สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โครงการพัฒนาระบบสร้างและตรวจจับสัญญาณโทรศัพท์ บน TMS320C6713

THE DEVELOPMENT OF TELEPHONE SIGNAL GENERATOR AND DETECTOR ON TMS320C6713

T061481

นายภาณุพงศ์ อ่องสุวรรณ
นายมาโนช ปัญญาวชิราภรณ์

เลขหมู่.....
เลขทะเบียน 61481
วัน,เดือน,ปี 18 ก.ค. 2549

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

โครงการพัฒนาระบบสร้างและตรวจจับสัญญาณโทรศัพท์ บน TMS320C6713

THE DEVELOPMENT OF TELEPHONE SIGNAL GENERATOR AND DETECTOR ON TMS320C6713

โดย

นายภาณุพงศ์ อ่องสุวรรณ

นายมาโนช ปัญญาวชิราภรณ์

อาจารย์ที่ปรึกษา

ผศ.ดร.อรณัตร จิตต์โสภัคตร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ ปีการศึกษา 2547

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โครงการพัฒนาระบบสร้างและตรวจจับสัญญาณโทรศัพท์ บน TMS320C6713

THE DEVELOPMENT OF TELEPHONE SIGNAL GENERATOR AND DETECTOR ON TMS320C6713

คณะผู้จัดทำ นายภาณุพงศ์ อ่องสุวรรณ รหัส 44010363

นายมาโนช ปัญญาวชิราภรณ์ รหัส 44010379

อรฉัตร จิตต์.....อาจารย์ที่ปรึกษา

(ผศ.ดร.อรฉัตร จิตต์โสภักตร์)

โครงการพัฒนาระบบสร้างและตรวจจับสัญญาณโทรศัพท์ บน TMS320C6713

นายภาณุพงศ์ อ่องสุวรรณ 44010363
นายมาโนช ปัญญาวชิราภรณ์ 44010379
ผศ.ดร.อรฉัตร จิตต์โสภักตร์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

โครงงานนี้เป็นการพัฒนา Library ระบบสร้างและตรวจจับสัญญาณโทรศัพท์ บน TMS320C6713 ซึ่งเป็น DSP Chip ของบริษัท Texas Instrument Inc. โดยจะทำการปรับ Code ให้อยู่ในรูปแบบของ TMS320 DSP Algorithm Standard ซึ่งประเภทของสัญญาณโทรศัพท์ที่จะต้องสามารถทำการสร้าง และตรวจจับให้ได้ ประกอบด้วย [1] สัญญาณ DTMF, [2] สัญญาณ R2 MF, และ [3] สัญญาณ CPT ซึ่งระบบที่สร้างขึ้นนั้นได้รับการออกแบบมาเพื่อทำงานที่ Sampling Rate เท่ากับ 8000 Hz ในการสร้างสัญญาณนั้นจะใช้ค่าคงที่ตารางค่าไซน์ขนาด 512 ตัว มาทำการสร้างสัญญาณไซน์ส่วนประกอบ เพื่อนำไปใช้ในการสร้างสัญญาณทางโทรศัพท์ที่ต้องการ ซึ่งค่าจังหวะและความถี่ส่วนประกอบ ของสัญญาณที่ต้องการสร้างนั้น สามารถทำการกำหนดได้โดยผู้ใช้งาน Library ส่วนในการตรวจจับสัญญาณนั้น จะประกอบด้วย การตรวจสอบจังหวะของสัญญาณ, การตรวจสอบระดับค่าพลังงานของสัญญาณ, และการตรวจสอบความถี่ของสัญญาณไซน์ส่วนประกอบ โดยข้อกำหนดในการตรวจสอบต่างๆนั้น สามารถกำหนดได้โดยผู้ใช้งาน Library เช่นกัน สำหรับการตรวจสอบความถี่ของสัญญาณไซน์ส่วนประกอบนั้นจะใช้ Goertzel Algorithm ซึ่งได้รับการปรับปรุงมาจากการทำ Discrete Fourier Transform (DFT) โดยจะทำการแปลงสัญญาณให้อยู่ในรูปของ Frequency Domain เพื่อให้ในการหาว่ามีความถี่ของสัญญาณไซน์ส่วนประกอบใดที่เข้ามากับสัญญาณบ้าง จากการทดสอบ Library ของระบบสร้างและตรวจจับสัญญาณโทรศัพท์ พบว่า ส่วนสร้างสัญญาณสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้ แต่ในส่วนของการตรวจจับสัญญาณนั้น พบว่า ยังมีปัญหาเล็กน้อยในส่วนของการเบี่ยงเบนความถี่ และปัญหาในกรณีที่จังหวะของสัญญาณก้ำกึ่งกับค่าที่ผู้ใช้กำหนด

