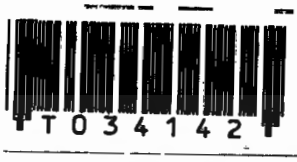
การใช้ไมโครคอนโทรลเลอร์ควบคุมมอเตอร์เหนี่ยวนำแบบเวคตอร์

IMPLEMENTATION OF FLUX VECTOR CONTROL OF 3 PHASE INDUCTION MOTOR DRIVE USING A MICROCONTROLLER



โดย

นายธีรเดช พิญญพงษ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปีการศึกษา 2541

การใช้ไมโครคอนโทรลเลอร์ควบคุมมอเตอร์เหนี่ยวนำแบบเวคเตอร์

IMPLEMENTATION OF FLUX VECTOR CONTROL OF 3 PHASE INDUCTION MOTOR DRIVE USING A MICROCONTROLLER

โดย

นายธีรเดช พิญญพงษ์

อาจารย์ที่ปรึกษา

ดร. วิจิตร กิณเรศ

ดุสิต สุขสวัสดิ์

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้ไมโครคอนโทรลเลอร์ควบคุมมอเตอร์เหนี่ยวนำแบบเวคเตอร์

ผู้จัดทำ

นายธีรเดช พิญญพงษ์

.....อาจารย์ที่ปรึกษา

(ดร. วิจิตร กิณเรศ)

.....อาจารย์ที่ปรึกษา

(ดุสิต สุขสวัสดิ์)

การใช้ไมโครคอนโทรลเลอร์ควบคุมมอเตอร์เหนี่ยวนำแบบเวคตอร์

นายธีรเดช พิญญพงษ์
ดร. วิจิตร กิณเรศ อาจารย์ที่ปรึกษา
ดุสิต สุขสวัสดิ์
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์เล่มนี้เป็นการนำเสนอการควบคุมมอเตอร์เหนี่ยวนำกระแสสลับสามเฟสแบบเวคเตอร์ โดยใช้ไมโครคอนโทรลเลอร์ 80C196KB ขนาด 16 บิท ที่ทำงานในความถี่ 12 เมกกะเฮิร์ท ซึ่งหลักการที่ใช้คือการควบคุมกระแสไฟฟ้าบนขดลวดสเตเตอร์ เพื่อปรับเปลี่ยนสนามแม่เหล็กที่โรเตอร์แบบอ้อม จุดประสงค์ของการควบคุมแบบนี้คือการปรับพฤติกรรมของมอเตอร์เหนี่ยวนำกระแสสลับให้เสมือนกับมอเตอร์กระแสตรงแบบแยกกระตุ้น ซึ่งส่งผลให้สามารถควบคุมแรงบิดและสนามแม่เหล็กแยกออกจากกันเป็นอิสระได้ รวมถึงการจำลองการทำงานโดยใช้โปรแกรม MATLAB เพื่อศึกษาผลการเปลี่ยนแปลงความเร็วที่ต้องการกับกระแสที่ขดลวดสเตเตอร์เทียบกับผลการทดลอง

IMPLEMENTATION OF FLUX VECTOR CONTROL OF INDUCTION MOTOR DRIVE USING A MICROCONTROLLER

TERADACH PINYAPONG
Dr. VIJIT KINNARES ADVISOR
DUSIT SUKSAWAT
1998

abstract

This project presents an implementation of indirect flux vector control for 3-phase , 2 hp , induction motor using 16 bit 80c196 microcontroller. The principle of such control is that stator currents are constrained to orientate rotor flux indirectly. As a result, it gives an independence between flux and torque controls in AC induction motor, which is similar to separately excited DC motor control. In addition, simulation is employed for the verification of experimental results to study speed response and stator current characteristics.

สารบัญ

หน้า

สารบัญรูปภาพ

กำหนดสัญลักษณ์

$I_s^s(t)$	:	กระแสเฟสเซอร์ของสเตเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{sa}^s(t)$	:	กระแสสเตเตอร์เฟส a ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{sb}^s(t)$	:	กระแสสเตเตอร์เฟส b ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{sc}^s(t)$	:	กระแสสเตเตอร์เฟส c ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{ds}^s(t)$	:	กระแสสเตเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{qs}^s(t)$	:	กระแสสเตเตอร์บนแกนขวาง (quadrature axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_r^r(t)$	:	กระแสเฟสเซอร์ของโรเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_{dr}^r(t)$	:	กระแสโรเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_{qr}^r(t)$	:	กระแสโรเตอร์บนแกนขวาง (quadrature axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_r^s(t)$	:	กระแสเฟสเซอร์ของโรเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{dr}^s(t)$	:	กระแสโรเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{qr}^s(t)$	:	กระแสโรเตอร์บนแกนขวาง (quadrature axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_s^r(t)$	:	กระแสเฟสเซอร์ของสเตเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_{ds}^r(t)$	:	กระแสสเตเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_{qs}^r(t)$	:	กระแสสเตเตอร์บนแกนขวาง (quadrature axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$I_{mr}^s(t)$	:	กระแสเฟสเซอร์กระตุ้นจากสมามแม่เหล็กที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$I_{mr}^e(t)$	:	กระแสเฟสเซอร์กระตุ้นจากสมามแม่เหล็กที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น
$I_s^e(t)$	:	กระแสเฟสเซอร์สเตเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น
$I_r^e(t)$	:	กระแสเฟสเซอร์โรเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น
I_s^s	:	กระแสเฟสเซอร์ของสเตเตอร์ ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
I_r^r	:	กระแสเฟสเซอร์ของโรเตอร์ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่โรเตอร์
I_{sa}^s	:	กระแสสเตเตอร์เฟส a ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
I_{sb}^s	:	กระแสสเตเตอร์เฟส b ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
I_{sc}^s	:	กระแสสเตเตอร์เฟส c ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
I_r^s	:	กระแสเฟสเซอร์ของโรเตอร์ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

I^s_{ds} : กระแสเฟสเซอร์ของสเตเตอร์ขณะใดขณะหนึ่งบนบนแกนตรงที่กรอบอ้างอิงที่สเตเตอร์

I^s_{qs} : กระแสเฟสเซอร์ของสเตเตอร์ ขณะใดขณะหนึ่งบนแกนขวางที่กรอบอ้างอิงที่สเตเตอร์

I^s_{dr} : กระแสเฟสเซอร์ของโรเตอร์ขณะใดขณะหนึ่งบนแกนตรงที่กรอบอ้างอิงที่สเตเตอร์

$I^s_{q.}$: กระแสเฟสเซอร์ของโรเตอร์ขณะใดขณะหนึ่งบนแกนขวางที่กรอบอ้างอิงที่สเตเตอร์

$U^s_s(t)$: แรงดันเฟสเซอร์สเตเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^s_{sa}(t)$: แรงดันสเตเตอร์เฟส a ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^s_{sb}(t)$: แรงดันสเตเตอร์เฟส b ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^s_{sc}(t)$: แรงดันสเตเตอร์เฟส c ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^s_{ds}(t)$: แรงดันสเตเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^s_{qs}(t)$: แรงดันสเตเตอร์บนแกนขวาง (quadrature axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$U^r_r(t)$: แรงดันเฟสเซอร์โรเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์

$U^{'}_s(t)$: แรงดันเฟสเซอร์สเตเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น

$U^e_r(t)$: แรงดันเฟสเซอร์โรเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น

U^s_s : แรงดันเฟสเซอร์สเตเตอร์ ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

U^s_r : แรงดันเฟสเซอร์โรเตอร์ ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

U^s_{ds} : แรงดันสเตเตอร์บนแกนตรง (direct axis) ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

U^s_{qs} : แรงดันสเตเตอร์บนแกนขวาง (quadrature axis) ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

U^s_{dr} : แรงดันโรเตอร์บนแกนตรง (direct axis) ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

$U^{'}_{qr}$: แรงดันโรเตอร์บนแกนขวาง (quadrature axis) ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์

$\Psi^s_s(t)$: สนามแม่เหล็กเฟสเซอร์สเตเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$\Psi^s_r(t)$: สนามแม่เหล็กเฟสเซอร์โรเตอร์ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์

$\Psi^s_{sa}(t)$	:	สนามแม่เหล็กสเตเตอร์เฟส a ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^s_{sb}(t)$	:	สนามแม่เหล็กสเตเตอร์เฟส b ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^s_{sc}(t)$	:	สนามแม่เหล็กสเตเตอร์เฟส c ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^s_{ds}(t)$	:	สนามแม่เหล็กสเตเตอร์บนแกนตรง (direct axis) ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^s_{qs}(t)$	:	สนามแม่เหล็กสเตเตอร์บนแกนขวาง (quadrature axis)ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^s_{ds}(t)$	:	สนามแม่เหล็กสเตเตอร์เฟส a ที่เวลาใดๆ บนกรอบอ้างอิงที่สเตเตอร์
$\Psi^r_r(t)$	:	สนามแม่เหล็กเฟสเซอร์โรเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงที่โรเตอร์
$\Psi^e_s(t)$	:	สนามแม่เหล็กเฟสเซอร์สเตเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น
$\Psi^e_r(t)$	:	สนามแม่เหล็กเฟสเซอร์โรเตอร์ ที่เวลาใดๆ บนกรอบอ้างอิงกระตุ้น
Ψ^s_s	:	สนามแม่เหล็กเฟสเซอร์สเตเตอร์ ที่ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
Ψ^s_r	:	สนามแม่เหล็กเฟสเซอร์โรเตอร์ที่ขณะใดขณะหนึ่งบนกรอบอ้างอิงที่สเตเตอร์
T^s_e(t)	:	แรงบิดทางไฟฟ้าบนกรอบอ้างอิงที่สเตเตอร์ที่เวลาใด ๆ
T^e_e(t)	:	แรงบิดทางไฟฟ้าบนกรอบอ้างอิงกระตุ้นที่เวลาใด ๆ
T_r	:	ค่าคงที่เวลาของโรเตอร์
M	:	ค่าความเหนี่ยวนำร่วม
L_s	:	ค่าความเหนี่ยวนำที่สเตเตอร์
L_r	:	ค่าความเหนี่ยวนำที่โรเตอร์
R_s	:	ค่าความต้านทานที่สเตเตอร์
R_r	:	ค่าความต้านทานที่โรเตอร์
P	:	จำนวนขั้วของมอเตอร์
$\alpha,\beta,\delta,\varepsilon,\theta,\rho,\zeta$	:	มุม
γ	:	$2\pi/3$
ω_f	:	ความเร็วเชิงมุมของสนามแม่เหล็กกระตุ้น
ω_r	:	ความเร็วเชิงมุมของโรเตอร์
ω_l	:	ความเร็วเชิงมุมสลิป
ω_e	:	ความเร็วเชิงมุมไฟฟ้า
δ_r	:	ค่าตัวประกอบการสูญเสียของโรเตอร์

บทที่ 1 บทนำ

ปัจจุบันมอเตอร์มีบทบาททางอุตสาหกรรมและในชีวิตประจำวันเป็นอย่างมาก ในฐานะของเครื่องจักรกลที่เปลี่ยนแปลงพลังงานไฟฟ้าเป็นพลังงานกล ในอดีตมอเตอร์กระแสตรงเป็นที่นิยมใช้กันอย่างแพร่หลายแม้กระทั้งในปัจจุบัน เนื่องจากการควบคุมที่ง่ายให้แรงบิดเริ่มต้นสูงและชุดควบคุมมีราคาถูกเมื่อเทียบกับมอเตอร์กระแสสลับ แต่ข้อเสียของมอเตอร์กระแสตรงก็มีอยู่มาก ที่เห็นได้ชัดคือมีราคาสูง มีน้ำหนักมาก และต้องมีการดูแลรักษาเป็นอย่างดีโดยเฉพาะแปลงถ่านและซี่คอมมิวเตเตอร์ ที่เป็นจุดด้อยสำคัญที่สุดและต้องมีการดูแลรักษาหรือเปลี่ยนแปลงเสมอ และสถานที่ทำงานที่ใช้มอเตอร์กระแสตรงต้องไม่มีฝุ่นผง และไม่สามารถทำงานในที่ที่มีวัสดุไวไฟเนื่องจากบริเวณหน้าสัมผัสระหว่างซี่คอมมิวเตเตอร์และแปลงถ่านจะเกิดประกายไฟได้ง่าย แต่ในมอเตอร์กระแสสลับไม่จำเป็นที่จะต้องดูแลรักษามาก แต่ข้อเสียของมอเตอร์กระแสสลับคือการควบคุมที่มีความซับซ้อนมาก ทำให้ชุดควบคุมมีราคาแพงมาก เป็นเหตุให้มอเตอร์กระแสตรงเป็นที่นิยมใช้กันอย่างแพร่หลาย แต่เมื่อเทคโนโลยีทางด้านไมโครคอนโทรลเลอร์และอุปกรณ์สารกึ่งตัวนำที่มีการพัฒนาขีดความสามารถอย่างรวดเร็วในด้าน ความเร็วในการคำนวณที่ซับซ้อนและความถี่ในการเปิด-ปิดของสารกึ่งตัวนำ ทำให้การควบคุมมอเตอร์กระแสสลับมีประสิทธิภาพมากขึ้น และราคาของชุดควบคุมมีราคาต่ำลง ทำให้มอเตอร์กระแสสลับเริ่มใช้กันมากขึ้นและได้มีการพัฒนาการควบคุมมอเตอร์กระแสสลับให้มีประสิทธิภาพสูงเช่นการควบคุมเวคเตอร์ของสนามแม่เหล็ก ทำให้ลักษณะคล้ายกับมอเตอร์กระแสตรงชนิดแยกขดลวดกระตุ้น (Seperate Excited) แต่ต่างกันตรงที่มอเตอร์กระแสตรงควบคุมตำแหน่งและขนาดของกระแสสนามและกระแสอาร์มาเจอร์ด้วยแปลงถ่านและซี่คอมมิวเตเตอร์ แต่มอเตอร์กระแสสลับควบคุมตำแหน่งและขนาดของฟลักซ์และแรงบิดเป็นอิสระต่อกันโดยการควบคุมขนาดและมุมเฟสของกระแสหรือแรงดันที่สเตเตอร์ทั้งสามเฟสซึ่งจะมองในรูปของการรวมเวคเตอร์ของกระแสหรือแรงดัน (Space Vector)

การควบคุมมอเตอร์แบ่งเป็นสองประเภทดังนี้

1.) การควบคุมเฉพาะขนาด (Scalar Control)
   - การควบคุมอัตราส่วนระหว่างแรงดันและความถี่ให้คงที่
   - การควบคุมกระแสที่สเตเตอร์และการควบคุมความถี่สลิป (Stator current and Slip Frequency Control)

2.) การควบคุมแบบเวคเตอร์ (Vector Control)
   - การควบคุมสนาม (Field Orientation Control : FOC)
     - วิธีตรง (Direct Method)
     - วิธีอ้อม (Indirect Method)

- การควบคุมแรงบิดโดยตรงและการควบคุมสนามแม่เหล็กที่สเตเตอร์ (Direct Torque and Stator Flux Vector Control)

วัตถุประสงค์

1.) ศึกษา และสร้างระบบควบคุมแบบเวคเตอร์ของมอเตอร์เหนี่ยวนำได้

2.) สามารถใช้งานไมโครคอนโทรลเลอร์ INTEL 80C196KB ในการควบคุมมอเตอร์เหนี่ยวนำแบบเวคเตอร์ได้

ขอบเขต PROJECT I

1.) ศึกษาการทำงานและสร้างบอร์ดเดี่ยวของไมโครคอนโทรลเลอร์ 80C196KB ได้

2.) ศึกษาและเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ 80C196KB ควบคุมมอเตอร์กระแสสลับแบบเวคเตอร์ได้

ประโยชน์ที่คาดว่าจะได้รับ

1.) เป็นจุดเริ่มในการศึกษาการควบคุมมอเตอร์แบบเวคเตอร์ และศึกษาการทำงานของมอเตอร์

2.) พัฒนาการควบคุมให้มีประสิทธิภาพดี , มีความน่าเชื่อถือ และสามารสารถใช้งานได้จริงในระบบอุตสาหกรรมในอนาคต

3.) สามารถนำความรู้ที่ได้ไปประกอบอาชีพในอนาคต

โครงร่างของโปรเจค

ปริญญานิพนธ์จะกล่าวถึงการนำทฤษฎีการควบคุมแบบเวคเตอร์บนมอเตอร์เหนี่ยวนำกระแสสลับ 3 เฟส 2 แรงม้า โดยใช้ไมโครคอนโทรลเลอร์ เป็นการควบคุมสนามในโรเตอร์ทางอ้อม ผ่านทางกระแสไฟฟ้า และทำการจำลองการทำงานโดยโปรแกรม MATLAB

บท ที่1 บทนำ

กล่าวถึงเหตุผลในการทำโครงงาน ,วัตถุประสงค์ ,ขอบเขตของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ และโครงร่างของปริญญานิพนธ์

บทที่ 2 ทฤษฎีการควบคุมแบบเวคเตอร์

กล่าวถึงทฤษฎีการควบคุมแบบเวคเตอร์ การนำมาใช้งาน และโครงสร้างทางพลวัตของมอเตอร์เหนี่ยวนำกระแสสลับ3 เฟส

บทที่ 3 การสร้างแบบจำลองการควบคุมมอเตอร์เหนี่ยวนำกระแสสลับแบบเวคเตอร์

เป็นการนำโปรแกรม MATLAB สร้างแบบจำลองของระบบ และสังเกตผลการเปลี่ยนแปลงความเร็วกับกระแส และแรงบิด

บทที่ 4 การประยุกต์ใช้งาน

กล่าวถึงวงจรการทำงาน บอร์ดไมโครคอนโทรลเลอร์ วงจรขับ Inverter วงจรตรวจจับสัญญาณจาก encoder และวงจรการสร้างสัญญาณ PWM

บทที่ 5 การเขียนโปรแกรมควบคุมการทำงาน

โครงสร้าง Flow chart ของโปรแกรม

บทที่ 6 ผลการทดลอง

กราฟแสดงผลการหมุนมอเตอร์และสรุปผลจากกราฟ

บทที่ 7 สรุป

สรุป ปัญหาที่พบ และแนวทางในการพัฒนา

ภาคผนวก

I. Application Note Using the 80c196KB

II. Data Sheet IGBT model

III. Data Sheet EXB841

บทที่ 2 ทฤษฎี

2.1 การวิเคราะห์แบบเวคเตอร์ของมอเตอร์เหนี่ยวนำกระแสสลับ

กำหนดให้มอเตอร์เหนี่ยวนำที่ใช้มีลักษณะตัวถังสเตเตอร์เป็นแท่งเหล็กกลวงทรงกระบอกและที่โรเตอร์เป็นแท่งเหล็กตันที่วางขนานกัน มีพื้นผิวเรียบมีระยะห่างระหว่างช่องอากาศเท่ากันตลอดและที่ขดลวดสเตเตอร์พันกันอย่างสมดุลทั้งสามเฟสไม่คิดผลของความลึกของขดลวดและผลของการพันขดลวด แรงเคลื่อนแม่เหล็ก (Magnetomotive force : MMF) มีการกระจายเป็นรูปคลื่น sine ไม่มีผลที่เกิดจากการอิ่มตัวของแกนเหล็ก ผลของจุดสิ้นสุดการพันของขดลวด

การวิเคราะห์แบบเวคเตอร์ของมอเตอร์เหนี่ยวนำเป็นการวิเคราะห์คุณสมบัติของมอเตอร์ที่ซับซ้อน เนื่องจากเป็นการพิจารณาผลของทั้งส่วนที่เคลื่อนที่(Rotor)และส่วนที่หยุดนิ่ง(Stator)ที่มีความสัมพันธ์กันจะเปลี่ยนแปลงตามตำแหน่งของ Rotor ซึ่งขึ้นอยู่กับกระแสที่สเตเตอร์เพียงอย่างเดียวซึ่งต่างจาก มอเตอร์กระแสตรงหรือซิงโครนัสมอเตอร์ที่มีกระแสจากขดลวดสนาม(Field winding) ในการพิจารณโครงสร้างจึงอยู่ในรูปของสมการเชิงอนุพันธ์ที่ค่าสัมประสิทธิ์เปลี่ยนแปลงตามเวลา

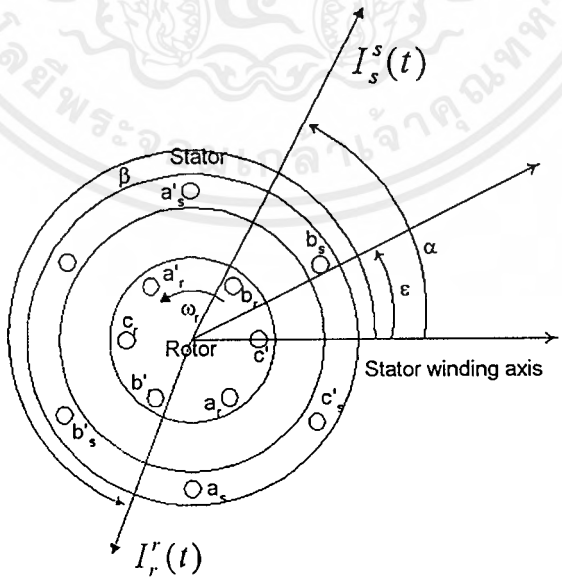
การวิเคราะห์แบบเวคเตอร์ใช้ทฤษฎีพื้นฐานหลัก 3 ทฤษฎีคือ

1.) การสังเคราะห์แบบ Space Vector

2.) วิเคราะห์แบบ 2 แกน (d-q axis Transformation)

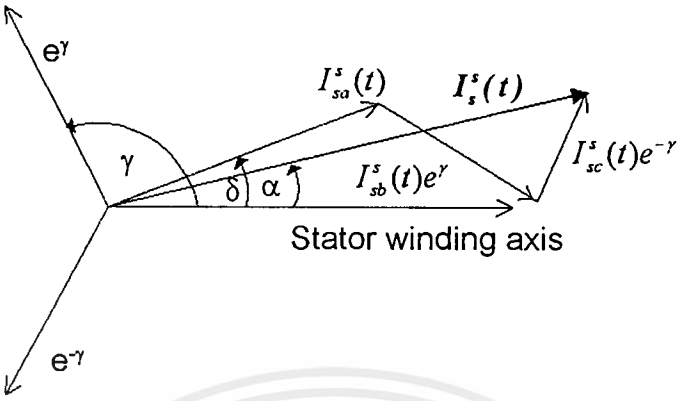
3.) การเปลี่ยนกรอบอ้างอิง (Frame Mapping)

2.2 Space Vector



รูปที่2.1 แสดงโครงสร้างของมอเตอร์เหนียวนำและSpace Vector ของกระแส

การสังเคราะห์แบบ Space Vector เป็นการรวมผลลัพธ์ของกระแส , แรงดัน และสนามแม่เหล็กที่มอเตอร์ได้รับจากแหล่งจ่ายโดยที่ผลลัพธ์ที่ได้อยู่ในรูปของเวคเตอร์ที่มีทั้งขนาดและทิศทาง



รูปที่ 2.2 เฟสเซอร์ของกระแสที่สเตเตอร์บนกรอบอ้างอิงของขดลวดสเตเตอร์

จากรูปสามารถหาค่าของกระแสลัพธ์ได้ดังนี้

$$I^s_s(t) = I^s_{sa}(t) + I^s_{sb}(t)e^{j\gamma} + I^s_{sc}(t)e^{-j\gamma} \quad ....(1)$$

$$I^s_s(t) = I^s_s e^{j\alpha} \quad ....(2)$$

$$I^s_{sa}(t) = I^s_{sa} e^{j(\omega_e t+\theta)} \quad ....(3)$$

$$I^s_{sb}(t) = I^s_{sb} e^{j(\omega_e t+\theta)} \quad ....(4)$$

$$I^s_{sc}(t) = I^s_{sc} e^{j(\omega_e t+\theta)} \quad ....(5)$$

โดยกำหนดให้ $I_{sa}(t)$ มากกว่าศูนย์ ส่วน $I_{sb}(t)$, $I_{sc}(t)$ มีค่าน้อยกว่าศูนย์

จากสมการที่ (1). สามารรถเขียนให้อยู่ในรูปตรีโกณมิติได้ดังนี้

$$I^s_s(t) = I^s_{sa}(t) + I^s_{sb}(t)\left(\cos\gamma + j\sin\gamma\right) + I^s_{sc}(t)\left(\cos(-\gamma) + j\sin(-\gamma)\right) ....(6)$$

2.3 การเปลี่ยนแกน (Transformation Axis)

การจัดรูปสมการให้อยู่ในรูปส่วนจริงและส่วนจินตภาพโดยที่ส่วนจริงจะอยู่ในแกน direct และส่วนจินตภาพจะอยู่ในแกน quadrature ดังนี้

$$I^s_s(t) = I^s_{ds}(t) + jI^s_{qs}(t) \quad ....(7)$$

$$I^s_{ds}(t) = I^s_{sa}(t) + I^s_{sb}(t)\cos\gamma + I^s_{sc}(t)\cos(-\gamma)$$

$$= I^s_{sa}(t) - \frac{1}{2}I^s_{sb}(t) - \frac{1}{2}I^s_{sc}(t) \quad ....(8)$$

$$I^s_{qs}(t) = I^s_{sb}(t)\sin\gamma + I^s_{sc}(t)\sin(-\gamma)$$

$$= \frac{\sqrt{3}}{2}I^s_{sb}(t) + \frac{\sqrt{3}}{2}I^s_{sc}(t) \quad ....(9)$$

จากกระแสที่สมมาตรผลรวมของกระแสทั้งสามเฟสมีค่าเท่ากับศูนย์

$$I^s_{sa}(t) + I^s_{sb}(t) + I^s_{sc}(t) = 0 \quad \text{....(10)}$$

ดังนั้นสามารถเขียนสมการ $I_{sa}(t)$, $I_{sb}(t)$ และ $I_{sc}(t)$ ในรูปของ $I^d_s(t)$ และ $I^q_s(t)$ เป็น

$$I^s_{sa}(t) = \frac{2}{3} I^s_{ds}(t) \quad \text{....(11)}$$

$$I^s_{sb}(t) = -\frac{1}{3} I^s_{ds}(t) + \frac{1}{\sqrt{3}} I^s_{qs}(t) \quad \text{....(12)}$$

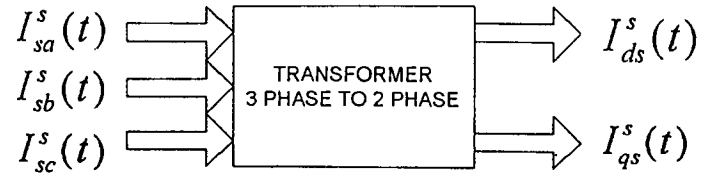
$$I^s_{sc}(t) = -\frac{1}{3} I^s_{ds}(t) - \frac{1}{\sqrt{3}} I^s_{qs}(t) \quad \text{....(13)}$$

จากสมการที่ 8 , 9 , 11 , 12 และ 13 เขียนให้อยู่ในรูปของเมทริกซ์

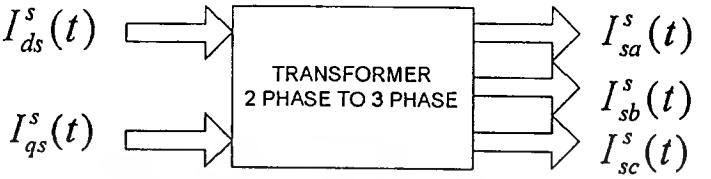
$$\begin{bmatrix} I^s_{ds}(t) \\ I^s_{qs}(t) \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} I^s_{sa}(t) \\ I^a_{sb}(t) \\ I^s_{sc}(t) \end{bmatrix} \quad \text{....(14)}$$

$$\begin{bmatrix} I^s_{sa}(t) \\ I^s_{sb}(t) \\ I^s_{sc}(t) \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & 0 \\ -\frac{1}{3} & \frac{1}{\sqrt{3}} \\ -\frac{1}{3} & -\frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} I^s_{ds}(t) \\ I^s_{qs}(t) \end{bmatrix} \quad \text{....(15)}$$

จากสมการที่ (14) และ (15) สามารถเขียนเป็น Blockdiagram ได้ดังนี้



(a)



(b)

(a) การแปลงแกนของกระแส 3 เฟสเป็นกระแส 2 เฟสที่สเตเตอร์

(b) การแปลงแกนของกระแส 2 เฟสเป็นกระแส 3 เฟสที่สเตเตอร์

รูปที่ 2.3 Block Diagram การแปลงแกน

และเช่นเดียวกันแรงดันและสนามแม่เหล็กก็สามารถแสดงได้ดังสมการ

$$U^s_s(t) = U^a_{sa}(t) + U^s_{sb}(t)e^{j\gamma} + U^s_{sc}(t)e^{-j\gamma} \quad ....(16)$$

$$\Psi^s_s(t) = \Psi^s_{sa}(t) + \Psi^s_{sb}(t)e^{j\gamma} + \Psi^s_{sc}(t)e^{-j\gamma} \quad ....(17)$$

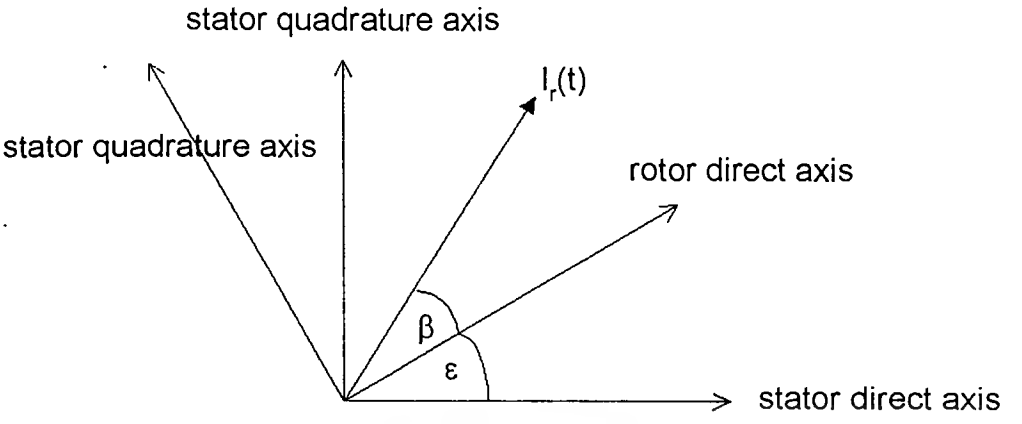
และสามารถเขียนให้อยู่ในรูปของแกน Direct และ Quadrature ได้ดังนี้

$$U^s_s(t) = U^s_{ds}(t) + jU^s_{qs}(t) \quad ....(18)$$

$$\Psi^s_s(t) = \Psi^s_{ds}(t) + j\Psi^s_{qs}(t) \quad ....(19)$$

2.4 การเปลี่ยนกรอบอ้างอิง (Frame Mapping)

ผลของกระแสและสนามแม่เหล็กเกี่ยวคล้องที่เกิดจากสเตเตอร์ทำให้เกิดกระแสและสนามแม่เหล็กที่โรเตอร์ซึ่งกรอบอ้างอิงของสเตเตอร์ (Stator Referenec Frame) และกรอบอ้างของโรเตอร์ (Rotor Reference Frame) ของตัวมอเตอร์นั้นไม่เป็นกรอบเดียวกันเนื่องจากเป็นมอเตอร์ที่มีการกระตุ้นที่เดียว (Single Excited) เฉพาะที่สเตเตอร์เท่านั้น ซึ่งจะมีมุมห่างกันเท่ากับ ε ดังรูป



รูปที่ 2.4 กระแสที่โรเตอร์บนกรอบอ้างอิงสเตเตอร์และโรเตอร์

จากรูปจะได้ค่าของกระแสโรเตอร์ดังนี้

$$I_r^r(t) = I_r^r e^{j\beta} = I_{dr}^r(t) + jI_{qr}^r(t) \quad ....(20)$$

เมื่อพิจารณากระแสโรเตอร์จากกรอบอ้างอิงของสเตเตอร์จะได้ค่าของกระแสโรเตอร์เป็นดังนี้

$$I_r^s(t) = I_r^r e^{j(\beta+\varepsilon)} = I_r^r e^{j\beta} e^{j\varepsilon}$$

$$= \left(I_{dr}^r(t) + jI_{qr}^r(t)\right)\left(\cos\varepsilon + j\sin\varepsilon\right) \quad ....(21)$$

จัดผลคูณให้อยู่ในรูปของส่วนจริงและส่วนจินตภาพ โดยกำหนดให้

$$I_r^s(t) = I_{dr}^s(t) + jI_{qr}^s(t) \quad ....(22)$$

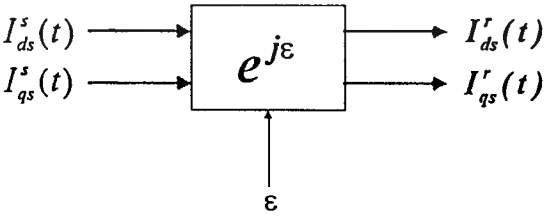
$$I_{dr}^s(t) = I_{dr}^r(t)\cos\varepsilon - jI_{qr}^r(t)\sin\varepsilon \quad ....(23)$$

$$I_{qr}^s(t) = I_{dr}^r(t)\sin\varepsilon + jI_{qr}^r(t)\cos\varepsilon \quad ....(24)$$

สามารถแสดงในรูปของเมตริกได้ดังนี้

$$\left[I_r^s(t)\right] = \begin{bmatrix} I_{dr}^s(t) \\ I_{qr}^s(t) \end{bmatrix} = \begin{bmatrix} \cos\varepsilon & -\sin\varepsilon \\ \sin\varepsilon & \cos\varepsilon \end{bmatrix} \begin{bmatrix} I_{dr}^r(t) \\ I_{qr}^r(t) \end{bmatrix} \quad ....(25)$$

และเขียนในรูปของBlock diagram ได้ดังนี้



รูปที่ 2.5 การเปลี่ยนกรอบอ้างอิงของกระแสจากโรเตอร์ไปยังสเตเตอร์

เช่นเดียวกันกับการเปลี่ยนกรอบอ้างอิงจากสเตเตอร์ไปยังโรเตอร์สามารถเปลี่ยนได้ดังนี้

$$I_s^r(t) = \left(I_{ds}^r(t) + jI_{qs}^r(t)\right)e^{-j\varepsilon} \quad ....(26)$$

$$\left[I_s^r(t)\right] = \begin{bmatrix} I_{ds}^r(t) \\ I_{qs}^r(t) \end{bmatrix} = \begin{bmatrix} \cos\varepsilon & \sin\varepsilon \\ -\sin\varepsilon & \cos\varepsilon \end{bmatrix} \begin{bmatrix} I_{ds}^s(t) \\ I_{qs}^s(t) \end{bmatrix} \quad ....(27)$$

และเมื่อนำการแปลงแกนและการเปลี่ยนกรอบอ้างอิงมารวมกันจะได้

$$\begin{bmatrix} I_{sa}^s(t) \\ I_{sb}^s(t) \\ I_{sc}^s(t) \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\varepsilon & -\sin\varepsilon \\ \cos(\varepsilon-120) & -\sin(\varepsilon-120) \\ \cos(\varepsilon+120) & -\sin(\varepsilon+120) \end{bmatrix} \begin{bmatrix} I_{ds}^r(t) \\ I_{qs}^r(t) \end{bmatrix} \quad ....(28)$$

ผลของกระแสจะเหนี่ยวนำทำให้เกิดสนามแม่เหล็กทั้งที่สเตเตอร์และที่โรเตอร์โดยที่สนามแม่เหล็กเกิดจากค่าความเหนี่ยวนำและกระแสซึ่งสามารถพิจารณาได้เป็น 2 ส่วนคือส่วนที่เกิดจากตัวเองสร้างขึ้นและส่วนที่เกิดจากการเกี่ยวคล้องจากตัวอื่น ๆ รอบบริเวณนั้น และสามารถเขียนเป็นสมการได้ดังต่อไปนี้

$$\Psi_s^s(t) = L_s I_s^s(t) + M I_r^s(t) = L_s I_s^s(t) + M I_r^r(t) e^{j\varepsilon} \quad ....(29)$$

$$\Psi_r^r(t) = L_r I_r^r(t) + M I_s^r(t) = L_r I_r^r(t) + M I_s^s(t) e^{-j\varepsilon} ....(30)$$

$$\Psi_r^s(t) = L_r I_r^s(t) + M I_s^s(t) = L_r I_r^r(t) e^{j\varepsilon} + M I_s^s(t) \quad ....(31)$$

จาก Space Vector สามารถเขียนสมการของแรงดันที่สเตเตอร์ได้ดังนี้

$$U_s^s(t) = R_s I_s^s(t) + \frac{d\Psi_r^r(t)}{dt} \quad ....(32)$$

$$U_r^r(t) = R_r I_r^r(t) + \frac{d\Psi_s^s(t)}{dt} = 0 \quad ....(33)$$

2.5 <u>แรงบิด</u> (Torque)

