## รายการอ้างอิง


[1] Chen,J.H. Cox, R. Lin, Y. Jayant, N. and Melchner, M. A low-delay CELP coder for the CCITT 16 kbps speech coding standard. Special Issue on Speech and Image Coding. IEEE Trans. Selected Areas Commun 10 ( June 1992 ) : 830-849.

[2] ITU-T. Recommendation G.728, "Coding of speech at 16 kbps using low-delay code-excited linear prediction ( LD-CELP ) . Geneva. 1992

[3] Chen,J. and Gersho, A. Adaptive postfiltering for quality enhancement of coded speech. IEEE Transaction on Speech and Audio Processing 3 ( Jan 1995) : 59-71.

[4] Chen,J. and Cox, R. A fixed-point LD-CELP algorithm and its real-time implementation. Proc.ICASSP-91 Toronto Ont. Canada. ( May 1991) : 21-24 .

[5] Texas Instruments Incorporated. TMS320C5x User's Guide. Digital Signal Processing Products Texas Instruments Incorporated. 1990

[6] Texas Instruments Incorporated. TMS320C5x DSP Starter Kit User's Guide. Microprocessor Development Systems. Texas Instruments Incorporated. 1994

[7] Spanias,A.S. Speech coding: A tutorial review. Proceedings of IEEE. 82. No.10.(October 1994 ) : 1541-1582.

[8] Texas Instruments Incorporated. TMS320C5x C Source Debugger User'sGuide. Microprocessor Development Systems. Texas Instruments Incorporated. 1991

[9] Johansen,F.T. and Cox,R. Test Verification of LD-CELP:an Objective Measurement Approach for a Non-bit exact Standard. Proceedings of IEEE (1992)

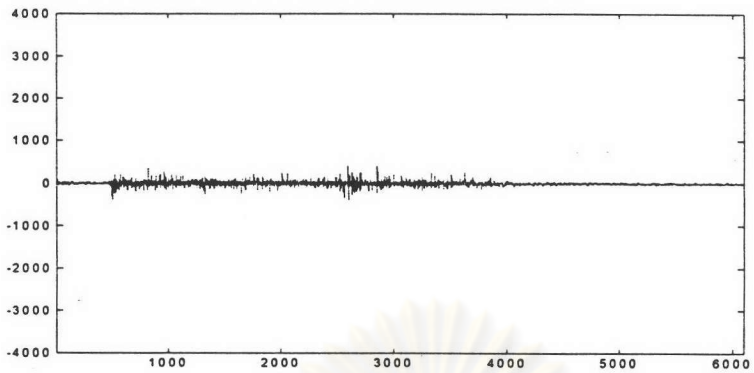
# ภาคผนวก

# ภาคผนวก ก

## ผลการทดลองเพิ่มเติม

รูปที่แสดงจะเรียงลำดับจากสัญญาณเสียงต้นฉบับ สัญญาณเสียงที่ผ่านการเข้ารหัสบน MATLAB สัญญาณเสียงที่ผ่านการเข้ารหัสและผ่านโพสต์ฟิลเตอร์บน MATLAB และสัญญาณเสียงที่ผ่านการเข้ารหัสบนตัวจำลองโปรแกรมภาษาแอสเซมบลี แต่ละรูปประกอบด้วยรูปสัญญาณเสียง รูปผลต่างของสัญญาณเสียงจากสัญญาณเสียงต้นฉบับ สเปกตรัม และค่า SNR เมื่อเทียบกับสัญญาณเสียงต้นฉบับ สัญญาณเสียงที่มีรูปแสดงนี้ประกอบด้วยคำว่า "เดิน" "วันอาทิตย์" "หนึ่ง" "สอง" "หก" "เจ็ด" และ "แปด" ตามลำดับ

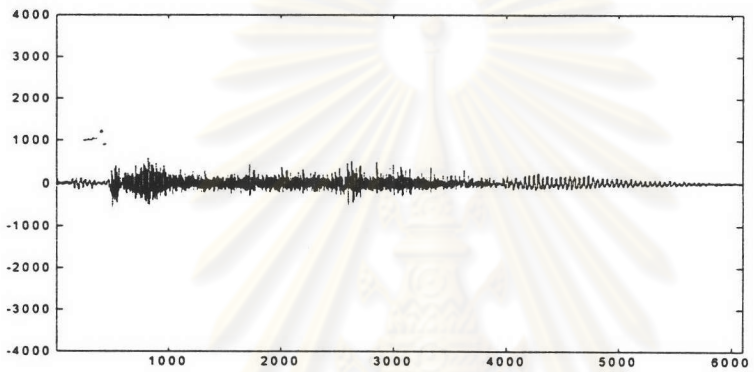
เสียงทุกตัวอย่างที่ใช้ในการทดลองนี้มีอัตราการสุ่มข้อมูลเท่ากับ 8.0 กิโลเฮิรตซ์หรือ 8,000 ตัวอย่างสุ่มใน 1 วินาที เสียง 1 ตัวอย่างสุ่มเท่ากับเวลา 125 ไมโครวินาที เสียง 1,000 ตัวอย่างสุ่มเท่ากับ 0.125 วินาที




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

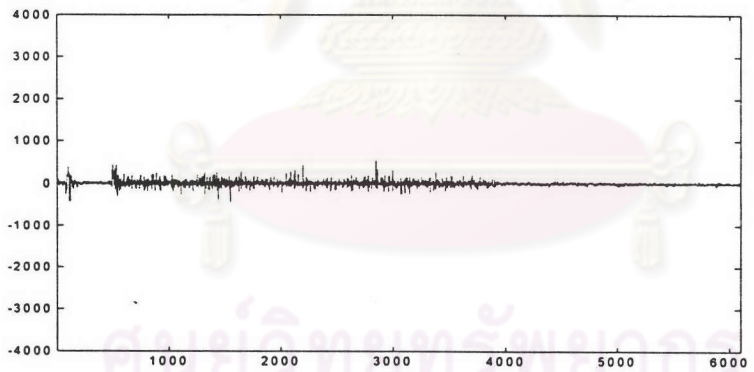
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

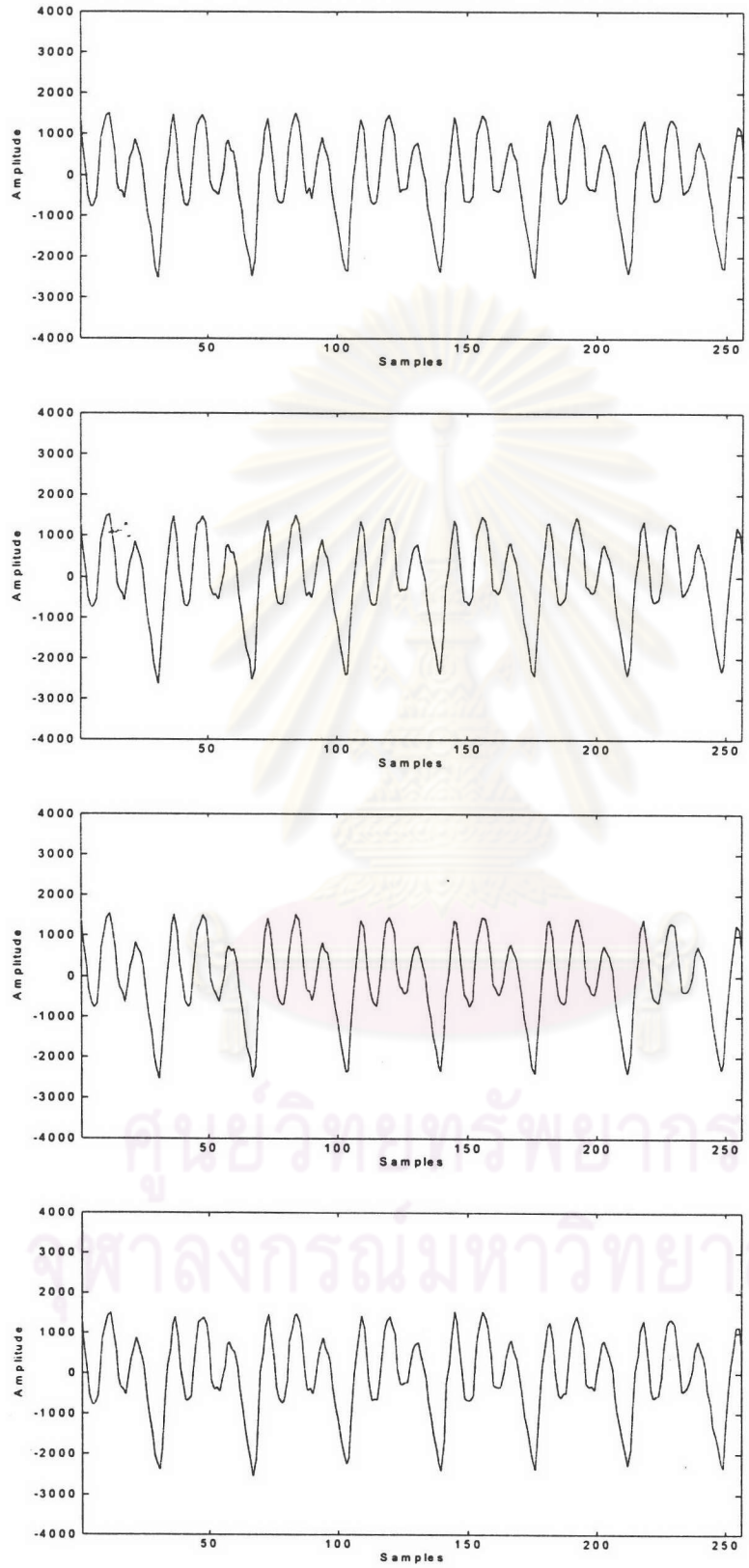
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR = 24.1 เดซิเบล

(ข) SNR =17.5 เดซิเบล

(ค) SNR =22.9 เดซิเบล

รูป ก.2 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "เดิน"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

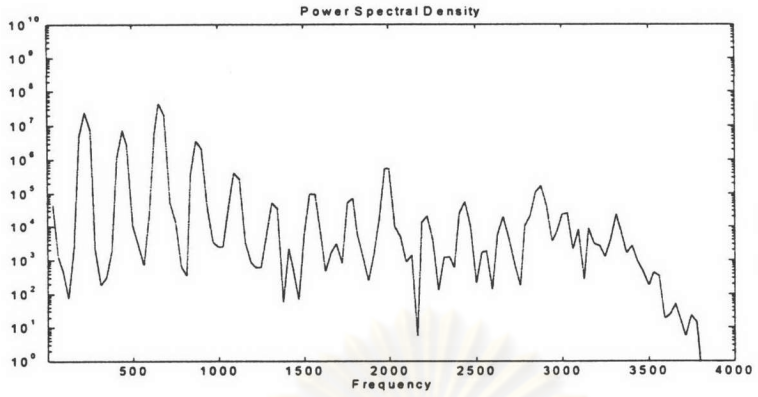
แกนนอน = ตัวอย่างสุ่ม

(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

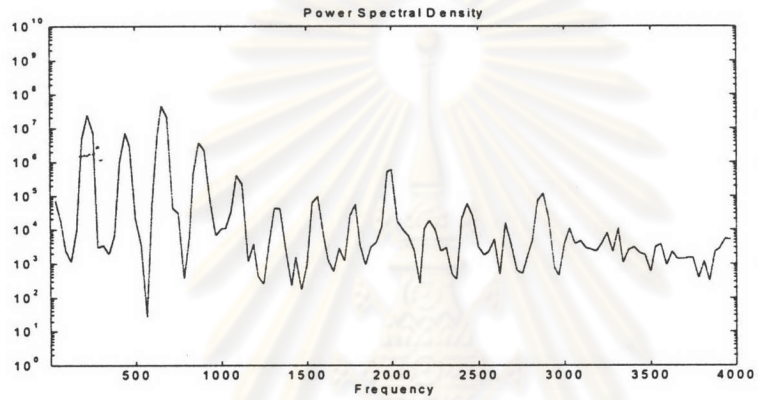
แกนนอน = ตัวอย่างสุ่ม

รูป ก.3 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.1



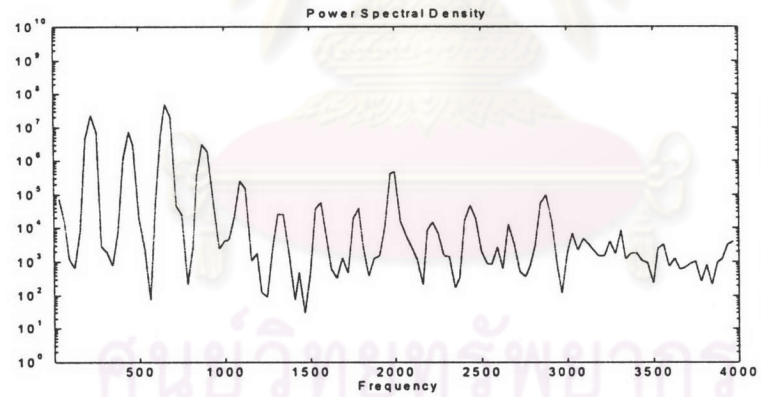

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



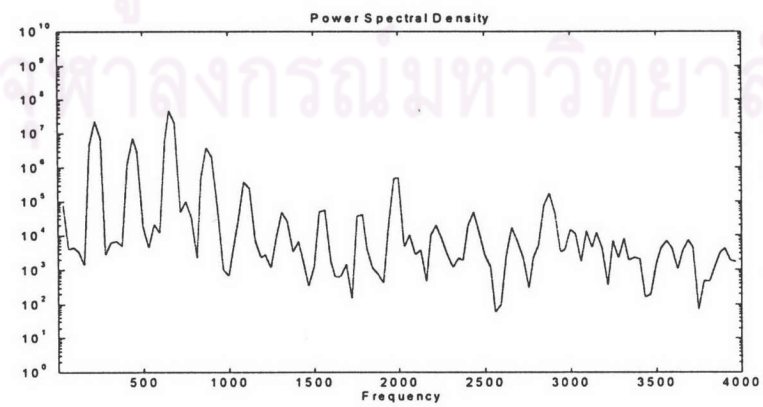

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

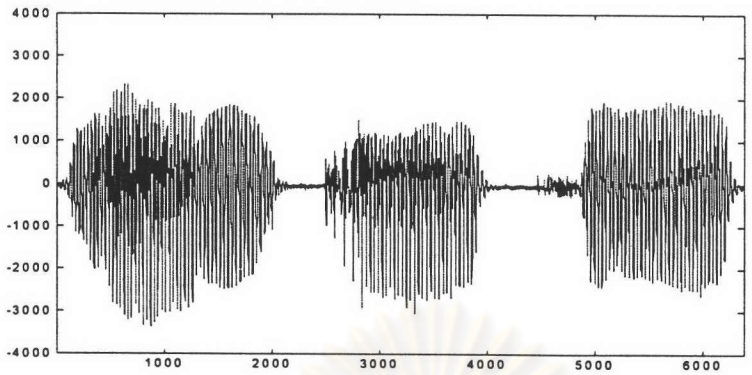
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

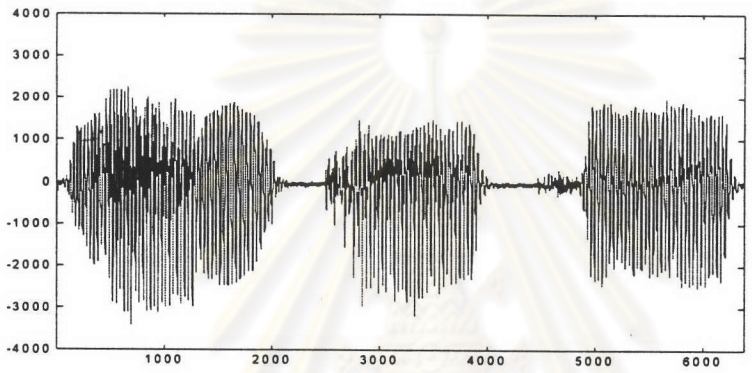
รูป ก.4 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.1




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

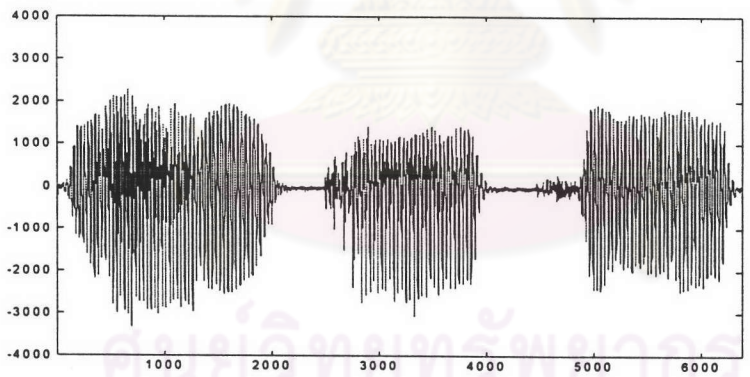
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

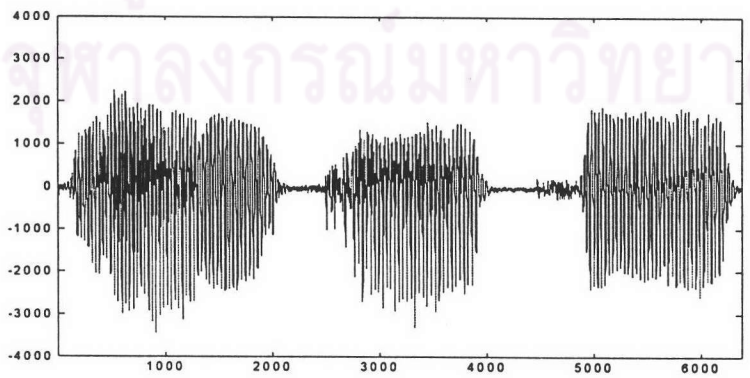
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

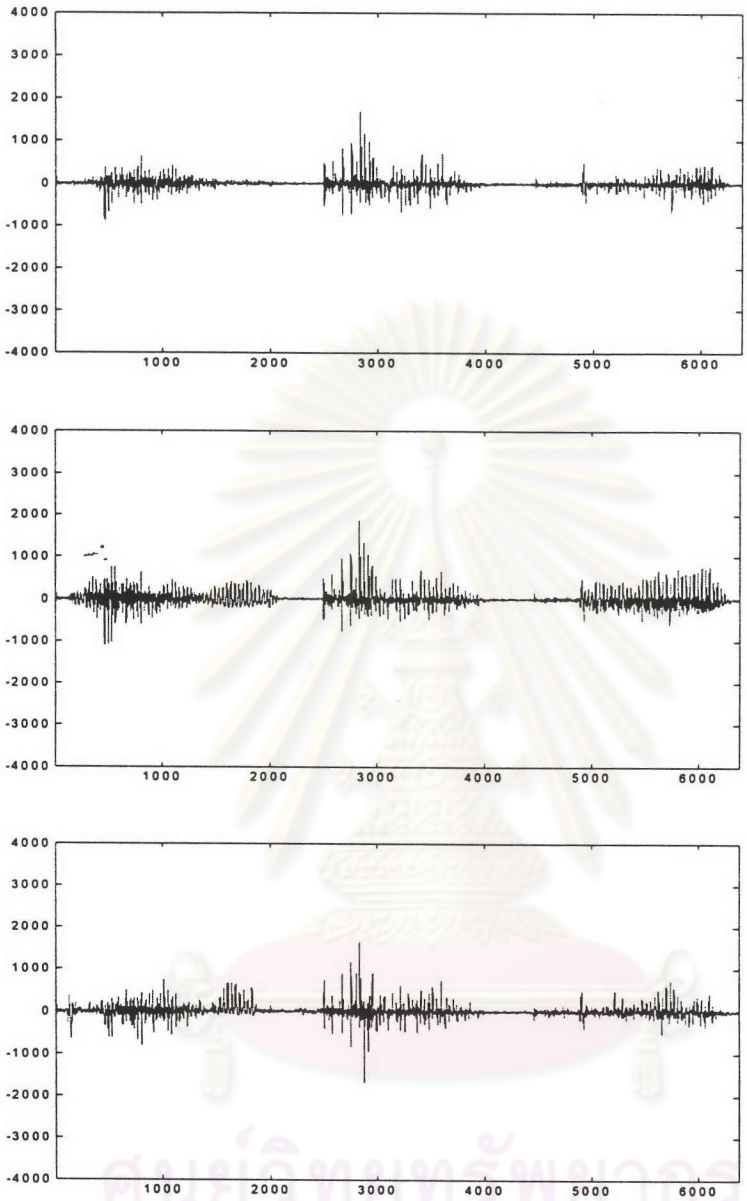



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

รูป ก.5 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "วันอาทิตย์"




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

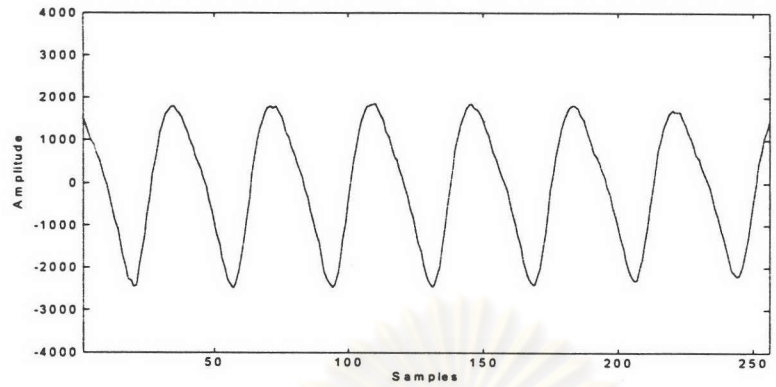
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR = 18.6 เดซิเบล

(ข) SNR =15.0 เดซิเบล

(ค) SNR =17.2 เดซิเบล

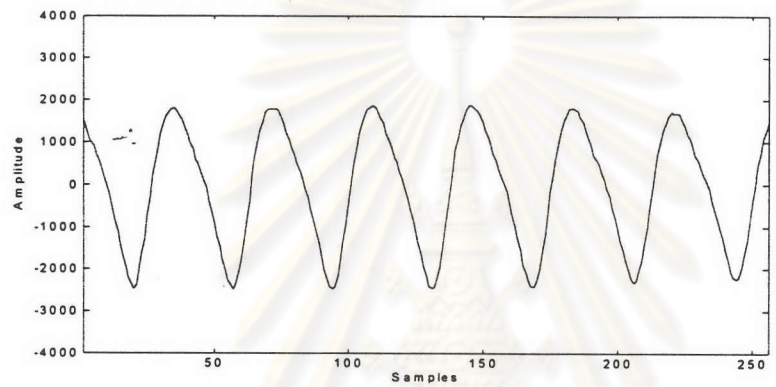
รูป ก.6 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "วันอาทิตย์"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

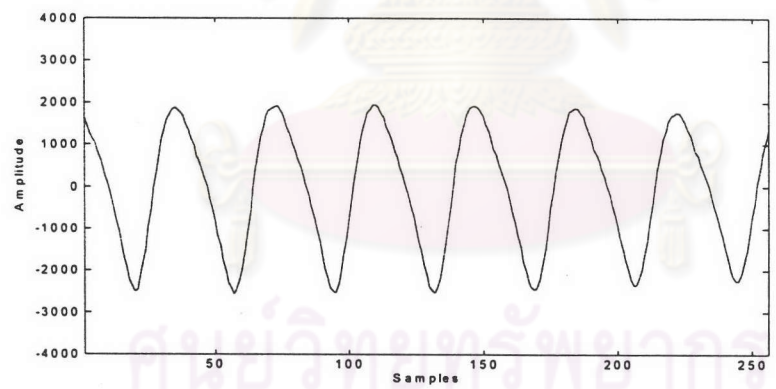
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

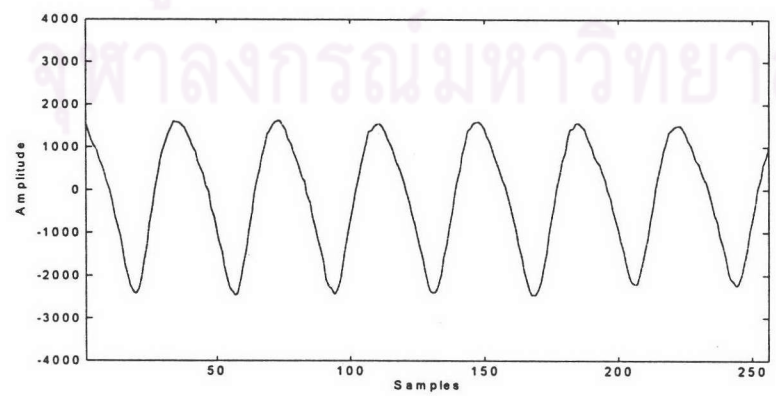
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

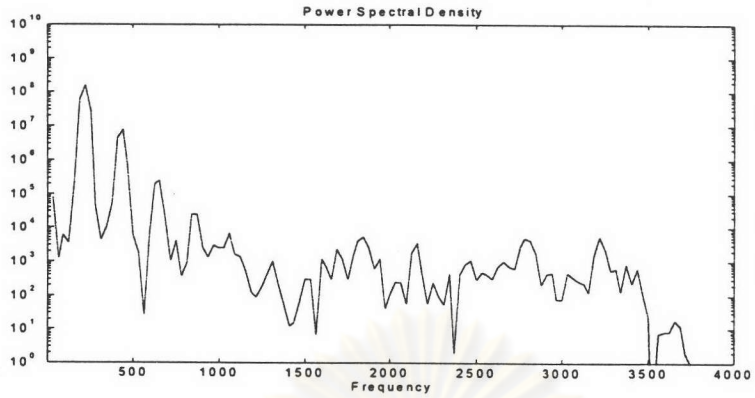



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

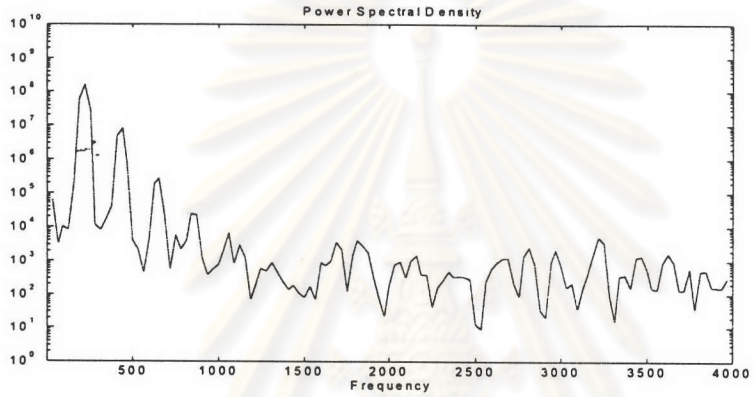
แกนนอน = ตัวอย่างสุ่ม

รูป ก.7 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1501 ถึง 1756 ของเสียงจากรูป ก.5