THE DEVELOPMENT OF TELEPHONE SIGNAL GENERATOR AND DETECTOR ON TMS320C6713

Mr. Phanupong Ongsuwan

Mr. Manoch Panyavachiraporn

Asst. Prof. Dr. Orachat Chitsobhuk Advisor

Academic Year 2004

ABSTRACT

This thesis is to develop the telephone signal generator and detector library on TMS320 DSP Algorithm Standard approach on TMS320C6713, DSP Chip from Texas Instrument Inc. The type of telephone signal that the library must able to generate and detect are [1] DTMF, [2] R2 MF, and [3] CPT. The library is designed to work at 8000 Hz sampling rate. The 512-element sine table is used in signal generation to produces the composite sinusoid signal, and then the desired telephone signal is made of these composite signals. The signal timing and the frequency of the composite sinusoid signal can be chosen in signal generation. The signal detection consists of 3 sub operations such as signal timing checking, signal power level checking, and composite sinusoids' frequency checking. The specification of signal detection can be defined by the user, also. In composite sinusoids' frequency checking operation, Goertzel Algorithm, which has been developed from Discrete Fourier Transform (DFT), is used. The Goertzel Algorithm transforms the input signal into frequency domain form. So it can be used to find that the input signal contains the composite frequency of the specific telephone signal or not. In the library testing phase, it shows that the generation library is work properly as it has been designed. However, the detection library has some problems in frequency deviation and in case of input signal timing is near the accept/reject edge in signal detection specification.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ และคำปรึกษา จากหลายๆท่าน โดยเฉพาะท่านอาจารย์ที่ปรึกษา คือ ผศ.ดร.อรฉัตร จิตต์โสภักตร์ รวมถึง ดร.เสาวลักษณ์ แก้วกำเนิด นักวิจัยประจำศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำหรับ DSP Board ที่นำมาใช้ในการทำงานในครั้งนี้ รวมถึงเจ้าหน้าที่ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติทุกท่านที่ให้การดูแล และความรู้ ในช่วงที่ไปทำการฝึกงานภาคฤดูร้อน ประจำปีการศึกษา 2546

ขอขอบคุณ บิดา มารดา ที่คอยเลี้ยงดู สั่งสอน และให้โอกาสในการศึกษาอย่างเต็มที่ อีกทั้งยังคอยให้กำลังใจในการทำงานต่างๆ มาโดยตลอด

สุดท้ายนี้ ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ และทุกๆท่านที่ไม่ได้เอ่ยถึงในที่นี้ ที่ได้ให้คำปรึกษาต่างๆ ทั้งในเรื่องการทำงาน และเรื่องอื่นๆ ขอขอบคุณมา ณ โอกาสนี้ด้วย

นายภาณุพงศ์ อ่องสุวรรณ
นายมาโนช ปัญญาวชิราภรณ์

สารบัญ

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญตาราง IX
สารบัญรูป XIV
บทที่ 1 บทนำ 1
1.1 หลักการและเหตุผล 1
1.2 ขอบเขตของโครงงาน 1
1.2.1 ขอบเขตของ Library สร้างสัญญาณ DTMF 2
1.2.2 ขอบเขตของ Library ตรวจจับสัญญาณ DTMF 2
1.2.3 ขอบเขตของ Library สร้างสัญญาณ R2 MF 2
1.2.4 ขอบเขตของ Library ตรวจจับสัญญาณ R2 MF 3
1.2.5 ขอบเขตของ Library สร้างสัญญาณ CPT 3
1.2.6 ขอบเขตของ Library ตรวจจับสัญญาณ CPT 3
1.3 วัตถุประสงค์ของโครงงาน 3
1.4 เป้าหมายของโครงงาน 3
บทที่ 2 ทฤษฎี สัญญาณทางโทรศัพท์ 5
2.1 สัญญาณ DTMF 5
2.2 สัญญาณ R2 MF 6
2.3 สัญญาณ CPT 8
2.3.1 มาตรฐาน CPT ฝั่งผู้ส่ง (Transmitter) 9
2.3.2 มาตรฐาน CPT ฝั่งผู้รับ (Receiver) 10
บทที่ 3 ทฤษฎี การสร้างและการตรวจจับสัญญาณ 12
3.1 การสร้างสัญญาณ 12
3.2 การตรวจจับสัญญาณ 13
3.3 A-Law Companding 18
บทที่ 4 ทฤษฎี TMS320 DSP Algorithm Standard 20
4.1 Reentrancy 22
บทที่ 5 การสร้าง ส่วนสร้างสัญญาณโทรศัพท์ 23
5.1 ตารางค่าไซน์ 24
5.2 การใช้ตารางค่าไซน์ในการสร้างสัญญาณไซน์ 24

5.3 การสร้าง ส่วนสร้างสัญญาณ DTMF (DTMFG) 26
5.3.1 อัลกอริทึมของการสร้างสัญญาณ DTMF 30
5.4 การสร้าง ส่วนสร้างสัญญาณ R2 MF (R2MFG) 32
5.4.1 อัลกอริทึมของการสร้างสัญญาณ R2 MF 35
5.5 การสร้าง ส่วนสร้างสัญญาณ CPT (CPTGEN) 36
5.5.1 อัลกอริทึมของการสร้างสัญญาณ CPT 36
บทที่ 6 การสร้าง ส่วนตรวจจับสัญญาณโทรศัพท์ 40
6.1 การตรวจสอบจังหวะของสัญญาณ 42
6.2 การตรวจสอบระดับค่าพลังงานของสัญญาณ 42
6.3 การนำ Goertzel Algorithm มาประยุกต์ใช้ 42
6.4 การสร้าง ส่วนตรวจจับสัญญาณ DTMF (DTMFD) 44
6.4.1 อัลกอริทึมของการตรวจจับสัญญาณ DTMF 48
6.5 การสร้าง ส่วนตรวจจับสัญญาณ R2 MF (R2MFD) 52
6.5.1 อัลกอริทึมของการตรวจจับสัญญาณ R2 MF 55
6.6 การสร้าง ส่วนตรวจจับสัญญาณ CPT (CPTDETECT) 56
6.6.1 อัลกอริทึมของการตรวจจับสัญญาณ CPT 56
บทที่ 7 การพัฒนา Library ของส่วนสร้างสัญญาณโทรศัพท์ 64
7.1 การพัฒนา Library ของส่วนสร้างสัญญาณ DTMF (DTMFG) 64
7.1.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 64
7.1.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 65
7.1.3 กำหนดการทำงานของฟังก์ชั่น algInit 65
7.1.4 กำหนดการทำงานของฟังก์ชั่น algActivate 65
7.1.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate 66
7.1.6 กำหนดการทำงานของฟังก์ชั่น algMoved 66
7.1.7 กำหนดการทำงานของฟังก์ชั่น algFree 66
7.1.8 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_apply 66
7.1.9 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_enqueue 66
7.1.10 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_setEnable 66
7.2 การพัฒนา Library ของส่วนสร้างสัญญาณ R2 MF (R2MFG) 66
7.2.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 67
7.2.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 67
7.2.3 กำหนดการทำงานของฟังก์ชั่น algInit 67
7.2.4 กำหนดการทำงานของฟังก์ชั่น algActivate 68
7.2.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate 68