แรงบิดของมอเตอร์ที่เกิดจากสเตเตอร์และโรเตอร์สามารถเขียนเป็นสมการได้ดังนี้

$$T_e^s(t) = K\left(I_{qs}^s(t) I_{dr}^s(t) - I_{ds}^s(t) I_{qr}^s(t)\right) \quad ....(34)$$

และสามารถเขียนในรูปอย่างง่ายได้ดังนี้

$$T_e^s(t) = K\, Im\left(I_s^s(t) \times I_r^{s*}(t)\right) \quad ....(35)$$

โดยที่ $K = \frac{PM}{3}$

กำหนดให้

$$\Psi_r^s(t) = M I_{mr}^s(t) \quad ....(36)$$

จากสมการที่ (31) และสมการที่ (36) จะได้

$$I_{mr}^s(t) = \frac{L_r I_r^s(t)}{M} + I_s^s(t)$$

$$= I_s^s(t) + (1+\delta_r) I_r^s(t) \quad ....(37)$$

โดยกำหนดให้ δ_r คือ ค่าตัวประกอบความสูญเสียในโรเตอร์ (Rotor Leakage Factor)

$$1+\delta_r = \frac{L_r}{M} \quad \text{....(38)}$$

จากสมการที่ (36)สามารถเขียนสมการของกระแสโรเตอร์บนกรอบอ้างอิงที่สเตเตอร์ได้ดังนี้

$$I_r^s(t) = \frac{I_{mr}^s(t) - I_s^s(t)}{1+\delta_r} \quad \text{....(39)}$$

จากสมการของแรงบิด (34) แทนค่า $I_r^s(t)$ จากสมการ (38) จะได้

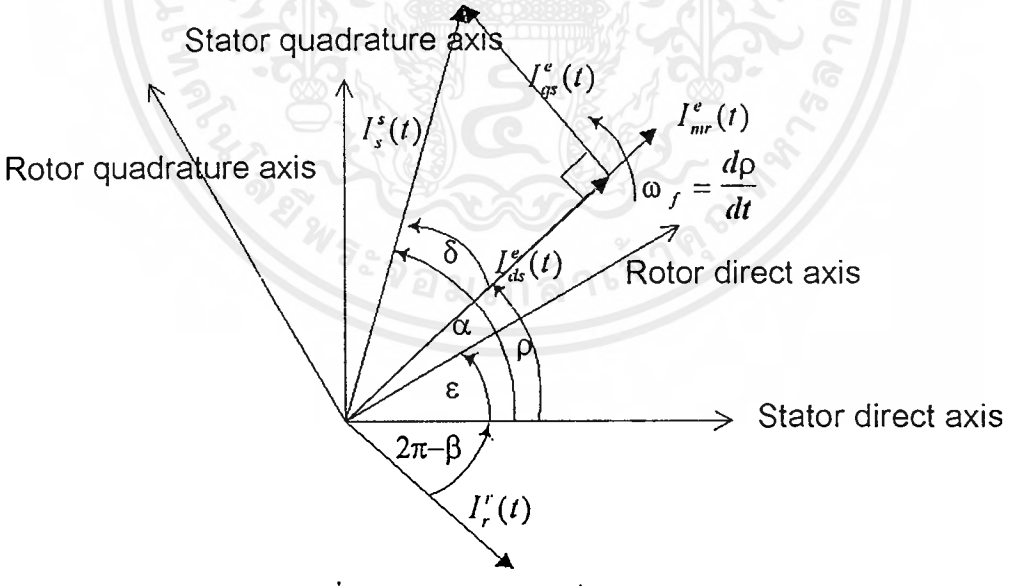
$$T_e^s(t) = K\,Im\left(I_s^s(t) \times \left(\frac{I_{mr}^s(t) - I_s^s(t)}{1+\delta_r} \right)^* \right) \quad \text{....(40)}$$

$$T_e^s(t) = K_1\,Im\left(I_s^s(t) \times I_{mr}^s(t)^* \right) \quad \text{....(41)}$$

โดยที่ $K_1 = \dfrac{PM}{3(1+\delta_r)}$

2.6 กรอบอ้างอิงกระตุ้น (Excitation Frame)

กรอบอ้างอิงกระตุ้นหรือกรอบอ้างอิงเคลื่อนที่ (Moving Frame) หรือ กรอบอ้างอิงสนาม (Field Orientation Frame) เป็นกรอบอ้างอิงที่เคลื่อนที่ตามการตัวหมุน (rotor) หมายความว่าความเร็วสัมพัทธ์ระหว่างกรอบอ้างอิงนี้กับส่วนที่เคลื่อนที่ของมอเตอร์มีค่าเป็น 0



รูปที่ 2.6 แสดงความสัมพันธ์ของกรอบอ้างอิง

$$U_s^s(t) = U_s^e e^{j\omega t} \quad \text{....(42)}$$

$$I_s^s(t) = I_s^e e^{j\omega t} \quad \text{....(43)}$$

กำหนดให้ U_s^e , I_s^e เป็นค่าของกระแสที่กรอบอ้างอิงกระตุ้น (Excitation Frame) และสามารถเปลี่ยนแกนที่สเตเตอร์ไปยังกรอบอ้างอิงกระตุ้นได้ดังนี้

$$U_s^e(t) = U_s^s e^{-j\omega t} \quad \text{....(44)}$$

$$I_s^e(t) = I_s^s e^{-j\omega t} \quad \text{....(45)}$$

จะพบว่ากรอบอ้างอิงกระตุ้นนี้ไม่เปลี่ยนตามเวลาดังนั้นจะพิจารณาสมการและตัวแปรทั้งหมดในกรอบอ้างอิงกระตุ้นสามารถเขียนได้ดังนี้

$$\Psi_s^e = L_s I_s^e + M I_r^e \quad \text{....(46)}$$

$$\Psi_r^e = L_r I_r^e + M I_s^e \quad \text{....(47)}$$

$$U_s^e = R_s I_s^e + \left(p + j\omega\right)\Psi_s^e \quad \text{....(48)}$$

$$U_r^e = R_r I_r^e + \left(p + j\omega_l\right)\Psi_r^e \quad \text{....(49)}$$

$$T_e^e = K\, Im\left(I_s^e I_{mr}^{e\,*}\right) \quad \text{....(50)}$$

2.7 การปรับสนามแม่เหล็กที่โรเตอร์

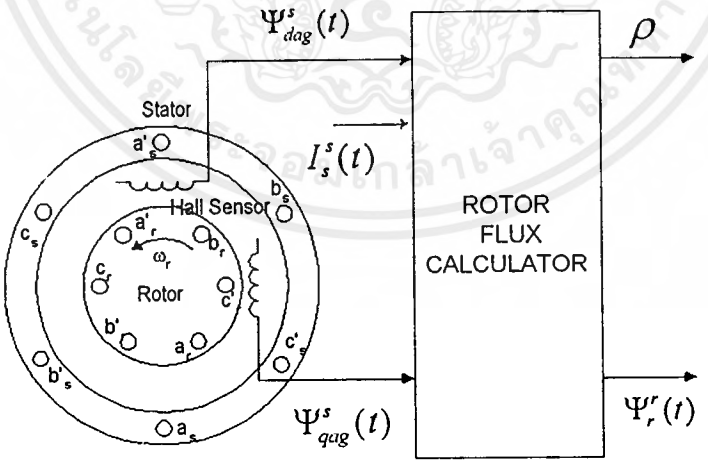
จากสมการที่กล่าวมาข้างต้นนั้นจะเห็นได้ว่าสิ่งสำคัญที่เราต้องทราบคือตำแหน่งที่แท้จริงของกรอบอ้างอิงที่เคลื่อนที่ ซึ่งสามารถหาค่าดังกล่าวได้ 2 วิธีคือ

1.) การหาค่าโดยตรง (Direct Rotor Flux Orientation)

2.) การหาค่าทางอ้อม (Indirect Rotor Flux Orientation)

2.7.1 การหาค่าโดยตรง

สามารถหาค่าของตำแหน่งและขนาดตามลำดับดังรูป



รูปที่ 2.7 แสดงการหาค่าของสนามแม่เหล็กโดยตรง (Direct Rotor Flux Orientation)

วิธีนี้จำเป็นต้องใสเซนเซอร์สนามแม่เหล็กที่ช่องอากาศของตัวมอเตอร์ที่เรียกว่า Hall Sensor โดยที่จะต้องวางไว้ในตำแหน่งที่ 90 องศาทางไฟฟ้าจากรูปเป็นมอเตอร์ชนิด 2 ขั้วถ้าเป็นขดลวดชนิด P ขั้วจะต้องห่างกัน $\frac{\pi}{P}$ การที่ต้องติดตั้ง Sensor นั้นทำในความแข็งแรงของมอเตอร์และความยุ่งยากซับซ้อนและราคาที่สูงขึ้นและในการผลิตมอเตอร์เหนี่ยวนำส่วนใหญ่จะไม่ได้ออกแบบรองรับทำให้เกิดความลำบากในการใช้งานในเชิงอุตสาหกรรม แต่ข้อดีของการควบคุมแบบนี้ก็คือการที่ค่าของสนามแม่เหล็กที่ได้จากตัว เซนเซอร์นั้นไม่เปลี่ยนแปลงตามค่าตัวแปรของมอเตอร์

2.7.2 การควบคุมทางอ้อม

เป็นการหาค่าของตำแหน่งมุมของ กรอบอ้างอิงกระตุ้นจากตำแหน่งของตัวโรเตอร์โดยการต่อเอ็นโคดเดอร์ (Encoder) ในการหาค่าของตำแหน่งแล้วทำการคำนวณหาค่าต่าง ๆ

$$\omega_f = \frac{d\rho}{dt} \qquad ....(51)$$

และ

$$\omega_f = \omega_r + \omega_l \qquad ....(52)$$

จากสมการที่ (46) และ (48) จะได้

$$I_s^e = T_r \frac{dI_{mr}^e}{dt} + (1 + j\omega_l T_r) I_{mr}^e \qquad ....(53)$$

ฉะนั้นจะได้ว่า

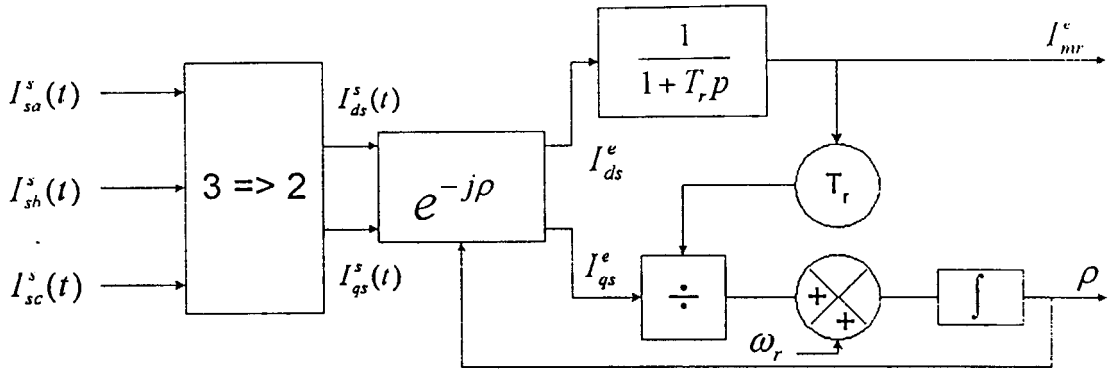
$$I_{ds}^e = T_r \frac{dI_{mr}^e}{dt} + I_{mr}^e \qquad ....(54)$$

$$I_{qs}^e = T_r I_{mr}^e \omega_l \qquad ....(55)$$

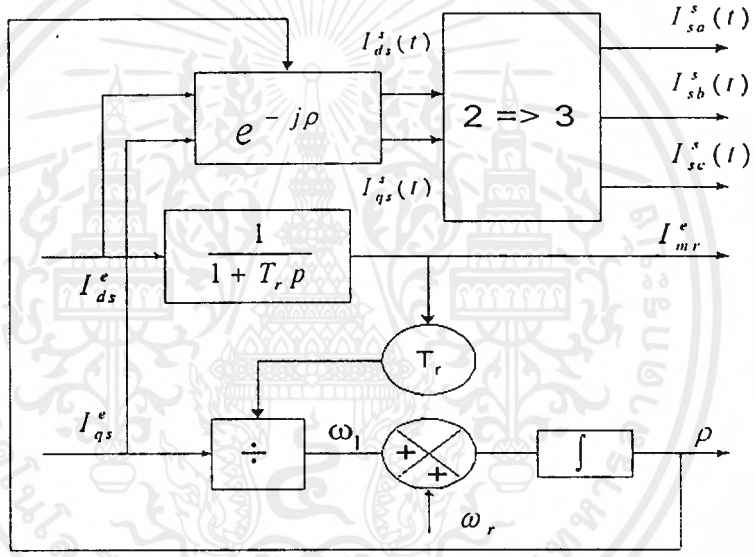
หรือสามารถเขียนในรูปของ I_{mr}^e และω_l

$$I_{mr}^e = \frac{I_{ds}^e}{1 + T_r p} \qquad ....(56)$$

$$\omega_l = \frac{I_{qs}^e}{T_r I_{mr}^e} \qquad ....(57)$$



รูปที่ 2.8 แสดง Block diagram ในการหาค่า I^e_{mr} และ ρ



รูปที่ 2.9 Indirect Rotor Flux Orientation

จากที่ได้กล่าวมาข้างต้นเป็นการควบคุมในส่วนของสนามแม่เหล็กในโรเตอร์เท่านั้นแต่หลักการควบคุมแบบเวคเตอร์สามารถที่จะควบคุมตามค่าในสนามแม่เหล็กได้ 3 แบบ [3]

1.) การควบคุมสนามในโรเตอร์

2.) การควบคุมสนามในสเตเตอร์

3.) การควบคุมสนามในช่องว่างอากาศ

ซึ่งเป็นการเปลี่ยนรูปของสมการแรงบิดให้เป็นค่าของสนามในส่วนต่าง ๆ ทั้งสามส่วน

2.8 การวิเคราะห์คุณสมบัติทางพลวัตของมอเตอร์เหนี่ยวนำกระแสสลับสามเฟส

ในการวิเคราะห์มอเตอร์เหนี่ยวนำกระแสสลับสามเฟส ทั่วไปจะเป็นการวิเคราะห์จากวงจรเสมือน (Equivalent Circuit) ซึ่งเป็นคุณสมบัติในสภาวะอยู่ตัวของมอเตอร์ (Steady State) ที่ความเร็วคงที่และมีแรงดันไฟฟ้าสมดุลกันทั้งสามเฟสเป็นรูปคลื่นไซน์ แต่ในกรณีที่มีการเปลี่ยนแปลงขึ้นกับมอเตอร์อย่างทันทีทันใดเช่นการเพิ่มหรือลดภาระกระทันหัน หรือการเปลี่ยนแปลงความเร็วเป็นต้น จะไม่สามารถวิเคราะห์การเปลี่ยนแปลงของมอเตอร์ได้จึงจำเป็นที่จะต้องวิเคราะห์มอเตอร์แบบพลวัต (Kovecs and Racs ,1959) ซึ่งสามารถแสดงพฤติกรรมของมอเตอร์เมื่อเกิดการเปลี่ยนแปลงทันที่ทันใด หรือในสภาวะคงตัวมีการวิเคราะห์ได้ดังนี้

จากสมการที่ (29) และ (31) จะได้สนามที่ขณะใดขณะหนึ่งเป็น

$$\begin{bmatrix}\Psi_s^s\\ \Psi_r^s\end{bmatrix} = \begin{bmatrix}L_s & M\\ M & L_s\end{bmatrix}\begin{bmatrix}I_s^s\\ I_r^s\end{bmatrix} \qquad ....(58)$$

จากสมการที่ (30) , (31) และ (33) จะได้

$$U_r^s = R_r I_r^s + (p - j\varepsilon)\Psi_r^s \qquad ....(59)$$

จากสมการข้างต้นและสมการที่ (32) จะได้

$$\begin{bmatrix}U_s^s\\ U_r^s\end{bmatrix} = \begin{bmatrix}R_s + pL_s & pM\\ (p - j\varepsilon)M & R_r + (p - j\varepsilon)L_r\end{bmatrix}\begin{bmatrix}I_s^s\\ I_r^s\end{bmatrix} \qquad ....(60)$$

จากสมการที่ได้ยังไม่เหมาะสมกับการวิเคราะห์ทางพลวัตของมอเตอร์เหนี่ยวนำกระแสสลับสามเฟส จึงทำการแยกองค์ประกอบในการวิเคราะห์บนแกม d-q จะได้

$$U_{qs}^s = (R_s + pL_s)I_{qs}^s + pMI_{qr}^s \qquad ....(62)$$

$$U_{ds}^s = (R_s + pL_s)I_{ds}^s + pMI_{dr}^s \qquad ....(61)$$

$$U_{dr}^s = pMI_{ds}^s + \varepsilon MI_{qs}^s + (R_r + pL_r)I_{dr}^s + \varepsilon L_r I_{qr}^s \qquad ....(63)$$

$$U_{qr}^s = -\varepsilon MI_{ds}^s + pMI_{qs}^s - \varepsilon L_r I_{dr}^s + (R_r + pL_r)I_{qr}^s \qquad ....(64)$$

สามารถจัดรูปใหม่ในรูปของเมตริกซ์ได้ดังนี้

$$\begin{bmatrix}U_{ds}^s\\ U_{qs}^s\\ U_{dr}^s\\ U_{qr}^s\end{bmatrix} = \begin{bmatrix}R_s + pL_s & 0 & pM & 0\\ 0 & R_s + pL_s & 0 & pM\\ pM & \varepsilon M & R_r + pL_r & \varepsilon L_r\\ -\varepsilon M & pM & -\varepsilon L_r & R_r + pL_r\end{bmatrix}\begin{bmatrix}I_{ds}^s\\ I_{qs}^s\\ I_{dr}^s\\ I_{qr}^s\end{bmatrix} \qquad ....(65)$$

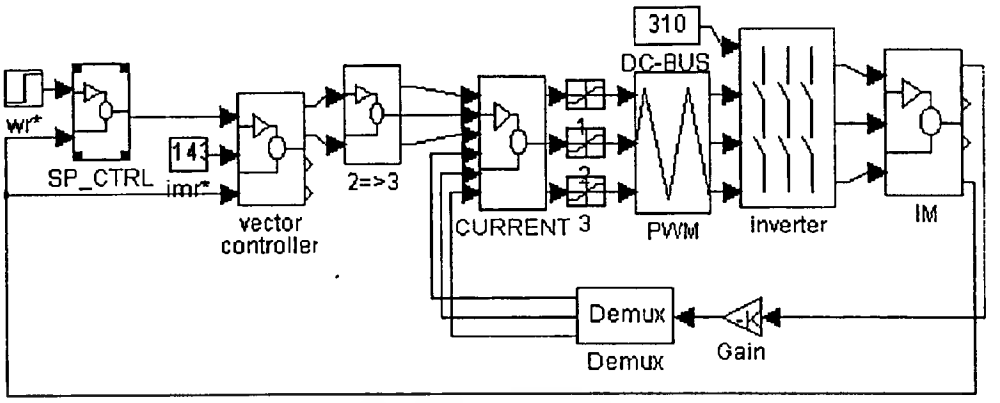
บทที่ 3

การสร้างแบบจำลองการควบคุมมอเตอร์เหนี่ยวนำกระแสสลับแบบเวคเตอร์

จากทฤษฎีของเวคเตอร์คอนโทรลที่ได้กล่าวมาข้างต้น ก่อนที่จะนำมาประยุกต์ใช้งานจริงนั้น จะนำทฤษฎีมาทำการทดสอบ โดยการสร้างแบบจำลองของระบบขึ้น โดยโปรแกรมทางคอมพิวเตอร์ ซึ่งใช้โปรแกรม MATLAB ใช้ในส่วนของ SIMULINK และใช้เครื่องมือที่ได้ถูกพัฒนาขึ้น[5]

ค่าตัวแปรต่าง ๆ ที่นำมาทำการจำลองทั้งที่ได้กำหนดขึ้นและหาได้จากการทดสอบมอเตอร์เพื่อหาวงจรสมมูลมีค่า ดังนี้

พิกัดกำลัง	มีค่าเท่ากับ	2	แรงม้า
แรงดันที่พิกัด	มีค่าเท่ากับ	220	V (Delta)
กระแสที่พิกัด	มีค่าเท่ากับ	6	A
กระแสที่สภาวะไร้ภาระ	มีค่าเท่ากับ	1.87	A
ความเร็วที่พิกัด	มีค่าเท่ากับ	2800	rpm
ความต้านทานของขดลวดที่สเตเตอร์(R_s)	มีค่าเท่ากับ	2	Ω
ความต้านทานของขดลวดที่โรเตอร์(R_r)	มีค่าเท่ากับ	1.559	Ω
ความเหนี่ยวนำที่สเตเตอร์ (L_s)	มีค่าเท่ากับ	197.94	mH
ความเหนี่ยวนำที่โรเตอร์ (L_r)	มีค่าเท่ากับ	197.94	mH
ความเหนี่ยวนำแมกนิไตซิ่ง (M)	มีค่าเท่ากับ	194.3	mH
อัตราขยายตัวควบคุม			
K_p	มีค่าเท่ากับ	25	
K_i	มีค่าเท่ากับ	0.25	
ความถี่ที่ใช้ทดสอบ	มีค่าเท่ากับ	15 - -15Hz	
DC bus	มีค่าเท่ากับ	310	V
pole	มีค่าเท่ากับ	2	



รูปที่ 3.1 การจำลองระบบควบคุมมอเตอร์เหนี่ยวนำแบบเวคเตอร์

จากรูปที่ 10. ประกอบด้วย blockที่สำคัญดังนี้

3.1 การควบคุมความเร็วใน

รูปที่ 3.2 วงรอบความเร็ว

สัญญาณขาเข้าอันที่ 1 เป็นสัญญาณคำสั่งความเร็ว ส่วนอีกอันหนึ่งเป็นสัญญาณความเร็วของมอเตอร์ จะนำมาลบกัน เป็นสัญญาณความคลาดเคลื่อนและจะผ่านไปยังส่วนควบคุม PI จะเป็นสัญญาณกระแส $I^r_{qs}(t)$ และมีการควบคุมด้วยตัวจำกัดกระแส

3.2 vector control block [5]

input : $I^r_{ds}(t)\,.\,I^r_{qs}(t)\,.\,\omega_r$

output : $I^s_{ds}(t)\,.\,I^s_{qs}(t)\,.\,\omega_s\,.\,\varepsilon$

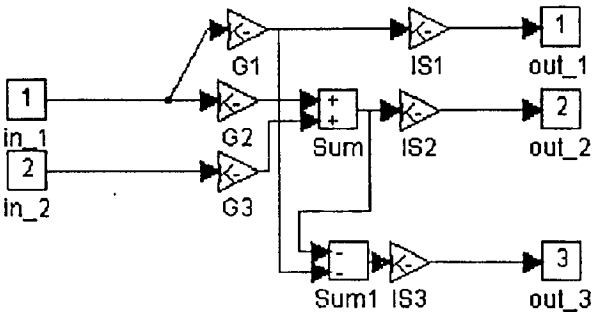
ทำหน้าที่ในการเปลี่ยนกรอบอ้างอิงจากโรเตอร์เป็นกรอบอ้างอิงที่สเตเตอร์ตามความสัมพันธ์ในสมการที่ (27) และมุมของโรเตอร์หาได้จากสมการที่ (52) และ (55)

3.3 การแปลงแกน

input : $I^s_{ds}(t)\,.\,I^s_{qs}(t)$

output . $I^s_{sa}(t)\,.\,I^s_{sb}(t)\,.\,I^s_{sc}(t)$

เป็นการแปลง แกน direct และ quadrature เป็นกระแสทั้งสามเฟสดังสมการที่ (15)



รูปที่ 3.3 การแปลงแกน

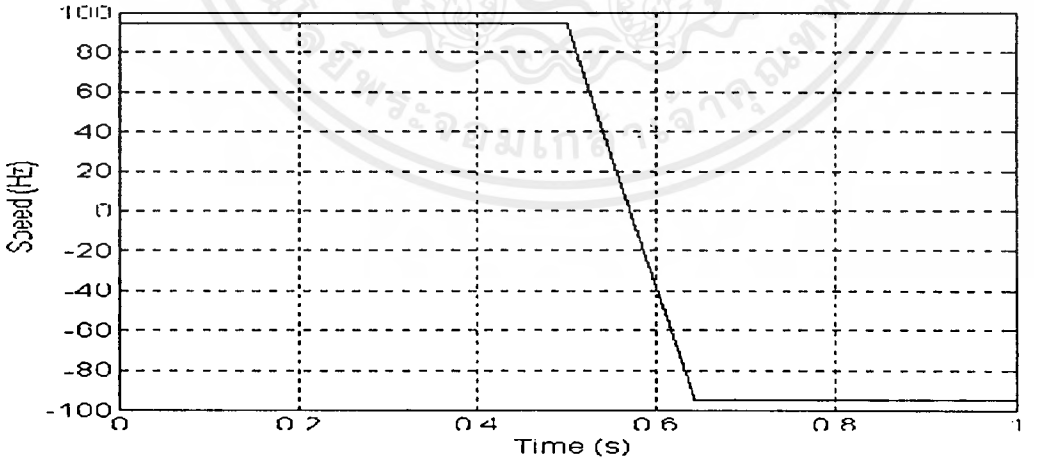
3.4 การเปรียบเทียบกระแส และสร้างสัญญาณ PWM ขับมอเตอร์ [3]

ทำหน้าที่ในการเปรียบเทียบกระแสคำสั่งควบคุมกับกระแสที่มอเตอร์ โดยผ่านการควบคุมแบบ PI และนำค่าความคลาดเคลื่อนที่ได้ไปสร้างสัญญาณ PWM เรียกวิธีนี้ว่า Subharmonic หลังจากนั้นจะนำสัญญาณที่ได้ส่งไปยัง INVERTER [3] และ ขับมอเตอร์เหนี่ยวนำกระแสสลับ[3] ต่อไป

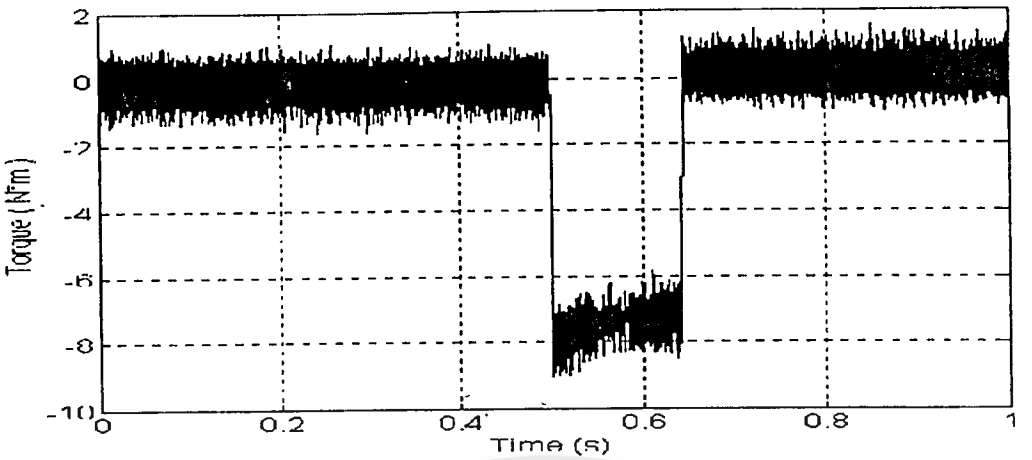
3.5 ผลการจำลอง

3.5.1 การกลับทิศทางการหมุน

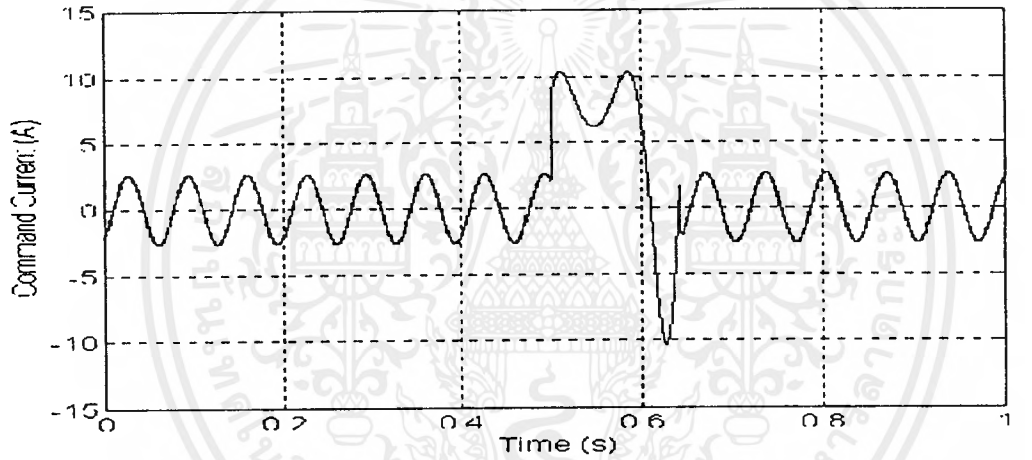
เมื่อได้แบบจำลองแล้วจะทำการจำลองการทำงานโดยการป้อนคำสั่งความเร็วแบบขั้นขันได โดยเปลี่ยนแปลงจาก 15 Hz เป็น -15 HZ ที่เวลา 1 วินาที จะได้ผลการจำลองดังนี้



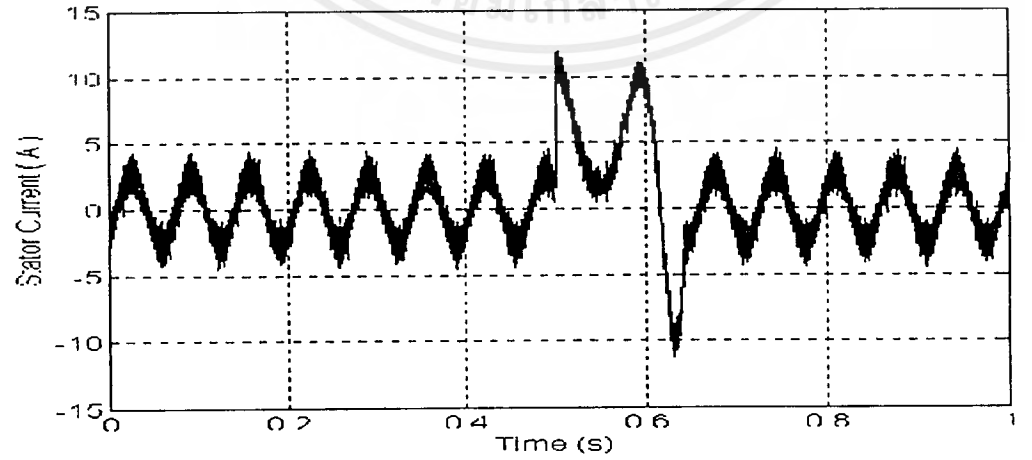
รูปที่ 3.4 แสดงความเร็วของโรเตอร์ขณะกลับทิศทางหมุนที่สภาวะไร้ภาระ



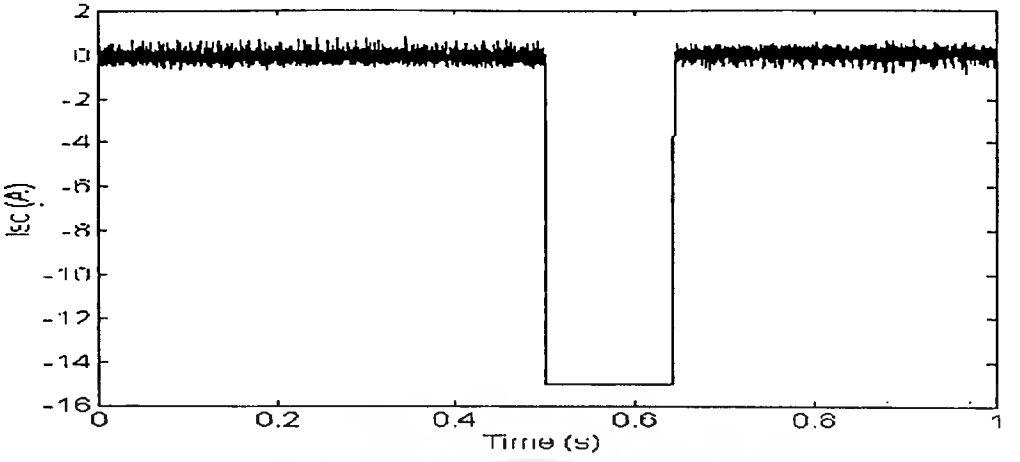
รูปที่ 3.5 แสดงแรงบิด (TORQUE) ขณะกลับทิศทางหมุนที่สภาวะไร้ภาระ



รูปที่ 3.6 แสดงคำสั่งกระแสที่สร้างขึ้นจากไมโครคอนโทรลเลอร์ขณะกลับทิศทางหมุนที่สภาวะไร้ภาระ



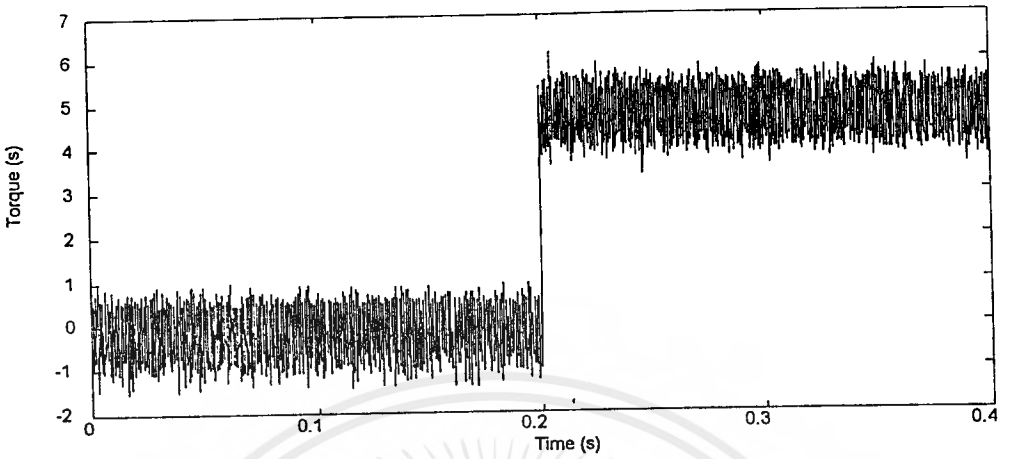
รูปที่ 3.7 แสดงกระแสที่สเตเตอร์ขณะกลับทิศทางหมุนที่สภาวะไร้ภาระ



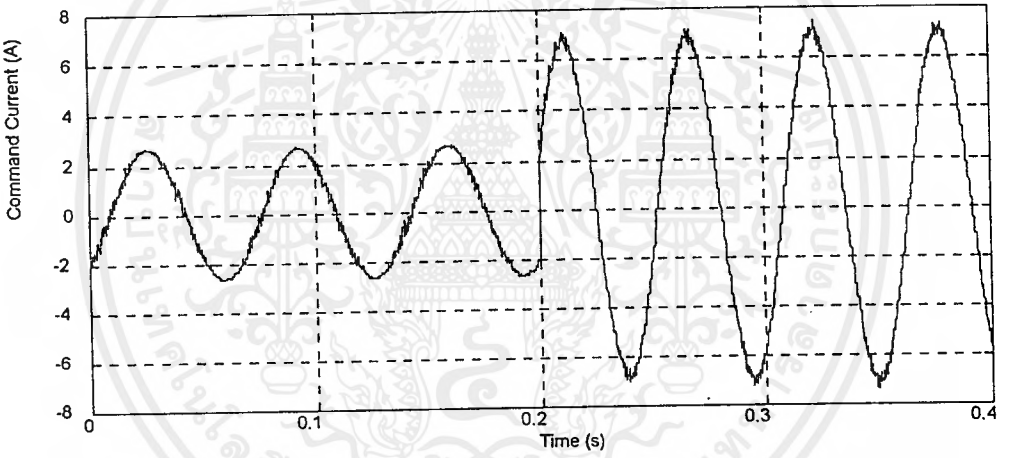
รูปที่ 3.8 แสดงกระแส $I_{qs}^{r}(t)$ เป็นคำสั่งแรงบิดขณะกลับทิศทางหมุนที่สภาวะไร้ภาระ

จากผลการทดลอง พบว่าที่สภาวะไม่มีภาระกระแส $I_{qs}^{r}(t)$ มีค่าเป็น 0 คือไม่เกิดแรงบิด เมื่อมอเตอร์เริ่มกลับทิศกระแส $I_{qs}^{r}(t)$ เป็นลบ แรงบิดก็จะกลับทิศเช่นเดียวกันจะเห็นว่าการเปลี่ยนของกระแสและแรงบิดดีมาก คือสามารถเปลี่ยนแปลงทันที กระแสที่สเตเตอร์ก็จะเพิ่มขึ้นและความถี่มีค่าลดลงจนกระทั่งเริ่มกลับทิศการหมุนความถี่จะสูงขึ้นจนถึงความเร็วที่ต้องการ ขนาดจะลดลงเท่ากับที่ไม่มีภาระและความถี่คงที่ แรงบิดจะตกลงและคงที่จนได้ความเร็วที่ต้องการและจะลดลงเป็นศูนย์ การกลับทิศทางหมุนที่ความถี่ 15 Hz นี้ใช้เวลาประมาณ 140 มิลิวินาที