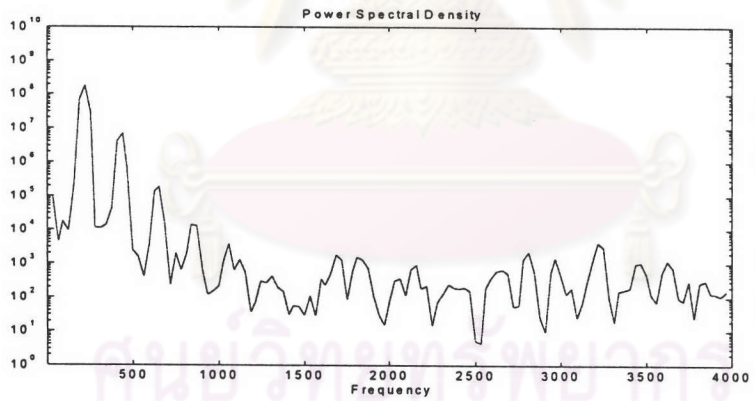

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



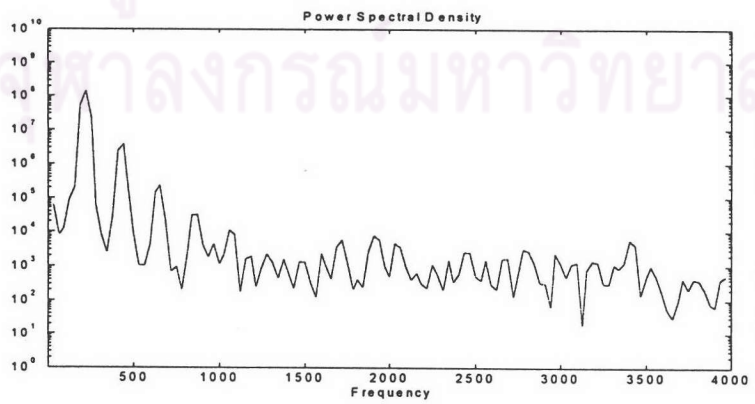

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

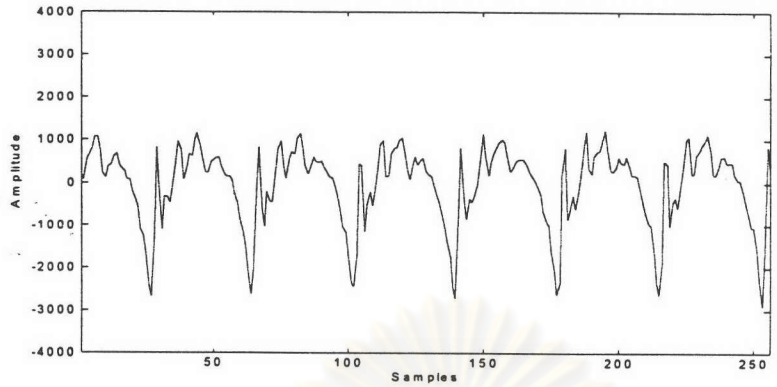
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

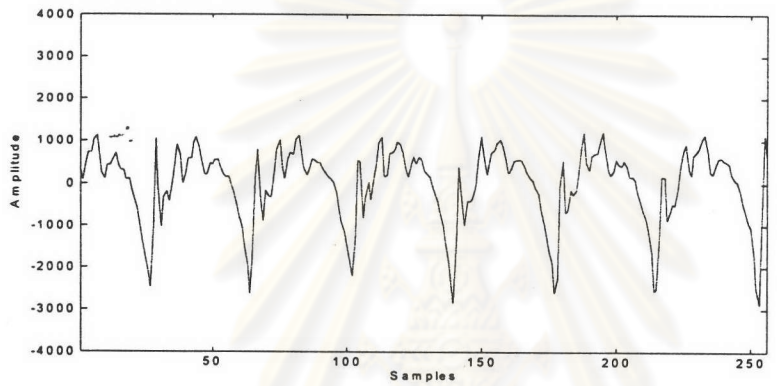
รูป ก.8 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1501 ถึง 1756 ของเสียงจากรูป ก.5




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

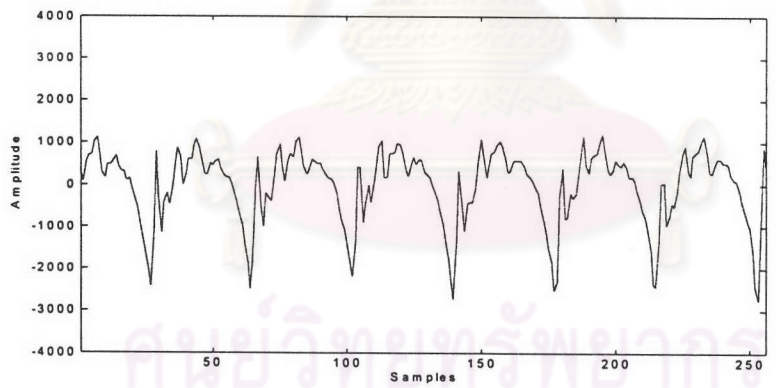
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

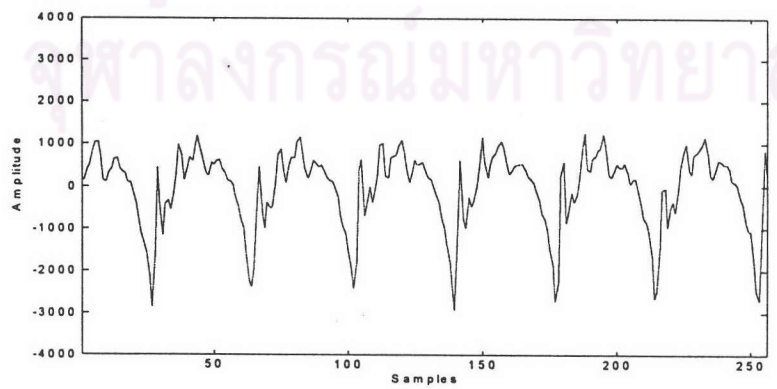
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

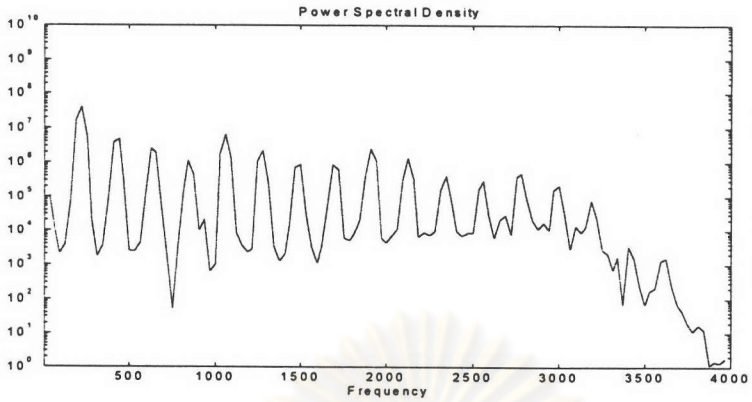



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

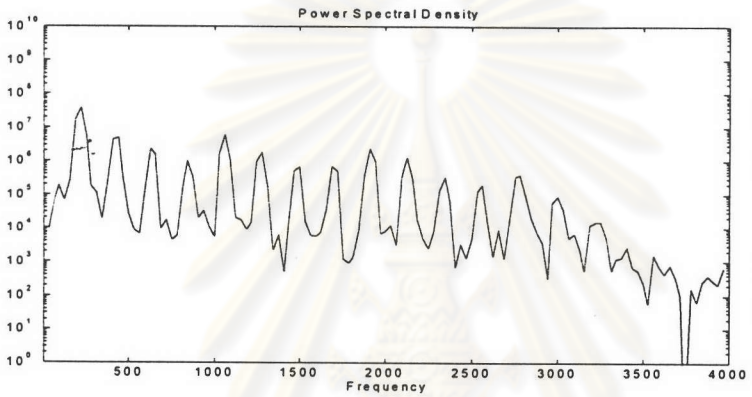
แกนนอน = ตัวอย่างสุ่ม

รูป ก.9 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 3001 ถึง 3256 ของเสียงจากรูป ก.5



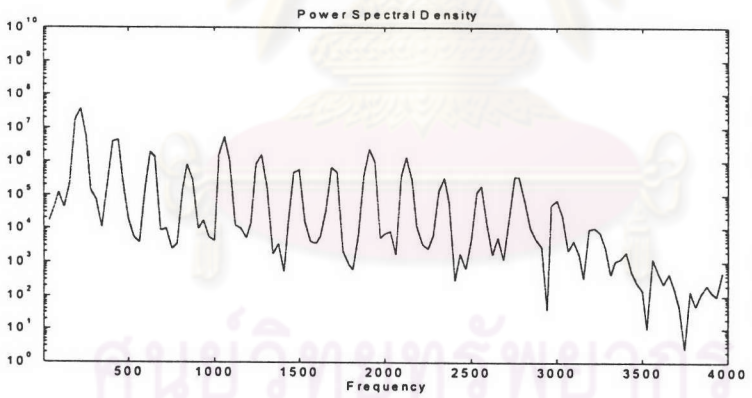

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



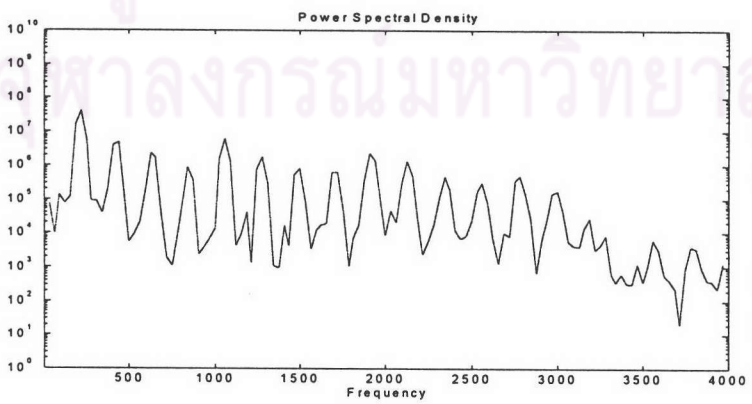

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

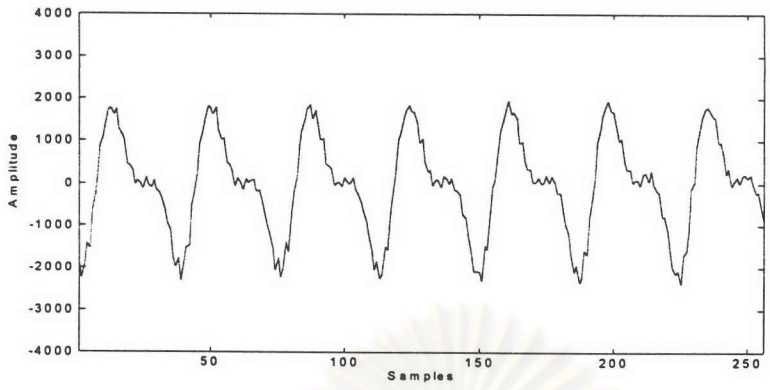
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

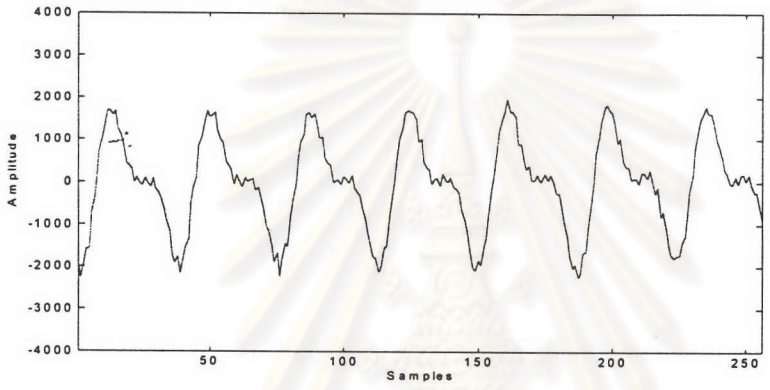
รูป ก.10 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 3001 ถึง 3256 ของเสียงจากรูป ก.5




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

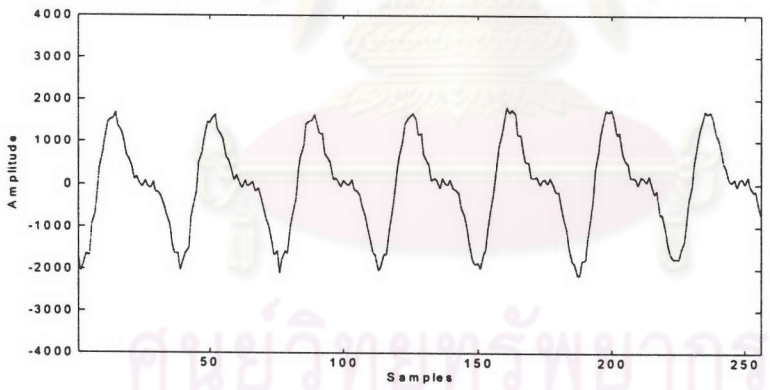
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

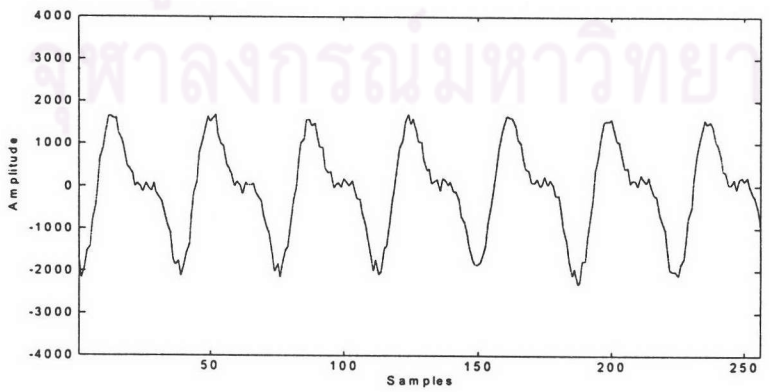
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

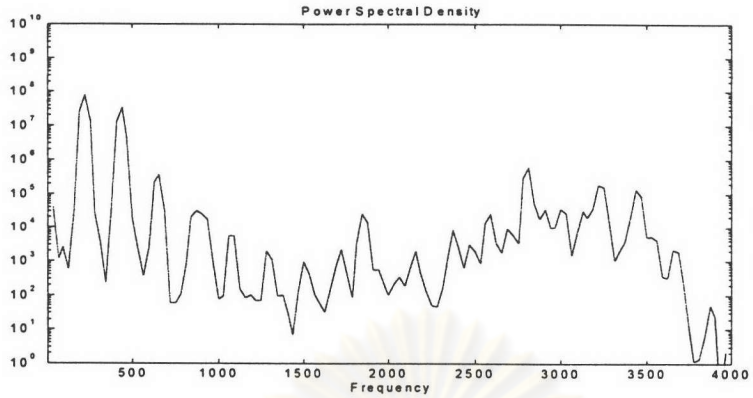



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

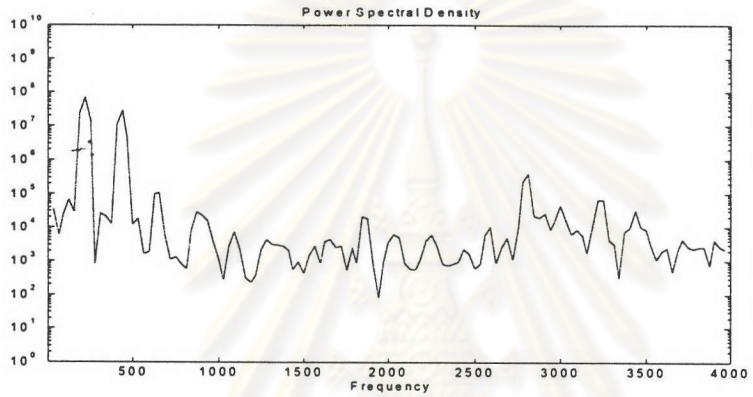
แกนนอน = ตัวอย่างสุ่ม

รูป ก.11 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 5501 ถึง 5756 ของเสียงจากรูป ก.5



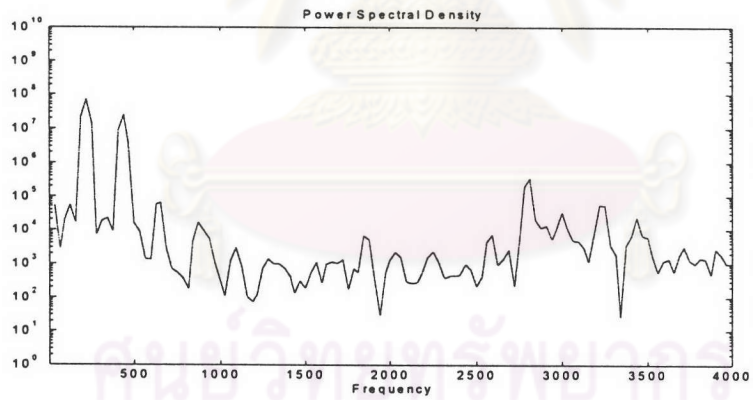

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



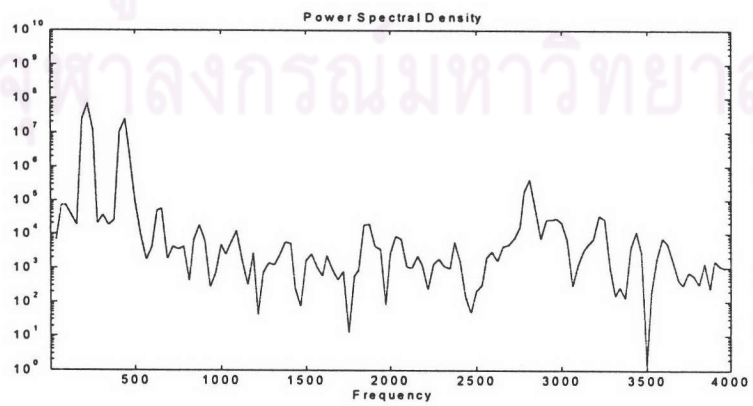

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

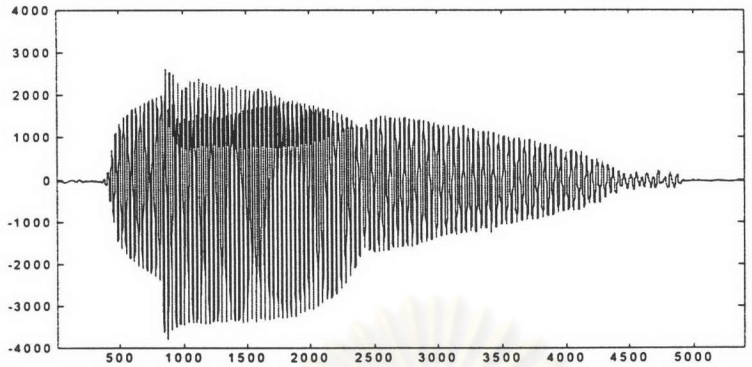
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

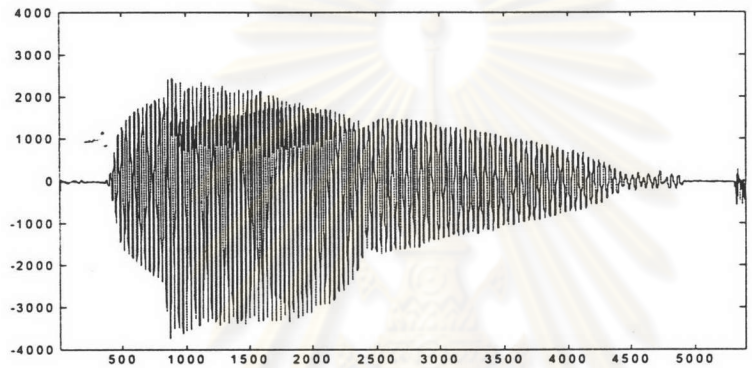
รูป ก.12 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 5501 ถึง 5756 ของเสียงจากรูป ก.5




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

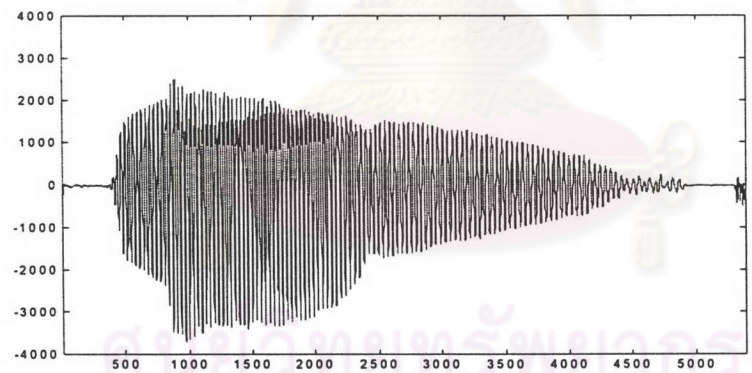
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

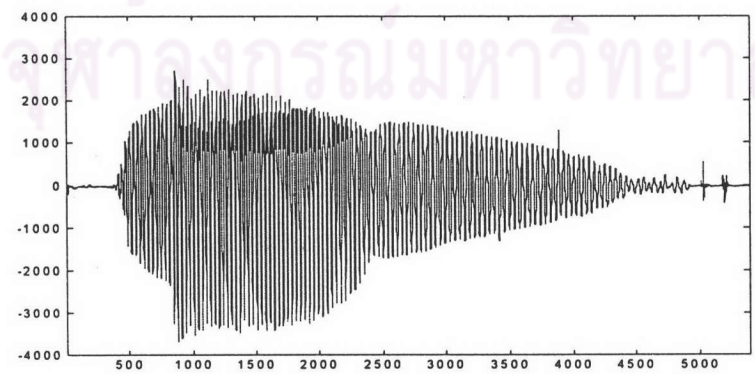
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

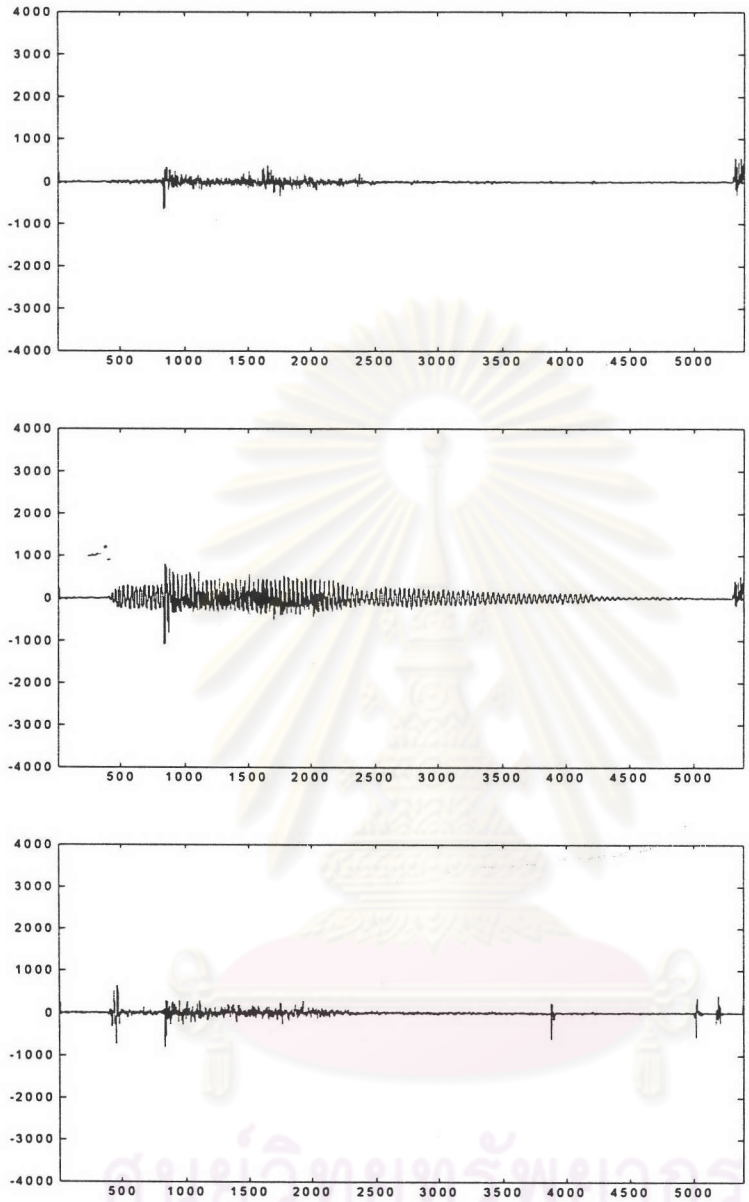



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

รูป ก.13 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "หนึ่ง"