7.2.6 กำหนดการทำงานของฟังก์ชั่น algMoved 68
7.2.7 กำหนดการทำงานของฟังก์ชั่น algFree 68
7.2.8 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_apply 68
7.2.9 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_setDigit 68
7.2.10 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_setState 68
7.3 การพัฒนา Library ของส่วนสร้างสัญญาณ CPT (CPTGEN) 68
7.3.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 69
7.3.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 70
7.3.3 กำหนดการทำงานของฟังก์ชั่น algInit 71
7.3.4 กำหนดการทำงานของฟังก์ชั่น algMoved 72
7.3.5 กำหนดการทำงานของฟังก์ชั่น algFree 72
7.3.6 กำหนดการทำงานของฟังก์ชั่น *apply* 73
บทที่ 8 การพัฒนา Library ของส่วนตรวจจับสัญญาณโทรศัพท์ 74
8.1 การพัฒนา Library ของส่วนตรวจจับสัญญาณ DTMF (DTMFD) 74
8.1.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 74
8.1.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 75
8.1.3 กำหนดการทำงานของฟังก์ชั่น *algInit* 75
8.1.4 กำหนดการทำงานของฟังก์ชั่น algActivate 75
8.1.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate 75
8.1.6 กำหนดการทำงานของฟังก์ชั่น algMoved 75
8.1.7 กำหนดการทำงานของฟังก์ชั่น algFree 75
8.1.8 กำหนดการทำงานของฟังก์ชั่น *DTMFD_KASATKA_apply* 75
8.2 การพัฒนา Library ของส่วนตรวจจับสัญญาณ R2 MF (R2MFD) 75
8.2.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 76
8.2.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 76
8.2.3 กำหนดการทำงานของฟังก์ชั่น algInit 76
8.2.4 กำหนดการทำงานของฟังก์ชั่น *algActivate* 77
8.2.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate 77
8.2.6 กำหนดการทำงานของฟังก์ชั่น algMoved 77
8.2.7 กำหนดการทำงานของฟังก์ชั่น algFree 77
8.2.8 กำหนดการทำงานของฟังก์ชั่น R2MFD_KASATKA_apply 77
8.3 การพัฒนา Library ของส่วนตรวจจับสัญญาณ CPT (CPTDETECT) 77
8.3.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure 78
8.3.2 กำหนดการทำงานของฟังก์ชั่น algAlloc 78

8.3.3 กำหนดการทำงานของฟังก์ชั่น algInit 79
8.3.4 กำหนดการทำงานของฟังก์ชั่น algMoved 80
8.3.5 กำหนดการทำงานของฟังก์ชั่น algFree 81
8.3.6 กำหนดการทำงานของฟังก์ชั่น apply 81
บทที่ 9 การทดลอง เพื่อประเมินค่า THRESHOLD ที่จะนำมาใช้ 82
9.1 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น 82
9.1.1 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น สำหรับสัญญาณ DTMF 83
9.1.2 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น สำหรับสัญญาณ R2 MF 87
9.1.3 ค่า THRESHOLD เริ่มต้น ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF 93
9.2 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ DTMF 93
9.3 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ R2 MF 95
9.4 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ CPT 95
9.5 สรุปค่า THRESHOLD ที่จะนำไปใช้ ของ DTMF, R2MF 102
บทที่ 10 การทดสอบ การทำงานของ Library 103
10.1 การทดสอบ การทำงานของ Library สร้างสัญญาณ DTMF 103
10.2 การทดสอบ การทำงานของ Library สร้างสัญญาณ R2 MF 103
10.3 การทดสอบ การทำงานของ Library สร้างสัญญาณ CPT 104
10.4 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ DTMF 104
10.4.1 การทดสอบทั่วไป ของการตรวจจับสัญญาณ DTMF 104
10.4.2 การทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ DTMF 104
10.4.3 การทดสอบการเปลี่ยนแปลงจังหวะสัญญาณ ของการตรวจจับสัญญาณ DTMF 105
10.5 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ R2 MF 106
10.5.1 การทดสอบทั่วไป ของการตรวจจับสัญญาณ R2 MF 106
10.5.2 การทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ R2 MF 107
10.6 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ CPT 107
10.6.1 Dial Tone 109
10.6.2 Ring Tone 111
10.6.3 Busy Tone 113
10.6.4 Congestion Tone 115
10.6.5 Special Information 117
10.6.6 Warning Tone 121
10.6.7 Payphone Tone 123
10.6.8 Call Waiting 126
10.6.9 Recall Dial Tone 128

10.6.10 Intercept Tone 131
10.6.11 *Confirmation Tone* 134
10.6.12 Comfort Tone 137
10.6.13 Busy Verification 139
10.6.14 *Caller Waiting* 142
10.6.15 สรุปผลการทดลอง CPT 145
บทที่ 11 บทวิจารณ์และสรุป 148
11.1 เกี่ยวกับ Algorithm Standard 148
11.2 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ DTMF 148
11.3 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ R2 MF 148
11.4 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ CPT 149
11.5 ข้อจำกัดในการใช้งาน ระบบสร้างและตรวจจับสัญญาณโทรศัพท์ 149
บรรณานุกรม 152

