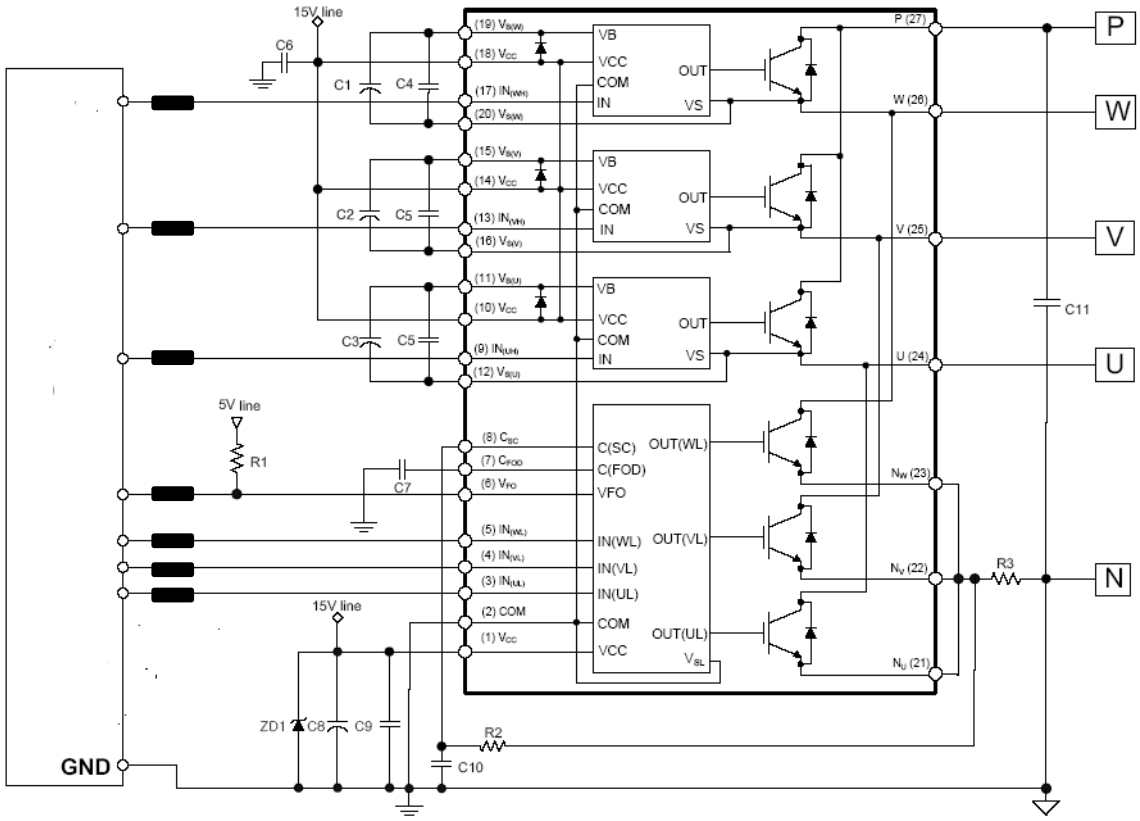
2.3.1.9. Sơ đồ chân của DSP trên Board điều khiển





Hình 2.18. Sơ đồ chân của DSP trên Board điều khiển

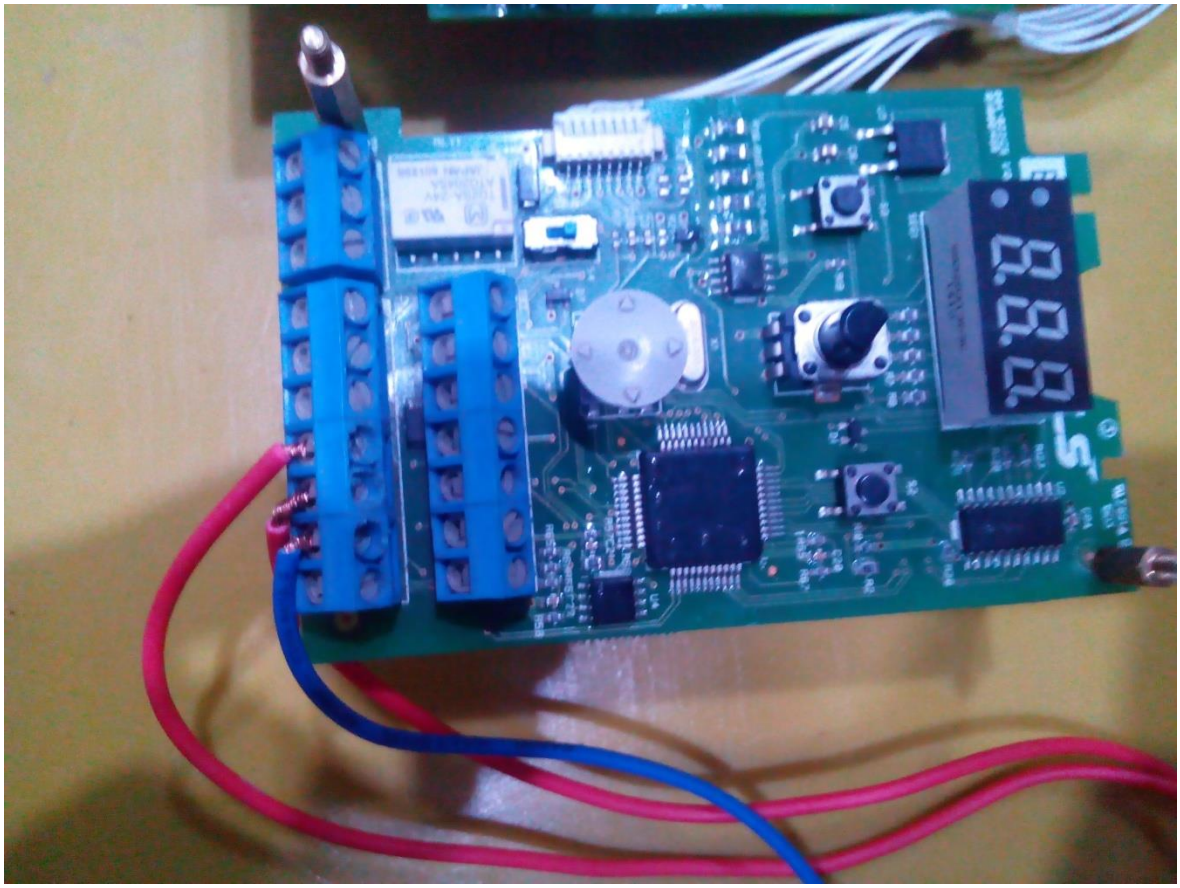
2.3.2. Ghép nối TMS320 F2812 với module công suất thông minh FSBB30CH60C



Hình 2.19. Sơ đồ mạch Ghép nối TMS320 F2812 với module công suất thông minh FSBB30CH60C

2.3.3. Một số hình ảnh về phần cứng

Hình 2.20. Board mạch lực hệ thống



Hình 2.21. Board mạch giao tiếp điều khiển hệ thống

Hình 2.22. Hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng

Hình 2.23. Kết nối hệ thống với PLC và máy tính PC

CHƯƠNG 3. THIẾT KẾ PHẦN MỀM

3.1. Thiết kế bộ điều khiển dòng điện cho động cơ tuyến tính ba pha kiểu đồng bộ

3.1.1. Mô hình toán học động cơ tuyến tính ba pha kích thích vĩnh cửu

Theo [2], ta có hệ phương trình toán học mô tả ĐCTT-KTVC như sau:

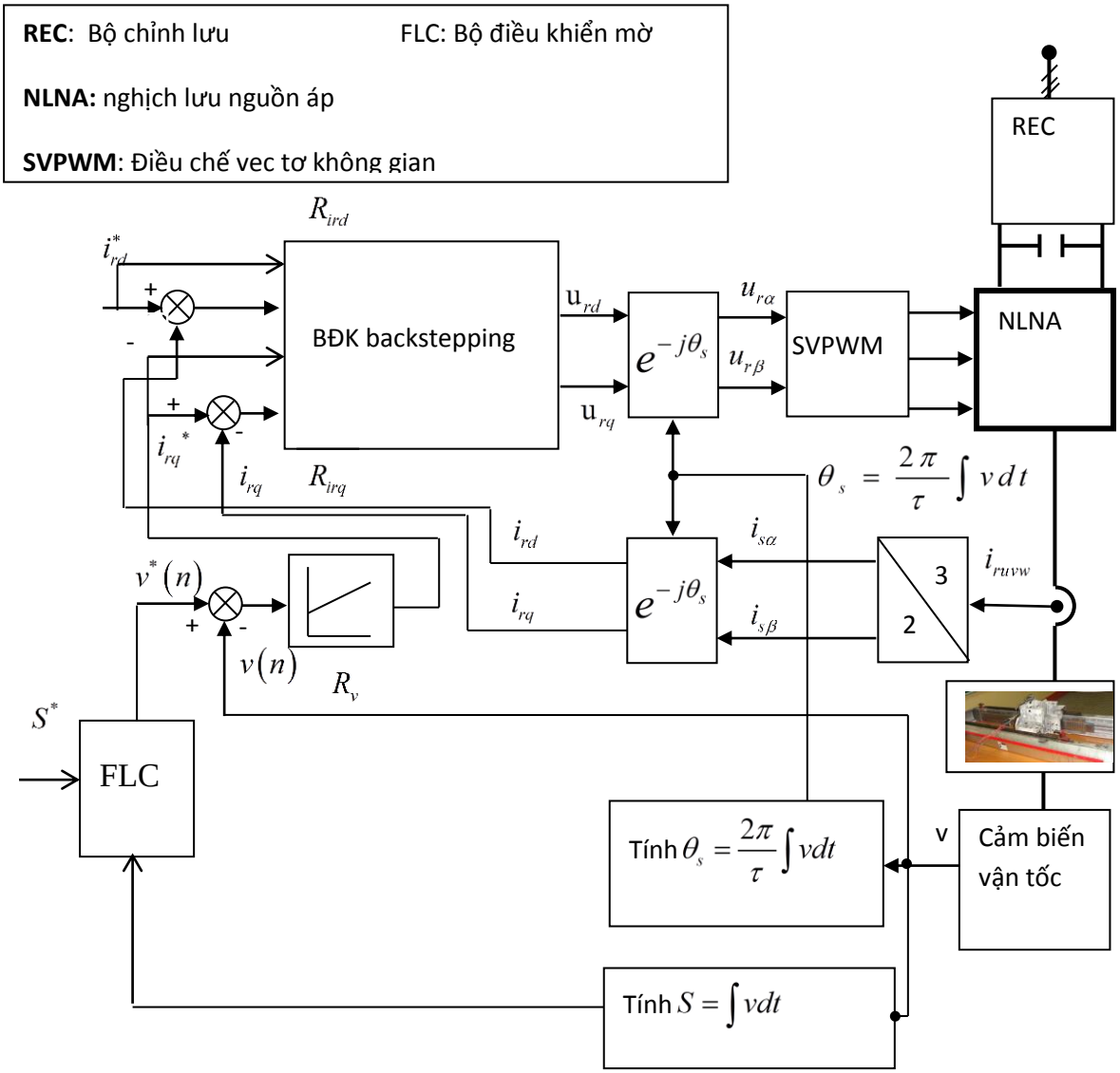
$$\begin{cases} \dfrac{di_{\mathrm{rd}}}{dt} = -\dfrac{1}{T_{rd}} i_{rd} + \left(\dfrac{2\pi}{\tau} v\right)\dfrac{L_{rq}}{L_{rd}} i_{rq} + \dfrac{1}{L_{rd}} u_{\mathrm{rd}} \\ \dfrac{di_{\mathrm{rq}}}{dt} = -\left(\dfrac{2\pi}{\tau} v\right)\dfrac{L_{rd}}{L_{rq}} i_{rd} - \dfrac{1}{T_{rq}} i_{rq} + \dfrac{1}{L_{rq}} u_{\mathrm{rq}} - \left(\dfrac{2\pi}{\tau} v\right)\dfrac{\psi_p}{L_{rq}} \\ \dfrac{dS}{dt} = v \end{cases} \tag{3.1}$$

Theo [2] lực điện từ của ĐCTTKTVC được xác định như (3.2) và phương trình chuyển động của động cơ như (3.3)

$$F = \frac{3\pi}{\tau}\left(\psi_p i_{rq} + \left(L_{rd} - L_{rq}\right) i_{rd} i_{rq}\right) \tag{3.2}$$

$$F - F_c = m\frac{dv}{dt} \tag{3.3}$$

3.1.2 Thiết kế bộ điều khiển dòng điện theo phương pháp Backsteping cho động cơ tuyến tính.



Hình 3.1. Cấu trúc điều khiển ĐCTT loại ĐB - KTVC 3 pha sử dụng phương pháp Backstepping

Để nâng cao được chất lượng điều khiển của hệ thống, cần tập trung nâng cao chất lượng của bộ điều chỉnh dòng cho động cơ, bằng cách áp dụng phương pháp thiết kế phi tuyến Backstepping để tổng hợp bộ điều khiển dòng. Sơ đồ cấu trúc điều khiển động cơ trong đó khâu điều chỉnh dòng được thiết kế theo phương pháp phi tuyến Backstepping. Nội dung của phần tổng hợp bộ điều chỉnh dòng phía động cơ bao gồm các công việc chính sau:

- Tổng hợp khâu điều chỉnh dòng Backstepping cơ bản phía động cơ: bao gồm tổng hợp các bộ điều chỉnh thành phần i_{rd} và i_{rq} trên miền liên tục, sau đó số hoá bộ

điều chỉnh dòng để có thể thực hiện việc cài đặt bộ điều chỉnh trên hệ thống xử lý tín hiệu số DSP. Qua bộ điều chỉnh dòng cơ bản, ta sẽ thấy được bộ điều chỉnh đã thực hiện được việc tách kênh thông qua bù các thành phần liên kết ngang $\frac{2\pi v}{\tau} i_{rd}$ và $\frac{2\pi v}{\tau} i_{rq}$, việc bù trực tiếp điện áp nguồn (điện áp đặt vào phần chuyển động), bù từ thông phần chuyển động, bù vận tốc chuyển động của động cơ. Tuy nhiên, do bộ điều chỉnh dòng chưa có thành phần tích phân, nên sẽ tồn tại sai lệch tĩnh.

- Khắc phục sai lệch tĩnh. Do bộ điều chỉnh dòng Backstepping cơ bản chưa có thành phần tích phân, nên để khử sai lệch tĩnh, ta đưa thành phần tích phân vào trên cơ sở kỹ thuật Backstepping.

3.1.2.1. Tổng hợp bộ điều chỉnh thành phần i_{rd} trên miền liên tục.

Chọn i_{rd} là biến điều khiển, giá trị mong muốn của nó i_{rd}^* được lấy từ bộ điều chỉnh mô men thông qua khâu tính toán giá trị đặt. Gọi sai lệch giữa i_{rd} và giá trị đặt i_{rd}^* là :

$$z_1 = i_{rd} - i_{rd}^* \tag{3.4}$$

Chọn hàm điều khiển Lyapunov là: $\nu_1 = \frac{1}{2} z_1^2$. Lấy đạo hàm theo thời gian, ta có: $\dot{\nu}_1 = z_1 \dot{z}_1$. Ta lại có: $\dot{z}_1 = \frac{di_{rd}}{dt} - \frac{di_{rd}^*}{dt}$.

Từ (3.1), ta có :

$$\frac{di_{rd}}{dt} = -\frac{1}{T_{rd}} i_{rd} + \left(\frac{2\pi}{\tau} v\right) \frac{L_{rq}}{L_{rd}} i_{rq} + \frac{1}{L_{rd}} u_{rd}$$

Do đó:

$$\dot{z}_1 = -\frac{1}{T_{rd}} i_{rd} + \left(\frac{2\pi}{\tau} v\right) \frac{L_{rq}}{L_{rd}} i_{rq} + \frac{1}{L_{rd}} u_{rd} - \frac{di_{rd}^*}{dt} \tag{3.5}$$

Chọn biến điều khiển là $\frac{1}{L_{rd}} u_{rd}$, để $\dot{\nu}_1 = z_1 \dot{z}_1 < 0$, thì giá trị của biến điều khiển là:

$$\frac{1}{L_{rd}}u_{rd} = \frac{1}{T_{rd}}i_{rd} - \left(\frac{2\pi}{\tau}v\right)\frac{L_{rq}}{L_{rd}}i_{rq} + \frac{di_{rd}^{*}}{dt} - k_1 z_1 \qquad (3.6)$$

Với k_1 là hằng số dương.

3.1.2.2. Tổng hợp bộ điều chỉnh thành phần i_{rq} trên miền liên tục

Chọn i_{rq} là biến điều khiển, giá trị mong muốn của nó i_{rq}^{*} được lấy từ bộ điều chỉnh công suất thông qua khâu tính toán giá trị đặt TSP. Gọi sai lệch giữa i_{rq} và giá trị đặt i_{rq}^{*} là :

$$z_2 = i_{rq} - i_{rq}^{*} \qquad (3.7)$$

Chọn hàm điều khiển Lyapunov là : $\nu_2 = \frac{1}{2}z_2^2$. Lấy đạo hàm theo thời gian, ta có: $\dot{\nu}_2 = z_2\dot{z}_2$. Ta lại có:

$$\dot{z}_2 = \frac{di_{rq}}{dt} - \frac{di_{rq}^{*}}{dt} \qquad (3.8)$$

Từ (3.1), ta có :

$$\frac{di_{rq}}{dt} = -\left(\frac{2\pi}{\tau}v\right)\frac{L_{rd}}{L_{rq}}i_{rd} - \frac{1}{T_{rq}}i_{rq} + \frac{1}{L_{rq}}u_{rq} - \left(\frac{2\pi}{\tau}v\right)\frac{\psi_p}{L_{rq}} \qquad (3.9)$$

Do đó:

$$\dot{z}_2 = -\left(\frac{2\pi}{\tau}v\right)\frac{L_{rd}}{L_{rq}}i_{rd} - \frac{1}{T_{rq}}i_{rq} + \frac{1}{L_{rq}}u_{rq} - \left(\frac{2\pi}{\tau}v\right)\frac{\psi_p}{L_{rq}} - \frac{di_{rq}^{*}}{dt} \qquad (3.10)$$

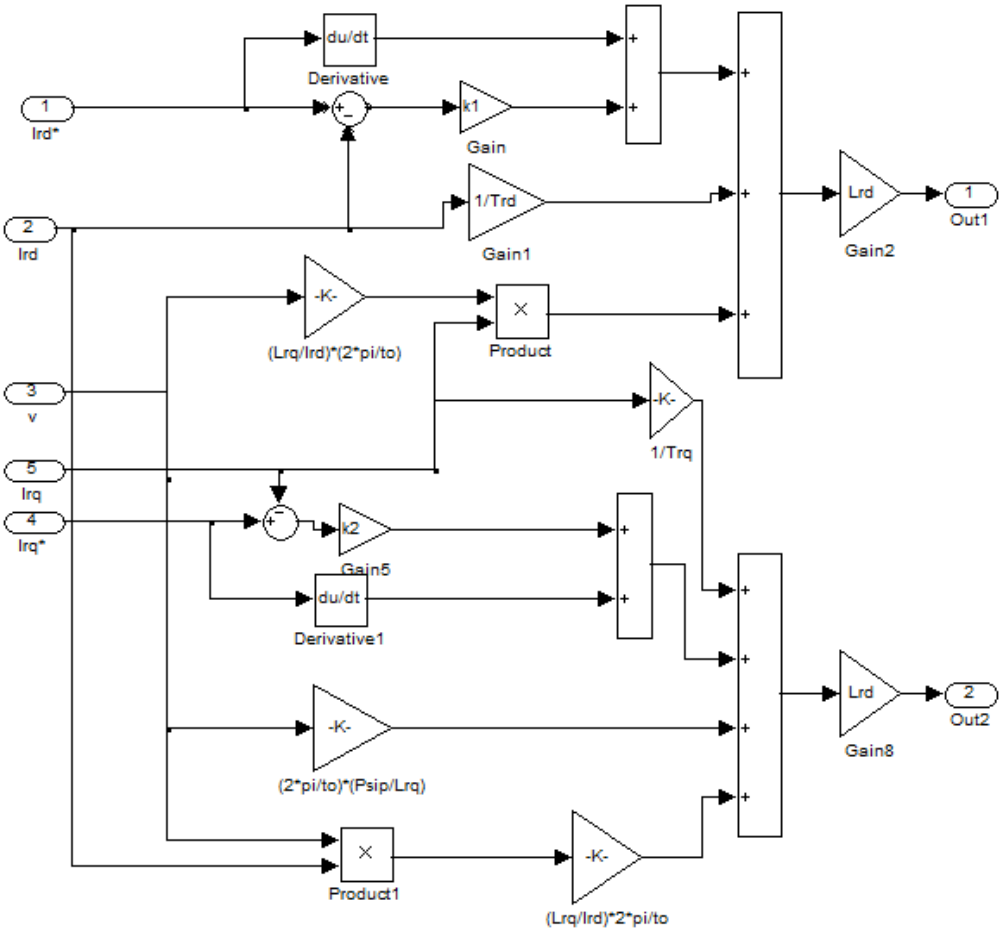
Chọn biến điều khiển là $\frac{1}{L_{rq}}u_{rq}$, để $\dot{\nu}_2 = z_2\dot{z}_2 < 0$, thì giá trị của biến điều khiển là:

$$\frac{1}{L_{rq}}u_{rq} = \left(\frac{2\pi}{\tau}v\right)\frac{L_{rd}}{L_{rq}}i_{rd} + \frac{1}{T_{rq}}i_{rq} + \left(\frac{2\pi}{\tau}v\right)\frac{\psi_p}{L_{rq}} + \frac{di_{rq}^{*}}{dt} - k_2 z_2 \qquad (3.11)$$

Với k_2 là hằng số dương.

Bộ điều khiển thành phần i_{rq} có thể được viết lại ở dạng ngắn gọn hơn như sau:

Từ các bộ điều chỉnh dòng backstepping cơ bản (3.6) và (3.11), ta có sơ đồ khối bộ điều chỉnh như hình 3.16.



Hình 3.2. Sơ đồ bộ điều chỉnh dòng Backstepping

Nhận xét: Bộ điều chỉnh dòng Backstepping đã thực hiện được các vấn đề sau:

- Thực hiện tách kênh thông qua bù thành phần liên kết ngang $\frac{2\pi}{\tau} v i_{rd}$ và $\frac{2\pi}{\tau} v i_{rq}$.

- Bù từ thông.

- Bù vận tốc phần chuyển động v.

Tuy nhiên, do bộ điều chỉnh không có thành phần tích phân, nên sẽ gây sai lệch tĩnh.

3.1.2.3. Tính ổn định của hệ có bộ điều chỉnh dòng Backstepping

Với các khâu điều chỉnh (3.6) và (3.11), thay chúng vào các phương trình (3.5) và (3.10), ta được các phương trình mô tả mô hình dòng của ĐCTT-KTVC trên không gian các biến trạng thái mới z_1 và z_2 như sau :

$$\begin{cases} \dot{z}_1 = -k_1 z_1 \\ \dot{z}_2 = -k_2 z_2 \end{cases} \tag{3.12}$$

Có thể viết hệ ở dạng sau:

$$\frac{d}{dt}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -k_1 & 0 \\ 0 & -k_2 \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \tag{3.13}$$

Hệ có điểm cân bằng : $(z_1,z_2)^T = (0,0)^T$

Chọn hàm điều khiển Lyapunov : $v = \frac{1}{2}z_1^2 + \frac{1}{2}z_2^2$. Lấy đạo hàm của v, ta có:

$\dot{v} = z_1\dot{z}_1 + z_2\dot{z}_2 = -k_1 z_1^2 - k_2 z_2^2 \le 0$, ta kết luận, hệ ổn định tại điểm cân bằng $(z_1,z_2)^T=(0,0)^T$. Bộ điều khiển đã thiết kế đảm bảo yêu cầu ổn định toàn cục và $i_{rd} \to i_{rd}^*, i_{rq} \to i_{rq}^*$.

3.1.2.4. Số hoá bộ điều chỉnh dòng Backstepping cơ bản.

a) Số hoá bộ điều chỉnh dòng i_{rd}.

Vì bộ điều khiển được thực hiện trên hệ thống DSP, do đó ta cần gián đoạn bộ điều chỉnh (3.6), thực hiện sai phân (3.6), ta có:

$$\frac{1}{L_{rd}}u_{rd}(k) = \frac{1}{T_{rd}}i_{rd}(k) - \left(\frac{2\pi}{\tau}v(k)\right)\frac{L_{rq}}{L_{rd}}i_{rq}(k) + \left.\frac{di_{rd}^*}{dt}\right|_{t=kT} - k_1\left[i_{rd}(k) - i_{rd}^*(k)\right] \tag{3.14}$$

Thực hiện xấp xỉ bậc hai đạo hàm $\left.\frac{di_{rd}^*}{dt}\right|_{t=kT}$, ta được:

$$\left.\frac{di_{rd}^*}{dt}\right|_{t=kT} = \frac{1}{2T}[3i_{rd}^*(k) - 4i_{rd}^*(k-1) + i_{rd}^*(k-2)]$$

Ta được:

$$\frac{1}{L_{rd}}u_{rd}(k)=\frac{1}{T_{rd}}i_{rd}(k)-\left(\frac{2\pi}{\tau}v(k)\right)\frac{L_{rq}}{L_{rd}}i_{rq}(k)+$$
$$+\frac{1}{2T}[3i^{*}_{rd}(k)-4i^{*}_{rd}(k-1)+i^{*}_{rd}(k-2)]-k_1\left[i_{rd}(k)-i^{*}_{rd}(k)\right] \quad (3.15)$$

b) Số hoá bộ điều chỉnh dòng i_{rq}.

Vì bộ điều chỉnh được thực hiện trên hệ thống DSP, do đó ta cần gián đoạn bộ điều chỉnh (3.11), thực hiện sai phân (3.11), ta có:

$$\frac{1}{L_{rq}}u_{rq}(k)=\left(\frac{2\pi}{\tau}v(k)\right)\frac{L_{rd}}{L_{rq}}i_{rd}(k)+\frac{1}{T_{rq}}i_{rq}(k)+\left(\frac{2\pi}{\tau}v(k)\right)\frac{\psi_p(k)}{L_{rq}}+$$
$$+\left.\frac{di^{*}_{rq}}{dt}\right|_{t=kT}-k_2\left[i_{rq}(k)-i^{*}_{rq}(k)\right] \quad (3.16)$$

Thực hiện xấp xỉ bậc hai đạo hàm $\left.\frac{di^{*}_{rq}}{dt}\right|_{t=kT}$, ta được:

$$\left.\frac{di^{*}_{rq}}{dt}\right|_{t=kT}=\frac{1}{2T}[3i^{*}_{rq}(k)-4i^{*}_{rq}(k-1)+i^{*}_{rq}(k-2)] \quad (3.17)$$

Ta được:

$$\frac{1}{L_{rq}}u_{rq}(k)=\left(\frac{2\pi}{\tau}v(k)\right)\frac{L_{rd}}{L_{rq}}i_{rd}(k)+\frac{1}{T_{rq}}i_{rq}(k)+\left(\frac{2\pi}{\tau}v(k)\right)\frac{\psi_p(k)}{L_{rq}}+$$
$$+\frac{1}{2T}[3i^{*}_{rq}(k)-4i^{*}_{rq}(k-1)+i^{*}_{rq}(k-2)]-k_2\left[i_{rq}(k)-i^{*}_{rq}(k)\right] \quad (3.18)$$

3.1.2.5. Khắc phục sai lệch tĩnh

Với bộ điều chỉnh dòng Backstepping cơ bản, do trong bộ điều chỉnh chưa có thành phần tích phân nên sẽ có sai lệch tĩnh, ngoài ra nguyên nhân dẫn đến có sai lệch tĩnh còn do ảnh hưởng của tần số lấy mẫu, các tham số của động cơ trong quá trình làm việc. Để khắc phục sai lệch tĩnh, mục đích đặt ra là phải đưa được thành phần tích phân vào thuật toán backstepping cơ bản. Thành phần tích phân được thêm vào mô hình đối tượng, và sau đó trượt về phía phương trình bộ điều khiển. Để thực hiện được mục đích trên, ta xem xét lại bản chất của bộ điều chỉnh dòng

backstepping cơ bản trên quan điểm tuyến tính hóa và tách kênh trực tiếp của phương pháp backstepping, trên cơ sở đó, đưa ra một cách áp dụng mới để tổng hợp bộ điều chỉnh dòng Backstepping cho động cơ tuyến tính kiểu đồng bộ kích thích vĩnh cửu.