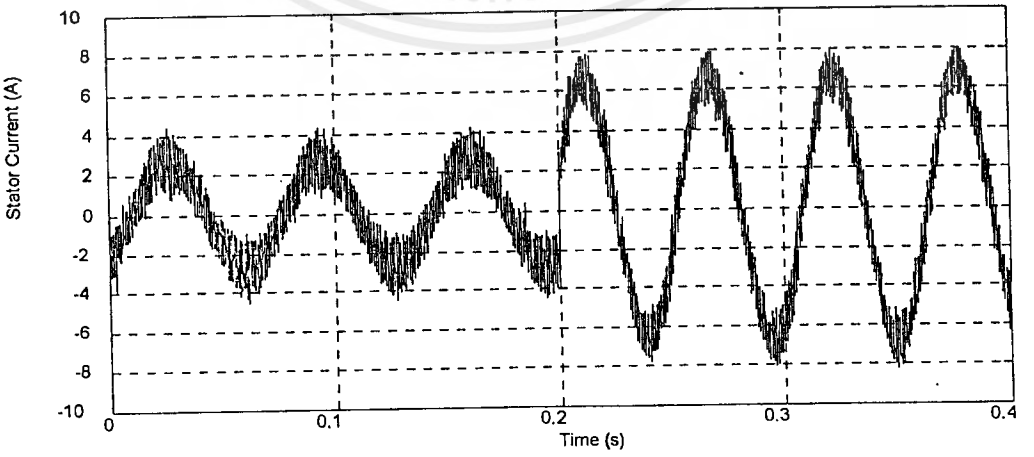
3.5.2 ผลการจำลองขณะมีการเปลี่ยนแปลงภาระโดยการเพิ่มจากสภาวะไม่มีภาระ จนกระทั่งถึงแรงบิด 5 N*m ได้ผลดังนี้



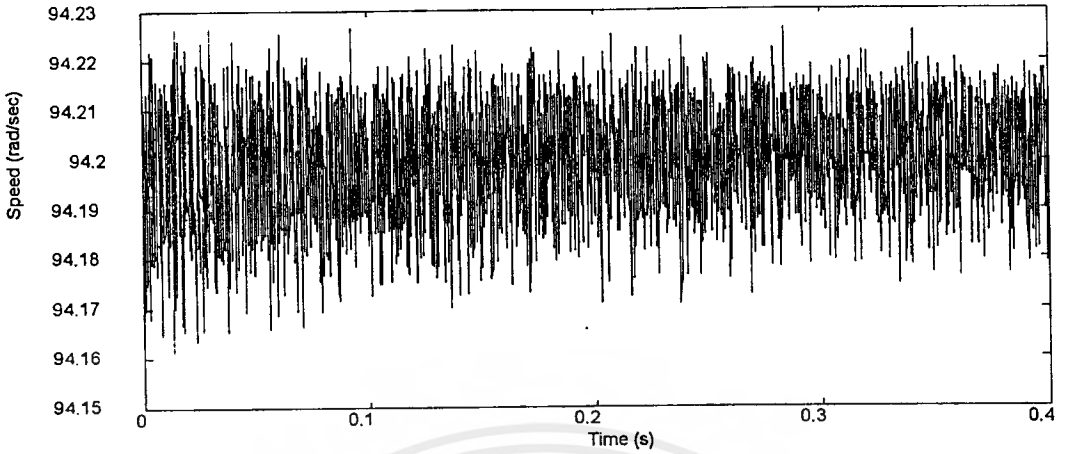
รูปที่ 3.9 แสดงแรงบิดเมื่อมีการเพิ่มภาระเป็น 5 N*m



รูปที่ 3.10 แสดงคำสั่งกระแสเมื่อมีการเพิ่มภาระเป็น 5 N*m



รูปที่ 3.11 แสดงกระแสจริงที่สเตเตอร์เมื่อมีการเพิ่มภาระเป็น 5 N*m



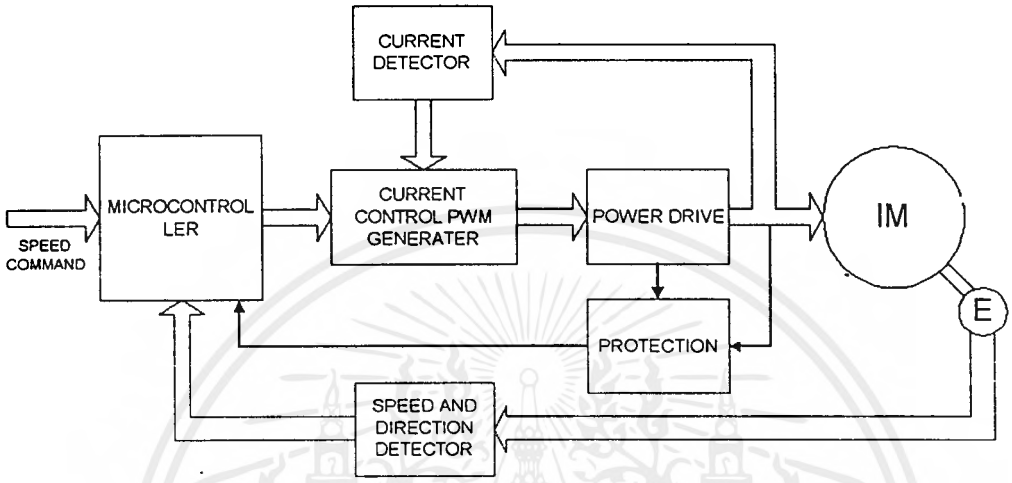
รูปที่ 3.12 แสดงความเร็วเมื่อมีการเพิ่มภาระเป็น 5 N*m

จะเห็นได้ว่าเมื่อมีการเปลี่ยนแปลงภาระเกิดขึ้นขนาดของกระแสที่สเตเตอร์จะมีค่าสูงขึ้น และคตวามถี่ก็มีค่าสูงขึ้นด้วยเพื่อสร้างแรงบิดให้สามารถรับภาระที่เพิ่มขึ้นได้จะเห็นได้ว่าความเร็ค่อนข้างคงที่เพราะเราต้องการรักษาความเร็วให้มีค่าคงที่

บทที่4

การประยุกต์ใช้งาน

การควบคุมที่นำมาประยุกต์นั้นเป็นแบบปรับค่าตามสนามแม่เหล็กที่โรเตอร์ทางอ้อมดังรูป



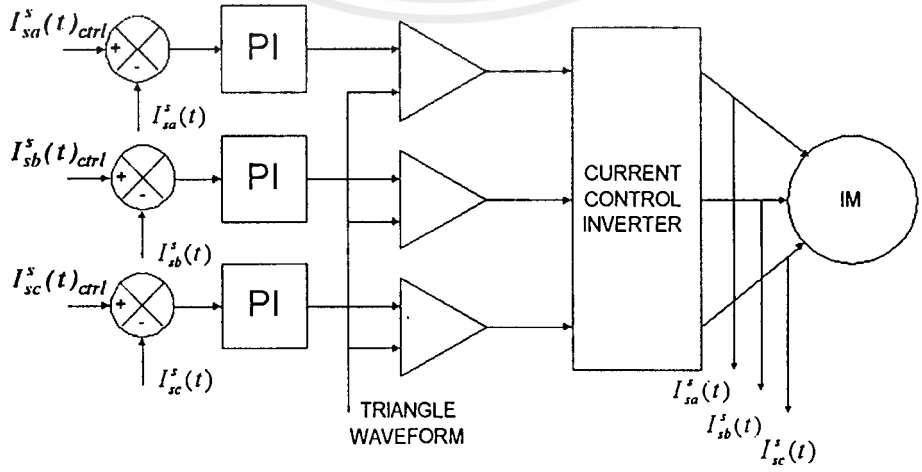
IMPLEMENTATION OF VECTOR CONTROL BLOCK DIAGRAM

รูปที่ 4.1 แสดงการประยุกต์ใช้งานการควบคุมแบบเวคเตอร์

จากรูปข้างต้นแบ่งการทำงานได้ดังนี้

1.) power drive ประกอบด้วยอุปกรณ์ switching สารกึ่งตัวนำใช้ IGBT เบอร์ 6MB20L-060 มีพิกัดแรงดัน 600 V พิกัดกระแส 20 A ในหนึ่งโมดูลจะประกอบด้วย IGBT 6 ตัว ต่อวงจรแบบบริดจ์ สามเฟส [1] และวงจรขับเกตใช้ไอซีไอบริดจ์เบอร์ EXB841 ของบริษัท FUJI[1]

2.) ส่วนการควบคุมกระแสและสร้างสัญญาณ PWM [1] และใช้ ไอซีเบอร์ ICL8038 [1] สร้างสัญญาณคลื่นสามเหลี่ยมดังรูป



รูปที่ 4.2 อินเวอร์เตอร์แบบควบคุมกระแส

ทำหน้าที่ในการเปรียบเทียบสัญญาณคำสั่งกระแสกับกระแสที่สเตเตอร์และค่าความคลาดเคลื่อนที่ได้และสัญญาณที่คลาดเคลื่อนจะผ่านการควบคุมแบบ PI และทำการเปรียบเทียบกับสัญญาณคลื่นสามเหลี่ยมจะได้สัญญาณกระแสที่เป็นPWM และนำไปขับอินเวอร์เตอร์

3.) วงจรวัดค่ากระแส (current detector) [3] สามารถวัดได้ตั้งแต่ความถี่ 0 - 150 KHz โดยช่วงกระแสตั้งแต่ 0 - 41.152 A เท่ากับช่วงของแรงดัน 0 - 10 V ซึ่งอัตราการขยายกระแสของอุปกรณ์วัดกระแส มีอัตรา 1 : 1000

4.) วงจรป้องกัน (PROTECTION) [1] ทำหน้าที่ในการป้องกันกระแสเกินพิกัด และกระแสลัดวงจรโดยการรับสัญญาณจาก EXB841 และแสดงผลที่หน้าจอ LCD

6.) วงจรวัดความเร็วของเอ็นโคดเดอร์ โดยที่เอ็นโคดเดอร์เป็นแบบ Incremental โดยมีอัตรา 2000 pulse / rev เมื่อผ่านวงจรก็จะเพิ่มความถี่เป็นสองเท่า และทำการบอกทิศทางเป็น logic และมีสัญญาณอนาลอก บวกลบ 5 V บอกความเร็วรอบ ตั้งแต่ -3000 rev /s ถึง 3000 rev /s

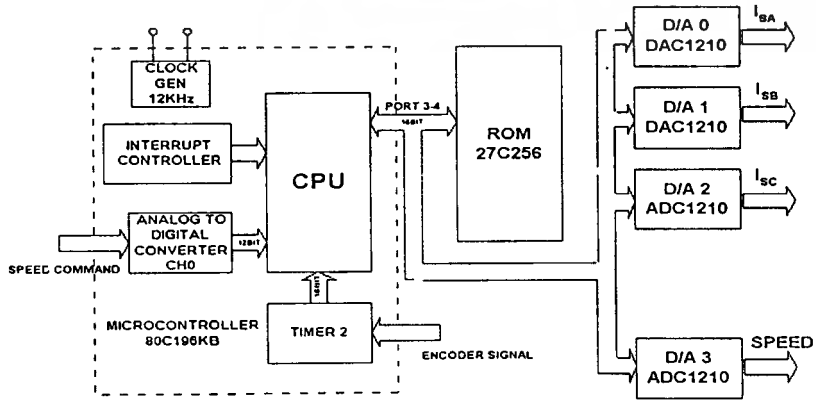
7.) MICROCONTROLLER

ทำหน้าที่ในการรับคำสั่งกระแสและความเร็วของมอเตอร์เพื่อสร้างสัญญาณกระแสสามเฟสเพื่อส่งไปควบคุมต่อยังส่วนของการควบคุมกระแสและทำหน้าที่เป็นส่วนที่แสดงผลที่หน้าจอภาพ LCD เมื่อเกิดการลัดวงจร หรือรับภาระมากเกินไป

บอร์ดไมโครคอรโทรลเลอร์ 80C196KB มีคุณสมบัติดังนี้

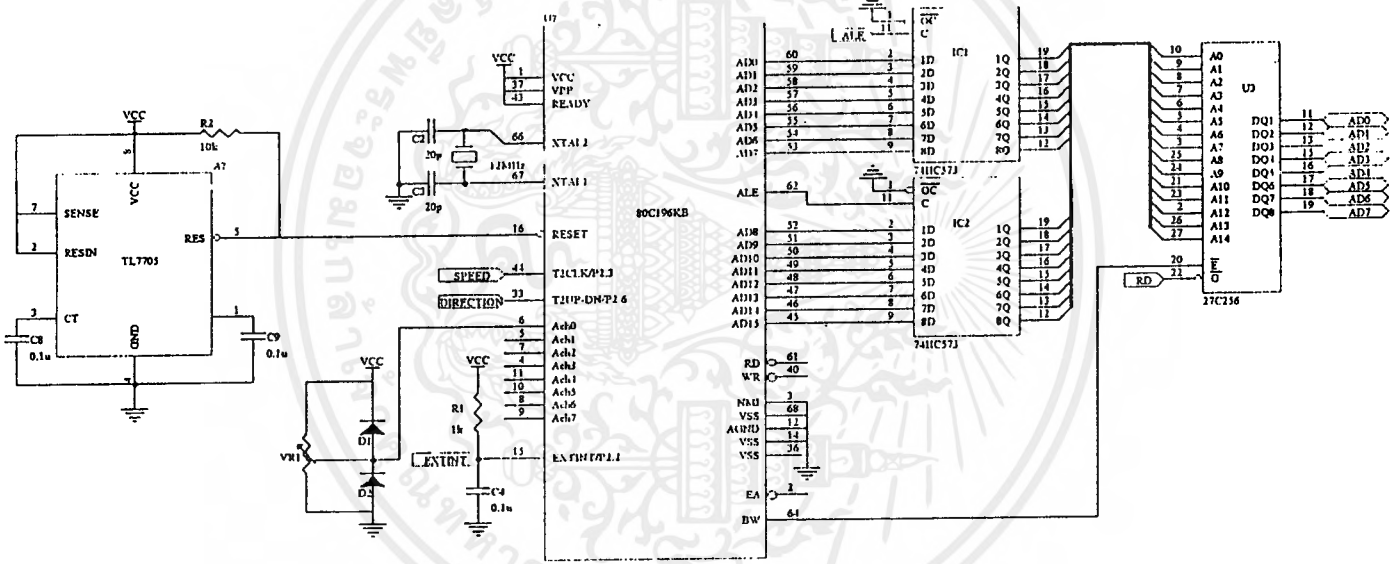
- มีขนาด 16 บิตมีความเร็วในการทำงานเท่ากับ 12 MHz
- มี 8 บิท input port สำหรับสวิทช์ควบคุม , ส่วนป้องกันกระแสเกิน และการรับภาระเกินพิกัด
- output port สำหรับ LCD และสัญญาณในการขับเกต
- มี D/A 12 บิท 4 ช่อง สำหรับสร้างสัญญาณกระแสและส่งค่าความเร็วและแรงบิดของมอเตอร์ มีค่าระหว่าง -10 - 10 V
- หน่วยความจำ ใช้ ROM 32 Kbyte

จากคุณสมบัติดังกล่าวสามารถสร้างวงจรได้ดังนี้



BLOCK DIAGRAM SINGLE BOARD MICROCONTROLLER 80C196KB

รูปที่ 4.3 บอร์ดไมโครคอนโทรเลอร์



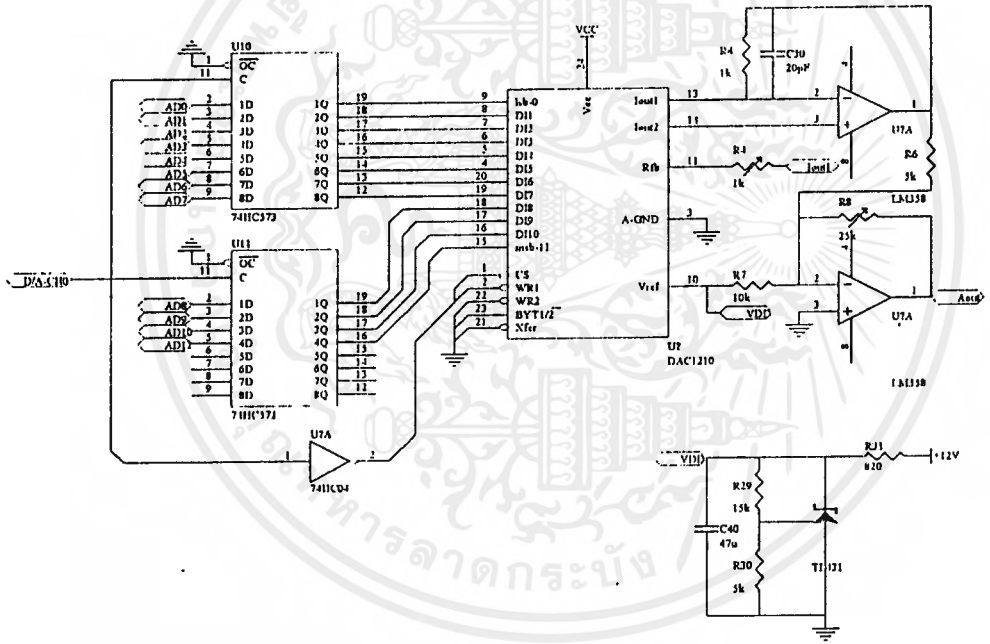
Title: Microcontroller With External Memory

Size: B | Number | Revision

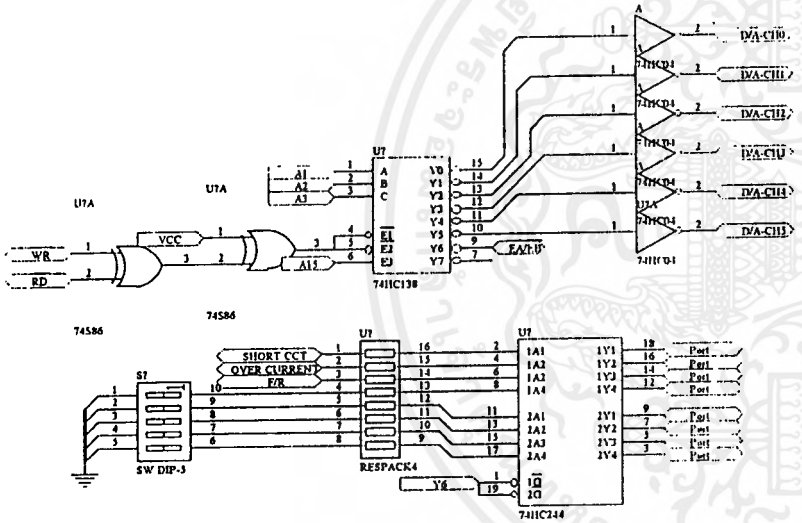
Date: 13-Oct-1998 | Sheet of

File: E:\PROJECT\SCH\EPROM.SCH | Drawn By:

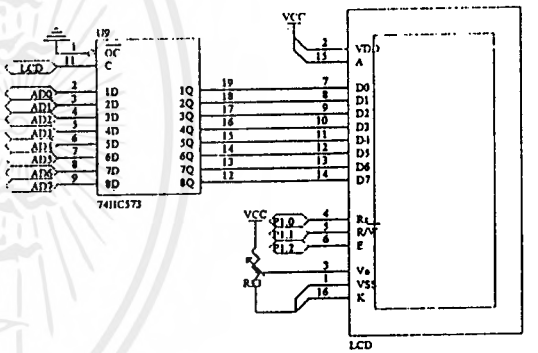
page 20



Title	D/A Converter Circuit		
Size: B	Number: 2	Revision:	
Date: 13-Oct-1998	Sheet of		
File: F:\PROJECT\SCH\DAC.SCH	Drawn By:		



DECODER



LCD

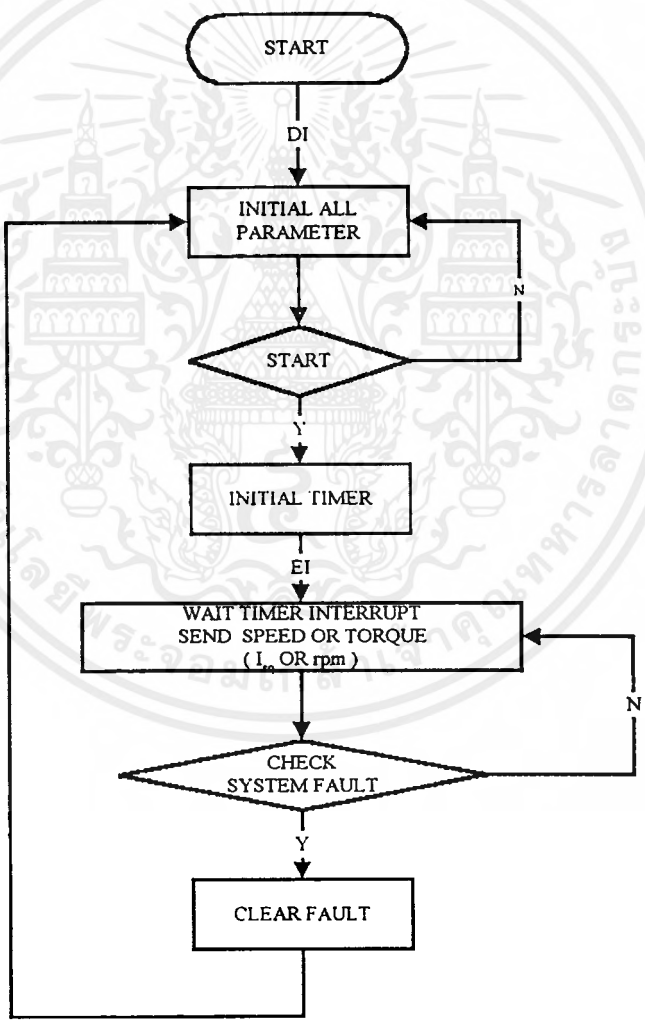
Title	Decoder circuit and LCD	
Size: B	Number	Revision
Date: 13-Oct-1998	Sheet of	
File: D:\USERS\TERADA\CIT\THESIS\SEL.EX	Drawn By:	

บทที่ 5

การเขียนโปรแกรมควบคุมการทำงาน

การเขียนโปรแกรมใช้ภาษาแอสเซมบลี้โดยใช้แอสเซมเบอร์ cross32 และแบ่งโปรแกรมเป็นส่วน ๆ ได้ดังนี้

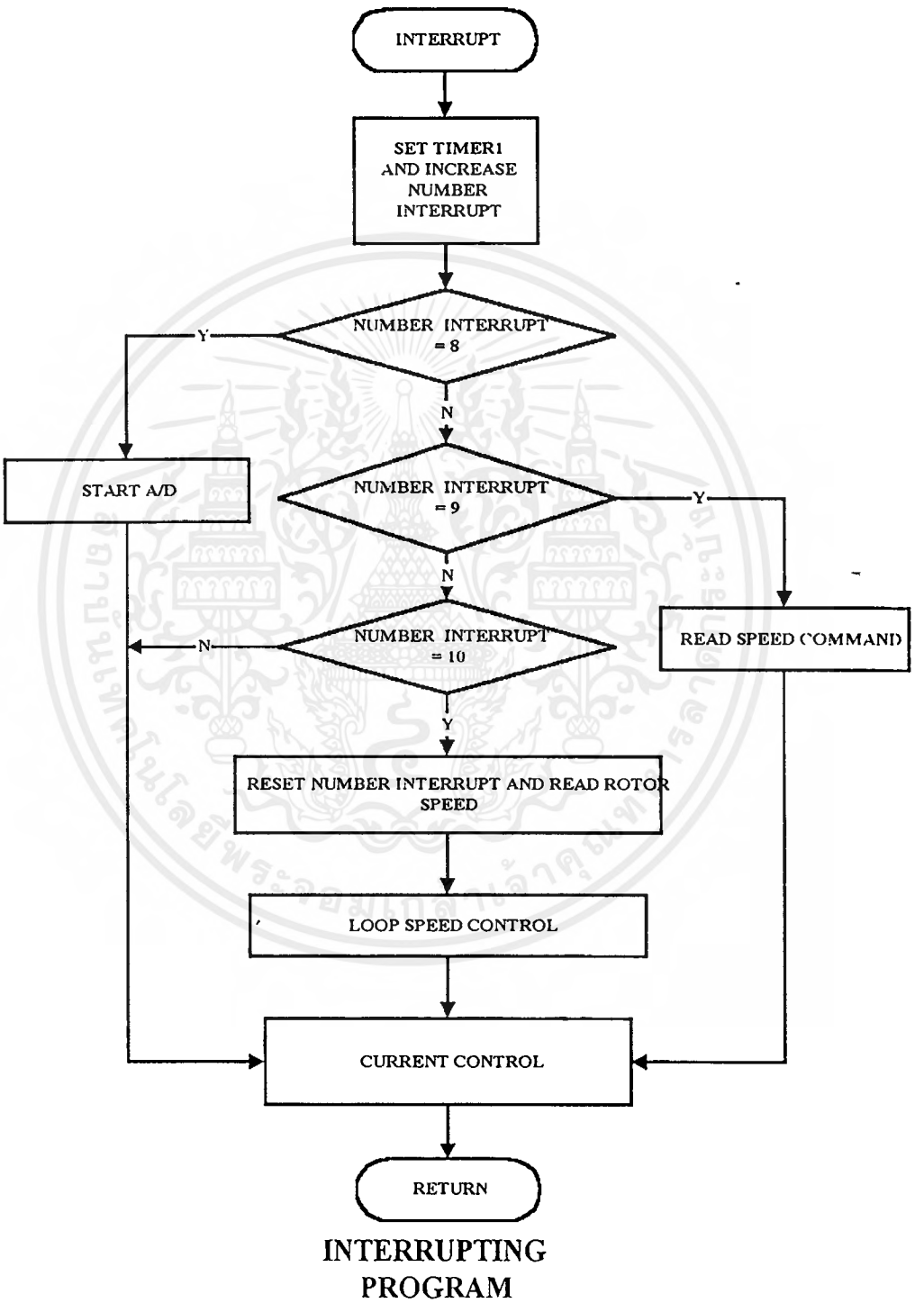
1.) โปรแกรมหลัก ทำหน้าที่ในการตั้งค่าเริ่มต้น ตั้งเวลาในการขัดจังหวะและทำการส่งค่าความเร็วหรือแรงบิดออกมายัง ช่องสัญญาณที่ 3 และ 4 และถ้าระบบมีการทำงานที่ผิดปกติก็จะหยุดการทำงานและรอสัญญาณเมื่อจัดการความผิดพลาดเสร็จแล้วก็จะกลับไปทำงานต่อ ในช่วงการทำงานก็จะรอการขัดจังหวะ ถ้ามีการขัดจังหวะเกิดขึ้นก็ให้ไปทำงานตามที่เวคเตอร์การขัดจังหวะชี้ไปและเมื่อเสร็จก็จะกลับมาทำงานที่ก่อนการขัดจังหวะต่อดังรูป



MAIN PROGRAM

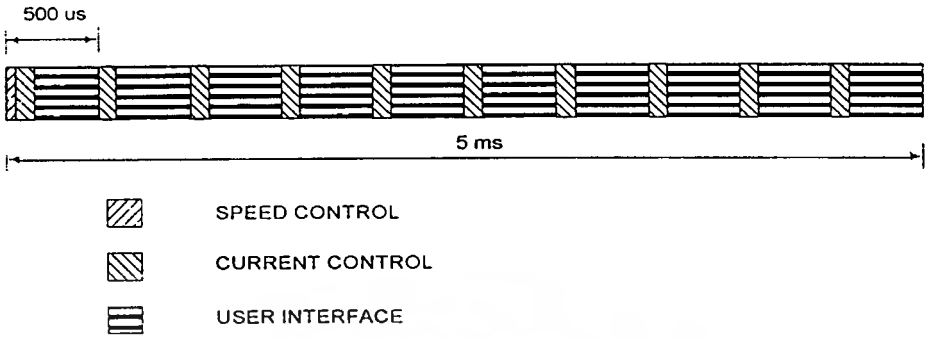
รูปที่ 5.1 ผังการทำงานของโปรแกรมหลัก

2.) เมื่อเกิดการขัดจังหวะจาก timer1 ขึ้น เวคเตอร์การขัดจังหวะจะชี้ไปยังตำแหน่ง 2200 และจะเริ่มทำงานในส่วนของงานในการขัดจังหวะ โดยที่จะมีการอ่านค่าความเร็วและทำคำสั่งความเร็วทุก ๆ 5ms แต่จะคำนวณคำสั่งกระแสทุก ๆ 500 us ดังรูป



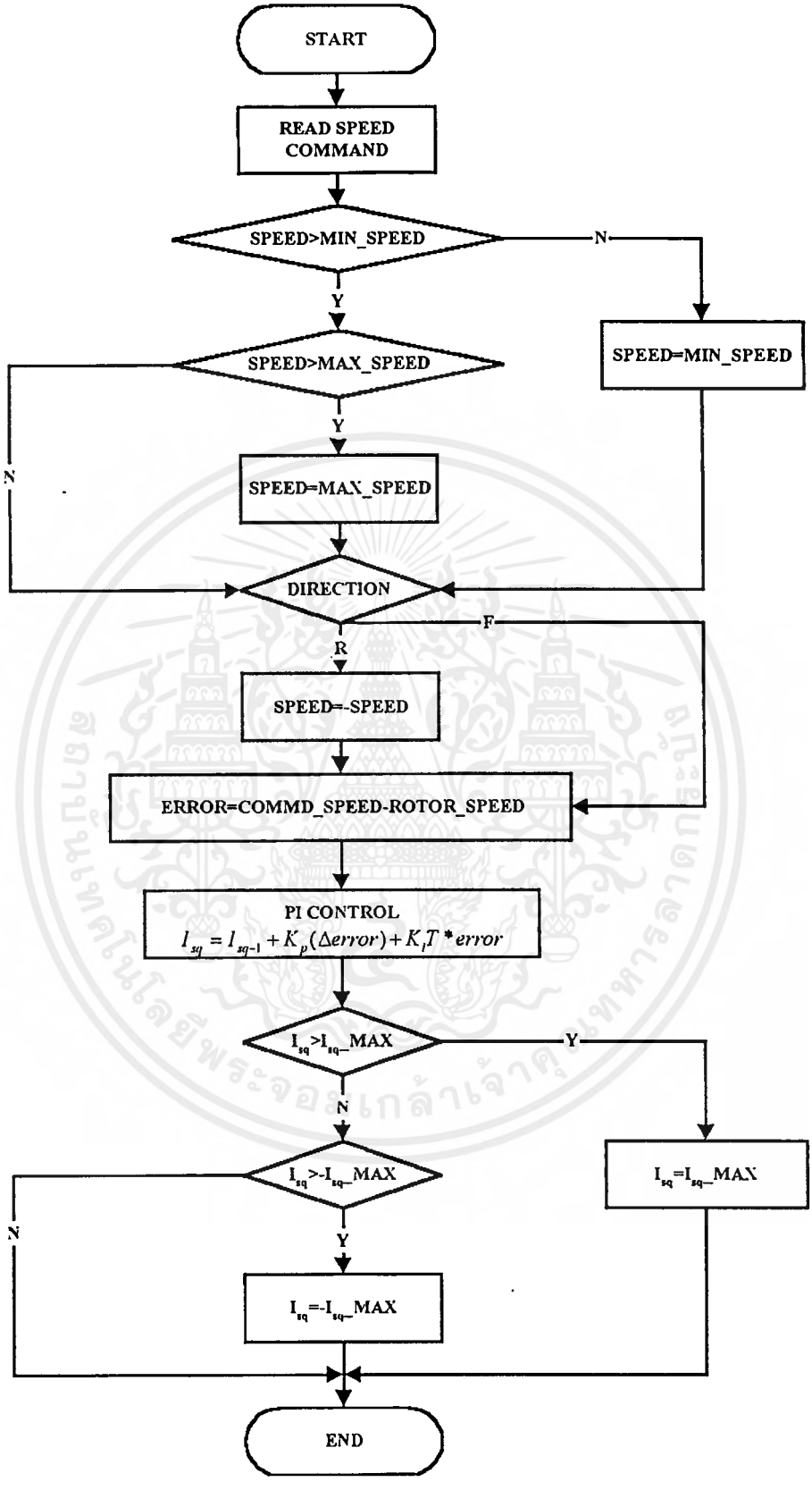
รูปที่ 5.2 ผังการทำงานของการขัดจังหวะ

การขัดจังหวะจะอาศัยการเกิด Overflow ของ Timer1 ในการขัดจังหวะแต่ละครั้งจะมีตัวนับการขัดจังหวะเพื่อที่จะอ่านค่าความเร็วที่ทุก ๆ 5 มิลิวินาที ซึ่งตารางเวลาการทำงานแสดงได้ดังนี้



รูปที่ 5.3 แสดงตารางเวลาในการทำงานของโปรแกรม

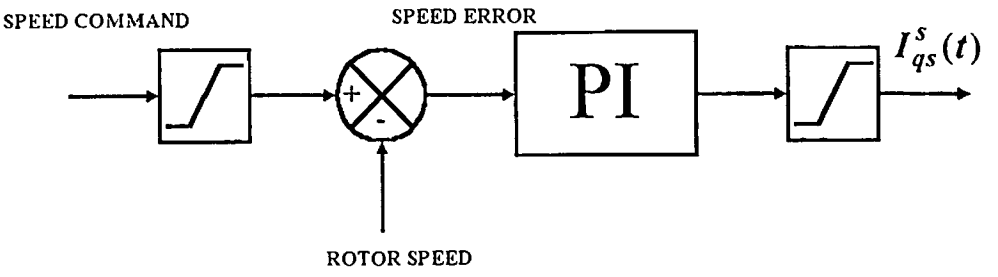
3.) speed loop จะทำการอ่านคำสั่งตรวจสอบค่าที่ได้รับทั้งขนาดและตำแหน่งและทำการควบคุมแบบ PI ซึ่งก็จะได้ค่ากระแส $I_{qs}^{r}(t)$ ดังรูป



SPEED CONTROL

รูปที่ 5.4 การทำงานวงรอบความเร็ว

จาก Flowchart สามารถเขียนเป็น Block diagram ได้ดังนี้



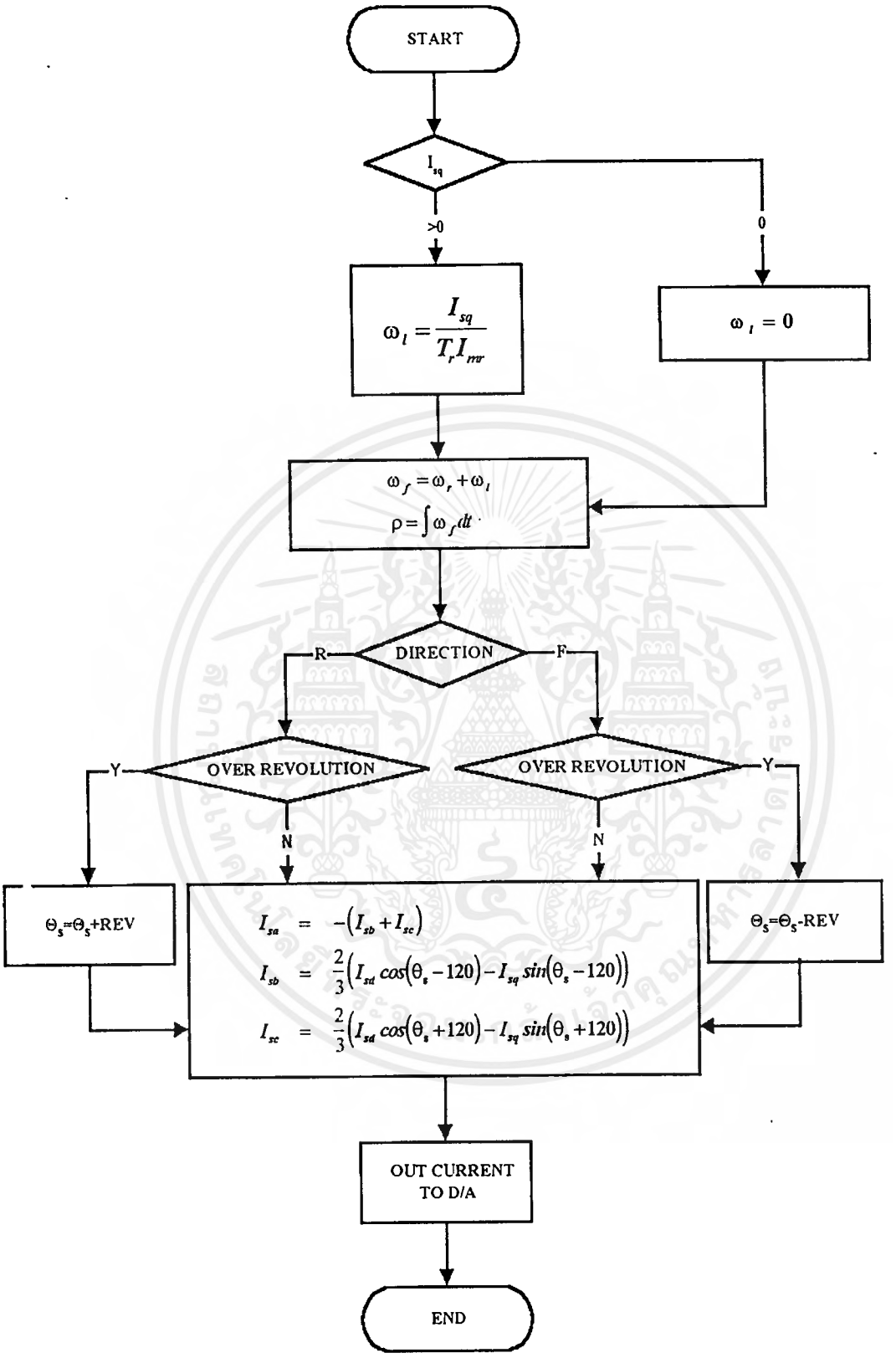
รูปที่ 5.5 แสดง Block diagram ของวงรอบความเร็ว