สารบัญตาราง

ตารางที่ 2-1	คู่ความถี่ที่ใช้ในการสร้างสัญญาณ DTMF	5
ตารางที่ 2-2	ข้อกำหนดในการตรวจสอบสัญญาณ DTMF	6
ตารางที่ 2-3	R2 MF, Forward Signals	7
ตารางที่ 2-4	R2 MF, Backward Signals	7
ตารางที่ 2-5	ข้อกำหนดในการตรวจสอบสัญญาณ R2 MF	7
ตารางที่ 2-6	มาตรฐาน CPT ฝั่งผู้ส่ง	9
ตารางที่ 2-7	มาตรฐาน CPT ฝั่งผู้รับ	10
ตารางที่ 3-1	สมการหลักที่ใช้ในการสร้างสัญญาณ	12
ตารางที่ 3-2	สมการจาก Goertzel Algorithm ที่ใช้ในการสร้างส่วนตรวจจับสัญญาณ	17
ตารางที่ 5-1	คำสั่งในการสร้าง Sine Table	24
ตารางที่ 5-2	คำสั่งในการคำนวณค่า Step Size / Sample	25
ตารางที่ 5-3	ตัวอย่างคำสั่งภาษา C ในการสร้างสัญญาณทางโทรศัพท์ที่ต้องการ	25
ตารางที่ 5-4	Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนสร้างสัญญาณ DTMF	26
ตารางที่ 5-5	Structure เพื่อใช้ในการเก็บตัวแปรของส่วนสร้างสัญญาณ DTMF	28
ตารางที่ 5-6	ค่าที่ใช้ในการสร้างสัญญาณ DTMF เทียบกับ หมายเลข DTMF จริง	28
ตารางที่ 5-7	Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนสร้างสัญญาณ R2 MF	32
ตารางที่ 5-8	Structure เพื่อใช้ในการเก็บตัวแปรของส่วนสร้างสัญญาณ R2 MF	33
ตารางที่ 5-9	การตั้งค่าที่ใช้ในการสร้างสัญญาณ R2 MF	34
ตารางที่ 5-10	ใช้ switch case ในการสร้างสัญญาณ	37
ตารางที่ 5-11	ตัวอย่างการนำ อัลกอริทึม ไปใช้	38
ตารางที่ 5-12	Structure ในที่ใช้ในการบันทึก state ในการสร้างสัญญาณ	38
ตารางที่ 6-1	สูตรที่ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ	42
ตารางที่ 6-2	Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนตรวจจับสัญญาณ DTMF	45
ตารางที่ 6-3	Structure เพื่อใช้ในการเก็บค่าตัวแปรของส่วนตรวจจับสัญญาณ DTMF	46
ตารางที่ 6-4	ความหมายค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ DTMF	47
ตารางที่ 6-5	การหาค่าตัวคูณ เพื่อนำไปใช้ในการขยายสัญญาณ	51
ตารางที่ 6-6	Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนตรวจจับสัญญาณ R2 MF	53
ตารางที่ 6-7	Structure เพื่อใช้ในการเก็บค่าตัวแปรของส่วนตรวจจับสัญญาณ R2 MF	54
ตารางที่ 6-8	ความหมายค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ R2 MF	54

ตารางที่ 6-9	Structure ที่ใช้ในการบันทึก State ของการทำ CPT Detection	57
ตารางที่ 6-10	Source code อย่างย่อในการตรวจสอบ Timing ของ CPT	58
ตารางที่ 6-11	*Source Code อย่างย่อในการใช้หลักการ State ในการตรวจสอบสัญญาณ*	59
ตารางที่ 6-12	มาตรฐานของ CPT ที่นำมาจากบางส่วนของ DTMF	60
ตารางที่ 6-13	ฟังก์ชันในการตรวจสอบ Power และความถี่ของสัญญาณ	62
ตารางที่ 6-14	Source code ในการคำนวณหาค่าจากสมการ Goertzel Algorithm	63
ตารางที่ 7-1	Code ที่ใช้ในการกำหนด memTab ของ Library DTMFG	64
ตารางที่ 7-2	Code ที่ใช้ในการกำหนด memTab ของ Library R2MFG	67
ตารางที่ 7-3	ฟังก์ชั่น ที่จะถูก Implement ใน vector table	69
ตารางที่ 7-4	Structure ที่ไว้บันทึก State ใน Generation Algorithm	69
ตารางที่ 7-5	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algAlloc	70
ตารางที่ 7-6	Structure ที่ใช้เป็นตัวแปร Global เพื่อให้ อัลกอรึทึม ทำงานได้หลาย Channel	71
ตารางที่ 7-7	*ฟังก์ชั่น ที่ถูกเรียกใช้แทน algInit*	71
ตารางที่ 7-8	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algMoved	72
ตารางที่ 7-9	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algFree	72
ตารางที่ 8-1	Code ที่ใช้ในการกำหนด memTab ของ Library DTMFD	74
ตารางที่ 8-2	Code ที่ใช้ในการกำหนด memTab ของ Library R2MFD	76
ตารางที่ 8-3	*ฟังก์ชั่น ที่ถูก Implement ใน Vector Table*	77
ตารางที่ 8-4	Structure ที่ใช้บันทึก state ของ CPT Detection Algorithm	78
ตารางที่ 8-5	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algAlloc	79
ตารางที่ 8-6	Structure ที่ใช้เป็นตัวแปร Global เพื่อให้ อัลกอรึทึม ทำงานได้หลาย Channel	79
ตารางที่ 8-7	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algInit	80
ตารางที่ 8-8	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algMoved	80
ตารางที่ 8-9	ฟังก์ชั่น ที่ถูกเรียกใช้แทน algFree	81
ตารางที่ 9-1	ตัวอย่างการตีความค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ R2 MF	55
ตารางที่ 9-2	ค่า THRESHOLD เริ่มต้น ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF	93
ตารางที่ 9-3	สรุปผลการทดลอง การประเมินค่า THRESHOLD สำหรับสัญญาณ DTMF	94
ตารางที่ 9-4	สรุปผลการทดลอง การประเมินค่า THRESHOLD สำหรับสัญญาณ R2 MF	95
ตารางที่ 9-5	ค่า THRESHOLD ที่ปรับแล้ว ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF	102