3.1.2.6. Đưa thành phần tích phân vào thuật toán backstepping cơ bản để khử sai lệch tĩnh

Theo tài liệu [7], để đưa thành phần tích phân vào thuật toán backstepping cơ bản nhằm khử sai lệch tĩnh, thực chất là thực hiện biến đổi hệ trục tọa độ trạng thái, đưa mô hình về dạng tuyến tính và tách kênh, khi đó, mô hình dòng của ĐCTT-KTVC sẽ có dạng như sau :

$$\begin{cases} \dfrac{di_{rd}}{dt} = w_1 \\ \dfrac{di_{rq}}{dt} = w_2 \end{cases} \qquad (3.19a,b)$$

Trong đó :

$$\begin{cases} w_1 = -\dfrac{1}{T_{rd}} i_{rd} + \left(\dfrac{2\pi}{\tau} v\right)\dfrac{L_{rq}}{L_{rd}} i_{rq} + \dfrac{1}{L_{rd}} u_{rd} \\ w_2 = -\left(\dfrac{2\pi}{\tau} v\right)\dfrac{L_{rd}}{L_{rq}} i_{rd} - \dfrac{1}{T_{rq}} i_{rq} + \dfrac{1}{L_{rq}} u_{rq} - \left(\dfrac{2\pi}{\tau} v\right)\dfrac{\psi_p}{L_{rq}} \end{cases} \qquad (3.20a,b)$$

Mô hình sau khi thực hiện biến đổi hệ trục toạ độ trạng thái (3.19a,b) sẽ tạo điều kiện cho việc đưa thành phần tích phân vào thuật toán backstepping cơ bản để khử sai lệch tĩnh. Thành phần tích phân được thêm vào mô hình đối tượng, và sau đó trượt về phía phương trình bộ điều khiển.

a) Tổng hợp bộ điều khiển thành phần i_{rd}

Xét phương trình (3.19a), chuyển sang toán tử Laplace ta có :

$$sI_{rd}(s) = W_1(s) \qquad (3.21)$$

Vậy hàm truyền đạt sẽ là :

$$G_1(s) = \frac{Y_1(s)}{W_1(s)} = \frac{X_{11}(s)}{W_1(s)} = \frac{I_{rd}(s)}{W_1(s)} = \frac{1}{s} \qquad (3.22)$$

Hàm truyền đạt sau khi thêm thành phần tích phân vào đối tượng sẽ là:

$$G_1^*(s) = \frac{Y_1(s)}{W_1(s)s} = \frac{1}{s^2} = \frac{Y_1(s)}{U_1(s)} \qquad (3.23)$$

Phương trình (3.23) được viết dưới dạng các phương trình trạng thái như sau:

$$\begin{cases} y_1 = x_{11} \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = u_1 \end{cases} \qquad (3.24a,b,c)$$

trong đó: $$U_1(s) = W_1(s).s \qquad (3.25)$$

- Bước1: Gọi $y_1^* = i_{rd}^*$ - là giá trị đầu ra mong muốn. Biến sai lệch thứ nhất được định nghĩa là:

$$\varepsilon_{11} = y_1 - y_1^* = x_{11} - y_1^* = i_{rd} - i_{rd}^* \qquad (3.26)$$

Hàm điều khiển Lyapunov thứ nhất được chọn là:

$$v_{11} = \frac{1}{2}\varepsilon_{11}^2 \qquad (3.27)$$

Đạo hàm (3.27), ta có:

$$\dot{v}_{11} = \varepsilon_{11}\dot{\varepsilon}_{11} = \varepsilon_{11}\left(\dot{x}_{11} - \dot{y}_1^*\right) = \varepsilon_{11}\left(x_{12} - \dot{y}_1^*\right) \qquad (3.28)$$

Để $\dot{v}_{11} < 0$, x_{12} được chọn là biến điều khiển ảo, giá trị mong muốn của nó là:

$$\alpha_{11} = \left(x_{12}\right)_d = -k_{11}\varepsilon_{11} + \dot{y}_1^* \qquad (3.29)$$

Trong đó, k_{11} là một hằng số dương. Với cách chọn như trên thì:

$$\dot{v}_{11} = -k_{11}\varepsilon_{11}^2 < 0.$$

- Bước 2: Vì x_{12} là biến điều khiển ảo, chưa phải là biến điều khiển thực. Sai lệch giữa giá trị thực và mong muốn của nó là:

$$\varepsilon_{12} = x_{12} - \alpha_{11} = x_{12} + k_{11}\varepsilon_{11} - \dot{y}_1^* \quad (3.30)$$

Chọn hàm điều khiển Lyapunov là:

$$\nu_{12} = \frac{1}{2}\varepsilon_{11}^2 + \frac{1}{2}\varepsilon_{12}^2 \quad (3.31)$$

Vì:

$$\dot{\varepsilon}_{11} = x_{12} - \dot{y}_1^* = \varepsilon_{12} - k_{11}\varepsilon_{11} \quad (3.32)$$

Nên đạo hàm $\dot{\nu}_{12}$ sẽ là:

$$\begin{aligned}\dot{\nu}_{12} &= \varepsilon_{11}\dot{\varepsilon}_{11} + \varepsilon_{12}\dot{\varepsilon}_{12} \\ &= \varepsilon_{11}(\varepsilon_{12} - k_{11}\varepsilon_{11}) + \varepsilon_{12}(\dot{x}_{12} + k_{11}\dot{\varepsilon}_{11} - \ddot{y}_1^*) \\ &= -k_{11}\varepsilon_{11}^2 + \varepsilon_{12}\left[\left(1 - k_{11}^2\right)\varepsilon_{11} + k_{11}\varepsilon_{12} + u_1 - \ddot{y}_1^*\right]\end{aligned} \quad (3.33)$$

Ta thấy, trong (3.33) xuất hiện biến điều khiển thực u_1. Để $\dot{\nu}_{12} < 0$, ta chọn u_1 như sau :

$$u_1 = \left(k_{11}^2 - 1\right)\varepsilon_{11} - \left(k_{11} + k_{12}\right)\varepsilon_{12} + \ddot{y}_1^* \quad (3.34)$$

Trong đó, k_{12} là hằng số dương.

Thay biểu thức của ε_{11} và ε_{12} vào (3.34), và chuyển sang toán tử Laplace ta được:

$$U_1(s) = \left(1 + k_{11} * k_{12}\right)\left[Y_1^*(s) - X_{11}(s)\right] + s.\left(k_{11} + k_{12}\right)\left[Y_1^*(s) - X_{11}(s)\right] + s^2 Y_1^*(s) \quad (3.35)$$

Bây giờ, ta sẽ trượt thành phần tích phân về phía phương trình bộ điều khiển, từ công thức (3.25), ta có :

$$W_1(s) = \frac{U_1(s)}{s} \quad (3.35)$$

Vậy phương trình bộ điều khiển thành phần i_{rd} sẽ là :

$$W_1(s) = \frac{\left(1 + k_{11}k_{12}\right)}{s}\left[Y_1^*(s) - X_{11}(s)\right] + \left(k_{11} + k_{12}\right)\left[Y_1^*(s) - X_{11}(s)\right] + sY_1^*(s) \quad (3.36)$$

Hay viết gọn hơn ta có:

$$W_1(s) = -\frac{k_{I1}}{s} z_1 - k_{P1} z_1 + s i_{rd}^*(s) \quad (3.37)$$

Hay:

$$W_1 = -k_{I1} \int z_1 dt - k_{P1} z_1 + \frac{di_{rd}^*}{dt} \quad (3.38)$$

Trong đó : $k_{I1} = 1 + k_{11} * k_{12}$; $k_{P1} = k_{11} + k_{12}$

Chuyển sang miền toán tử z, ta có:

$$w_1(z) = \frac{(1 + k_{11}k_{12})Tz^{-1}}{1 - z^{-1}} \left[y_1^*(z) - x_{11}(z) \right] + (k_{11} + k_{12}) \left[y_1^*(z) - x_{11}(z) \right] + \frac{1 - z^{-1}}{T * z^{-1}} y_1^*(z) \quad (3.39)$$

b) Tổng hợp bộ điều khiển thành phần i_{rq}

Xét phương trình (3.19b), chuyển sang toán tử Laplace ta có:

$$sI_{rq}(s) = W_2(s) \quad (3.39)$$

Vậy hàm truyền đạt sẽ là:

$$G_2(s) = \frac{Y_2(s)}{W_2(s)} = \frac{X_2(s)}{W_2(s)} = \frac{I_{rq}(s)}{W_2(s)} = \frac{1}{s} \quad (3.40)$$

Hàm truyền đạt sau khi thêm thành phần tích phân vào đối tượng sẽ là:

$$G_2^*(s) = \frac{Y_2(s)}{W_2(s)s} = \frac{1}{s^2} = \frac{Y_2(s)}{U_2(s)} \quad (3.41)$$

Phương trình (3.41) được viết dưới dạng các phương trình trạng thái như sau:

$$\begin{cases} y_2 = x_{21} \\ \dot{x}_{21} = x_{22} \\ \dot{x}_{22} = u_2 \end{cases} \quad (3.42a,b,c)$$

trong đó: $U_2(s) = W_2(s)s$ (3.43)

- Bước1: Gọi $y_2^* = i_{rq}^*$ - là giá trị đầu ra mong muốn. Biến sai lệch thứ nhất được định nghĩa là:

$$\varepsilon_{21} = y_2 - y_2^* = x_{21} - y_2^* = i_{rq} - i_{rq}^* \tag{3.44}$$

Hàm điều khiển Lyapunov thứ nhất được chọn là:

$$\nu_{21} = \frac{1}{2}\varepsilon_{21}^2 \tag{3.45}$$

Đạo hàm (3.45), ta có:

$$\dot{\nu}_{21} = \varepsilon_{21}\dot{\varepsilon}_{21} = \varepsilon_{21}\left(\dot{x}_{21} - \dot{y}_2^*\right) = \varepsilon_{21}\left(x_{22} - \dot{y}_2^*\right) \tag{3.46}$$

Để $\dot{\nu}_{21} < 0$, x_{22} được chọn là biến điều khiển ảo, giá trị mong muốn của nó là :

$$\alpha_{21} = \left(x_{22}\right)_d = -k_{21}\varepsilon_{21} + \dot{y}_2^* \tag{3.47}$$

Trong đó, k_{21} là một hằng số dương. Với cách chọn như trên thì $\dot{\nu}_{21} = -k_{21}\varepsilon_{21}^2 < 0$.

- Bước 2 : Vì x_{22} là biến điều khiển ảo, chưa phải là biến điều khiển thực. Sai lệch giữa giá trị thực và mong muốn của nó là:

$$\varepsilon_{22} = x_{22} - \alpha_{21} = x_{22} + k_{21}\varepsilon_{21} - \dot{y}_2^* \tag{3.48}$$

Chọn hàm điều khiển Lyapunov là:

$$\nu_{22} = \frac{1}{2}\varepsilon_{21}^2 + \frac{1}{2}\varepsilon_{22}^2 \tag{3.49}$$

Vì:

$$\dot{\varepsilon}_{21} = x_{22} - \dot{y}_2^* = \varepsilon_{22} - k_{21}\varepsilon_{21} \tag{3.50}$$

Nên đạo hàm $\dot{\nu}_{22}$ sẽ là:

$$\begin{aligned}
\dot{\nu}_{22} &= \varepsilon_{21}\dot{\varepsilon}_{21} + \varepsilon_{22}\dot{\varepsilon}_{22} \\
&= \varepsilon_{21}(\varepsilon_{22} - k_{21}\varepsilon_{21}) + \varepsilon_{22}(\dot{x}_{22} + k_{21}\dot{\varepsilon}_{21} - \ddot{y}_2^*) \\
&= -k_{21}\varepsilon_{21}^2 + \varepsilon_{22}\left[\left(1 - k_{21}^2\right)\varepsilon_{21} + k_{21}\varepsilon_{22} + u_2 - \ddot{y}_2^*\right]
\end{aligned} \tag{3.51}$$

Ta thấy, trong (3.51) xuất hiện biến điều khiển thực u_2. Để $\dot{\varepsilon}_{22} < 0$, ta chọn u_2 như sau:

$$u_2 = \left(k_{21}^2 - 1\right)\varepsilon_{21} - \left(k_{21} + k_{22}\right)\varepsilon_{22} + \ddot{y}_2^* \qquad (3.52)$$

Trong đó, k_{22} là hằng số dương.

Thay biểu thức của ε_{21} và ε_{22} vào (3.52), và chuyển sang toán tử Laplace ta được:

$$U_2(s) = \left(1 + k_{21}k_{22}\right)\left[Y_2^*(s) - X_{21}(s)\right] + s\left(k_{21} + k_{22}\right)\left[Y_2^*(s) - X_{21}(s)\right] + s^2 Y_2^*(s) \quad (3.53)$$

Bây giờ, ta sẽ trượt thành phần tích phân về phía phương trình bộ điều khiển, từ công thức (3.43), ta có:

$$W_2(s) = \frac{U_2(s)}{s} \qquad (3.53)$$

Vậy phương trình bộ điều khiển thành phần i_{rq} sẽ là:

$$W_2(s) = \frac{\left(1 + k_{21}k_{22}\right)}{s}\left[Y_2^*(s) - X_{21}(s)\right] + \left(k_{21} + k_{22}\right)\left[Y_2^*(s) - X_{21}(s)\right] + sY_2^*(s) \quad (3.54)$$

Hay viết gọn hơn ta có:

$$W_2(s) = -\frac{k_{I2}}{s} z_2 - k_{P2} z_2 + s i_{rq}^*(s) \qquad (3.54)$$

Hay:

$$W_2 = -k_{I2}\int z_2 dt - k_{P2} z_2 + \frac{di_{rq}^*}{dt} \qquad (3.55)$$

Trong đó: $k_{I2} = 1 + k_{21}k_{22}$; $k_{P2} = k_{21} + k_{22}$

Chuyển sang miền toán tử z, ta có:

$$w_2(z) = \frac{\left(1 + k_{21}k_{22}\right)Tz^{-1}}{1 - z^{-1}}\left[y_2^*(z) - x_{21}(z)\right] + \left(k_{21} + k_{22}\right)\left[y_2^*(z) - x_{21}(z)\right] + \frac{1 - z^{-1}}{T * z^{-1}} y_2^*(z) \quad (3.56)$$

c) Chọn giá trị các điểm cực

- *Chọn các hệ số k_{11} và k_{12}*

Với bộ điều khiển thành phần i_{rd} (3.36) hoặc (3.39), ta có phương trình các biến sai lệch như sau:

$$\begin{bmatrix} \dot{\varepsilon}_{11} \\ \dot{\varepsilon}_{12} \end{bmatrix} = \begin{bmatrix} -k_{11} & 1 \\ -1 & -k_{12} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{12} \end{bmatrix} = A_{10}\varepsilon_1 \tag{3.57}$$

Trong đó:

$$A_{10} = \begin{bmatrix} -k_{11} & 1 \\ -1 & -k_{12} \end{bmatrix}, \varepsilon_1 = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} \end{bmatrix}^T$$

Phương trình (3.57) có thể được viết lại thành:

$$D_1(s)\varepsilon_1 = 0 \tag{3.58}$$

Trong đó: $$D_1(s) = \left[Is\text{-}A_{10}\right] = \begin{bmatrix} (s+k_{11}) & -1 \\ 1 & (s+k_{12}) \end{bmatrix} \tag{3.59}$$

Để xác định các điểm cực, ta tính định thức:

$$\det[D_1(s)] = \left|Is\text{-}A_{10}\right| = \begin{vmatrix} (s+k_{11}) & -1 \\ 1 & (s+k_{12}) \end{vmatrix} = (s+k_{11})(s+k_{12})+1 = s^2 + (k_{11}+k_{12})s + k_{11}k_{12} + 1 \tag{3.60}$$

Nếu ta chọn k_{11} và k_{12} sao cho: $k_{11}k_{12} >> 1$, thì:

$$\det[D_1(s)] \approx (s+k_{11})(s+k_{12}) \tag{3.61}$$

Từ (3.61), ta có được các điểm cực là:

$p_{11} = -k_{11}, p_{12} = -k_{12}$, với điều kiện: $k_{11}k_{12} >> 1$

- *Chọn các hệ số k_{21} và k_{22}*

Với bộ điều khiển thành phần i_{rq} (3.54) hoặc (3.56), ta có phương trình các biến sai lệch như sau:

$$\begin{bmatrix} \dot{\varepsilon}_{21} \\ \dot{\varepsilon}_{22} \end{bmatrix} = \begin{bmatrix} -k_{21} & 1 \\ -1 & -k_{22} \end{bmatrix} \begin{bmatrix} \varepsilon_{21} \\ \varepsilon_{22} \end{bmatrix} = A_{20}\varepsilon_2 \tag{3.62}$$

Trong đó:

$$A_{20} = \begin{bmatrix} -k_{21} & 1 \\ -1 & -k_{22} \end{bmatrix}, \varepsilon_2 = \begin{bmatrix} \varepsilon_{21} & \varepsilon_{22} \end{bmatrix}^T \quad (3.63)$$

Phương trình (3.62) có thể được viết lại thành:

$$D_2(s)\varepsilon_2 = 0 \quad (3.64)$$

Trong đó:

$$D_2(s) = \left[\text{Is-}A_{20}\right] = \begin{bmatrix} (s+k_{21}) & -1 \\ 1 & (s+k_{22}) \end{bmatrix} \quad (3.65)$$

Để xác định các điểm cực, ta tính định thức:

$$\det[D_2(s)] = \left|\text{Is-}A_{20}\right| = \begin{vmatrix} (s+k_{21}) & -1 \\ 1 & (s+k_{22}) \end{vmatrix} \quad (3.66)$$
$$= (s+k_{21})(s+k_{22}) + 1 = s^2 + (k_{21}+k_{22})s + k_{21}k_{22} + 1$$

Nếu ta chọn k_{21} và k_{22} sao cho: $k_{21}k_{22} >> 1$, thì:

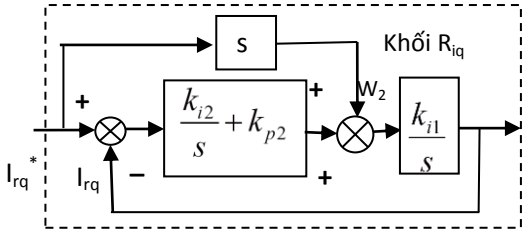
$$\det[D_2(s)] \approx (s+k_{21})(s+k_{22}) \quad (3.67)$$

Từ (3.67), ta có được các điểm cực là:

$p_{21} = -k_{21}, p_{22} = -k_{22}$, với điều kiện: $k_{21}k_{22} >> 1$

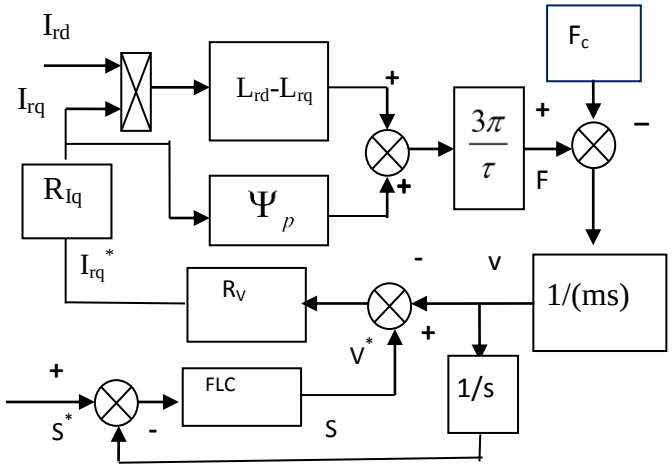
3.2. Thiết kế bộ điều khiển vận tốc

Từ (3.39) và (3.54), ta có sơ đồ khối mạch vòng điều chỉnh thành phần dòng i_{rq} như hình 3.3.



Hình 3.3. Sơ đồ khối mạch vòng điều chỉnh thành phần dòng điện i_{rq}

Từ (3.1),(3.2),(3.3) và sơ đồ khối hình 3.3, ta có sơ đồ khối mạch vòng vận tốc và vị trí như hình 3.4.



Hình 3.4. Sơ đồ khối mạch vòng điều chỉnh vận tốc và mạch vòng điều khiển vị trísử dụng bộ điều khiển PID mờ FLC.

Từ các sơ đồ cấu trúc hình 3.3 và 3.4, áp dụng tiêu chuẩn mô đun đối xứng, ta tìm được bộ điều khiển vận tốc như (3.68)

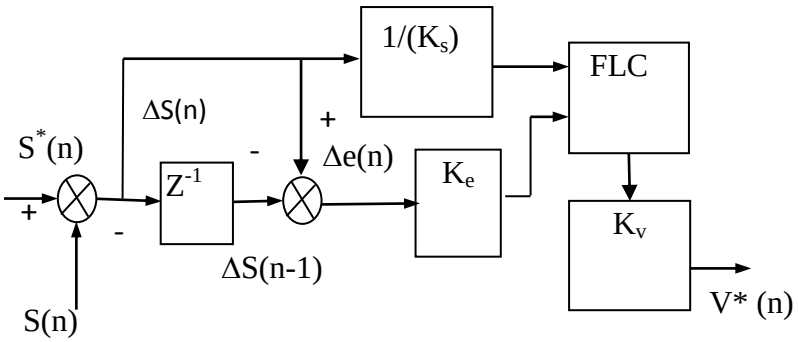
$$R_v(s) = \frac{\tau m\left(s^2 + k_{p2}s + k_{i2}\right)}{6\pi\tau_\sigma\psi_p\left(k_{p2}s + k_{i2}\right)\left(1+\tau_\sigma s\right)} \qquad (3.68)$$

3.3. Thiết kế bộ điều khiển PID mờ cho mạch vòng điều chỉnh vị trí

3.3.1. Xác định các biến ngôn ngữ vào và ra

Với mục đích chính của hệ thống điều khiển là đạt được vị trí yêu cầu với độ chính xác cao, theo kinh nghiệm thực tế, ta chọn các biến ngôn ngữ vào là sai số vị trí (quãng đường) ΔS và đạo hàm sai số vị trí Δe, biến ngôn ngữ đầu ra là vận tốc yêu cầu v^*, do đó quan hệ hàm của FLC là:

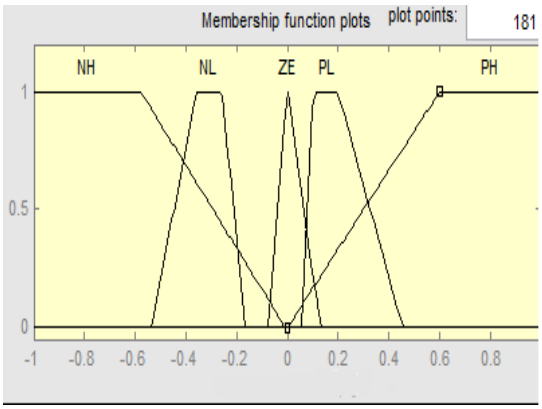
$V^*(n) = f(\Delta e(n), \Delta S(n))$. Trong đó: $\Delta e(n) = \Delta S(n) - \Delta S(n-1)$ là đạo hàm sai số vị trí, $\Delta S(n) = S^*(n) - S(n)$ là mẫu hiện tại của sai số vị trí, $\Delta S(n-1)$ là mẫu quá khứ của sai số vị trí, S(n) là mẫu hiện tại của vị trí thực, $S^*(n)$ là mẫu hiện tại của vị trí yêu cầu, f là kí hiệu của hàm phi tuyến. Cấu trúc bộ điều khiển vị trí kiểu PID mờ được thể hiện như hình 3.5. Các hệ số K_e, K_s được chọn sao cho các giá trị được chuẩn hoá của sai số vị trí $\Delta S(n)$, và đạo hàm sai số vị trí $\Delta e(n)$ biến thiên trong khoảng [-1,+1]. Hệ số K_V được chọn sao cho đầu ra của bộ điều khiển là vận tốc định mức yêu cầu. Ở đây, các hệ số được lấy là: $K_S = S^*$ (vị trí đặt), $K_e = 10$, $K_V = 12,5$.



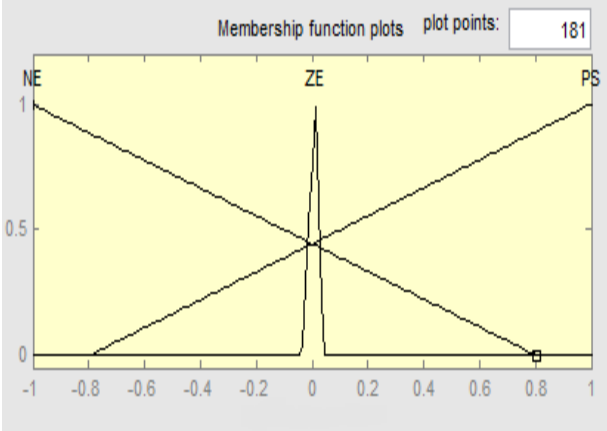
Hình 3.5. Cấu trúc bộ điều khiển vị trí kiểu PID mờ.

3.3.2. Xác định dạng các hàm liên thuộc và các giá trị của biến ngôn ngữ.

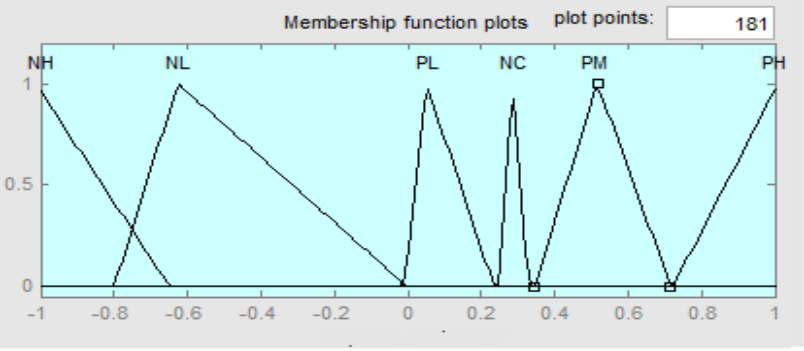
Ở đây, để đơn giản ta lựa chọn hàm liên thuộc dạng hình thang và hình tam giác, là dạng hàm liên thuộc có mức chuyển đổi tuyến tính. Dạng các hàm liên thuộc và giá trị của các biến ngôn ngữ đầu vào và ra được chỉ ra ở hình 3.6, 3.7, 3.8.



Hình 3.6. Dạng các hàm liên thuộc và giá trị các biến ngôn ngữ vào SLVT(ΔS)



Hình 3.7. Dạng các hàm liên thuộc và giá trị các biến ngôn ngữ vào VT(Δe)



Hình 3.8. Dạng các hàm liên thuộc và giá trị các biến ngôn ngữ ra v^*

3.3.3. Xây dựng các luật điều khiển " nếu .. thì ".

Căn cứ vào bản chất vật lý, số liệu và kinh nghiệm, ta chọn 7 luật như sau:

Luật 1: if SLVT is PH (dương cao), then V^* is PH (dương cao);

Luật 2: if SLVT is PL (dương thấp), then V^* is PM (dương trung bình);

Luật 3: if (SLVT is ZE (bằng 0)) and (VT is PS (dương)), then V^* is PL (dương thấp);

Luật 4: if (SLVT is ZE (bằng 0)) and (VT is NE (âm)), then V^* is NC (không thay đổi);

Luật 5: if (SLVT is ZE (bằng 0)) and (VT is ZE (bằng 0)), then V^* is NC (không thay đổi);

Luật 6: if SLVT is NL (âm thấp), then V^* is NL (âm thấp);

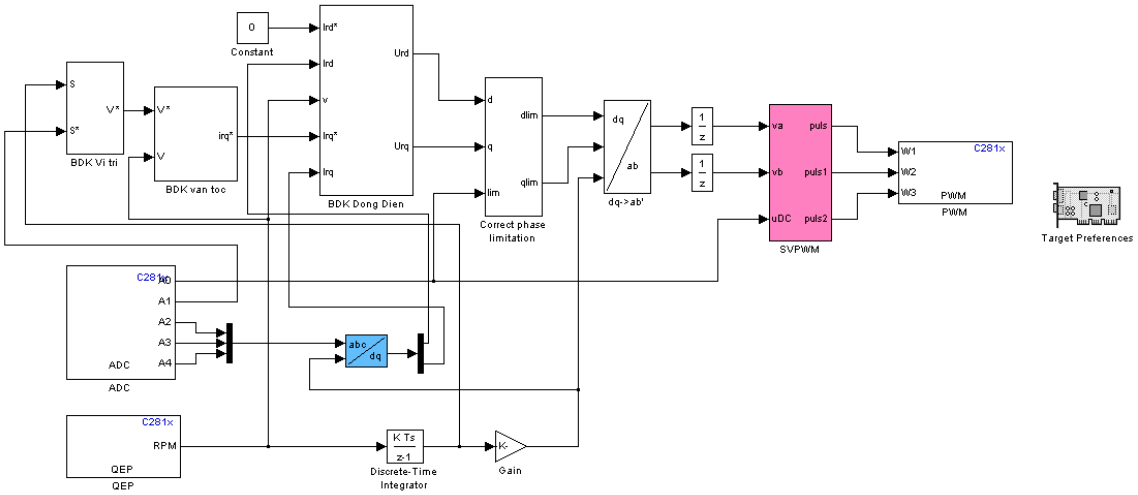
Luật 7: if SLVT is NH (âm cao), then V^* is NH (âm cao).

3.3.4. Chọn luật hợp thành và giải mờ.

Ở đây, ta chọn luật hợp thành max-min và giải mờ theo phương pháp điểm trọng tâm.

3.4 Xây dựng sơ đồ Matlab/Simulink tạo code nạp vào DSP TMS320F2812

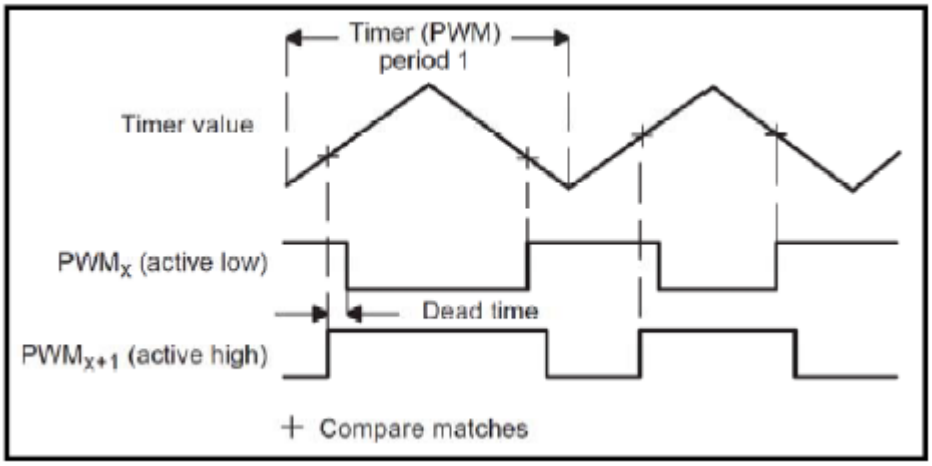
3.4.1. Sơ đồ Matlab/Simulink tạo code nạp vào DSP TMS320F2812 toàn bộ hệ thống.



Hình 3.9. Sơ đồ Matlab/Simulink hệ thống

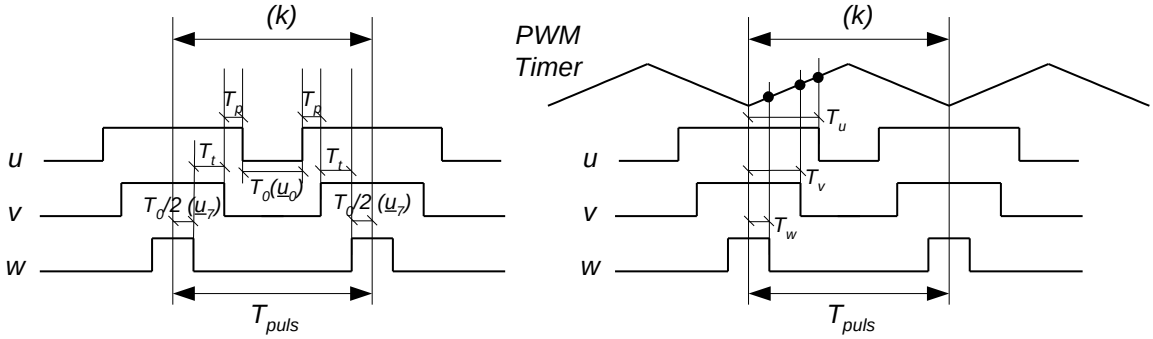
3.4.2. Khâu điều chế vector không gian

GP timer của TMS320F2812 được thiết lập ở chế độ continuous up/down counting mode, các xung ở cac cổng ra tương ứng xuất hiện ở giữa cac chu kỳ xung được lưu trong GP timer như hình 3.10.