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

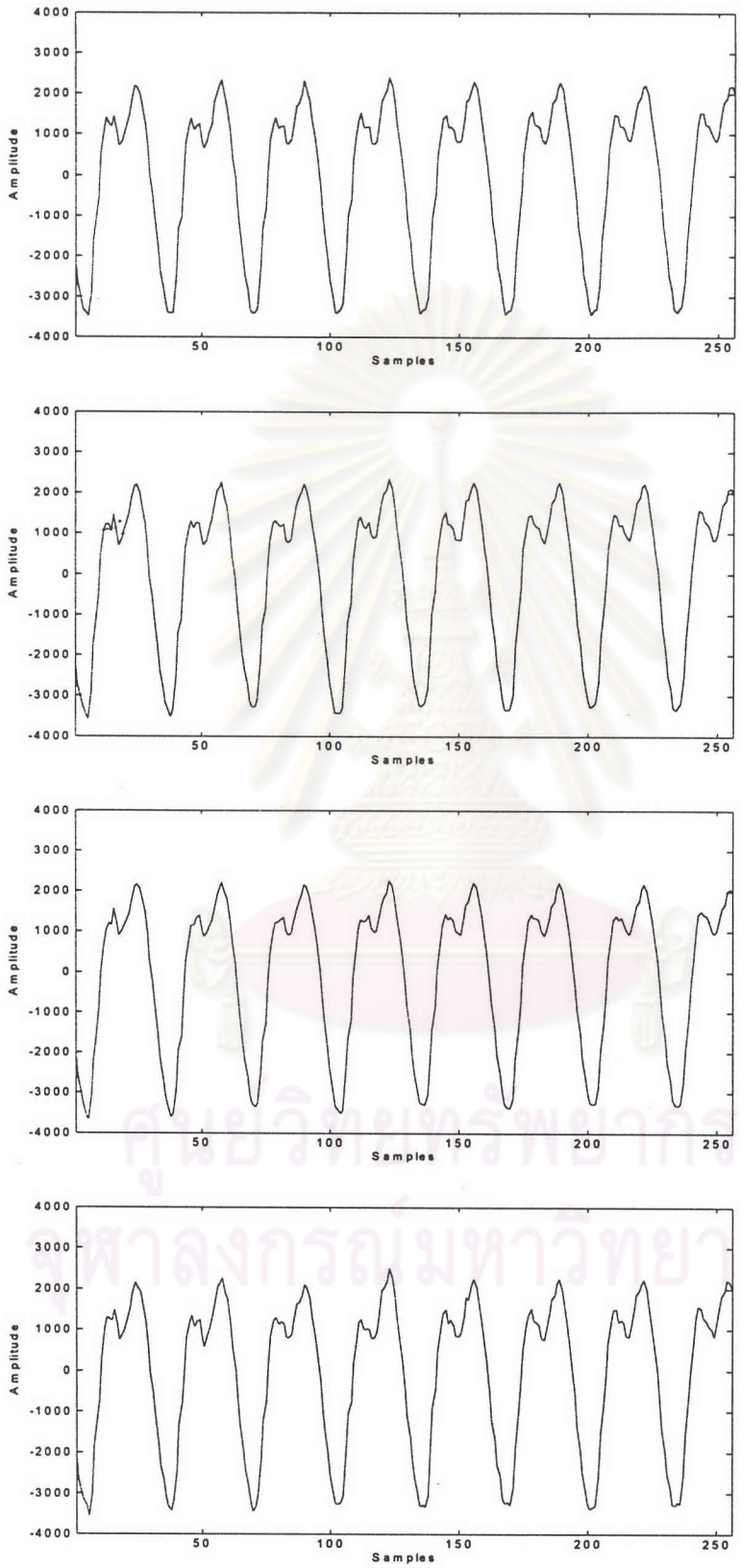
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR = 26.5 เดซิเบล

(ข) SNR =17.3 เดซิเบล

(ค) SNR =26.0 เดซิเบล

รูป ก.14 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "หนึ่ง"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

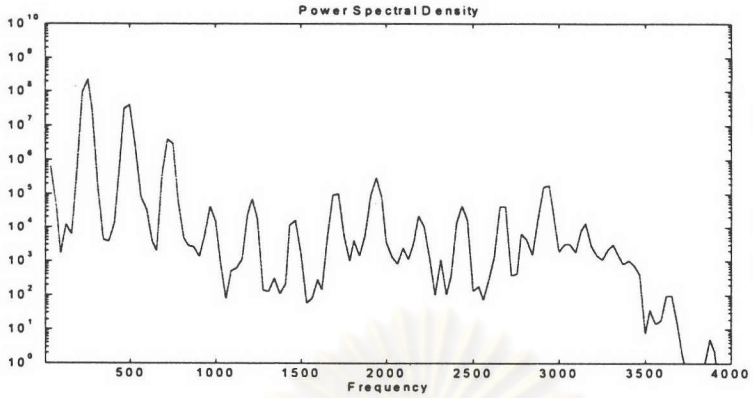
แกนนอน = ตัวอย่างสุ่ม

(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

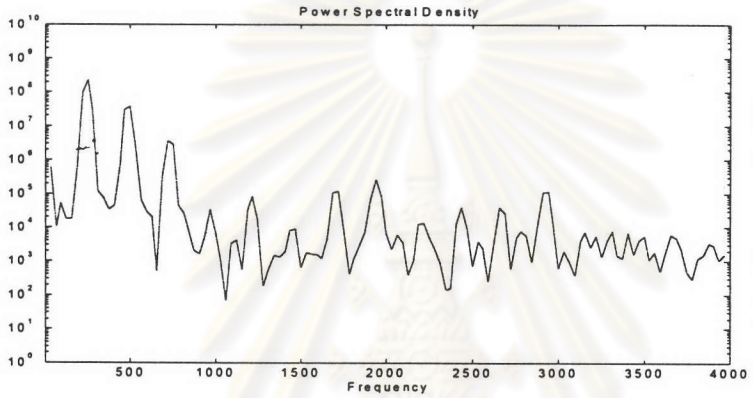
แกนนอน = ตัวอย่างสุ่ม

รูป ก.15 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.13



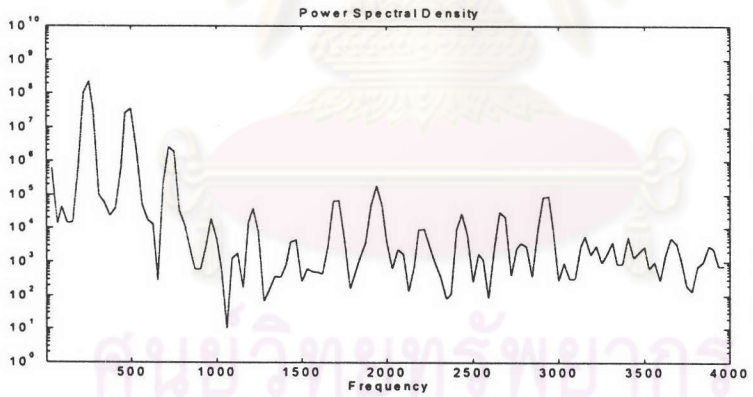

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



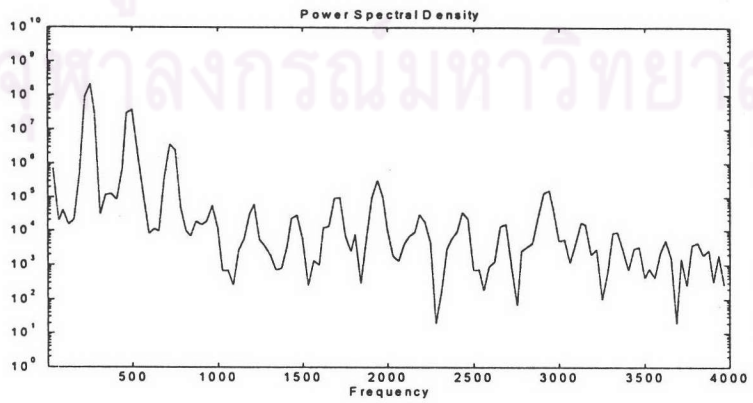

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

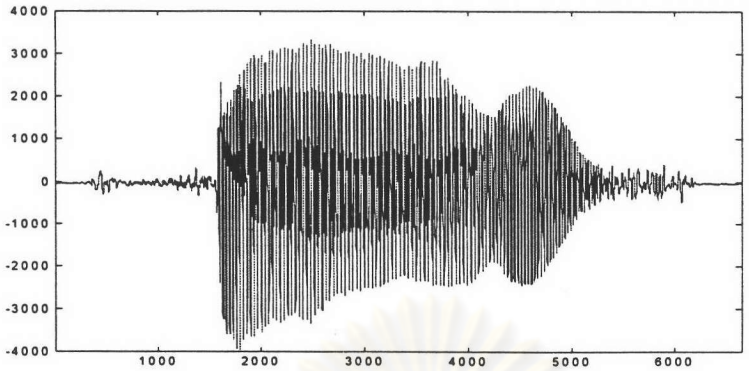
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

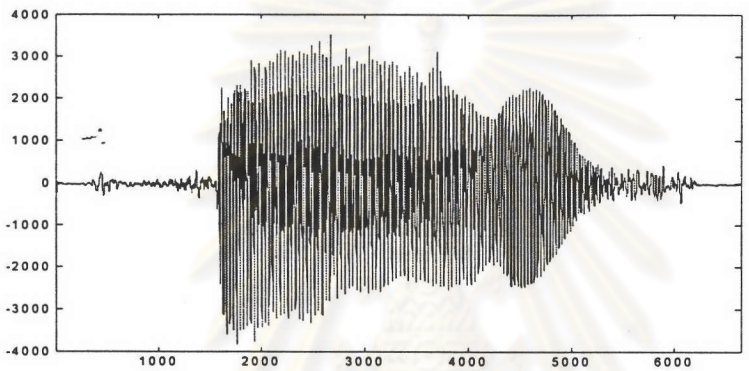
รูป ก.16 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.13




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

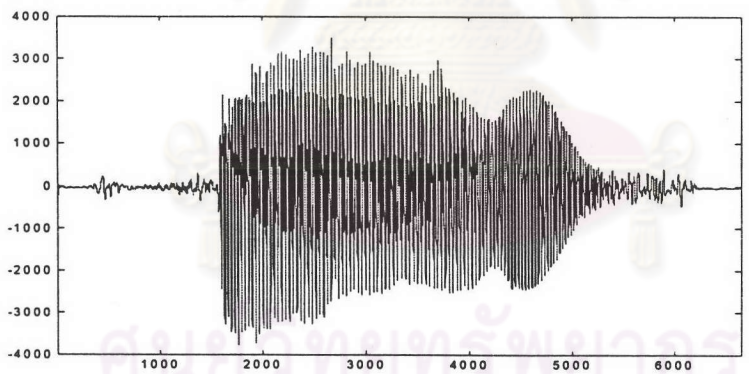
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

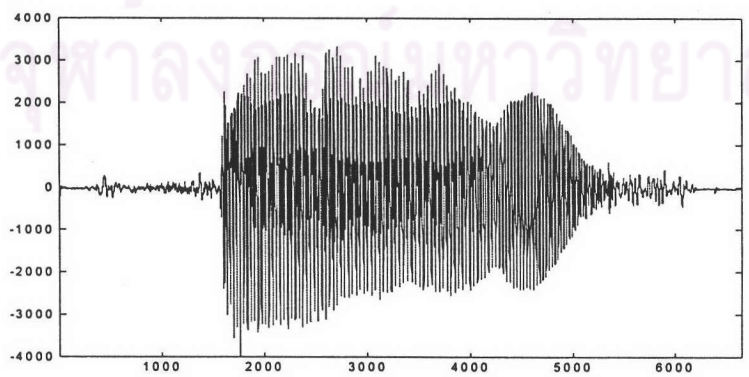
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

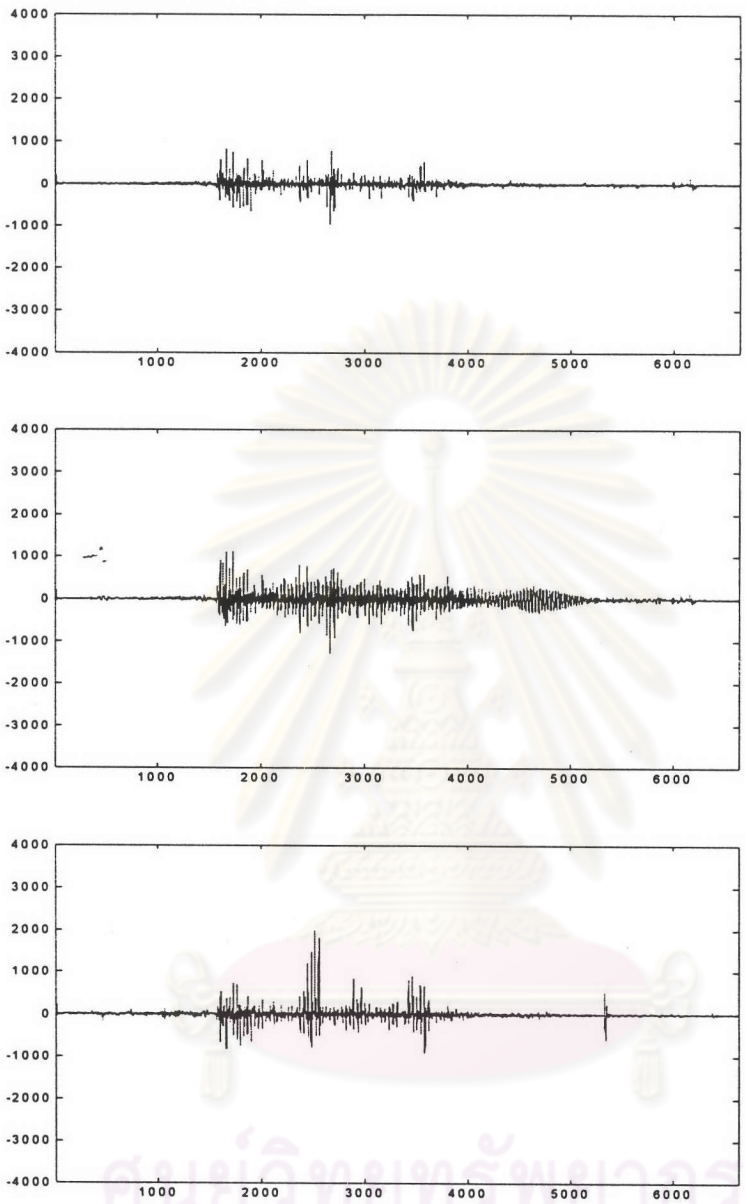



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

รูป ก.17 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "สอง"




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

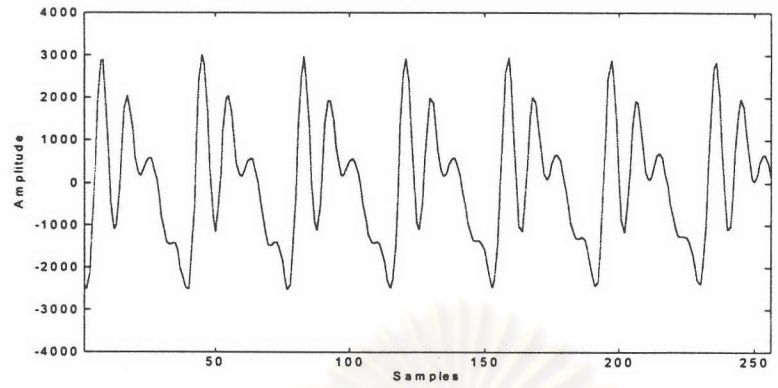
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR = 22.9 เดซิเบล

(ข) SNR =17.6 เดซิเบล

(ค) SNR =19.3 เดซิเบล

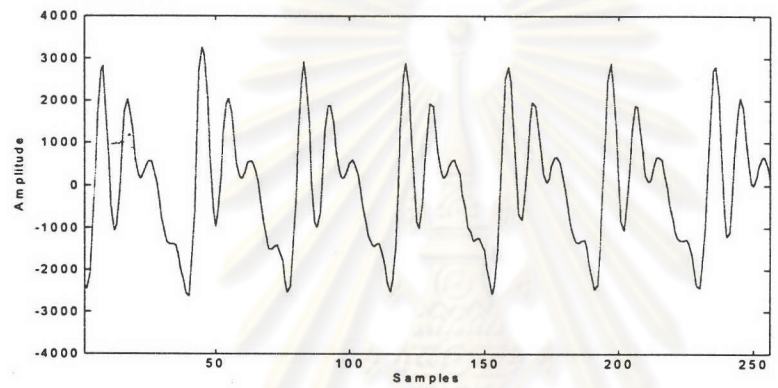
รูป ก.18 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "สอง"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

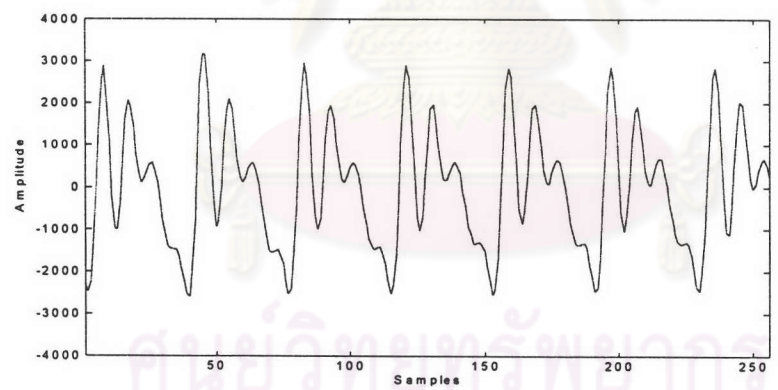
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

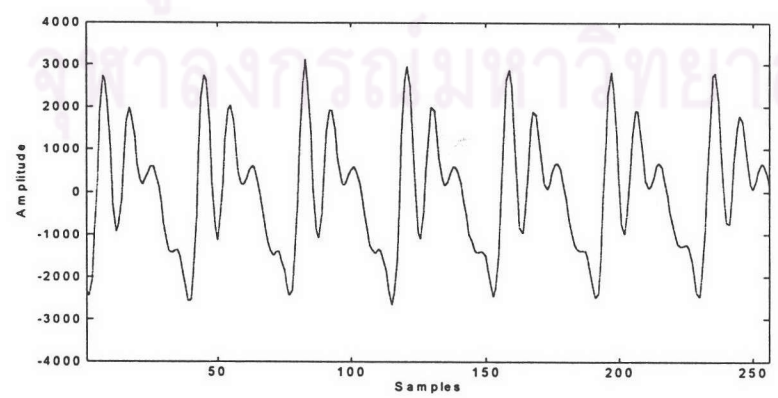
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

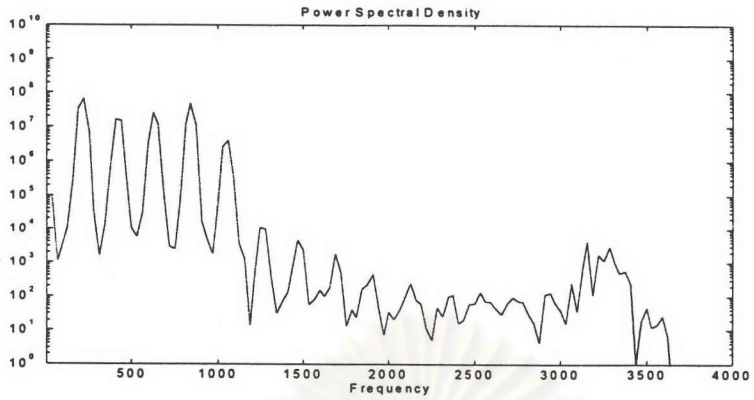



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

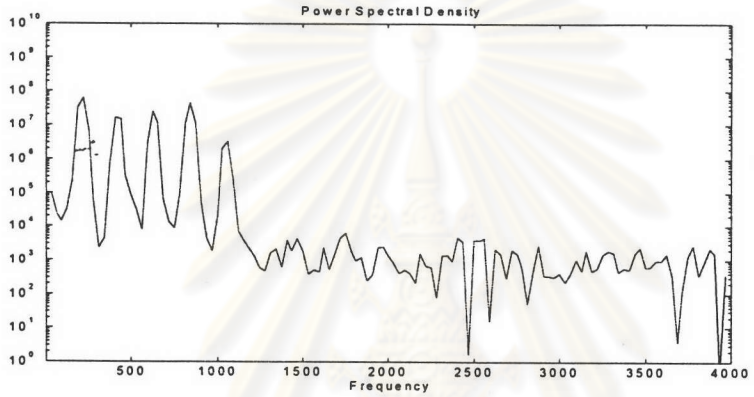
แกนนอน = ตัวอย่างสุ่ม

รูป ก.19 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 3001 ถึง 3256 ของเสียงจากรูป ก.17



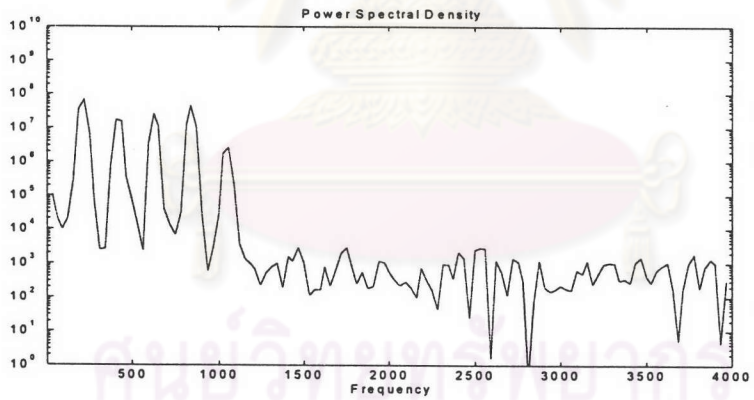

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



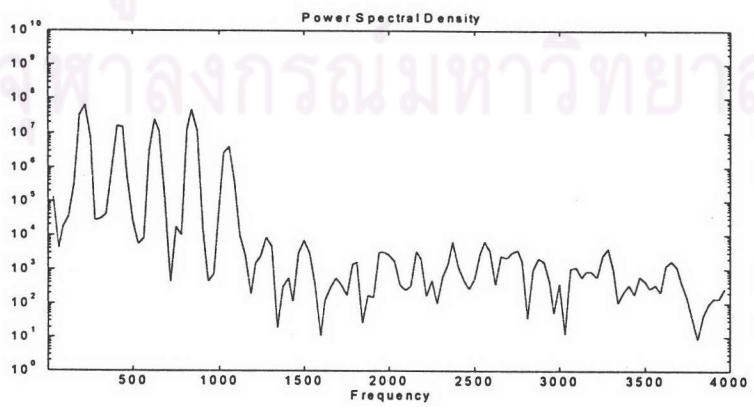

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

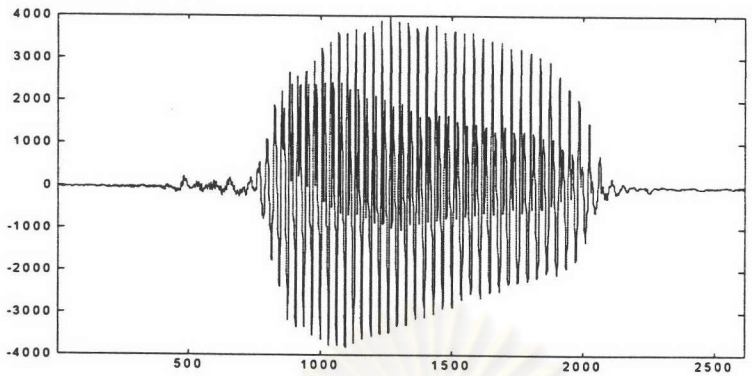
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

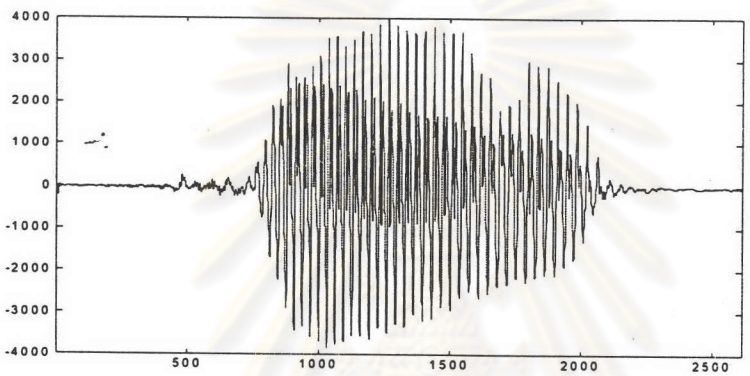
รูป ก.20 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 3001 ถึง 3256 ของเสียงจากรูป ก.17




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

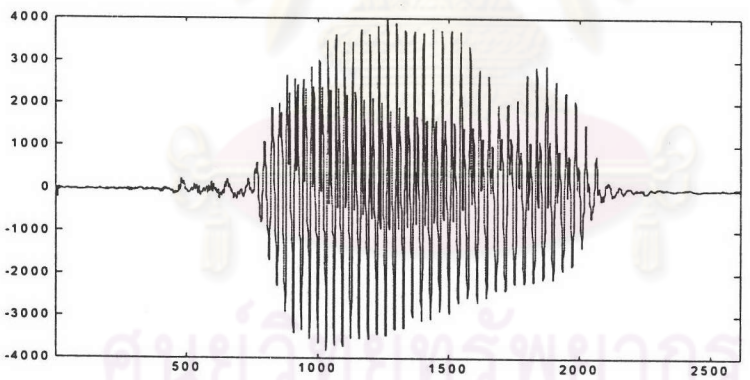
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

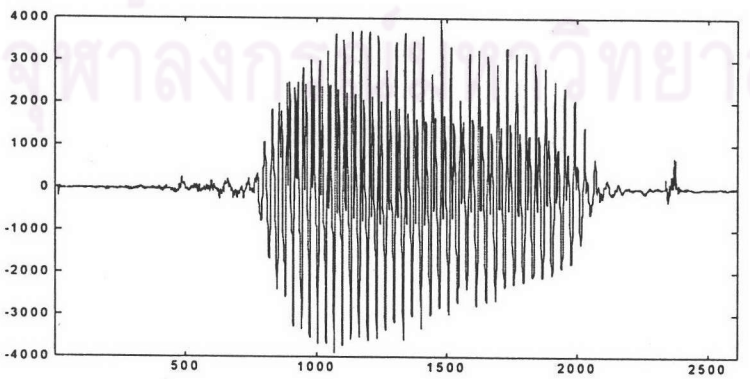
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