ตารางที่ 10-1	การตั้งค่าในการทดสอบ การทำงานของ Library สร้างสัญญาณ DTMF	103
ตารางที่ 10-2	การตั้งค่าในการทดสอบ การทำงานของ Library สร้างสัญญาณ R2 MF	103
ตารางที่ 10-3	*ผลการทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ DTMF*	105
ตารางที่ 10-4	ผลการทดสอบเปลี่ยนแปลงจังหวะสัญญาณ ของการตรวจจับสัญญาณ DTMF	106
ตารางที่ 10-5	ผลการทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ R2 MF	107
ตารางที่ 10-6	ผลทดสอบความถี่ของสัญญาณ Dial Tone	109
ตารางที่ 10-7	ผลทดสอบ Power ของสัญญาณ Dial Tone	110
ตารางที่ 10-8	*ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Dial Tone*	110
ตารางที่ 10-9	สรุปการทดลองการตรวจจับสัญญาณ Dial Tone	110
ตารางที่ 10-10	ผลทดสอบความถี่ของสัญญาณ Ring Tone	111
ตารางที่ 10-11	ผลทดสอบ Timing ของสัญญาณ Ring Tone	111
ตารางที่ 10-12	ผลทดสอบ Power ของสัญญาณ Ring Tone	112
ตารางที่ 10-13	*ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Ring Tone*	112
ตารางที่ 10-14	สรุปการทดลองการตรวจจับสัญญาณ Ring Tone	112
ตารางที่ 10-15	ผลทดสอบความถี่ของสัญญาณ Busy Tone	113
ตารางที่ 10-16	ผลทดสอบ Timing ของสัญญาณ Busy Tone	113
ตารางที่ 10-17	ผลทดสอบ Power ของสัญญาณ Busy Tone	114
ตารางที่ 10-18	*ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Busy Tone*	114
ตารางที่ 10-19	สรุปการทดลองการตรวจจับสัญญาณ Busy Tone	114
ตารางที่ 10-20	ผลทดสอบความถี่ของสัญญาณ Congestion Tone	115
ตารางที่ 10-21	ผลทดสอบ Timing ของสัญญาณ Congestion Tone	115
ตารางที่ 10-22	ผลทดสอบ Power ของสัญญาณ Congestion Tone	116
ตารางที่ 10-23	*ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Congestion Tone*	116
ตารางที่ 10-24	สรุปการทดลองการตรวจจับสัญญาณ Congestion Tone	116
ตารางที่ 10-25	ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 960 Hz	117
ตารางที่ 10-26	ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 1400 Hz	117
ตารางที่ 10-27	ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 1800 Hz	118
ตารางที่ 10-28	*ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 1*	118
ตารางที่ 10-29	ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 2	119
ตารางที่ 10-30	ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 3	119
ตารางที่ 10-31	ผลทดสอบ Power ของสัญญาณ Special Information	120
ตารางที่ 10-32	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Special Information	120
ตารางที่ 10-33	*สรุปการทดลองการตรวจจับสัญญาณ Special Information*	120
ตารางที่ 10-34	ผลทดสอบความถี่ของสัญญาณ Warning Tone	121

ตารางที่ 10-35	ผลทดสอบ Timing ของสัญญาณ Warning Tone	121
ตารางที่ 10-36	ผลทดสอบ Power ของสัญญาณ Warning Tone	122
ตารางที่ 10-37	*ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Warning Tone*	122
ตารางที่ 10-38	สรุปการทดลองการตรวจจับสัญญาณ Warning Tone	122
ตารางที่ 10-39	ผลทดสอบความถี่ของสัญญาณ Payphone Tone ที่ 1400 Hz	123
ตารางที่ 10-40	ผลทดสอบความถี่ของสัญญาณ Payphone Tone ที่ 1100 Hz	123
ตารางที่ 10-41	ผลทดสอบ Timing ของสัญญาณ Payphone Tone ช่วงที่ 1	124
ตารางที่ 10-42	*ผลทดสอบ Timing ของสัญญาณ Payphone Tone ช่วงที่ 2*	124
ตารางที่ 10-43	ผลทดสอบ Power ของสัญญาณ Payphone Tone	125
ตารางที่ 10-44	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Payphone Tone	125
ตารางที่ 10-45	สรุปการทดลองการตรวจจับสัญญาณ Payphone Tone	125
ตารางที่ 10-46	ผลทดสอบความถี่ของสัญญาณ Call Waiting	126
ตารางที่ 10-47	*ผลทดสอบ Timing ของสัญญาณ Call Waiting*	126
ตารางที่ 10-48	ผลทดสอบ Power ของสัญญาณ Call Waiting	127
ตารางที่ 10-49	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Call Waiting	127
ตารางที่ 10-50	สรุปการทดลองการตรวจจับสัญญาณ Call Waiting	127
ตารางที่ 10-51	ผลทดสอบความถี่ของสัญญาณ Recall Dial Tone	128
ตารางที่ 10-52	*ผลทดสอบ Timing ของสัญญาณ Recall Dial Tone*	129
ตารางที่ 10-53	ผลทดสอบ Power ของสัญญาณ Recall Dial Tone	129
ตารางที่ 10-54	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Recall Dial Tone	129
ตารางที่ 10-55	สรุปการทดลองการตรวจจับสัญญาณ Recall Dial Tone	130
ตารางที่ 10-56	ผลทดสอบความถี่ของสัญญาณ Intercept Tone	131
ตารางที่ 10-57	*ผลทดสอบ Timing ของสัญญาณ Intercept Tone*	132
ตารางที่ 10-58	ผลทดสอบ Power ของสัญญาณ Intercept Tone	132
ตารางที่ 10-59	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Intercept Tone	132
ตารางที่ 10-60	สรุปการทดลองการตรวจจับสัญญาณ Intercept Tone	133
ตารางที่ 10-61	ผลทดสอบความถี่ของสัญญาณ Confirmation Tone	134
ตารางที่ 10-62	*ผลทดสอบ Timing ของสัญญาณ Confirmation Tone*	135
ตารางที่ 10-63	ผลทดสอบ Power ของสัญญาณ Confirmation Tone	135
ตารางที่ 10-64	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Confirmation Tone	135
ตารางที่ 10-65	สรุปการทดลองการตรวจจับสัญญาณ Confirmation Tone	136
ตารางที่ 10-66	ผลทดสอบความถี่ของสัญญาณ Comfort Tone	137
ตารางที่ 10-67	*ผลทดสอบ Timing ของสัญญาณ Comfort Tone*	138
ตารางที่ 10-68	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Comfort Tone	138