Hình 3.10. Xung được tạo ra khi so sánh giá trị trong các thanh ghi so sánh và thanh ghi GPtimer

Hình 3.12 là khâu điều chế vector không gian mô phỏng trên Simulink. Sau khi tính được thời gian thực hiện các vector biên T_p, T_t, T_o/T_7 (theo tài liệu [9]), ta sử dụng khối *sequencing factors* để tính thời gian đóng mở van. Ý tưởng như sau: Bằng cách so sánh mẫu xung điều chế ở hình 5.13a với sơ đồ định nghĩa thời gian đóng ngắt van ở hình 5.13b (cho trường hợp Sector 1) ta thu được kết quả sau:



Hình 3.11a,b. Mẫu xung điều chế và sơ đồ định nghĩa thời gian đóng ngắt van

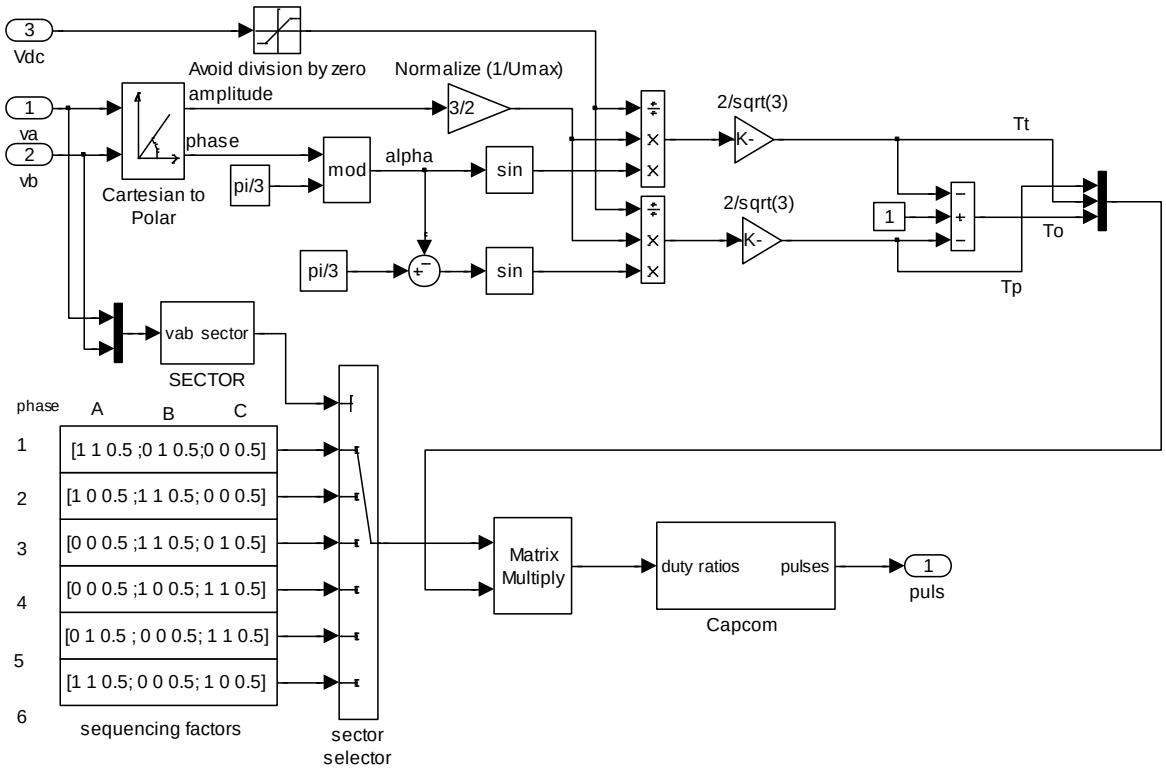
$$T_{u\_on} = T_{u\_off} = T_u = T_p + T_t + 0{,}5T_0$$

$$T_{v\_on} = T_{v\_off} = T_v = T_t + 0{,}5T_0$$

$$T_{w\_on} = T_{w\_off} = T_w = 0{,}5T_0$$

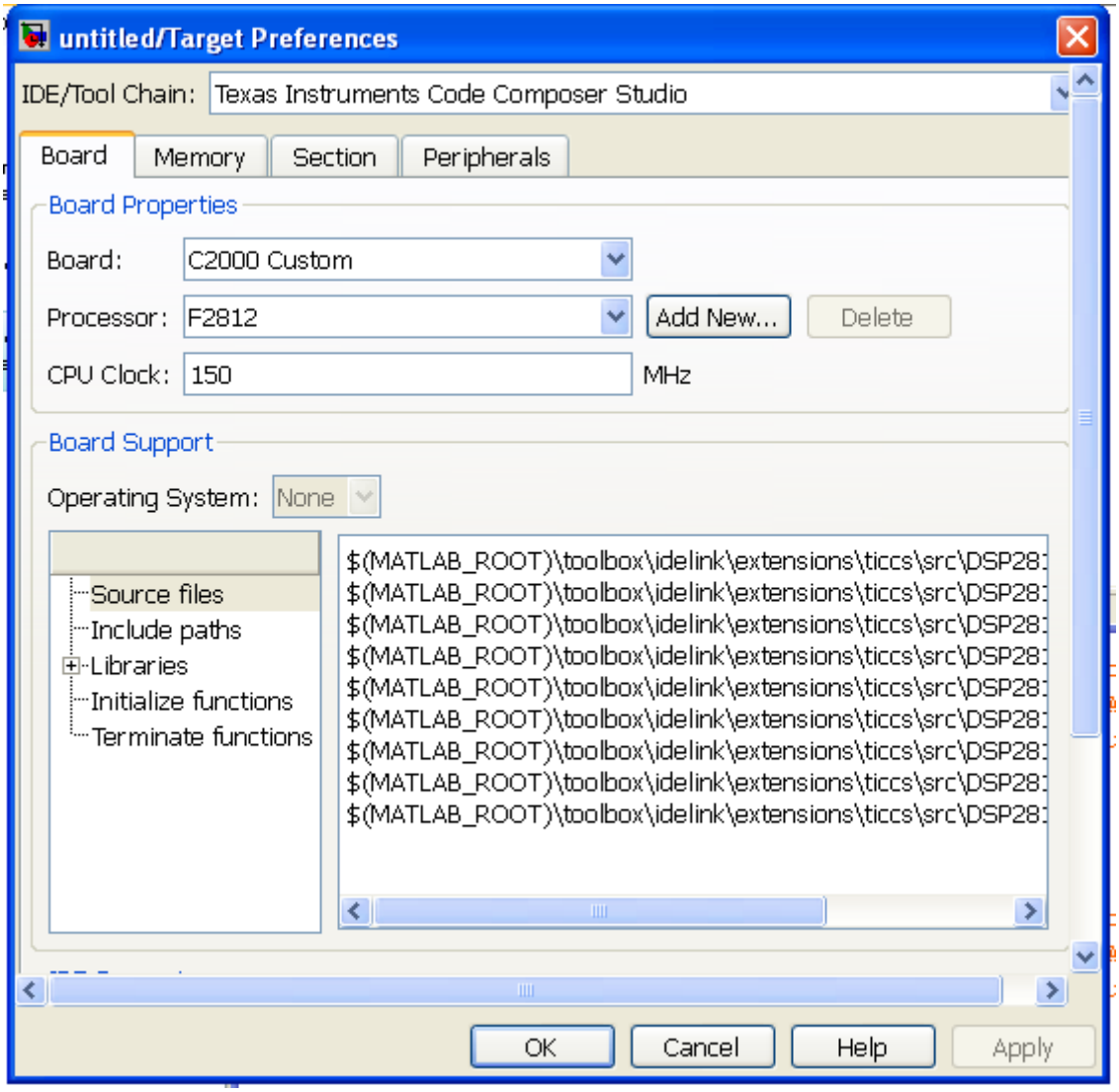
Do đó ta có: $$\begin{bmatrix} T_u \\ T_v \\ T_w \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0{,}5 \\ 0 & 1 & 0{,}5 \\ 0 & 0 & 0{,}5 \end{bmatrix} \begin{bmatrix} T_p \\ T_t \\ T_0 \end{bmatrix}.$$

Tương tự ta cũng có thể xác định được T_u, T_v, T_w cho các sector còn lại.



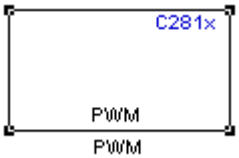
Hình 3.12. Sơ đồ Simulink mô phỏng khâu ĐCVTKG ("*SVPWM*")

3.4.3. Khối tạo kết nối MatLab/Simulink với phần mềm tạo code và nạp code cho TMS320F2812

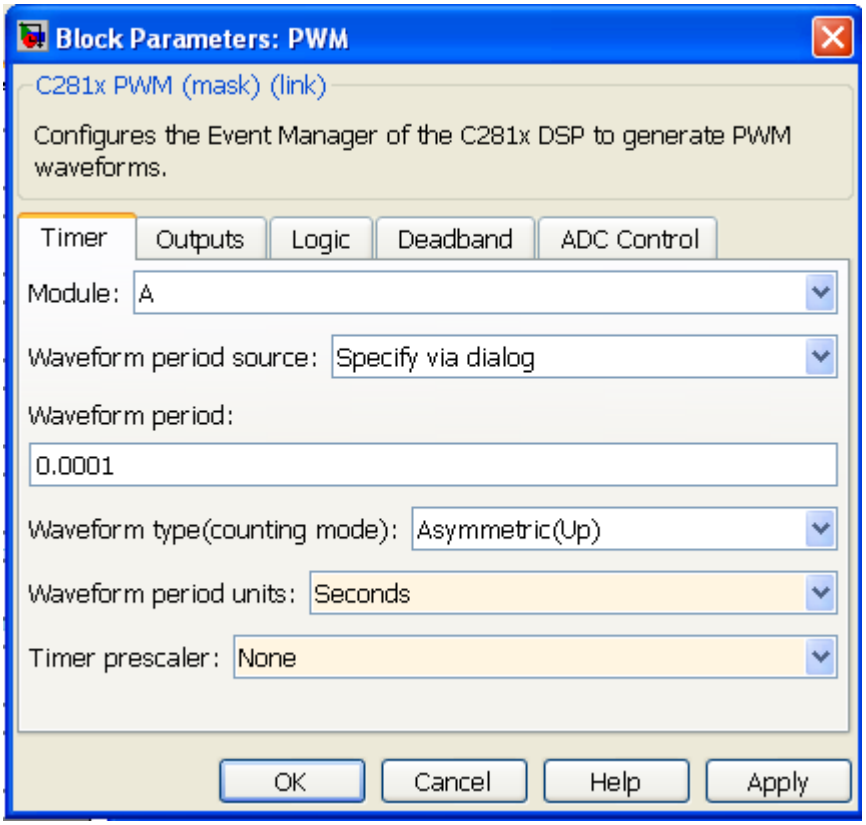


Hình 3.13. Khối tạo kết nối MatLab/Simulink với phần mềm tạo code và nạp code cho TMS320F2812

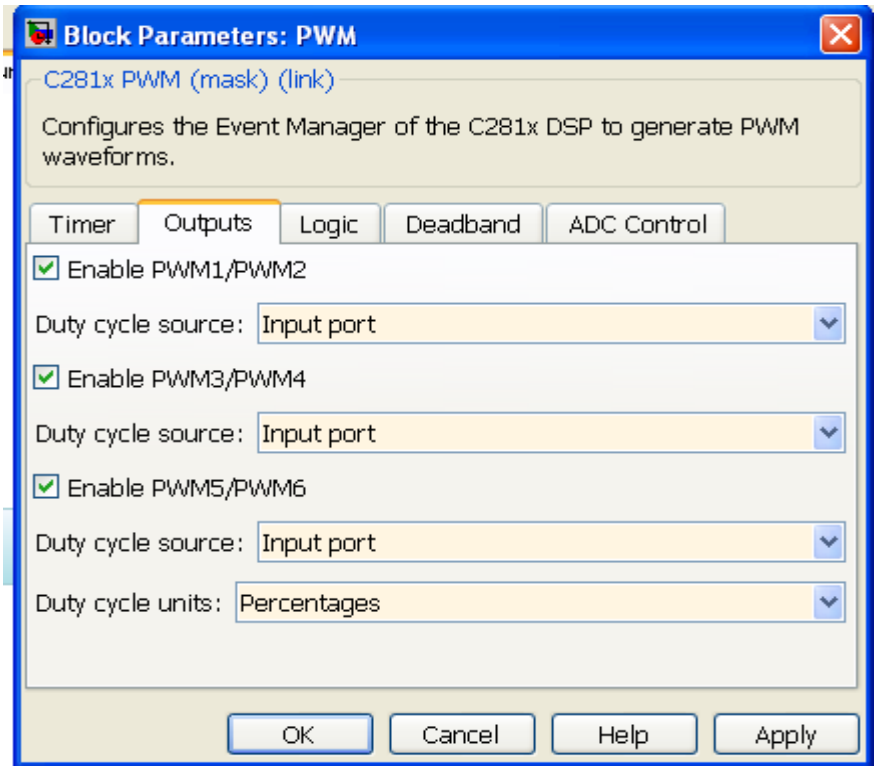
3.4.4. Khối tạo xung đưa vào các cực điều khiển của các IGBT sử dụng TMS320F2812



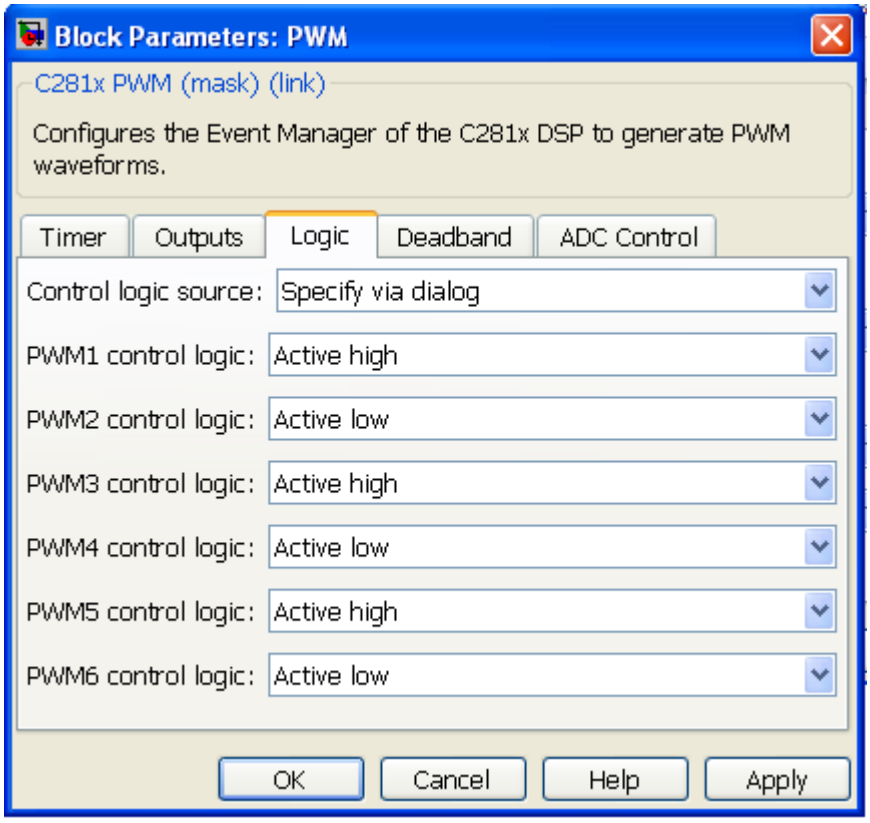
Hình 3.14. Biểu tượng của khối PWM trong MatLab/Simulink



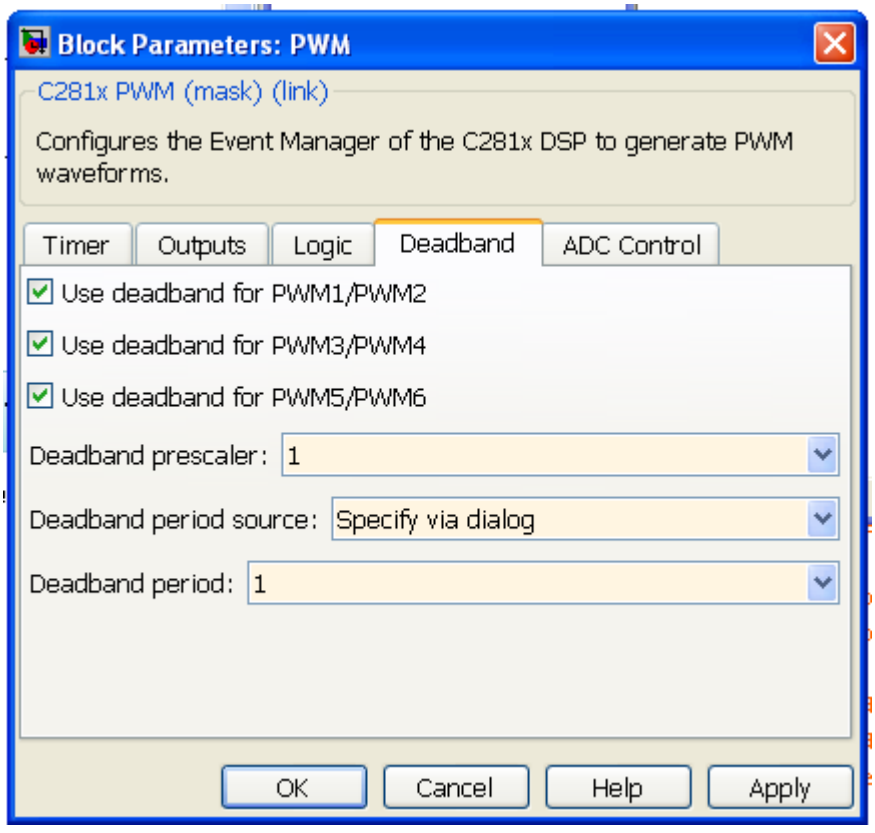
Hình 3.15. Tab Timer của khối PWM



Hình 3.16. Tab Outputs của khối PWM

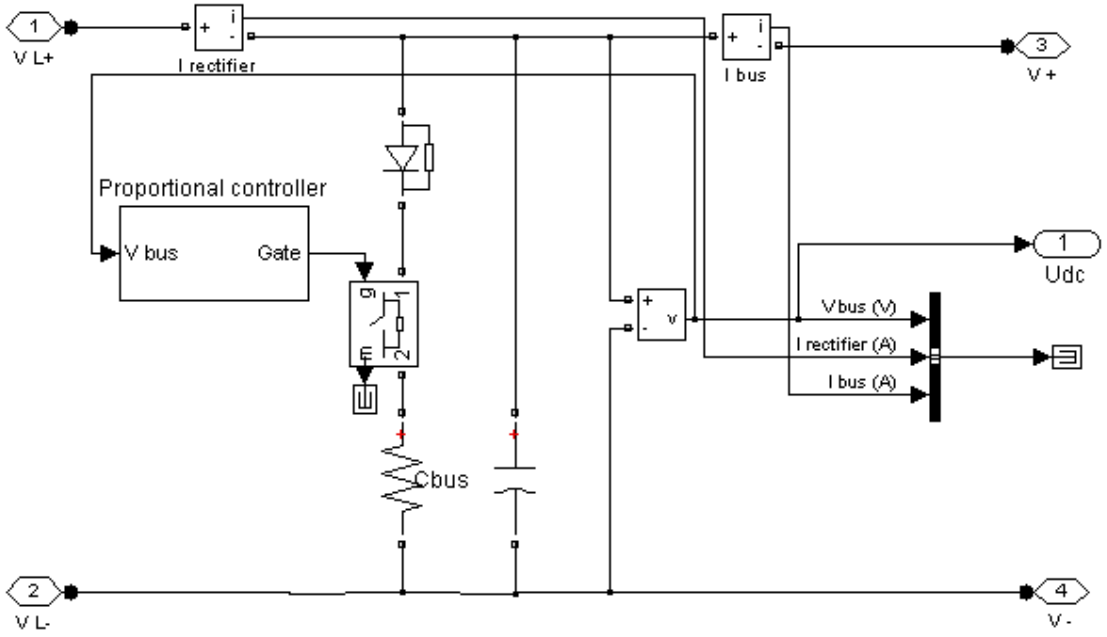


Hình 3.17. Tab Logic của khối PWM



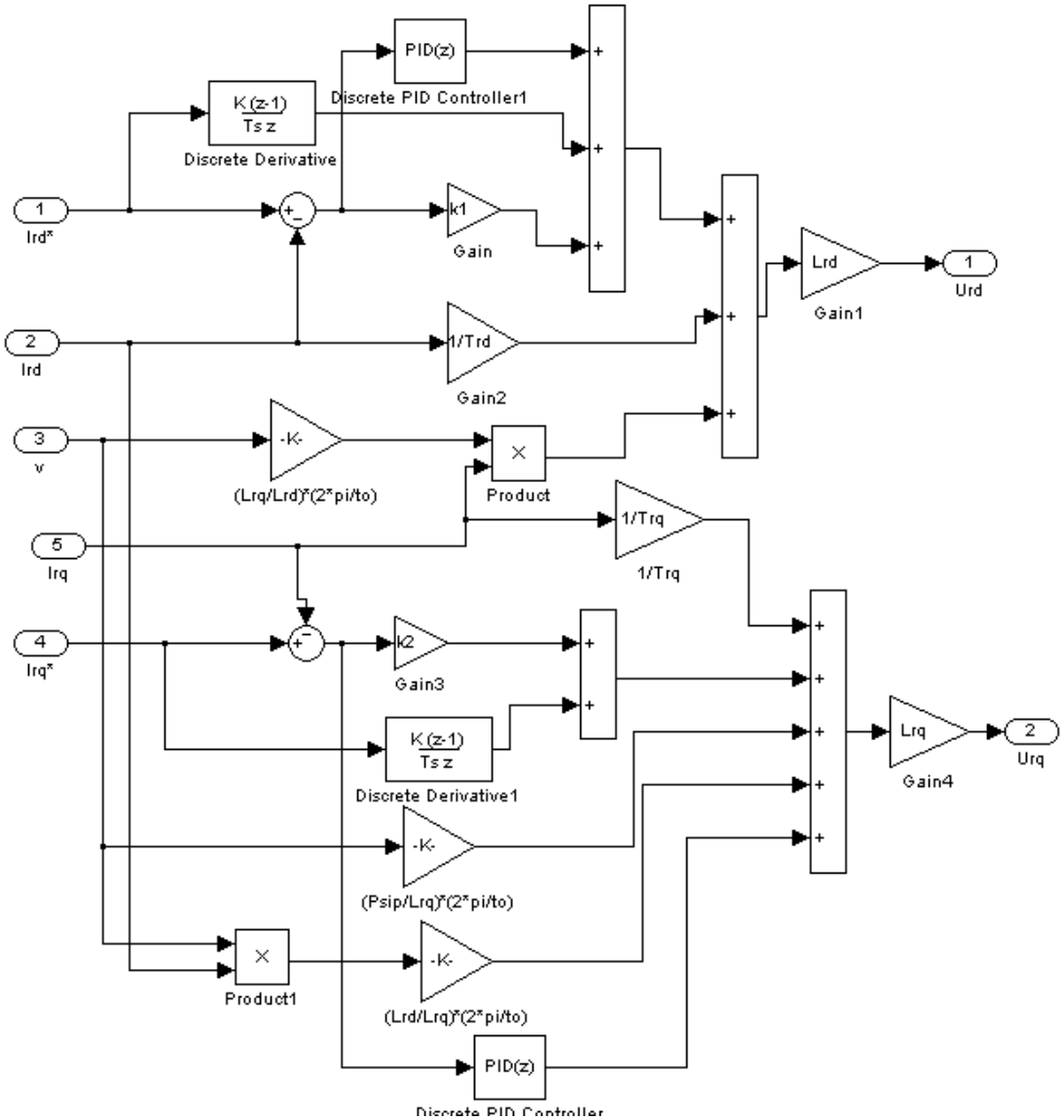
Hình 3.18. Tab Deadband của khối PWM

3.4.5. Sơ đồ simulink mạch hãm động năng



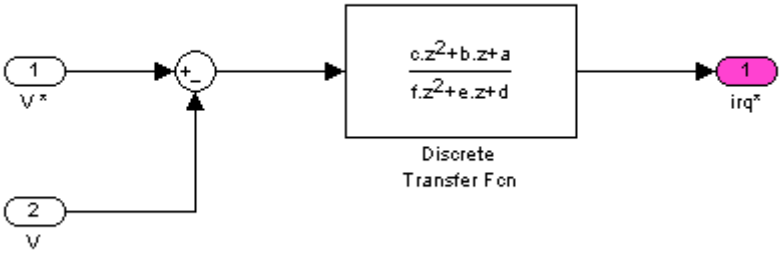
Hình 3.19. Sơ đồ simulink mạch hãm động năng

3.4.6. Sơ đồ simulink bộ điều khiển dòng điện



Hình 3.20. Sơ đồ simulink bộ điều khiển dòng điện

3.4.7. Sơ đồ simulink bộ điều khiển vận tốc



Hình 3.21. Sơ đồ simulink bộ điều khiển vận tốc

Trong đó:

$$a = \tau m\left(k_{p2}T^2 - k_{p2}T + 1\right)$$

$$b = \tau m\left(k_{p2} - 2\right)$$

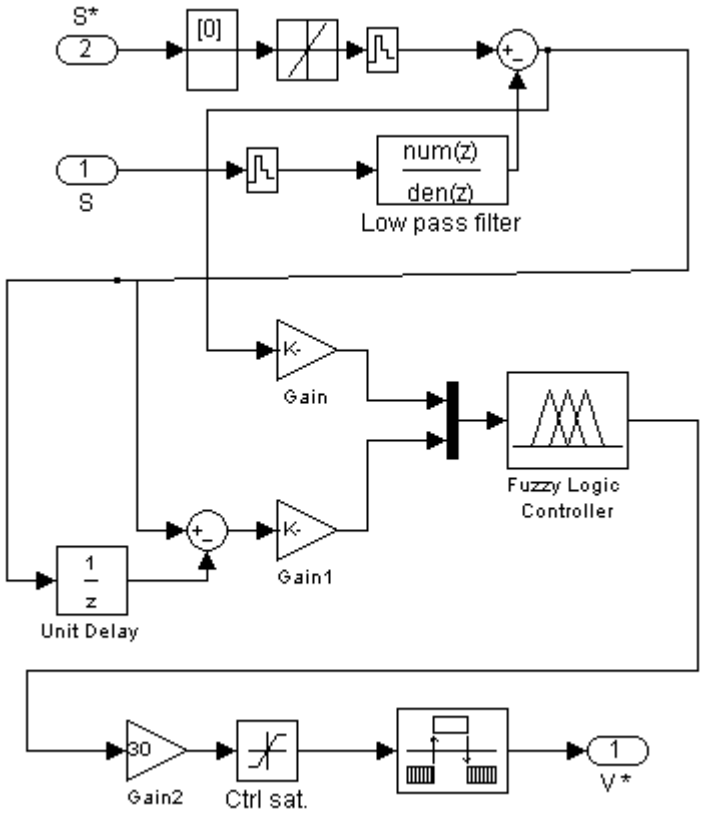
$$c = \tau m$$

$$d = 6\pi\tau_\sigma\psi_p\left[k_{p2}\tau_\sigma - T\left(k_{p2} + k_{i2}\tau_\sigma\right) + k_{i2}T^2\right]$$

$$e = 6\pi\tau_\sigma\psi_p\left[\left(k_{p2} + k_{i2}\tau_\sigma\right)T - 2k_{p2}\tau_\sigma\right]$$

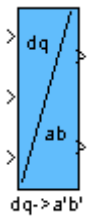
$$f = 6\pi\tau_\sigma^2\psi_p k_{p2}$$

3.4.8. Sơ đồ simulink bộ điều khiển vị trí

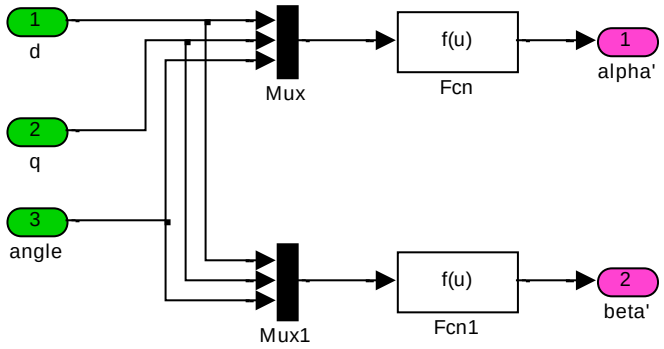


Hình 3.22. Sơ đồ simulink bộ điều khiển vị trí

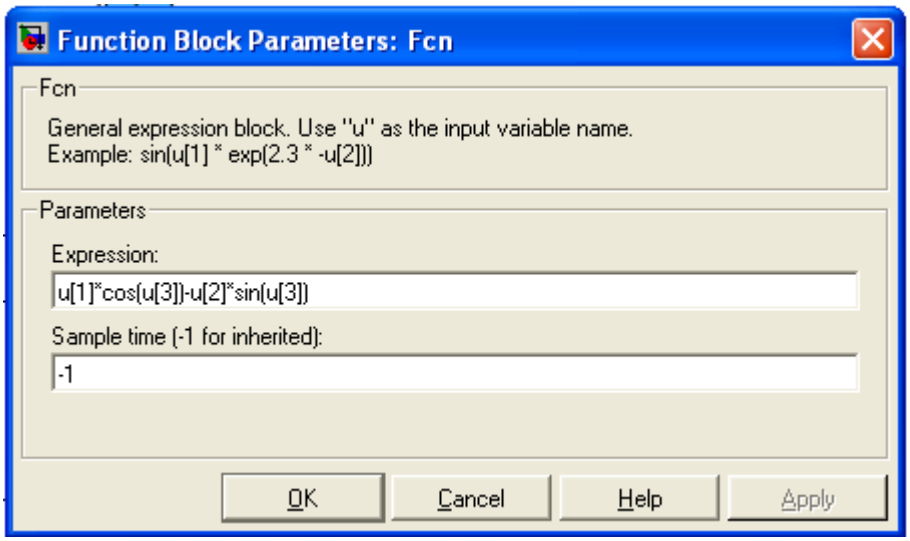
3.4.9. Khâu chuyển hệ toạ độ từ (d,q) sang (a,b)



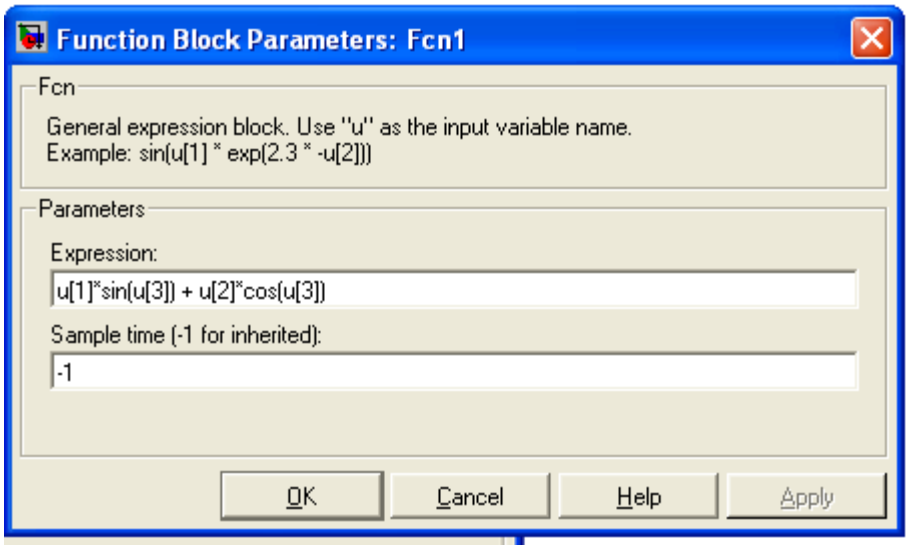
Hình 3.23. Biểu tượng khâu chuyển hệ toạ độ từ (d,q) sang (a,b)



Hình 3.24. Cấu trúc khâu chuyển hệ toạ độ từ (d,q) sang (a,b)

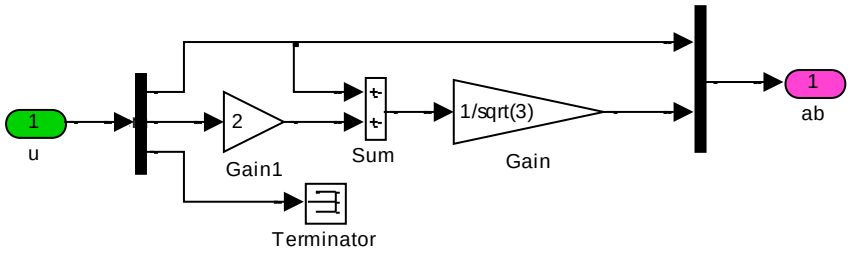


Hình 3.25. Hàm Fcn trong khâu chuyển hệ toạ độ từ (d,q) sang (a,b)

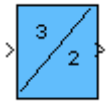


Hình 3.26. Hàm Fcn1 trong khâu chuyển hệ toạ độ từ (d,q) sang (a,b)

3.4.10. Khâu chuyển hệ toạ độ từ (a,b,c) sang (a,b).

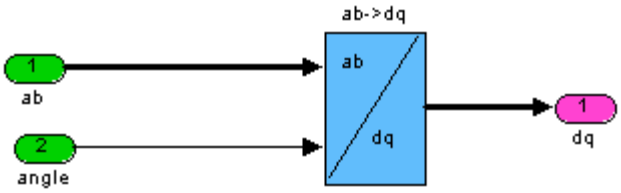


Hình 3.27. Khâu chuyển hệ toạ độ từ (a,b,c) sang (a,b)

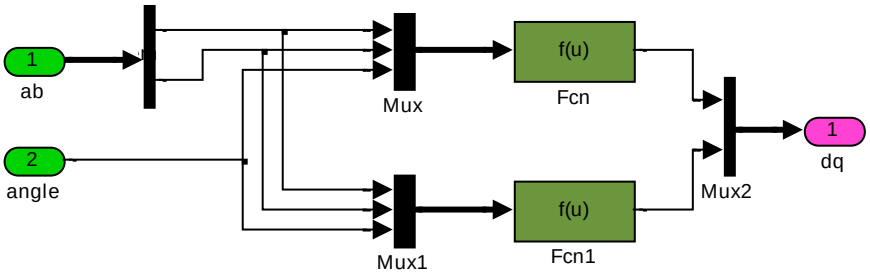


Hình 3.28. Biểu tượng khâu chuyển hệ toạ độ từ (a,b,c) sang (a,b)

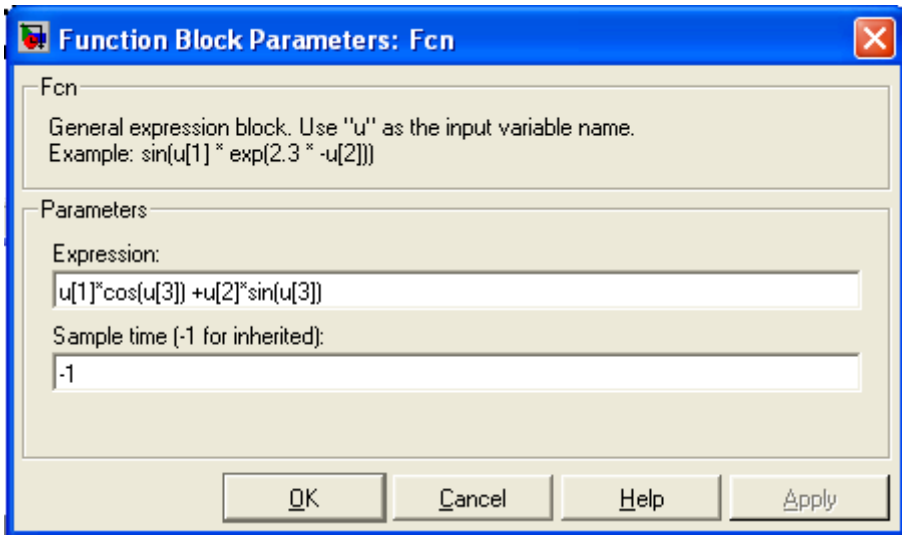
3.4.11. Khâu chuyển hệ toạ độ từ (a,b) sang (d,q).