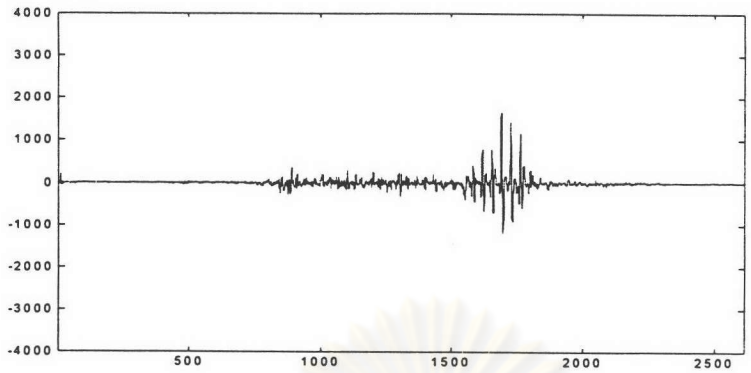



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

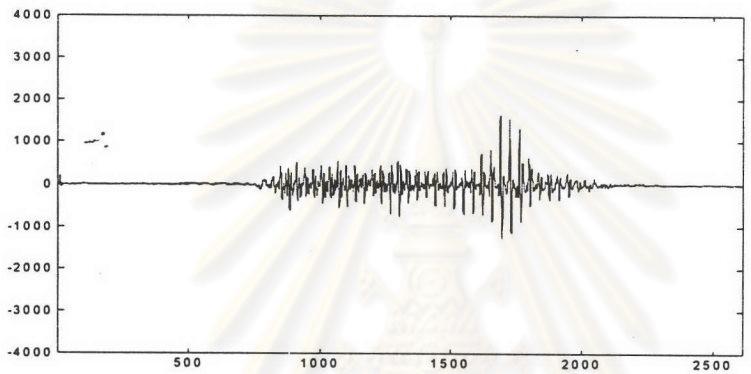
แกนนอน = ตัวอย่างสุ่ม

รูป ก.21 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "หก"



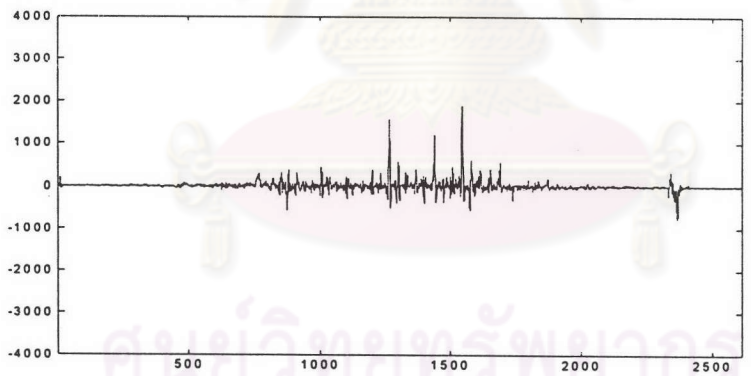

(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด
แกนนอน = ตัวอย่างสุ่ม

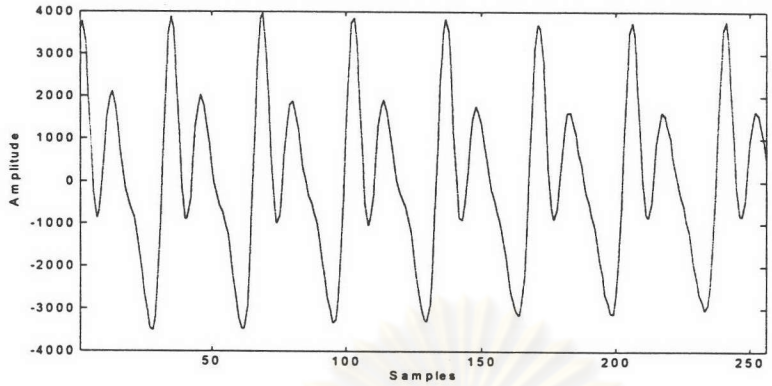



(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด
แกนนอน = ตัวอย่างสุ่ม

SNR =19.0 เดซิเบล
SNR =16.2 เดซิเบล
SNR =19.3 เดซิเบล

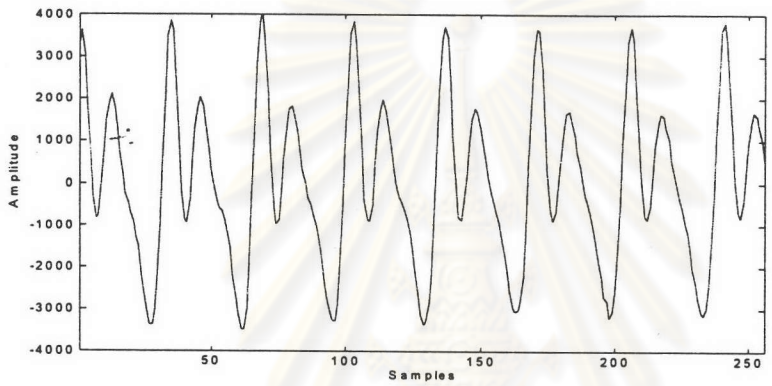
รูป ก.22 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "หก"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

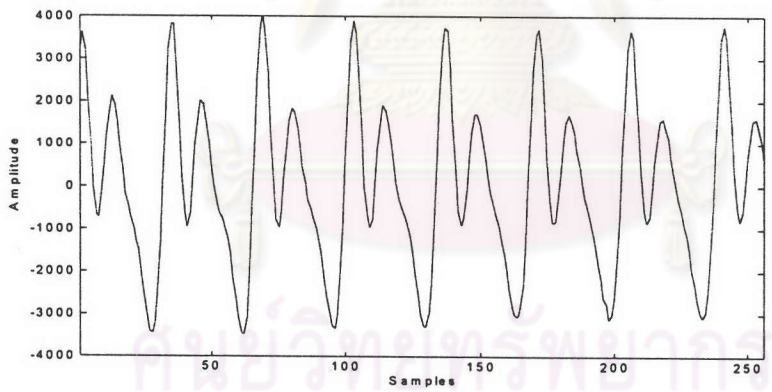
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

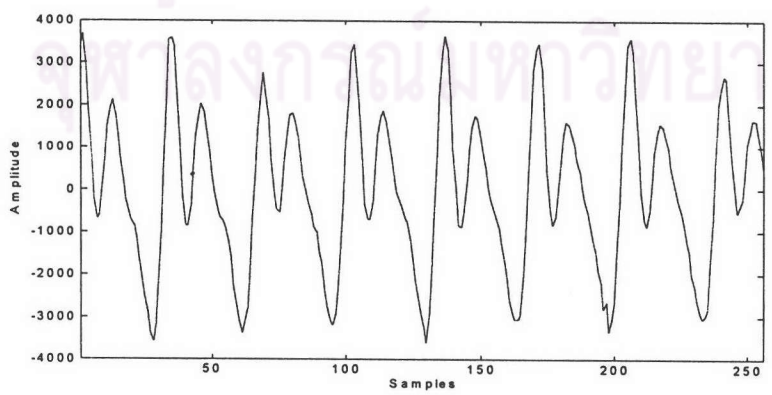
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

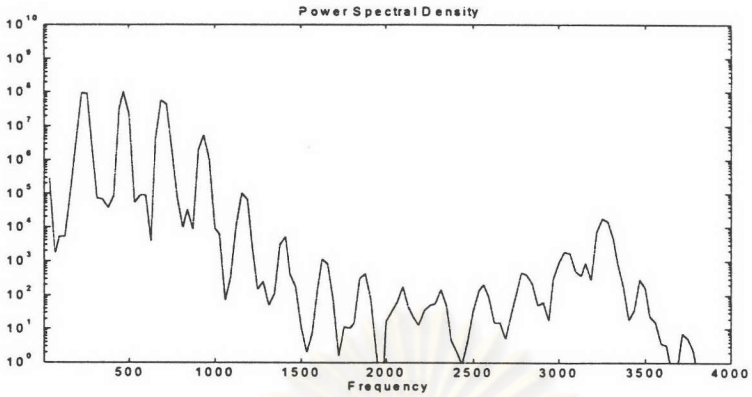



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

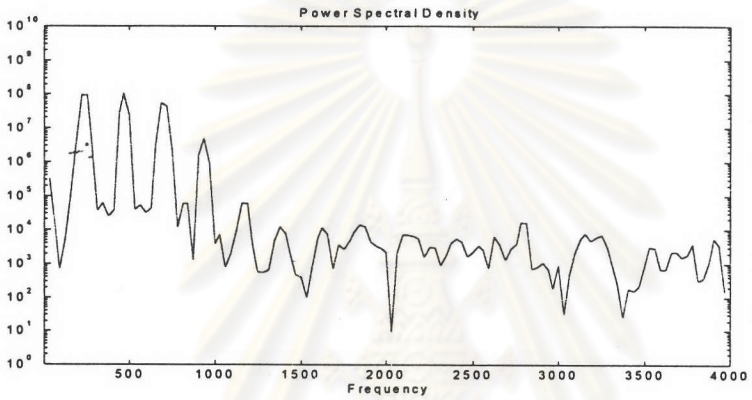
แกนนอน = ตัวอย่างสุ่ม

รูป ก.23 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1201 ถึง 1456 ของเสียงจากรูป ก.21



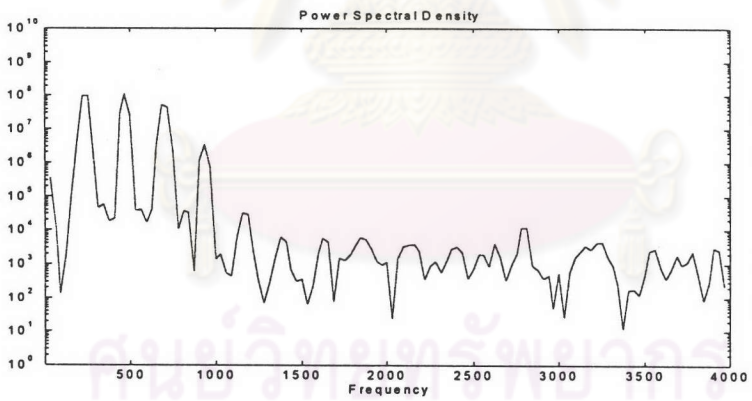

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



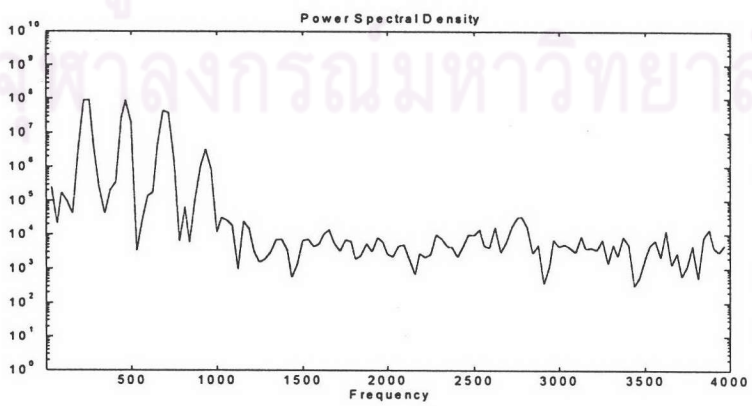

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

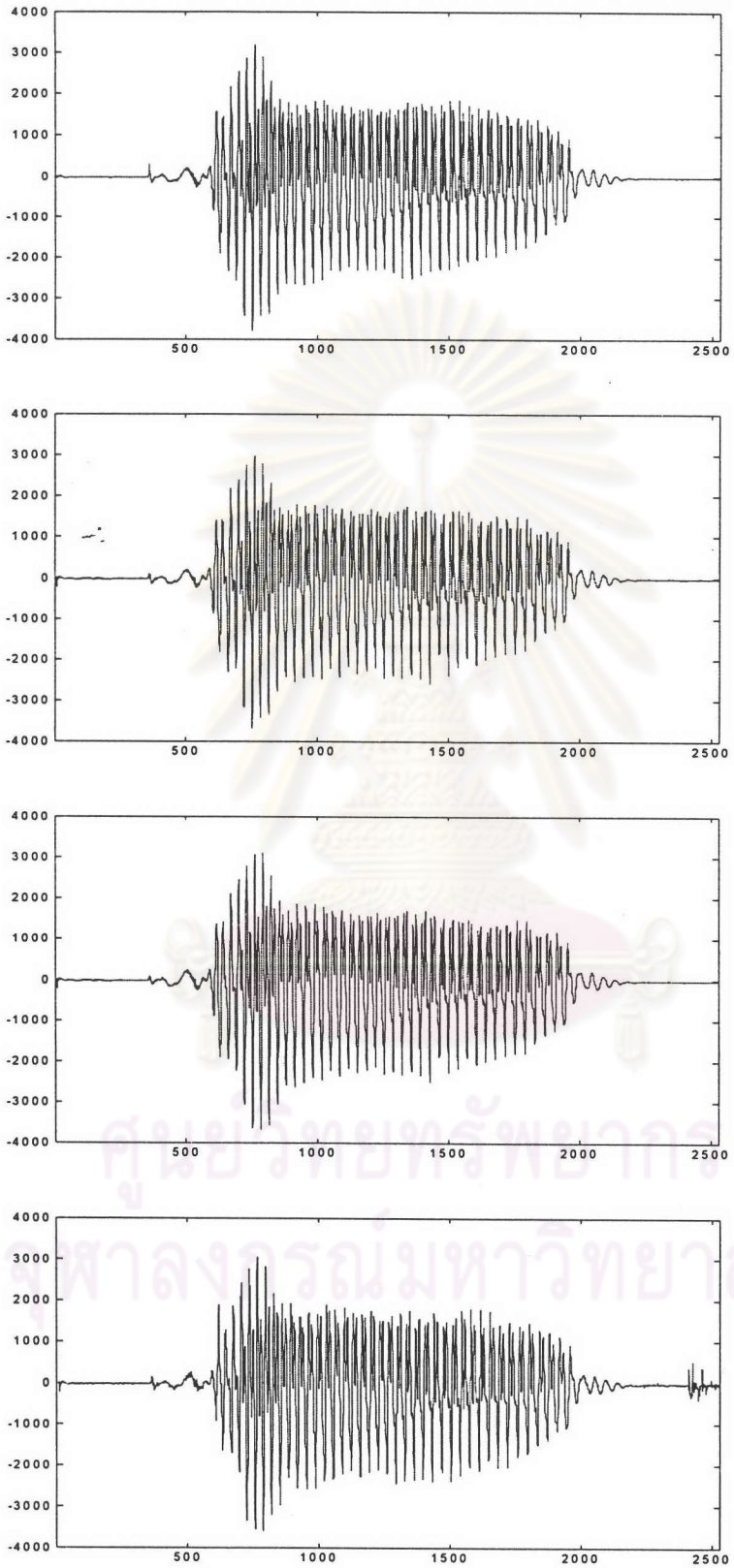
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

รูป ก.24 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1201 ถึง 1456 ของเสียงจากรูป ก.21




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

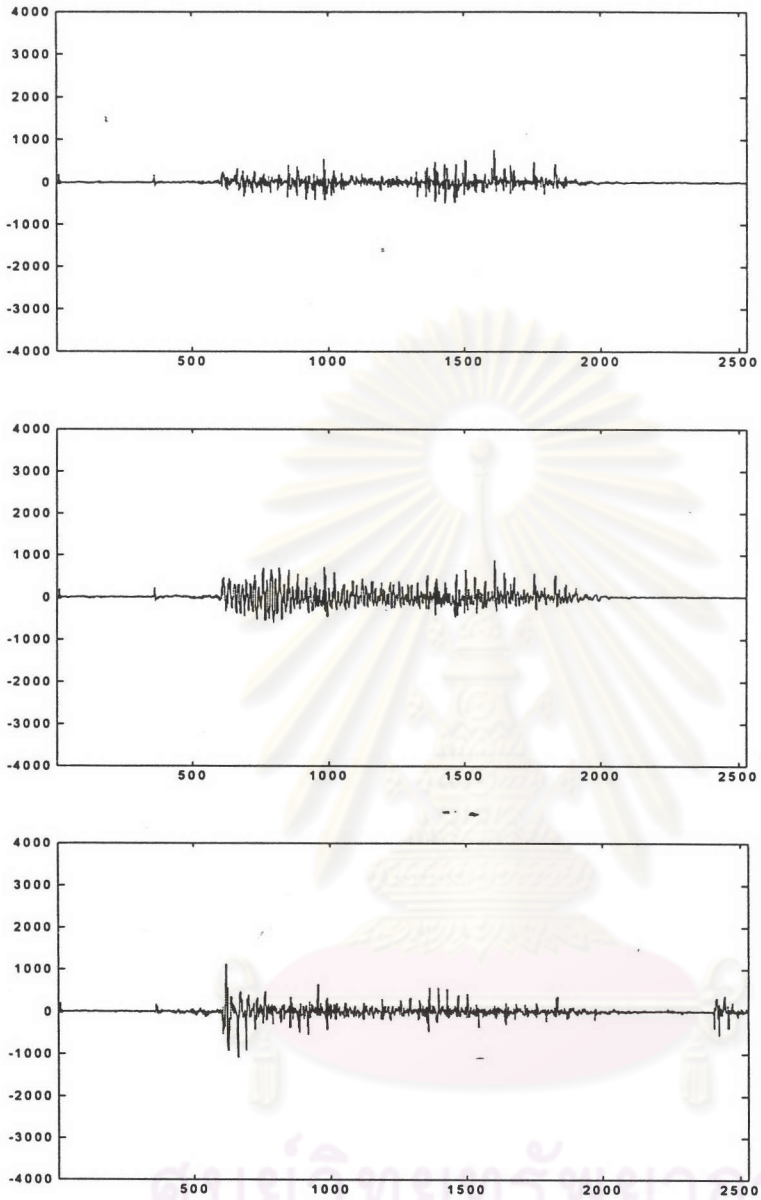
แกนนอน = ตัวอย่างสุ่ม

(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

รูป ก.25 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "เจ็ด"




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

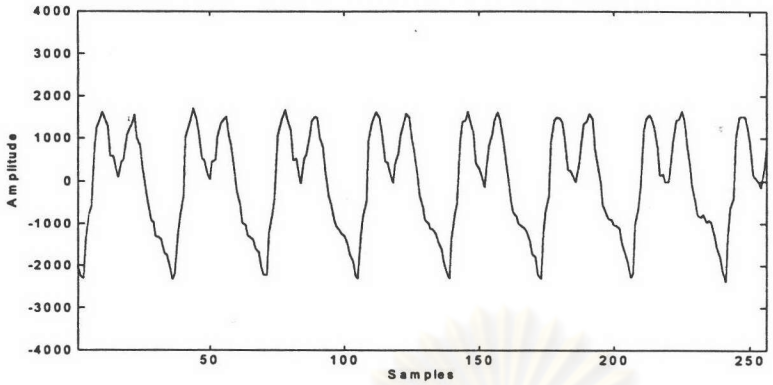
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR =20.1 เดซิเบล

(ข) SNR =15.6 เดซิเบล

(ค) SNR =17.9 เดซิเบล

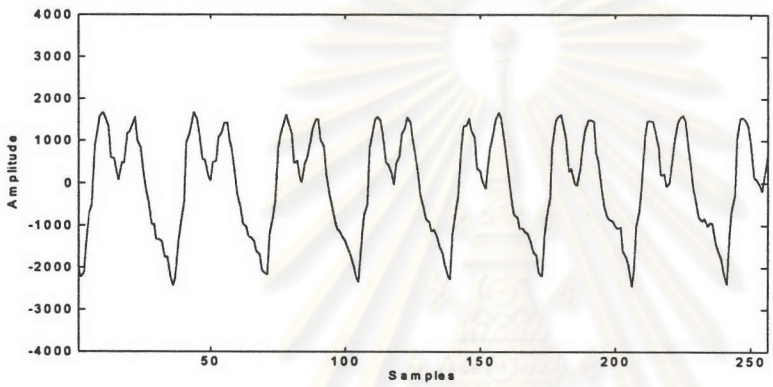
รูป ก.26 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "เจ็ด"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

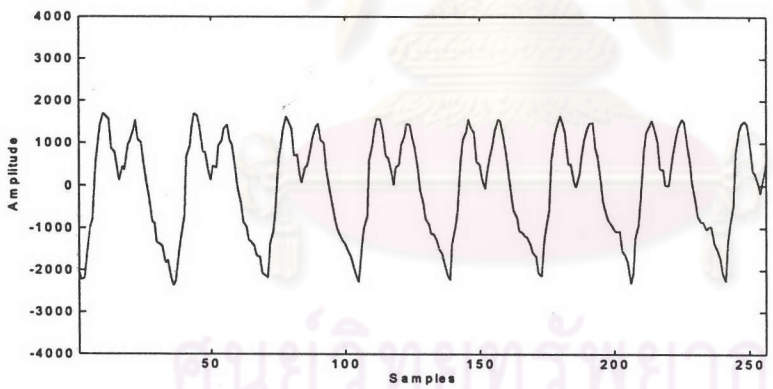
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

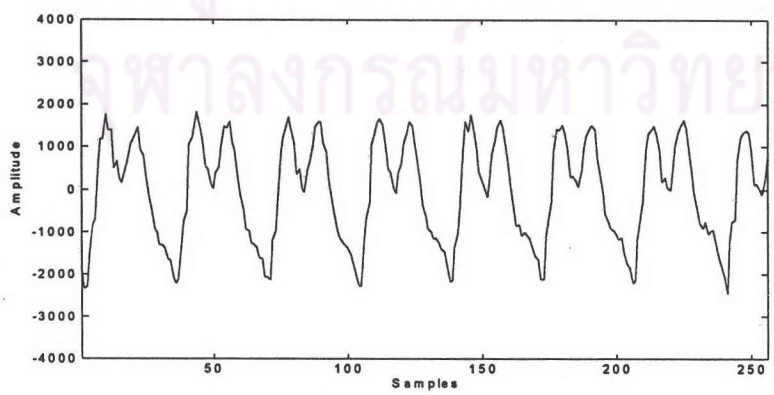
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

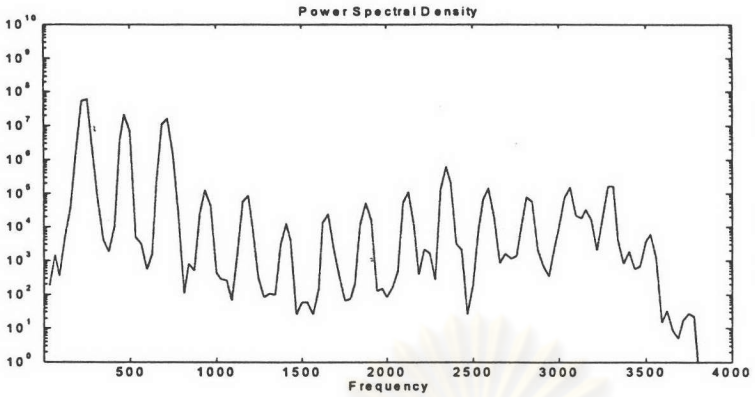



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

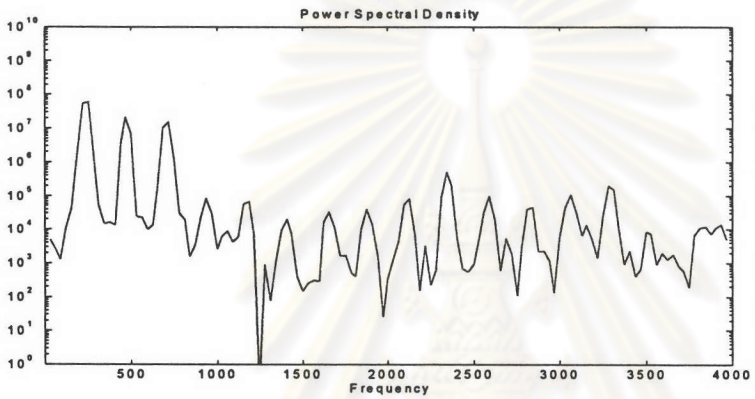
แกนนอน = ตัวอย่างสุ่ม

รูป ก.27 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1051 ถึง 1306 ของเสียงจากรูป ก.25



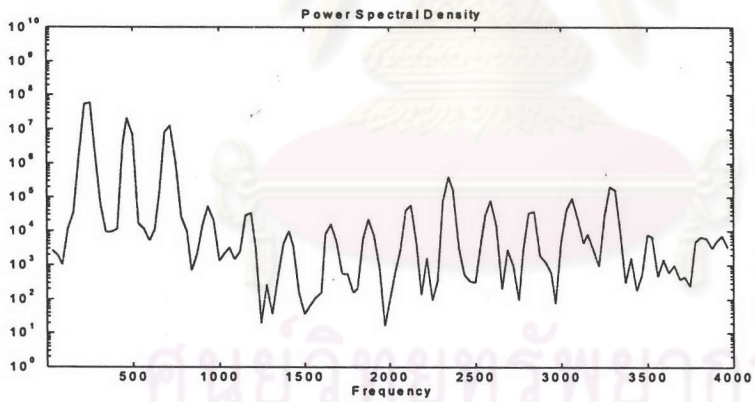

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



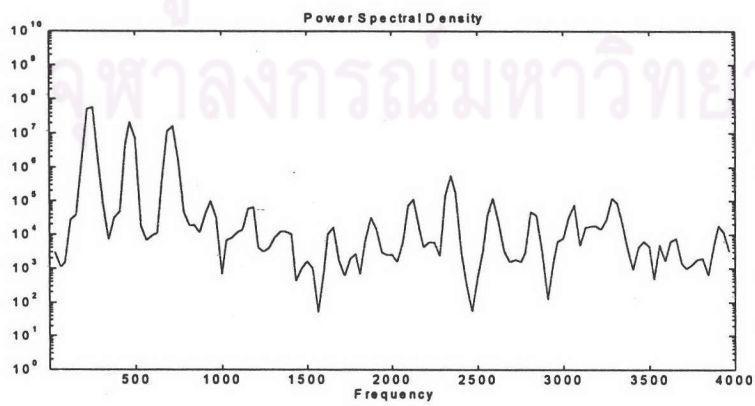

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

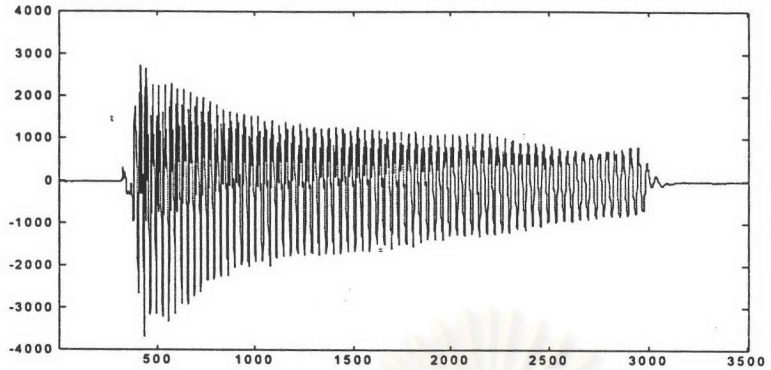
แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