ตารางที่ 10-69	สรุปการทดลองการตรวจจับสัญญาณ Comfort Tone	138
ตารางที่ 10-70	ผลทดสอบความถี่ของสัญญาณ Busy Verification	139
ตารางที่ 10-71	ผลทดสอบ Timing ของสัญญาณ Busy Verification ช่วงที่ 1	139
ตารางที่ 10-72	ผลทดสอบ Timing ของสัญญาณ Busy Verification ช่วงที่ 1	140
ตารางที่ 10-73	ผลทดสอบ Power ของสัญญาณ Busy Verification	140
ตารางที่ 10-74	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Busy Verification	141
ตารางที่ 10-75	สรุปการทดลองการตรวจจับสัญญาณ Busy Verification	141
ตารางที่ 10-76	ผลทดสอบความถี่ของสัญญาณ Caller Waiting	142
ตารางที่ 10-77	ผลทดสอบ Timing ของสัญญาณ Caller Waiting	143
ตารางที่ 10-78	ผลทดสอบ Power ของสัญญาณ Caller Waiting	143
ตารางที่ 10-79	ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Caller Waiting	143
ตารางที่ 10-80	สรุปการทดลองการตรวจจับสัญญาณ Caller Waiting	143
ตารางที่ 10-81	สรุปผลการทดสอบความถี่ของ CPT	145
ตารางที่ 10-82	สรุปผลการทดสอบ Timing ของ CPT	146
ตารางที่ 10-83	สรุปผลการทดสอบ Power ของสัญญาณของ CPT	147

สารบัญรูป

รูปที่ 2-1	ความหมายของคำต่างๆ ที่ใช้ใน ITU-T Recommendation Q.24	5
รูปที่ 2-2	ตัวอย่างการส่งสัญญาณ R2 MF Interregister Signaling	6
รูปที่ 2-3	ค่าความถี่ และจังหวะสัญญาณ ของ สัญญาณ CPT	11
รูปที่ 3-1	รูปคลื่นสัญญาณไซน์	13
รูปที่ 3-2	รูปคลื่นสัญญาณ DTMF หมายเลข 7	13
รูปที่ 3-3	*Two-Pole Resonator ของ Goertzel Algorithm*	16
รูปที่ 3-4	การทำ A-Law Companding ทางฝั่งค่าบวก	18
รูปที่ 4-1	ภาพรวมของ Application ที่ใช้ IALG ในการทำเป็น Algorithm Standard	20
รูปที่ 4-2	ลำดับในการ Call ฟังก์ชั่นของ IALG APIs ที่เป็นไปได้	21
รูปที่ 4-3	*การใช้ Supplementary APIs ในการติดต่อกับ IALG APIs*	22
รูปที่ 5-1	แผนภาพ ขั้นตอนในการสร้างสัญญาณโทรศัพท์	24
รูปที่ 5-2	สัญญาณที่ได้จากการสร้างสัญญาณ DTMF จำนวน 1 หมายเลข	29
รูปที่ 5-3	ลำดับการใช้งาน Library การสร้างสัญญาณ DTMF	29
รูปที่ 5-4	ขั้นตอนการทำงานของฟังก์ชั่นสร้างสัญญาณ DTMF	31
รูปที่ 5-5	การเปลี่ยน State ของ Library สร้างสัญญาณ DTMF	32
รูปที่ 5-6	ลำดับการใช้งาน Library การสร้างสัญญาณ R2 MF	35
รูปที่ 5-7	ขั้นตอนการทำงานของฟังก์ชั่นสร้างสัญญาณ R2 MF	36
รูปที่ 6-1	แผนภาพ ขั้นตอนในการตรวจจับสัญญาณโทรศัพท์	41
รูปที่ 6-2	การนำสัญญาณที่รับเข้ามาไปตรวจสอบ	41
รูปที่ 6-3	การนำ Goertzel Algorithm ไปใช้ในการตรวจจับความถี่ของสัญญาณ	43
รูปที่ 6-4	ลำดับการใช้งาน Library การตรวจจับสัญญาณ DTMF	47
รูปที่ 6-5	ค่าคงที่ ที่ใช้ในการกำหนดจังหวะของสัญญาณ DTMF	48
รูปที่ 6-6	ขั้นตอนการทำงานของฟังก์ชั่นตรวจจับสัญญาณ DTMF	49
รูปที่ 6-7	การทำงานของฟังก์ชั่นที่ทำหน้าที่ตัดสินว่าเป็น Pause Sample/ Non-Pause Sample	51
รูปที่ 6-8	การทำงานของฟังก์ชั่นที่ทำหน้าที่ตรวจสอบระดับค่าพลังงาน และหาความถี่ส่วนประกอบ	52
รูปที่ 6-9	ลำดับการใช้งาน Library การตรวจจับสัญญาณ R2 MF	55
รูปที่ 6-10	ขั้นตอนการทำงานของฟังก์ชั่นตรวจจับสัญญาณ R2 MF	56
รูปที่ 6-11	การดูช่วง Tone Duration และ Pause Duration ของสัญญาณ	58