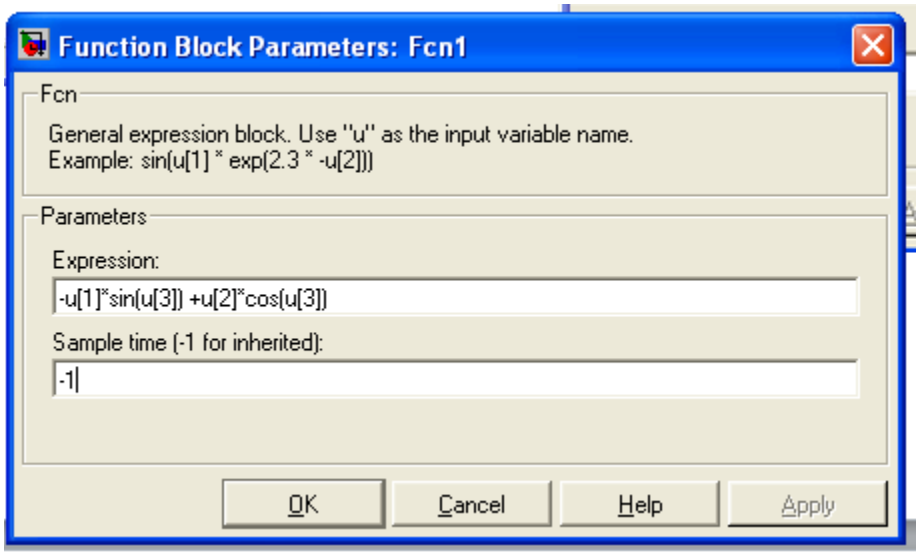
Hình 3.29. Biểu tượng khâu chuyển hệ toạ độ từ (a,b) sang (d,q).



Hình 3.30. Khâu chuyển hệ toạ độ từ (a,b) sang (d,q).

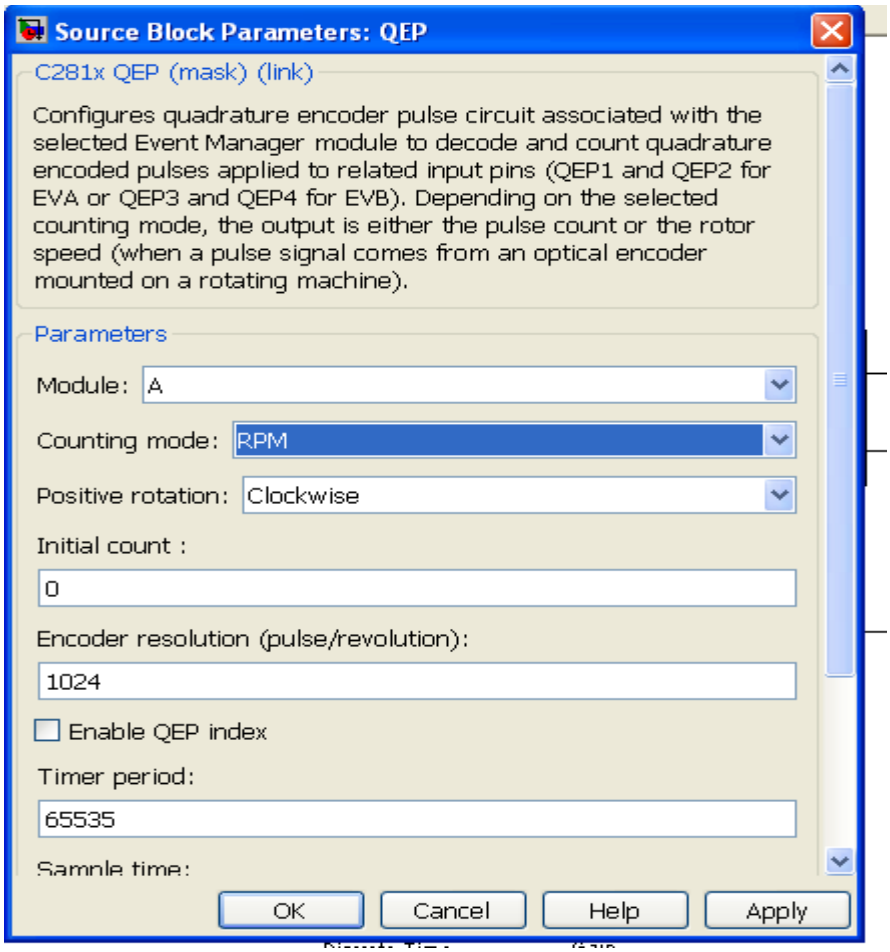


Hình 3.31.Hàm Fcn của khâu chuyển hệ toạ độ từ (a,b) sang (d,q).



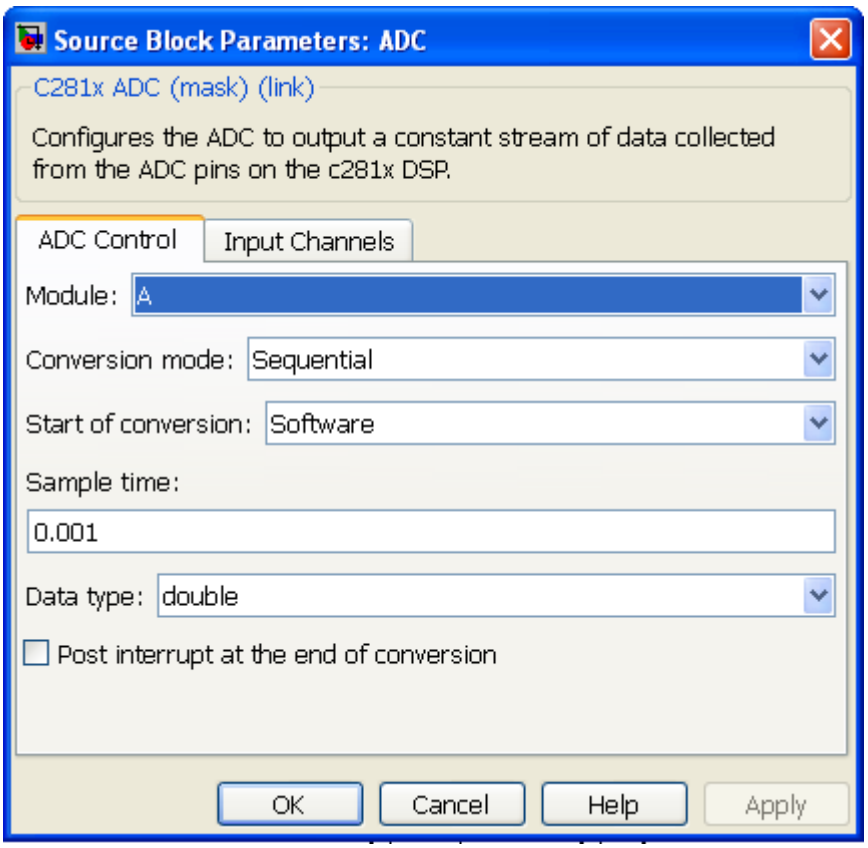
Hình 3.32. Hàm Fcn1 của khâu chuyển hệ toạ độ từ (a,b) sang (d,q).

3.4.12. Khâu đọc vận tốc từ sensor của TMS320F2812.

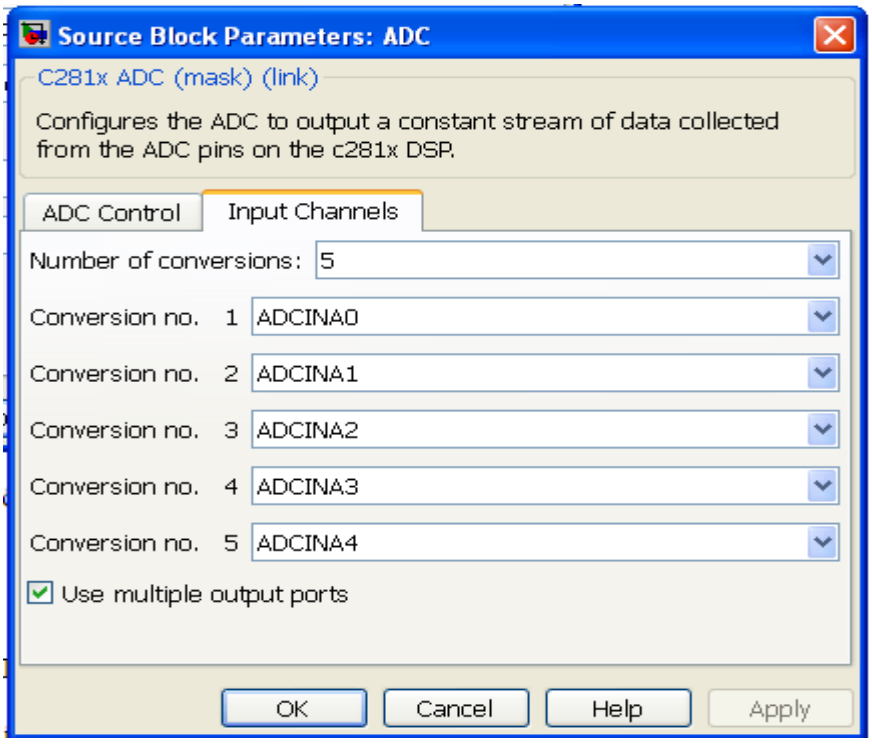


Hình 3.33. Khâu đọc vận tốc từ sensor của TMS320F2812

3.4.13. Khâu đọc dòng điện ba pha từ sensor dòng điện, đọc giá trị đặt của vị trí và đọc giá trị điện áp một chiều trung gian của TMS320F2812 (khâu ADC).



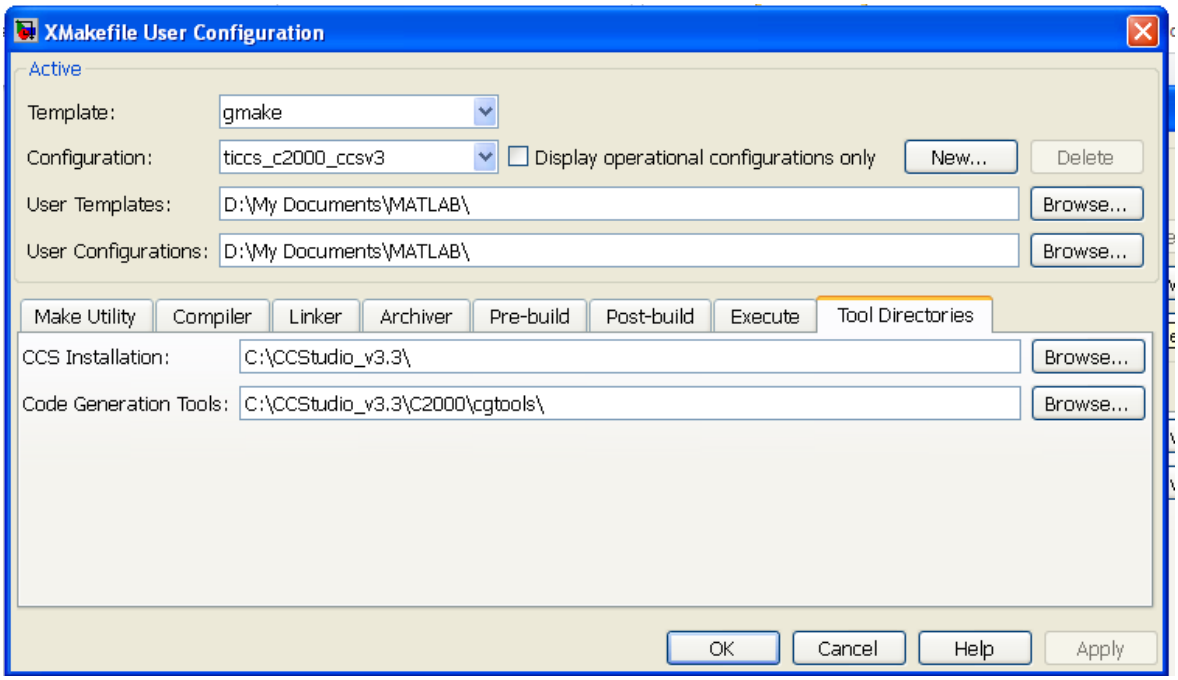
Hình 3.34. Thiết lập các tham số của Tab ADC Control



Hình 3.35.Thiết lập các tham số của Tab Input Channels của ADC Control

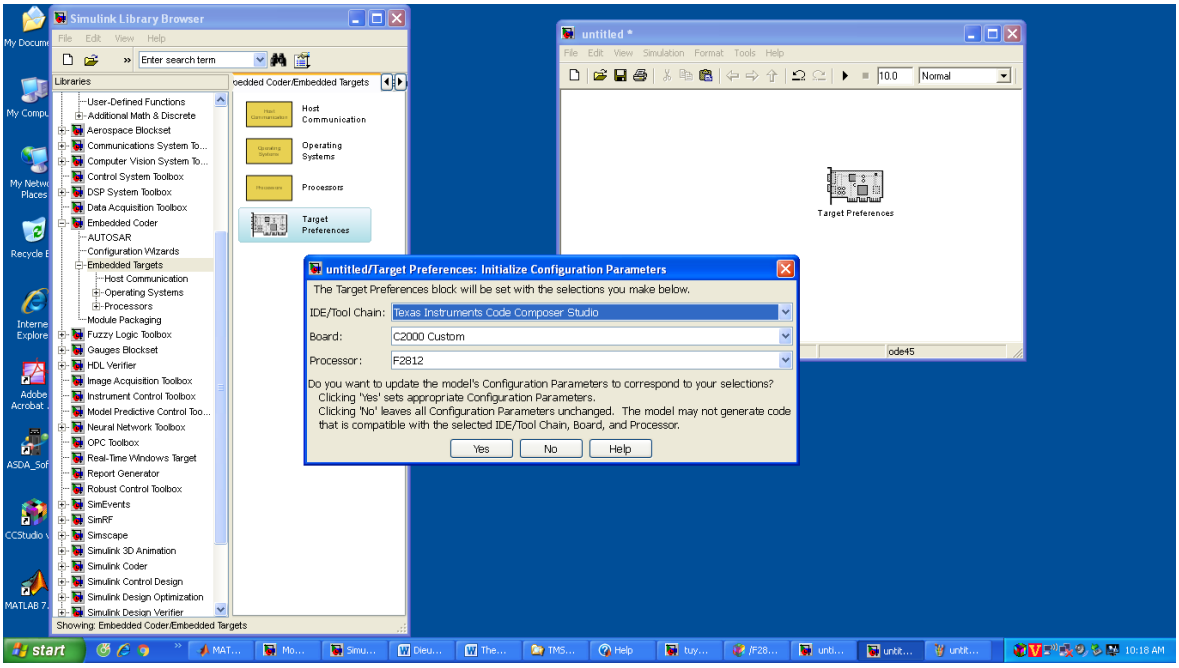
3.5. Lập trình DSPTMS320F2812 từ Matlab/Simulink và CCS

Bước 1: Gõ lệnh "xmakefilesetup" từ dấu nhắc của Matlab, matlab sẽ xuất ra hộp hội thoại XMakefile User Configuration như hình 3.36.



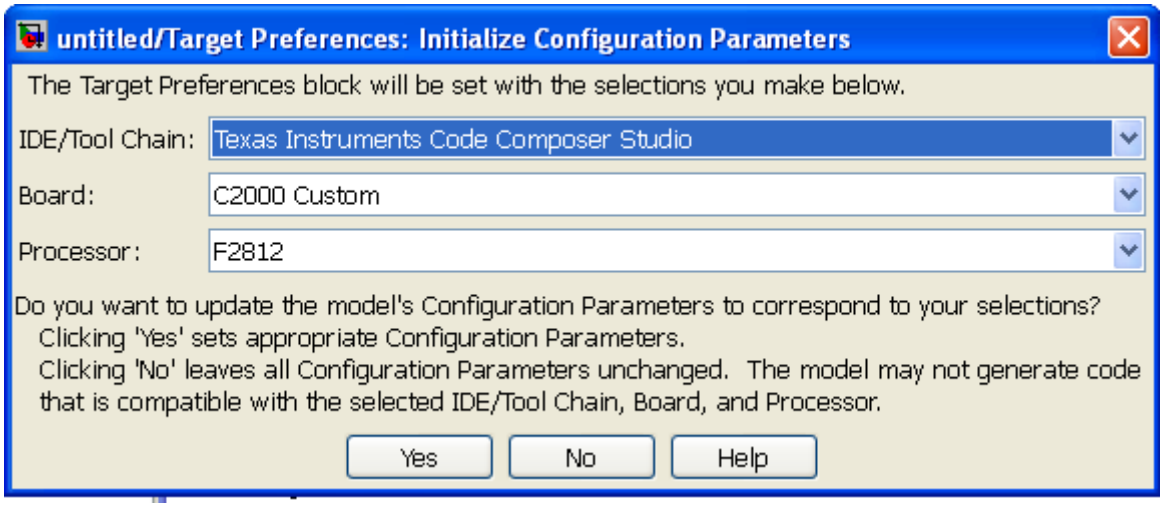
Hình 3.36. Hộp thoại XMakefile User Configuration

Bước 2: Tạo mô File Mô hình bằng lệch: File/new/Model



Hình 3.37. Tạo mô File Mô hình bằng lệch: File/new/Model

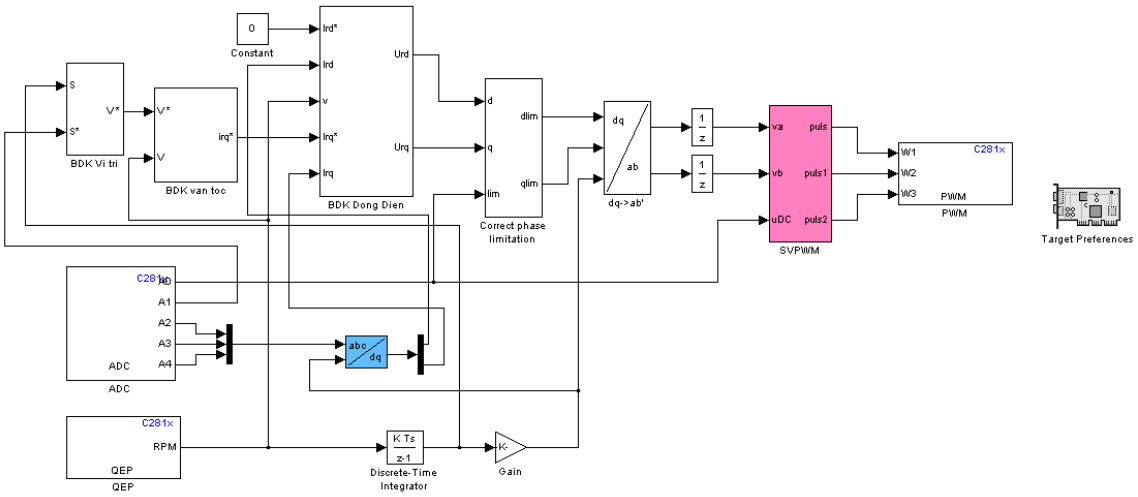
Bước 3: Chọn khối Target Preferences từ thư viện Simulink đưa vào File Model, thiết lập cấu hình cho khối



Hình 3.38. thiết lập cấu hình cho khối Target Preferences

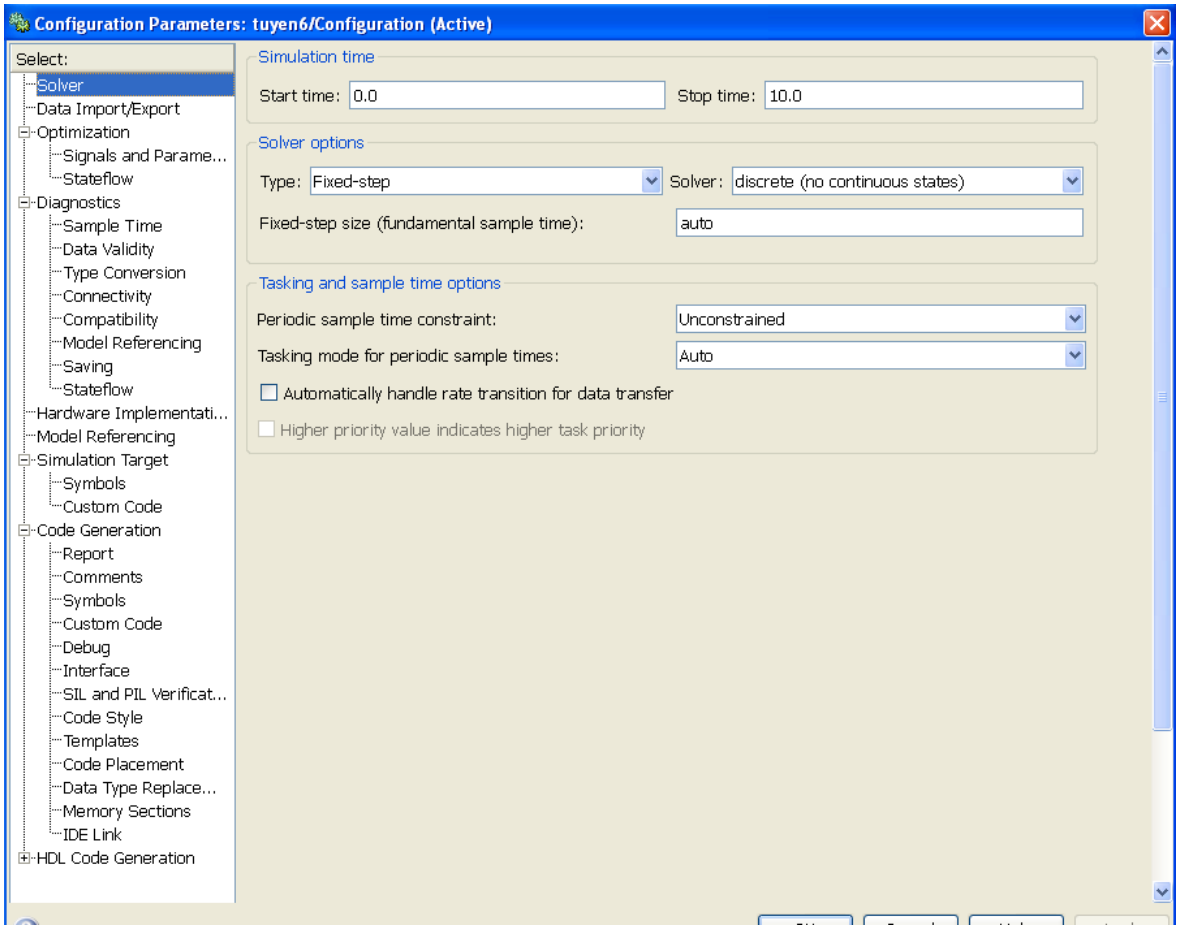
Bước 4:

Xây dựng các khối Simulink cho các bộ điều khiển dòng điện, vận tốc, vị trí, , khối điều chế vector không gian, khối tạo xung đưa vào các cực điều khiển của các IGBT sử dụng TMS320F2812, các khối chuyển toạ độ, khối đọc vận tốc từ sensor của TMS320F2812, khối đọc dòng điện ba pha từ sensor dòng điện, đọc giá trị đặt của vị trí và đọc giá trị điện áp một chiều trung gian của TMS320F2812 (khâu ADC) ở file MODEL như hình 3.39.

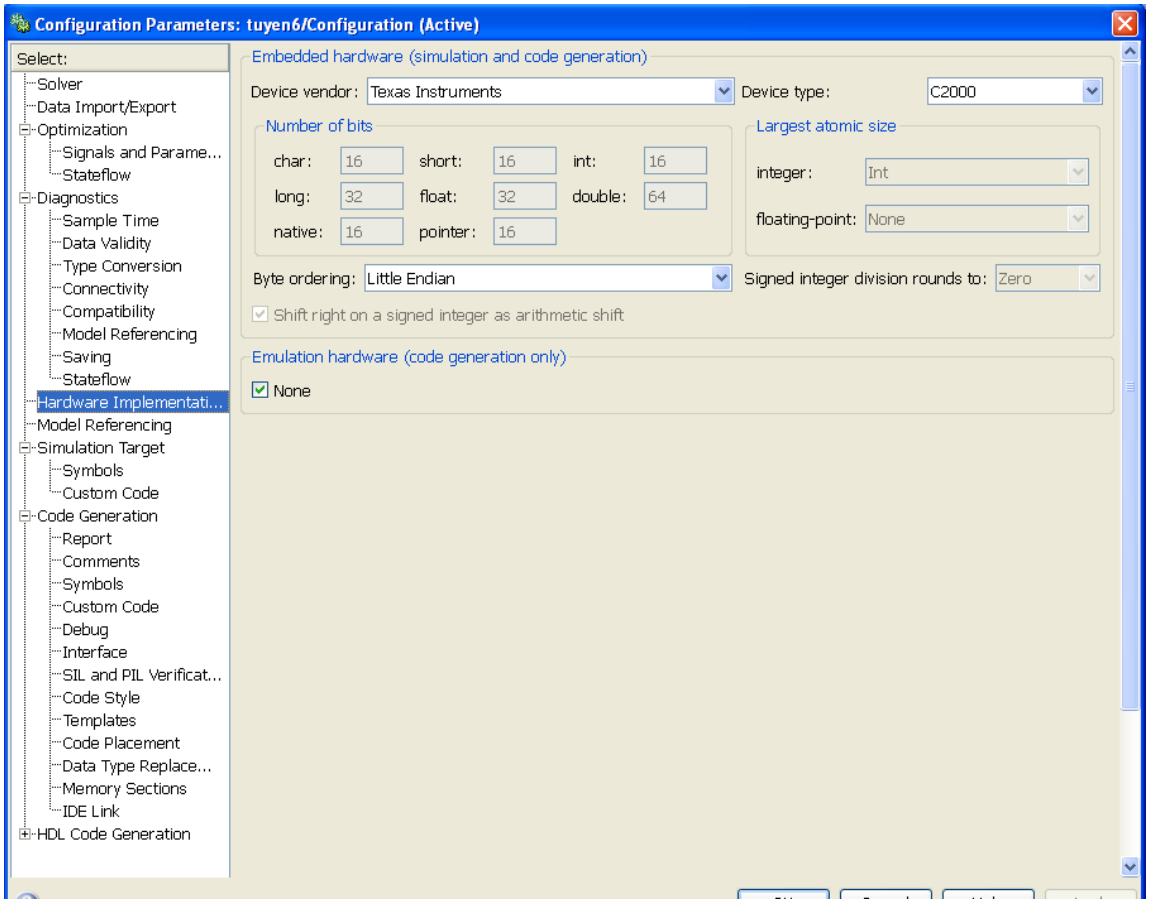


Hình 3.39. Sơ đồ Matlab/Simulink tạo code nạp vào DSP TMS320F2812

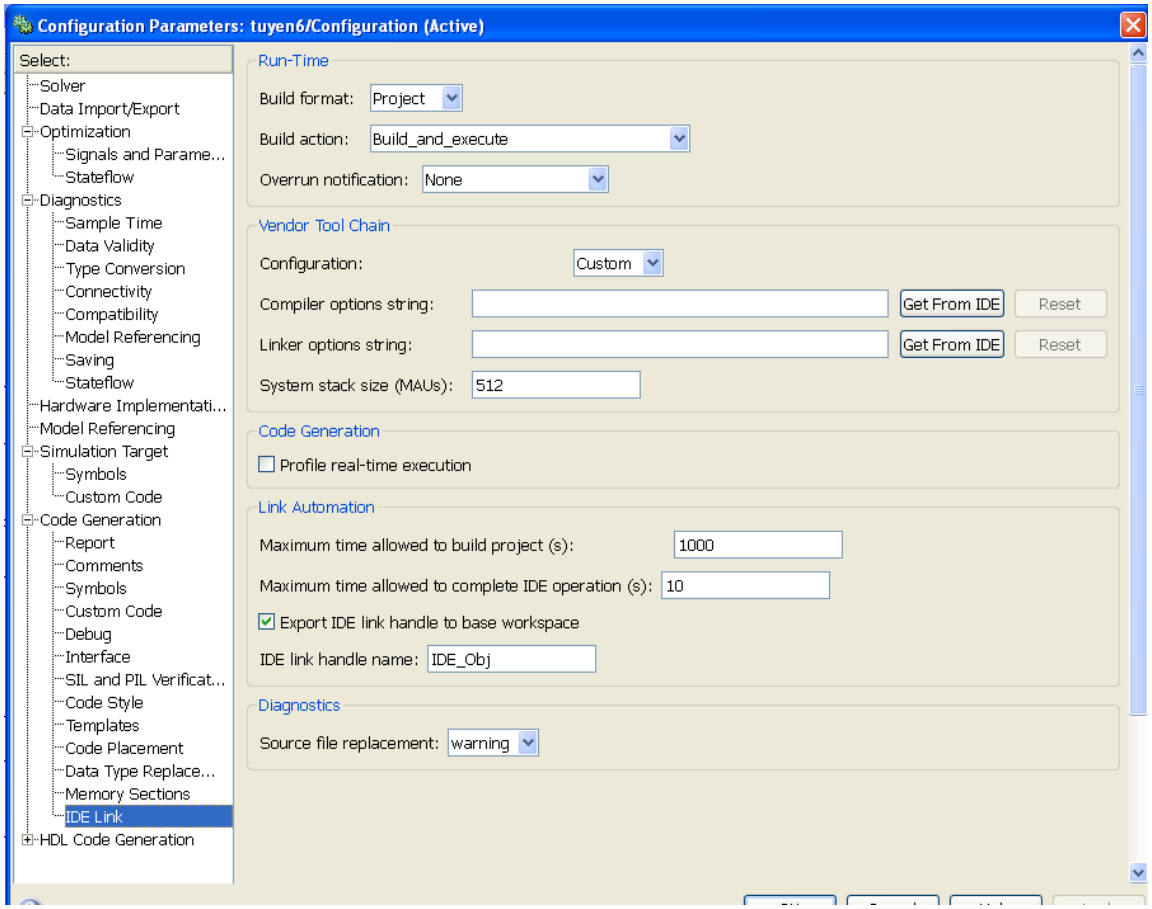
Bước 5: Thiết lập cấu hình cho file Model