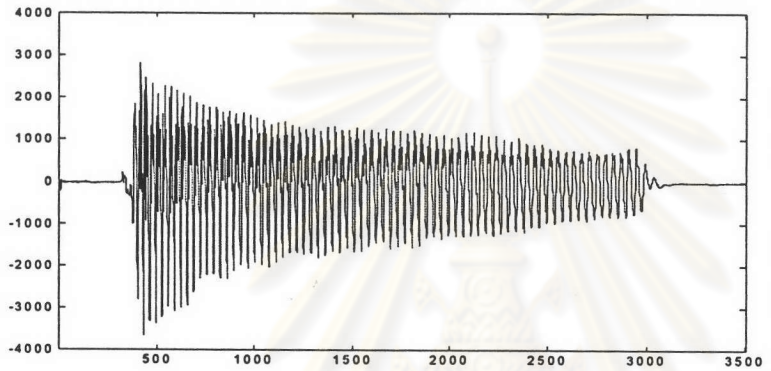
รูป ก.28 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1051 ถึง 1306 ของเสียงจากรูป ก.25




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

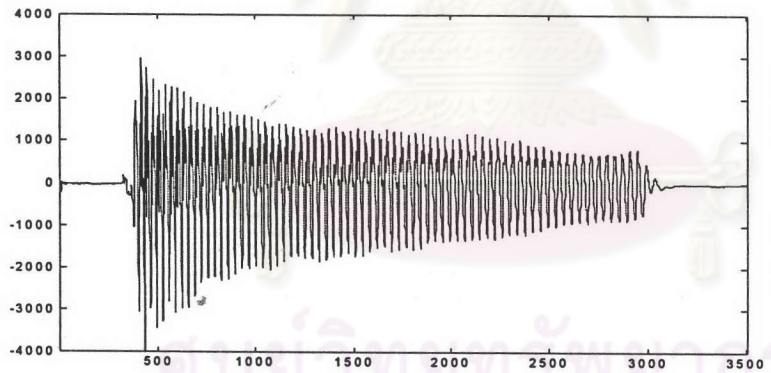
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

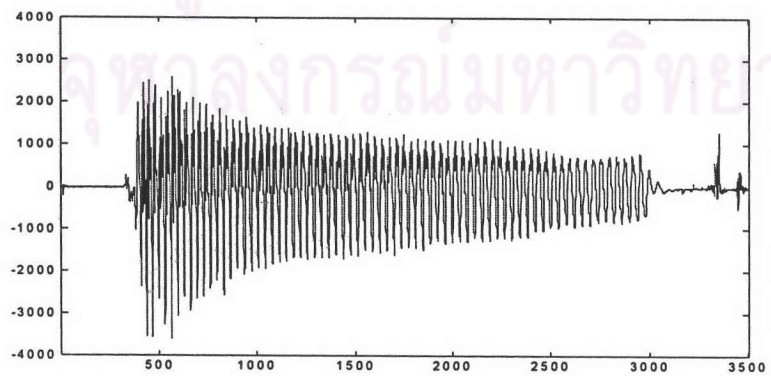
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

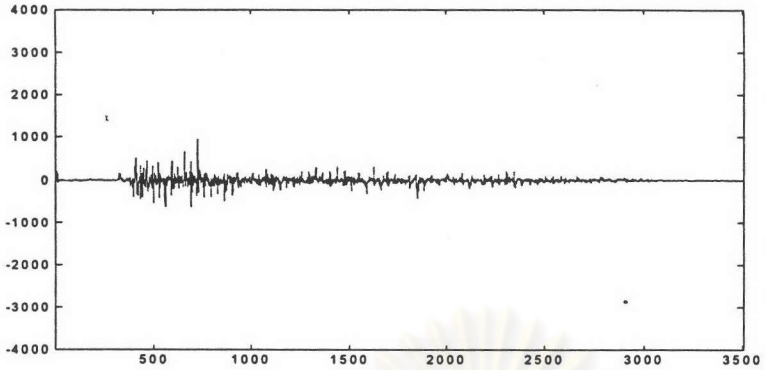



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

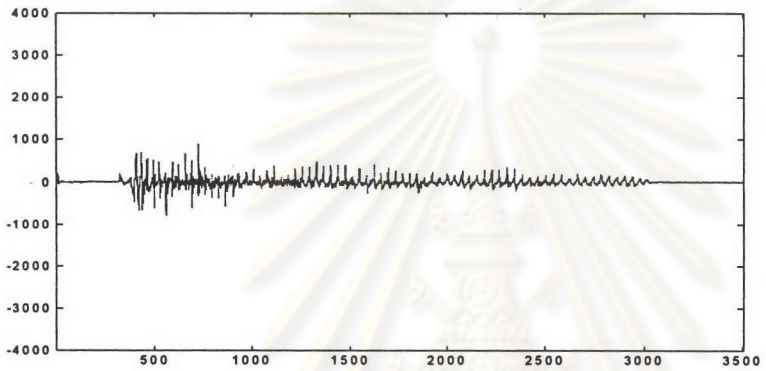
รูป ก.29 สัญญาณเสียงทั้งหมดของเสียงพูดคำว่า "แปด"




(ก) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

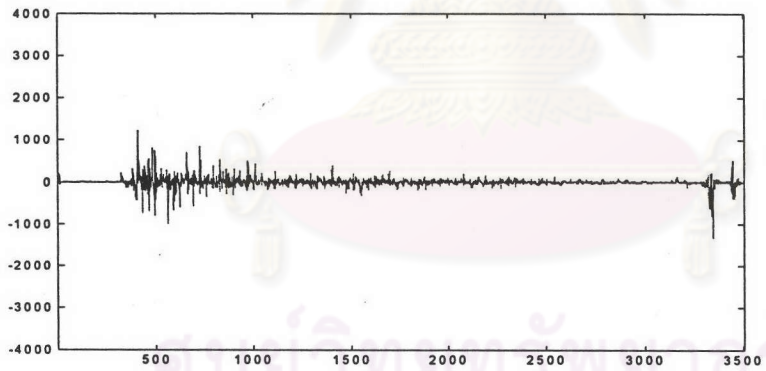
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

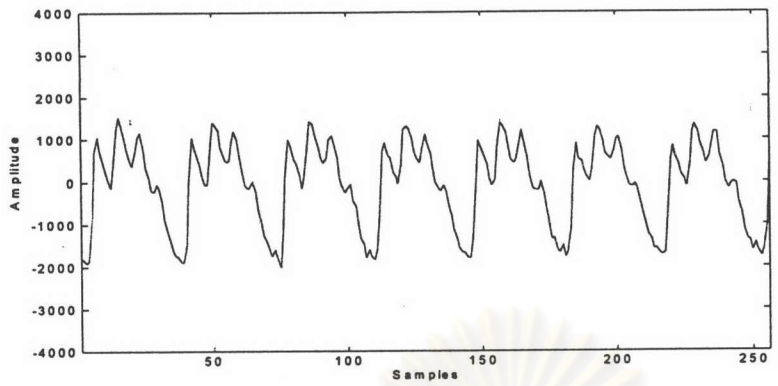
แกนนอน = ตัวอย่างสุ่ม

(ก) SNR =20.3 เดซิเบล

(ข) SNR =18.1 เดซิเบล

(ค) SNR =17.4 เดซิเบล

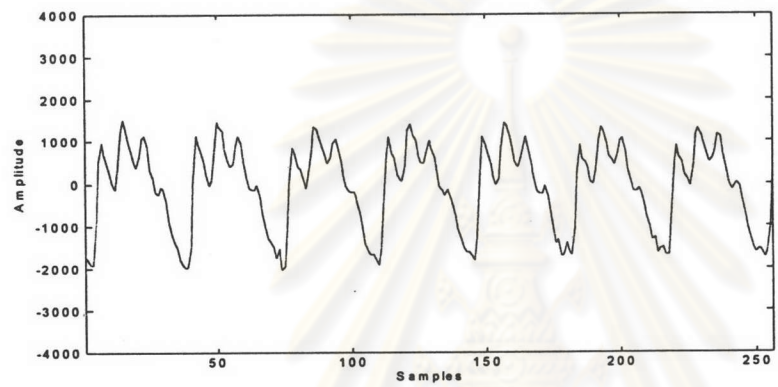
รูป ก.30 ผลต่างระหว่างสัญญาณเสียงจากการเข้ารหัสทั้งหมดกับสัญญาณเสียงต้นฉบับคำว่า "แปด"




(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาด

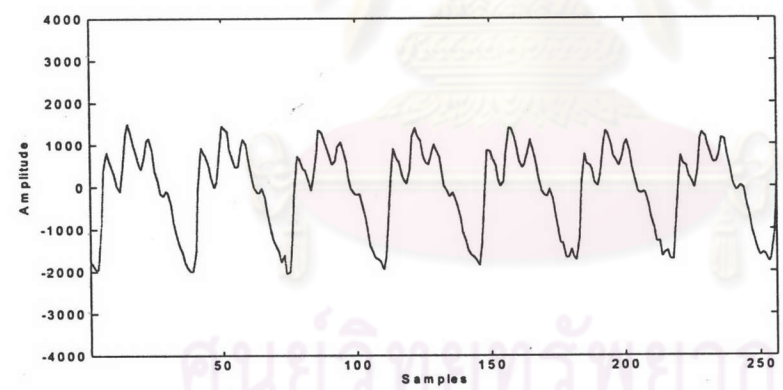
แกนนอน = ตัวอย่างสุ่ม




(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาด

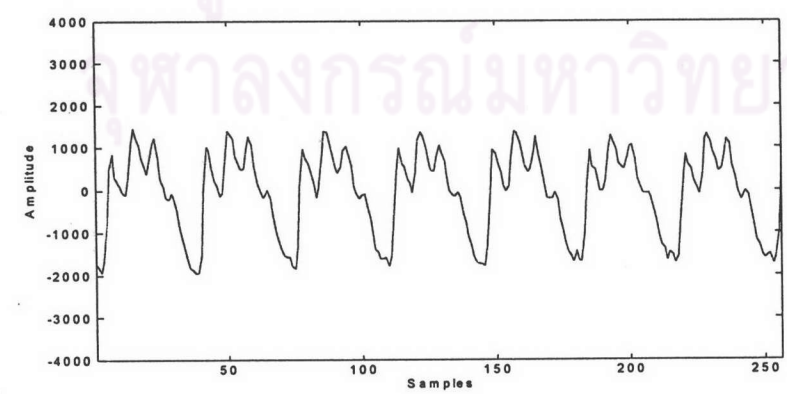
แกนนอน = ตัวอย่างสุ่ม




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาด

แกนนอน = ตัวอย่างสุ่ม

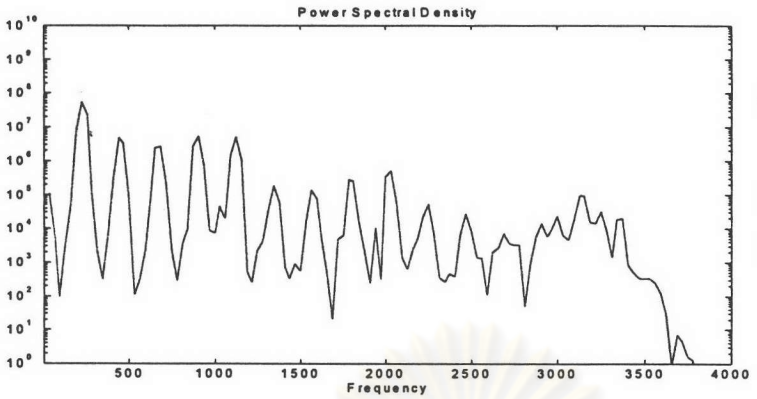



(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาด

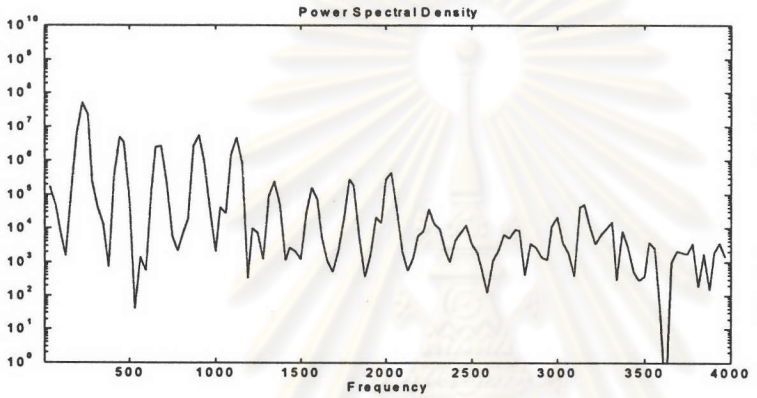
แกนนอน = ตัวอย่างสุ่ม

รูป ก.31 สัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.29



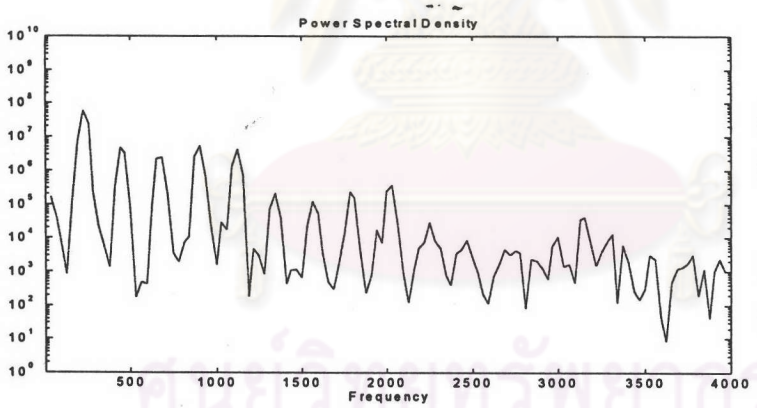

(ก) สัญญาณเสียงต้นฉบับ

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่



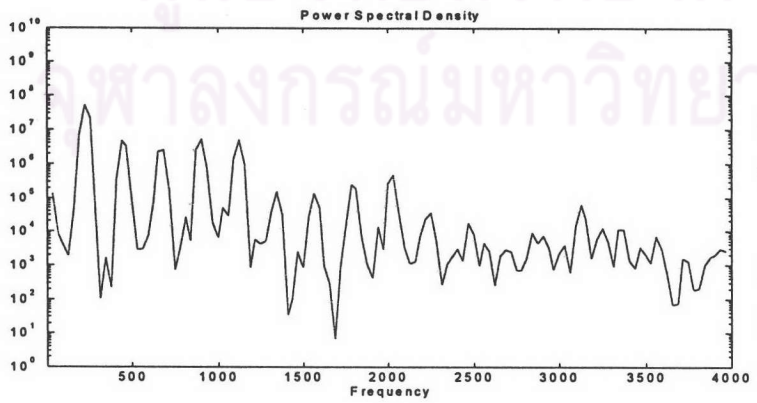

(ข) สัญญาณเสียงจากการเข้ารหัสบน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ค) สัญญาณเสียงจากโพสต์ฟิลเตอร์บน MATLAB

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่




(ง) สัญญาณเสียงจากการเข้ารหัสบนแอสเซมบลี

แกนตั้ง = ขนาดเชิงลอการิธึม
แกนนอน = ความถี่

รูป ก.32 สเปกตรัมของสัญญาณเสียงในช่วงตัวอย่างสุ่มที่ 1001 ถึง 1256 ของเสียงจากรูป ก.29

## ภาคผนวก ข

### ข้อแนะนำในการเขียนโปรแกรม

-ปัญหาเกี่ยวกับไปป์ไลน์ (pipeline) C5x มีขั้นตอนของการทำงานของคำสั่งอยู่ 4 ระดับหรือเรียกว่ามีไปป์ไลน์ที่ลึก 4 ระดับดังนี้ fetch decode operand และ execute ในขณะหนึ่งนั้นคำสั่งทั่วไปจะทำงานได้พร้อมกันตั้งแต่ 1 ถึง 4 คำสั่งโดยทำงานที่ระดับของไปป์ไลน์ที่ต่างกันทำงานต่อเนื่องกันไปจะใช้เวลาในการทำงานประมาณ 1 ถึง 2 รอบคำสั่ง ส่วนคำสั่งที่แสดงความไม่ต่อเนื่องของโปรแกรมคือคำสั่งที่เปลี่ยนค่าโปรแกรมเคาน์เตอร์จากค่าเดิมไปเป็นค่าอื่น นั่นคือทำให้โปรแกรมกระโดดจากจุดหนึ่งที่กำลังทำงานอยู่ไปทำงานต่อที่จุดอื่นจะทำให้ไปป์ไลน์ทำงานไม่ต่อเนื่องส่งผลให้คำสั่งจำพวกนี้เป็นคำสั่งที่ใช้เวลาในการทำงาน 4 รอบคำสั่ง เช่นคำสั่ง B, BCND, BACC เป็นต้น

รีจิสเตอร์ที่สามารถอ้างถึงได้จากหน่วยความจำ (memory-mapped register หรือ MMR )บางตัวจะมีปัญหาเรื่องเวลาในการเขียนค่าด้วยคำสั่งบางตัวเช่น SAMM, LMMR, SMMR, SACL, SPLK เป็นต้น เนื่องมาจากการเขียนค่าลงในรีจิสเตอร์เหล่านี้จะทำงานในระดับไปป์ไลน์สุดท้ายคือระดับ execute แต่การอ่านค่าออกมาจากรีจิสเตอร์เหล่านี้ทำงานในระดับไปป์ไลน์ที่ 2 หรือ 3 ทำให้การใช้ค่าของรีจิสเตอร์เหล่านี้ทันทีหลังจากการเขียนค่าของมันจะได้ค่าที่ไม่ใช่ค่าที่ต้องการเพราะการเขียนยังไม่เสร็จสมบูรณ์ ค่าที่อ่านได้จึงเป็นค่าเก่าที่เก็บไว้ก่อนหน้านี้ ดังตัวอย่างต่อไปนี้

```
MAR     *,AR1       ;
LAR     AR1,#64     ; AR1=64
LACC    #60
SAMM    AR1         ; AR1=64
MAR     *-          ; AR1=63
MAR     *-          ; AR1=60
MAR     *-          ; AR1=59
```

การป้องกันปัญหาในเรื่องนี้ต้องมีการแทรกคำสั่งที่ไม่เกี่ยวข้องกับการใช้ค่า MMR ที่เพิ่งจะเปลี่ยนค่าไป ก่อนที่จะเป็นคำสั่งที่อ่านค่า MMR นั้นกลับมาใช้ คำสั่งที่แทรกอาจจะเป็น NOP (no operation)ก็ได้ จำนวนคำสั่งที่ต้องแทรกก็ขึ้นอยู่กับความต้องการของแต่ละ MMR ว่าต้องการเวลามากแค่ไหนในการทำการเขียนให้เสร็จสมบูรณ์ ค่าดังกล่าวของรีจิสเตอร์ที่กล่าวข้างต้นแสดงดังตารางต่อไปนี้

ตาราง ข.1 จำนวนรอบคำสั่งที่ต้องการสำหรับการเขียนค่าลงในรีจิสเตอร์

| ชื่อ | จำนวนคำสั่ง |
|---|---|
| GREG | 1 |
| PMST | 2 |
| TREG1 | 1 |
| TREG2 | 1 |
| ARx | 2 |
| INDX | 2 |

ตาราง ข.1(ต่อ) จำนวนรอบคำสั่งที่ต้องการสำหรับการเขียนค่าลงในรีจิสเตอร์

| ชื่อ | จำนวนคำสั่ง |
|---|---|
| ARCR | 2 |
| CBSR | 2 |
| CBER | 2 |
| CBCR | 2 |
| BMAR | 1 |
| PDWSR | 1 |
| IOWSR | 1 |
| CWSR | 1 |
| CNF | 2 |

- การใช้บัฟเฟอร์วงกลม (Circular buffer) ควรจะจำกัดการใช้เฉพาะส่วนที่จำเป็นจริง ๆ เพราะการใช้แต่ละครั้งต้องมีการกำหนดค่าเริ่มต้นหลายค่า เป็นการเพิ่มจำนวนคำสั่งในโปรแกรมให้มากขึ้นอันไม่เป็นผลดีเนื่องมาจากการทำงานตามเวลาจริงต้องประหยัดจำนวนคำสั่งให้มากที่สุด ที่สำคัญไม่ควรใช้บัฟเฟอร์วงกลมอยู่ภายในเส้นทางบริการอินเตอรัปต์ เพราะมีค่ารีจิสเตอร์ที่ต้องใช้ในการทำงานมากและรีจิสเตอร์เหล่านั้นไม่ได้ถูกเก็บลงสแตกแบบอัตโนมัติ จำเป็นจะต้องเขียนคำสั่งให้เก็บค่าของรีจิสเตอร์เหล่านี้เองเพื่อไม่ให้โปรแกรมทำงานผิดพลาดในกรณีที่เกิดอินเตอรัปในขณะที่มีบัฟเฟอร์วงกลมตัวหนึ่งกำลังทำงานอยู่เพราะไม่เช่นนั้นค่าของรีจิสเตอร์ของบัฟเฟอร์วงกลมทั้ง 2 ตัวจะทำงานซ้ำซ้อนกัน ความผิดพลาดที่เกิดขึ้นจะมีความสำคัญอย่างยิ่งถ้าบัฟเฟอร์วงกลมนั้นทำงานเกี่ยวกับคำสั่งเขียนค่าลงบนหน่วยความจำ ถ้าค่าของรีจิสเตอร์ที่แสดงตำแหน่งของจุดสิ้นสุดของบัฟเฟอร์วงกลมกับค่าของตัวชี้ตำแหน่งของบัฟเฟอร์วงกลมทำงานไม่สัมพันธ์กันแล้วอาจจะเกิดการเขียนค่าของข้อมูลทับลงไปบนส่วนคำสั่งหรือโปรแกรมจะทำให้โปรแกรมทำงานผิดพลาดไปและอาจหยุดทำงานไปเลยก็ได้

- คำสั่งย้ายค่าเข้าแอคคิวมิวเลเตอร์ (ACC) อย่างเช่น LAMM หรือ LACL นั้นไม่มีคุณสมบัติขยายเครื่องหมาย (sign extended) ถ้าต้องการให้มีคุณสมบัตินี้ควรเลือกใช้คำสั่ง LACC แทน

- คำสั่ง LACL ที่ใช้กับค่าคงที่ k หรือ LACL #k ( -128<=k<=127 )ใช้การคำนวณเพียง 1 รอบการทำงาน ในขณะที่คำสั่ง LACC #lk ( -32768<=lk<=32767 )ใช้การคำนวณ 2 รอบการทำงาน

- คำสั่ง SAR ทำงานได้คล้ายกับคำสั่ง SMMR แต่ใช้การคำนวณที่น้อยกว่า 1 รอบการทำงาน

- พยายามใช้ค่าของตำแหน่งหน่วยความจำของข้อมูล( data memory address หรือ dma )ให้มากที่สุดและหลีกเลี่ยงการใช้ค่าคงที่ k เพราะทำให้สิ้นเปลืองการคำนวณมากกว่า

- ในการคำนวณถ้าสามารถสร้างตารางเอาไว้ล่วงหน้าแล้วจะลดปริมาณการคำนวณลงไปได้มาก

- คำสั่งเลื่อนข้อมูลไปทางซ้าย (Shift Left) หรือคำสั่ง SFL ไม่มีคุณสมบัติขยายเครื่องหมายแต่คำสั่งเลื่อนข้อมูลไปทางขวา (Shift Right) หรือ SFR มีคุณสมบัตินี้

- ไม่มีคำสั่งเพิ่มหรือลดค่าข้อมูลในหน่วยความจำหรือในรีจิสเตอร์ได้โดยตรงเหมือนอย่างในตัวประมวลผลทั่ว ๆ ไป มีเพียงรีจิสเตอร์ช่วย (Auxiliary Register หรือ AR) เท่านั้นที่สามารถทำเช่นนั้นได้ การเพิ่มค่าให้แก่หน่วยความจำหรือรีจิสเตอร์ตัวอื่น ๆ ต้องมีการคำนวณที่หน่วยประมวลผลหลักเท่านั้น

- คำสั่งที่ใช้ตัวแปร (Oparand) 2 ตัว ตัวอย่างเช่น MAC และ MACD ที่ใช้ตัวแปรตัวแรกจากตำแหน่งหน่วยความจำของโปรแกรม (programme memory address - pma) และตัวแปรตัวที่ 2 จาก dma ต้องระวังเรื่องตำแหน่งของตัวแปรด้วย ถ้าตัวแปรทั้งสองอยู่ในบล็อกของข้อมูลเดียวกันแล้วการคำนวณของคำสั่งจะช้ากว่ากรณีที่ตัวแปรทั้งสองอยู่ต่างบล็อกข้อมูลกันอยู่ 1 รอบการทำงานต่อหนึ่งคำสั่ง เนื่องจากคำสั่งจำพวกนี้ส่วนใหญ่จะทำงานแบบทำซ้ำหลายครั้งการคำนวณที่ได้จึงต่างกันได้หลายรอบการทำงานขึ้นอยู่กับจำนวณรอบที่ต้องทำงาน