รูปที่ 6-12	การทำงานของการตรวจสอบความถี่และระดับค่าพลังงาน ของสัญญาณ	61
รูปที่ 9-1	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 697 Hz ใช้ 250 Samples	83
รูปที่ 9-2	กราฟค่า Goertzel ที่ได้จากการใช้ความถีตรวจจับที่ 770 Hz ใช้ 250 Samples	83
รูปที่ 9-3	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 852 Hz ใช้ 250 Samples	84
รูปที่ 9-4	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 941 Hz ใช้ 250 Samples	84
รูปที่ 9-5	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1209 Hz ใช้ 250 Samples	85
รูปที่ 9-6	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1336 Hz ใช้ 250 Samples	85
รูปที่ 9-7	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1477 Hz ใช้ 250 Samples	86
รูปที่ 9-8	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1633 Hz ใช้ 250 Samples	86
รูปที่ 9-9	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1380 Hz ใช้ 250 Samples	87
รูปที่ 9-10	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1500 Hz ใช้ 250 Samples	87
รูปที่ 9-11	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1620 Hz ใช้ 250 Samples	88
รูปที่ 9-12	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1740 Hz ใช้ 250 Samples	88
รูปที่ 9-13	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1860 Hz ใช้ 250 Samples	89
รูปที่ 9-14	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1980 Hz ใช้ 250 Samples	89
รูปที่ 9-15	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 540 Hz ใช้ 250 Samples	90
รูปที่ 9-16	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 660 Hz ใช้ 250 Samples	90
รูปที่ 9-17	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 780 Hz ใช้ 250 Samples	91
รูปที่ 9-18	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 900 Hz ใช้ 250 Samples	91
รูปที่ 9-19	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1020 Hz ใช้ 250 Samples	92
รูปที่ 9-20	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1140 Hz ใช้ 250 Samples	92
รูปที่ 9-21	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 360 Hz ใช้ 100 Samples	96
รูปที่ 9-22	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 440 Hz ใช้ 100 Samples	97
รูปที่ 9-23	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 360 Hz รวมกับความถี่ต่างๆ ใช้ 100 Samples	98
รูปที่ 9-24	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 950 Hz ใช้ 100 Samples	99
รูปที่ 9-25	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1400 Hz ใช้ 100 Samples	99
รูปที่ 9-26	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1800 Hz ใช้ 100 Samples	100
รูปที่ 9-27	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1400 Hz ใช้ 20 Samples	100
รูปที่ 9-28	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1100 Hz ใช้ 16 Samples	101
รูปที่ 9-29	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 620 Hz ใช้ 100 Samples	101
รูปที่ 9-30	กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 480 Hz ใช้ 100 Samples	102

รูปที่ 11-1 การเกิดสัญญาณรบกวนแบบ FIR 15

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ปัจจุบันได้มีการนำ DSP Chip หรือ Digital Signal Processor Chip ซึ่งเป็นไมโครโปรเซสเซอร์ (Microprocessor) ประเภทหนึ่ง มาใช้ในงานด้านการประมวลผลสัญญาณดิจิตอลที่ต้องการการทำงานแบบ Real-time เนื่องจากเหตุผลทั้งทางด้านประสิทธิภาพการทำงานที่สูงของ DSP Chip, การใช้พลังงานที่ต่ำ, ความยืดหยุ่นในการโปรแกรมการทำงาน เป็นต้น

ในการสื่อสารทางโทรศัพท์นั้น จะมีรูปแบบสัญญาณที่ใช้ในการติดต่ออยู่หลายแบบ เช่น สัญญาณ DTMF (Dual Tone Multi-Frequency) ที่ใช้ในการส่งหมายเลขโทรศัพท์ปลายทางที่ต้องการ ไปยังระบบชุมสายโทรศัพท์อัตโนมัติ, และสัญญาณ CPT (Call Progress Tone) ที่ระบบชุมสายโทรศัพท์อัตโนมัติ ใช้ในการส่งสถานะ การโทรออก กลับมายังผู้โทรออก เป็นต้น ดังนั้น หากต้องการจะนำ DSP Chip ไปประยุกต์ใช้ในงานด้านที่เกี่ยวของกับสัญญาณเหล่านี้ เช่น ตู้ชุมสายโทรศัพท์ภายในระบบดิจิตอล, หรือระบบ VoIP (Voice over IP) จึงจำเป็นที่จะต้องให้ DSP Chip นั้นสามารถทำการสร้างสัญญาณ และตรวจจับสัญญาณต่างๆที่ใช้ในการสื่อสารเหล่านั้นให้ได้ โครงงานนี้จึงมีเป้าหมายในการพัฒนา Library ที่ใช้ในการสร้างและตรวจจับสัญญาณโทรศัพท์ดังต่อไปนี้ [1] สัญญาณ DTMF, [2] สัญญาณ R2 MF, และ [3] สัญญาณ CPT ซึ่งเป็นรูปแบบสัญญาณที่มีการใช้งานมาก ในการสื่อสารทางโทรศัพท์ โดยจะทำการพัฒนา Library ให้สามารถทำงานได้บน DSP Chip TMS320C6713 ของบริษัท Texas Instruments