Hình 3.40. Thiết lập cấu hình cho Tab solver của file Model

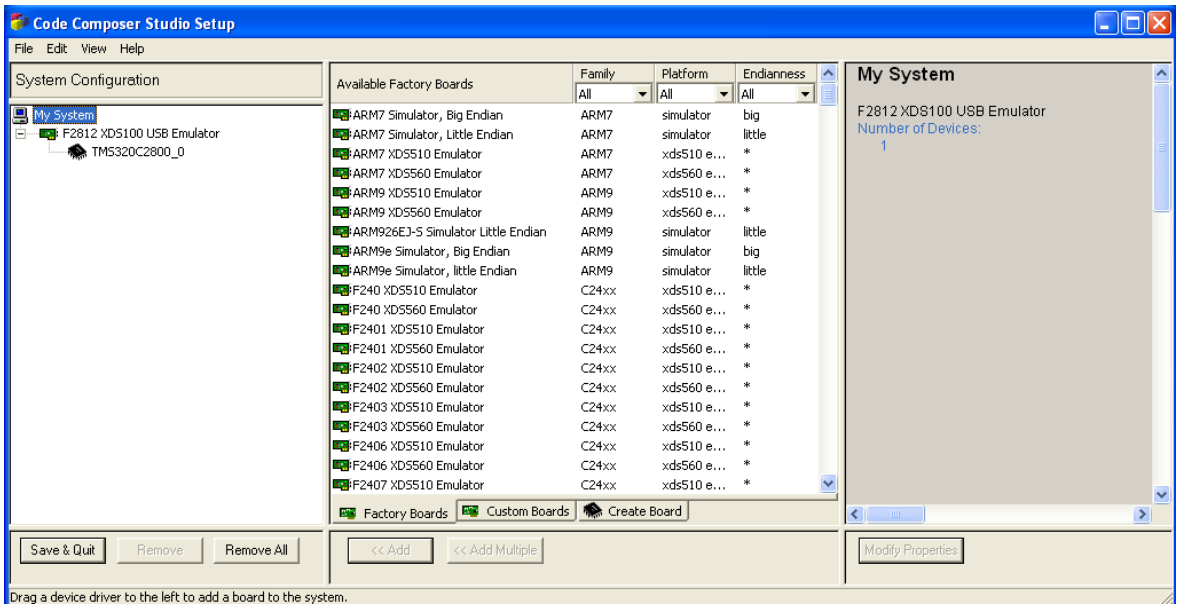


Hình 3.41. Thiết lập cấu hình cho Tab Hardware Implementationcủa file Model



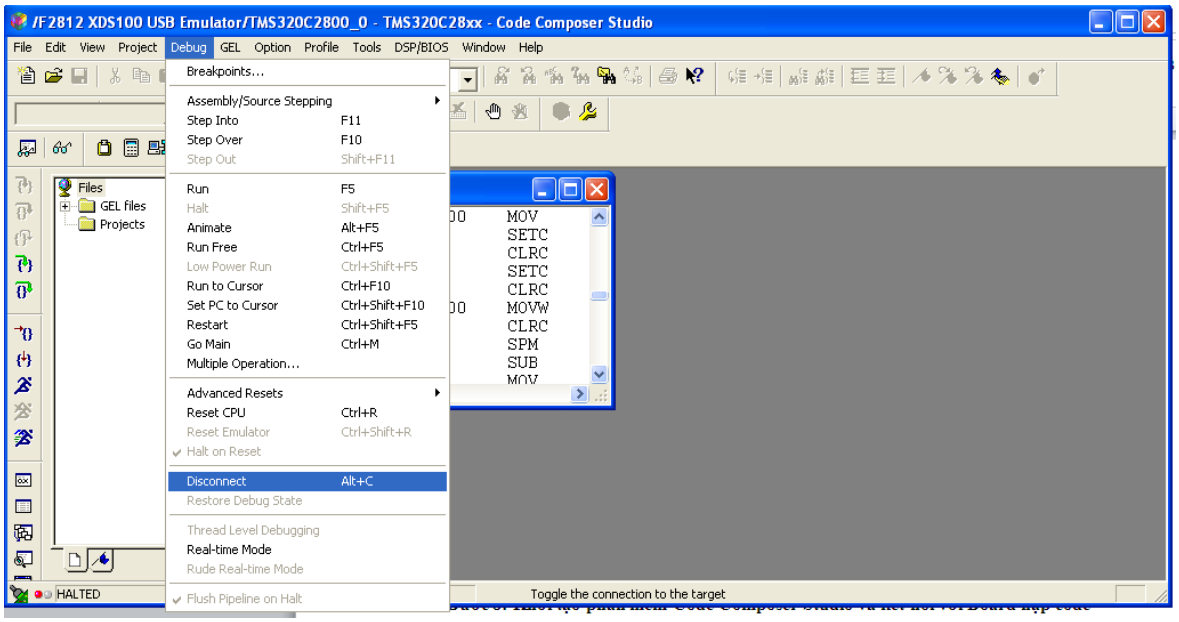
Hình 3.42. Thiết lập cấu hình cho Tab IDE link của file Model

Bước 6: Khởi tạo phần mềm Code Composer Setup, thiết lập cấu hình hệ thống

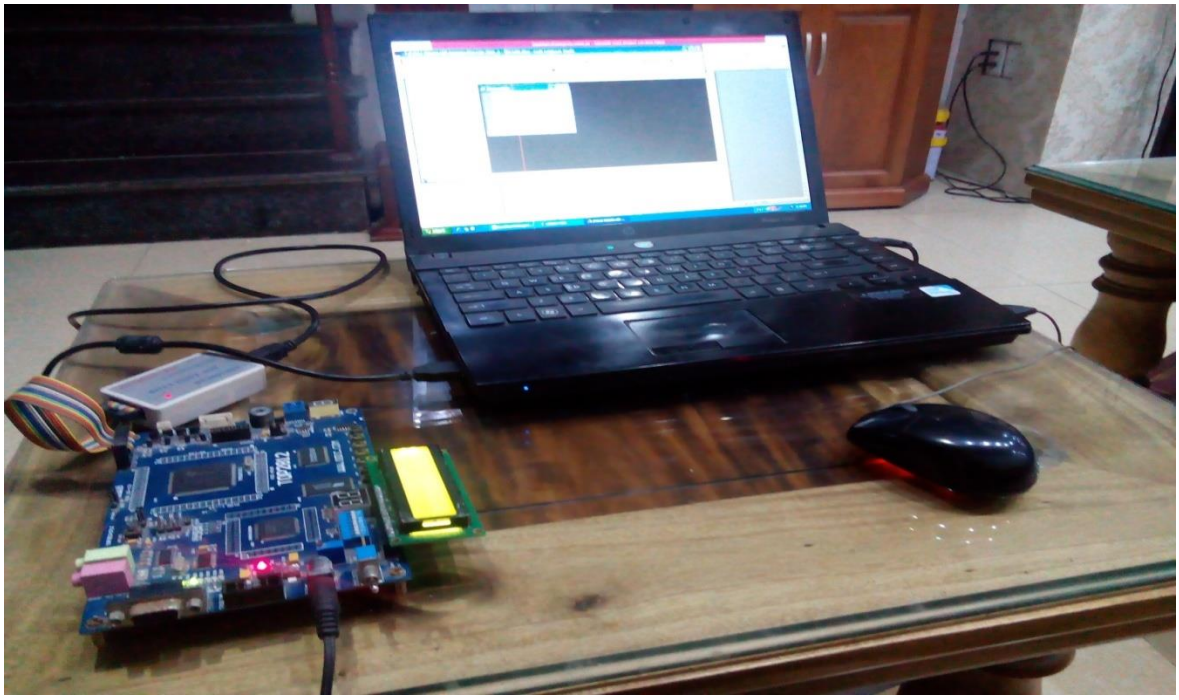


Hình 3.43. Khởi tạo phần mềm Code Composer Setup, thiết lập cấu hình hệ thống

Bước 7: Khởi tạo phần mềm Code Composer Studio và kết nối với Board nạp code cho TMS320F2812 sau khi đã nối cổng giao tiếp DSP JTAG của Board nạp code với cổng USB của máy tính thông qua thiết bị XDS100USB DSP EMULATOR và cấp nguồn cho Board nạp code.

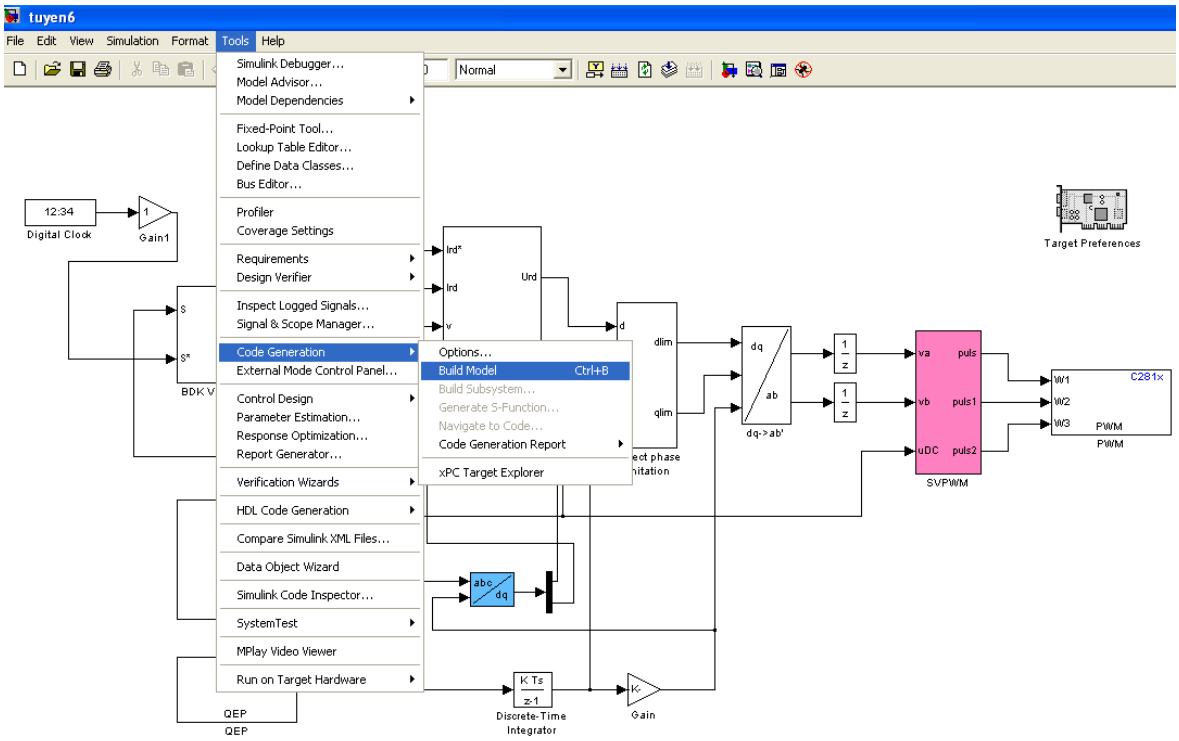


Hình 3.44. Thực hiện lệch kết nối với Board nạp code cho TMS320F2812



Hình 3.45. Kết nối máy tính PC với Board nạp code cho TMS320F2812

Bước 8: Thực hiện lệnh tạo code cho mô hình, chương trình sẽ tự chuyển từ mô hình Simulink sang mã chương trình ngôn ngữ C cho TMS320F2812, dịch mã nguồn và tự động nạp vào bộ nhớ của TMS320F2812.

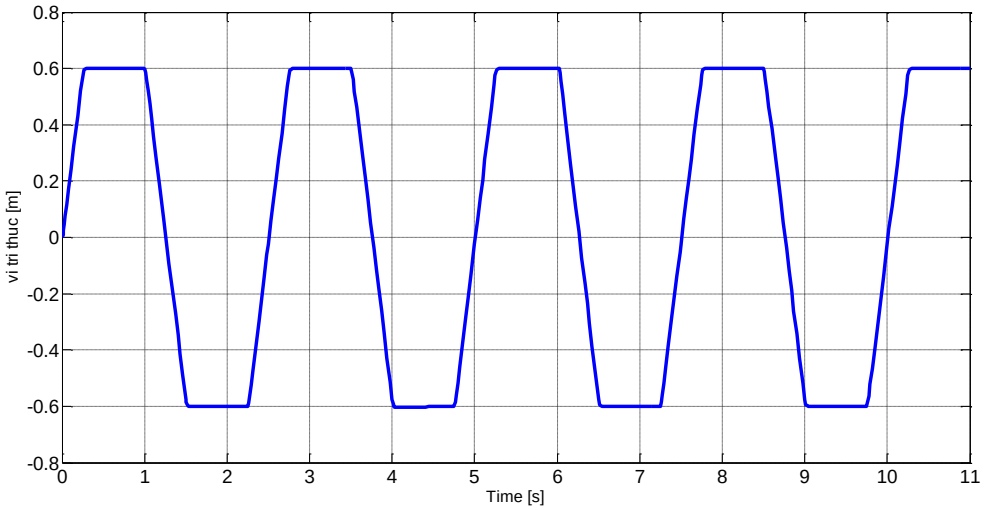


Hình 3.46. Thực hiện lệnh tạo code cho mô hình

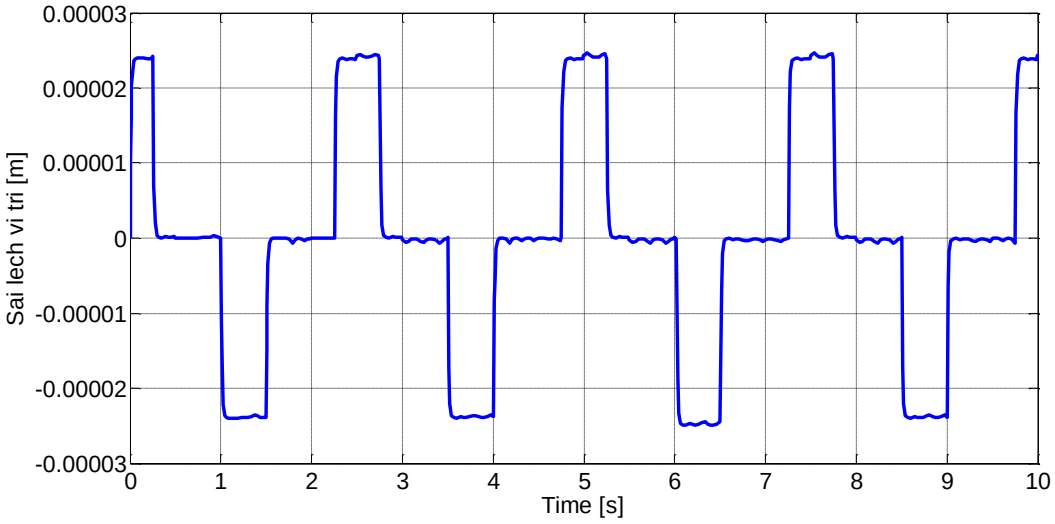
CHƯƠNG 4. KẾT QUẢ THỬ NGHIỆM, ĐÁNH GIÁ, KẾT LUẬN VÀ KIẾN NGHỊ

4.1. Kết quả thử nghiệm

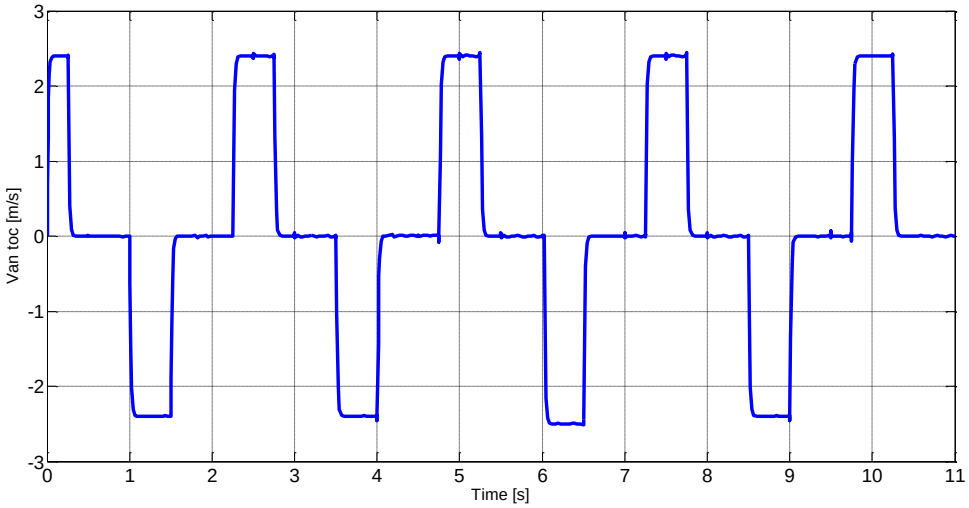
4.1.1. Thử nghiệm ở tốc độ cao



a) Vị trí đặt



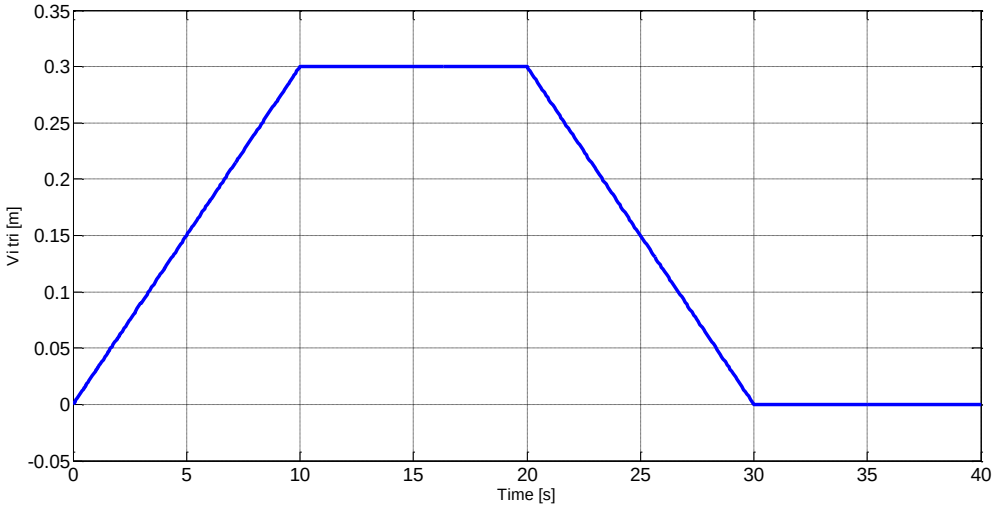
b) Sai lệch vị trí



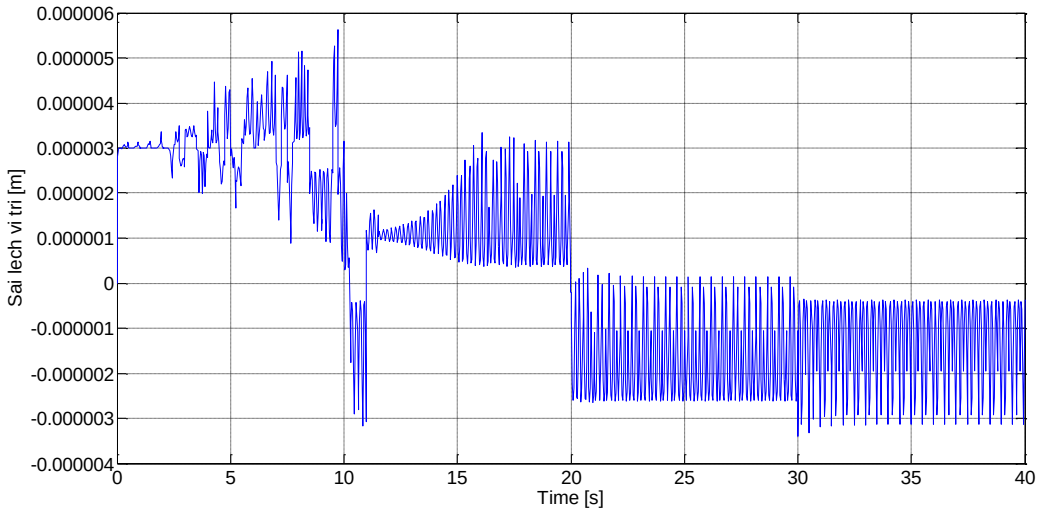
c). Đáp ứng vận tốc

Hình 4.1. Thử nghiệm ở tốc độ cao

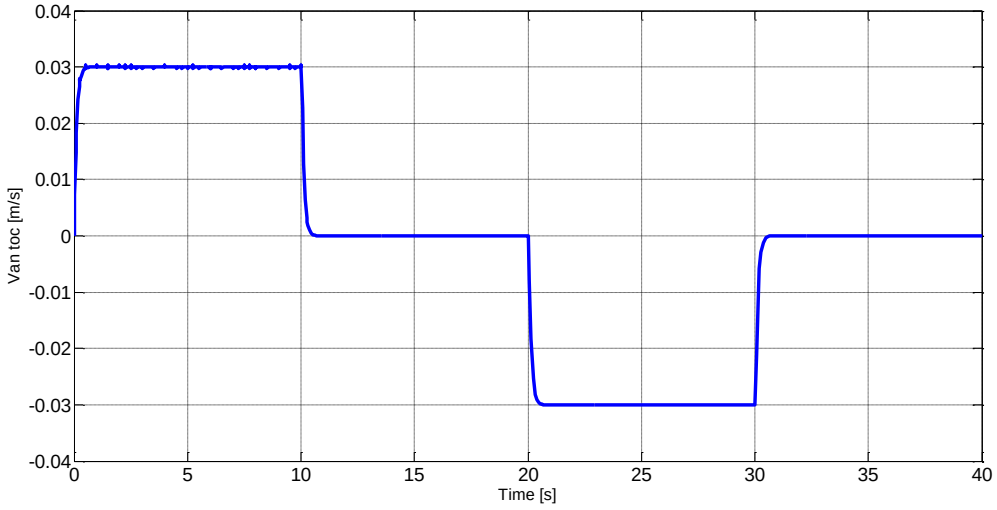
4.1.2. Thử nghiệm ở tốc độ thấp



a) Vị trí đặt



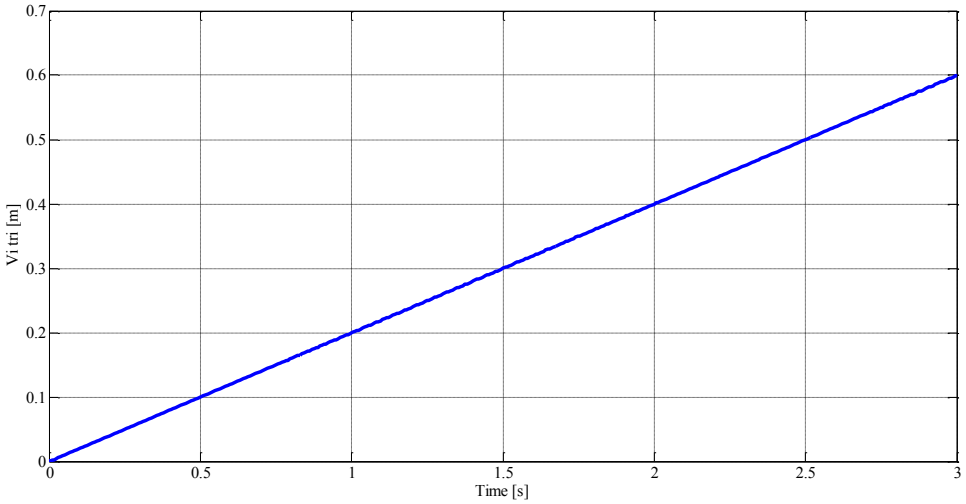
b) Sai lệch vị trí



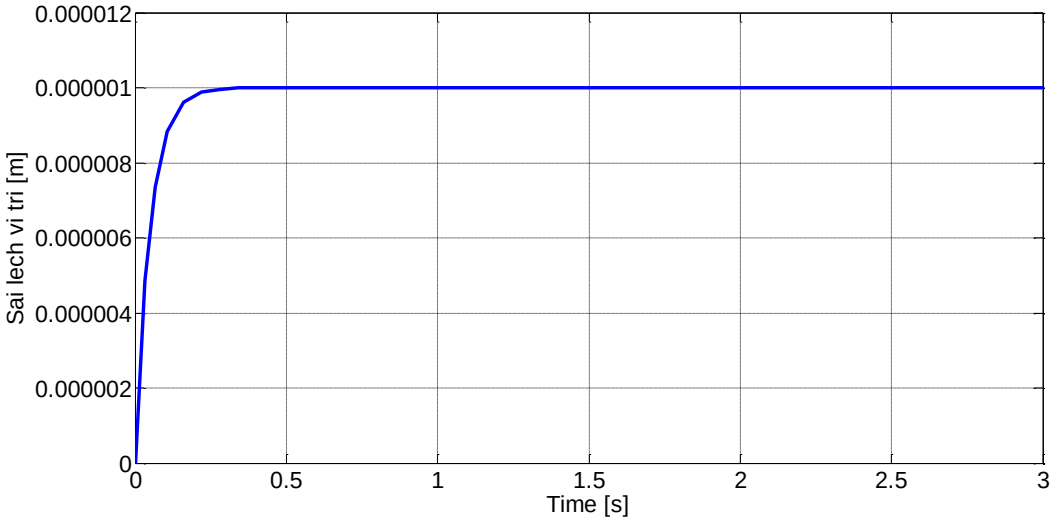
c)Đáp ứng vận tốc

Hình 4.2. Thử nghiệm ở tốc độ thấp

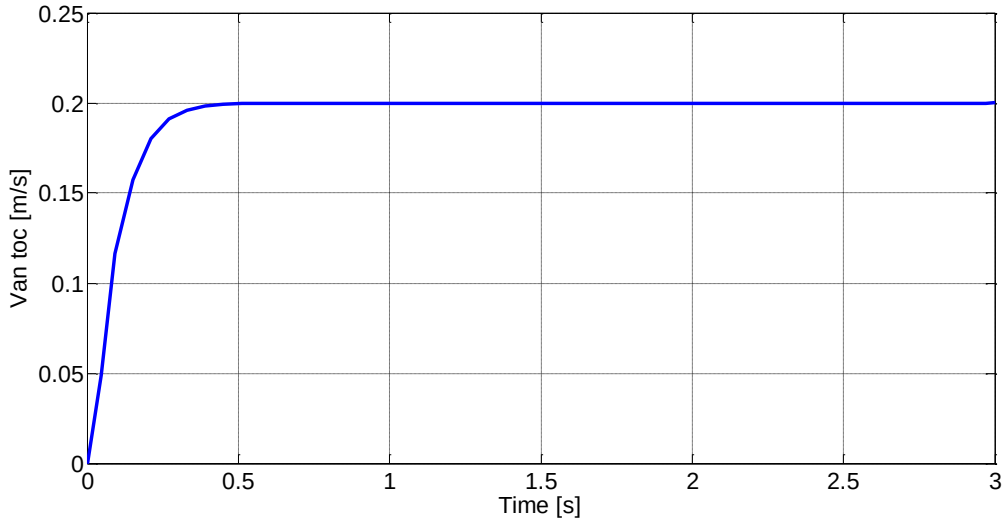
4.1.3. Thử nghiệm với quỹ đạo đặt $s^*=0,2t$



a) Quỹ đạo thực



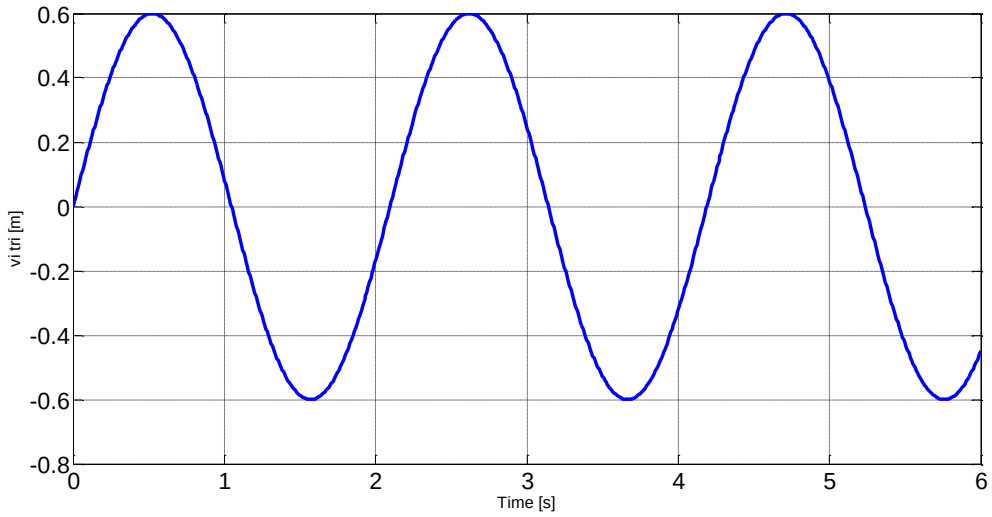
b) Sai lệch quỹ đạo



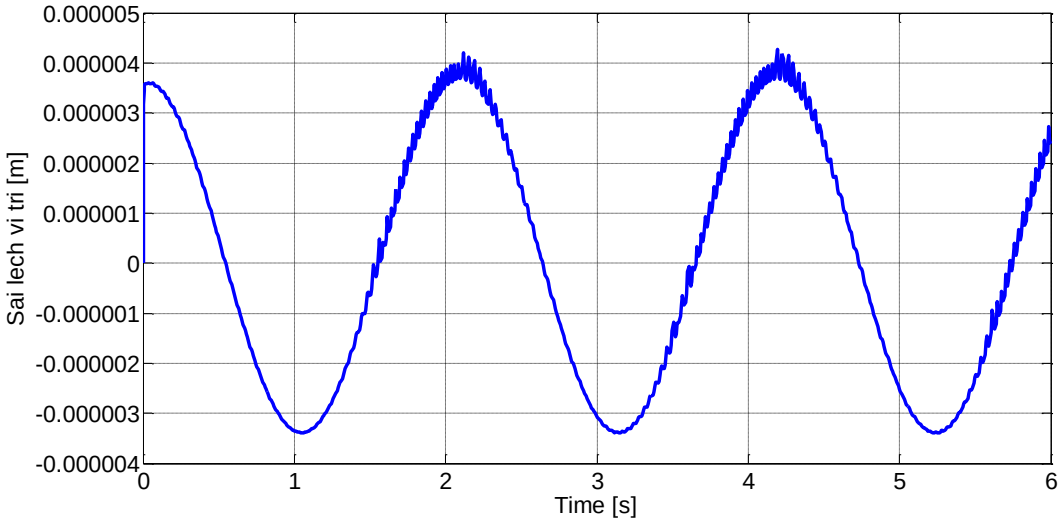
c) Đáp ứng vận tốc

Hình 4.3. Thử nghiệm với quỹ đạo đặt $s^{*}=0.2t$

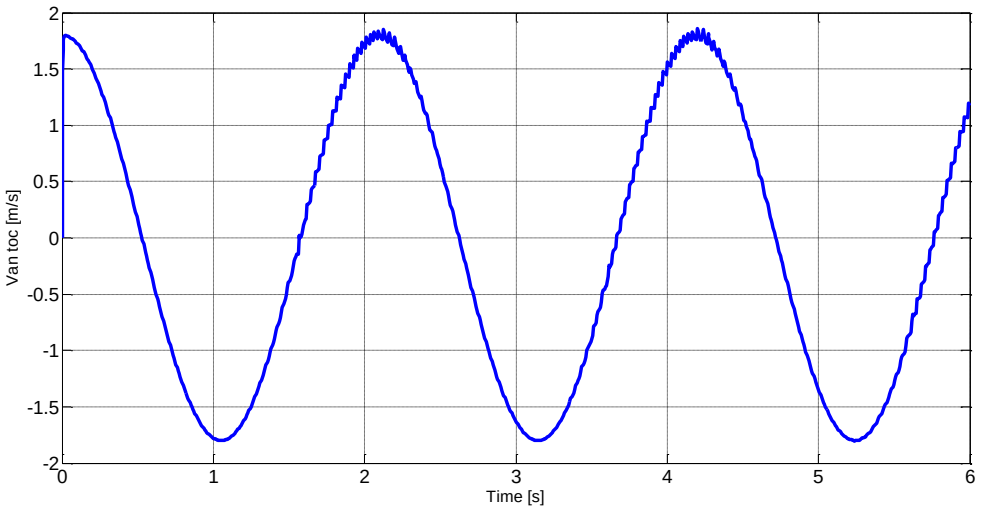
4.1.4. Thử nghiệm với quỹ đạo đặt là hình sin x=0,6sin3t



a) Quỹ đạo thực



b) Sai lệch quỹ đạo



c) Đáp ứng vận tốc

Hình 4.4. Thử nghiệm với quỹ đạo đặt hình sin: S^{*} = 0.6sin3t

4.2. Đánh giá, kết luận và kiến nghị

4.2.1. Đánh giá khả năng làm việc của hệ thống.