- การป้อนค่า RA , RB และค่าที่ใช้ในการควบคุม AIC ค่าอื่น ๆ สามารถทำได้โดยการส่งค่าผ่านทาง รีจิสเตอร์ DXR ไปยัง AIC โดยการเปลี่ยนแปลงค่าใน 2 LSB 's จากข้อมูล 16 บิตจากค่าเดิมที่ใช้ในการส่งข้อมูลทั่ว ๆ ไปที่ต้องเป็น 00 ให้เป็นค่าอื่นคือ 01 , 10 หรือ 11 จะทำให้ AIC รับรู้ว่ามีความต้องการในการปรับเปลี่ยนค่าในการทำงานของ AIC หลังจากนั้นจึงส่งค่าของ RA , RB หรือค่าอื่น ๆ ออกไปทาง DXR เมื่อ AIC ได้รับค่าแล้วก็จะทำการปรับเปลี่ยนค่าให้ตามที่ต้องการ จึงเป็นข้อสังเกตว่าในการรับส่งข้อมูลตามปกติ AIC จะคาดหวังว่า 2 LSB 's จากข้อมูล 16 บิตที่ส่งมาจาก DXR จะต้องมีค่าเป็น 00 เสมอ ดังนั้นข้อมูลที่อยู่ใน DXR จะต้องถูกบังคับให้ 2 บิตสุดท้ายเป็น 00 มิฉะนั้นจะทำให้ AIC คิดว่าจะมีการปรับเปลี่ยนค่าในการทำงานของตัวมัน และส่งผลให้มีการทำงานในแบบที่ไม่ได้ตั้งใจเกิดขึ้นได้ วิธีการที่ใช้เพื่อป้องกันกรณีนี้คือค่าที่อยู่ที่แอคคิวมิวเลเตอร์ (ACC) จะถูกทำให้ 2 บิตสุดท้ายเป็น 00 ก่อนที่จะเก็บค่าลงที่ DXR ด้วยชุดคำสั่งต่อไปนี้

```
LACC    OUTPUT    ; load output into ACC
AND     #0FFFCh   ; And  low ACC with 1111 1111 1111 1100b
SAMM    DXR       ; store low  ACC  into  DXR
```

รายละเอียดนอกเหนือจากนี้สามารถอ่านได้ใน [5] และ [6]

## ภาคผนวก ค

### การกำหนดค่าเริ่มต้นให้แก่ AIC ( AIC Initialization )

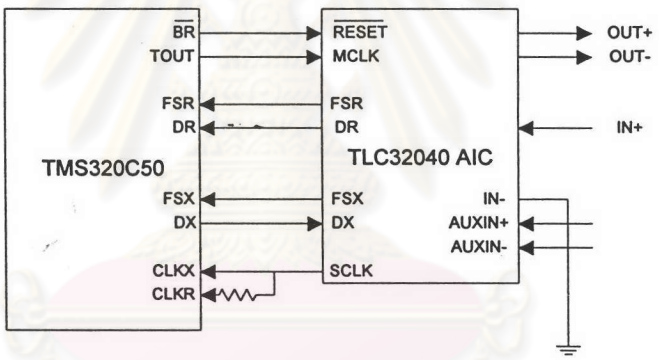
การกำหนดค่าเริ่มต้นให้แก่ AIC นั้นมีความสำคัญมากสำหรับการทำงานบน DSK ซึ่งจำเป็นต้องทำก่อนการทำงานทุกครั้งถ้าต้องมีการติดต่อกับสัญญาณอนาลอกภายนอก การกำหนดค่าเริ่มต้นนั้นสามารถทำได้ด้วยวิธีการทางซอฟท์แวร์ โดยมีสิ่งที่ต้องการการกำหนดค่าเริ่มต้นที่สำคัญ 3 อย่างดังนี้

1) สัญญาณนาฬิกาบนชิปของ TMS320C50 ( TMS320C50 on-chip timer )

2) พอร์ตอนุกรมของ 'C5x ( 'C5x serial port )

3) วงจรเชื่อมต่อสัญญาณอนาลอก ( AIC )

AIC คืออุปกรณ์ที่เชื่อมต่อรูปแบบสัญญาณอนาลอกจากภายนอกเข้ากับการจัดการสัญญาณแบบดิจิตอลภายในตัว DSK โดยรูปแบบการต่อจะเป็นดังรูป ค.1




รูป ค.1 การติดต่อแบบฮาร์ดแวร์ระหว่าง 'C50 และ AIC

**สัญญาณนาฬิกาบนชิปของ TMS320C50**

จากรูป ค.1 ค่า MASTER CLOCK (MCLK) ของ AIC ได้มาจากค่า TOUT ใน 'C50 จาก[5]หน้า 5-45 บอกว่าการกำหนดค่าของ TOUT จะได้ค่าสูงสุดเป็นครึ่งหนึ่งของค่าเวลาในการทำงานหนึ่งคำสั่งของ 'C50 นั่นคือ 10 เมกะเฮิรตซ์ โดยมีคำสั่งที่ต้องเขียนดังนี้

```
SPLK    #01h,PRD    ; Load PRD reg. for period of 100 ns. TDDR=0
SPLK    #20h,TCR    ; RE-load and begin Timer
```

**พอร์ตอนุกรมของ 'C5x**

การกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรมของ 'C5x ต้องดัดแปลงค่าของรีจิสเตอร์ควบคุมพอร์ตอนุกรม( Serial-Port-Control register หรือ SPC ) มีรายละเอียดของรีจิสเตอร์ควบคุมพอร์ตอนุกรมใน [5] หน้า 5-18 โดยมีคำสั่งที่ต้องเขียนดังนี้

```
SPLK    #08h,SPC    ; FSM=1, XRST and RRST = 00
SPLK    #c8h,SPC    ; FSM=1, XRST and RRST = 11
```

**วงจรเชื่อมต่อสัญญาณอนาลอก( AIC )**

เมื่อตั้งค่าของ MCLK และกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรมเสร็จเรียบร้อยแล้ว ขั้นต่อไปคือการรีเซ็ต AIC เพื่อให้ AIC ไปอยู่ในสถานะที่รู้จัก ขารีเซ็ตของ AIC อยู่ติดกับขา BR ของ 'C50 ซึ่งการทำให้ขา BR ทำงานต้องมีการเข้าถึงข้อมูลแบบโกลบอลภายนอก( external global memory access ) มีรายละเอียดใน [5] หน้า 6-29 โดยมีคำสั่งที่ต้องเขียนดังนี้

```
LACC    #80h        ; init 8000h-FFFFh as Global memory
SACL    GREG        ; Store to Global memory allocation reg.
LAR     AR0,#0FFFFh ; Use AR0 to point to location FFFFh
RPT     #10000      ; Access Global memory 10,000 times
LACC    *,0,AR0     ; to drive pin low for duration
SACH    GREG        ; restore GREG to 0000
```

เมื่อรีเซ็ตค่าของ AIC แล้วต่อไปคือการกำหนดค่าในการทำงานของ AIC ซึ่งขึ้นอยู่กับสัญญาณนาฬิกาต่าง ๆ ตัวกำเนิดสัญญาณนาฬิกา (Crystal Oscillator) ที่มีอยู่ในบอร์ด DSK มีค่าความถี่เท่ากับ 40 เมกะเฮิรตซ์ จะเป็นตัวกำหนดค่าของสัญญาณนาฬิกาอื่น ๆ ที่ใช้ในการประมวลผลของ C50 และตัว AIC สัญญาณนาฬิกาที่สำคัญที่ใช้ในตัว AIC มีดังนี้

- MASTER CLOCK (MCLK) มีค่าเท่ากับ 40/4 หรือ 10 เมกะเฮิรตซ์ เป็นสัญญาณนาฬิกาที่สำคัญในการกำหนดค่าของสัญญาณนาฬิกาตัวอื่น

- SHIFT CLOCK มีค่าเท่ากับ (MCLK)/4 เท่ากับ 2.5 เมกะเฮิรตซ์ ใช้เป็นสัญญาณนาฬิกาในการรับส่งข้อมูลแบบอนุกรมของ AIC กับตัวประมวลผลสัญญาณ

- SWITCHED CAPACITOR FILTER CLOCK (SCF CLK) ใช้ในการกำหนดค่าผลตอบของตัวกรองผ่านความถี่ต่ำในด้านขาออกและตัวกรองผ่านแถบความถี่ในด้านขาเข้า ถ้า SCF CLK มีค่าเท่ากับ 288 กิโลเฮิรตซ์ผลตอบจะได้ตรงตามรูป...แต่ถ้า SCF CLK ไม่เท่ากับ 288 กิโลเฮิรตซ์ผลตอบของตัวกรองก็จะเลื่อนขึ้นลงไปตามอัตราส่วนของค่า SCF CLK ต่อค่า 288 กิโลเฮิรตซ์ โดยสามารถคำนวณค่า SCF CLK ได้ดังนี้

SCF CLK = MCLK / (2*Tx counter A) สำหรับด้านขาออก

SCF CLK = MCLK / (2*Rx counter A) สำหรับด้านขาเข้า

- ความถี่ในการแปลงค่าอนาลอกเป็นดิจิตอล (A/D Conversion frequency) = SCF CLK / Rx counter B

- ความถี่ในการแปลงค่าดิจิตอลเป็นอนาลอก (D/A Conversion frequency) = SCF CLK / Tx counter B

**รูปแบบการรับส่งข้อมูลของ AIC**

ความละเอียดของการแปลงข้อมูลของ AIC ระหว่างอนาลอกและดิจิตอลมีค่าเท่ากับ 14 บิต ในขณะที่รีจิสเตอร์ DRR และ DXR ที่ใช้ในการรับส่งค่ามีความละเอียด 16 บิต จึงเหลือตัวเลขอยู่ 2 บิตที่ไม่ได้ใช้ในการรับส่งข้อมูล 2 บิตนี้คือ 2 บิตที่มีความสำคัญน้อยที่สุด (Least Significant Bit) แสดงดังในรูป ค.2

MSB LSB

| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

|<---------- Used bits ----------->| Not used |

รูป ค.2 รูปแบบการเก็บข้อมูลแบบ 16 บิตในการรับส่งข้อมูลของ AIC

AIC ใช้ 2 บิตที่กล่าวถึงนี้ ($d_1$ และ $d_0$) ในการควบคุมการทำงานของ AIC ในการรับส่งค่าตามปกติค่าของ $d_1d_0$ ควรจะเป็น 00 ค่า Rx counter A = RA ค่า Tx counter A = TA ค่า Rx counter B= RB ค่า Tx counter B = TB เมื่อค่าใน $d_1d_0$ ไม่เป็น 00 AIC จะมีการเปลี่ยนแปลงดังนี้

$d_1d_0$ = 01 ค่า Rx counter A = RA + RA' ค่า Tx counter A = TA + TA' ค่า Rx counter B= RB ค่า Tx counter B = TB

$d_1d_0$ = 10 ค่า Rx counter A = RA - RA' ค่า Tx counter A = TA - TA' ค่า Rx counter B= RB ค่า Tx counter B = TB

$d_1d_0$ = 11 ค่า Rx counter A,B และค่า Tx counter A,B จะเท่ากับค่าเดิมตามที่กำหนดไว้ก่อนหน้านี้ และหลังจากที่ SHIFT CLK ผ่านไปได้ 4 รอบจะมีการส่งค่ารอบที่ 2 ตามมาเพื่อเปลี่ยนค่าของรีจิสเตอร์ต่าง ๆ ของตัว AIC รูปแบบของการส่งค่ารอบที่ 2 แสดงดังในรูป ค.3

| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | หมายเหตุ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $d_{15}$ | $d_{14}$ | TA | | | | | $d_8$ | $d_7$ | RA | | | | | 0 | 0 | TA,RA ขนาด 5 บิต |
| $d_{15}$ | TA' | | | | | | $d_8$ | RA' | | | | | | 0 | 1 | TA',RA' ขนาด 6 บิต |
| $d_{15}$ | TB | | | | | | $d_8$ | RB | | | | | | 1 | 0 | TB,RB ขนาด 6 บิต |
| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | control bits | | | | | | 1 | 1 | บิตควบคุมขนาด 6 บิต |
| control bits = $b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$ | | | | | | | | | | | | | | | | |
| $b_0$ = 0/1 | | | | | | | | | | | | | | | | ไม่ใช้/ใช้ ตัวกรองผ่านแถบความถี่ |
| $b_1$ = 0/1 | | | | | | | | | | | | | | | | ไม่ใช้/ใช้ ฟังก์ชันลูปแบก |
| $b_2$ = 0/1 | | | | | | | | | | | | | | | | ไม่ใช้/ใช้ ขาป้อนข้อมูลช่วย |
| $b_3$ = 0/1 | | | | | | | | | | | | | | | | ไม่ใช้/ใช้ การทำงานแบบเข้าจังหวะ |
| $b_5b_4$ = 00/01/10/11 | | | | | | | | | | | | | | | | อัตราขยายด้านขาเข้าเท่ากับ 1/2/4/1 |

รูป ค.3 รูปแบบการเก็บข้อมูลแบบ 16 บิตในการรับส่งข้อมูลครั้งที่ 2 ของ AIC

จากรูปจะเห็นว่าผู้ใช้สามารถตั้งค่าของรีจิสเตอร์ RA RA' RB TA TA' TB และค่าที่ใช้ในการควบคุมการทำงานของ AIC ตัวอื่น ๆ ได้ตามที่ต้องการในการส่งค่ารอบที่ 2 นี้เอง

ขั้นตอนในการตั้งค่าเริ่มต้นให้แก่ AIC

1) กำหนดค่าของรีจิสเตอร์ต่าง ๆ และค่าบิตควบคุมของ AIC ไว้ในหน่วยความจำ

2) ตั้งให้อินเตอรัปต์ไม่สามารถทำงานได้ (Disable interrupt) แล้วจัดค่าให้ข้อมูลที่ส่งไปยัง AIC มีค่า $d_1d_0$ = 11 แล้วจึงทำให้อินเตอรัปต์ทำงานได้และใช้คำสั่ง Idle รอจนกว่าจะมีอินเตอรัปต์เข้ามา

3) เมื่อมีอินเตอรัปเข้ามาจึงดึงค่าของรีจิสเตอร์ต่าง ๆ และค่าบิตควบคุมจากหน่วยความจำมาใส่ไว้ในค่าที่จะส่งไปยัง AIC โดยตั้งค่าให้ถูกต้องว่าจะส่งไปยังรีจิสเตอร์ตัวใดเพราะในการส่งค่าแต่ละครั้งนั้นจะส่งค่าไปยังรีจิสเตอร์ได้เพียง 2 ตัวเป็นอย่างมากเท่านั้นและใช้คำสั่ง Idle เพื่อรอจนกว่าจะมีอินเตอรัปต์เข้ามา ทำซ้ำข้อ 3) นี้ไปจนกว่าจะส่งค่าไปยังรีจิสเตอร์ได้ครบตามที่ต้องการ

ในการทดลองนี้เป็นการทำงานกับสัญญาณเสียงที่มีแถบความถี่กว้างไม่เกิน 4 กิโลเฮิรตซ์ ดังนั้นจึงต้องใช้อัตราการสุ่มข้อมูล 8 กิโลเฮิรตซ์ ตัวอย่างการเขียนโปรแกรมโดยให้ค่า RA , RB ที่ใช้เท่ากับ 17 และ 37 ตามลำดับทำให้ได้อัตราการสุ่มข้อมูลที่ A/D เท่ากับ 8.0 กิโลเฮิรตซ์และเลือกใช้การทำงานเข้าจังหวะกันระหว่างด้านขาเข้าและด้านขาออก ทำให้ได้อัตราการสุ่มข้อมูลที่ D/A เท่ากับ 8.0 กิโลเฮิรตซ์ด้วย ตัวอย่างการเขียนโปรแกรมดังกล่าวคือโปรแกรม AIC_INI.INC ซึ่งแสดงดังต่อไปนี้ รายละเอียดนอกจากนี้อ่านได้ในคู่มือการทำงานของ DSK [6]

```
*****************************************************************
*       AIC_INI.INC                                             *
*  DESCRIPTION: This routine initializes the TLC320C40 for      *
*       a 8.0 Khz sample rate with a gain setting of 1          *
*****************************************************************
        .data
TA      .word   17              ; Fcut = 4 KHz
RA      .word   17              ; Fcut = 4 KHz
TAp     .word   31              ;
RAp     .word   31              ;
TB      .word   37              ; Fs = 2*Fcut
RB      .word   37              ; Fs = 2*Fcut
AIC_CTR  .word 19h              ; Use Band Pass Filter
        .text
START   SETC    INTM            ; Disable interrupts
        LDP     #0              ; Set data page pointer
        OPL     #0836h,PMST
        LACC    #0
        SAMM    CWSR            ; Set software wait state to 0
        SAMM    PDWSR           ;
        SPLK    #012h,IMR       ; Using XINT syn TX & RX
        CALL    AICINIT         ; initialize AIC and enable interrupts
        ...
AICINIT:        SPLK    #20h,TCR        ; To generate 10 MHz from Tout
        SPLK    #01h,PRD        ; for AIC master clock
        MAR     *,AR0
        LACC    #0008h          ; Non continuous mode
        SACL    SPC             ; FSX as input
        LACC    #00c8h          ; 16 bit words
        SACL    SPC
        LACC    #080h           ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; and taking it high after 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG
        LDP     #TA             ;
        SETC    SXM             ;
        LACC    TA,9            ; Initialized TA and RA register
        ADD     RA,2            ;
        CALL    AIC_2ND         ; call subroutine
        LDP     #TB
        LACC    TB,9            ; Initialized TB and RB register
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ; call subroutine
        LDP     #AIC_CTR
        LACC    AIC_CTR,2       ; Initialized control register
        ADD     #03h            ;
        CALL    AIC_2ND         ; call subroutine
        RET                     ;
AIC_2ND:
        LDP     #0
        SACH    DXR             ;
        CLRC    INTM
        IDLE
        ADD     #6h,15          ; 0000 0000 0000 0011 xxxx xxxx xxxx xxxx b
        SACH    DXR             ;
        IDLE
        SACL    DXR             ;
        IDLE
        LACL    #0              ;
        SACL    DXR             ; make sure the word got sent
        IDLE
        SETC    INTM
        RET                     ;
```

# ภาคผนวก ง

## ตัวอย่างโปรแกรม

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%program convert Wave file of 11 kHz sampling rate TO 8kHz sampling
%           W11T08.M                                %
% running on MATLAB VERSION 4.2c.1 with Signal Processing Toolbox%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('Convert "11"kHz-sampling wavefile to "8"kHz-sampling
wavefile ');
file=input('What wave file do you want to convert (*.wav) ? ','s');
                                                        % input
file name of 11 kHz wavefile
xx=fopen(file,'rb');
h1=fread(xx,40);
h2=fread( xx, 1, 'long' );
qu=h2/2;
if qu-fix(qu/5)*5~=0;
   h3=fread(xx,qu-fix(qu/5)*5,'short');
end;
lp=fix(qu/5);
vector=0;
vector(1:5*lp)=fread(xx,5*lp,'short');
v=interp(vector,8);                                     %
resampling wavefile to be 88.2 sampling rate
d=decimate(v,11);                                       %
resampling wavefile again to be 8.017 sampling rate
b=[18770 17990 8226 0 16727 17750 28006 8308 16 0 1 1 8000 0 16000
0 2 16 24932 24948];
s=length(d);
b(3)=2*s+36;
e=2*s;
wavename=input('input  wavename','s');          % input file name
of 8 kHz wavefile
a=fopen(wavename,'w');
fwrite(a,b,'short');
fwrite(a,e,'long');
fwrite(a,d,'short');
fclose(a);fclose(xx);                  % End of program W11T08.M


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           program to convert Decimal vector into Hexadecimal
Data file
%                       D2H.M                             %
%           Dec vector is in name "input"  and  |input(i)|<32768
for
%           i=1:length(input)                       %
%           Hex file is in name "in1.dat"
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a=fopen('in1.dat','w');
b=length(input)
for i=1:b;
            d=input(i);
            if d<0;d=65536+d;end;
            in=dec2hex(d);
            fprintf(a,'%s\n',in);
end;
fclose(a);              % End of program D2H.M


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program to Convert Hexadecimal Data file into Decimal vector
%           %
%                       H2D.M                             %
%           Hex file name can be entered by user
%
%           Dec vector is in name "out"
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

out=0;
file=input('Enter file (.dat) to convert to decimal ', 's');
aa=fopen(file);
bb=1;cc=1;
bb=fscanf(aa,'%s',1);
while bb;
          out(cc)=hex2dec(bb);
          if out(cc)>32767; out(cc)=out(cc)-65536;end;
          cc=cc+1;
          bb=fscanf(aa,'%s',1);
end;
fclose(aa);             % End of program H2D.M


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           COMMAND FILE   for Encoder of G.728
%           %
%                       ENCODE.CMD
%                       %
%           ENTRY       must be Global variable define in other
program     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
-e ENTRY
-m encode.map
MEMORY
{
        PAGE 0 :
                RAM                     : origin = 800h ,length = 2400h
        PAGE 1 :
                RAM                     : origin =   0h ,length =   80h
                RAMB    : origin =  100h , length =  400h
                DATA    : origin = 0a00h , length = 2200h
}
SECTIONS
{
            vector      822h        : { }   PAGE 0
            .text       0a00h       : { }   PAGE 0
            .data       2000h       : { }   PAGE 1
            ramb         100h       : { }   PAGE 1
            ytable      1800h       : { }   PAGE 1
}                       %           End of ENCODE.CMD


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulator Command File          %
%           INIT.CMD              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If you DONT WANT to use the SERIAL PORTS use the following map%

ma 0,0,0x2c00,ram          ; Allocate PROGRAM MEMORY    0 - 0xffff

;ma 0,1,0x80,ram           ; Allocate DATA MEMORY       0 - 0x7f
                           ; RESERVED DATA MEMORY     0x80 - 0xff
ma 0x100,1,0x400,ram       ; Allocate DATA MEMORY    0x100 - 0x4ff
                           ; RESERVED DATA MEMORY    0x500 - 0x7ff
ma 0x800,1,0x2400,ram      ; Allocate DATA MEMORY    0x800 - 0xffff

; To use the 2 SERIAL PORTS use the following instead

;ma 0,0,0x10000,ram        ; Allocate PROGRAM MEMORY    0 - 0xffff
ma 0,1,0x20,ram            ; Allocate DATA MEMORY       0 - 0x20
ma 0x20,1,1,IPORT          ; Configure DRR(0x20) as Input Port
ma 0x21,1,1,OPORT          ; Configure DXR(0x21) as Output Port
ma 0x22,1,0xe,ram          ; Allocate DATA MEMORY    0x22 - 0x2f
ma 0x30,1,1,IPORT          ; Configure TRCV(0x30) as Input Port
ma 0x31,1,1,OPORT          ; Configure TDXR(0x31) as Output Port
ma 0x32,1,0x4e,ram         ; Allocate DATA MEMORY    0x32 - 0x7f
                           ; RESERVED DATA MEMORY     0x80 - 0xff
;ma 0x100,1,0x400,ram      ; Allocate DATA MEMORY    0x100 - 0x4ff
                           ; RESERVED DATA MEMORY    0x500 - 0x7ff
;ma 0x800,1,0x1000,ram     ; Allocate DATA MEMORY    0x800 - 0xffff

; Sample Connect commands to connect Input and Output Files to the
; Serial Ports
 mc 0x20,1,in.dat,read
 mc 0x21,1,out.dat,write
 mc 0x30,1,in2.dat,read
 mc 0x31,1,out2.dat,write

e *0x28=0
e *0x29=0
e *0x2a=0
                           % End of INIT.CMD


;*******************************************************
;
;       File:           TENCODE.ASM
;       Date:           August 4, 1996
;       Author:         Poonlap Lamsichan
;
;       Purpose:    Test A/D and D/A by receive input from A/D store
;                   in SV ( CIRCULAR BUFFER )
;                   then  send to ST and send out to D/A
;                   program is in main program  and receive
;                   interrupt service routine
;                   include encode.asm
;                   Sampling rate less than 8kHz (about 7.2 kHz)
;COPYRIGHT (C) 1996 COMM LAB., Chulalongkorn University Thailand
;-------------------------------------------------------
        .version 50
        .mmregs
        .global START
*
;-------------------------------------------------------------;
; Bit definition of the control register in the AIC           ;
;-------------------------------------------------------------;
      ;+-------------------------+
      ;|LP xx G1 G0 | SY AX LB BP|              G1  G0  gain
      ;+------------+------------+             ---------------
      ; |      GAIN     |  |  |  +-- BP Filter    0   0    4
      ; |      Synch --+ |  |  +----- Loopback     1   1    4
      ; |      Auxin -----+                       0   1    2
      ; + (sinx)/x filter                        1   0    1
```

```
        .data
TA      .word   18              ; Fcut = 4 KHz
RA      .word   18              ; Fcut = 4 KHz
TAp     .word   31              ;
RAp     .word   31              ;
TB      .word   39              ; Fs = 2*Fcut
RB      .word   39              ; Fs = 2*Fcut
AIC_CTR .word   18h             ; No use Band Pass Filter
;AIC_CTR  .word   19h           ; Use Band Pass Filter
*****************************************************************
*   Set up the ISR vector                                       *
*****************************************************************
        .sect   "vector"
rint:   B       RECEIVE  ;0A; Serial prot receive interrupt RINT.
xint:   B       TRANSMIT ;0C; Serial port transmit interrupt XINT.
;---------------------------------------------------------------
*
*****************************************************************
* TMS32C05X INITIALIZATION                                      *
*                                                               *
*****************************************************************
       .text
START:  SETC    INTM            ; Disable interrupts
        LDP     #0              ; Set data page pointer
        OPL     #0836h,PMST
        LACC    #0
        SAMM    CWSR            ; Set software wait state to 0
        SAMM    PDWSR           ;