เริ่มโดยการรับคำสั่งความเร็วจาก A/D แล้วตรวจสอบค่าที่ได้ว่าอยู่ในช่วงที่กำหนดหรือไม่ ในที่นี้กำหนดค่าความเร็วอยู่ในย่าน $\pm$ 50 Hz แล้วนำคำสั่งความเร็วมาเปรียบเทียบกับความเร็วของโรเตอร์ โดยการควบคุมแบบ PI แล้วส่งคำสั่งกระแส $I_{qs}^{r}(t)$ ตมสมการดังนี้

$$I_{sq} = I_{sq-1} + K_p(\Delta error) + K_i T * error \quad ....(66)$$

และกำหนดให้กระแสมีค่าอยู่ในช่วง $\pm$ 8.84 A_{peak}

4.) การคำนวณกระแส เมื่อได้ $I_{qs}^{r}(t)$ จะนำไปหาค่าความเร็วของสลิปและนำไปรวมกับความเร็วของโรเตอร์ก็จะได้ความเร็วเชิงมุมของกระแส นำไปตรวจสอบทิศทางและขนาดและนำค่า $I_{ds}^{r}(t)$, $I_{qs}^{r}(t)$ และมุมที่คำนวณได้แทนค่าในสมการที่ (28) ก็จะได้กระแสทั้งสามเฟสออกมาดังรูป การคำนวณหากระแสทั้งสามเฟสค่าของ ไซน์ และ โคไซน์จะหาได้จากการเปิดตารางไซน์ โดยที่ตารางค่าไซน์ที่ใช้กำหนดให้มีความละเอียดเท่ากับ 1000 จุดต่อ 1 รอบ และขนาดจะกำหนดให้มีขนาดเป็น 2/3 เท่าของค่าสูงสุด ฉะนั้นสัญญาณไซน์ที่สร้างได้จะมีลักษณะเป็นขั้น ๆ



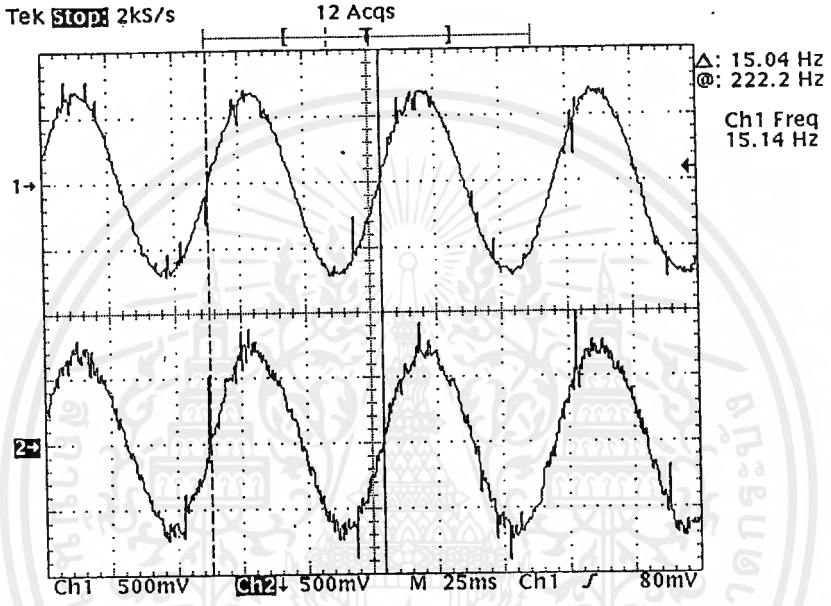
CURRENT CONTROL

รูปที่ 5.6 ผังการทำงานของการควบคุมกระแส

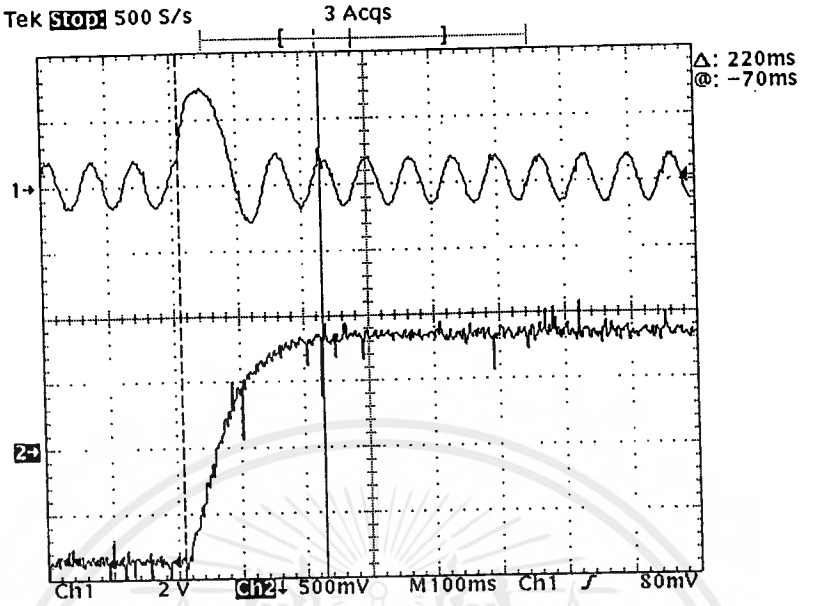
บทที่ 6

ผลการทดลอง

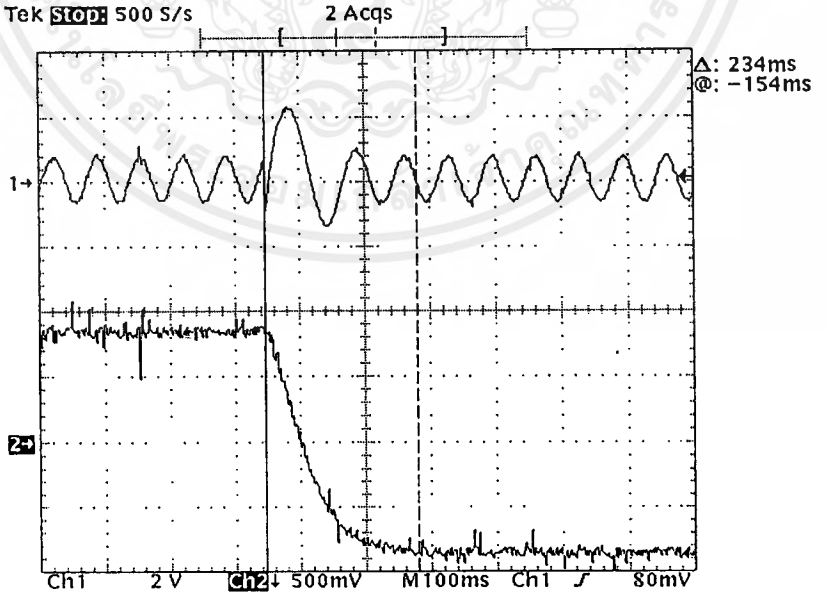
6.1 แสดงสัญญาณเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ กับกระแสจริง การหมุนกลับทิศ เมื่อมีภาระและไม่มีภาระ และการเปลี่ยนแปลงภาระดังรูป



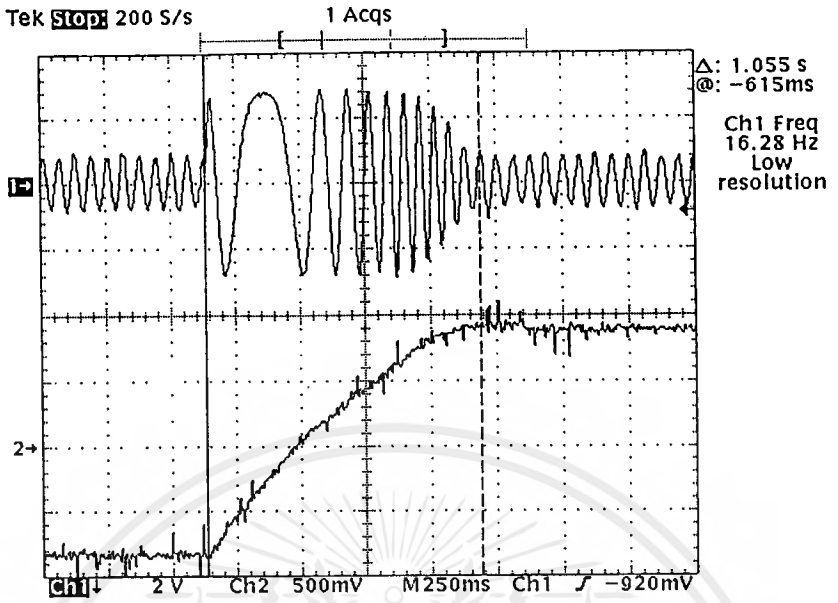
รูปที่ 6.1 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 15 Hz



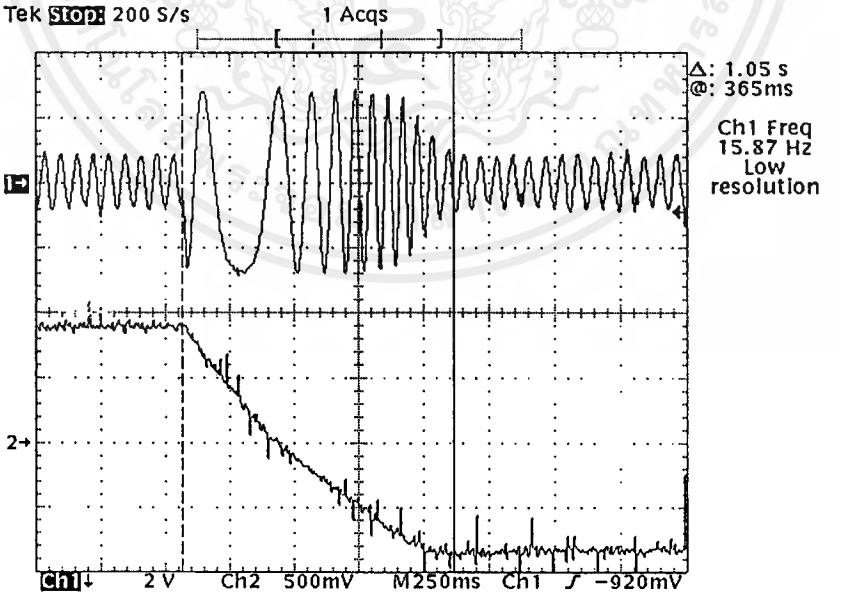
รูปที่ 6.2 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก -15 Hz จนเป็น 15 Hz ขณะไม่มีภาระ



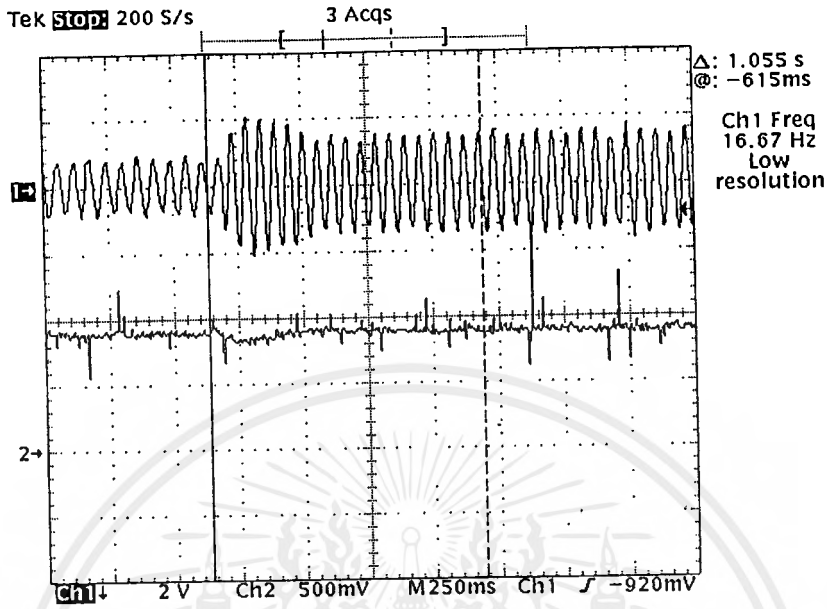
รูปที่ 6.3 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก 15 Hz จนเป็น -15 Hz ขณะไม่มีภาระ



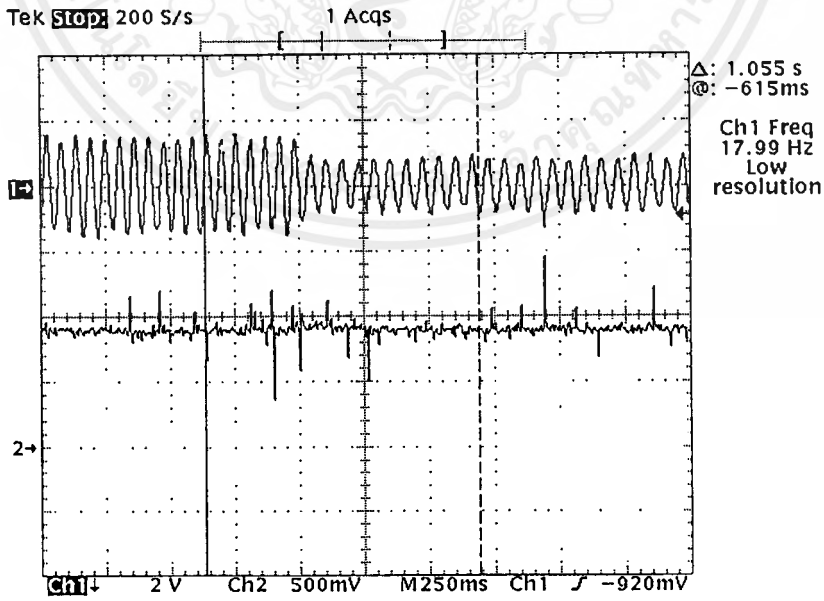
รูปที่ 6.4 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก -15 Hz จนเป็น 15Hz ขณะมีภาระ



รูปที่ 6.5 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก 15 Hz จนเป็น -15 Hz ขณะมีภาระ

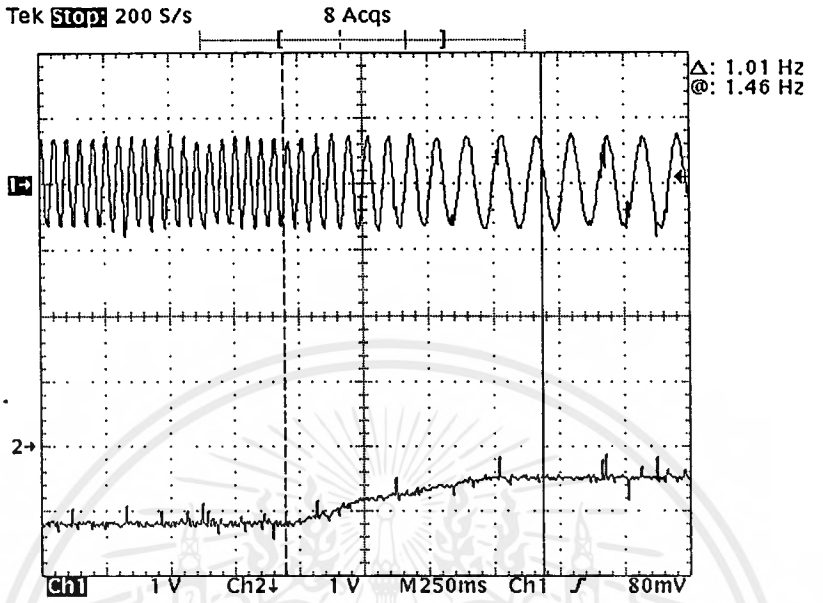


รูปที่ 6.6 แสดงการเปลี่ยนแปลงของสัญญาณกระแสและความเร็วเมื่อมีการเพิ่มภาระเป็นขั้นที่ความถี่ 25 Hz

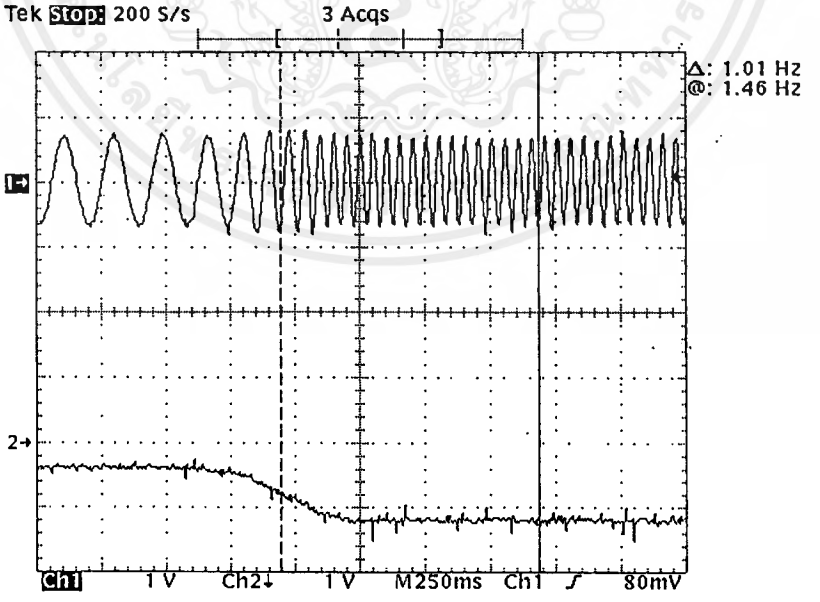


รูปที่ 6.7 แสดงการเปลี่ยนแปลงของสัญญาณกระแสและความเร็วเมื่อมีการลดภาระเป็นขั้นที่ความถี่ 25 Hz

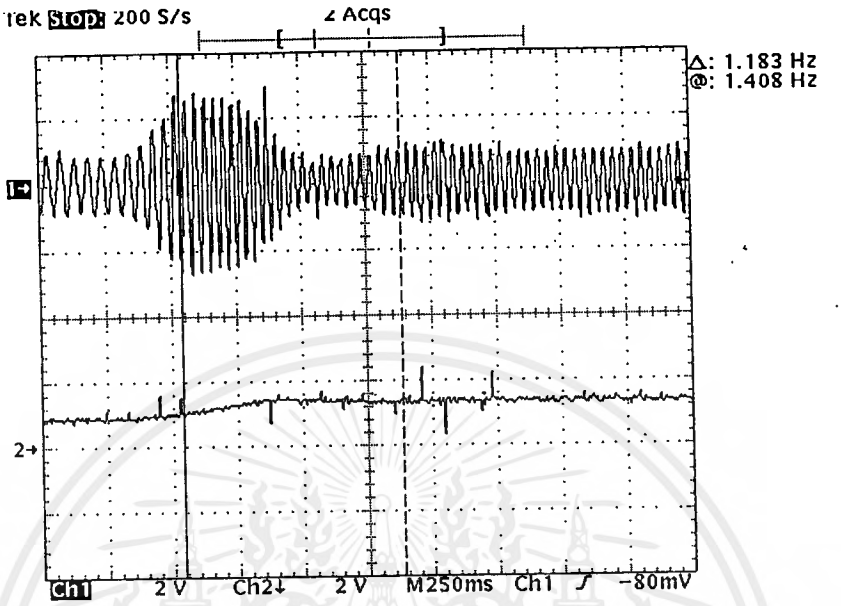
6.2 แสดงการเพิ่ม-ลดความเร็วขณะไม่มีภาระและมีภาระตามลำดับ



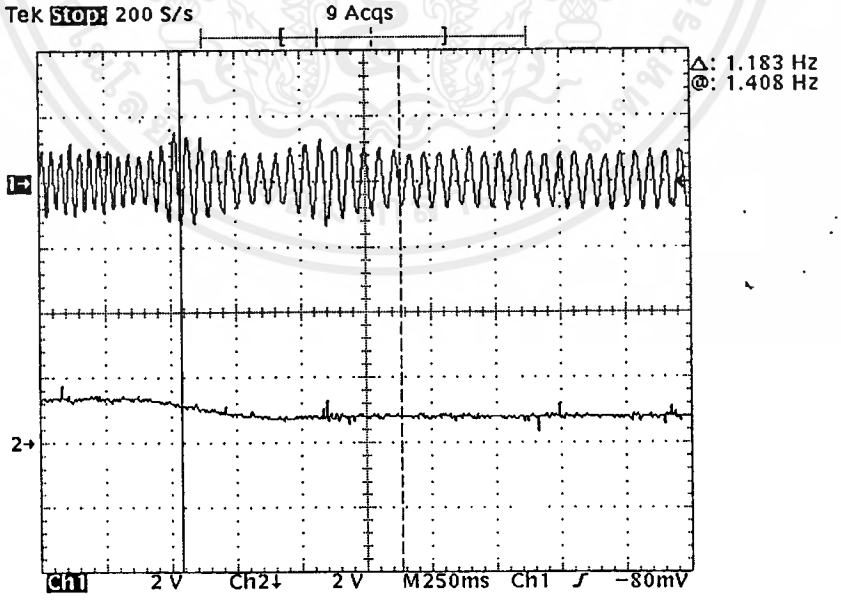
รูปที่ 6.8 แสดงสัญญาณกระแสเมื่อมีการเปลี่ยนความเร็วเพิ่มเป็นขั้นขณะไม่มีภาระ



รูปที่ 6.9 แสดงสัญญาณกระแสเมื่อมีการเปลี่ยนความเร็วลดเป็นขั้นขณะไม่มีภาระ

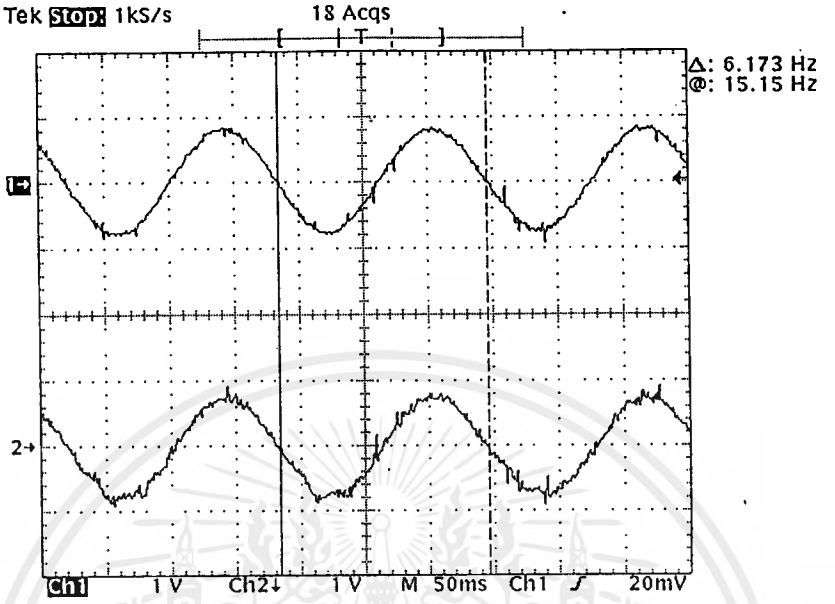


รูปที่ 6.10 แสดงสัญญาณกระแสเมื่อมีการเปลี่ยนความเร็วเพิ่มเป็นขั้นขณะมีภาระ

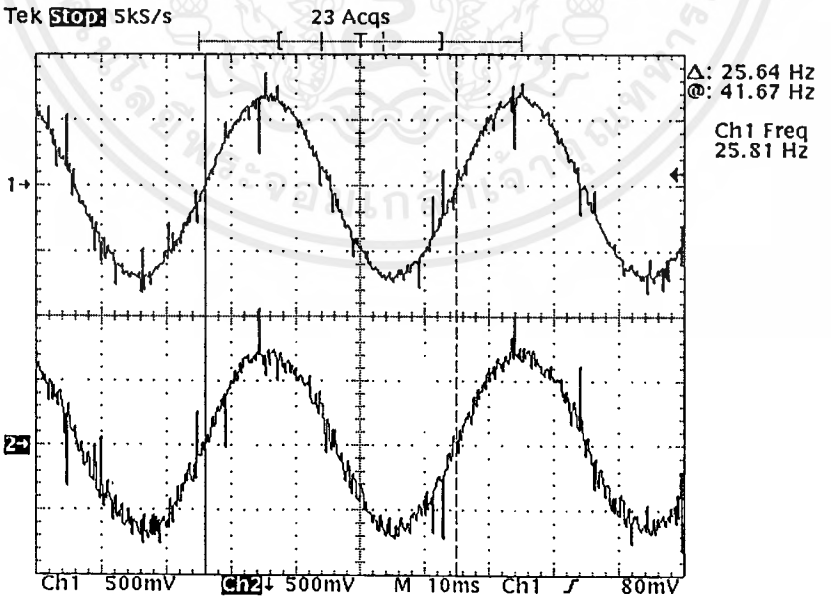


รูปที่ 6.11 แสดงสัญญาณกระแสเมื่อมีการเปลี่ยนความเร็วลดเป็นขั้นขณะมีภาระ

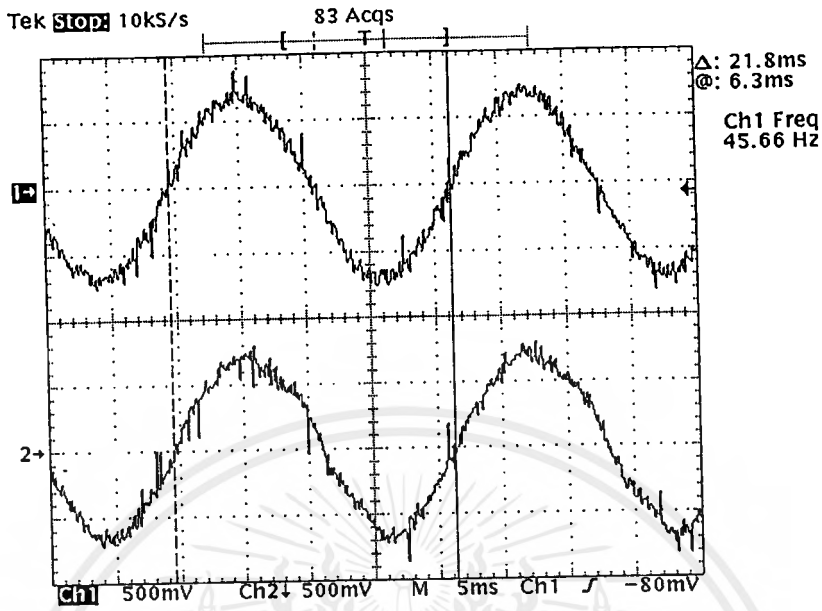
6.3 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์กับกระแสจริงที่ความถี่ 5 , 25 , 45 , 50 Hz ตามลำดับ



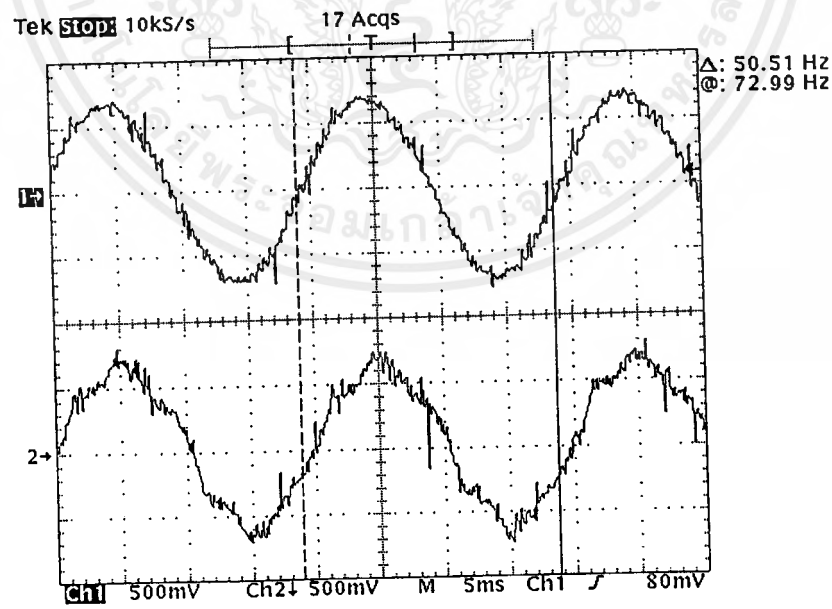
รูปที่ 6.12 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 5 Hz



รูปที่ 6.13 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 25 Hz

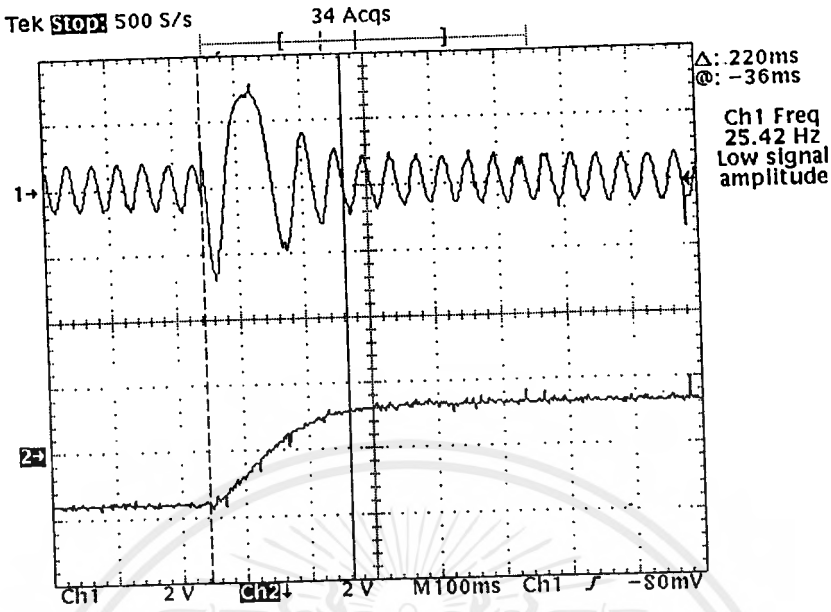


รูปที่ 6.14 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 45 Hz

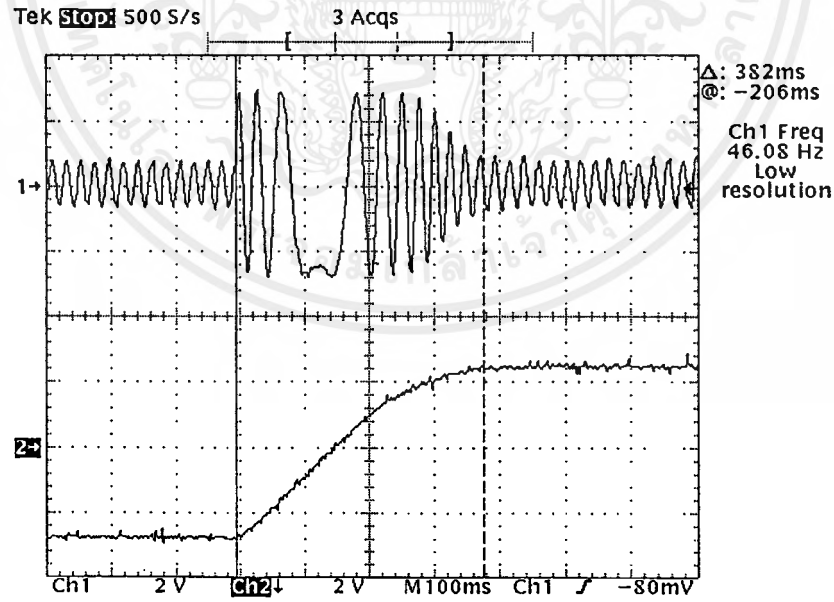


รูปที่ 6.15 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 50 Hz

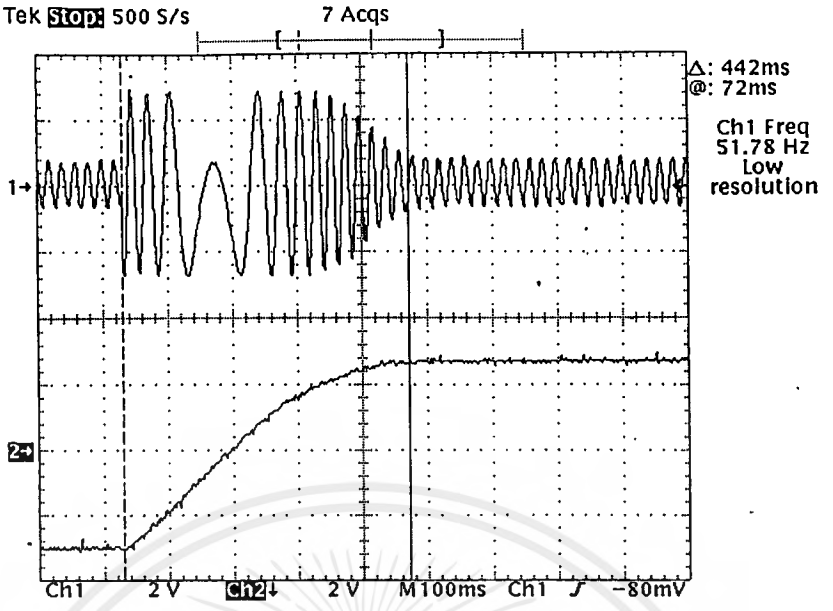
6.4 แสดงความเร็วในการกลับทิศที่ความถี่ 5 , 15 , 45 , 50 Hz ตามลำดับ



<u>รูปที่ 6.16</u> แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก -15 Hz จนเป็น 15 Hz ขณะไม่มีภาระ

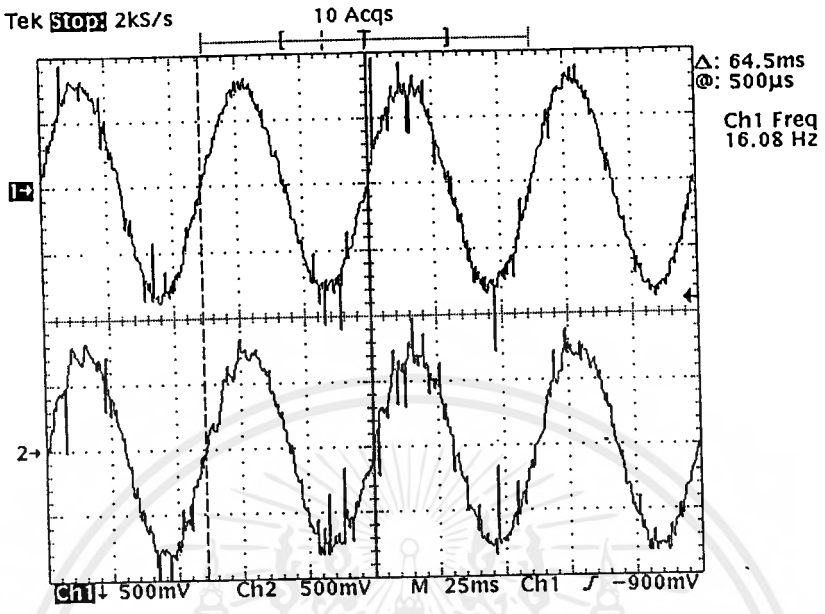


<u>รูปที่ 6.17</u> แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก - 45 Hz จนเป็น 45 Hz ขณะไม่มีภาระ

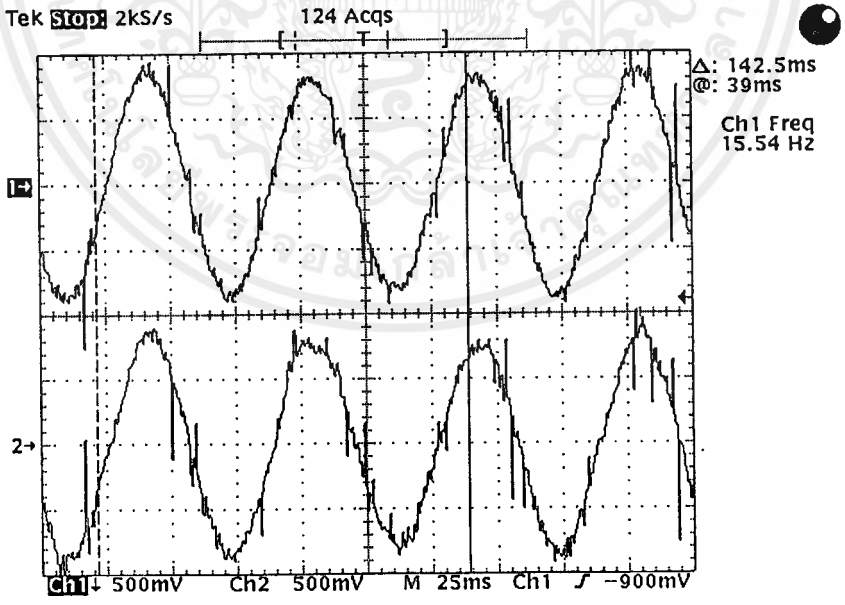


รูปที่ 6.18 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก - 50 Hz จนเป็น 50 Hz ขณะไม่มีภาระ