Hệ thống đã được thử nghiệm với các quỹ đạo đặt và vận tốc chuyển động nhanh chậm khác nhau để đánh giá mức độ bám vị trí của hệ thống. Cụu thể:

-Khi thử nghiệm hệ thống từ vị trí 0 đế vị trí 0.6m, dừng, rồi đảo chiều chuyển động đến vị trí -0.6m với tốc độ 2,4 m/s, kết quả bám vị trí đặt với sai số là

$2,4.10^{-5}$ m. Kết quả này chứng tỏ khả năng bám vị trí chính xác của hệ thống ở tốc độ cao.

- Khi thử nghiệm hệ thống từ vị trí 0 đến vị trí 0.3m, dừng, rồi đảo chiều chuyển động trở lại vi trí ban đầu (vị trí 0) với tốc độ thấp v=0.03 m/s, kết quả bám vị trí đặt với sai số là $5,5.10^{-6}$ m. Kết quả này chứng tỏ khả năng bám vị trí chính xác của hệ thống ở tốc độ thấp với sai số nhỏ hơn ở tốc độ cao.

- Khi thử nghiệm hệ thống với quỹ đạo đặt là x =0,2.t, chuyển động từ vị trí 0 ban đầu đến vị trí 0,6m, hệ thống đã bám chính xác quỹ đạo đặt với sai số 1.10^{-6}m, với vận tốc không đổi là 0,2 m/s. Kết quả này cũng khẳng định khả năng bám vị trí chính xác của hệ thống theo quỹ đạo đặt là đường thẳng.

- Khi thử nghiệm với quỹ đạo đặt là hình sin x=0,6sin3t, hệ thống đã bám theo quỹ đạo đặt với sai số vị trí là 4.10^{-6} m . Kết quả này cũng khẳng định khả năng bám vị trí chính xác của hệ thống theo quỹ đạo đặt là hình sin.

Các thử nghiệm trên đã khẳng định khả năng bám vị trí chính xác của hệ thống ở các quỹ đạo đặt khác nhau.

4.2.2. Kết luận

Với mục tiêu nghiên cứu thiết kế và chế tạo hệ thống xác định chính xác vị trí của động cơ tuyến tính trong các hệ thống chuyển động thẳng , đề tài đã đạt được những kết quả sau:

- Thết kế và chế tạo phần cứng của hệ thống trên cơ sở sử dụng DSP TMS320F2812.

- Đã đề xuất phương pháp điều khiển phi tuyến Backstepping và bộ điều khiển PID mờ để thiết kế bộ điều khiển cho hệ thống.

- Thiết kế được phần mềm điều khiển hệ thống trên cơ sở áp dụng phần mềm MatLab/Simulink và code composer Studio để tạo code nạp vào DSP TMS 320 F2812.

Các kết quả của đề tài đã được công bố thông qua 01 bài báo đăng tại tạp chí khoa học và công nghệ đại học Thái Nguyên, hướng dẫn thành công 02 học viên cao học tại trường đại học Kỹ thuật Công nghiệp Thái Nguyên.

4.2.3. Kiến nghị

Triển khai nhân rộng với các động cơ tuyến tính khác, như đông cơ tuyến tính không đồng bộ, động cơ tuyến tính kiểu động cơ bước với các công suất khác nhau và áp dụng các hệ thống đó vào thực tế sản xuất.

Tiếp tục phát triển đề tài theo hướng nghiên cứu và áp dụng các phương pháp điều khiển mới để nâng cao hơn nữa độ chính xác bám vị trí của hệ thống.

Tiếp tục nghiên cứu về lý thuyết hiệu ứng đầu cuối (end effect) để bổ sung vào cấu trúc điều khiển , qua đó nâng cao hơn nữa chất lượng của hệ thống và nâng cao khă năng áp dụng của hệ thống trong các máy công nghiệp, robot yêu câud độ chính xác cao trong điều khiển vị trí.

TÀI LIỆU THAM KHẢO

I.Tiếng Việt

1. Phạm Lê Chi, Nguyễn Quang Tuấn, Nguyễn Phùng Quang (2005)**, "**Cấu trúc tách kênh trực tiếp điều khiển hệ thống máy phát điện không đồng bộ nguồn kép", *Chuyên san Kỹ thuật điều khiển tự động* , 3 (6), pp. 28-35.

2. Đào Phương Nam (2012), *Nâng cao chất lượng các hệ chuyển động thẳng bằng cách sử dụng hệ truyền động động cơ tuyến tính*, Luận án tiến sỹ tự động hoá xí nghiệp công nghiệp, Đại học Bách khoa Hà nội.

3. Đào Phương Nam, Nguyễn Phùng Quang (2011), "Xác định vị trí đỉnh cực ban đầu của động cơ tuyến tính loại đồng bộ kích thích vĩnh cửu sử dụng phương pháp điều khiển lực đẩy", *Hội nghị Điều khiển và Tự động hóa toàn quốc lần thứ nhất- VCCA- 201*,1(2), pp.55-60.

4. Nguyễn Doãn Phước, Phan Xuân Minh, Hán Thành Trung (2003), *Lý thuyết điều khiển phi tuyến*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.

5. Nguyễn Phùng Quang (2016), *Điều khiển vector truyền động điện xoay chiều ba pha*, Nhà xuất bản Bách khoa Hà Nội.

6. Nguyễn Phùng Quang (2002), "Động cơ tuyến tính: Đối tượng công nghệ bị bỏ quên?". *Tạp chí Tự động hóa ngày nay*, 8(11), tr. 26 – 29.

7. Cao Xuân Tuyển (2008), *Tổng hợp các thuật toán phi tuyến trên cơ sở phương pháp Backstepping để điều khiển máy điện dị bộ nguần kép trong hệ thống máy điện sức gió,* Luận án tiến sĩ kỹ thuật, Trường đại học bách khoa Hà Nội.

8. Trương Minh Tuấn (2013), *Nghiên cứu cải thiện đặc tính lực động cơ không đồng bộ ba pha tuyến tính*, Luận án tiến sỹ Kỹ thuật điện, Đại học Bách khoa Hà Nội.

II. Tiếng Anh

9. Chin – I Huang, Li – Chen Fu (2002), "Adaptive Backstepping Speed/Position Control with Friction Compensation for Linear Induction Motor". *Proceeding of the 41st IEEE Conference on Decision and Control*, USA, pp. 474 – 479.

10. Do. Huyn Jang, Duck – Yong Yoon (2003),” Space – Vector PWM Technique for Two – Phase Inverter – Fed Two – Phase Induction Motor”, *IEEE Transactions on Industry Application*, 39(2), pp. 542 – 549.

11. Fang Zeng Peng, Alan Joseph, Jin Wang, Lihua Chen, Zhiguo Pan, Eduardo Ortiz – Rivera, Yi Huang (2005), “Z – Source Inverter for Motor Drives”, *IEEE Transactions of Power Electronics*, 20(4), pp. 857– 863.

12. Ghislain Remy, Pierre-Jean Barre, Philippe Degobert, Jean-Paul Hautier (2006), *Application of the Causal Ordering Graph to Synchronous Motors, Part I: Model Description*, Laboratory of Power Electronics and Electrical Engineering of Lille (L2EP) Technological Research Team (ERT), The University of Science and Technology of Lille

13. Ghislain Remy, Pierre-Jean Barre, Jean-Paul Hautier (2006), *Application of the Causal Ordering Graph to Synchronous Motors, Part II: Control Design*, Laboratory of Power Electronics and Electrical Engineering of Lille (L2EP) Technological Research Team (ERT), The University of Science and Technology of Lille

14. Ghislain Remy, Jeng Jia, Pierre-Jean Barre, Philippe Degobert, Jean- Paul Hautier (2006), *Non-Sinusoidal Electromotive Force Compensation of the PMLSM with Multiple-Frequency Resonant Controller*, Laboratory of Power Electronics and Electrical Engineering of Lille (L2EP) Technological Research Team (ERT), The University of Science and Technology of Lille

15. G. Kang, K. Nam (2005), “Field – Oriented Control scheme for linear induction motor with the end – effect”, *IEE Proc.-Electr. Power Appl*, 152(6), pp. 1565 – 1572.

16. Henk Polinder, Johannes G. Slootweg, Martin J. Hoeijmakers, John C. Compter (2003), “Modeling of a Linear PM Machine including Magnetic Saturation and End Effects: Maximum Force – to – Current Ration”, *IEEE Transactions on Industry Applications*, 39 (6), pp. 1681 – 1688.

17. Jeffrey W. Moscrop (2008), *Modelling, analysis and control of linear feed axes in precision machine tools*, PhD Thesis, University of Wollongong

18. Jacek F.Gireas, Zbignew J.Piech (2000), *Synchronous motor transportation and automation*, CRC Press

19. Jun – Koo Kang, Seung – Ki Sul (1999), "New Direct Torque Control of Induction Motor for Minimum Torque Ripple and Constant Switching frequency", *IEEE Transactions on Industry Applications*, 35 (5), pp. 1076 – 1082.

20. Jeng Jia (2005), *High – Perfomance Control of the permanent magnet synchronous motor using self – tuning multiply – frequenecy resonant Controllers*, PhD Thesis, University of Science and Technology of Lille.

21. Jul – Ki Seok, Jong – Kun Lee, Dong – Choon Lee (2006), "Sensorless Speed Control of Nonsalient Permanent Magnet Synchronous Motor Using Rotor – Position – Tracking PI Controller", *IEEE Transactions on Industrial Electronics*, 53 (2), pp.399 – 405.

22. Jean Lévine (2009), *Analysis and Control of Nonlinear Systems – A Flatness-based Approach*, Springer Dordrect Heidelberg London New York.

23. John Chiasson (2005), *Modeling and high perfomance Control of Electric Machines*, A John Wiley & Sons, Inc., Publication.

24. M. A. Jabbar, Ashwin M. Khambadkone, Zhang Yanfeng (2004), "Space – Vector Modulation in a Two Phase Induction Motor Drive for Constant Power Operation". *IEEE Transactions on Industry Application*, 51 (5), pp. 1081 – 1088.

25. Mariusz Malinowski (2001), *Sensorless Control Strategies Three –Phase PWM Rectifiers*, PhD Thesis, Warsaw University of Technology.

26. Nasar S.A, Boldea (1987), *Linear Electric Motors.* Prentice – Hall, Inc., Englewood Cliffs, New Jersey.

27. Oskar Wallmark (2006), *Control of Permanent Magnet Synchronous Motor with Non – sinusoidal Flux Density Distribution*, MSc Thesis, Chalmers University of Technology, Göterborg, Sweden.

28. Paven Ponomarev (2009), *Control of Permanent Magnet Linear Synchronous Motor in Motion Control Applications*, MSc Thesis, Lappenranta University of Technology

29. Peter Vas (1990), *Vector Control of AC Machines*, Clarendo Press Oxford.

30. Quang N.P., Dittrich J.A., (2008) ,*Vector Control of Three-Phase AC Machines – System Development in the Practice*, Springer-Verlag Berlin Heidelberg.

31. V. Nedic, T.A. Lipo (2006), "Low – cost Current – Fed PMSM Drive System with Sinusoidal Input Current", *IEEE Transactions on Industry Application*, 42 (3), pp. 753 – 762.

32. Rolf hellinger and peter mnich (2009), "Linear motor-powered transportation: history, present status and future outlook", *proceedings of the IEEE*, 97(11), pp. 20-25.

33. Yuan – Rui Chen, Jie Wu, Nobert Cheung (2004), " Lyapunov's Stability Theory – Based Model Reference Adaptive Control for Permanent Magnet Linear Motor Drives", *Proc of Power Electronics Systems and Application*, 97(11), pp. 260 – 266.

34. Yuichiro Nozaki, Terufumi Yamaguchi, Takafumi Koseki (2003), "A Equivalent Circuit Model to Assist Vector Control of a Linear Induction Motor for Urban Transportation System Considering End – Effect", *International Symposium on Speed – up and Service Technology for Railway and Maglev Systems*, 100(9), pp. 26 – 31.

PHỤ LỤC

A) THAM SỐ THỬ NGHIỆM HỆ THỐNG

```
% Machine parameters
Rr = 0.0264;
Rs = 0.0107;
Lm = 0.0163;
Lrs = 0.0005;
Lss = 0.0003;
Lr = Lrs+Lm;
Ls = Lss +Lm;
Ts = Ls/Rs;
Tr = Lr/Rr;
zp = 2;
J  = 150;
Lrd = Lr;
Lrq = Lr;
Psip = 10;
k2 = 1;
k1 = 1;
Trq = Lrq/Rr;
Trd = Lrd/Rr;
% Auxilary variables
sm  = 1-Lm*Lm/Ls/Lr;
a   = 1/sm/Tr + (1-sm)/sm/Ts;
Tsm = 1/a;
b   = (1-sm)/sm;
c   = 1/(sm*Lr);
```

```
d   = b/Lm;
e   = b/Ts;
Td = Ld/Rd;
sigma = 1-Lm*Lm/Ls/Lr;
% Control parameters
fc = 2500;
Vdc0 = 1000;
T = 1/fc;
fpuls = 5000;
Tpuls = 1/fpuls;
pi = 3.14;
to = 0.02;
f = 1;
kI1=2;
kI2 = 2;
```

B) CODE CHƯƠNG TRÌNH ĐIỀU KHIỂN

```
// FILE:          DSP281x_Adc.c
#include "DSP281x_Device.h"     // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"   // DSP281x Examples Include File
#define ADC_usDELAY  8000L
#define ADC_usDELAY2 20L
void InitAdc(void)
{
        extern void DSP28x_usDelay(Uint32 Count);
```

```
        AdcRegs.ADCTRL3.bit.ADCBGRFDN = 0x3;
        DELAY_US(ADC_usDELAY);          AdcRegs.ADCTRL3.bit.ADCPWDN
= 1;          // Power up rest of ADC
        DELAY_US(ADC_usDELAY2);         // Delay after powering up ADC
}
```

```
// FILE:  DSP281x_CodeStartBranch.asm

WD_DISABLE      .set 1
                .ref _c_int00

     .sect "codestart"
code_start:
     .if WD_DISABLE == 1
         LB wd_disable
     .else
         LB _c_int00
     .endif

     .if WD_DISABLE == 1

     .text
wd_disable:
     SETC OBJMODE
     EALLOW
     MOVZ DP, #7029h>>6  ;Set data page for WDCR register
     MOV @7029h, #0068h  ;Set WDDIS bit in WDCR to disable WD
     EDIS
     LB _c_int00
     .endif
      .end

// FILE:  DSP281x_CpuTimers.c

#include "DSP281x_Device.h"     // DSP281x Headerfile Include
File
#include "DSP281x_Examples.h"
struct CPUTIMER_VARS CpuTimer0;
struct CPUTIMER_VARS CpuTimer1;
struct CPUTIMER_VARS CpuTimer2;

void InitCpuTimers(void)
{
      CpuTimer0.RegsAddr = &CpuTimer0Regs;
      CpuTimer0Regs.PRD.all  = 0xFFFFFFFF;
      CpuTimer0Regs.TPR.all  = 0;
```

```
    CpuTimer0Regs.TPRH.all = 0;
    CpuTimer0Regs.TCR.bit.TSS = 1;
    CpuTimer0Regs.TCR.bit.TRB = 1;
    CpuTimer0.InterruptCount = 0;
    CpuTimer1.RegsAddr = &CpuTimer1Regs;
    CpuTimer2.RegsAddr = &CpuTimer2Regs;
    CpuTimer1Regs.PRD.all  = 0xFFFFFFFF;
    CpuTimer2Regs.PRD.all  = 0xFFFFFFFF;
    CpuTimer1Regs.TCR.bit.TSS = 1;
    CpuTimer2Regs.TCR.bit.TSS = 1;
    CpuTimer1Regs.TCR.bit.TRB = 1;
    CpuTimer2Regs.TCR.bit.TRB = 1;
    CpuTimer1.InterruptCount = 0;
    CpuTimer2.InterruptCount = 0;
}

void ConfigCpuTimer(struct CPUTIMER_VARS *Timer, float Freq,
float Period)
{
    Uint32    temp;
    Timer->CPUFreqInMHz = Freq;
    Timer->PeriodInUSec = Period;
    temp = (long) (Freq * Period);
    Timer->RegsAddr->PRD.all = temp;
    Timer->RegsAddr->TPR.all  = 0;
    Timer->RegsAddr->TPRH.all  = 0;
    Timer->RegsAddr->TCR.bit.TSS = 1;
    Timer->RegsAddr->TCR.bit.TRB = 1;
    Timer->RegsAddr->TCR.bit.SOFT = 1;
    Timer->RegsAddr->TCR.bit.FREE = 1;
    Timer->RegsAddr->TCR.bit.TIE = 1;
    Timer->InterruptCount = 0;
}


// FILE:  DSP281x_GlobalVariableDefs.c
#include "DSP281x_Device.h"
#ifdef __cplusplus
#pragma DATA_SECTION("AdcRegsFile")
#else
#pragma DATA_SECTION(AdcRegs,"AdcRegsFile");
#endif
volatile struct ADC_REGS AdcRegs;

#ifdef __cplusplus
#pragma DATA_SECTION("CpuTimer0RegsFile")
#else
#pragma DATA_SECTION(CpuTimer0Regs,"CpuTimer0RegsFile");
#endif
volatile struct CPUTIMER_REGS CpuTimer0Regs;
#pragma DATA_SECTION(CpuTimer1Regs,"CpuTimer1RegsFile");
```

```
volatile struct CPUTIMER_REGS CpuTimer1Regs;
#pragma DATA_SECTION(CpuTimer2Regs,"CpuTimer2RegsFile");
volatile struct CPUTIMER_REGS CpuTimer2Regs;
#ifdef __cplusplus
#pragma DATA_SECTION("ECanaRegsFile")
#else
#pragma DATA_SECTION(ECanaRegs,"ECanaRegsFile");
#endif
volatile struct ECAN_REGS ECanaRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("ECanaMboxesFile")
#else
#pragma DATA_SECTION(ECanaMboxes,"ECanaMboxesFile");
#endif
volatile struct ECAN_MBOXES ECanaMboxes;
#ifdef __cplusplus
#pragma DATA_SECTION("ECanaLAMRegsFile")
#else
#pragma DATA_SECTION(ECanaLAMRegs,"ECanaLAMRegsFile");
#endif
volatile struct LAM_REGS ECanaLAMRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("ECanaMOTSRegsFile")
#else
#pragma DATA_SECTION(ECanaMOTSRegs,"ECanaMOTSRegsFile");
#endif
volatile struct MOTS_REGS ECanaMOTSRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("ECanaMOTORegsFile")
#else
#pragma DATA_SECTION(ECanaMOTORegs,"ECanaMOTORegsFile");
#endif
volatile struct MOTO_REGS ECanaMOTORegs;
#ifdef __cplusplus
#pragma DATA_SECTION("EvaRegsFile")
#else
#pragma DATA_SECTION(EvaRegs,"EvaRegsFile");
#endif
volatile struct EVA_REGS EvaRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("EvbRegsFile")
#else
#pragma DATA_SECTION(EvbRegs,"EvbRegsFile");
#endif
volatile struct EVB_REGS EvbRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("GpioDataRegsFile")
#else
#pragma DATA_SECTION(GpioDataRegs,"GpioDataRegsFile");
#endif
volatile struct GPIO_DATA_REGS GpioDataRegs;
```

```
#ifdef __cplusplus
#pragma DATA_SECTION("GpioMuxRegsFile")
#else
#pragma DATA_SECTION(GpioMuxRegs,"GpioMuxRegsFile");
#endif
volatile struct GPIO_MUX_REGS GpioMuxRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("McbspaRegsFile")
#else
#pragma DATA_SECTION(McbspaRegs,"McbspaRegsFile");
#endif
volatile struct MCBSP_REGS McbspaRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("PieCtrlRegsFile")
#else
#pragma DATA_SECTION(PieCtrlRegs,"PieCtrlRegsFile");
#endif
volatile struct PIE_CTRL_REGS PieCtrlRegs;
#ifdef __cplusplus
#pragma DATA_SECTION("PieVectTableFile")
#else
#pragma DATA_SECTION(PieVectTable,"PieVectTableFile");
#endif
struct PIE_VECT_TABLE PieVectTable;

#ifdef __cplusplus
#pragma DATA_SECTION("SciaRegsFile")
#else
#pragma DATA_SECTION(SciaRegs,"SciaRegsFile");
#endif
volatile struct SCI_REGS SciaRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("ScibRegsFile")
#else
#pragma DATA_SECTION(ScibRegs,"ScibRegsFile");
#endif
volatile struct SCI_REGS ScibRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("SpiaRegsFile")
#else
#pragma DATA_SECTION(SpiaRegs,"SpiaRegsFile");
#endif
volatile struct SPI_REGS SpiaRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("SysCtrlRegsFile")
```

```
#else
#pragma DATA_SECTION(SysCtrlRegs,"SysCtrlRegsFile");
#endif
volatile struct SYS_CTRL_REGS SysCtrlRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("DevEmuRegsFile")
#else
#pragma DATA_SECTION(DevEmuRegs,"DevEmuRegsFile");
#endif
volatile struct DEV_EMU_REGS DevEmuRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("CsmRegsFile")
#else
#pragma DATA_SECTION(CsmRegs,"CsmRegsFile");
#endif
volatile struct CSM_REGS CsmRegs;

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("CsmPwlFile")
#else
#pragma DATA_SECTION(CsmPwl,"CsmPwlFile");
#endif
volatile struct CSM_PWL CsmPwl;


//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("FlashRegsFile")
#else
#pragma DATA_SECTION(FlashRegs,"FlashRegsFile");
#endif
volatile struct FLASH_REGS FlashRegs;

#if DSP28_F2812
//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("XintfRegsFile")
#else
#pragma DATA_SECTION(XintfRegs,"XintfRegsFile");
#endif
volatile struct XINTF_REGS XintfRegs;
#endif

//---------------------------------------
#ifdef __cplusplus
#pragma DATA_SECTION("XIntruptRegsFile")
```

```
#else
#pragma DATA_SECTION(XIntruptRegs,"XIntruptRegsFile");
#endif
volatile struct XINTRUPT_REGS XIntruptRegs;


// The following are provided to support alternate notation
// that was used in an early version of the header files

#define ADCRegs       AdcRegs
#define CPUTimer0Regs CpuTimer0Regs
#define CPUTimer1Regs CpuTimer1Regs
#define CPUTimer2Regs CpuTimer2Regs
#define ECANARegs     ECanaRegs
#define ECANAMboxes   ECanaMboxes
#define EVARegs       EvaRegs
#define GPIODataRegs  GpioDataRegs
#define GPIOMuxRegs   GpioMuxRegs
#define MCBSPARegs    McbspaRegs
#define PIECtrlRegs   PieCtrlRegs
#define PIEVectTable  PieVectTable
#define SCIARegs      SciaRegs
#define SCIBRegs      ScibRegs
#define SYSCtrlRegs   SysCtrlRegs
#define DEVEmuRegs    DevEmuRegs
#define CSMRegs       CsmRegs
#define CSMPwl        CsmPwl
#define FLASHRegs     FlashRegs
#define XINTFRegs     XintfRegs
#define XINTRUPTRegs  XIntruptRegs
```