* Reset AIC by writing to PA2 (address >52) on EVM

        SPLK    #022h,IMR   ; Using XINT syn TX & RX
        CALL    AICINIT     ; initialize AIC and enable interrupts
*
*****************************************************************
* This routine enables serial port rx interrupts & configures   *
* TLC32046 due to the frame sync. When RINT generate read a dummy *
* data from the AIC then generate a sine wave to send out.      *
*****************************************************************
;
        SETC    OVM             ; OVM = 1
        SPM     0               ; PM = 0
        SPLK    #012h,IMR
        CLRC    INTM            ; enable interrupt
        SETC    SXM
        MAR     *,AR1
;
;---------------------------------------------------------------;
;       Main program start here                                 ;
;---------------------------------------------------------------;

                                ; wait for a transmit or receiwe
WAIT    MAR     *,AR1
        LDP     #SVPTR
WAIT2   CPL     #SV,SVPTR
        BCND    WAIT2,NTC
        .include encoder.asm        ; include encode part
        B       WAIT
;------- end of main program ---------------------------------;

;---------------------------------------------------------
; RECIEVER INTERRUPT SERVICE ROUTINE
;---------------------------------------------------------
RECEIVE:
        .include        receive.inc
        RETE


;---------------------------------------------------------
; TRANSMIT INTERRUPT SERVICE ROUTINE
;---------------------------------------------------------
TRANSMIT:
        RETE

;---------------------------------------------------------------;
;       include table of exp2() and log2() functions            ;
;---------------------------------------------------------------;
        .include        bexp2.inc
        .include        blog2.inc
        .include        pm.inc
;---------------------------------------------------------------;
;---------------------------------------------------------
; Include AIC Initialization routine
;---------------------------------------------------------

        .include aic_ini.inc

;---------------------------------------------------------


;---------------------------------------------------------
; Include Data memory and Variables Initialialization routine
;---------------------------------------------------------
         .include enc2_ini.inc
;---------------------------------------------------------
;-----------------
; End of program
;-----------------
        .end


;================================================
;  bexp2.inc       Exp2() : return exp2(ACCHI)
;  assume a 16 bits number, 0 < x < 7FFFh
;  there are up to 15 sign bits in x
;  NORM 14 times to remove upto 14 sign bits
;
;  return with :
;       ACCL = exp2(x) in Q5
;================================================
```

```
        .text
;  exp2(beta) in Q14    ;index      beta        exp2(beta)
;
exp2_tbl:
.word       16406       ;0          0.001953    1.001355
.word       16451       ;1          0.005859    1.004070
.word       16495       ;2          0.009766    1.006792
.word       16540       ;3          0.013672    1.009522
.word       16585       ;4          0.017578    1.012259
.word       16630       ;5          0.021484    1.015003
.word       16675       ;6          0.025391    1.017755
.word       16720       ;7          0.029297    1.020515
.word       16765       ;8          0.033203    1.023282
.word       16811       ;9          0.037109    1.026056
.word       16856       ;10         0.041016    1.028838
.word       16902       ;11         0.044922    1.031627
.word       16948       ;12         0.048828    1.034424
.word       16994       ;13         0.052734    1.037229
.word       17040       ;14         0.056641    1.040041
.word       17086       ;15         0.060547    1.042861
.word       17133       ;16         0.064453    1.045688
.word       17179       ;17         0.068359    1.048524
.word       17226       ;18         0.072266    1.051366
.word       17272       ;19         0.076172    1.054217
.word       17319       ;20         0.080078    1.057075
.word       17366       ;21         0.083984    1.059941
.word       17413       ;22         0.087891    1.062815
.word       17460       ;23         0.091797    1.065697
.word       17508       ;24         0.095703    1.068586
.word       17555       ;25         0.099609    1.071483
.word       17603       ;26         0.103516    1.074388
.word       17651       ;27         0.107422    1.077301
.word       17698       ;28         0.111328    1.080222
.word       17746       ;29         0.115234    1.083151
.word       17794       ;30         0.119141    1.086088
.word       17843       ;31         0.123047    1.089032
.word       17891       ;32         0.126953    1.091985
.word       17940       ;33         0.130859    1.094946
.word       17988       ;34         0.134766    1.097914
.word       18037       ;35         0.138672    1.100891
.word       18086       ;36         0.142578    1.103876
.word       18135       ;37         0.146484    1.106869
.word       18184       ;38         0.150391    1.109870
.word       18233       ;39         0.154297    1.112879
.word       18283       ;40         0.158203    1.115896
.word       18332       ;41         0.162109    1.118922
.word       18382       ;42         0.166016    1.121956
.word       18432       ;43         0.169922    1.124998
.word       18482       ;44         0.173828    1.128048
.word       18532       ;45         0.177734    1.131106
.word       18582       ;46         0.181641    1.134173
.word       18633       ;47         0.185547    1.137248
.word       18683       ;48         0.189453    1.140331
.word       18734       ;49         0.193359    1.143423
.word       18785       ;50         0.197266    1.146523
.word       18836       ;51         0.201172    1.149632
.word       18887       ;52         0.205078    1.152749
.word       18938       ;53         0.208984    1.155874
.word       18989       ;54         0.212891    1.159008
.word       19041       ;55         0.216797    1.162150
.word       19092       ;56         0.220703    1.165301
.word       19144       ;57         0.224609    1.168461
.word       19196       ;58         0.228516    1.171629
.word       19248       ;59         0.232422    1.174805
.word       19300       ;60         0.236328    1.177991
.word       19353       ;61         0.240234    1.181185
.word       19405       ;62         0.244141    1.184387
.word       19458       ;63         0.248047    1.187598
.word       19510       ;64         0.251953    1.190818
.word       19563       ;65         0.255859    1.194047
.word       19616       ;66         0.259766    1.197284
.word       19669       ;67         0.263672    1.200530
.word       19723       ;68         0.267578    1.203785
.word       19776       ;69         0.271484    1.207049
.word       19830       ;70         0.275391    1.210322
.word       19884       ;71         0.279297    1.213603
.word       19938       ;72         0.283203    1.216894
.word       19992       ;73         0.287109    1.220193
.word       20046       ;74         0.291016    1.223501
.word       20100       ;75         0.294922    1.226819
.word       20155       ;76         0.298828    1.230145
.word       20209       ;77         0.302734    1.233480
.word       20264       ;78         0.306641    1.236824
.word       20319       ;79         0.310547    1.240178
.word       20374       ;80         0.314453    1.243540
.word       20429       ;81         0.318359    1.246912
.word       20485       ;82         0.322266    1.250292
.word       20540       ;83         0.326172    1.253682
.word       20596       ;84         0.330078    1.257081
.word       20652       ;85         0.333984    1.260490
.word       20708       ;86         0.337891    1.263907
.word       20764       ;87         0.341797    1.267334
.word       20820       ;88         0.345703    1.270770
.word       20877       ;89         0.349609    1.274216
.word       20933       ;90         0.353516    1.277670
.word       20990       ;91         0.357422    1.281134
.word       21047       ;92         0.361328    1.284608
.word       21104       ;93         0.365234    1.288091
.word       21161       ;94         0.369141    1.291583
.word       21219       ;95         0.373047    1.295085
.word       21276       ;96         0.376953    1.298596
.word       21334       ;97         0.380859    1.302117
.word       21392       ;98         0.384766    1.305648
.word       21450       ;99         0.388672    1.309188
.word       21508       ;100        0.392578    1.312737
.word       21566       ;101        0.396484    1.316296
.word       21625       ;102        0.400391    1.319865
.word       21683       ;103        0.404297    1.323444
.word       21742       ;104        0.408203    1.327032
```

```
.word   21801   ;105    0.412109    1.330630
.word   21860   ;106    0.416016    1.334238
.word   21919   ;107    0.419922    1.337855
.word   21979   ;108    0.423828    1.341482
.word   22038   ;109    0.427734    1.345120
.word   22098   ;110    0.431641    1.348767
.word   22158   ;111    0.435547    1.352423
.word   22218   ;112    0.439453    1.356090
.word   22278   ;113    0.443359    1.359767
.word   22339   ;114    0.447266    1.363454
.word   22399   ;115    0.451172    1.367150
.word   22460   ;116    0.455078    1.370857
.word   22521   ;117    0.458984    1.374574
.word   22582   ;118    0.462891    1.378301
.word   22643   ;119    0.466797    1.382038
.word   22705   ;120    0.470703    1.385785
.word   22766   ;121    0.474609    1.389542
.word   22828   ;122    0.478516    1.393309
.word   22890   ;123    0.482422    1.397087
.word   22952   ;124    0.486328    1.400875
.word   23014   ;125    0.490234    1.404673
.word   23077   ;126    0.494141    1.408482
.word   23139   ;127    0.498047    1.412300
.word   23202   ;128    0.501953    1.416129
.word   23265   ;129    0.505859    1.419969
.word   23328   ;130    0.509766    1.423819
.word   23391   ;131    0.513672    1.427679
.word   23455   ;132    0.517578    1.431550
.word   23518   ;133    0.521484    1.435431
.word   23582   ;134    0.525391    1.439323
.word   23646   ;135    0.529297    1.443226
.word   23710   ;136    0.533203    1.447139
.word   23774   ;137    0.537109    1.451062
.word   23839   ;138    0.541016    1.454996
.word   23903   ;139    0.544922    1.458941
.word   23968   ;140    0.548828    1.462897
.word   24033   ;141    0.552734    1.466863
.word   24098   ;142    0.556641    1.470840
.word   24164   ;143    0.560547    1.474828
.word   24229   ;144    0.564453    1.478827
.word   24295   ;145    0.568359    1.482836
.word   24361   ;146    0.572266    1.486857
.word   24427   ;147    0.576172    1.490888
.word   24493   ;148    0.580078    1.494930
.word   24559   ;149    0.583984    1.498983
.word   24626   ;150    0.587891    1.503048
.word   24693   ;151    0.591797    1.507123
.word   24760   ;152    0.595703    1.511209
.word   24827   ;153    0.599609    1.515306
.word   24894   ;154    0.603516    1.519415
.word   24962   ;155    0.607422    1.523534
.word   25029   ;156    0.611328    1.527665
.word   25097   ;157    0.615234    1.531807
.word   25165   ;158    0.619141    1.535960
.word   25233   ;159    0.623047    1.540124
.word   25302   ;160    0.626953    1.544300
.word   25370   ;161    0.630859    1.548487
.word   25439   ;162    0.634766    1.552685
.word   25508   ;163    0.638672    1.556895
.word   25577   ;164    0.642578    1.561116
.word   25647   ;165    0.646484    1.565349
.word   25716   ;166    0.650391    1.569593
.word   25786   ;167    0.654297    1.573849
.word   25856   ;168    0.658203    1.578116
.word   25926   ;169    0.662109    1.582395
.word   25996   ;170    0.666016    1.586685
.word   26067   ;171    0.669922    1.590987
.word   26137   ;172    0.673828    1.595300
.word   26208   ;173    0.677734    1.599626
.word   26279   ;174    0.681641    1.603963
.word   26351   ;175    0.685547    1.608312
.word   26422   ;176    0.689453    1.612672
.word   26494   ;177    0.693359    1.617044
.word   26565   ;178    0.697266    1.621429
.word   26638   ;179    0.701172    1.625825
.word   26710   ;180    0.705078    1.630233
.word   26782   ;181    0.708984    1.634653
.word   26855   ;182    0.712891    1.639085
.word   26928   ;183    0.716797    1.643529
.word   27001   ;184    0.720703    1.647985
.word   27074   ;185    0.724609    1.652453
.word   27147   ;186    0.728516    1.656933
.word   27221   ;187    0.732422    1.661426
.word   27295   ;188    0.736328    1.665930
.word   27369   ;189    0.740234    1.670447
.word   27443   ;190    0.744141    1.674976
.word   27517   ;191    0.748047    1.679518
.word   27592   ;192    0.751953    1.684071
.word   27667   ;193    0.755859    1.688637
.word   27742   ;194    0.759766    1.693216
.word   27817   ;195    0.763672    1.697806
.word   27892   ;196    0.767578    1.702410
.word   27968   ;197    0.771484    1.707025
.word   28044   ;198    0.775391    1.711653
.word   28120   ;199    0.779297    1.716294
.word   28196   ;200    0.783203    1.720948
.word   28272   ;201    0.787109    1.725614
.word   28349   ;202    0.791016    1.730292
.word   28426   ;203    0.794922    1.734983
.word   28503   ;204    0.798828    1.739687
.word   28580   ;205    0.802734    1.744404
.word   28658   ;206    0.806641    1.749134
.word   28736   ;207    0.810547    1.753876
.word   28813   ;208    0.814453    1.758631
.word   28892   ;209    0.818359    1.763400
.word   28970   ;210    0.822266    1.768181
.word   29048   ;211    0.826172    1.772975
.word   29127   ;212    0.830078    1.777782
.word   29206   ;213    0.833984    1.782602
.word   29285   ;214    0.837891    1.787435
.word   29365   ;215    0.841797    1.792281
.word   29444   ;216    0.845703    1.797140
.word   29524   ;217    0.849609    1.802013
.word   29604   ;218    0.853516    1.806899
.word   29684   ;219    0.857422    1.811798
.word   29765   ;220    0.861328    1.816710
.word   29846   ;221    0.865234    1.821636
.word   29927   ;222    0.869141    1.826575
.word   30008   ;223    0.873047    1.831527
.word   30089   ;224    0.876953    1.836493
.word   30171   ;225    0.880859    1.841472
.word   30252   ;226    0.884766    1.846465
.word   30334   ;227    0.888672    1.851471
.word   30417   ;228    0.892578    1.856491
.word   30499   ;229    0.896484    1.861524
.word   30582   ;230    0.900391    1.866571
.word   30665   ;231    0.904297    1.871632
.word   30748   ;232    0.908203    1.876707
.word   30831   ;233    0.912109    1.881795
.word   30915   ;234    0.916016    1.886897
.word   30999   ;235    0.919922    1.892013
.word   31083   ;236    0.923828    1.897143
.word   31167   ;237    0.927734    1.902286
.word   31252   ;238    0.931641    1.907444
.word   31336   ;239    0.935547    1.912616
.word   31421   ;240    0.939453    1.917801
.word   31506   ;241    0.943359    1.923001
.word   31592   ;242    0.947266    1.928215
.word   31678   ;243    0.951172    1.933443
.word   31763   ;244    0.955078    1.938685
.word   31850   ;245    0.958984    1.943941
.word   31936   ;246    0.962891    1.949211
.word   32022   ;247    0.966797    1.954496
.word   32109   ;248    0.970703    1.959796
.word   32196   ;249    0.974609    1.965109
.word   32284   ;250    0.978516    1.970437
.word   32371   ;251    0.982422    1.975779
.word   32459   ;252    0.986328    1.981136
.word   32547   ;253    0.990234    1.986508
.word   32635   ;254    0.994141    1.991894
.word   32724   ;255    0.998047    1.997294

;-----------------------------------------------
; blog2.inc Log2() : return log2(ACCHI)
;  assume a 16 bits number, 0 < x < 7FFFh
;  there are up to 15 sign bits in x
;  NORM 14 times to remove upto 14 sign bits
;
;  return with :
;       ACCL = log2(x) in Q8
;-----------------------------------------------
        .text
;  log2(beta) in Q10    ;index     beta          log2(beta)
;
log2_tbl:
.word   3               ;0          1.001953    0.002815
.word   9               ;1          1.005859    0.008429
.word   14              ;2          1.009766    0.014020
.word   20              ;3          1.013672    0.019591
.word   26              ;4          1.017578    0.025140
.word   31              ;5          1.021484    0.030667
.word   37              ;6          1.025391    0.036174
.word   43              ;7          1.029297    0.041659
.word   48              ;8          1.033203    0.047124
.word   54              ;9          1.037109    0.052568
.word   59              ;10         1.041016    0.057992
.word   65              ;11         1.044922    0.063395
.word   70              ;12         1.048828    0.068778
.word   76              ;13         1.052734    0.074141
.word   81              ;14         1.056641    0.079485
.word   87              ;15         1.060547    0.084808
.word   92              ;16         1.064453    0.090112
.word   98              ;17         1.068359    0.095397
.word   103             ;18         1.072266    0.100662
.word   108             ;19         1.076172    0.105909
.word   114             ;20         1.080078    0.111136
.word   119             ;21         1.083984    0.116344
.word   124             ;22         1.087891    0.121534
.word   130             ;23         1.091797    0.126704
.word   135             ;24         1.095703    0.131857
.word   140             ;25         1.099609    0.136991
.word   146             ;26         1.103516    0.142107
.word   151             ;27         1.107422    0.147205
.word   156             ;28         1.111328    0.152285
.word   161             ;29         1.115234    0.157347
.word   166             ;30         1.119141    0.162391
.word   171             ;31         1.123047    0.167418
.word   177             ;32         1.126953    0.172428
.word   182             ;33         1.130859    0.177420
.word   187             ;34         1.134766    0.182394
.word   192             ;35         1.138672    0.187352
.word   197             ;36         1.142578    0.192293
.word   202             ;37         1.146484    0.197217
.word   207             ;38         1.150391    0.202124
.word   212             ;39         1.154297    0.207014
.word   217             ;40         1.158203    0.211888
.word   222             ;41         1.162109    0.216746
.word   227             ;42         1.166016    0.221587
.word   232             ;43         1.169922    0.226412
.word   237             ;44         1.173828    0.231221
.word   242             ;45         1.177734    0.236014
.word   247             ;46         1.181641    0.240791
.word   251             ;47         1.185547    0.245553
.word   256             ;48         1.189453    0.250298
.word   261             ;49         1.193359    0.255029
.word   266             ;50         1.197266    0.259743
.word   271             ;51         1.201172    0.264443
.word   276             ;52         1.205078    0.269127
.word   280             ;53         1.208984    0.273796
.word   285             ;54         1.212891    0.278449
```

```
.word   290     ;55     1.216797    0.283088
.word   295     ;56     1.220703    0.287712
.word   299     ;57     1.224609    0.292322
.word   304     ;58     1.228516    0.296916
.word   309     ;59     1.232422    0.301496
.word   313     ;60     1.236328    0.306062
.word   318     ;61     1.240234    0.310613
.word   323     ;62     1.244141    0.315150
.word   327     ;63     1.248047    0.319672
.word   332     ;64     1.251953    0.324181
.word   337     ;65     1.255859    0.328675
.word   341     ;66     1.259766    0.333155
.word   346     ;67     1.263672    0.337622
.word   350     ;68     1.267578    0.342075
.word   355     ;69     1.271484    0.346514
.word   359     ;70     1.275391    0.350939
.word   364     ;71     1.279297    0.355351
.word   368     ;72     1.283203    0.359750
.word   373     ;73     1.287109    0.364135
.word   377     ;74     1.291016    0.368506
.word   382     ;75     1.294922    0.372865
.word   386     ;76     1.298828    0.377211
.word   391     ;77     1.302734    0.381543
.word   395     ;78     1.306641    0.385862
.word   400     ;79     1.310547    0.390169
.word   404     ;80     1.314453    0.394463
.word   408     ;81     1.318359    0.398744
.word   413     ;82     1.322266    0.403012
.word   417     ;83     1.326172    0.407268
.word   421     ;84     1.330078    0.411511
.word   426     ;85     1.333984    0.415742
.word   430     ;86     1.337891    0.419960
.word   434     ;87     1.341797    0.424166
.word   439     ;88     1.345703    0.428360
.word   443     ;89     1.349609    0.432542
.word   447     ;90     1.353516    0.436712
.word   451     ;91     1.357422    0.440869
.word   456     ;92     1.361328    0.445015
.word   460     ;93     1.365234    0.449149
.word   464     ;94     1.369141    0.453271
.word   468     ;95     1.373047    0.457381
.word   473     ;96     1.376953    0.461479
.word   477     ;97     1.380859    0.465566
.word   481     ;98     1.384766    0.469642
.word   485     ;99     1.388672    0.473706
.word   489     ;100    1.392578    0.477758
.word   493     ;101    1.396484    0.481799
.word   497     ;102    1.400391    0.485829
.word   502     ;103    1.404297    0.489848
.word   506     ;104    1.408203    0.493855
.word   510     ;105    1.412109    0.497852
.word   514     ;106    1.416016    0.501837
.word   518     ;107    1.419922    0.505812
.word   522     ;108    1.423828    0.509775
.word   526     ;109    1.427734    0.513728
.word   530     ;110    1.431641    0.517669
.word   534     ;111    1.435547    0.521600
.word   538     ;112    1.439453    0.525521
.word   542     ;113    1.443359    0.529431
.word   546     ;114    1.447266    0.533330
.word   550     ;115    1.451172    0.537218
.word   554     ;116    1.455078    0.541097
.word   558     ;117    1.458984    0.544964
.word   562     ;118    1.462891    0.548822
.word   566     ;119    1.466797    0.552669
.word   570     ;120    1.470703    0.556506
.word   574     ;121    1.474609    0.560333
.word   578     ;122    1.478516    0.564149
.word   582     ;123    1.482422    0.567956
.word   585     ;124    1.486328    0.571753
.word   589     ;125    1.490234    0.575539
.word   593     ;126    1.494141    0.579316
.word   597     ;127    1.498047    0.583083
.word   601     ;128    1.501953    0.586840
.word   605     ;129    1.505859    0.590587
.word   609     ;130    1.509766    0.594325
.word   612     ;131    1.513672    0.598053
.word   616     ;132    1.517578    0.601771
.word   620     ;133    1.521484    0.605480
.word   624     ;134    1.525391    0.609179
.word   628     ;135    1.529297    0.612868
.word   631     ;136    1.533203    0.616549
.word   635     ;137    1.537109    0.620220
.word   639     ;138    1.541016    0.623881
.word   643     ;139    1.544922    0.627534
.word   646     ;140    1.548828    0.631177
.word   650     ;141    1.552734    0.634811
.word   654     ;142    1.556641    0.638436
.word   657     ;143    1.560547    0.642052
.word   661     ;144    1.564453    0.645658
.word   665     ;145    1.568359    0.649256
.word   669     ;146    1.572266    0.652845
.word   672     ;147    1.576172    0.656425
.word   676     ;148    1.580078    0.659996
.word   679     ;149    1.583984    0.663558
.word   683     ;150    1.587891    0.667112
.word   687     ;151    1.591797    0.670656
.word   690     ;152    1.595703    0.674192
.word   694     ;153    1.599609    0.677720
.word   698     ;154    1.603516    0.681238
.word   701     ;155    1.607422    0.684749
.word   705     ;156    1.611328    0.688250
.word   708     ;157    1.615234    0.691744
.word   712     ;158    1.619141    0.695228
.word   715     ;159    1.623047    0.698705
.word   719     ;160    1.626953    0.702173
.word   723     ;161    1.630859    0.705632
.word   726     ;162    1.634766    0.709084
.word   730     ;163    1.638672    0.712527
.word   733     ;164    1.642578    0.715962
```

```
.word   737     ;165    1.646484    0.719389
.word   740     ;166    1.650391    0.722808
.word   744     ;167    1.654297    0.726218
.word   747     ;168    1.658203    0.729621
.word   751     ;169    1.662109    0.733015
.word   754     ;170    1.666016    0.736402
.word   758     ;171    1.669922    0.739781
.word   761     ;172    1.673828    0.743151
.word   764     ;173    1.677734    0.746514
.word   768     ;174    1.681641    0.749869
.word   771     ;175    1.685547    0.753217
.word   775     ;176    1.689453    0.756556
.word   778     ;177    1.693359    0.759888
.word   782     ;178    1.697266    0.763212
.word   785     ;179    1.701172    0.766529
.word   788     ;180    1.705078    0.769838
.word   792     ;181    1.708984    0.773139
.word   795     ;182    1.712891    0.776433
.word   798     ;183    1.716797    0.779719
.word   802     ;184    1.720703    0.782998
.word   805     ;185    1.724609    0.786270
.word   808     ;186    1.728516    0.789534
.word   812     ;187    1.732422    0.792790
.word   815     ;188    1.736328    0.796040
.word   818     ;189    1.740234    0.799282
.word   822     ;190    1.744141    0.802516
.word   825     ;191    1.748047    0.805744
.word   828     ;192    1.751953    0.808964
.word   832     ;193    1.755859    0.812177
.word   835     ;194    1.759766    0.815383
.word   838     ;195    1.763672    0.818582
.word   841     ;196    1.767578    0.821774
.word   845     ;197    1.771484    0.824959
.word   848     ;198    1.775391    0.828136
.word   851     ;199    1.779297    0.831307
.word   854     ;200    1.783203    0.834471
.word   858     ;201    1.787109    0.837628
.word   861     ;202    1.791016    0.840778
.word   864     ;203    1.794922    0.843921
.word   867     ;204    1.798828    0.847057
.word   871     ;205    1.802734    0.850187
.word   874     ;206    1.806641    0.853310
.word   877     ;207    1.810547    0.856426
.word   880     ;208    1.814453    0.859535
.word   883     ;209    1.818359    0.862637
.word   887     ;210    1.822266    0.865733
.word   890     ;211    1.826172    0.868823
word    893     ;212    1.830078    0.871905
.word   896     ;213    1.833984    0.874981
.word   899     ;214    1.837891    0.878051
.word   902     ;215    1.841797    0.881114
.word   905     ;216    1.845703    0.884171
.word   909     ;217    1.849609    0.887221
.word   912     ;218    1.853516    0.890264
.word   915     ;219    1.857422    0.893302
.word   918     ;220    1.861328    0.896332
.word   921     ;221    1.865234    0.899357
.word   924     ;222    1.869141    0.902375
.word   927     ;223    1.873047    0.905387
.word   930     ;224    1.876953    0.908393
.word   933     ;225    1.880859    0.911392
.word   936     ;226    1.884766    0.914385
.word   939     ;227    1.888672    0.917372
.word   942     ;228    1.892578    0.920353
.word   945     ;229    1.896484    0.923327
.word   949     ;230    1.900391    0.926296
.word   952     ;231    1.904297    0.929258
.word   955     ;232    1.908203    0.932215
.word   958     ;233    1.912109    0.935165
.word   961     ;234    1.916016    0.938109
.word   964     ;235    1.919922    0.941048
.word   967     ;236    1.923828    0.943980
.word   970     ;237    1.927734    0.946906
.word   973     ;238    1.931641    0.949827
.word   976     ;239    1.935547    0.952741
.word   979     ;240    1.939453    0.955650
.word   982     ;241    1.943359    0.958553
.word   985     ;242    1.947266    0.961450
.word   987     ;243    1.951172    0.964341
.word   990     ;244    1.955078    0.967226
.word   993     ;245    1.958984    0.970106
.word   996     ;246    1.962891    0.972980
.word   999     ;247    1.966797    0.975848
.word   1002    ;248    1.970703    0.978710
.word   1005    ;249    1.974609    0.981567
.word   1008    ;250    1.978516    0.984418
.word   1011    ;251    1.982422    0.987264
.word   1014    ;252    1.986328    0.990104
.word   1017    ;253    1.990234    0.992938
.word   1020    ;254    1.994141    0.995767
.word   1023    ;255    1.998047    0.998590