6.5 แสดงสัญญาณคำสั่งกระแสและกระแสจริงเมื่อเปลี่ยนค่า Tr ที่ความถี่ 15 Hz เป็น $0.8T_r$ และ $1.2T_r$ ตามลำดับ

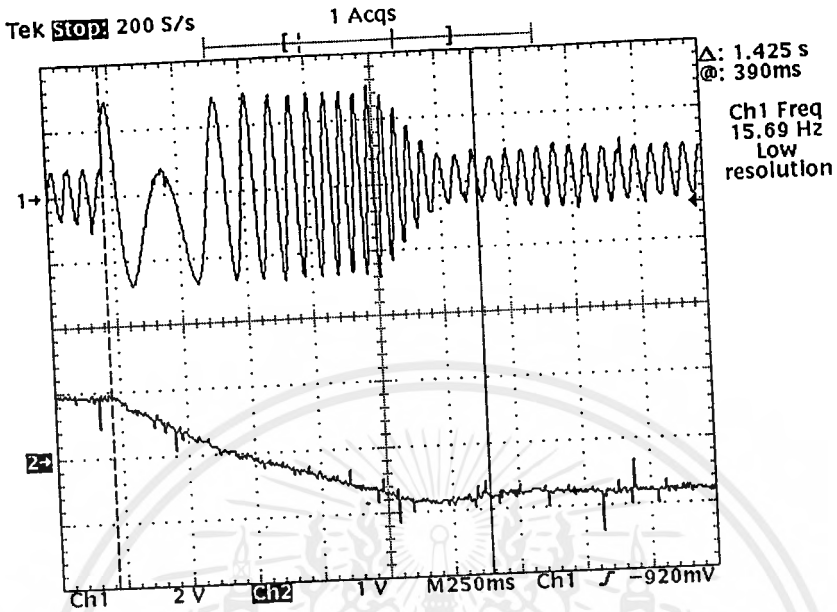


รูปที่ 6.19 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 15 Hz ($0.8T_r$)

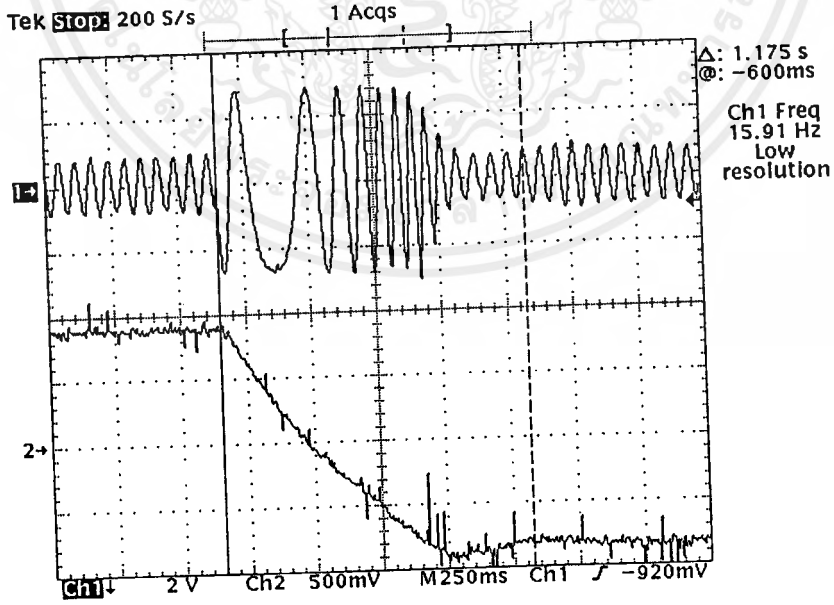


รูปที่ 6.20 แสดงการเปรียบเทียบสัญญาณคำสั่งกระแสที่ควบคุมโดยไมโครคอนโทรลเลอร์ (บน) และกระแสจริง (ล่าง) ที่ความถี่ 15 Hz ($1.2T_r$)

6. แสดงการหมุนกลับทิศเมื่อเปลี่ยนค่า Tr ที่ความถี่ 15 Hz ขณะมีภาระเป็น $0.8T_r$ และ $1.2T_r$ ตามลำดับ

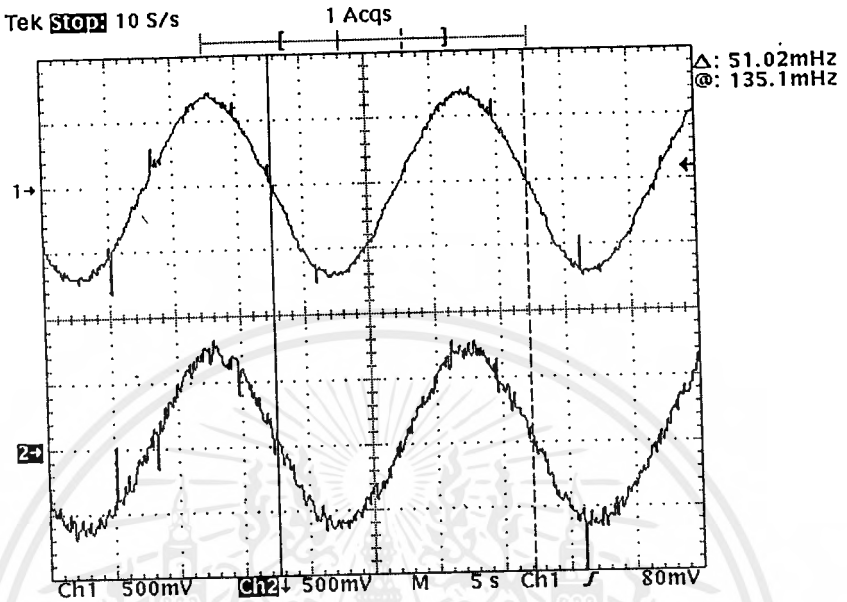


รูปที่ 6.21 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก 15 Hz จนเป็น -15 Hz ขณะมีภาระ($0.8T_r$)

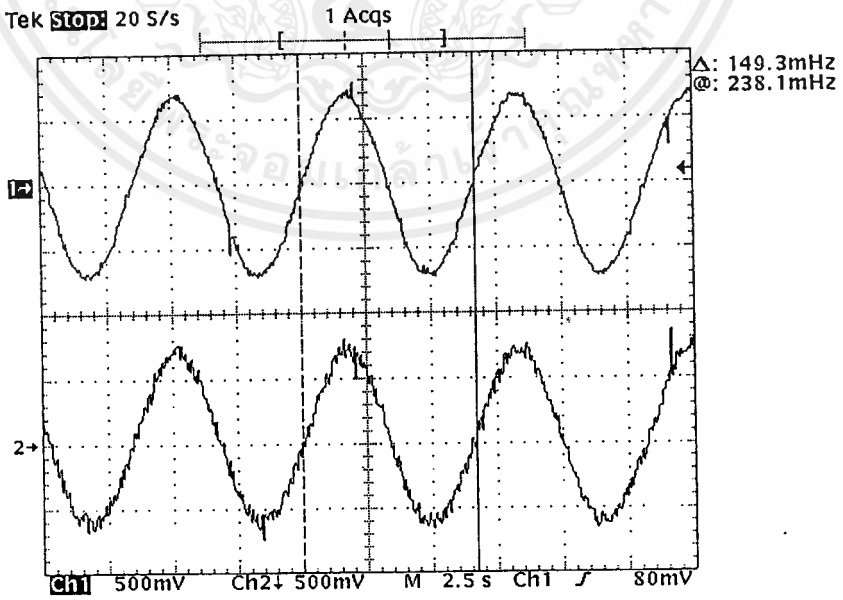


รูปที่ 6.22 แสดงสัญญาณกระแสและความเร็วเมื่อกลับทิศการหมุนจาก 15 Hz จนเป็น -15 Hz ขณะมีภาระ($1.2T_r$)

6.7 ความถี่ต่ำสุดในการเริ่มหมุนของมอเตอร์ขณะไม่มีภาระและมีภาระตามลำดับ



รูปที่ 6.23 แสดงความถี่ต่ำสุดเมื่อเริ่มหมุนมอเตอร์ขณะไม่มีภาระ



รูปที่ 6.24 แสดงความถี่ต่ำสุดเมื่อเริ่มหมุนมอเตอร์ขณะมีภาระ

สรุปผลและวิเคราะห์ผลการทดลอง

1.) จากรูปที่ 6.1 จะเห็นได้ว่าสามารถที่จะควบคุมกระแสที่สเตเตอร์ได้ตามที่เราต้องการและรูปคลื่นกระแสที่ได้ก็มีความผิดเพี้ยนน้อยมากเมื่อเทียบกับคำสั่งกระแสและรูปคลื่นที่ได้ก็เป็นคลื่นไซน์

2.) รูปที่ 6.2 และ 6.3 แสดงการกลับทิศทางการหมุนของมอเตอร์ที่ความถี่ 15 Hz ในสภาวะที่ไม่มีภาระซึ่งจะใช้เวลาประมาณ 220 ms สังเกตกระแสที่สเตเตอร์ที่ ความเร็วผ่านการควบคุมแบบ PI กระแสที่สเตเตอร์จะมีขนาดเพิ่มขึ้นและมีความถี่ต่ำลงจนกระทั่งที่ความเร็วเป็นศูนย์ก็จะมีความถี่สูงขึ้นและเฟสของกระแสจะกลับเฟสเพื่อหมุนในทิศทางใหม่

3.) รูปที่ 6.4 และ 6.5 แสดงการกลับทิศขณะมีภาระ จะเห็นได้ว่าจะช้ากว่าเมื่อไม่มีภาระมาก

4.) รูปที่ 6.6 และ 6.7 แสดงการเพิ่มและลดภาระเป็นขั้นกับมอเตอร์ความเร็วจะตกลงและก็สามารถกลับเข้าสู่ความเร็วเดิมได้ กระแสที่ สเตเตอร์จะมีขนาดเพิ่มขึ้นเนื่องจากต้องการแรงบิดมาก

5.) จากรูปที่ 6.8 ถึงรูปที่ 6.11 เป็นการแสดงการเปลี่ยนแปลงความเร็วเป็นขั้นทั้งการเพิ่มและลดความเร็วในขณะที่มีภาระและไม่มีภาระ ขณะไม่มีภาระขนาดของกระแสสเตเตอร์จะค่อนข้างคงที่แต่จะลดหรือเพิ่มความถี่เท่านั้นเนื่องจากใช้แรงบิดน้อยมาก แต่เมื่อเทียบกับที่สภาวะมีภาระกระแสที่สเตเตอร์จะมีขนาดสูงขึ้นเมื่อเกิดการเปลี่ยนแปลงความเร็วเนื่องจากต้องการแรงบิดมากในการเพิ่มความเร็วแต่ในขณะลดความเร็วจะใช้แรงบิดต้านต่ำกว่า

6.) รูปที่ 6.12 ถึงรูปที่ 6.15 แสดงสัญญาณคำสั่งกระแสเปรียบเทียบกับกระแสจริงที่สเตเตอร์ที่ความถี่ต่าง ๆ จะเห็นได้ว่ากระแสจริงมีความคล้ายกับกระแสคำสังมากและรูปร่างจะมีความผิดเพี้ยนเมื่อความถี่สูงขึ้นเช่นที่ความถี่ 45 Hz กระแสจริงจะเริ่มมีความผิดเพี้ยนและที่ความถี่ 50 Hz รูปคลื่นกระแสจริงจะมีความผิดเพี้ยนมากและที่สำคัญจะเกิดการเลื่อนเฟสขึ้นซึ่งเกิดจากแกนเหล็กอิ่มตัวทำให้ไม่สามารถควบคุมแรงบิดได้ เนื่องมาจากความเร็วของโรเตอร์สูงสุดมีค่า 46.67 Hz

ถ้าต้องการให้มอเตอร์หมุนได้ที่ความเร็วสูงกว่า 46.67 Hz จะต้องลดค่ากระแสสนาม (Field weakening) และให้มอเตอร์ทำงานในโหมดกำลังคงที่แทน

1.) รูปที่ 6.16 ถึงรูปที่ 6.18 แสดงการกลับทิศความเร็วที่ความถี่ต่าง ๆ ที่สภาวะไม่มีภาระซึ่งใช้เวลาดังนี้

ความถี่	เวลา (ms)
25	220
45	380
50	442

8.) รูปที่ 6.19 ถึงรูปที่ 6.20 แสดงสัญญาณกระแสคำสั่งเทียบกับกระแสจริงเมื่อมีการเปลี่ยนแปลงค่า Tr โดยเปลี่ยนแปลงจากค่าเดิม 20 % รูปของกระแสมีความเพี้ยนเล็กน้อยแต่เมื่อทำการกลับทิศทางการหมุนขณะมีภาระ ดังรูปที่ 6.21 และ 6.22 ที่ Tr ลดลงจะทำให้การกลับทิศช้าลง และที่ Tr เพิ่มขึ้นความเร็วจะเกิด Over damp ขึ้น

9.) รูปที่ 6.23 และรูปที่ 6.24 แสดงกระแสที่สเตเตอร์ ที่หมุนที่ความเร็วต่ำสุด ในสภาวะมีภาระและไม่มีภาระตามลำดับ

บทที่ 7

สรุปผล ปัญหาที่พบ และแนวทางในการพัฒนา

7.1 สรุปปริญญานิพนธ์

ปริญญานิพนธ์นี้เป็นการประยุกต์นำไมโครคอนโทรเลอร์เข้ามาใช้ในการควบคุมมอเตอร์เหนี่ยวนำ 3 เฟส มีพิกัดกำลัง 2 แรงม้า โดยควบคุมการปรับสนามแม่เหล็กทางอ้อม ซึ่งไมโครคอนโทรลเลอร์ที่ใช้เป็นไมโครคอนโทรลเลอร์ขนาด 16 บิท ความเร็วของสัญญาณนาฬิกาเท่ากับ 12 MHz เบอร์ 80C196KB ในการสร้างสัญญาณกระแสโดยรับคำสั่งความเร็ว และความเร็วจริงจาก Encoder ที่มีความละเอียด 2000 ลูกคลื่นต่อหนึ่งรอบ สัญญาณคำสั่งกระแสที่ได้จะถูกนำมาควบคุมรวมกับกระแสที่ได้จากตัวตรวจจับสัญญาณกระแสที่สเตเตอร์ที่มีอัตราส่วน 1 V ต่อกระแส 4 A โดยการควบคุมแบบ PI จะได้สัญญาณกระแสคำสั่งแล้วนำมาเปรียบเทียบกับคลื่นสัญญาณสามเหลี่ยม ซึ่งจะได้สัญญาณ PWM แล้วนำสัญญาณ PWM ไปผ่านวงจรขับขาเกต EXB841 สัญญาณที่ได้จากวงจรขับเกตจะไปขับ IGBT ที่มีพิกัดแรงดัน 600 V และสามารถทนกระแสได้ถึง 20 A โดยมีความถี่สวิทช์ 5 kHz ซึ่งในหนึ่งอันจะมี IGBT 6 ตัว สามารถดูลายละเอียดเพิ่มเติมได้จากภาคผนวก วิธีการออกแบบวงจรควบคุมนี้ใช้การออกแบบในเชิงเวลา (Time Domain) โดยมองคำสั่งกระแสเป็นคลื่นไซน์ลักษณะเป็นขั้นบันไดดดยที่เวลาในการเข้าสู่คำสั่งจะมีค่าน้อยกว่าเวลาในการเปลี่ยนแปลงรูปคลื่นของกระแสซึ่งมีเวลาประมาณ 500 ไมโครวินาที และจะคำนวณความเร็วทุก ๆ 5 มิลิวินาที นอกจากนี้ยังมีสวิทช์ควบคุมการเปิด – ปิด และ กลับทิศทางหมุนและจอ LCD แสดงสถานะการทำงาน ส่วนของมอเตอร์ที่ใช้มีพิกัดกำลังเป็น 2 แรงม้า ความเร็วที่พิกัดมีค่าเท่ากับ 2800 รอบต่อนาที

7.2 ปัญหาที่พบ

จากวิธีการที่ใช้ในการควบคุมจำเป็นที่จะต้องทราบคุณสมบัติทางไฟฟ้าและทางกลของมอเตอร์ เช่นค่าคงที่ทางเวลาที่โรเตอร์ ค่ากระแสสนามในโรเตอร์ และค่าโมเมนต์ความเฉื่อยของมอเตอร์ เป็นต้นซึ่งค่าดังกล่าวจะเป็นค่าที่ทำการหาในสภาวะคงตัวของมอเตอร์ (Steady State) แต่ในการใช้งานจริง ค่าดังกล่าวมีกสรเปลี่ยนแปลงอยู่ตลอดเวลาเช่น ค่าคงที่ทางเวลาของโรเตอร์ ซึ่งจะเปลี่ยนแปลงตามความถี่และอุณหภูมิ ส่วนค่ากระแสสนามถ้ากำหนดค่ามากไปก็จะทำให้เกิดการอิ่มตัวของแกนเหล็กที่โรเตอร์ได้ง่าย และถ้าต้องการความเร็วมากกว่าความเร็วที่พิกัดก็จะต้องทำการลดค่ากระแสสนาม (Field Weakening) ตัวนี้ลงมิฉะนั้นจะทำให้ไม่สามารถควบคุมกระแสได้ และการลดค่าก็ไม่เป็นเชิงเส้น เนื่องจากต้องการให้กำลังงานคงที่ และสุดท้ายคือค่าของโมเมนต์ความเฉื่อยซึ่งเป็นตัวกำหนดแรงบิดเริ่มต้นในการหมุนและความเร็วในการหมุนกลับทิศ ซึ่งค่าที่หาได้ก็เป็นค่าประมาณไม่สามารถที่จะหาค่าแท้จริงได้

7.3 แนวทางในการพัฒนา

การควบคุมที่ใช้เป็นการควบคุมสนามที่สเตเตอร์ ทำให้ต้องใช้วงจรอนาลอกช่วยในการคำนวณคำสั่งกระแสแต่สามารถแก้ไขได้โดยการควบคุมสนามที่โรเตอร์แทนซึ่งค่าที่ควบคุมจะเป็นกระแสตรงง่ายต่อการควบคุมง่ายต่อการควบคุมแต่ต้องเสียเวลาในการเปลี่ยนกรอบอ้างอิงมาก

นอกจากนี้การควบคุมที่ใช้จำเป็นต้องใช้ Encoder ซึ่งทำให้ไม่มีความสะดวกในการติดตั้งกับมอเตอร์แต่ควรจะคำนวณจากกระแสและแรงดันของมอเตอร์แทน หรือใช้การควบคุมที่ไม่มีตัวตรวจจับ (Sensorless Control) และตัวขับมอเตอร์ต้องสามารถตรวจสอบคุณสมบัติของมอเตอร์ได้ด้วยตัวเองและสามารถหาค่าที่ถูกต้องในสภาวะต่าง ๆ โดยการสอนให้ตัวขับมอเตอร์มีการเรียนรู้และพัฒนาในการขับมอเตอร์ในสภาวะต่าง ๆ

ภาคผนวก I

Application Note Using the 80c196KB

1.0 INTRODUCTION

The MCS®-96 family members are all high performance microcontrollers with a 16-bit CPU and at least 230 bytes of on chip RAM. The Intel MCS-96 family of 16-bit embedded controllers easily handles high speed calculations and fast input/output (I/O) operations. Typical applications using the MCS-96 products include closed-loop control and mid-range digital signal processing. Modems, motor control system, printers, engine control system, photocopiers, anti-lock brakes, air conditioner control systems, disk drives and medical instrumentation all use MCS-96 products.

The 80C196KB is a CHMOS member of the MCS-96 family. All of the MCS-96 components share a common instruction set and architecture. However, the CHMOS components have enhancements to provide higher performance with lower power consumption. To further decrease power usage, idle and power-down modes are available on these devices. The 80C196KB contains a dedicated I/O subsystem and can perform 16-bit arithmetic instructions including multiply and divide operations.

This application note briefly describes the 80C196KB, and provides software examples using its key features. For further information on the 80C196KB and its use consult the sources listed in the bibliography. Figure 1-1 shows a block diagram of the 80C196KB. Included in this application note are descriptions of the CPU and architecture, the interrupt structure and the peripherals. These peripherals include a Pulse Width Modulation output, an A/D Converter, a Serial Port and High Speed I/O Unit with two 16-bit timer/counters.

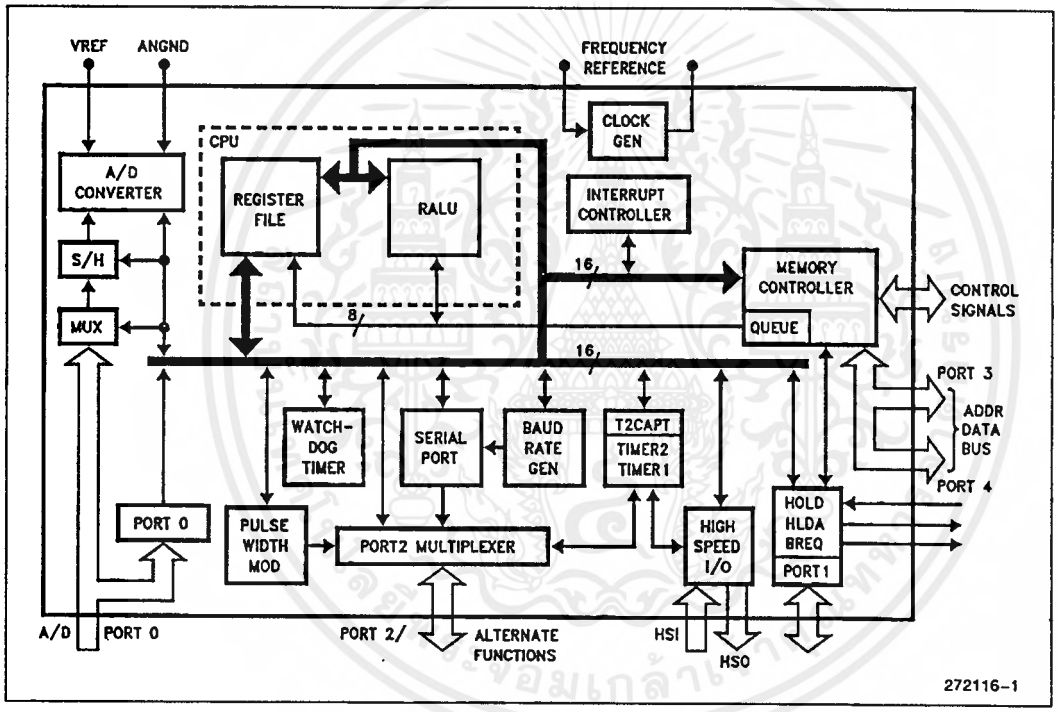


Figure 1-1. 80C196KB Block Diagram

2.0 THE CPU

The major components of the 80C196KB CPU are the Register File and the Register/Arithmetic Logic Unit (RALU). The Register File contains 256 internal register locations (00H through 0FFH), all of which remain alive during power-down mode. Locations 00H through 17H are the I/O control registers or Special Function Registers (SFRs). Locations 18H and 19H contain the stack pointer, which can serve as general purpose RAM when not performing stack operations. The remaining 230 bytes serve as general purpose RAM, accessible as bytes, words or double-words.

Calculations performed by the 80C196KB take place in the RALU. The RALU shown in Figure 2-1 contains a 17-bit ALU, the Program Status Word (PSW), the Program Counter (PC), a loop counter, and three temporary registers. The RALU operates directly on the Register File, thus eliminating accumulator bottleneck and providing for direct control of I/O operations through the SFRs.

The SFRs control all the 80C196KB peripheral devices except Ports 3 and 4. Figure 2-2 shows the layout of these registers. Three SFR windows exist on the 80C196KB. The value in the Window Select Register (WSR) determines the SFR window; WSR = 0 selects Window 0 and WSR = 15 selects Window 15. Window 0 consists of 24 SFRs. Some of these registers serve one function when read and another function when written. The read-only registers in Window 0 become write-only registers in Window 15; and the write-only registers in Window 0 become read-only registers in Window 15. Figure 2-3 contains descriptions of the SFRs.

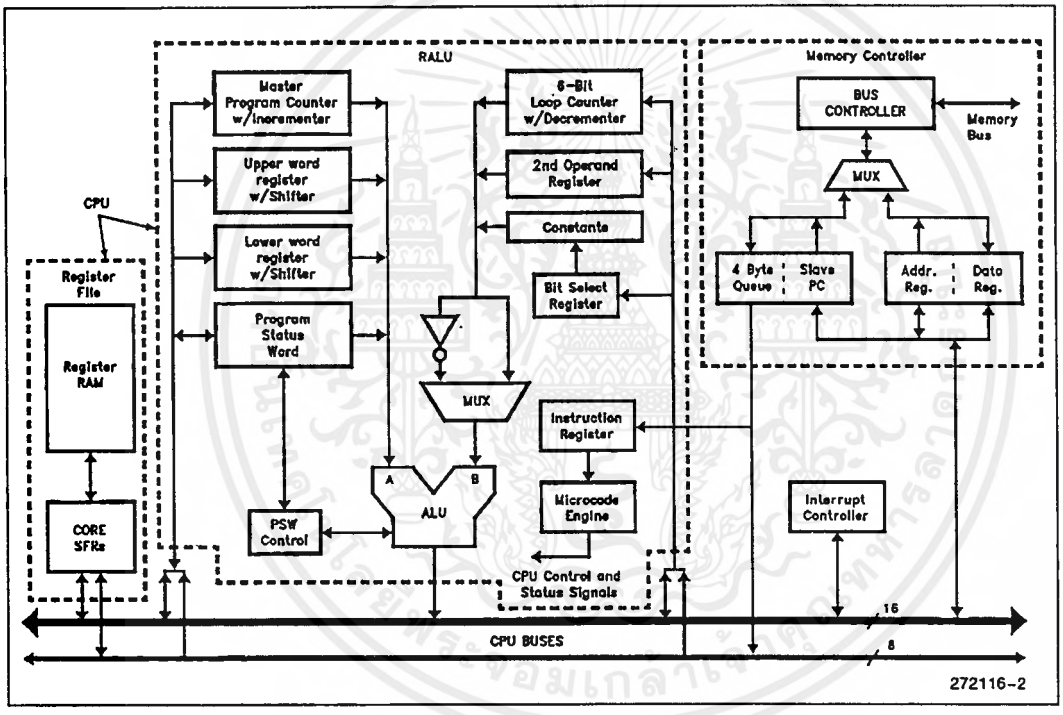


Figure 2-1. Block Diagram of the Register File, RALU, Interrupt Controller and Memory Controller

	WHEN READ WSR = 0	WHEN WRITTEN WSR = 0	WHEN READ WSR = 15	WHEN WRITTEN WSR = 15
19H	STACK POINTER	STACK POINTER	STACK POINTER	STACK POINTER
18H				
17H	*IOS2	PWM_CONTROL	PWM_CONTROL	*IOS2
16H	IOS1	IOC1	IOC1	IOS1
15H	IOS0	IOC0	IOC0	IOS0
14H	*WSR	*WSR	*WSR	*WSR
13H	*INT_MASK1	*INT_MASK1	*INT_MASK1	*INT_MASK1
12H	*INT_PEND1	*INT_PEND1	*INT_PEND1	*INT_PEND1
11H	*SP_STAT	*SP_CON	*SP_CON	*SP_STAT
10H	PORT2	PORT2	RESERVED**	RESERVED**
0FH	PORT1	PORT1	RESERVED**	RESERVED**
0EH	PORT0	BAUD RATE	RESERVED**	RESERVED**
0DH	TIMER2(HI)	TIMER2(HI)	T2CAPTURE(HI)	T2CAPTURE(HI)
0CH	TIMER2(LO)	TIMER2(LO)	T2CAPTURE(LO)	T2CAPTURE(LO)
0BH	TIMER1(HI)	*IOC2	*IOC2	TIMER1(HI)
0AH	TIMER1(LO)	WATCHDOG	WATCHDOG	TIMER1(LO)
09H	INT_PEND	INT_PEND	INT_PEND	INT_PEND
08H	INT_MASK	INT_MASK	INT_MASK	INT_MASK
07H	SBUF(RX)	SBUF(TX)	SBUF(TX)	SBUF(RX)
06H	HSI_STATUS	HSO_COMMAND	HSO_COMMAND	HSI_STATUS
05H	HSI_TIME(HI)	HSO_TIME(HI)	HSO_TIME(HI)	HSI_TIME(HI)
04H	HSI_TIME(LO)	HSO_TIME(LO)	HSO_TIME(LO)	HSI_TIME(LO)
03H	AD_RESULT(HI)	HSI_MODE	HSI_MODE	AD_RESULT(HI)
02H	AD_RESULT(LO)	AD_COMMAND	AD_COMMAND	AD_RESULT(LO)
01H	ZERO_REG(HI)	ZERO_REG(HI)	ZERO_REG(HI)	ZERO_REG(HI)
00H	ZERO_REG(LO)	ZERO_REG(LO)	ZERO_REG(LO)	ZERO_REG(LO)

NOTES:
*New or changed register function from 8096BH
**Reserved registers should not be written or read

Figure 2-2. Special Function Registers

Register	Description
ZERO_REG	Zero Register - Always reads as a zero, useful for a base when indexing and as a constant for calculations and compares.
AD_RESULT	A/D Result Hi/Low - Low and high order results of the A/D converter
AD_COMMAND	A/D Command Register - Controls the A/D
HSI_MODE	HSI Mode Register - Sets the mode of the High Speed Input unit.
HSI_TIME	HSI Time Hi/Lo - Contains the time at which the High Speed Input unit was triggered.
HSO_TIME	HSO Time Hi/Lo - Sets the time or count for the High Speed Output to execute the command in the Command Register.
HSO_COMMAND	HSO Command Register - Determines what will happen at the time loaded into the HSO Time registers.
HSI_STATUS	HSI Status Registers - Indicates which HSI pins were detected at the time in the HSI Time registers and the current state of the pins. In Window 15 - Writes to pin detected bits, but not current state bits.
SBUF(TX)	Transmit buffer for the serial port, holds contents to be outputted. Last written value is readable in Window 15.
SBUF(RX)	Receive buffer for the serial port, holds the byte just received by the serial port. Writable in Window 15.
INT_MASK	Interrupt Mask Register - Enables or disables the individual interrupts.
INT_PEND	Interrupt Pending Register - Indicates that an interrupt signal has occurred on one of the sources and has not been serviced. (also INT_PENDING)
WATCHDOG	Watchdog Timer Register - Written periodically to hold off automatic reset every 64K state times. Returns upper byte of WDT counter in Window 15.
TIMER1	Timer 1 Hi/Lo - Timer1 high and low bytes.
TIMER2	Timer 2 Hi/Lo - Timer2 high and low bytes.
IOPORT0	Port 0 Register - Levels on pins of Port 0. Reserved in Window 15.
BAUD_RATE	Register which determines the baud rate, this register is loaded sequentially. Reserved in Window 15.
IOPORT1	Port 1 Register - Used to read or write to Port 1. Reserved in Window 15
IOPORT2	Port 2 Register - Used to read or write to Port 2. Reserved in Window 15
SP_STAT	Serial Port Status - Indicates the status of the serial port.
SP_CON	Serial Port Control - Used to set the mode of the serial port.
IOS0	I/O Status Register 0 - Contains information on the HSO status. Writes to HSO pins in Window 15.
IOS1	I/O Status Register 1 - Contains information on the status of the timers and of the HSI.
IOC0	I/O Control Register 0 - Controls alternate functions of HSI pins, Timer 2 reset sources and Timer 2 clock sources.
IOC1	I/O Control Register 1 - Controls alternate functions of Port 2 pins, timer interrupts and HSI interrupts.
PWM_CONTROL	Pulse Width Modulation Control Register - Sets the duration of the PWM pulse.
INT_PEND1	Interrupt Pending register for the 8 new interrupt vectors (also INT_PENDING1)
INT_MASK1	Interrupt Mask register for the 8 new interrupt vectors
IOC2	I/O Control Register 2 - Controls new 80C196KB features
IOS2	I/O Status Register 2 - Contains information on HSO events
WSR	Window Select Register - Selects register window

Figure 2-3. Special Function Register Descriptions

3.0 THE ARCHITECTURE

The 80C196KB supports 106 instructions. This instruction set includes bit operations, byte operations, word operations, double-word operations (unsigned 32-bit) long operations (signed 32-bit), flag manipulations as well as jump and call instructions. All the standard logical and arithmetic instructions function as both byte and word operations. The Jump Bit Set and Jump Bit Clear instructions can operate on any of the SFRs or bytes in the register file. These fast bit manipulations allow for rapid I/O functions.

Byte and word operations make-up most of the 80C196KB instruction set. The assembly language for the 80C196KB (ASM-96) uses a "B" suffix on a mnemonic for a byte operation, otherwise the mnemonic refers to a word operation. One, two or three operand forms exist for many of the instructions.

A one operand instruction has the form:

```
NOT Value1          ;Value1 = 1's complement (Value1)
```

A two operand instruction has the form:

```
ADD Value2, Value1 ;Value2 = Value2 + Value1
```

A three operand instruction has the form:

```
MUL Value3, Value2, Value1 ;Value3 = Value2 * Value1.
```

Long and double-word operations include shifts, normalize, multiply and divide. The divide instruction functions as a 32-bit by 16-bit divide that generates a 16-bit quotient and 16-bit remainder. The word multiply operates as a 16-bit by 16-bit multiply with a 32-bit result. Both operations can function in either the signed or unsigned mode. The direct unsigned modes of these instructions take only 3.0 μs (at 16 MHz) for divide and 1.75 μs (at 16 MHz) for multiply. The normalize instruction and sticky bit flag provide hardware support for the software floating point package (FPAL-96).

3.1 Addressing Modes

The 80C196KB instruction set supports the following addressing modes: register-direct, indirect, indirect with auto-increment, immediate, short-indexed and long-indexed. These modes increase the flexibility and overall execution speed of the 80C196KB. Each instruction uses at least one of the addressing modes. These modes and formats are shown in Figure 3-1.

```
Mnem        Dest or Src1                ;One Operand Direct
Mnem        Dest, Src1                  ;Two Operand Direct
Mnem        Dest, Src1, Src2            ;Three Operand Direct

Mnem        #Src1                       ;One Operand Immediate
Mnem        Dest, #Src1                 ;Two Operand Immediate
Mnem        Dest, Src1, #Src2           ;Three Operand Immediate

Mnem        [addr]                      ;One Operand Indirect
Mnem        [addr]+                     ;One Operand Indirect Auto-Increment
Mnem        Dest, [addr]                ;Two Operand Indirect
Mnem        Dest, [addr]+               ;Two Operand Indirect Auto-Increment
Mnem        Dest, Src1, [addr]          ;Three Operand Indirect
Mnem        Dest, Src1, [addr]+         ;Three Operand Indirect Auto-Increment

Mnem        Dest, offs[addr]            ;Two Operand Indexed (Short or Long)
Mnem        Dest, Src1, offs[addr]      ;Three Operand Indexed (Short or Long)
```

Where:
Mnem = Instruction Mnemonic
Dest = Destination Register
Src1, Src2 = Source Registers
addr = Word register used in computing the address of an operand
offs = Offset used in computing the address of an operand

Figure 3-1. Instruction Format

The register-direct and immediate addressing modes execute faster than the other addressing modes. The register-direct addressing mode provides access to the addresses in the register file and the SFRs. The indexed modes provide for direct access to the remainder of the 64K address space. Immediate addressing uses the data following the opcode as the operand.

Both of the indirect addressing modes use the value in a word register as the address of the operand. The indirect auto-increment mode increments a word address by one after a byte operation and two after a word operation. This addressing mode provides easy access into look-up tables.

The long-indexed addressing mode provides direct access to any of the locations in the 64K address space. This mode forms the address of the operand by adding a 16-bit 2's complement value to the contents of a word register. Indexing with the zero register allows "direct" addressing to any location. The short-indexed addressing mode forms the address of the operand by adding an 8-bit 2's complement value to the contents of a word register.

The multiple addressing modes of the 80C196KB make it easy to program in assembly language and provide an excellent interface to high level languages. The instructions accepted by the assembler consist of mnemonics followed by either addresses or data. Table 3-1 lists the mnemonics and their functions. The MCS-96 Macro Assembler Users Guide, listed in the bibliography, contains additional information on 80C196KB assembly language.