// FILE: DSP281x_MemCopy.c

```
#include "DSP281x_Device.h"

void MemCopy(Uint16 *SourceAddr, Uint16* SourceEndAddr, Uint16* DestAddr)

{

  while(SourceAddr < SourceEndAddr)

  {

    *DestAddr++ = *SourceAddr++;

  }

  return;
```

```
}
// FILE:        DSP281x_PieCtrl.c
#include "DSP281x_Device.h"     // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"   // DSP281x Examples Include File
void InitPieCtrl(void)
{
   DINT;
        PieCtrlRegs.PIECRTL.bit.ENPIE = 0;
        PieCtrlRegs.PIEIER1.all = 0;
        PieCtrlRegs.PIEIER2.all = 0;
        PieCtrlRegs.PIEIER3.all = 0;
        PieCtrlRegs.PIEIER4.all = 0;
        PieCtrlRegs.PIEIER5.all = 0;
        PieCtrlRegs.PIEIER6.all = 0;
        PieCtrlRegs.PIEIER7.all = 0;
        PieCtrlRegs.PIEIER8.all = 0;
        PieCtrlRegs.PIEIER9.all = 0;
        PieCtrlRegs.PIEIER10.all = 0;
        PieCtrlRegs.PIEIER11.all = 0;
        PieCtrlRegs.PIEIER12.all = 0;
        PieCtrlRegs.PIEIFR1.all = 0;
        PieCtrlRegs.PIEIFR2.all = 0;
        PieCtrlRegs.PIEIFR3.all = 0;
        PieCtrlRegs.PIEIFR4.all = 0;
        PieCtrlRegs.PIEIFR5.all = 0;
        PieCtrlRegs.PIEIFR6.all = 0;
```

```
        PieCtrlRegs.PIEIFR7.all = 0;
        PieCtrlRegs.PIEIFR8.all = 0;
        PieCtrlRegs.PIEIFR9.all = 0;
        PieCtrlRegs.PIEIFR10.all = 0;
        PieCtrlRegs.PIEIFR11.all = 0;
        PieCtrlRegs.PIEIFR12.all = 0;
}
void EnableInterrupts()
{
   PieCtrlRegs.PIECRTL.bit.ENPIE = 1;
   PieCtrlRegs.PIEACK.all = 0xFFFF;
   EINT;
}
// FILE:        DSP281x_PieVect.c
#include "DSP281x_Device.h"     // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"   // DSP281x Examples Include File
const struct PIE_VECT_TABLE PieVectTableInit = { PIE_RESERVED,
PIE_RESERVED,  PIE_RESERVED,  PIE_RESERVED,   PIE_RESERVED,
 PIE_RESERVED,   PIE_RESERVED, PIE_RESERVED,   PIE_RESERVED,
PIE_RESERVED, PIE_RESERVED,   PIE_RESERVED,  PIE_RESERVED,
INT13_ISR, INT14_ISR,  DATALOG_ISR, RTOSINT_ISR,  EMUINT_ISR,
NMI_ISR, ILLEGAL_ISR,  USER1_ISR, USER2_ISR, USER3_ISR, USER4_ISR,
USER5_ISR, USER6_ISR, USER7_ISR,   USER8_ISR,  USER9_ISR,
USER10_ISR,  USER11_ISR,  USER12_ISR, PDPINTA_ISR,  PDPINTB_ISR,
rsvd_ISR,XINT1_ISR,  XINT2_ISR,ADCINT_ISR, TINT0_ISR,  WAKEINT_ISR,
CMP1INT_ISR, CMP2INT_ISR, CMP3INT_ISR, T1PINT_ISR, T1CINT_ISR,
T1UFINT_ISR,  T1OFINT_ISR,  rsvd_ISR,T2PINT_ISR, T2CINT_ISR,
T2UFINT_ISR,  T2OFINT_ISR, CAPINT1_ISR,  CAPINT2_ISR, CAPINT3_ISR,
rsvd_ISR,CMP4INT_ISR, CMP5INT_ISR,  CMP6INT_ISR,  T3PINT_ISR,
```

```
T3CINT_ISR, T3UFINT_ISR, T3OFINT_ISR, rsvd_ISR,
T4PINT_ISR,T4CINT_ISR, T4UFINT_ISR, T4OFINT_ISR,  CAPINT4_ISR,
CAPINT5_ISR,  CAPINT6_ISR, rsvd_ISR, SPIRXINTA_ISR, SPITXINTA_ISR,
rsvd_ISR,rsvd_ISR,MRINTA_ISR, MXINTA_ISR, rsvd_ISR,rsvd_ISR,rsvd_ISR,
rsvd_ISR, rsvd_ISR, rsvd_ISR,   rsvd_ISR,  rsvd_ISR, rsvd_ISR, rsvd_ISR,
rsvd_ISR,rsvd_ISR,rsvd_ISR,rsvd_ISR,rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR,
SCIRXINTA_ISR, SCITXINTA_ISR,  SCIRXINTB_ISR,  SCITXINTB_ISR,
ECAN0INTA_ISR, ECAN1INTA_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR,
rsvd_ISR, rsvd_ISR,  rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR,
rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR, rsvd_ISR,  rsvd_ISR,
rsvd_ISR, rsvd_ISR,   rsvd_ISR, rsvd_ISR,  rsvd_ISR,    rsvd_ISR,   rsvd_ISR };

void InitPieVectTable(void)

{

        int16   i;

        Uint32 *Source = (void *) &PieVectTableInit;

        Uint32 *Dest = (void *) &PieVectTable;

        EALLOW;

        for(i=0; i < 128; i++)

                *Dest++ = *Source++;

        EDIS;

        PieCtrlRegs.PIECRTL.bit.ENPIE = 1;


}

// FILE:   DSP281x_SysCtrl.c

#include "DSP281x_Device.h"     // DSP281x Headerfile Include File

#include "DSP281x_Examples.h"   // DSP281x Examples Include File

#pragma CODE_SECTION(InitFlash, "ramfuncs");

void InitSysCtrl(void)
```

```
{
   DisableDog();
   InitPll(0xA);
   InitPeripheralClocks();
}
void InitFlash(void)
{
   EALLOW;
   FlashRegs.FOPT.bit.ENPIPE = 1;
   FlashRegs.FBANKWAIT.bit.RANDWAIT = 5;
   FlashRegs.FBANKWAIT.bit.PAGEWAIT = 5;
   FlashRegs.FSTDBYWAIT.bit.STDBYWAIT = 0x01FF;
   FlashRegs.FACTIVEWAIT.bit.ACTIVEWAIT = 0x01FF;
   EDIS;
asm(" RPT #7 || NOP");
}
void KickDog(void)
{
    EALLOW;
    SysCtrlRegs.WDKEY = 0x0055;
    SysCtrlRegs.WDKEY = 0x00AA;
    EDIS;
}
void DisableDog(void)
{
    EALLOW;
```

```
    SysCtrlRegs.WDCR= 0x0068;
    EDIS;
}
void InitPll(Uint16 val)
{
   volatile Uint16 iVol;
   if (SysCtrlRegs.PLLCR.bit.DIV != val)
   {
      EALLOW;
      SysCtrlRegs.PLLCR.bit.DIV = val;
      EDIS;
      DisableDog();

      for(iVol= 0; iVol< ( (131072/2)/12 ); iVol++)
      {
      }
   }
}

void InitPeripheralClocks(void)
{
   EALLOW;
   SysCtrlRegs.HISPCP.all = 0x0001;
   SysCtrlRegs.LOSPCP.all = 0x0002;
   SysCtrlRegs.PCLKCR.bit.EVAENCLK=1;
   SysCtrlRegs.PCLKCR.bit.EVBENCLK=1;
```

```
   SysCtrlRegs.PCLKCR.bit.SCIAENCLK=1;
   SysCtrlRegs.PCLKCR.bit.SCIBENCLK=1;
   SysCtrlRegs.PCLKCR.bit.MCBSPENCLK=1;
   SysCtrlRegs.PCLKCR.bit.SPIENCLK=1;
   SysCtrlRegs.PCLKCR.bit.ECANENCLK=1;
   SysCtrlRegs.PCLKCR.bit.ADCENCLK=1;
   EDIS;
}
#define STATUS_FAIL         0
#define STATUS_SUCCESS      1
Uint16 CsmUnlock()
{
   volatile Uint16 temp;
   EALLOW;
   CsmRegs.KEY0 = 0xFFFF;
   CsmRegs.KEY1 = 0xFFFF;
   CsmRegs.KEY2 = 0xFFFF;
   CsmRegs.KEY3 = 0xFFFF;
   CsmRegs.KEY4 = 0xFFFF;
   CsmRegs.KEY5 = 0xFFFF;
   CsmRegs.KEY6 = 0xFFFF;
   CsmRegs.KEY7 = 0xFFFF;
   EDIS;
   temp = CsmPwl.PSWD0;
   temp = CsmPwl.PSWD1;
   temp = CsmPwl.PSWD2;
```

```
   temp = CsmPwl.PSWD3;
   temp = CsmPwl.PSWD4;
   temp = CsmPwl.PSWD5;
   temp = CsmPwl.PSWD6;
   temp = CsmPwl.PSWD7;
   if (CsmRegs.CSMSCR.bit.SECURE == 0) return STATUS_SUCCESS;
   else return STATUS_FAIL;


}
// FILE:  DSP281x_usDelay.asm
        .def _DSP28x_usDelay
        .sect "ramfuncs"
         .global  __DSP28x_usDelay
_DSP28x_usDelay:
         SUB    ACC,#1
         BF     _DSP28x_usDelay,GEQ    ;; Loop if ACC >= 0
         LRETR

#include "DSP281x_Device.h"
#include "DSP281x_GlobalPrototypes.h"
#include "rtwtypes.h"
#include "tuyen2.h"
#include "tuyen2_private.h"


void rt_OneStep(void);
void config_schedulerTimer(void)
{
  InitCpuTimers();
  ConfigCpuTimer(&CpuTimer0, 150, 0.001 * 1000000);
  StartCpuTimer0();
}
```

```
void disableWatchdog(void)
{
  int *WatchdogWDCR = (void *) 0x7029;
  asm(" EALLOW ");
  *WatchdogWDCR = 0x0068;
  asm(" EDIS ");
}
void config_ADC_A(uint16_T maxConv, uint16_T adcChselSEQ1Reg, uint16_T
                adcChselSEQ2Reg, uint16_T adcChselSEQ3Reg, uint16_T
                adcChselSEQ4Reg)
{
  AdcRegs.ADCTRL1.bit.SUSMOD = 0x0;    /* Emulation suspend ignored*/
  AdcRegs.ADCTRL1.bit.ACQ_PS = 12;     /* Acquisition window size*/
  AdcRegs.ADCTRL1.bit.CPS = 0;         /* Core clock pre-scaler*/
  AdcRegs.ADCTRL1.bit.CONT_RUN = 0x0;  /* Start-Stop sequencer mode*/
  AdcRegs.ADCTRL3.bit.ADCBGRFDN = 0x3;
AdcRegs.ADCTRL3.bit.ADCCLKPS = 6;    /* Core clock divider*/
  AdcRegs.ADCTRL3.bit.SMODE_SEL = 0;   /* Sequential sampling*/
  AdcRegs.ADCMAXCONV.bit.MAX_CONV1 = maxConv;/* Number of conversions
in CONV1 when using A and "A and B" module*/
  AdcRegs.ADCCHSELSEQ1.all = adcChselSEQ1Reg;/* Channels for conversion*/
  AdcRegs.ADCCHSELSEQ2.all = adcChselSEQ2Reg;/* Channels for conversion*/
  AdcRegs.ADCTRL1.bit.SEQ_CASC = 0;    /* Dual sequencer mode*/
}
void config_PWM_A(uint16_T timerPeriod, uint16_T waveformType,
                uint16_T unit1Status, char* unit1Source, uint16_T unit1Value,
                uint16_T unit2Status, char* unit2Source, uint16_T unit2Value,
```

```
            uint16_T unit3Status, char* unit3Source, uint16_T unit3Value,
            uint16_T controlLogic,
            uint16_T enableDeadband1, uint16_T enableDeadband2, uint16_T
            enableDeadband3,
            uint16_T deadbandPrescaler, uint16_T deadbandPeriod, uint16_T
            timerToADC, uint16_T preScaler)
{
 const uint16_T DISABLED= 0;
 EvaRegs.T1PR = timerPeriod;         /* period*/
 EvaRegs.T1CNT = 0x0000;             /* counter*/
 EvaRegs.T1CON.all = 0x1042;         /* enable; compare enable; default TMODE */
 EvaRegs.T1CON.bit.TMODE = waveformType;/* adjust Timer TMODE*/
 EvaRegs.T1CON.bit.TPS = preScaler;   /* Input clock prescaler*/
 EvaRegs.GPTCONA.bit.T1TOADC = timerToADC;
 EvaRegs.CMPR1 = unit1Status ? unit1Value : DISABLED;
 EvaRegs.CMPR2 = unit2Status ? unit2Value : DISABLED;
 EvaRegs.CMPR3 = unit3Status ? unit3Value : DISABLED;

 EvaRegs.ACTRA.all = controlLogic;
 EvaRegs.DBTCONA.bit.EDBT1 = enableDeadband1;
 EvaRegs.DBTCONA.bit.EDBT2 = enableDeadband2;
 EvaRegs.DBTCONA.bit.EDBT3 = enableDeadband3;
 EvaRegs.DBTCONA.bit.DBT = deadbandPeriod;
 EvaRegs.DBTCONA.bit.DBTPS = deadbandPrescaler;
 EvaRegs.COMCONA.all = 0xA600;
}
```

```
void config_QEP_A(uint16_T initialCount, uint16_T period, uint16_T qepIndex,
                  uint16_T qepIndexQual)
{
 EvaRegs.T2CNT = initialCount;
  EvaRegs.T2CON.all = 0x1070;
  EALLOW;
  GpioMuxRegs.GPAMUX.all |= 0x0300;
  if (qepIndex == 1) {
    GpioMuxRegs.GPAMUX.all |= 0x0400;  /* I/O Mux Control Register GPAMUX:
Bit 10 Bit as QEP Index*/
    EvaRegs.EXTCONA.bit.QEPIE = 1;
    if (qepIndexQual == 1) {
      EvaRegs.EXTCONA.bit.QEPIQUAL = 1;/* CAP3_QEPI1 Index Qualification
Mode*/
    }
  }

  EDIS;
  EvaRegs.CAPCONA.all |= 0xE000;       /* Enable QEP circuit. Disable Capture
Units 1/2 and 3*/
}
interrupt void schedulerTimer_ISR(void)
{
  volatile unsigned int PIEIER1_stack_save = PieCtrlRegs.PIEIER1.all;
  PieCtrlRegs.PIEIER1.all &= ~64;
  asm(" RPT #5 || NOP");              /*wait 5 cycles        */
  IFR &= ~1;
```

```
  PieCtrlRegs.PIEACK.all = 1;
  IER |= 1;
  EINT;                         /*global interrupt enable*/
  rt_OneStep();
  DINT;
  PieCtrlRegs.PIEIER1.all = PIEIER1_stack_save;/*restore PIEIER register that was
modified*/
}
void enable_interrupts()
{
  EALLOW;
  PieVectTable.TINT0 = &schedulerTimer_ISR;/* Hook interrupt to the ISR*/
  EDIS;
  PieCtrlRegs.PIEIER1.bit.INTx7 = 1;   /* Enable TINT0 in the PIE: Group 1
interrupt 7*/
  IER |= M_INT1;                    /* Enable Global INT1 (CPU INT1)*/

  EINT;                         /* Enable Global interrupt INTM*/
  ERTM;                          /* Enable Global realtime interrupt DBGM*/
}

void disable_interrupts()
{
  IER &= M_INT1;                    /* Disable Global INT1 (CPU Interrupt Group 1) */
  DINT;                         /* Disable Global interrupt INTM*/
}
void init_board ()
```

```
{
 DisableDog();
 InitPll(10);
 InitPeripheralClocks();
 XintfRegs.XINTCNF2.bit.XTIMCLK = 0;  /* XTIMCLK=SYSCLKOUT/1*/
 XintfRegs.XINTCNF2.bit.CLKOFF = 0;   /* XCLKOUT is enabled*/
 XintfRegs.XINTCNF2.bit.CLKMODE = 0;  /* XCLKOUT = XTIMCLK*/
 while (XintfRegs.XINTCNF2.bit.WLEVEL != 0) ;/* poll the WLEVEL bit*/
 XintfRegs.XINTCNF2.bit.WRBUFF = 0;   /* No write buffering*/
 XintfRegs.XBANK.bit.BCYC = 7;        /* Add 7 cycles*/
 XintfRegs.XBANK.bit.BANK = 7;        /* select zone 7*/
 XintfRegs.XTIMING0.bit.X2TIMING = 0; /* Timing scale factor = 1*/
 XintfRegs.XTIMING0.bit.XSIZE = 3;    /* Always write as 11b*/
 XintfRegs.XTIMING0.bit.READYMODE = 1;/* XREADY is asynchronous*/
 XintfRegs.XTIMING0.bit.USEREADY = 0; /* Disable XREADY*/
 XintfRegs.XTIMING0.bit.XRDLEAD = 1;  /* Read lead time*/
 XintfRegs.XTIMING0.bit.XRDACTIVE = 2;/* Read active time*/
 XintfRegs.XTIMING0.bit.XRDTRAIL = 0; /* Read trail time*/
 XintfRegs.XTIMING0.bit.XWRLEAD = 1;  /* Write lead time*/
 XintfRegs.XTIMING0.bit.XWRACTIVE = 2;/* Write active time*/
 XintfRegs.XTIMING0.bit.XWRTRAIL = 0; /* Write trail time*/
 XintfRegs.XTIMING1.bit.X2TIMING = 0; /* Timing scale factor = 1*/
 XintfRegs.XTIMING1.bit.XSIZE = 3;    /* Always write as 11b*/
 XintfRegs.XTIMING1.bit.READYMODE = 1;/* XREADY is asynchronous*/
 XintfRegs.XTIMING1.bit.USEREADY = 0; /* Disable XREADY*/
 XintfRegs.XTIMING1.bit.XRDLEAD = 1;  /* Read lead time*/
```

```
XintfRegs.XTIMING1.bit.XRDACTIVE = 2;/* Read active time*/
XintfRegs.XTIMING1.bit.XRDTRAIL = 0; /* Read trail time*/
XintfRegs.XTIMING1.bit.XWRLEAD = 1;  /* Write lead time*/
XintfRegs.XTIMING1.bit.XWRACTIVE = 2;/* Write active time*/
XintfRegs.XTIMING1.bit.XWRTRAIL = 0; /* Write trail time*/
XintfRegs.XTIMING2.bit.X2TIMING = 0; /* Timing scale factor = 1*/
XintfRegs.XTIMING2.bit.XSIZE = 3;    /* Always write as 11b*/
XintfRegs.XTIMING2.bit.READYMODE = 1;/* XREADY is asynchronous*/
XintfRegs.XTIMING2.bit.USEREADY = 0; /* Disable XREADY*/
XintfRegs.XTIMING2.bit.XRDLEAD = 1;  /* Read lead time*/
XintfRegs.XTIMING2.bit.XRDACTIVE = 2;/* Read active time*/
XintfRegs.XTIMING2.bit.XRDTRAIL = 0; /* Read trail time*/
XintfRegs.XTIMING2.bit.XWRLEAD = 1;  /* Write lead time*/
XintfRegs.XTIMING2.bit.XWRACTIVE = 2;/* Write active time*/
XintfRegs.XTIMING2.bit.XWRTRAIL = 0; /* Write trail time*/

XintfRegs.XTIMING6.bit.X2TIMING = 0; /* Timing scale factor = 1*/
XintfRegs.XTIMING6.bit.XSIZE = 3;    /* Always write as 11b*/
XintfRegs.XTIMING6.bit.READYMODE = 1;/* XREADY is asynchronous*/
XintfRegs.XTIMING6.bit.USEREADY = 0; /* Disable XREADY*/
XintfRegs.XTIMING6.bit.XRDLEAD = 1;  /* Read lead time*/
XintfRegs.XTIMING6.bit.XRDACTIVE = 2;/* Read active time*/
XintfRegs.XTIMING6.bit.XRDTRAIL = 0; /* Read trail time*/
XintfRegs.XTIMING6.bit.XWRLEAD = 1;  /* Write lead time*/
XintfRegs.XTIMING6.bit.XWRACTIVE = 2;/* Write active time*/
XintfRegs.XTIMING6.bit.XWRTRAIL = 0; /* Write trail time*/
```

```
 XintfRegs.XTIMING7.bit.X2TIMING = 0; /* Timing scale factor = 1*/
 XintfRegs.XTIMING7.bit.XSIZE = 3;    /* Always write as 11b*/
 XintfRegs.XTIMING7.bit.READYMODE = 1;/* XREADY is asynchronous*/
 XintfRegs.XTIMING7.bit.USEREADY = 0; /* Disable XREADY*/
 XintfRegs.XTIMING7.bit.XRDLEAD = 1;  /* Read lead time*/
 XintfRegs.XTIMING7.bit.XRDACTIVE = 2;/* Read active time*/
 XintfRegs.XTIMING7.bit.XRDTRAIL = 0; /* Read trail time*/
 XintfRegs.XTIMING7.bit.XWRLEAD = 1;  /* Write lead time*/
 XintfRegs.XTIMING7.bit.XWRACTIVE = 2;/* Write active time*/
 XintfRegs.XTIMING7.bit.XWRTRAIL = 0; /* Write trail time*/
 asm(" RPT #6 || NOP");
 DINT;
 IER = 0x0000;
 IFR = 0x0000;
 InitPieCtrl();
 InitPieVectTable();
 InitCpuTimers();
 EALLOW;
 GpioMuxRegs.GPAQUAL.bit.QUALPRD = 0;
 GpioMuxRegs.GPBQUAL.bit.QUALPRD = 0;
 GpioMuxRegs.GPDQUAL.bit.QUALPRD = 0;
 GpioMuxRegs.GPEQUAL.bit.QUALPRD = 0;
 EDIS;
}
/*
 * File: rt_nonfinite.c
```

```
*/
#include "rt_nonfinite.h"
#include "rtGetNaN.h"
#include "rtGetInf.h"
real_T rtInf;
real_T rtMinusInf;
real_T rtNaN;
real32_T rtInfF;
real32_T rtMinusInfF;
real32_T rtNaNF;
void rt_InitInfAndNaN(size_t realSize)
{
  (void) (realSize);
  rtNaN = rtGetNaN();
  rtNaNF = rtGetNaNF();
  rtInf = rtGetInf();
  rtInfF = rtGetInfF();
  rtMinusInf = rtGetMinusInf();
  rtMinusInfF = rtGetMinusInfF();
}
boolean_T rtIsInf(real_T value)
{
  return (boolean_T)((value==rtInf || value==rtMinusInf) ? 1U : 0U);
}
boolean_T rtIsInfF(real32_T value)
{
```

return (boolean_T)(((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U);

}

boolean_T rtIsNaN(real_T value)

{

return (boolean_T)((value!=value) ? 1U : 0U);

}

boolean_T rtIsNaNF(real32_T value)

{

return (boolean_T)(((value!=value) ? 1U : 0U));

}

```
/* File: rtGetInf.c
 */
#include "rtGetInf.h"
#define NumBitsPerChar                 16U
real_T rtGetInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T inf = 0.0;
  if (bitsPerReal == 32U) {
    inf = rtGetInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0x7FF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    inf = tmpVal.fltVal;
  }
```

```
  return inf;
}
real32_T rtGetInfF(void)
{
  IEEESingle infF;
  infF.wordL.wordLuint = 0x7F800000U;
  return infF.wordL.wordLreal;
}
real_T rtGetMinusInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T minf = 0.0;
  if (bitsPerReal == 32U) {
    minf = rtGetMinusInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    minf = tmpVal.fltVal;
  }

  return minf;
}

real32_T rtGetMinusInfF(void)
{
  IEEESingle minfF;
  minfF.wordL.wordLuint = 0xFF800000U;
  return minfF.wordL.wordLreal;
}
```

```
/*
 * File: rtGetNaN.c
 */
#include "rtGetNaN.h"
#define NumBitsPerChar                16U
real_T rtGetNaN(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T nan = 0.0;
  if (bitsPerReal == 32U) {
    nan = rtGetNaNF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF80000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    nan = tmpVal.fltVal;
  }

  return nan;
}
real32_T rtGetNaNF(void)
{
```

IEEESingle nanF = { { 0 } };

nanF.wordL.wordLuint = 0xFFC00000U;

return nanF.wordL.wordLreal;