;-------------------------------------------------;
; PM.INC        Assign program memory for variables section
;
;-------------------------------------------------;
        .text
PN      .word   0,0,0,0,0
H       .word   16384,0,0,0,0
WSPM    .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0
RTEMP   .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        .word   0,0,0,0,0,0,0,0,0,0
```

```
;================================================
;           RECEIVE.INC
;================================================
;-------------------------------------------------------;
;       receive part
;-------------------------------------------------------;
        LAMM    AR1             ; load AR1 to ACC and
        PUSH                    ; push it into stack ( save context )
        LMMR    AR1,#SVPTR      ; load SV pointer
        LACC    #SV+5           ;
        SAMM    ARCR            ; load end_of_SV_buffer+1 to ARCR
        MAR     *,AR1
        LAMM    DRR
        SACL    *+              ; store input value to SV buffer
        CMPR    0               ; compare whether AR1 equal to ARCR
        NOP                     ; no operation before XC
        XC      2,TC            ; if true
         LAR    AR1,#SV         ; load address of SV to AR1
        SMMR    AR1,#SVPTR      ; store AR1 back to SV pointer

;-------------------------------------------------------;
;       transmit part
;-------------------------------------------------------;

        LMMR    AR1,#STPTR      ; load ST pointer it AR1
        LACC    #ST+5           ;
        SAMM    ARCR            ; load end_of_ST_buffer+1 to ARCR
        LACC    *+              ; load value from ST buffer to low ACC
        AND     #0fffch         ; clear 2 LSBs of ACC for
        SAMM    DXR             ; sending to DXR
        CMPR    0               ; compare whether AR1 equal to ARCR
        NOP
        XC      2,TC            ; if true
         LAR    AR1,#ST         ; load address of ST to AR1
        SMMR    AR1,#STPTR      ; store AR1 back to ST pointer
        POP                     ; pop old AR1 out of stack
        SAMM    AR1             ; store it back to AR1
;        NOP
;        RETE                       ; program


;=================================================
;           block37.asm  Levinson-Durbin 10th order
;=================================================
; check whether ill-conditioned
        LACC    R+10            ; ACC=R(11)
        BCND    END37_3,EQ      ; if R(11)=0 goto end
        LACC    R
        BCND    END37_3,LEQ     ; if R(1)<=0 goto end

; if not ill-conditioned, continue

; MINC = 1
;
;       RC(1)=R(2)/R(1)
;
;-------------------------------; division of (NUMERA/DENOM)
        LACC    R+1,16          ;
        ABS                     ; ACCH = abs(NUMERA)
        RPT     #14
        SUBC    R               ; ACCL = abs(NUMERA/DENOM)
        BIT     R+1,0           ; test sign of NUMERA
        AND     TMP9            ; clear High ACC
        XC      1,TC
         NEG            ; if NUMERA is negative then negate results
;-------------------------------;
;
        SAMM    TREG0           ; TREG0 = RC(1) = R(2)/R(1)
        LAR     AR1,#AWZTMP+1
        BSAR    1
        SACL    *               ; AWZTMP(2) = RC(1)     in Q15
;
        SPM     0
        MPY     R+1             ; PREG = R(2)*RC(1) in Q2
        LACC    R,15            ; ACCH = R(1)
        APAC                    ; ACCH = R(2)*RC(1) + R(1) in Q-13
        SACH    DENOM           ; ALPHA = ACCH          in Q-14

        BCND    END37_3,LEQ     ; if ALPHA<=0 goto END

;-----------------------------------------------------------------
;       If ALPHA valids continue throught the rest of program
;
        SPM     1

        LAR     AR1,#R+1
        RPT     #9
        BLDD    *+,#WSPM        ; move R to program memory

        LAR     AR1,#(AWZTMP+1) ; AR2 = address of AWZTMP(3)

; MINC = 2, second iteration
;-------------------------------;
        RPTZ    #1              ;
        MAC     WSPM,*-         ;
;-------------------------------;
        APAC                    ; ACCH = AWZTMP(1:2)*R(3:2)'in Q-13
        ADRK    #3
;
;       calculates -(SUM/ALPHA)
;
        SACH    NUMERA          ; store SUM in NUMERA       in Q-14
;
;-------------------------------; division of -(NUMERA/DENOM)
        LACC    NUMERA,16       ;
        ABS                     ; ACCH = abs(NUMERA)
        RPT     #14
        SUBC    DENOM           ; ACCL = abs(NUMERA/DENOM)
        BIT     NUMERA,0        ; test sign of NUMERA
        AND     TMP9            ; clear High ACC
        XC      1,NTC
        NEG             ; if NUMERA is positive then negate results
;-------------------------------;
        SAMM    TREG0           ; TREG0 = RC(2) = -SUM/ALPHA
        BSAR    1
        SACL    *-              ; AWZTMP(3) = RC(2)
        MPY     *       ; PREG = RC(2)*AWZTMP(2)        in Q30
        LACC    *,16
        APAC
        SACH    *+      ; AWZTMP(2) = AWZTMP(2)+RC(2)*AWZTMP(2) Q15
        MPY     NUMERA          ; PREG = RC(2)*SUM          in Q2
        LACC    DENOM,16
        APAC
        SACH    DENOM           ; ALPHA = ALPHA + RC(2)*SUM in Q-13
        BCND    END37_3,LEQ     ; if ALPHA<=0 ill-conditioned occurs
                                ; goto END37_3

; MINC = 3,4,...,8,9,10  total 8 iterations using 8/2 = 4 loops

        LAR     AR0,#3          ; loop for 4 times
        LAR     AR3,#(AWZTMP+2) ; AR3=address of R(MINC+1) for the
                                ; first loop
        SPLK    #2,TMP1         ; TMP1=MINC-1= 2 for the first loop
        LAR     AR5,#(AWZTMP+3) ; AR5 = address of AWZTMP(MINC+1)
        LAR     AR4,#0          ; inner loop repeat 1 times for
                                ; first outer loop
        LAR     AR6,#3
;-------------------------------------------------------;
BEG37_1
        LAMM    AR3             ;
        SAMM    AR1             ; AR1 = address of R(MINC+1)
        ZAP
;-------------------------------;
        RPT     TMP1            ; TREG0 = R(MINC-IP+2) in Q-13
        MAC     WSPM,*-           ; PREG = AWZTMP*R         in Q2
;-------------------------------;
        APAC                    ; ACCH = SUM + AWZTMP*R   in Q-13
                                ; SUM=SUM + R(MINC-IP+2)*AWZTMP(IP)
                                ; in Q-13
;
;       calculates  -(SUM/ALPHA)
;
        SACH    NUMERA          ; store SUM to numerator    in Q-13
;
;-------------------------------; division of -(NUMERA/DENOM)
        LACC    NUMERA,16       ;
        ABS                     ; ACCH = abs(NUMERA)
        RPT     #14
        SUBC    DENOM           ; ACCL = abs(NUMERA/DENOM)
        BIT     NUMERA,0        ; test sign of NUMERA
        AND     TMP9            ; clear High ACC
        XC      1,NTC
        NEG                     ; if NUMERA is positive then negate
                                ; results
;-------------------------------;
;
        SAMM    TREG0           ; TREG0 = RC(MINC)

        SAR     AR6,TMP1        ; TMP1 = 3,5,7,9
        LMMR    BRCR,#AR4       ; inner loop run for 1,2,3,4 times
        LMMR    AR1,#AR5        ; AR1 = address of AWZTMP(IB)
        LAR     AR2,#(AWZTMP+1) ; AR2 = address of AWZTMP(IP)

        BSAR    1
        SACL    *-              ; AWZTMP(MINC+1) = RC(MINC)  in Q15
;-------------------------------;
        RPTB    END37_1
        MPY     *,AR2           ; PREG = RC(MINC)*AWZTMP(IB) in Q30
        LACC    *,16            ; ACCH = AWZTMP(IP)             in Q15
        MPYA    * ; ACCH = AWZTMP(IP)+RC(MINC)*AWZTMP(IB) in Q15
                                ; PREG = RC(MINC)*AWZTMP(IP) in Q30
        SACH *+,AR1;AWZTMP(IP)=AWZTMP(IP)+RC(MINC)*AWZTMP(IB)
        LACC    *,16            ; ACCH = AWZTMP(IB)             in Q15
        APAC                    ;
END37_1 SACH    *-      ; AWZTMP(IB)=AWZTMP(IB)+RC(MINC)*
                        ;AWZTMP(IP) Q15
;-------------------------------;

        MPY     NUMERA          ; PREG = RC(MINC)*SUM       in Q2
        LACC    DENOM,16
        APAC
        SACH    DENOM         ; ALPHA = ALPHA+RC(MINC)*SUM in Q-13
        BCND    END37_3,LEQ     ; if ALPHA <= 0 goto end

        MAR     *,AR3
        MAR     *+,AR5          ; update AR3
        MAR     *+,AR6          ; update AR5
        MAR     *+,AR1

        LAMM    AR3             ;
        SAMM    AR1             ; AR1 = address of R(MINC+1)
        ZAP
;-------------------------------;
        RPT     TMP1            ; TREG0 = R(MINC-IP+2) in Q-13
        MAC     WSPM,*-           ; PREG = AWZTMP*R         in Q2
;-------------------------------;
        APAC                    ; ACCH = SUM + AWZTMP*R   in Q-13
                                ; SUM=SUM+ R(MINC-IP+2)*AWZTMP(IP)
                                ; in Q-13
;
;       calculates  -(SUM/ALPHA)
;
        SACH    NUMERA          ; store SUM to numerator
;
;-------------------------------; division of -(NUMERA/DENOM)
        LACC    NUMERA,16       ;
        ABS                     ; ACCH = abs(NUMERA)
        RPT     #14
        SUBC    DENOM           ; ACCL = abs(NUMERA/DENOM)
```

```
        BIT     NUMERA,0        ; test sign of NUMERA
        AND     TMP9            ; clear High ACC
        XC      1,NTC
        NEG                     ; if NUMERA is positive then negate
                                ;results
;-------------------------------;
;
        SAMM    TREG0           ; TREG0 = RC(MINC)

        SAR     AR6,TMP1        ; TMP1 = 4,6,8
        LMMR    BRCR,#AR4       ; inner loop run 1,2,3,4 times
        LMMR    AR1,#AR5        ; AR1 = address of AWZTMP(IB)
        LAR     AR2,#(AWZTMP+1) ; AR2 = address of AWZTMP(IP)

        BSAR    1
        SACL    *-              ; AWZTMP(MINC+1) = RC(MINC)  in Q15
;-------------------------------;
        RPTB    END37_2
        MPY     *,AR2           ; PREG = RC(MINC)*AWZTMP(IB) in Q30
        LACC    *,16        ; ACCH = AWZTMP(IP)             in Q15
        MPYA  * ; ACCH = AWZTMP(IP)+RC(MINC)*AWZTMP(IB) in Q15
                                ; PREG = RC(MINC)*AWZTMP(IP) in Q30
        SACH    *+,AR1          ; AWZTMP(IP)=AWZTMP(IP)+RC(MINC)*
                                ; AWZTMP(IB) Q15
        LACC    *,16            ; ACCH = AWZTMP(IB)           in Q15
        APAC                    ;
END37_2 SACH    *-              ; AWZTMP(IB)=AWZTMP(IB)+RC(MINC)*
                                ; AWZTMP(IP) Q15
;-------------------------------;
        MPY     *
        LACC    *,16
        APAC
        SACH  * ; AWZTMP(MINC/2+1)=AWZTMP(MINC/2+1)          in Q15
                                ; +RC(MINC)*AWZTMP(MINC/2+1)
        MPY     NUMERA          ; PREG = RC(MINC)*SUM       in Q2
        LACC    DENOM,16
        APAC
        SACH    DENOM         ; ALPHA = ALPHA+RC(MINC)*SUM in Q-13
        BCND    END37_3,LEQ     ; if ALPHA <= 0 goto end

        MAR     *,AR3
        MAR     *+,AR6
        MAR     *+,AR0
        BANZD   BEG37_1,*-,AR4  ; goto BEG37_1
        MAR     *+,AR5
        MAR     *+,AR1
;-----------------------------------------------------------------

;================================================
;           block47.asm
;================================================
        LACC    GAIN,16
        BCND    END47_1,LEQ     ; if GAIN <= 0 goto END47_1
        LACC    GAIN
        SUB     OFFSET4         ; ACCL = ACCL - 60 Q9
        BCND    END47_2,GEQ     ; if GAIN >= 60 goto END47_2
;
        LACC    GAIN            ; restore GAIN
;-------GAIN=10^(GAIN/20)-----------------------------------------
;       assume 10^(GAIN/20) = x
;       we got  GAIN = 20log10(x) = 20log2(x)/log2(10)
;       then    GAIN*log2(10)/20 = log2(x)
;       as a result
;            2^(GAIN*log2(10)/20) = 2^(log2(x)) = x = 10^(GAIN/20)
;
        SAMM    TREG0           ; TREG0 = GAIN          in Q9
        MPY     TMP7            ; PREG = GAIN*(log2(10)/20) in Q26
;        CALLD   BEXP2           ; call with x in Q11 at ACCH
                                ; return with 2^x in Q5 at ACCL
        PAC                     ; ACCH = GAIN*(log2(10)/20) in Q11
;        NOP
        BSAR    11
        SACL    TMP0            ; store floating-point in TEMP0
        BSAR    16
;        AND     #0ffffh,16              ; clear low accumulator
        NEG
        ADD     #9              ; ACCH = 9 - integer
        SAMM    TREG1

        LACL    TMP0
        BSAR    8               ;
        ADD     EXP2            ; norm(x) - 1.0 IN Q5
        TBLR    TMP0

;        RETD
        LACC    TMP0            ; mantisca (Q8)
        SATL                  ; shift ACC right (TREG1) bits

;
        SACL    GAIN            ; GAIN = 10^(GAIN/20)   in Q5
        B       END47_3
END47_1:
        SPLK    #020h,GAIN      ; GAIN = 1      in Q5
        B       END47_3
END47_2:
        SPLK    #7d00h,GAIN     ; GAIN = 1000   in Q5

END47_3


;================================================
;           block67.asm
;================================================
        LAR     AR1,#ET
        ZAP                     ; ACC = PREG = 0
        SPM     2               ; PM=2 , shift product left 4 bits
        RPT     #4
        SQRA    *+              ; ACC = ACC + (ET(I))^2
        APAC                    ; ACC = SUM((ET(I))^2),I=1-5 in Q10
        BSAR    16              ; ACCL = SUM((ET(I)^2)) in Q-6
```

```
        SAMM    TREG0           ; TREG0 = ETRMS         in Q-6
        MPY     #0ccch          ; PREG = ETRMS*0.2      in Q11
        PAC                     ; ACCH = 0.2*ETRMS      in Q-4
        SUB     OFFSET0         ; ACCH = ETRMS - 1      in Q-4
        BCND    HERE,LEQ        ; if ETRMS < 1 set ETRMS = 1
;       ETRMS = 10*LOG(ETRMS)
;             = 10*LOG2(ETRMS)/LOG2(10) = 10*LOG10(2)*LOG2(ETRMS)
;
;       suboroutine BILOG2 assumes ETRMS at ACCH is in Q14  so when
;       we get returned value at ACCL in Q8 we must add 18 Q8 to it

        ADD     OFFSET0
;
        LAR     AR1,#15-15      ; after NORM here is still 1 sign bit
        RPT     #(14-1)                 ; norm(x) in Q14
        NORM    *+                      ; 1 < norm(x) < 2
        BCND    HERE,EQ         ; check for exceptional case
        BSAR    16                      ; ACCH(Q14) --> ACCL(Q14)
        BSAR    14-8            ; form Q8 index from ACCL(Q14)
        ADD     LOG2                    ; norm(x) - 1.0 IN Q8
        TBLR    TMP0

        LACC    TMP0                  ; mantisca (Q10)
        SAR     AR1,TMP0        ; combine with the exponent in AR1
        SUB     TMP0,10               ; return(A=log2(x)) in Q10
        ADD     OFFSET1         ; add 18 Q10 to log2(ETRMS)  in Q10
        SAMM    TREG0           ; TREG0 = log2(ETRMS)        in Q10
        MPY     #01815h         ; PREG=10*log10(2)*log2(ETRMS)in Q21
        PAC                     ; ACCH = 10log10(ETRMS)          in Q9
                                ;      = dB value of ETRMS       in Q9
        SUB     OFFSET,16       ; ACCL = ETRMS - 32      in Q9
        SACH    GSTATE          ; GSTATE(1) = ACCL       in Q9
        B       END67_1
;
HERE:
        SPLK    #0c000h,GSTATE  ; GSTATE(1) = -32 when ETRMS<1

END67_1
        LMMR    CBSR1,#GS
        LMMR    CBER1,#GE
        LMMR    AR1,#GPTR
        LACL    #9h
        SAMM    CBCR
        SPM     1
        LACC    GSTATE          ; store GSTATE to GP buffer
        SACL    *+
        SMMR    AR1,#GPTR
```

## ภาคผนวก จ

ตารางเปรียบเทียบปริมาณการคำนวณและหน่วยความจำที่ต้องการ

| | ตัวประมวลผล | ตัวเข้ารหัส (Encoder )ของ LD-CELP | | |
|---|---|---|---|---|
| | | หน่วยความจำสำหรับโปรแกรม | หน่ายความจำสำหรับข้อมูล | MIPS |
| DSPSE*** | TMS320C5x | 6.2 kwords | 1.9 kwords | 21.9** |
| SASL**** | TMS320C5x | 4 kwords | 1 kwords | 23.0* |
| งานวิจัยนี้ | TMS320C50 | 4.5 kwords | 2 kwords | 21* |

หมายเหตุ * ปริมาณการคำนวณสูงสุดในกรณี worst-case

** ปริมาณการคำนวณเฉลี่ย

*** บริษัท DSP Software Engineering,Inc

**** บริษัท Signal and Software Limited

ที่มาของข้อมูล บริษัทเท็กซัสอินสตรูเมนต์ โฮมเพจ "TMS320 Software Cooperative Resource Guide" (http://www.ti.com/sc/docs/dsps/softcoop/3voctabl.htm)

# G.728 Audio Coder for the TMS320C5x

*by Signals and Software Limited*

## Software Overview

DSP software which implements the fixed-point variant of the ITU G.728 speech-coding specification. This code's narrow-band speech (300–3400 Hz, sampled at 8 kHz) to a rate of 16 kb/s. The G.728 specification for floating-point DSPs has been in existence for some time. The fixed-point algorithm interworks directly with the floating-point version and enables the standard to be implemented on low-cost fixed-point DSPs. G.728 is primarily intended for Digital Circuit Multiplication Equipment (DCME) and so features a low transcoding delay in order to avoid the creation of telephony sidetone echo. It is also becoming a de-facto standard for video conferencing audio.

## Features and Benefits

The G.728 specification describes a Low-Delay Code-Excited Linear Prediction (LD-CELP) algorithm which employs backward-adaptive Linear Predictive Coding (LPC) analysis in order to achieve a low transcoding delay. The decoder also incorporates an adaptive postfilter to enhance performance for multiple transcodings. The postfilter function can be omitted for single coder-decoder operation to reduce the processing power required on the DSP. The performance of the G.728 algorithm may be summarized as follows:

| Bit rate | 16 kb/s |
|---|---|
| **Audio quality** | |
| • Single transcode | • Comparable to 64-kb/s PCM (G.711), identical to G.726 at 32 kb/s |
| • Multiple transcodes | • Degradation only apparent after 6 tandem transcodes |
| • Errored transmission | • Better or equivalent to G.726 at 32 kb/s |
| • Transcoding delay | • 2 ms |
| • Music coding | • No detrimental effects |
| **Transmission properties** | |
| •DTMF/network signalling tones | • Equivalent to 64-kb/s PCM |
| •Modem signals | • Passes modem signals up to 2400 bps (note that the encoder perceptual weighting filter must be disabled) |

The software passes the full set of ITU G.728 (fixed-point) test vectors.

## Processor and System Requirements

| G.728 | Program Memory (Words) | | Data Memory (Words) | | Processing Load (MIPS)* |
|---|---|---|---|---|---|
| | Code | Data Tables | Permanent | Temporary | |
| Encoder + decoder + postfilter | 5206 | 1102 | 1840 | 160 | 37.5 |
| Encoder + decoder | 4285 | 1075 | 1219 | 160 | 33.5 |
| Encoder only | 2894 | 1075 | 754 | 160 | 23.0 |
| Decoder + postfilter | 2780 | 1020 | 1086 | 160 | 14.5 |
| Decoder only | 1859 | 993 | 465 | 160 | 10.5 |

* The MIPS figures above are worst-case values over a frame of 20 speech samples.

## Usage Limitations or Performance Considerations

The software consists of six subroutines: an initialization routine and a processing routine for each of the encoder, decoder, and postfilter functions. The audio input and output format is 8-kHz linear samples, and the frame size used is 20 samples. However, the processing routines operate on a subframe basis and are called every five samples. The encoder converts each subframe into a 10-bit codeword, which is used by the decoder to reconstruct the signal. The postfilter operates on the five sample decoder output subframe.

Applications include:

- Speech compression for DCME equipment
- Coding of the audio channel for video conferencing and desktop terminals where transmission bandwidth is limited
- Speech storage

## Availability

Now, under licence, for a one-off payment and/or royalties depending on the commercial application. An Application Note is available. Support consultancy for code integration is also available.

## Company Background and Contact Information

Signals and Software Limited (SASL), based in Harrow, Middlesex, UK is a design consultancy specializing in the area of Digital Signal Processing (DSP). From concept and algorithm design through to real-time DSP implementation, SASL is able to offer its clients fast and cost-effective solutions to their DSP needs. Services include: feasibility studies, DSP software to order, DSP research and algorithm design, computer simulations (C or PASCAL), and hardware design. In support of these services, SASL offers a range of "off-the-shelf" software.

**Contact:** David Morley
3 Jardine House
Bessborough Road
Harrovian Business Village
Harrow, Middlesex, HA1 3EX
United Kingdom
+44 (0) 181 426 9533
Fax: +44 (0) 181 869 1182
e-mail:
davem@sasl.demon.co.uk


# ITU G.728 LD-CELP Vocoder for TMS320C5x

*by DSP Software Engineering, Inc.*

## Software Overview

The G.728 LD-CELP vocoder software is an implementation of ITU (formerly CCITT) G.728 for the TMS320C5x. G.728 is an international standard for encoding 8-kHz sampled-speech signals for transmission over 16-kbps channels. G.728 provides approximately 4 kHz of speech bandwidth.

G.728 encodes 5 sample frames of 16-bit linear-PCM data into 10-bit code words. G.728 has numerous applications in products that require high-quality speech coding with very-low delay at 16 kbps. These include videoconferencing, digital telephony, and multimedia products.

## Features and Benefits

- Full ITU implementation on a single TMS320C5x
- 2.5-ms algorithmic delay (using a 5-sample frame size)
- Independent, C-callable functions
- In-band synchronization capable
- Run-time selectable, optimal post filtering
- SPOX compatible

## Processor and System Requirements

- **Devices supported:** TMS320C5x
- **Algorithm category:** Vocoder
- **Requirements:**
  All memory requirements are in units of 16-bit words. The MIPs ratings presented require the listed amounts of dual-access RAM (DARAM), zero-wait-state program and data memory, and a processing frame size of 5 samples. All measurements were made using an executable demonstration built for a third-party PC plug-in board.

| Function | Avg MIPS | DARAM | Data | Program |
|---|---|---|---|---|
| Encoder (half duplex) | 21.9 | 0.6 k | 1.3 k | 6.2 k |
| Decoder (half duplex) | 15.0 | 0.6 k | 1.6 k | 7.0 k |
| **Full Duplex*** | **36.9** | **1.0 k** | **2.2 k** | **10.5 k** |

* Significant portions of read-only program and data are shared by the encoder and decoder.

## User Functions

The G.728 implementation consists of two independent, C-callable functions that perform encoding and decoding operations. It also includes functions that reflect the implementation's object-based interface. The encoder and decoder interface with arrays of 16-bit linear-PCM samples and 10-bit code words.

| *G.728 Functions* | |
|---|---|
| G728_create (...) | Initializes data memory for encoder/decoder |
| G728_decode (...) | Decodes a frame of code words into a frame of 16-bit linear-PCM samples |
| G728_encode (...) | Encodes a frame of 16-bit linear-PCM samples into a frame of code words |

## Algorithm Verification

- Call DSPSE for verification details

## Availability

- Source and object code currently available for licensing.

## Company Background and Contact Information

DSP Software Engineering, Inc (DSPSE), founded in 1989, is a leading provider of highly-complex digital-signal-processing software used in telecommunications and multimedia applications such as computer telephony, digital wireless, PSTN, and satellite communications, and videoconferencing.

**Contact:** DSP Software Engineering, Inc
175 Middlesex Turnpike
Bedford, MA 01730 USA
(617) 275-3733
Fax: (617) 275-4323
e-mail: info@dspse.com
www: http://www.dspnet.com

# ประวัติผู้เขียน

นาย พูนลาภ ลามศรีจันทร์ เกิดเมื่อวันที่ 29 กรกฎาคม พ.ศ. 2515 ที่จังหวัดชัยภูมิ สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต เกียรตินิยมอันดับหนึ่ง สาขาวิศวกรรมไฟฟ้าสื่อสาร ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา พ.ศ. 2536 เข้าเป็นนิสิตในโครงการศิษฐ์ก้นกุฏิ เพื่อศึกษาต่อในหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้าสื่อสาร ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา พ.ศ. 2537