Table 3-1. Instruction Summary

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	D ← D + A	✓	✓	✓	✓	↑	–	
ADD/ADDB	3	D ← B + A	✓	✓	✓	✓	↑	–	
ADDC/ADDCB	2	D ← D + A + C	↓	✓	✓	✓	↑	–	
SUB/SUBB	2	D ← D − A	✓	✓	✓	✓	↑	–	
SUB/SUBB	3	D ← B − A	✓	✓	✓	✓	↑	–	
SUBC/SUBCB	2	D ← D − A + C − 1	↓	✓	✓	✓	↑	–	
CMP/CMPB	2	D − A	✓	✓	✓	✓	↑	–	
MUL/MULU	2	D,D + 2 ← D × A	–	–	–	–	–	–	2
MUL/MULU	3	D,D + 2 ← B × A	–	–	–	–	–	–	2
MULB/MULUB	2	D,D + 1 ← D × A	–	–	–	–	–	–	3
MULB/MULUB	3	D,D + 1 ← B × A	–	–	–	–	–	–	3
DIVU	2	D ← (D,D + 2) /A,D + 2 ← remainder	–	–	–	✓	↑	–	2
DIVUB	2	D ← (D,D + 1) /A,D + 1 ← remainder	–	–	–	✓	↑	–	3
DIV	2	D ← (D,D + 2) /A,D + 2 ← remainder	–	–	–	✓	↑	–	
DIVB	2	D ← (D,D + 1) /A,D + 1 ← remainder	–	–	–	✓	↑	–	
AND/ANDB	2	D ← D AND A	✓	✓	0	0	–	–	
AND/ANDB	3	D ← B AND A	✓	✓	0	0	–	–	
OR/ORB	2	D ← D OR A	✓	✓	0	0	–	–	
XOR/XORB	2	D ← D (ecxl. or) A	✓	✓	0	0	–	–	
LD/LDB	2	D ← A	–	–	–	–	–	–	
ST/STB	2	A ← D	–	–	–	–	–	–	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	–	–	–	–	–	–	3,4
LDBZE	2	D ← A; D + 1 ← 0	–	–	–	–	–	–	3,4
PUSH	1	SP ← SP − 2; (SP) ← A	–	–	–	–	–	–	
POP	1	A ← (SP); SP + 2	–	–	–	–	–	–	
PUSHF	0	SP ← SP − 2; (SP) ← PSW; PSW ← 0000H; I ← 0	0	0	0	0	0	0	
POPF	0	PSW ← (SP); SP ← SP + 2; I ← ✓	✓	✓	✓	✓	✓	✓	
SJMP	1	PC ← PC + 11-bit offset	–	–	–	–	–	–	5
LJMP	1	PC ← PC + 16-bit offset	–	–	–	–	–	–	5
BR[indirect]	1	PC ← (A)	–	–	–	–	–	–	
SCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 11-bit offset	–	–	–	–	–	–	5
LCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 16-bit offset	–	–	–	–	–	–	5

intel

Table 3-1. Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
RET	0	PC ← (SP); SP ← SP + 2	–	–	–	–	–	–	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	–	–	–	–	–	–	5
JC	1	Jump if C = 1	–	–	–	–	–	–	5
JNC	1	Jump if C = 0	–	–	–	–	–	–	5
JE	1	Jump if Z = 1	–	–	–	–	–	–	5
JNE	1	Jump if Z = 0	–	–	–	–	–	–	5
JGE	1	Jump if N = 0	–	–	–	–	–	–	5
JLT	1	Jump if N = 1	–	–	–	–	–	–	5
JGT	1	Jump if N = 0 and Z = 0	–	–	–	–	–	–	5
JLE	1	Jump if N = 1 or Z = 1	–	–	–	–	–	–	5
JH	1	Jump if C = 1 and Z = 0	–	–	–	–	–	–	5
JNH	1	Jump if C = 0 or Z = 1	–	–	–	–	–	–	5
JV	1	Jump if V = 1	–	–	–	–	–	–	5
JNV	1	Jump if V = 0	–	–	–	–	–	–	5
JVT	1	Jump if VT = 1; Clear VT	–	–	–	–	0	–	5
JNVT	1	Jump if VT = 0; Clear VT	–	–	–	–	0	–	5
JST	1	Jump if ST = 1	–	–	–	–	–	–	5
JNST	1	Jump if ST = 0	–	–	–	–	–	–	5
JBS	3	Jump if Specified Bit = 1	–	–	–	–	–	–	5,6
JBC	3	Jump if Specified Bit = 0	–	–	–	–	–	–	5,6
DJNZ/ DJNZW	1	D ← D − 1; If D ≠ 0 then PC ← PC + 8-bit offset	–	–	–	–	–	–	5 10
DEC/DECB	1	D ← D − 1	↕	↕	↕	↕	↑	–	
NEG/NEGB	1	D ← 0 − D	↕	↕	↕	↕	↑	–	
INC/INCB	1	D ← D + 1	↕	↕	↕	↕	↑	–	
EXT	1	D ← D; D + 2 ← Sign (D)	↕	↕	0	0	–	–	2
EXTB	1	D ← D; D + 1 ← Sign (D)	↕	↕	0	0	–	–	3
NOT/NOTB	1	D ← Logical Not (D)	↕	↕	0	0	–	–	
CLR/CLRB	1	D ← 0	1	0	0	0	–	–	
SHL/SHLB/SHLL	2	C ← msb - - - - - lsb ← 0	↕	↕	↕	↕	↑	–	7
SHR/SHRB/SHRL	2	0 → msb - - - - - lsb → C	↕	↕	↕	0	–	↕	7
SHRA/SHRAB/SHRAL	2	msb → msb - - - - - lsb → C	↕	↕	↕	0	–	↕	7
SETC	0	C ← 1	–	–	1	–	–	–	
CLRC	0	C ← 0	–	–	0	–	–	–	

Table 3-1. Instruction Summary (Continued)

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
CLRVT	0	VT ← 0	–	–	–	–	0	–	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	–	–	–	–	–	–	
EI	0	Enable All Interrupts (I ← 1)	–	–	–	–	–	–	
NOP	0	PC ← PC + 1	–	–	–	–	–	–	
SKIP	0	PC ← PC + 2	–	–	–	–	–	–	
NORML	2	Left shift till msb = 1; D ← shift count	✓	✓	0	–	–	–	7
TRAP	0	SP ← SP – 2; (SP) ← PC; PC ← (2010H)	–	–	–	–	–	–	9
PUSHA	1	SP ← SP-2; (SP) ← PSW; PSW ← 0000H; SP ← SP-2; (SP) ← IMASK1/WSR; IMASK1 ← 00H	0	0	0	0	0	0	
POPA	1	IMASK1/WSR ← (SP); SP ← SP+2 PSW ← (SP); SP ← SP+2	✓	✓	✓	✓	✓	✓	
IDLPD	1	IDLE MODE IF KEY=1; POWERDOWN MODE IF KEY =2; CHIP RESET OTHERWISE	–	–	–	–	–	–	
CMPL	2	D-A	✓	✓	✓	✓	↑	–	
BMOV	2	[PTR_HI]+ ← [PTR_LOW]+ ; UNTIL COUNT=0	–	–	–	–	–	–	

NOTES:
1. If the mnemonic ends in "B" a byte operation is performed, otherwise a word operation is done. Operands D, B and A must conform to the alignment rules for the required operand type. D and B are locations in the Register File; A can be located anywhere in memory.
2. D,D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D,D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to word.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling $\overline{\text{RESET}}$ low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.
10. The DJNZW instruction is not guaranteed to work. See Functional Deviations section.

Flag Settings. The modification to the flag setting is shown for each instruction. A checkmark (✓) means that the flag is set or cleared as appropriate. A hyphen (-) means that the flag is not modified. A one or zero (1) or (0) indicates that the flag will be in that state after the instruction. An up arrow (↑) indicates that the instruction may set the flag if it is appropriate but will not clear the flag. A down arrow (↓) indicates that the flag can be cleared but not set by the instruction.

3.2 Program Status Word

The Program Status Word (PSW) is a collection of Boolean flags which contain information concerning the state of the user's program. The high byte of the PSW contains status flags and the low byte contains an interrupt mask register. The PSW high byte is shown in Figure 3-2. Table 3-2 contains descriptions of the status flags.

Table 3-2. Status Flag Descriptions

Flag	Name	Function
ST	Sticky Bit	Indicates whether any 1's were lost due to a right shift operation; primarily used for floating-point routines.
I	Interrupt Enable	Master control for 80C196KB interrupts
C	Carry Flag	Set if there is a carry (or no borrow), and otherwise cleared, as a result of an ADD or SUB instruction.
VT	Overflow Trap Flag	Set whenever overflow flag is set; cleared only by a CLRVT, JVT or JNVT instruction.
V	Overflow Flag	Set if result is out of range for signed arithmetic operation.
N	Negative Flag	Holds the algebraically correct sign as the result of an operation.
Z	Zero Flag	Set if the result of an operation is zero.

PSW:

15	14	13	12	11	10	9	8
Z	N	V	VT	C	X	I	ST

Figure 3-2. The Program Status Word Register (High Byte)

4.0 INTERRUPTS

There are 28 different interrupt sources available on the 80C196KB. The 28 sources vector through 18 locations or interrupt vectors. The vector names and their sources are shown in Figure 4-1, and their locations are listed in Table 4-1. The four registers that control the interrupt system are: INT_PEND, INT_PEND1, INT_MASK, INT_MASK1. The Program Status Word (PSW) contains a global disable bit, I, which is set or cleared using the EI or DI instructions. Figure 4-2 shows a block diagram of the interrupt structure.

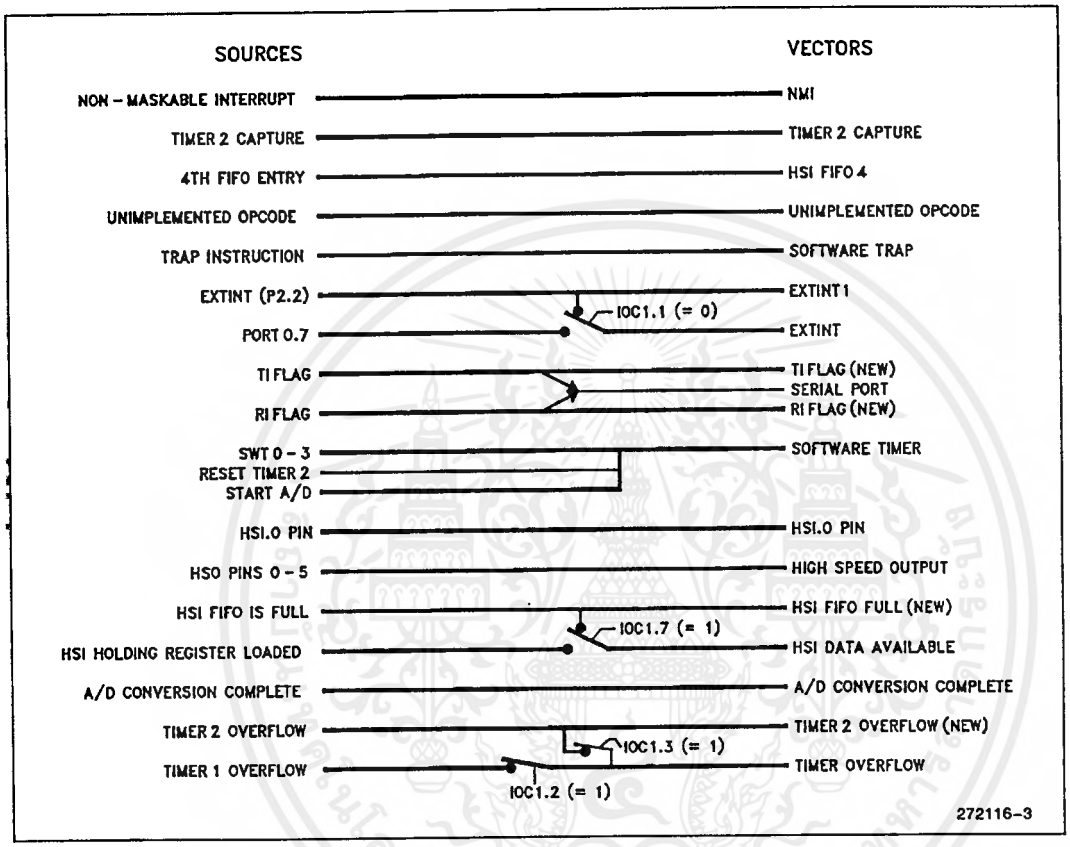


Figure 4-1. 80C196KB Interrupt Sources

Table 4-1. 80C196KB Interrupt Vector Locations

Number	Vector Name	Vector Location	Priority
INT15	NMI	203EH	15
INT14	HSI FIFO Full	203CH	14
INT13	EXTINT1	203AH	13
INT12	TIMER2 Overflow	2038H	12
INT11	TIMER2 Capture	2036H	11
INT10	4th Entry into HSI FIFO	2034H	10
INT09	RI	2032H	9
INT08	TI	2030H	8
SPECIAL	Unimplemented Opcode	2012H	N/A
SPECIAL	Trap	2010H	N/A
INT07	EXTINT	200EH	7
INT06	Serial Port	200CH	6
INT05	Software Timer	200AH	5
INT04	HSI.0 Pin	2008H	4
INT03	High Speed Outputs	2006H	3
INT02	HSI Data Available	2004H	2
INT01	A/D Conversion Complete	2002H	1
INT00	Timer Overflow	2000H	0

NOTE:
Priority 15 = highest, priority 0 = lowest

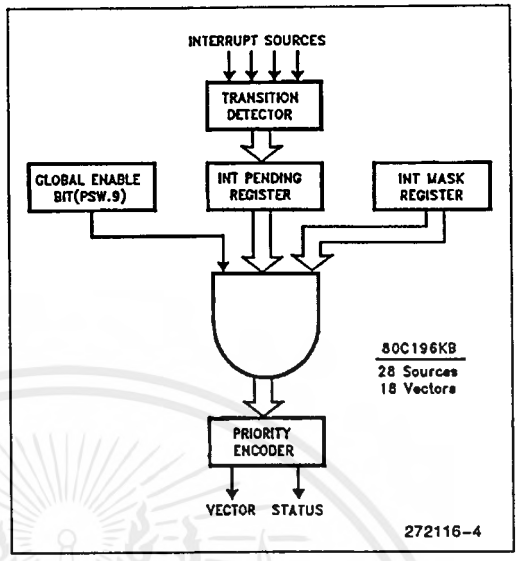


Figure 4-2. 80C196KB Interrupt Structure Block Diagram

Three special interrupts are available on the 80C196KB: the external Non-Maskable Interrupt (NMI), TRAP and Unimplemented Opcode. The external NMI pin generates an unmaskable interrupt for implementation of critical interrupt routines. The TRAP instruction is useful for developing custom software debuggers or generating software interrupts. The Unimplemented Opcode Interrupt generates an interrupt upon execution of unimplemented opcodes. This provides software recovery from random execution during hardware or software failures.

When the hardware detects one of the sixteen interrupts it sets the corresponding bit in one of two interrupt pending registers (INT_PEND and INT_PEND1). Individual interrupts are enabled or disabled by setting or clearing bits in the mask registers (INT_MASK and INT_MASK1). A one in any bit position will enable the corresponding interrupt source and a zero will disable it. The interrupt mask and pending registers are shown in Figure 4-3.

	7	6	5	4	3	2	1	0
12H INT_PEND1: 13H INT_MASK1:	NMI	FIFO FULL	EXT INT1	T2 OVF	T2 CAP	HSI4	RI	TI

	7	6	5	4	3	2	1	0
09H INT_PEND: 08H INT_MASK:	EXT INT	SER PORT	SOFT TIMER	HSI.0 PIN	HSO PIN	HSI DATA	A/D DONE	TIMER OVF

Figure 4-3. Interrupt Mask and Pending Registers

The priority encoder looks at all the interrupts that are both pending and enabled, and selects the one with the highest priority. The priorities are shown in Table 4-1 (15 is highest, 0 is lowest). When the interrupt controller decides to process an interrupt, it executes a "call" to an Interrupt Service Routine (ISR). The address of the ISR is contained in the corresponding interrupt vector location. The interrupt controller clears the associated pending bit then pushes the return address onto the stack. The ISR should use the PUSHA instruction to save the PSW, INT_MASK, INT_MASK1 and WSR on the stack. The PUSHA instruction also clears the PSW and interrupt mask registers, disabling all interrupts. The ISR software must then implement the interrupt priority structure desired for that routine by enabling only the desired interrupts. At the end of the ISR, the POPA instruction restores the PSW, INT_MASK, INT_MASK1 and WSR to their original states and restores the original priority structure. In most cases an Interrupt Service Routine will have the basic structure shown below.

```
INT_VECTOR:  PUSHA                        ; Save the PSW, INT_MASK,
                                          ;INT_MASK1, and WSR
             LDB INT_MASK, #xxxxxxxxB     ;Set-up New Interrupt
             LDB INT_MASK1,#xxxxxxxxB     ;Priorities
             EI                           ;Enable Interrupts Again
             -
             -                            ;Service the Interrupt
             -
             POPA                         ;Restore
             RET
```

5.0 TIMERS/COUNTERS

The 80C196KB has two 16-bit timers, Timer1 and Timer2, shown in Figure 5-1. Timer1 is readable in Window 0 and writable in Window 15 while Timer2 is readable and writable in Window 0. The 80C196KB also includes separate, dedicated timers for the baud rate generator and watchdog timer. The watchdog timer is an internal timer that can be used to reset the system if the software fails to operate properly.

The Timer1 value is incremented by the 80C196KB internal clock every 8 state times. (A state time is 2 oscillator periods, or 0.167 μs with a 12 MHz crystal.) Timer1 generates a Timer Overflow Interrupt (INT00) when crossing the 0FFFFH/0000H boundary. I/O Control Register 1 (IOC1) controls the Timer1 overflow interrupt. As shown in Figure 5-2, setting IOC1.2 enables Timer1 overflow to INT00. The status of Timer1 Overflow Interrupt is read in I/O Status Register 1 (IOS1) shown in Figure 5-3.

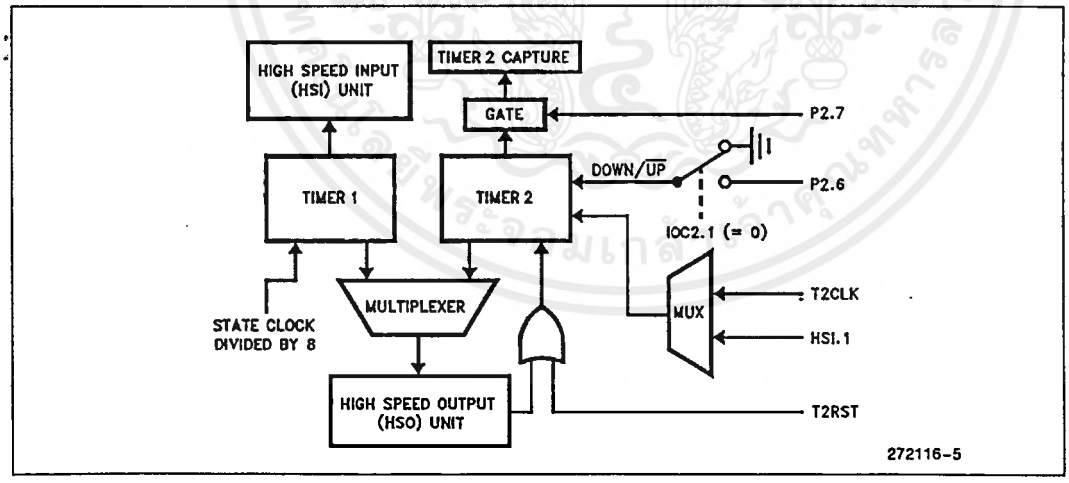


Figure 5-1. Timer Block Diagram

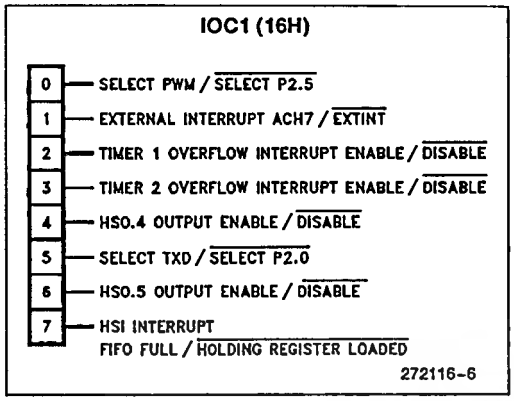


Figure 5-2. I/O Control Register 1 (IOC1)

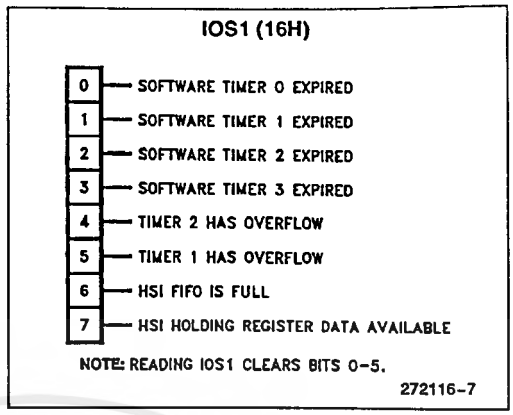


Figure 5-3. I/O Status Register 1 (IOS1)

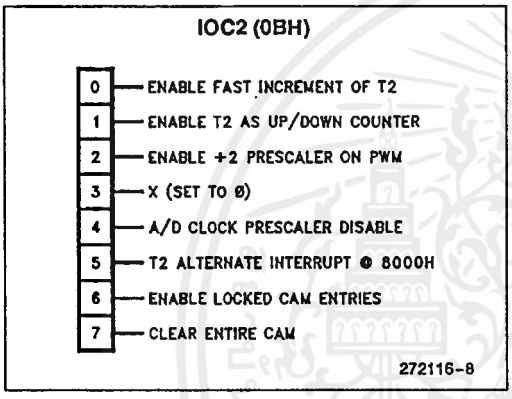


Figure 5-4. I/O Control Register 2 (IOC2)

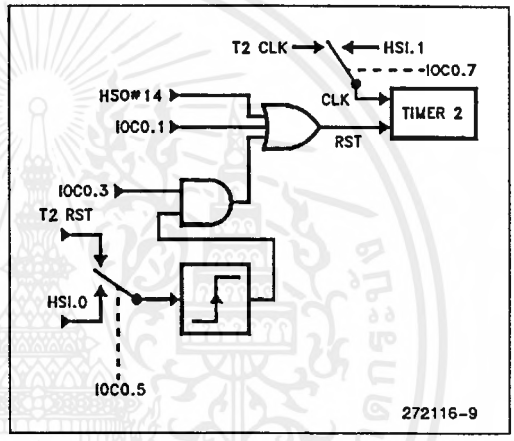


Figure 5-5. Timer2 Clock and Reset Options

I/O Control Register 1 (IOC1) and I/O Control Register 2 (IOC2) shown in Figure 5-4 determine the function of Timer2. Timer2 is driven by an external clock. Bit 7 of IOC0 controls whether the T2CLK pin or the HSI.1 pin function as the Timer2 clock input. Timer2 increments or decrements on every positive and negative transition. Bit 0 of IOC2 determines the maximum rate at which Timer2 can receive these transitions. When IOC2.0 = 1 the maximum transition speed is once per state time, and when IOC2.0 = 0 the maximum transition speed is once every 8 state times (Fast Increment Mode). Setting bit 1 of IOC2 enables Timer2 to function as an up/down counter. The T2UPDN pin determines the direction of Timer2 as an up/down counter; when T2UPDN = 1 Timer2 counts down and when T2UPDN = 0 Timer2 counts up. There are two possible external Timer2 reset sources. IOC0.3 enables the external reset function and IOC0.5 determines whether the T2RST pin or the HSI.0 pin will act as the reset source (Figure 6-4). It is also possible to reset Timer2 internally using the High Speed Output Unit or by clearing the Timer2 SFR. Figure 5-5 shows the Timer2 clock and reset options and Table 5-1 lists the Timer2 control bits.

Table 5-1. Timer2 Control Bits

	Bit = 1	Bit = 0
IOC0.1	Reset Timer2 each write	No action
IOC0.3	Enable external reset	Disable
IOC0.5	HSI.0 is ext. reset source	T2RST is reset source
IOC0.7	HSI.1 is T2 clock source	T2CLK is clock source
IOC1.3	Enable Timer2 overflow int.	Disable overflow interrupt
IOC2.0	Enable fast increment	Disable fast increment
IOC2.1	Enable downcount feature	Disable downcount
P2.6	Count down if IOC2.1 = 1	Count up
IOC2.5	Interrupt on 7FFFH/8000H	Interrupt on 0FFFFH/0000H
P2.7	Capture Timer2 into T2CAPTURE on rising edge	

Timer2 can generate three interrupts: The Timer Overflow Interrupt (INT00), The Timer2 Overflow Interrupt (INT12), and The Timer2 Capture Interrupt (INT11). IOC1 determines whether Timer1 and/or Timer2 will generate INT00. Timer2 generates an overflow interrupt when crossing the 0FFFFH/0000H boundary or the 7FFFH/8000H boundary as determined by IOC2.5. A Timer2 overflow interrupts through INT00 if IOC1.3 and INT_MASK.0 are set. Alternatively, Timer2 interrupts through INT12 if INT_MASK1.3 is set. Bit 4 of I/O Status Register 1 (IOS1.4), shown in Figure 5-3, indicates that status of Timer2 Overflow Interrupt.

6.0 HIGH SPEED INPUT UNIT

The High Speed Input Unit (HSI) can record times of external events with an 8 state time (1.33 μs at 12 MHz) resolution. It can capture the value of Timer1 when an event takes place on one of the four HSI lines (HSI.0 through HSI.3). The four types of events that can trigger a capture are: rising edges only, falling edges only, rising or falling edges, or every eighth rising edge. As shown in Figure 6-2, the four input lines are independently configurable via the HSI_MODE register. This register determines the capture modes of the four inputs. A block diagram of the HSI unit is shown in Figure 6-1.

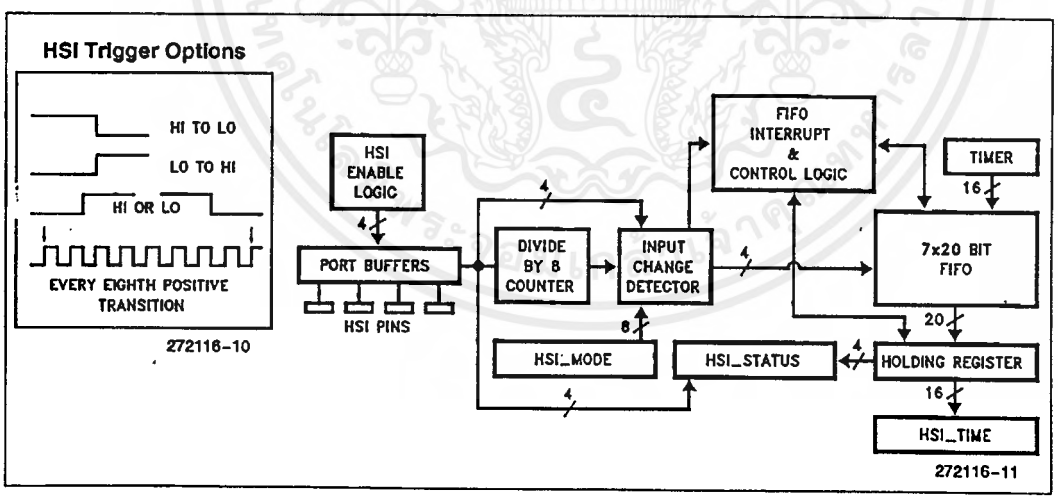


Figure 6-1. High Speed Input Unit

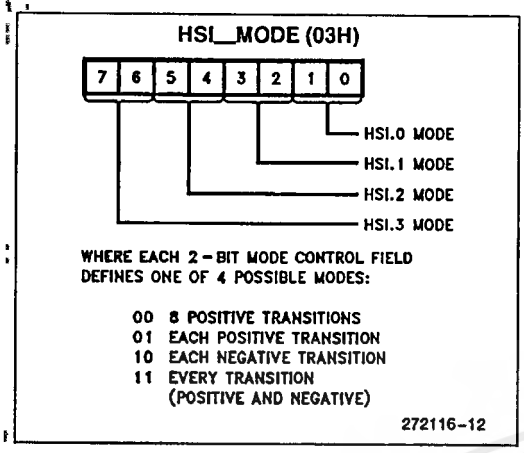


Figure 6-2. High Speed Input Mode Register (HSI_MODE)

The HSI unit stores the Timer1 value and 4 status bits in a 7 x 20 level FIFO and holding register. It is possible to store 8 entries, 7 in the FIFO and 1 in the holding register. The HSI unit will not store events occurring after the FIFO is full. The HSI holding register contains the earliest entry placed in the FIFO. Reading the holding register unloads one level of the FIFO. The HSI unit then places the next entry into the holding register.

The contents of the HSI holding register are obtained by first reading the HSI_STATUS register and then the HSI_TIME register. The HSI_TIME register returns the event time tag. The HSI_STATUS register returns a status and an input bit for each of the four HSI lines (see Figure 6-3). The status bit indicates which line(s) caused the event and the input bit indicates the current input level of the line, not the level when the event occurred. Reading the HSI_TIME register unloads one level of the FIFO.

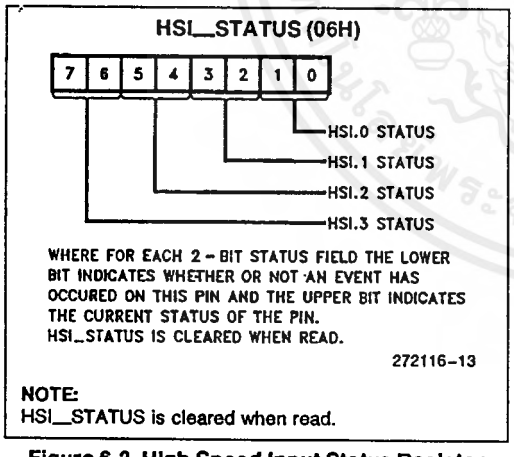


Figure 6-3. High Speed Input Status Register (HSI_STATUS)

To start the HSI use the following steps: 1) Flush the FIFO, 2) Enable the HSI interrupts, 3) Initialize and enable the HSI pins. The following section of code will flush the FIFO:

```
FLUSH:  LD ZERO_REG,        ;Unload one level of
        HSI_TIME            the FIFO
        SKIP ZERO_REG       ;Wait 4 state times
        SKIP ZERO_REG       ;Wait 4 state times
        JBS IOS1, 7, FLUSH  ;Check whether FIFO
                            is empty
```

I/O Control Register 0 (IOC0), shown in Figure 6-4, can individually enable or disable the four HSI lines (HSI.0 through HSI.3). Disabling an input line disconnects it from the FIFO, changing its function from an HSI line to a general purpose input line. However, the corresponding HSI_STATUS input bits indicate the current state of the line regardless of whether the line functions as an HSI input line or as a general purpose input line.

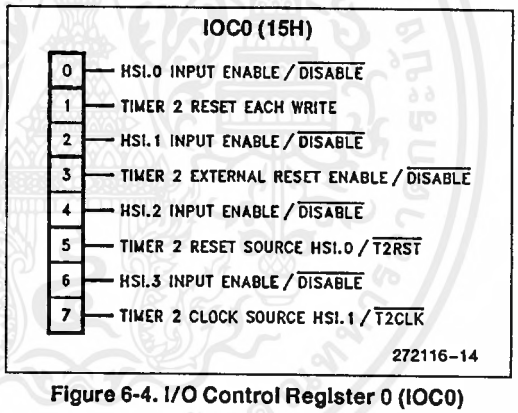


Figure 6-4. I/O Control Register 0 (IOC0)

The HSI unit can generate three interrupts: The HSI Data Available Interrupt (INT02), the HSI_FIFO_4 Interrupt (INT10) and the HSI FIFO FULL Interrupt (INT14). Bit 7 of I/O Control Register 1 (IOC1) controls the INT02 source. If IOC1.7 = 0 loading the holding register will cause INT02; otherwise if IOC1.7 = 1 loading the sixth entry into the FIFO (not including the holding register) will cause INT02. After INT02 occurs bits 6 and 7 of I/O Status Register 1 (IOS1) indicate which source caused the interrupt. The sources for INT10 and INT14 are independent of IOC1. Loading the fourth entry into the FIFO causes INT10 and loading the sixth entry into the FIFO causes INT14. Note if IOC1.7 is set, loading the sixth entry into the FIFO will cause both INT02 and INT14.

7.0 HIGH SPEED OUTPUT UNIT

The HSO unit can trigger events at specified times based on Timer1 or Timer2. These programmable events include: starting an A/D conversion, resetting Timer2, generating up to four software time delays and setting or clearing one or more of the 6 output lines (HSO.0 through HSO.5). The HSO unit stores pending events and their specified times in a CAM (Content Addressable Memory) file. Figure 7-1 shows a block diagram of the HSO unit.

The CAM file is the main component of the HSO. This file stores up to eight commands. Each CAM register is 24 bits wide. Sixteen bits specify the action time, and 8 bits specify the nature of the action and whether Timer1 or Timer2 is the reference. Timer2 transitions should not occur faster than once every 8 state times when it is used as a reference for the HSO. Commands for the HSO first enter the HSO holding register. They then enter the CAM when an empty CAM register is available. Commands must be in the CAM to execute; commands in the holding register will not execute. It takes one state time to compare each CAM location, so 8 state times (1.33 μs with a 12 MHz clock) are necessary for a complete CAM search. The HSO unit triggers the specified event when it finds a time match.

Writing to the HSO_COMMAND register and the HSO_TIME register loads the HSO holding register. When the next opening in the CAM file is available the contents of the HSO holding register move into it. The HSO_COMMAND register shown in Figure 7-2 specifies the event type, whether an interrupt is to occur, and the reference timer. The I/O Status Register 0 (IOS0) bits 6 and 7 indicate the status of the HSO unit. If IOS0.6 equals 0, the holding register is empty and at least one CAM register is empty. If IOS0.7 equals 0, the holding register is empty. The holding register must be empty before writing the action time to the HSO_TIME registers. If the holding register is not empty, writing to the HSO will overwrite the current holding register value. Always write the command byte first, followed by the time word.

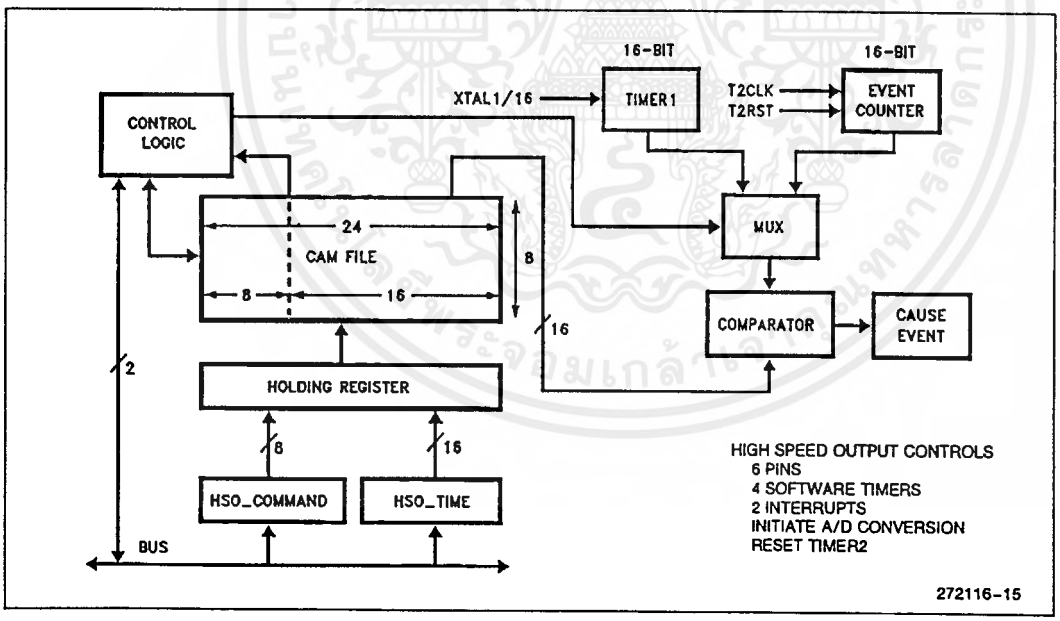


Figure 7-1. High Speed Output Block Diagram

HSO_COMMAND (06H)

	7	6	5	4	3	2	1	0
HSO_ COMMAND	CAM LOCK	TMR2/ $\overline{\text{TMR1}}$	SET/ CLEAR	INT/ $\overline{\text{INT}}$	CHANNEL			

CAM Lock — Locks event in CAM if this is enabled by IOC2.6 (ENA_LOCK)
TMR/$\overline{\text{TMR1}}$ — Events Based on Timer2/Based on Timer1 if 0
SET/$\overline{\text{CLEAR}}$ — Set HSO pin/Clear HSO pin if 0
INT/$\overline{\text{INT}}$ — Cause interrupt/No interrupt if 0
CHANNEL: (in Hex)
0–5: HSO pins 0–5 separately
6: HSO pins 0 and 1 together
7: HSO pins 2 and 3 together
8–B: Software Timers 0–3
C–D: Unflagged Events (Do not use for future compatibility)
E: Reset Timer2
F: Start A to D Conversion

Figure 7-2. High Speed Output Command Register (HSO_Command)

An entry placed in the CAM remains there until its execution unless a chip reset occurs or the CAM clear bit (IOC2.7) is set. It is possible to cancel an external pending event by writing the opposite event with the same time tag to the CAM. However, both events remain in the CAM until their time tag is matched or the CAM is cleared. Setting bit 2 of IOC2 enables the CAM locking function. Setting the CAM lock bit (HSO_COMMAND.7) locks the command in the CAM; a locked CAM entry will execute whenever its time tag matches the reference time. Locked entries are useful in applications requiring periodic or repetitive events to occur. The HSO unit can generate multiple PWM's by locking CAM entries and using Timer2 as a reference. (See Software Example 4)

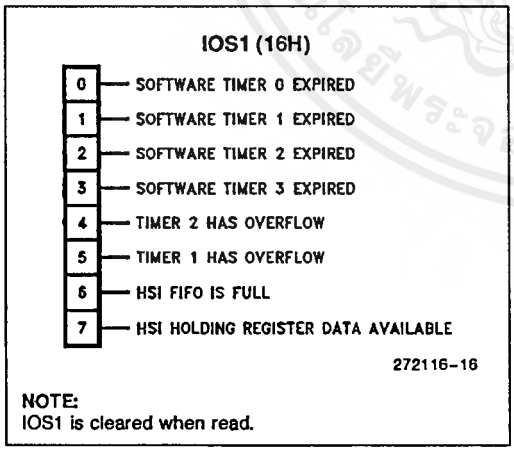


Figure 7-3. I/O Status Register 1 (IOS1)

The HSO unit can generate two interrupts (providing HSO_COMMAND.4 is set): The High Speed Output interrupt (INT03) and The Software Timer interrupt (INT05). The High Speed Output interrupt occurs as a result of changes on one or more of the six output pins. The other HSO commands, triggering the A/D Converter, resetting Timer2 and setting a Software Timer Flag, generate INT05. The I/O Status Registers IOS1 and IOS2, shown in Figure 7-3 and Figure 7-4 indicate which event(s) caused a HSO interrupt.