}

```
/*
 * File: tuyen2.c
*/

#include "tuyen2.h"
#include "tuyen2_private.h"

BlockIO_tuyen2 tuyen2_B;

D_Work_tuyen2 tuyen2_DWork;

RT_MODEL_tuyen2 tuyen2_M_;
RT_MODEL_tuyen2 *const tuyen2_M = &tuyen2_M_;

void tuyen2_IfActionSubsystem3(real_T rtu_x, real_T
*rty_Out1,
  rtP_IfActionSubsystem3_tuyen2 *localP)
{
    *rty_Out1 = (rtu_x - localP->a_Value) / (localP->b_Value
- localP->a_Value);
}

void tuyen2_IfActionSubsystem2(real_T rtu_x, real_T
*rty_Out1,
  rtP_IfActionSubsystem2_tuyen2 *localP)
{
    *rty_Out1 = 1.0 / (localP->c_Value - localP->b_Value) *
(localP->c_Value -
    rtu_x);
}

void tuyen2_IfActionSubsystem3_e(real_T rtu_x, real_T
*rty_Out1,
  rtP_IfActionSubsystem3_tuyen2_b *localP)
{
    *rty_Out1 = 1.0 / (localP->d_Value - localP->c_Value) *
(localP->d_Value -
    rtu_x);
}

real_T rt_powd_snf(real_T u0, real_T u1)
```

```
{
  real_T y;
  real_T tmp;
  real_T tmp_0;
  if (rtIsNaN(u0) || rtIsNaN(u1)) {
    y = (rtNaN);
  } else {
    tmp = fabs(u0);
    tmp_0 = fabs(u1);
    if (rtIsInf(u1)) {
      if (tmp == 1.0) {
        y = (rtNaN);
      } else if (tmp > 1.0) {
        if (u1 > 0.0) {
          y = (rtInf);
        } else {
          y = 0.0;
        }
      } else if (u1 > 0.0) {
        y = 0.0;
      } else {
        y = (rtInf);
      }
    } else if (tmp_0 == 0.0) {
      y = 1.0;
    } else if (tmp_0 == 1.0) {
      if (u1 > 0.0) {
        y = u0;
      } else {
        y = 1.0 / u0;
      }
    } else if (u1 == 2.0) {
      y = u0 * u0;
    } else if ((u1 == 0.5) && (u0 >= 0.0)) {
      y = sqrt(u0);
    } else if ((u0 < 0.0) && (u1 > floor(u1))) {
      y = (rtNaN);
    } else {
      y = pow(u0, u1);
    }
  }

  return y;
}

real_T rt_hypotd_snf(real_T u0, real_T u1)
{
  real_T y;
  real_T a;
  real_T b;
  a = fabs(u0);
```

```
  b = fabs(u1);
  if (a < b) {
    a /= b;
    y = sqrt(a * a + 1.0) * b;
  } else if (a > b) {
    b /= a;
    y = sqrt(b * b + 1.0) * a;
  } else if (rtIsNaN(b)) {
    y = b;
  } else {
    y = a * 1.4142135623730951;
  }

  return y;
}

real_T rt_atan2d_snf(real_T u0, real_T u1)
{
  real_T y;
  int16_T u;
  int16_T u_0;
  if (rtIsNaN(u0) || rtIsNaN(u1)) {
    y = (rtNaN);
  } else if (rtIsInf(u0) && rtIsInf(u1)) {
    if (u0 > 0.0) {
      u = 1;
    } else {
      u = -1;
    }

    if (u1 > 0.0) {
      u_0 = 1;
    } else {
      u_0 = -1;
    }

    y = atan2((real_T)u, (real_T)u_0);
  } else if (u1 == 0.0) {
    if (u0 > 0.0) {
      y = RT_PI / 2.0;
    } else if (u0 < 0.0) {
      y = -(RT_PI / 2.0);
    } else {
      y = 0.0;
    }
  } else {
    y = atan2(u0, u1);
  }

  return y;
}
```

```
real_T rt_roundd_snf(real_T u)
{
  real_T y;
  if (fabs(u) < 4.503599627370496E+15) {
    if (u >= 0.5) {
      y = floor(u + 0.5);
    } else if (u > -0.5) {
      y = -0.0;
    } else {
      y = ceil(u - 0.5);
    }
  } else {
    y = u;
  }

  return y;
}

real_T rt_modd_snf(real_T u0, real_T u1)
{
  real_T y;
  real_T tmp;
  if (u1 == 0.0) {
    y = u0;
  } else if (!((!rtIsNaN(u0)) && (!rtIsInf(u0)) &&
((!rtIsNaN(u1)) && (!rtIsInf
                    (u1))))) {
    y = (rtNaN);
  } else {
    tmp = u0 / u1;
    if (u1 <= floor(u1)) {
      y = u0 - floor(tmp) * u1;
    } else if (fabs(tmp - rt_roundd_snf(tmp)) <= DBL_EPSILON
* fabs(tmp)) {
      y = 0.0;
    } else {
      y = (tmp - floor(tmp)) * u1;
    }
  }

  return y;
}

/* Model step function */
void tuyen2_step(void)
{
  /* local block i/o variables */
  real_T rtb_Gain;
  real_T rtb_Merge;
  real_T rtb_Merge_b;
```

```
  real_T rtb_Gain1_m;
  real_T rtb_Merge_f;
  real_T rtb_Merge_bc;
  real_T rtb_Merge_k;
  real_T rtb_Merge_g;
  real_T rtb_Merge_bu;
  real_T rtb_Merge_m;
  real_T rtb_Fcn_l;
  real_T rtb_Gain_h;
  real_T rtb_DirectLookUpTablenD;
  real_T rtb_LrdLrq2pito;
  real_T rtb_LrqLrd2pito;
  real_T rtb_PsipLrq2pito;
  real_T rtb_Fcn1_g;
  real_T rtb_Trq;
  real_T rtb_TSamp;
  real_T rtb_Sum1;
  real_T rtb_Weighting;
  real_T rtb_Weighting_o;
  real_T rtb_Product4;
  real_T rtb_Product3;
  real_T rtb_Gain4;
  real_T rtb_Sum1_e;
  real_T rtb_Diff;
  real_T rtb_Sum_oq;
  int16_T i;
  real_T rtb_UnitDelay2_idx;

    tuyen2_B.UnitDelay1 = tuyen2_DWork.UnitDelay1_DSTATE;

    tuyen2_B.UnitDelay2 = tuyen2_DWork.UnitDelay2_DSTATE;

    rtb_Weighting_o =
rt_powd_snf(tuyen2_DWork.UnitDelay1_DSTATE, 2.0) +
    rt_powd_snf(tuyen2_DWork.UnitDelay2_DSTATE, 2.0);
  if (rtb_Weighting_o < 0.0) {
    rtb_Weighting_o = -sqrt(-rtb_Weighting_o);
  } else {
    rtb_Weighting_o = sqrt(rtb_Weighting_o);
  }

    if ((real_T)(fabs(tuyen2_DWork.UnitDelay1_DSTATE) <
rtb_Weighting_o / 2.0) >=
      tuyen2_P.Switch_Threshold) {
    rtb_Gain_h = tuyen2_P.Constant_Value;
  } else {
    rtb_Gain_h = (real_T)(tuyen2_DWork.UnitDelay1_DSTATE <=
0.0) +
      tuyen2_P.Constant1_Value;
  }
```

```
    if (rtb_Gain_h >= 2.0) {
    rtb_Gain_h = 2.0;
  } else {
    if (rtb_Gain_h <= 0.0) {
      rtb_Gain_h = 0.0;
    }
  }

  rtb_Sum1_e = floor(rtb_Gain_h);
  if (rtIsNaN(rtb_Sum1_e) || rtIsInf(rtb_Sum1_e)) {
    rtb_Sum1_e = 0.0;
  } else {
    rtb_Sum1_e = fmod(rtb_Sum1_e, 4.294967296E+9);
  }

  if ((real_T)(tuyen2_DWork.UnitDelay2_DSTATE <= 0.0) >= 1.0)
{
    rtb_UnitDelay2_idx = 1.0;
  } else {
    rtb_UnitDelay2_idx = 0.0;
  }

    switch
((int16_T)tuyen2_P.DirectLookUpTablenD_table[((uint32_T)rtb_S
um1_e <<
           1U) +
(uint32_T)(int16_T)fmod((real_T)(int16_T)rtb_UnitDelay2_idx,
           4.294967296E+9)]) {
   case 1:
    for (i = 0; i < 9; i++) { tuyen2_B.sectorselector[i] =
tuyen2_P.sector1_Value[i];
    }break;
   case 2:
    for (i = 0; i < 9; i++) {tuyen2_B.sectorselector[i] =
tuyen2_P.sector2_Value[i];
    }break;
   case 3:
    for (i = 0; i < 9; i++) {
tuyen2_B.sectorselector[i] = tuyen2_P.sector3_Value[i];
    }break;
   case 4:
    for (i = 0; i < 9; i++) {
      tuyen2_B.sectorselector[i] = tuyen2_P.sector4_Value[i];
    }break;
   case 5:
    for (i = 0; i < 9; i++) {
      tuyen2_B.sectorselector[i] = tuyen2_P.sector5_Value[i];
    }break;
   default:
    for (i = 0; i < 9; i++) {
      tuyen2_B.sectorselector[i] = tuyen2_P.sector6_Value[i];
```

```
    } break;
  }

    {
    AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;/* Sequencer reset*/
    AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 0x1
    while (AdcRegs.ADCST.bit.INT_SEQ1 == 0) {
    }
    asm(" RPT #11 || NOP");
    tuyen2_B.ADC_o1 = (AdcRegs.ADCRESULT0) >> 4;
    tuyen2_B.ADC_o2 = (AdcRegs.ADCRESULT1) >> 4;
    tuyen2_B.ADC_o3 = (AdcRegs.ADCRESULT2) >> 4;
    tuyen2_B.ADC_o4 = (AdcRegs.ADCRESULT3) >> 4;
    tuyen2_B.ADC_o5 = (AdcRegs.ADCRESULT4) >> 4;
    AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;/* Sequencer reset*/
    AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
  }

    if (tuyen2_B.ADC_o1 >= tuyen2_P.Saturation_UpperSat) {
    rtb_Gain_h = tuyen2_P.Saturation_UpperSat;
  } else if (tuyen2_B.ADC_o1 <= tuyen2_P.Saturation_LowerSat)
{
    rtb_Gain_h = tuyen2_P.Saturation_LowerSat;
  } else {
    rtb_Gain_h = tuyen2_B.ADC_o1;
  }

    rtb_Fcn_l = tuyen2_P.Normalize1Umax_Gain *
rt_hypotd_snf(tuyen2_B.UnitDelay1,
    tuyen2_B.UnitDelay2);

    tuyen2_B.UnitDelay1 =
rt_modd_snf(rt_atan2d_snf(tuyen2_B.UnitDelay2,
    tuyen2_B.UnitDelay1), tuyen2_P.pi3_Value);
  tuyen2_B.UnitDelay2 = 1.0 / rtb_Gain_h * rtb_Fcn_l *
sin(tuyen2_P.pi1_Value -
    tuyen2_B.UnitDelay1) * tuyen2_P.modulationindex3_Gain;

  tuyen2_B.UnitDelay1 = 1.0 / rtb_Gain_h * rtb_Fcn_l *
sin(tuyen2_B.UnitDelay1) *
    tuyen2_P.modulationindex2_Gain;

    rtb_UnitDelay2_idx = (tuyen2_P.one_Value -
tuyen2_B.UnitDelay1) -
    tuyen2_B.UnitDelay2;

  for (i = 0; i < 3; i++) {
    tuyen2_B.Product3[i] = 0.0;
    tuyen2_B.Product3[i] += tuyen2_B.sectorselector[i] *
tuyen2_B.UnitDelay2;
    tuyen2_B.Product3[i] += tuyen2_B.sectorselector[i + 3] *
```

```
tuyen2_B.UnitDelay1;
    tuyen2_B.Product3[i] += tuyen2_B.sectorselector[i + 6] *
rtb_UnitDelay2_idx;
  }
  {
    EvaRegs.CMPR1 = (uint16_T) (((double)EvaRegs.T1PR / 100)*
      tuyen2_B.Product3[0]);
    EvaRegs.CMPR2 = (uint16_T) (((double)EvaRegs.T1PR / 100)*
      tuyen2_B.Product3[1]);
    EvaRegs.CMPR3 = (uint16_T) (((double)EvaRegs.T1PR / 100)*
      tuyen2_B.Product3[2]);
  }

    {
    uint16_T* ptrOldPulseCount =
&tuyen2_DWork.QEP_OLD_PULSE_COUNT;
    tuyen2_B.QEP = ((int16_T)(*ptrOldPulseCount -
EvaRegs.T2CNT))*QEP_A_scaler;
    *ptrOldPulseCount = EvaRegs.T2CNT;
  }
  tuyen2_B.UnitDelay2 = (tuyen2_P.Gain1_Gain *
tuyen2_B.ADC_o4 + tuyen2_B.ADC_o3)* tuyen2_P.Gain_Gain;

    rtb_Gain_h =
tuyen2_P.Gain_Gain_j*tuyen2_DWork.DiscreteTimeIntegrator_DSTA
TE;
rtb_Fcn_l = tuyen2_B.ADC_o3 * cos(rtb_Gain_h) +
tuyen2_B.UnitDelay2 * sin(rtb_Gain_h);

rtb_LrdLrq2pito = tuyen2_B.QEP *
rtb_Fcn_l*tuyen2_P.LrdLrq2pito_Gain;

rtb_LrqLrd2pito = tuyen2_P.LrqLrd2pito_Gain * tuyen2_B.QEP;

rtb_PsipLrq2pito = tuyen2_P.PsipLrq2pito_Gain * tuyen2_B.QEP;
rtb_Fcn1_g = -tuyen2_B.ADC_o3 * sin(rtb_Gain_h) +
tuyen2_B.UnitDelay2 * cos(rtb_Gain_h);
rtb_Trq = tuyen2_P.Trq_Gain * rtb_Fcn1_g;

rtb_TSamp = tuyen2_P.Constant_Value_k *tuyen2_P.TSamp_WtEt;

rtb_Diff = rtb_TSamp - tuyen2_DWork.UD_DSTATE;
if (tuyen2_DWork.IC_FirstOutputTime) {
    tuyen2_DWork.IC_FirstOutputTime = FALSE;
    tuyen2_B.UnitDelay2 = tuyen2_P.IC_Value;
  } else {
    tuyen2_B.UnitDelay2 = tuyen2_B.ADC_o2;
  }
rtb_Sum1 = tuyen2_B.UnitDelay2 -
tuyen2_DWork.DiscreteTimeIntegrator_DSTATE;
rtb_Gain = tuyen2_P.Gain_Gain_jv * rtb_Sum1;
```

```
if ((rtb_Gain < 0.0) || (rtb_Gain > 1.0)) {
   rtb_Merge = tuyen2_P._Value_i;
  }
 else if ((rtb_Gain >= 0.6) && (rtb_Gain <= 1.0)) {
 rtb_Merge = tuyen2_P._Value_d;

      }
 else if (rtb_Gain < 0.6) {
  tuyen2_IfActionSubsystem3(rtb_Gain, &rtb_Merge,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem1);

      } else {
        tuyen2_IfActionSubsystem3_e(rtb_Gain, &rtb_Merge,
      (rtP_IfActionSubsystem3_tuyen2_b *)
&tuyen2_P.IfActionSubsystem3);

      }

    tuyen2_B.UnitDelay2 = tuyen2_P.Weight_Value * rtb_Merge;

    for (i = 0; i < 101; i++) {
    if ((tuyen2_B.UnitDelay2 <= tuyen2_P.PH_Value[i]) ||
rtIsNaN
        (tuyen2_P.PH_Value[i])) {
      tuyen2_B.ProductCOA[i] = tuyen2_B.UnitDelay2;
    } else {
      tuyen2_B.ProductCOA[i] = tuyen2_P.PH_Value[i];
    }
  }

    if ((rtb_Gain < 0.0556) || (rtb_Gain > 0.455)) {
        rtb_Merge_b = tuyen2_P._Value_e;

  } else if ((rtb_Gain >= 0.103) && (rtb_Gain <= 0.198)) {
        rtb_Merge_b = tuyen2_P._Value_b;

      } else if (rtb_Gain < 0.103) {
        tuyen2_IfActionSubsystem3(rtb_Gain, &rtb_Merge_b,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem1_o);
  } else {
        tuyen2_IfActionSubsystem3_e(rtb_Gain, &rtb_Merge_b,
      (rtP_IfActionSubsystem3_tuyen2_b *)
&tuyen2_P.IfActionSubsystem3_m);

      }

    tuyen2_B.UnitDelay1 = tuyen2_P.Weight_Value_b *
rtb_Merge_b;
```

```
    for (i = 0; i < 101; i++) {
    if ((tuyen2_B.UnitDelay1 <= tuyen2_P.PM_Value[i]) ||
rtIsNaN
        (tuyen2_P.PM_Value[i])) {
      tuyen2_B.minV[i] = tuyen2_B.UnitDelay1;
    } else {
      tuyen2_B.minV[i] = tuyen2_P.PM_Value[i];
    }
  }

   rtb_Gain1_m = (rtb_Sum1 - tuyen2_DWork.UnitDelay_DSTATE) *
    tuyen2_P.Gain1_Gain_p;

    if ((rtb_Gain1_m < -0.7963) || (rtb_Gain1_m > 1.8)) {
        rtb_Merge_f = tuyen2_P._Value_o;

      } else if (rtb_Gain1_m == 1.0) {
        rtb_Merge_f = tuyen2_P._Value_h;

      } else if (rtb_Gain1_m < 1.0) {
        tuyen2_IfActionSubsystem3(rtb_Gain1_m, &rtb_Merge_f,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem3_l);

      } else {
        tuyen2_IfActionSubsystem2(rtb_Gain1_m, &rtb_Merge_f,
      (rtP_IfActionSubsystem2_tuyen2 *)
&tuyen2_P.IfActionSubsystem2_n);

      }

    if ((rtb_Gain < -0.0767195767195766) || (rtb_Gain >
0.13)) {
        rtb_Merge_bc = tuyen2_P._Value_m;

      } else if (rtb_Gain == 0.0) {
        rtb_Merge_bc = tuyen2_P._Value_ou;

      } else if (rtb_Gain < 0.0) {
        tuyen2_IfActionSubsystem3(rtb_Gain, &rtb_Merge_bc,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem3_c);

      } else {
        tuyen2_IfActionSubsystem2(rtb_Gain, &rtb_Merge_bc,
      (rtP_IfActionSubsystem2_tuyen2 *)
&tuyen2_P.IfActionSubsystem2_l);

      }

    if ((rtb_Merge_bc <= rtb_Merge_f) ||
```

```
rtIsNaN(rtb_Merge_f)) {
    rtb_Sum1_e = rtb_Merge_bc;
  } else {
    rtb_Sum1_e = rtb_Merge_f;
  }

    rtb_Weighting = tuyen2_P.Weight_Value_g * rtb_Sum1_e;

  for (i = 0; i < 101; i++) {
    if ((rtb_Weighting <= tuyen2_P.PL_Value[i]) ||
rtIsNaN(tuyen2_P.PL_Value[i]))
    {
      tuyen2_B.minV_m[i] = rtb_Weighting;
    } else {
      tuyen2_B.minV_m[i] = tuyen2_P.PL_Value[i];
    }
  }

    if ((rtb_Gain1_m < -1.8) || (rtb_Gain1_m > 0.802)) {
        rtb_Merge_k = tuyen2_P._Value;

      } else if (rtb_Gain1_m == -1.008) {
        rtb_Merge_k = tuyen2_P._Value_k;

      } else if (rtb_Gain1_m < -1.008) {
        tuyen2_IfActionSubsystem3(rtb_Gain1_m, &rtb_Merge_k,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem3_m0);

      } else {
        tuyen2_IfActionSubsystem2(rtb_Gain1_m, &rtb_Merge_k,
      (rtP_IfActionSubsystem2_tuyen2 *)
&tuyen2_P.IfActionSubsystem2_p);

      }


  if ((rtb_Merge_bc <= rtb_Merge_k) || rtIsNaN(rtb_Merge_k))
{
    rtb_Sum1_e = rtb_Merge_bc;
  } else {
    rtb_Sum1_e = rtb_Merge_k;
  }

    rtb_Weighting_o = tuyen2_P.Weight_Value_l * rtb_Sum1_e;

    for (i = 0; i < 101; i++) {
    if ((rtb_Weighting_o <= tuyen2_P.NC_Value[i]) ||
rtIsNaN(tuyen2_P.NC_Value[i]))
    {
      tuyen2_B.minV_c[i] = rtb_Weighting_o;
```

```
    } else {
      tuyen2_B.minV_c[i] = tuyen2_P.NC_Value[i];
    }
  }

    if ((rtb_Gain1_m < -0.03529) || (rtb_Gain1_m > 0.03968))
{
        rtb_Merge_g = tuyen2_P._Value_f;

      } else if (rtb_Gain1_m == 0.01071) {
        rtb_Merge_g = tuyen2_P._Value_l;

      } else if (rtb_Gain1_m < 0.01071) {
        tuyen2_IfActionSubsystem3(rtb_Gain1_m, &rtb_Merge_g,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem3_d);

      } else {
        tuyen2_IfActionSubsystem2(rtb_Gain1_m, &rtb_Merge_g,
      (rtP_IfActionSubsystem2_tuyen2 *)
&tuyen2_P.IfActionSubsystem2_pj);

  }

  if ((rtb_Merge_bc <= rtb_Merge_g) || rtIsNaN(rtb_Merge_g))
{
    rtb_Sum1_e = rtb_Merge_bc;
  } else {
    rtb_Sum1_e = rtb_Merge_g;
  }

    rtb_Product4 = tuyen2_P.Weight_Value_c * rtb_Sum1_e;

    for (i = 0; i < 101; i++) {
    if ((rtb_Product4 <= tuyen2_P.NC_Value[i]) ||
rtIsNaN(tuyen2_P.NC_Value[i]))
    {
      tuyen2_B.minV_k[i] = rtb_Product4;
    } else {
      tuyen2_B.minV_k[i] = tuyen2_P.NC_Value[i];
    }
  }

    if ((rtb_Gain < -0.537) || (rtb_Gain > -0.167)) {
        rtb_Merge_bu = tuyen2_P._Value_n;

      } else if ((rtb_Gain >= -0.357142857142857) &&
(rtb_Gain <= -0.257)) {
        rtb_Merge_bu = tuyen2_P._Value_fc;

      } else if (rtb_Gain < -0.357142857142857) {
```

```
        tuyen2_IfActionSubsystem3(rtb_Gain, &rtb_Merge_bu,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem1_m);

      } else {
        tuyen2_IfActionSubsystem3_e(rtb_Gain, &rtb_Merge_bu,
      (rtP_IfActionSubsystem3_tuyen2_b *)
&tuyen2_P.IfActionSubsystem3_e);

  }

    rtb_Product3 = tuyen2_P.Weight_Value_bi * rtb_Merge_bu;

    for (i = 0; i < 101; i++) {
    if ((rtb_Product3 <= tuyen2_P.NL_Value[i]) ||
rtIsNaN(tuyen2_P.NL_Value[i]))
    {
      tuyen2_B.minV_cx[i] = rtb_Product3;
    } else {
      tuyen2_B.minV_cx[i] = tuyen2_P.NL_Value[i];
    }
  }

    if ((rtb_Gain < -1.72) || (rtb_Gain > -0.00794)) {
        rtb_Merge_m = tuyen2_P._Value_oq;

      } else if ((rtb_Gain >= -1.08) && (rtb_Gain <= -
0.579365079365079)) {
        rtb_Merge_m = tuyen2_P._Value_p;

      } else if (rtb_Gain < -1.08) {
        tuyen2_IfActionSubsystem3(rtb_Gain, &rtb_Merge_m,
      (rtP_IfActionSubsystem3_tuyen2 *)
&tuyen2_P.IfActionSubsystem1_g);

      } else {
        tuyen2_IfActionSubsystem3_e(rtb_Gain, &rtb_Merge_m,
      (rtP_IfActionSubsystem3_tuyen2_b *)
&tuyen2_P.IfActionSubsystem3_ef);

      }

    rtb_Gain4 = tuyen2_P.Weight_Value_gs * rtb_Merge_m;

    for (i = 0; i < 101; i++) {
    if ((tuyen2_B.ProductCOA[i] >= tuyen2_B.minV[i]) ||
rtIsNaN(tuyen2_B.minV[i]))
    {
      rtb_UnitDelay2_idx = tuyen2_B.ProductCOA[i];
    } else {
      rtb_UnitDelay2_idx = tuyen2_B.minV[i];
```

```
    }

    if (!((rtb_UnitDelay2_idx >= tuyen2_B.minV_m[i]) ||
rtIsNaN
          (tuyen2_B.minV_m[i]))) {
      rtb_UnitDelay2_idx = tuyen2_B.minV_m[i];
    }

    if (!((rtb_UnitDelay2_idx >= tuyen2_B.minV_c[i]) ||
rtIsNaN
          (tuyen2_B.minV_c[i]))) {
      rtb_UnitDelay2_idx = tuyen2_B.minV_c[i];
    }

    if (!((rtb_UnitDelay2_idx >= tuyen2_B.minV_k[i]) ||
rtIsNaN
          (tuyen2_B.minV_k[i]))) {
      rtb_UnitDelay2_idx = tuyen2_B.minV_k[i];
    }

    if (!((rtb_UnitDelay2_idx >= tuyen2_B.minV_cx[i]) ||
rtIsNaN
          (tuyen2_B.minV_cx[i]))) {
      rtb_UnitDelay2_idx = tuyen2_B.minV_cx[i];
    }

    if ((rtb_Gain4 <= tuyen2_P.NH_Value[i]) ||
rtIsNaN(tuyen2_P.NH_Value[i])) {
      rtb_Sum1_e = rtb_Gain4;
    } else {
      rtb_Sum1_e = tuyen2_P.NH_Value[i];
    }

    if ((rtb_UnitDelay2_idx >= rtb_Sum1_e) ||
rtIsNaN(rtb_Sum1_e)) {
      rtb_Sum1_e = rtb_UnitDelay2_idx;
    }

    tuyen2_B.ProductCOA[i] = rtb_Sum1_e;
  }

    rtb_Sum1_e = tuyen2_B.ProductCOA[0];
  for (i = 0; i < 100; i++) {
    rtb_Sum1_e += tuyen2_B.ProductCOA[i + 1];
  }

    if ((real_T)(((((tuyen2_B.UnitDelay2 +
tuyen2_B.UnitDelay1) + rtb_Weighting)
                   + rtb_Weighting_o) + rtb_Product4) +
rtb_Product3) + rtb_Gain4
                 > tuyen2_P.Zero_Value) >=
```

```
tuyen2_P.Switch_Threshold_l) {
        for (i = 0; i < 101; i++) {
      tuyen2_B.ProductCOA[i] *= tuyen2_P.xdata_Value[i];
    }

    rtb_Sum_oq = tuyen2_B.ProductCOA[0];
    for (i = 0; i < 100; i++) {
      rtb_Sum_oq += tuyen2_B.ProductCOA[i + 1];
    }

    if (rtb_Sum1_e <= 0.0) {
      rtb_Sum1_e = tuyen2_P.One_Value;

          }

        rtb_Gain4 = rtb_Sum_oq / rtb_Sum1_e;
  } else {
    rtb_Gain4 = tuyen2_P.MidRange_Value;
  }

    rtb_Sum1_e = (((tuyen2_P.Gain2_Gain * rtb_Gain4 -
tuyen2_B.QEP) -
tuyen2_P.DiscreteTransferFcn_DenCoef[1L]*tuyen2_DWork.Discret
eTransferFcn_DSTATE[0L]) -
                tuyen2_P.DiscreteTransferFcn_DenCoef[2L] *
                tuyen2_DWork.DiscreteTransferFcn_DSTATE[1L])/
 tuyen2_P.DiscreteTransferFcn_DenCoef[0];
rtb_Gain4 = (tuyen2_P.DiscreteTransferFcn_NumCoef[0] *
rtb_Sum1_e +
               tuyen2_P.DiscreteTransferFcn_NumCoef[1L] *
               tuyen2_DWork.DiscreteTransferFcn_DSTATE[0L]) +
    tuyen2_P.DiscreteTransferFcn_NumCoef[2L] *
    tuyen2_DWork.DiscreteTransferFcn_DSTATE[1L];

    rtb_DirectLookUpTablenD = rtb_Gain4 *
tuyen2_P.TSamp_WtEt_c;

    rtb_Gain4 -= rtb_Fcn1_g;

    tuyen2_B.UnitDelay1 = (tuyen2_P.DerivativeGain_Gain *
rtb_Gain4 -
    tuyen2_DWork.Filter_DSTATE) *
tuyen2_P.FilterCoefficient_Gain;

    tuyen2_B.UnitDelay2 = tuyen2_P.IntegralGain_Gain *
rtb_Gain4;

    rtb_Product3 = tuyen2_P.Constant_Value_k - rtb_Fcn_l;

    rtb_Weighting = (tuyen2_P.DerivativeGain_Gain_e *
rtb_Product3 -
```

```
                    tuyen2_DWork.Filter_DSTATE_e) *
    tuyen2_P.FilterCoefficient_Gain_f;

  rtb_Sum_oq = tuyen2_P.IntegralGain_Gain_g * rtb_Product3;

    rtb_Product3 = (((((tuyen2_P.ProportionalGain_Gain_o *
rtb_Product3 +
                       tuyen2_DWork.Integrator_DSTATE_g) +
rtb_Weighting) +
                      rtb_Diff) + tuyen2_P.Gain_Gain_e *
rtb_Product3) +
                    tuyen2_P.Gain2_Gain_p * rtb_Fcn_l) +
rtb_LrqLrd2pito *
                   rtb_Fcn1_g) * tuyen2_P.Gain1_Gain_m;

    rtb_Gain4 = (((((tuyen2_P.Gain3_Gain * rtb_Gain4 +
(rtb_DirectLookUpTablenD -
    tuyen2_DWork.UD_DSTATE_k)) + rtb_Trq) + rtb_PsipLrq2pito)
+ rtb_LrdLrq2pito)
                + ((tuyen2_P.ProportionalGain_Gain * rtb_Gain4
+
                    tuyen2_DWork.Integrator_DSTATE) +
tuyen2_B.UnitDelay1)) *
    tuyen2_P.Gain4_Gain;

    rtb_Weighting_o = rtb_Product3 * rtb_Product3 + rtb_Gain4
* rtb_Gain4;
  if (rtb_Weighting_o < 0.0) {
    rtb_Weighting_o = -sqrt(-rtb_Weighting_o);
  } else {
    rtb_Weighting_o = sqrt(rtb_Weighting_o);
  }

    if (rtb_Weighting_o >= tuyen2_P.Saturation_UpperSat_o) {
    rtb_UnitDelay2_idx = tuyen2_P.Saturation_UpperSat_o;
  } else if (rtb_Weighting_o <=
tuyen2_P.Saturation_LowerSat_k) {
    rtb_UnitDelay2_idx = tuyen2_P.Saturation_LowerSat_k;
  } else {
    rtb_UnitDelay2_idx = rtb_Weighting_o;
  }

    if (!(rtb_Weighting_o <= tuyen2_B.ADC_o1)) {
    rtb_Weighting_o = tuyen2_B.ADC_o1;
  }

 rtb_Product4 = 1.0 / rtb_UnitDelay2_idx * rtb_Weighting_o;
    rtb_Product3 *= rtb_Product4;
    rtb_Product4 *= rtb_Gain4;
    tuyen2_DWork.UnitDelay1_DSTATE = rtb_Product3 *
cos(rtb_Gain_h) - rtb_Product4
```

```
    * sin(rtb_Gain_h);
    tuyen2_DWork.UnitDelay2_DSTATE = rtb_Product3 *
sin(rtb_Gain_h) + rtb_Product4
    * cos(rtb_Gain_h);
    tuyen2_DWork.DiscreteTimeIntegrator_DSTATE +=
    tuyen2_P.DiscreteTimeIntegrator_gainval * tuyen2_B.QEP;
    tuyen2_DWork.UD_DSTATE = rtb_TSamp;
    tuyen2_DWork.UnitDelay_DSTATE = rtb_Sum1;
    tuyen2_DWork.DiscreteTransferFcn_DSTATE[1L] =
    tuyen2_DWork.DiscreteTransferFcn_DSTATE[0L];
  tuyen2_DWork.DiscreteTransferFcn_DSTATE[0L] = rtb_Sum1_e;
    tuyen2_DWork.UD_DSTATE_k = rtb_DirectLookUpTablenD;
    tuyen2_DWork.Filter_DSTATE += tuyen2_P.Filter_gainval *
tuyen2_B.UnitDelay1;
    tuyen2_DWork.Integrator_DSTATE +=
tuyen2_P.Integrator_gainval *
    tuyen2_B.UnitDelay2;
  tuyen2_DWork.Filter_DSTATE_e += tuyen2_P.Filter_gainval_d *
rtb_Weighting;
  tuyen2_DWork.Integrator_DSTATE_g +=
tuyen2_P.Integrator_gainval_m * rtb_Sum_oq;
}
void tuyen2_initialize(void)
{
  rt_InitInfAndNaN(sizeof(real_T));
  tuyen2_P.Saturation_UpperSat = rtInf;
  tuyen2_P.Saturation_UpperSat_o = rtInf;
  rtmSetErrorStatus(tuyen2_M, (NULL));
  (void) memset(((void *) &tuyen2_B),
0,sizeof(BlockIO_tuyen2));
  (void) memset((void *)&tuyen2_DWork, 0,
                sizeof(D_Work_tuyen2));
  InitAdc();
  config_ADC_A (4U, 12816U, 4U, 0U, 0U);
  EALLOW;
  GpioMuxRegs.GPAMUX.all |= 63U;
  config_PWM_A (3750.0,1,1,"INPUT_PORT",0.0,
                1,"INPUT_PORT",0.0,1,"INPUT_PORT",0.0,1638,
                0, 0, 0, 0, 1, 0, 0);
  EDIS;
config_QEP_A (0, 65535, 0, 0);
  tuyen2_DWork.IC_FirstOutputTime = TRUE;

tuyen2_DWork.UnitDelay1_DSTATE = tuyen2_P.UnitDelay1_X0;

tuyen2_DWork.UnitDelay2_DSTATE = tuyen2_P.UnitDelay2_X0;
tuyen2_DWork.DiscreteTimeIntegrator_DSTATE =
    tuyen2_P.DiscreteTimeIntegrator_IC;

 tuyen2_DWork.UD_DSTATE = tuyen2_P.UD_X0;
  tuyen2_DWork.UnitDelay_DSTATE = tuyen2_P.UnitDelay_X0;
```

```
  tuyen2_DWork.DiscreteTransferFcn_DSTATE[0] =
    tuyen2_P.DiscreteTransferFcn_InitialStat;
  tuyen2_DWork.DiscreteTransferFcn_DSTATE[1] =
    tuyen2_P.DiscreteTransferFcn_InitialStat;
  tuyen2_DWork.UD_DSTATE_k = tuyen2_P.UD_X0_o;

  tuyen2_DWork.Filter_DSTATE = tuyen2_P.Filter_IC;
  tuyen2_DWork.Integrator_DSTATE = tuyen2_P.Integrator_IC;
  tuyen2_DWork.Filter_DSTATE_e = tuyen2_P.Filter_IC_n;

    tuyen2_DWork.Integrator_DSTATE_g =
tuyen2_P.Integrator_IC_h;
}
```

/*

*** File: tuyen2_data.c**

***/**

#include "tuyen2.h"

#include "tuyen2_private.h"

Parameters_tuyen2 tuyen2_P = {

0.0, { -1.0, -0.98, -0.96, -0.94, -0.92, -0.9, -0.88, -0.86, -0.84,

-0.82000000000000006, -0.8, -0.78, -0.76, -0.74, -0.72, -0.7, -0.67999999999999994, -0.65999999999999992, -0.64, -0.62, -0.6, -0.58000000000000007, -0.56, -0.54, -0.52, -0.5, -0.48, -0.45999999999999996, -0.43999999999999995, -0.42000000000000004, -0.4, -0.38, -0.36,-0.33999999999999997, -0.31999999999999995, -0.30000000000000004, -0.28,-0.26, -0.24, -0.21999999999999997, -0.19999999999999996, -0.18000000000000005, -0.16000000000000003, -0.14, -0.12,

-0.099999999999999978, -0.079999999999999996, -0.060000000000000053,

-0.040000000000000036, -0.020000000000000018, 0.0, 0.020000000000000018,

0.040000000000000036, 0.060000000000000053, 0.080000000000000071,

0.10000000000000009, 0.12000000000000011, 0.1399999999999999,

0.15999999999999992, 0.17999999999999994, 0.19999999999999996,

0.21999999999999997, 0.24, 0.26, 0.28, 0.30000000000000004,

0.32000000000000006, 0.34000000000000008, 0.3600000000000001,

0.37999999999999989, 0.39999999999999991, 0.41999999999999993,
0.43999999999999995, 0.45999999999999996, 0.48, 0.5, 0.52, 0.54, 0.56,
0.58000000000000007, 0.60000000000000009, 0.62000000000000011,
0.6399999999999999, 0.65999999999999992, 0.67999999999999994, 0.7, 0.72,
0.74, 0.76, 0.78, 0.8, 0.82000000000000006, 0.84000000000000008,
0.8600000000000001, 0.87999999999999989, 0.89999999999999991,
0.91999999999999993, 0.94, 0.96, 0.98, 1.0 },0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0,
1.0,1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
0.0,1.0, 0.0, 0.0, 1.0, 1.0, { 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.5, 0.5, 0.5 },{ 0.0, 0.0, 1.0, 1.0, 0.0,
1.0, 0.5, 0.5, 0.5 },{ 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.5, 0.5, 0.5 },{ 0.0, 1.0, 0.0, 0.0, 1.0, 1.0,
0.5, 0.5, 0.5 },{ 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.5, 0.5, 0.5 },{ 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.5,
0.5, 0.5 },1.0, 0.0, 0.0, 0.0, 0.5,{ 2.0, 5.0, 1.0, 6.0, 3.0, 4.0 }, 0.0, 2.2204460492503131E-
16,1.5,1.0466666666666666,1.0466666666666666,1.1547005383792517,1.154700538379
2517, 1.0, 2.0, 0.57735026918962584, 0.001, 0.0, 314.0, 314.0, 314.0,
186904.76190476192, 1.5714285714285714, 0.0, 1000.0, 0.0, 1.0, 0.0,0.08,{ 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.062937062937062985, 0.13286713286713298, 0.20279720279720295,
0.27272727272727293, 0.34265734265734293, 0.41258741258741288,
0.48251748251748289, 0.55244755244755206, 0.62237762237762206,
0.69230769230769207, 0.76223776223776207, 0.83216783216783208,
0.902097902097902, 0.972027972027972 }, 1.0, { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0823529411764713,0.19999999999999948, 0.31764705882352895,
0.43529411764705844,0.55294117647058794, 0.67058823529411749,
0.78823529411764692,0.90588235294117636, 0.9800895968143355,
0.88053758088601286, 0.78098556495769011, 0.68143354902936748,

0.58188153310104485, 0.48232951717272221, 0.38277750124440063, 0.283225485316078,0.18367346938775536, 0.0841214534594327, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0},1.0, 0.0, 0.08, { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.18774193548387097, 0.51032258064516156,0.83290322580645215, 0.947903156074362, 0.839818417639429, 0.731733679204496, 0.62364894076956279, 0.515564202334631, 0.4074794638996978,

0.29939472546476475, 0.19130998702983162, 0.083225248594898549, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },1.0, { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0,0.35750000000000001, 0.85750000000000048, 0.674999999999999, 0.22045454545454396, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },1.0,1.0, { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.11494252873563225,0.2298850574712645, 0.3448275862068968, 0.459770114942529,0.57471264367816133, 0.68965517241379359, 0.80459770114942575, 0.91954022988505746, 0.99037465068003094, 0.95829015294680064,0.92620565521357057, 0.89412115748034027, 0.86203665974711008,0.82995216201387978, 0.79786766428064959, 0.76578316654741929,0.7336986688141891, 0.7016141710809588, 0.66952967334772873, 0.63744517561449843, 0.60536067788126813, 0.57327618014803794, 0.54119168241480764, 0.50910718468157734, 0.47702268694834732, 0.444938189215117, 0.41285369148188678, 0.38076919374865648,0.34868469601542618, 0.31660019828219593, 0.28451570054896586,0.25243120281573556, 0.22034670508250531, 0.18826220734927507,0.1561777096160448, 0.12409321188281452, 0.092008714149584434, 0.059924216416354169, 0.027839718683123908, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },1.0, { 0.96438356164383587, 0.90958904109589056,

```
0.85479452054794536, 0.8,0.74520547945205506, 0.69041095890410986,
0.63561643835616455,0.58082191780821923, 0.526027397260274,
0.47123287671232905, 0.41643835616438374, 0.36164383561643848,
0.30684931506849322,0.25205479452054796, 0.19726027397260268,
0.14246575342465739,0.087671232876712135, 0.032876712328766856, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0 },0.0,  1.0,
30.0,   0.0, { 1258.5291887793737, 20.143290371493478, 46.209098714798372 },

  { 1.0, 12.983928131022903, 1235.7846853676981 },1000.0, 0.0, 0.0,  0.001,  0.0, 100.0,
2.0,  0.001, 0.0, 0.0, 0.0, 0.001, 0.0, 100.0, 2.0, 0.001,  0.0, 0.0, 1.0,  1.5714285714285714,
0.0168, 1.0, 0.0168, 0.0, 1.0E-9,{0.0,  0.13 },{-0.0767195767195766,  0.0 },{0.198, 0.455
},{0.0556,  0.103 },{1.0, 1.0 },{ 0.0,  0.6  },{ -0.257,  -0.167 },{-0.537, -0.357142857142857
},{ -0.579365079365079, -0.00794  }, {-1.72,  -1.08 },{ 0.01071,  0.03968  },{ -0.03529,
0.01071 },{1.0,  1.8  },{  -0.7963, 1.0   },{-1.008,   0.802  },{ -1.8,  -1.008    }};

/*

 * File: tuyen2_main.c

*/

#include "tuyen2.h"

#include "rtwtypes.h"

#include "rt_nonfinite.h"

#include "tuyen2_private.h"

#include "c2000_main.h"

#include "DSP281x_Device.h"

#include "DSP281x_Examples.h"

#include <stdlib.h>

#include <stdio.h>

void init_board(void);
```

```
void enable_interrupts(void);
void config_schedulerTimer(void);
void disable_interrupts(void);
volatile int IsrOverrun = 0;
static boolean_T OverrunFlag = 0;
void rt_OneStep(void)
{
  /* Check for overrun. Protect OverrunFlag against
   * preemption.
   */
  if (OverrunFlag++) {
    IsrOverrun = 1;
    OverrunFlag--;
    return;
  }
asm(" SETC INTM");
  PieCtrlRegs.PIEIER1.all |= (1 << 6);
  asm(" CLRC INTM");
  tuyen2_step();
  asm(" SETC INTM");
  PieCtrlRegs.PIEIER1.all &= ~(1 << 6);
  asm(" RPT #5 || NOP");
  IFR &= 0xFFFE;
  PieCtrlRegs.PIEACK.all = 0x1;
  asm(" CLRC INTM");
  OverrunFlag--;
```

```
}
void main(void)
{
  volatile boolean_T noErr;
  init_board();
  rtmSetErrorStatus(tuyen2_M, 0);
  tuyen2_initialize();
  config_schedulerTimer();
  noErr =
    rtmGetErrorStatus(tuyen2_M) == (NULL);
  enable_interrupts();
  while (noErr ) {
  noErr = rtmGetErrorStatus(tuyen2_M) == (NULL);
  }
  tuyen2_terminate();
  disable_interrupts();
}
```