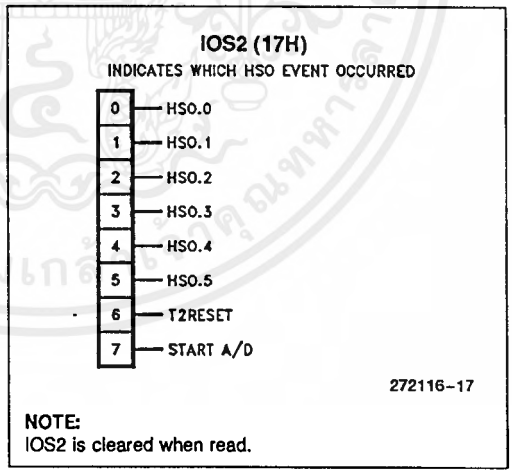


Figure 7-4. I/O Status Register 2 (IOS2)

The HSO unit can generate interrupts at preset times via four "Software Timers". Software Timer Flags are set in the I/O Status Register 1 (IOS1) at the prepro-

grammed times. If the interrupt bit in the HSO command register is set, a Software Timer Interrupt will also occur at the designated time. The interrupt service routine can then examine IOS1 to determine which software timer expired and caused the interrupt. The most common use of the software timers is to trigger interrupt routines that must occur at regular intervals.

8.0 PULSE WIDTH MODULATION OUTPUT

The Pulse Width Modulator of the 80C196KB, when used with external hardware, can provide useful signals for a variety of applications. The PWM output can perform digital to analog conversions and drive several types of motors which require a PWM waveform for more efficient operation. A block diagram of the PWM circuit is shown in Figure 8-1. Three registers control the PWM: I/O Control Register 1 (IOC1), I/O Control Register 2 (IOC2) and the PWM Register (PWM_CONTROL). The PWM output shares a pin with Port 2; setting IOC1.0 selects the PWM function rather then the standard port function.

The PWM output waveform is a variable duty cycle pulse that repeats every 256 state times (42.75 μs @ 12 MHz) or 512 state times (85.5 μs @ 12 MHz) if the prescaler bit (IOC2.2) is set. The PWM frequencies for different clock speeds are shown in Table 8-1. Writing a value between 0 and 255 to the PWM_CONTROL register will change the duty cycle. The PWM unit has an 8-bit counter that is incremented every state time or every other state time if the prescaler bit is set. When

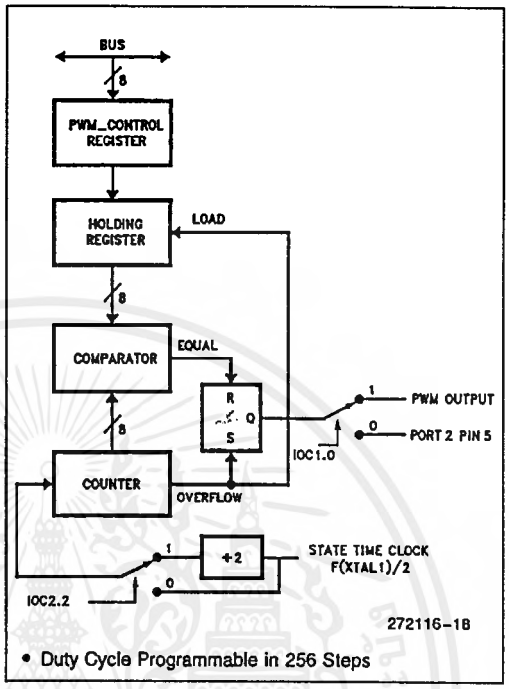


Figure 8-1. PWM Block Diagram

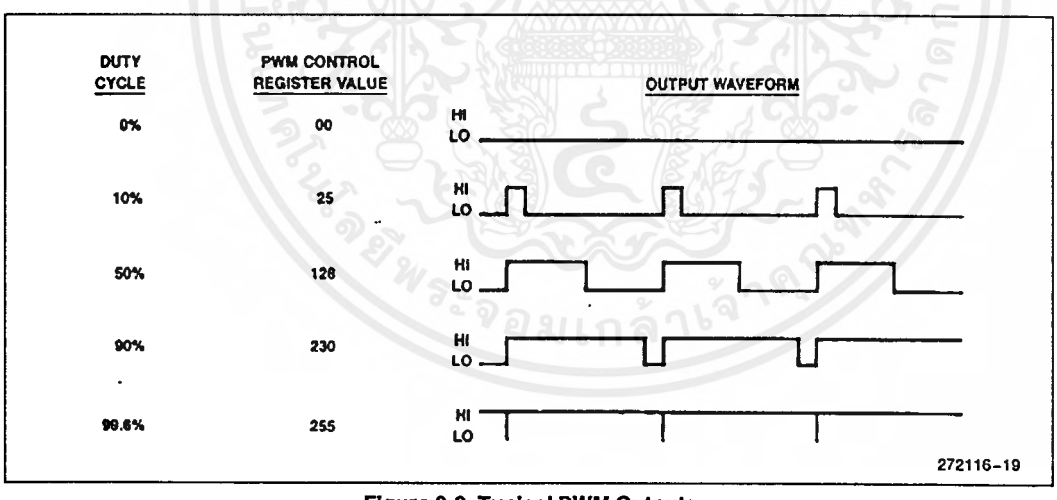


Figure 8-2. Typical PWM Outputs

the counter equals 0 the PWM output switches high; when the counter matches the value in the PWM_CONTROL register the PWM output switches low; and when the counter overflows the PWM output switches high again. Typical output waveforms are shown in Figure 8-2. Values written to PWM_CONTROL are loaded into a holding register when the counter overflows. This is so the compare circuit will not recognize a new value until the counter has expired, thus preventing missed PWM edges.

Table 8-1. PWM Frequencies

XTAL1 =	8 MHz	10 MHz	12 MHz
IOC2.2 = 0	15.6 KHz	19.6 KHz	23.6 KHz
IOC2.2 = 1	7.8 KHz	9.8 KHz	11.8 KHz

9.0 ANALOG OUTPUTS

Both the PWM output and the HSO unit can generate analog outputs. Either peripheral will generate a rectangular pulse train that varies in duty cycle and period. Filtering the output will create a smooth analog signal. This filtering is typically done after the signal is buffered to make it swing over the desired analog output voltage range. A block diagram of the type of circuit needed is shown in Figure 9-1. The filter can be a simple RC network or an active filter. Shown in Figure 9-2 is a circuit used for low output currents, (less than 100 μA or so). The PWM unit can generate these waveforms if a fixed period on the order of 42.75 μs or 85.5 μs (at 12 MHz) is acceptable. The HSO unit can generate waveforms with a period of up to 87.5 ms (using Timer1 at 12 MHz).

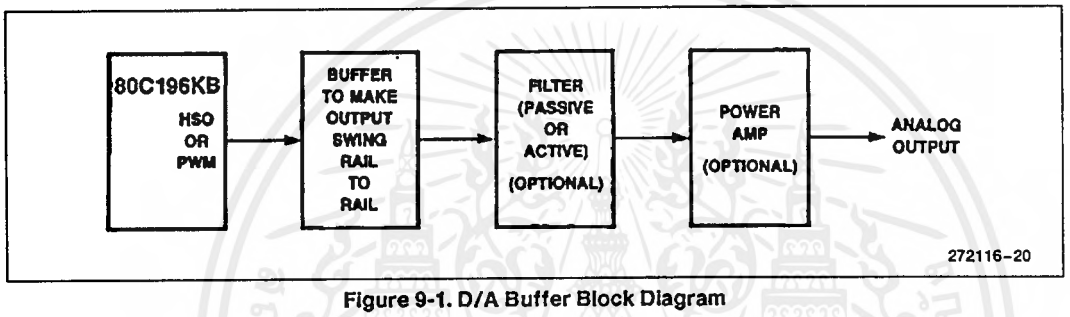


Figure 9-1. D/A Buffer Block Diagram

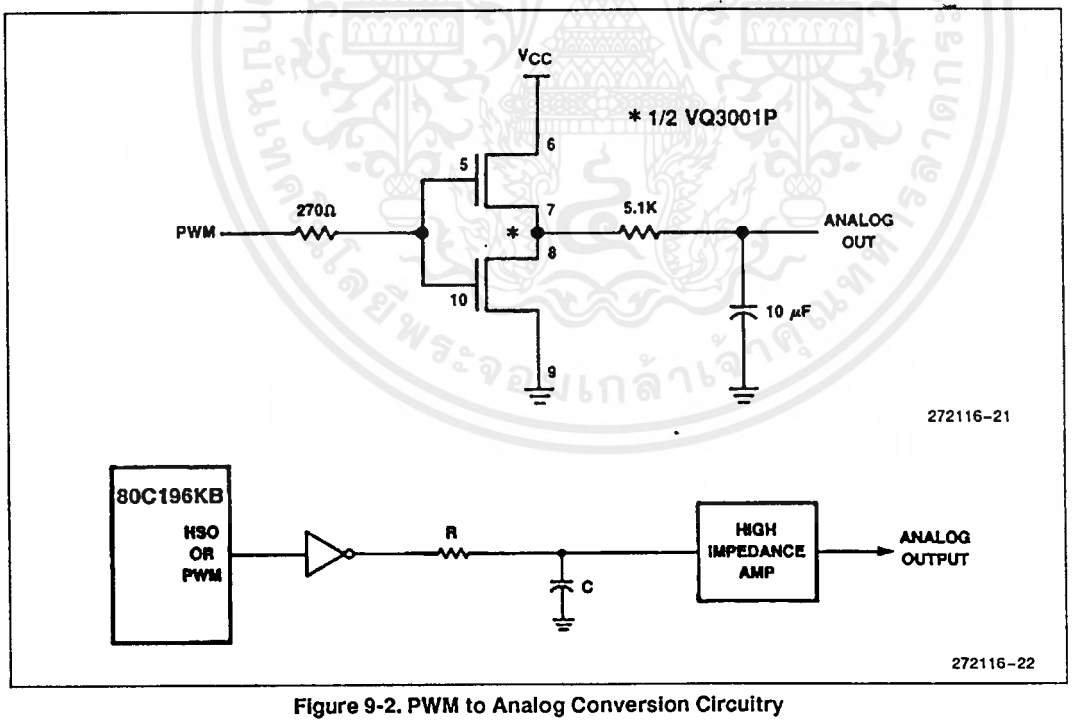


Figure 9-2. PWM to Analog Conversion Circuitry

10.0 ANALOG TO DIGITAL CONVERTER

The 80C196KB analog interface consists of a sample-and-hold, an 8 channel multiplexer, and a 10-bit analog-to-digital converter. A block diagram of the A/D converter is shown in Figure 10-1. Port 0, an input-only port, shares the analog inputs ACH0 through ACH7. The A/D Converter uses the successive approximation method to perform an A/D conversion on one input at a time. Three SFRs control the A/D Converter. The AD__COMMAND register controls which channel and when a conversion will start, and the AD__RESULT (low and high) registers store the 10-bit conversion result. Bit 4 of the I/O Control Register 2 (IOC2.4) controls the number of state times required for the conversion.

To set-up an analog-to-digital conversion load the desired analog input channel into the lower three bits of the AD__COMMAND register. The GO bit, bit 4 of the AD__COMMAND register, controls when the conversion will start. If the GO bit is set the conversion will start immediately, otherwise the HSO unit will trigger the conversion. The AD__COMMAND register is shown in Figure 10-2. The A/D result registers (AD__RESULT(hi) and AD__RESULT(lo)), shown in Figure 10-3 and Figure 10-4 contain the 10-bit conversion result. The AD__RESULT(hi) register contains the most significant 8 bits of the result. Bits 6 and 7 of the AD__RESULT(lo) register contain the remaining least significant bits (LSB's) of the result. Also, the lower four bits of the AD__RESULT(lo) register contain the A/D channel number and the A/D status as shown in Figure 10-3. The AD__RESULT(lo) status bit, when set, indicates that an A/D conversion is in progress. It takes 8 state times to set this bit after the start of an A/D conversion.

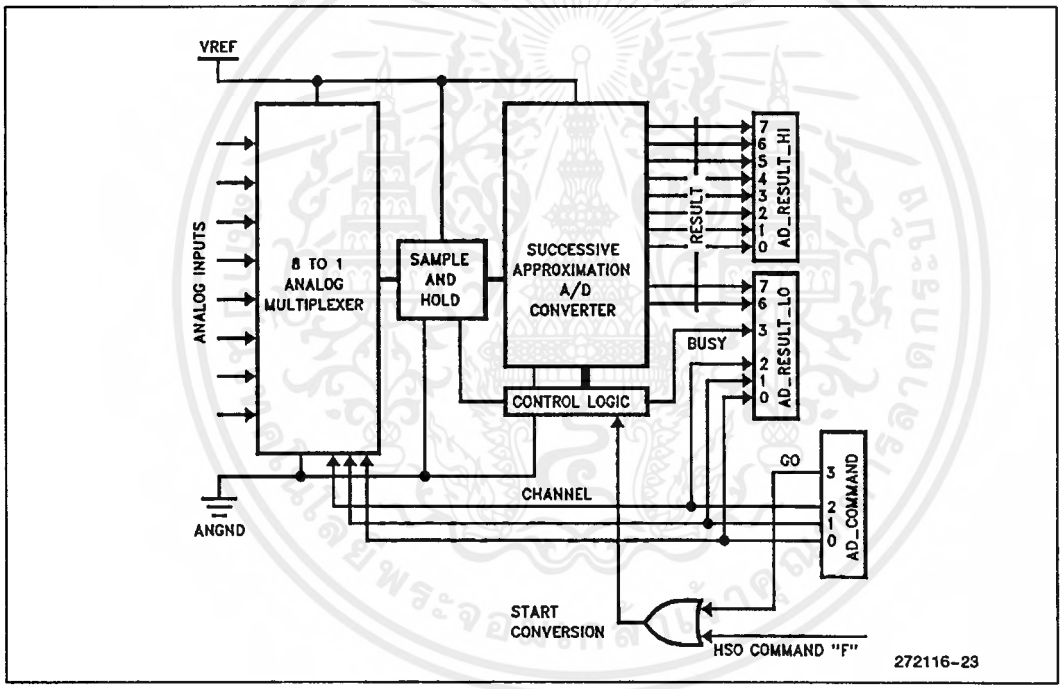


Figure 10-1. A/D Converter Block Diagram

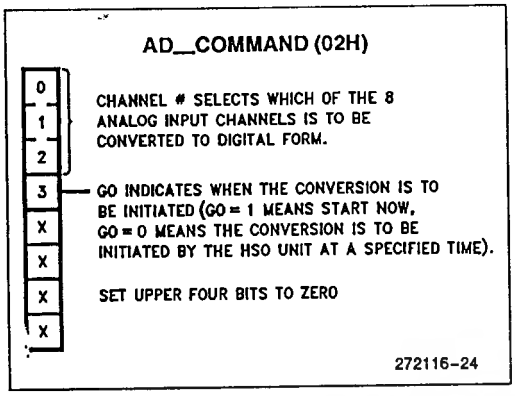


Figure 10-2. A/D Command Register (AD__COMMAND)

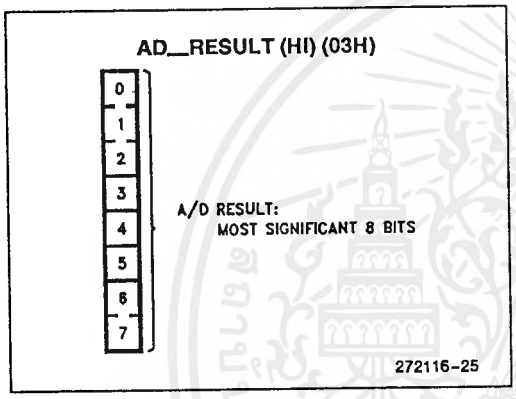


Figure 10-3. A/D Result High Register (AD__RESULT(HI))

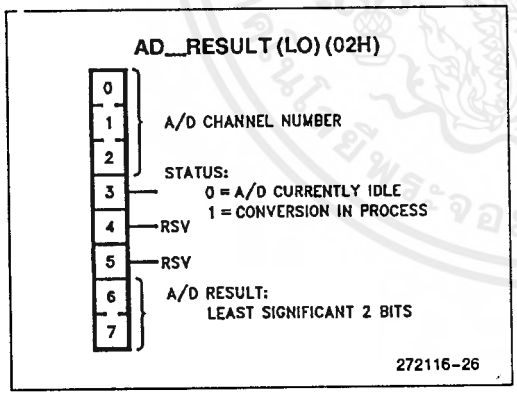


Figure 10-4. A/D Result Low Register (AD__RESULT(LO))

The clock prescaler bit of I/O Control Register 2 (IOC2.4) determines the number of state times required for an A/D conversion. High crystal frequencies require more states to complete a conversion to allow enough settling time for the internal comparator. When IOC2.4 = 1 the A/D conversion time is 91 state times (22.75 μs for an 8 MHz crystal) otherwise the A/D conversion time is 158 state times (26.33 μs for a 12 MHz crystal). An A/D Conversion Complete Interrupt (INT01) occurs on completion of an A/D conversion. It is possible to generate a Software Timer Interrupt (INT05) at the start of an A/D conversion by using the HSO unit to trigger the conversion.

11.0 SERIAL PORT

The Serial Port on the 80C196KB has one synchronous (Mode 0) and three asynchronous modes (Modes 1–3). The asynchronous modes are full duplex, meaning they can transmit and receive data simultaneously. The receiver on the 80C196KB is double buffered so the reception of a second byte may begin before the first byte is read. The transmitter is also double buffered and can generate continuous transmissions.

In the asynchronous modes, the TxD pin is the serial port transmission line and the RxD pin is the serial port reception line. Data to and from the serial port is transferred through the Serial Port Buffers. The Transmit Buffer SBUF(TX) contains data for transmission, the Receive Buffer SBUF(RX) stores the received data.

The Serial Port Control (SP__CON) register and the Serial Port Status (SP__STAT) register control the serial port. Bit 5 of the I/O Control Register 1 (IOC1), shown in Figure 5-2 enables the TxD pin for serial port use. Writing to location 11H in Window 0 accesses the SP__CON register while reading it accesses the SP__STAT register. The SP__CON register contains bits that: determine the Serial Mode (M1 and M2), enable parity (PEN), enable the receiver (REN), and determine the state and function of the 9th data bit when using Modes 2 and 3 (TB8). The SP__STAT register contains flags that indicate: receive Overrun Error (OE), Framing Error (FE), Transmitter Empty (TXE), Transmit Interrupt (TI), Receive Interrupt (RI), Receive Parity Error (RPE) and Receive Bit 8 (RB8). The SP__CON and SP__STAT registers are shown in Figure 11-1.

SP_CON:

7	6	5	4	3	2	1	0	
X	X	X	TB8	REN	PEN	M2	M1	11H

TB8 — Sets the ninth data bit for transmission. Cleared after each transmission. Not valid if parity is enabled.
REN — Enables the receiver
PEN — Enables the Parity function (even parity)
M2, M1 — Sets the mode. Mode0 = 00, Mode1 = 01, Mode2 = 10, Mode 3 = 11
X — Reserved bit. Must be written as 0.

SP_STAT:

7	6	5	4	3	2	1	0	
RB8/ RPE	RI	TI	FE	TXE	OE	X	X	11H

RB8 — Set if the 9th bit is high on reception (parity disabled)
RPE — Set if parity is enabled and a parity error occurred
RI — Set after the last data bit is sampled
TI — Set at the beginning of the STOP bit transmission
FE — Set if no STOP bit is found at the end of a reception
TXE — Set if two bytes can be transmitted
OE — Set if the receiver buffer is overwritten
Reading SP_STAT clears Bits 2, 4, 5, 6 and 7

Figure 11-1. Serial Port Control and Status Registers (SP_CON and SP_STAT)

The most common use of Mode 0, the synchronous mode, is to expand the I/O capability of the 80C196KB using shift registers. In this mode the port outputs a set of 8 clock pulses on the TxD pin and either transmits or receives data synchronously on the RxD pin. Data is transferred 8 bits at a time with the LSB first. A diagram of the relative timing of these signals is shown in Figure 11-2. A schematic of a typical circuit that uses shift registers is shown in Figure 11-3. Since this circuit inverts the input data bits, software must re-invert them. The users software routine must control two pins (PX.X) to load data into the 74165 and to enable the shift clock on the 74164.

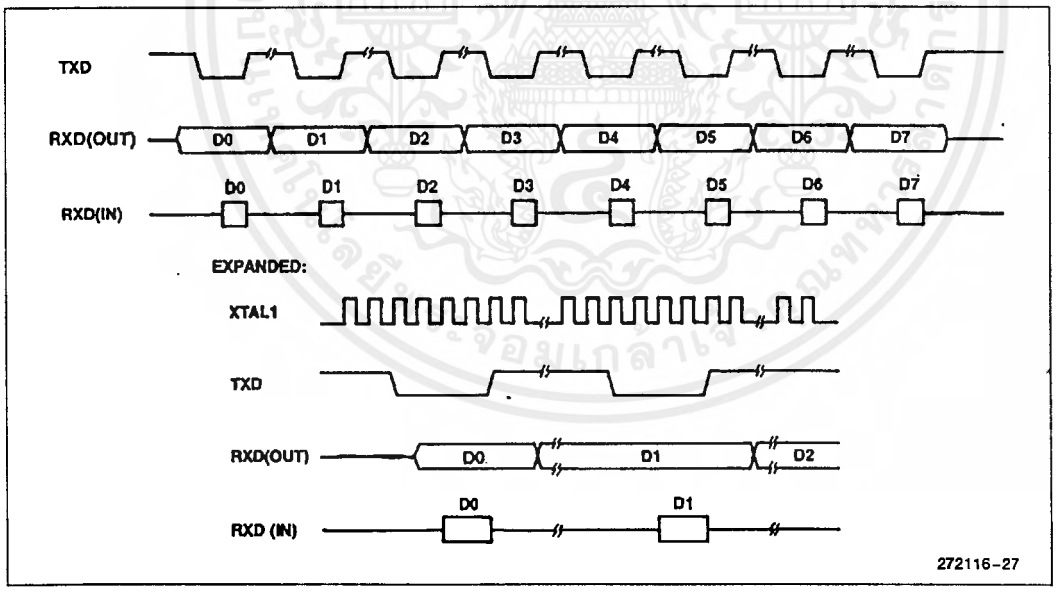


Figure 11-2. Mode 0 Timing

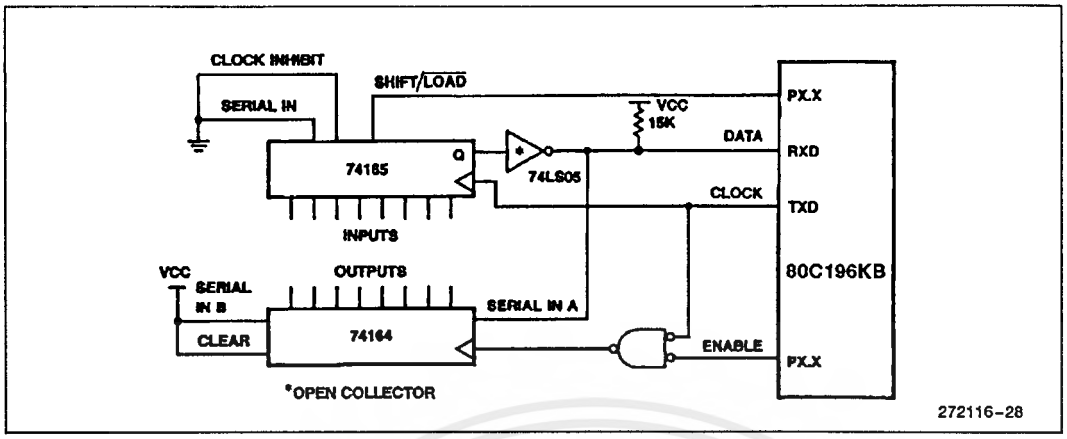


Figure 11-3. Typical Shift Register Circuit

Mode 1 is the standard asynchronous mode used for normal serial communication. The data frame used in this mode is shown in Figure 11-4. It consists of 10 bits: a start bit, 8 data bits (LSB first) and a stop bit. If parity is enabled (PEN = 1), an even parity bit is sent instead of the 8th data bit. Modes 2 and 3 are 9-bit modes commonly used for multi-processor communications. The data frame used in these modes, shown in Figure 11-4, consists of 11 bits: a start bit, nine data bits (LSB first) and a stop bit. Devices in Mode 2 will interrupt upon reception only if the 9th data bit is set. Devices in Mode 3 will always interrupt upon reception. Mode 3 also allows transmission of 8 data bits plus an even parity bit.

By making use of Modes 2 and 3 software can easily communicate between processors. Software sets the 9th data bit when sending an address or command for all the processors. In standby mode all the processors wait in Mode 2 for a byte with the 9th bit set. When they receive that byte, each processor determines if the next message is for them. The processor(s) that is to receive the message switches to Mode 3 and receives the information. Since the other processors remain in Mode 2, the software can send information with the 9th data bit cleared ensuring that only the previously addressed processor(s) will receive the information. This scheme minimizes the overall CPU time required for the serial port.

A typical connection diagram for multiprocessor communication is shown in Figure 11-5. This type of communication can connect peripherals to a desk top computer, an axis controller of a multi-axis machine, or any other group of microcontrollers.

The Serial Port sets the Transmit Interrupt (TI) and the Receive Interrupt (RI) flags in the SP_STAT register to indicate when operations are complete. TI is set when the last data bit is sent. RI is set when the last data bit is received (except in mode 2). In mode 2 the RI flag is set only when the 9th data bit of the reception is set. Reading the SP_STAT register clears the TI and RI flags. In response to the RI and TI flags the Serial Port generates three possible interrupts: the Transmit Interrupt (INT08), the Receive Interrupt (INT09) and the Serial Port Interrupt (INT06). Both the RI and TI flags generate INT06, which exists for compatibility with the 8096BH. Software should enable INT06 and disable both INT08 and INT09 for 8096BH compatibility. For normal operation software should disable INT06 and enable both INT08 and INT09.

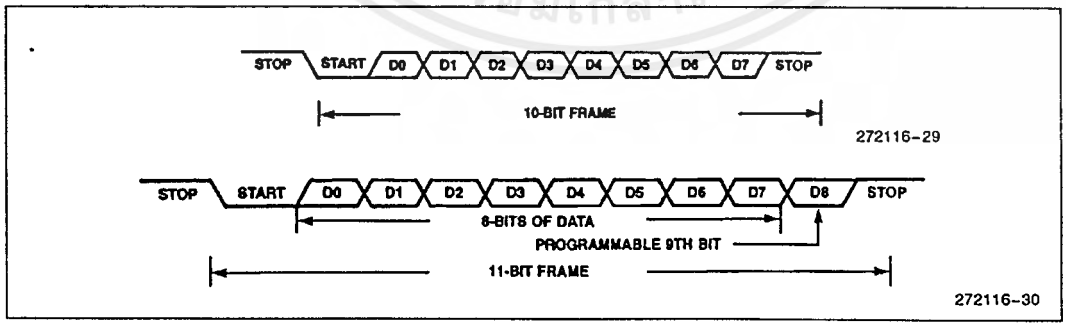


Figure 11-4. Serial Port Frames, Mode 1, 2 and 3

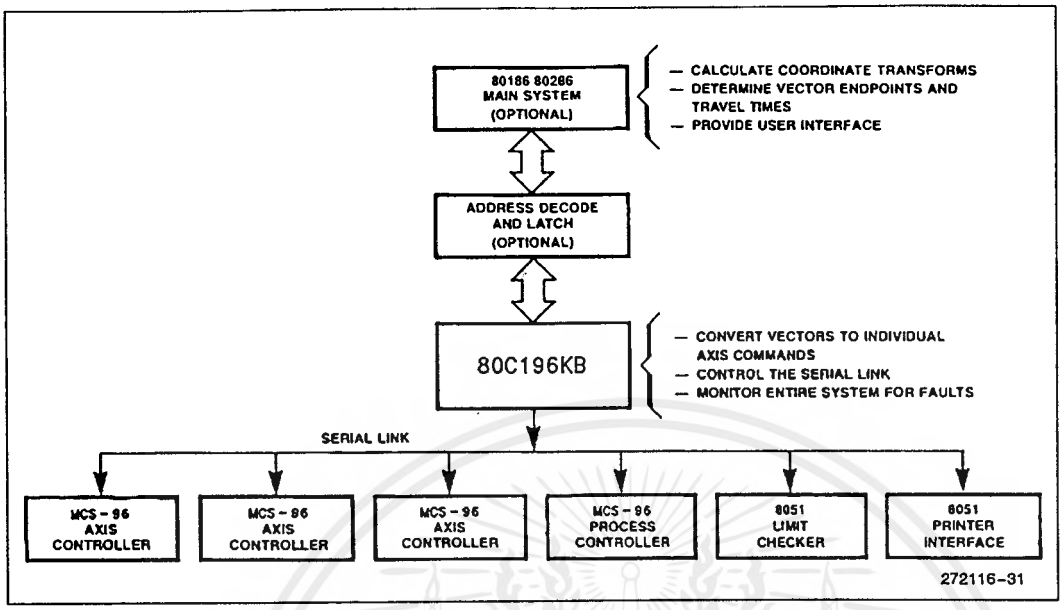


Figure 11-5. Multiprocessor Communication

The Baud Rate Register (BAUD_REG) controls the baud rates for the serial modes. This is a byte wide register that is loaded sequentially with two bytes, and internally stores the value as a word. The least significant byte is loaded to the register followed by the most significant byte. The most significant bit of the baud value determines the clock source for the baud rate generator. Setting this bit will select the XTAL1 pin as the clock, otherwise the T2CLK pin will function as the clock. To determine the baud value use the formulas shown in Figure 11-6. The baud values for common baud rates when using XTAL1 as the clock source are shown in Table 11-1. In most cases a serial link will work with up to 5.0% difference between baud rates.

Common baud rate values, using XTAL1 at 16 MHz, are shown below.

Table 11-1. Common Baud Rate Values

Baud Rate	Baud Register Value	
	Mode 0	Others
9600	8340H	8067H
4800	8682H	80CFH
2400	8D04H	81AOH
1200	9A0AH	8340H
300	E82BH	8D04H

Asynchronous Modes 1, 2 and 3:

$$\text{BAUD\_REG} = \frac{\text{XTAL1}}{\text{Baud Rate} * 16} - 1 \text{ OR } \frac{\text{T2CLK}}{\text{Baud Rate} * 8}$$

Synchronous Mode 0:

$$\text{BAUD\_REG} = \frac{\text{XTAL1}}{\text{Baud Rate} * 2} - 1 \text{ OR } \frac{\text{T2CLK}}{\text{Baud Rate}}$$

B must only equal 0 in modes 1, 2 or 3, when using XTAL1 as the clock source. Do not use B = 0 in mode 0.

Figure 11-6. Baud Rate Formulas

ภาคผนวก II

Data Sheet IGBT model

6MBI20L-060

MBI20L-060 (20A)

Fuji Power Module

IGBT MODULE (L series)

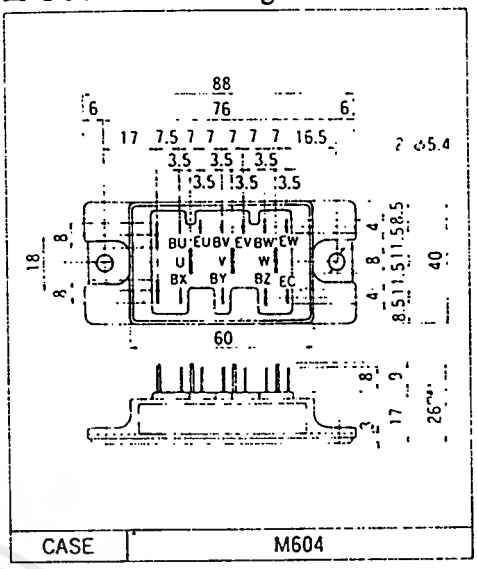
Features

- High Speed Switching
- Low Saturation Voltage
- Voltage Drive

Applications

- Inverter for Motor Drive
- AC and DC Servo Drive Amplifier
- Uninterruptible Power Supply
- Industrial Machines, such as Welding Machines

Outline Drawings



CASE	M604

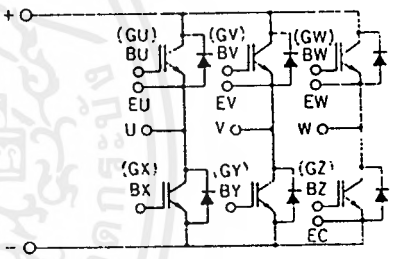
Maximum Ratings and Characteristics

Absolute Maximum Ratings

Items		Symbols	Ratings	Units
Collecter-Emitter Voltage		V_{CES}	600	V
Gate-Emitter Voltage		V_{GES}	±20	V
Collecter Current	Continuous	I_C	20	A
	1ms	$I_{C\ pulse}$	40	
	Continuous	$-I_C$	20	
	1ms	$-I_{C\ pulse}$	40	
Max. Power Dissipation		P_C	80	W
Operating Temperature		T_j	+150	°C
Storage Temperature		T_{stg}	−40 to +125	°C
Net. Weight			150	g
Isolation Voltage	AC. 1min.	V_{isol}	2500	V
Screw Torque		Mounting *1	35	kg·cm

*1 Recommendable Value 25 to 35kg·cm (M5)

Equilavelent Circuit Schematic



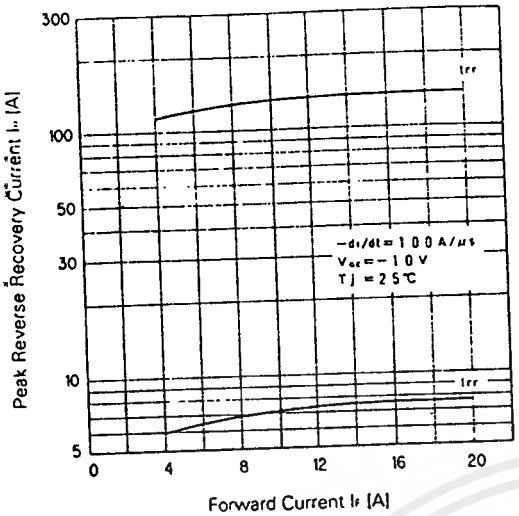
Electrical Characteristics (Tj=25°C)

Items	Symbols	Test Conditions	Min.	Typ.	Max.	Units
Zero Gate Voltage Collecter Current	I_{CES}	V_{GE}=0V V_{CE}=600V T_j=25°C			1.0	mA
Gate-Emitter Leackage Current	I_{GES}	V_{CE}=0V V_{GE}=±20V			100	nA
Gate-Emitter Threshold Voltage	$V_{GE(th)}$	V_{CE}=20V I_C=20mA	3.0	4.5	6.0	V
Collecter-Emitter Saturation Voltage	$V_{CE(sat)}$	V_{GE}=15V I_C=20A		2.7	3.5	V
Input Capacitance	C_{ies}	V_{CE}=0V		1900		pF
Output Capacitance	C_{oes}	V_{CC}=10V		–		
Reverse Transfer Capacitance	C_{res}	f=1MHz		–		
Turn-on Time *2	t_{on}	V_{CC}=300V		0.4	0.8	μs
	t_r	I_C=20A		0.3	0.6	
Turn-off Time *3	t_{off}	V_{GE}=±15V		0.6	1.0	
	t_f	R_G=120Ω		0.2	0.35	
Diode Forward On-Voltage	V_F	I_F=20A V_{GE}=0V			2.5	V
Reverse Recovery Time	t_{rr}	I_F=20A $-di_F/dt$=100A/μs V_{GE}=−10V			300	ns

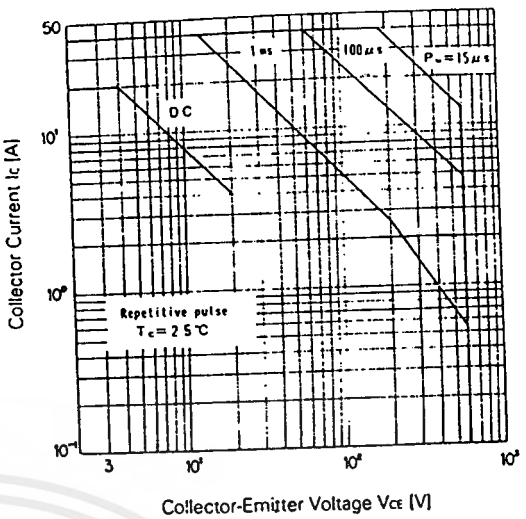
*2 Resistive load *3 Inductive load

Thermal Characteristics

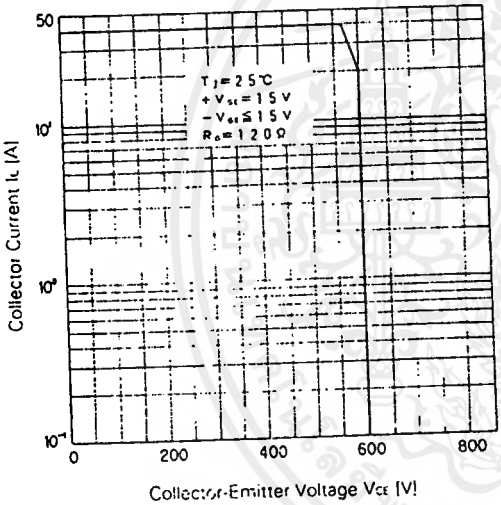
Items	Symbols	Test Conditions	Min.	Typ.	Max.	Units
Thermal Resistance	$R_{th(j-c)}$	IGBT			1.56	°C/W
	$R_{th(j-c)}$	Diode			3.01	
	$R_{th(c-f)}$	With Thermal compound		0.06		



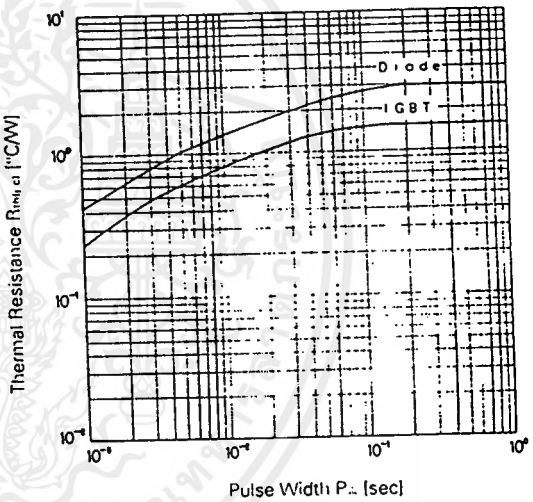
trr, Irr-IF



Safe Operating Area

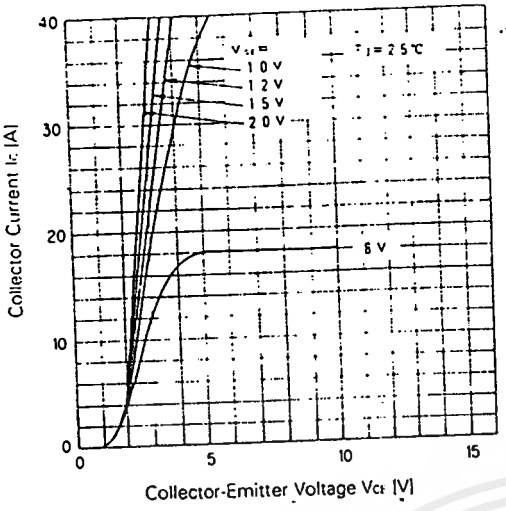


Reverse Biased Safe Operating Area

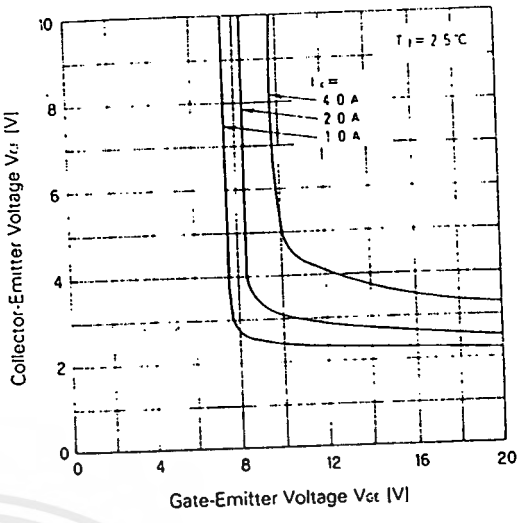


Transient Thermal Resistance

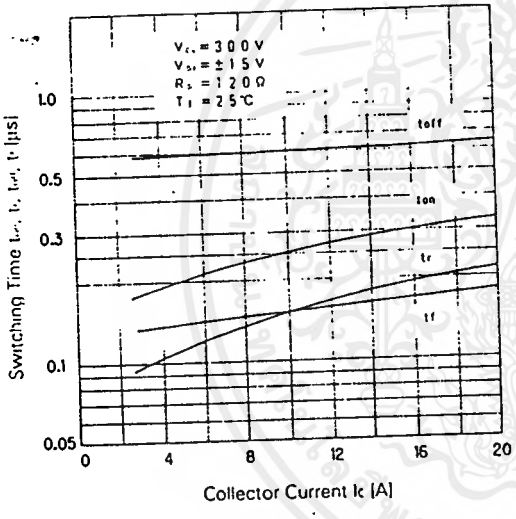
haracteristics



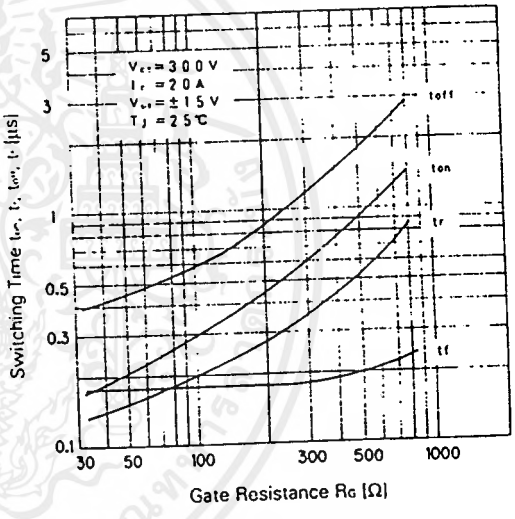
Collector Current vs. Collector-Emitter Voltage



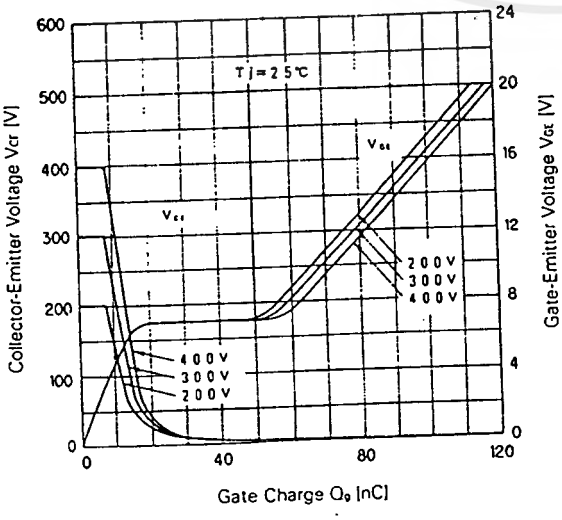
Collector-Emitter Voltage vs. Gate-Emitter Voltage



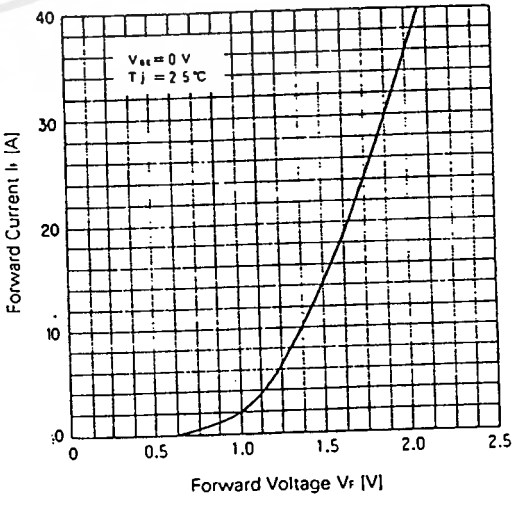
Switching Time



Switching Time-Gate Resistance



Dynamic Input Characteristic



Forward Voltage of Free Wheel Diode

ภาคผนวก III

Data Sheet EXB841

CONTENTS

1. Introduction

The insulated gate bipolar transistor (IGBT) is increasingly being used in small, low-noise, high-performance power supplies, inverters, uninterruptable power supplies (UPS), and motor speed controls.

Fuji's Hybrid IC driver for IGBTs was developed to take full advantage of the IGBT.

2. Features

- Various series

  Standard series: For up to 10 kHz operation

  High-speed series: For up to 40 kHz operation

  These series cover the full range of IGBT products.
- Built-in photocoupler for high isolation voltage: 2500 V AC for one minute
- Single supply operation
- Built-in overcurrent protection circuit
- Overcurrent detection output
- SIL package for high-density mounting

3. Applications

- General-purpose inverter and motor control
- Servo control
- Uninterruptable power supplies (UPS)
- Welding machines

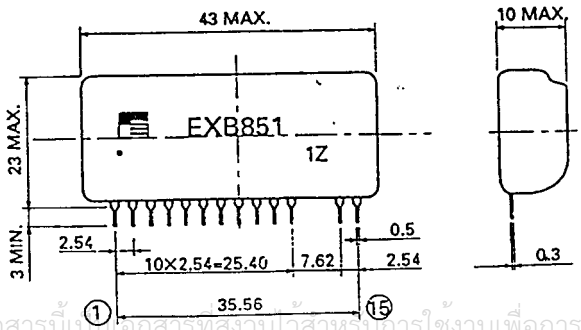
4. Comprehensive Chart

IGBT	600 V IGBT drive		1200 V IGBT drive	
	Up to 150A	Up to 400A	Up to 75A	Up to 300A
Standard type	EXB850	EXB851	EXB850	EXB851
High-speed type	EXB840	EXB841	EXB840	EXB841

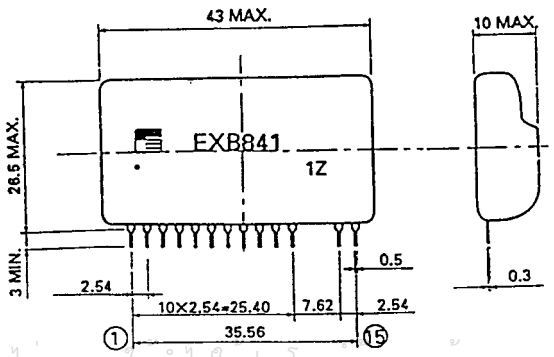
Notes: 1. Standard type: Signal delay in drive circuit; Up to 4μs (max.)
2. High-speed type: Signal delay in drive circuit; Up to 1.5μs (max.)

5. Dimensions, mm

EXB850
EXB840

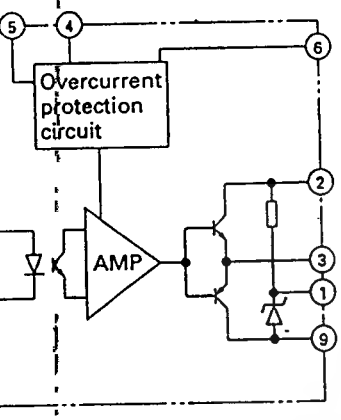


- EXB851
- EXB841

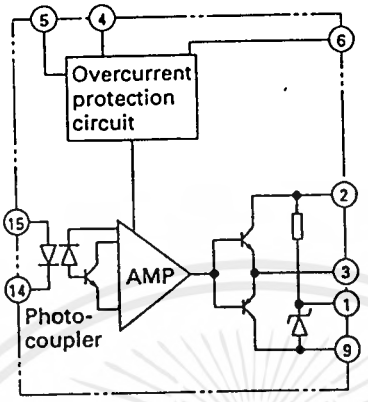


Functional Block Diagrams

850, 851



EXB840, 841



Notation common to all EXB series

Pin number	Description
①	Connected to smoothing capacitor for reverse bias power supply
②	Power supply (+20 V)
③	Drive output
④	For connecting an external capacitor to protect against malfunction of the overcurrent protection circuit. (The capacitor is not needed in most cases.)
⑤	Overcurrent detection output
⑥	Collector voltage monitoring
⑦ ⑧	Not connected
⑨	Power supply (0 V)
⑩ ⑪	Not connected
⑭	Drive signal input (–)
⑮	Drive signal input (+)

Ratings and Characteristics

Absolute maximum ratings

(Ta = 25°C)

tem	Symbol	Condition	Rating		Unit
			EXB850, EXB840 (Medium capacity)	EXB851, EXB841 (Large capacity)	
Supply voltage	V cc		25		V
Photocoupler input current	I in		10		mA
Forward bias output current	I g1	PW = 2 µs, duty at 0.05 or less	1.5	4.0	A
Reverse bias output current	I g2	PW = 2 µs, duty at 0.05 or less	1.5	4.0	A
Input/Output isolation voltage	V ISO	AC 50/60 Hz, 1 minute	2500		V
Operating surface temperature	T c		–10 to +85		°C
Storage temperature	T stg		–25 to +125		°C

Recommended operating conditions

Item	Symbol	Recommended operating conditions				Unit
		Standard type		High-speed type		
		EXB850	EXB851	EXB840	EXB841	
Supply voltage	V cc	20 ±1				V
Photocoupler input current	I in	5		10		mA

■ Electrical characteristics (Ta = 25°C)

Item	Symbol	Condition	Rating						Unit
			EXB840, EXB841 (High speed)			EXB850, EXB851 (Midium speed)			
			Min.	Typ.	Max.	Min.	Typ.	Max.	
Turn-on time 1	t on	V cc = 20 V, IF = 5 mA			1.5			2.0	µsec
Turn-on time 2	t off	V cc = 20 V, IF = 5 mA			1.5			4.0	µsec
Overcurrent protection voltage	t ocp	V cc = 20 V, IF = 5 mA		7.5			7.5		V
Overcurrent protection delay	t ocp	V cc = 20 V, IF = 5 mA			10			10	µsec
Alarm delay	t ALM	V cc = 20 V, IF = 5 mA			1			1	µsec
Reverse bias power supply voltage	V RB	V cc = 20 V		5			5		V

Note: EXB851 and EXB850 (medium speed) require IF overdrive shown in the application circuit.

8. Application Circuits

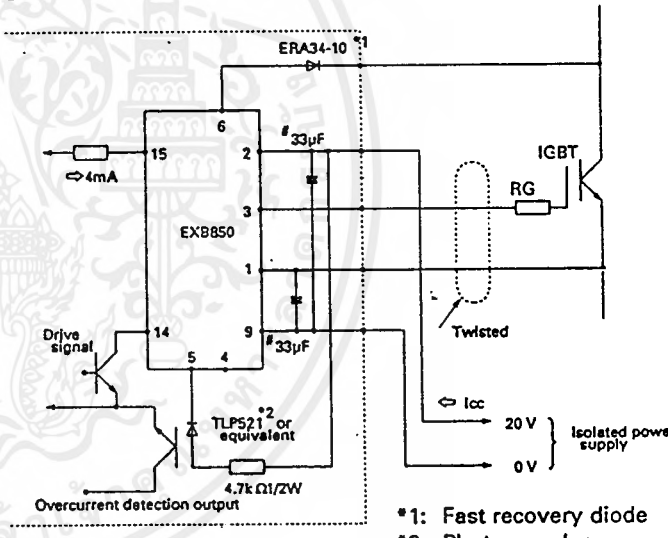
1 EXB850 Application Circuits

EXB850 is a hybrid IC capable of driving up to 150A for a 600V IGBT and up to 75A for a 1200 V IGBT. Since the signal delay in the drive circuit is 4µs or less, the hybrid IC is suitable for switching at up to about 10 kHz.

Note the following when using the hybrid IC:

- The IGBT gate-emitter drive loop wiring must be shorter than one meter.
- The IGBT gate-emitter drive wiring should be twisted.
- If a large voltage spike is generated at the collector of the IGBT, increase the IGBT's gate series resistor (RG).
- The 33 µF (#) capacitor absorbs changes in the supply voltage caused by the power supply wiring impedance. It is not a power supply filter capacitor.

Control circuit PC board



*1: Fast recovery diode
*2: Photo coupler

Recommended gate resistance and current consumption

IGBT rating	600 V	10 A	15 A	30 A	50 A	75 A	100 A	150 A
	1200 V	—	8 A	15 A	25 A	—	50 A	75 A
RG		250 Ω	150 Ω	82 Ω	50 Ω	33 Ω	25 Ω	15 Ω
I cc	5 kHz	24 mA			24 mA			26 mA
	10 kHz				25 mA			29 mA
	15 kHz	25 mA			27 mA			32 mA

XB851 Application Circuits

XB851 is a hybrid IC capable of driving up to 00A for 600 V IGBT and up to 300A for a 1200 V GBT. Since the signal delay in the drive circuit is µs or less, the hybrid IC is suitable for switching t up to about 10 kHz.

Note the following when using the hybrid IC:

The IGBT's gate-emitter drive loop wiring must be shorter than one meter.

The IGBT's gate-emitter drive wiring should be wisted.

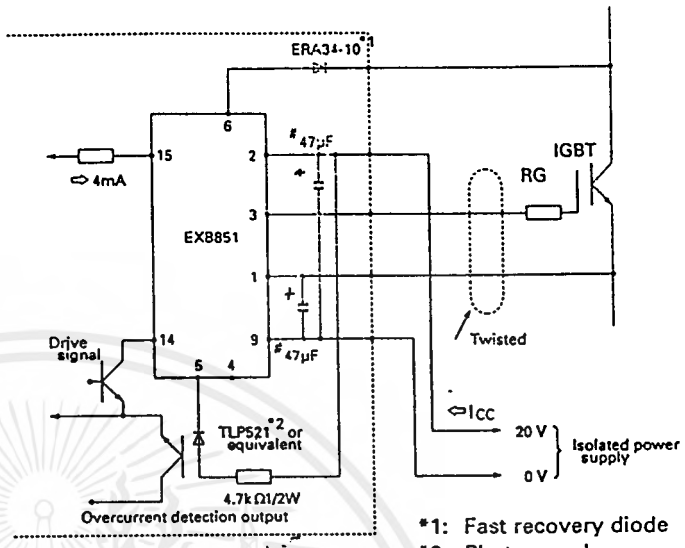
f a large voltage spike is generated at the collector of the IGBT, increase the IGBT's gate series esistor(RG).

The 47 µF (#) capacitor absorbs changes in the supply voltage caused by the power supply wiring impedance. It is not a power supply filter capacitor.

ommended gate resistance and current consumption

BT ing	600 V	200 A	300 A	400 A	—
	1200 V	200 A	150 A	200 A	300 A
		12 Ω	8.2 Ω	5 Ω	3.3 Ω
c	5 kHz	27 mA	29 mA	30 mA	34 mA
	10 kHz	31 mA	34 mA	37 mA	44 mA
	15 kHz	34 mA	39 mA	44 mA	54 mA

Control circuit PC board



*1: Fast recovery diode
*2: Photo coupler

EXB840 Application Circuits

EXB840 is a hybrid IC capable of driving up to 150A for 600 V IGBT and up to 75A for a 1200 V IGBT. Since the signal delay in the drive circuit is 1µs or less, the hybrid IC is suitable for switching at up to about 40 kHz.

Note the following when using the hybrid IC:

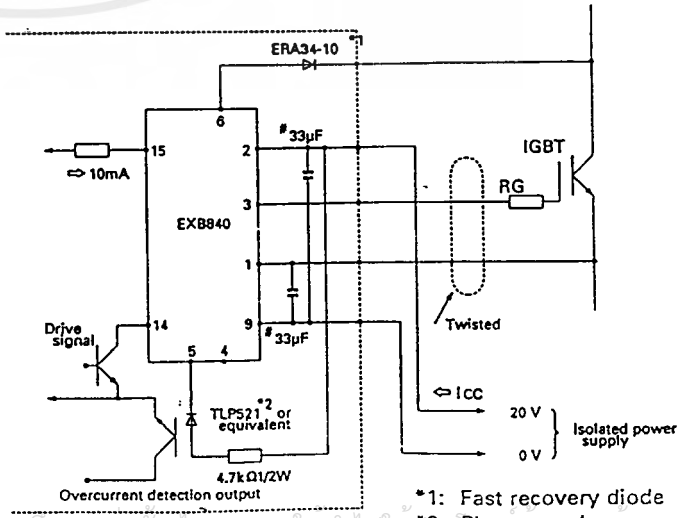
The IGBT's gate-emitter drive loop wiring must be shorter than one meter.

The IGBT's gate-emitter drive wiring should be twisted.

If a large voltage spike is generated at the collector of the IGBT, increase the IGBT's gate series resistor(RG).

The 33 µF (#) capacitor absorbs changes in the supply voltage caused by the power supply wiring impedance. It is not a power supply filter capacitor.

Control circuit PC board



*1: Fast recovery diode
*2: Photo coupler

Recommended gate resistance and current consumption

IGBT rating	600 V	10 A	15 A	30 A	50 A	75 A	100 A	150 A
	1200 V	—	8 A	15 A	25 A	—	50 A	75 A
RG		250 Ω	150 Ω	82 Ω	50 Ω	33 Ω	25 Ω	15 Ω
Icc	5 kHz	17 mA			17 mA			19 mA
	10 kHz				18 mA			22 mA
	15 kHz	18 mA			20 mA			25 mA

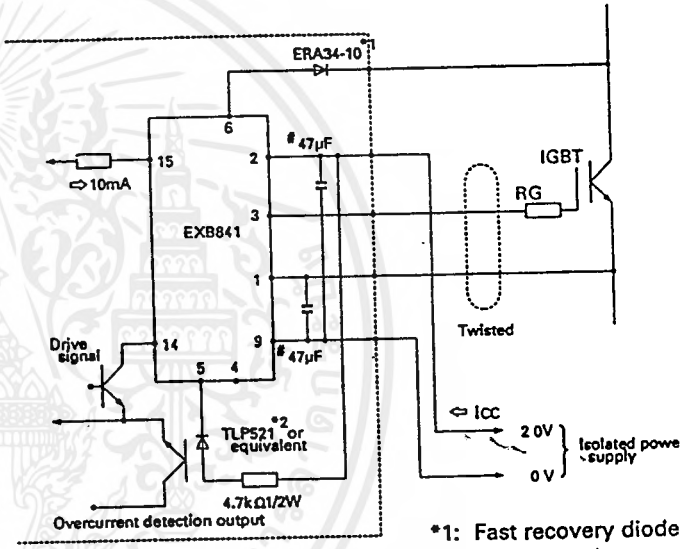
4 EXB841 Application Circuits

EXB841 is a hybrid IC capable of driving up to 400A for 600 V IGBT and up to 300A for a 1200 V IGBT. Since the signal delay in the drive circuit is 1μs or less, the hybrid IC is suitable for switching at up to about 40 kHz.

Note the following when using the hybrid IC:

- The IGBT's gate-emitter drive loop wiring must be shorter than one meter.
- The IGBT's gate-emitter drive wiring should be twisted.
- If a large voltage spike is generated at the collector of the IGBT, increase the IGBT's gate series resistor(RG).
- The 47 μF (#) capacitor absorbs changes in the supply voltage caused by the power supply wiring impedance. It is not a power supply filter capacitor.

Control circuit PC board



*1: Fast recovery diode
*2: Photo coupler

Recommended gate resistance and current consumption

IGBT rating	600 V	200 A	300 A	400 A	—
	1200 V	200 A	150 A	200 A	300 A
RG		12 Ω	8.2 Ω	5 Ω	3.3 Ω
I cc	5 kHz	20 mA	22 mA	23 mA	27 mA
	10 kHz	24 mA	27 mA	30 mA	37 mA
	15 kHz	27 mA	32 mA	374 mA	47 mA

1. Operating Waveforms

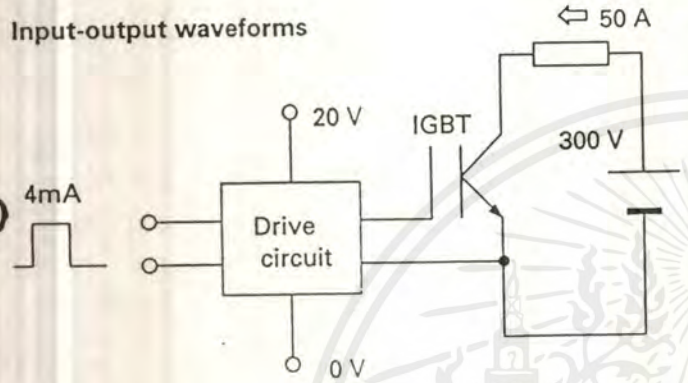
EXB850

Operating conditions

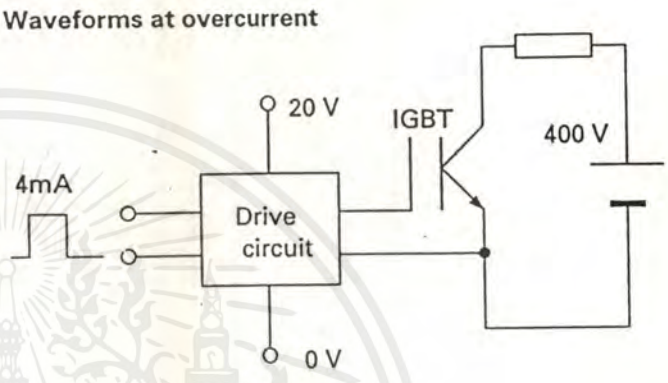
Vcc = 20 V, Iin = 4 mA, IGBT module: 2MB150-060

Test circuits

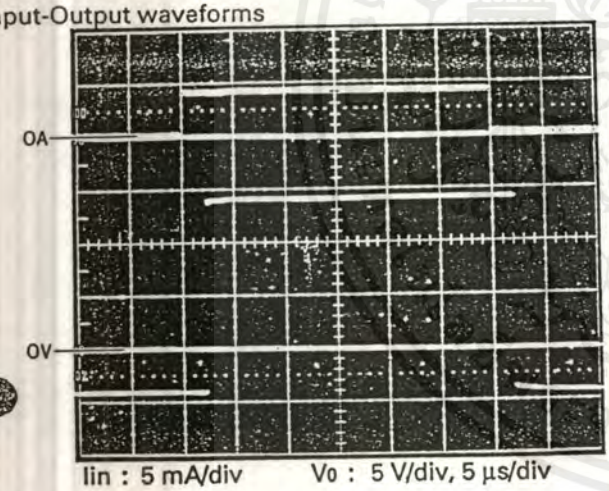
Input-output waveforms



Waveforms at overcurrent

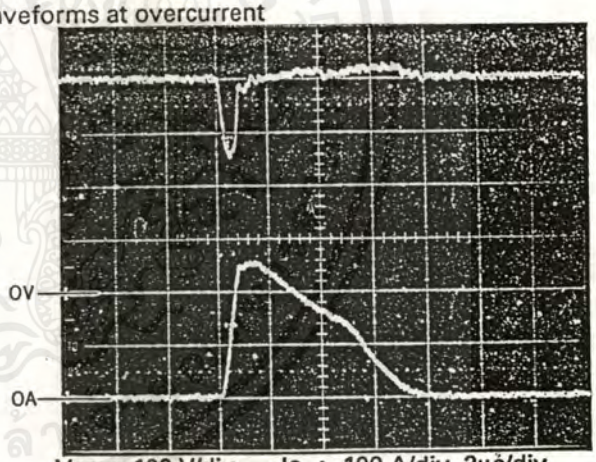


Input-Output waveforms

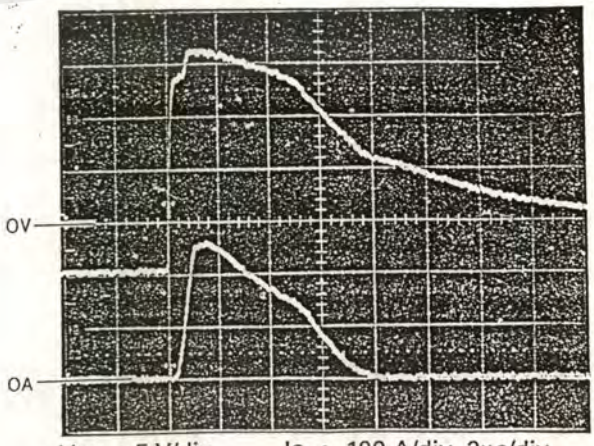


Iin : 5 mA/div V0 : 5 V/div, 5 µs/div

Waveforms at overcurrent



VCE : 100 V/div Ic : 100 A/div, 2µs/div



VGE : 5 V/div Ic : 100 A/div, 2µs/div

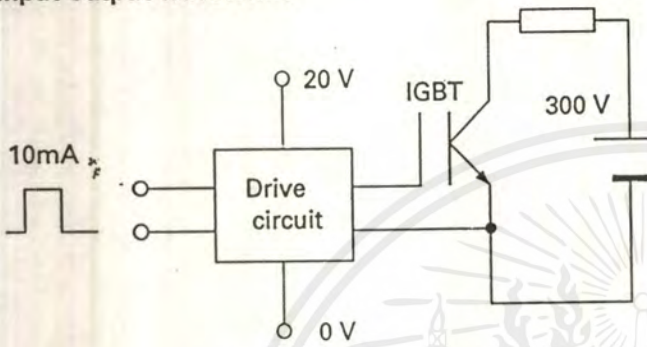
2 EXB841

Operating conditions

Vcc = 20 V, Iin = 10 mA, IGBT module: 2MB150-060

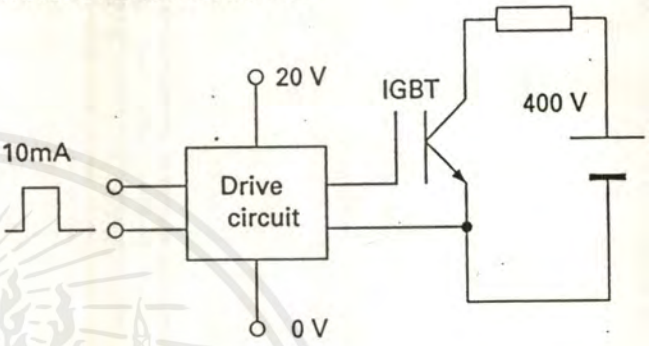
Test circuit

Input-output waveforms

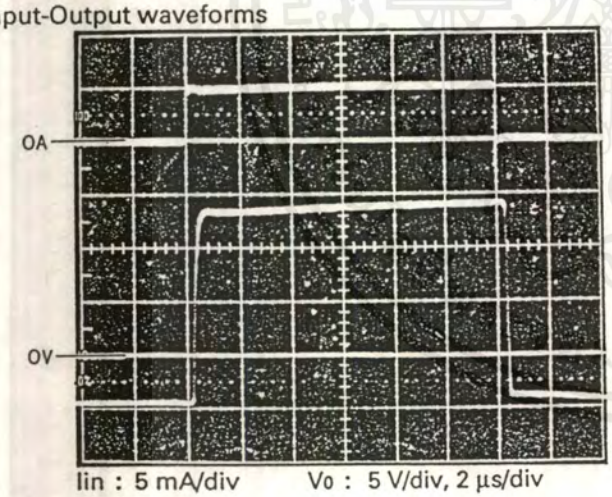


⇦ 50 A

Waveforms at overcurrent



Input-Output waveforms



Iin : 5 mA/div Vo : 5 V/div, 2 µs/div

Waveforms at overcurrent are the same as for EXB851.

กิตติกรรมประกาศ

การทำปริญญานิพนธ์นี้ ถ้ามีเพียงตัวของข้าพเจ้าทำเพียงลำพังคงจะไม่สามารถที่จะพบคำว่าสำเร็จได้อย่างแน่นอน ผู้ที่เป็นแรง และพลังในการสร้างชีวิตอันด้อยค่าของข้าพเจ้า เป็นเพียงสิ่งเดียวที่ทำให้ข้าพเจ้าสู้กับชีวิตที่เต็มไปด้วยขวากหนามและสังคมที่มีแต่ความโหดร้าย และเป็นที่พึ่งเพียงที่เดียวที่ของข้าพเจ้า คือคุณพ่อทองเส้งและคุณแม่เซ็งเซี้ยม พิญญพงษ์

ผู้ที่สร้างแนวทาง คอยให้ความช่วยเหลือในทุกด้าน เป็นที่ปรึกษา และคอยสั่งสอน แนะนำปลูกผังแนวคิดที่ดี ทำให้ข้าพเจ้ารู้ว่าในโลกนี้คนดียังไม่สิ้น " อาจารย์ สุธี ผู้เจริญชนะชัย "

ผู้ให้ความช่วยเหลือ ให้คำปรึกษา ช่วยแก้ปัญหาต่าง ๆ ให้ลุล่วงไปได้ด้วยดี และเอาใจใส่ตลอดการทำปริญญานิพนธ์นี้ " พี่พิสิทธิ วิสุทธิเมธีกร "

ขอบพระคุณอาจารย์ที่ปรึกษา " ดร. วิจิตร กิณเรศ และ อาจารย์ ดุสิต สุขสวัสดิ์ " สำหรับคำปรึกษาและความช่วยเหลือต่าง ๆ

" NECTEC (IEL LAB) " ผู้เกื้อหนุนอุปกรณ์ในการทำปริญญานิพนธ์นี้ ให้สำเร็จได้ด้วยดี

" พี่กนกเวช พี่พงพิศ พี่ถนัด พี่จิโรจน์ พี่ประกาศิต พี่จก พี่ประสิทธ์ พี่เจ็ด " ผู้ให้กำลังใจและช่วยเหลือในทุก ๆ ด้าน มาโดยตลอดและ เพิ่มเติมความรู้อันน้อยนิดให้มากขึ้น

คำแนะนำพร้อมความช่วยเหลือที่มากล้น " พี่แข็ง พี่แหน่ง พี่วิโรจน์ พี่เพิส์ท พี่สำฤทธิ์ และเพื่อน ๆ ที่ CT – LAB "

ความเป็นเพื่อน และแรงใจ " เพื่อน ๆ ทุก ๆ คน "

สุดท้ายก็ขอบคุณ " ชาวนา และคนในประเทศทุกท่าน " สำหรบภาษีของท่านที่ส่งผมเรียนจนจบการศึกษา

และคงไม่มีคำใดจะกล่าวได้มากกว่าคำว่า " ขอบคุณทุก ๆ คนครับ ผมจะพยายามเป็นคนดีของสังคม และจะใช้ความรู้ที่ได้มาให้มีประโยชน์สูงสุดเพื่อประเทศชาติของเรา "

ธีรเดช พิญญพงษ์

หนังสืออ้างอิง

[1] พิสิทธิ วิสุทธิเมธีกร." การพัฒนาระบบควบคุมมอเตอร์เหนี่ยวนำแบบปรับตามสนามแม่เหล็ก". บัณฑิตวิทยาลัยสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ , 2540

[2] โสภณ สมัยรัฐ."ระบบควบคุมมอเตอร์เหนี่ยวนำแบบเวคเตอร์โดยไมโครคอนโทรลเลอร์". บัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย,2538

[3] ANDRZEJ M.TRZYNADLOWSKI.THE FIELD ORIENTATION PRINCIPLE IN CONTROL OF INDUCTION MOTOR.,UNIVERSITY OF NEVADA, RENO, KLUWER ACADMIC PUBLISHERS,1994

[4] PETER VAS. VECTOR CONTROL OF AC MACHINES.OXFORD SCIENCE PUBLICATIONS ,1990

[5] Power Electronics Research Laboratory, Device for AC Drive in Power Electronics Block Library (Version 1.1) , Department of Electrical Engineering , College of Engineering ,Chulalongkorn Unversity

[6] W.LEONHARD.CONTROL OF ELECTRIC DRIVES. BERLIN,HEIDELBERG,GERMAY.1985.