ในอดีตนั้น การพัฒนา Library บน DSP Chip นั้น จะเป็นในลักษณะที่ไม่มีรูปแบบ โดยจะขึ้นอยู่กับทีมพัฒนาแต่ละ Library ทำให้เป็นการยากที่จะนำ Library ที่พัฒนาโดยทีมหนึ่ง ไปทำงานร่วมกับ Library ที่พัฒนาโดยอีกทีมงานหนึ่ง หากไม่ได้มีข้อกำหนดของระบบเอาไว้ก่อน เนื่องจากมักมีปัญหาความขัดแย้งที่เกิดจากการแย่งใช้ทรัพยากรของระบบ ทำให้ต้องเสียเวลาในการไปตรวจแก้ Code ดังนั้น เพื่อแก้ปัญหาดังกล่าว ทางบริษัท Texas Instruments จึงได้ออกมาตรฐานที่ใช้ในการพัฒนา Library ที่จะนำมาใช้กับ DSP Chip ของตน โดยเรียกว่า DSP Algorithm Standard ซึ่งในการพัฒนา Library ของโครงงานนี้ ก็จะพัฒนาตามรูปแบบนี้เช่นกัน เพื่อให้สามารถนำ Library ที่ได้ทำการพัฒนาขึ้นนี้ ไปทำงานร่วมกับ Library อื่นๆ (ที่ได้รับการพัฒนาตามรูปแบบ DSP Algorithm Standard เช่นกัน) ได้โดยไม่เกิดปัญหาในการทำงาน

1.2 ขอบเขตของโครงงาน

โครงงานนี้มีเป้าหมายในการพัฒนา Library ที่ใช้ในการสร้างและตรวจจับสัญญาณโทรศัพท์เพื่อที่จะนำไปทำงานบน DSP Chip TMS320C6713 ของบริษัท Texas Instruments โดยประเภทของสัญญาณที่จะต้องสร้างและตรวจจับในโครงงานนี้มีดังนี้ คือ [1] DTMF, [2] R2 MF, และ [3] CPT โดย

สัญญาณแต่ละประเภทนั้นก็มีรูปแบบต่างๆกันไป เช่น ความถี่ของสัญญาณที่ใช้ในการแทนค่า, ช่วงจังหวะของสัญญาณ เป็นต้น ดังนั้นในการสร้างและตรวจจับสัญญาณเหล่านี้นั้นจึงจำเป็นที่จะต้องทำการควบคุมให้เป็นไปตามมาตรฐานของสัญญาณแต่ละประเภท นอกจากนี้แล้ว เพื่อให้ Library ที่ทำการพัฒนาขึ้นสามารถที่จะถูกนำไป Re-use ได้ จึงมีความจำเป็นที่จะต้องพัฒนา Library ให้เป็นไปตามรูปแบบของ TMS320 DSP Algorithm Standard ซึ่งเป็นมาตรฐานการพัฒนา Library สำหรับ DSP Chip ตระกูล TMS320 ที่ทาง Texas Instruments กำหนดขึ้น โดยจะเป็นข้อบังคับในการเขียนโปรแกรม เช่น รูปแบบการทำ DMA, รูปแบบการใช้งานพื้นที่หน่วยความจำ เป็นต้น เพื่อประโยชน์ในการนำ Library หลายๆตัว (ที่เป็นไปตามรูปแบบของ TMS320 DSP Algorithm Standard) มา Re-use เป็นระบบใหญ่ และสามารถทำงานร่วมกันได้โดยไม่เกิดปัญหา

1.2.1 ขอบเขตของ Library สร้างสัญญาณ DTMF

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. สร้างสัญญาณ DTMF โดยอิงตามเอกสาร ITU-T Recommendation Q.24
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard
4. สามารถกำหนดจังหวะของการสร้างสัญญาณได้โดยผู้ใช้
5. สามารถกำหนดความถี่ส่วนประกอบที่ใช้ในการสร้างสัญญาณได้โดยผู้ใช้
6. สามารถกำหนดค่าตารางไซน์ที่ใช้ในการสร้างสัญญาณได้โดยผู้ใช้

1.2.2 ขอบเขตของ Library ตรวจจับสัญญาณ DTMF

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. ตรวจจับสัญญาณ DTMF โดยอิงตามเอกสาร ITU-T Recommendation Q.24
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard
4. สามารถกำหนดจังหวะของการตรวจจับสัญญาณได้โดยผู้ใช้
5. สามารถกำหนดค่า Coefficient ที่ใช้การการตรวจจับสัญญาณได้โดยผู้ใช้
6. สามารถกำหนดค่า THRESHOLD ที่ใช้การการตรวจจับสัญญาณได้โดยผู้ใช้

1.2.3 ขอบเขตของ Library สร้างสัญญาณ R2 MF

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. สร้างสัญญาณ R2 MF โดยอิงตามเอกสาร ITU-T Recommendation Q.455
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard
4. สามารถกำหนดความถี่ส่วนประกอบที่ใช้ในการสร้างสัญญาณได้โดยผู้ใช้
5. สามารถกำหนดค่าตารางไซน์ที่ใช้ในการสร้างสัญญาณได้โดยผู้ใช้

1.2.4 ขอบเขตของ Library ตรวจจับสัญญาณ R2 MF

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. ตรวจจับสัญญาณ DTMF โดยอิงตามเอกสาร ITU-T Recommendation Q.455
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard
4. สามารถกำหนดค่า Coefficient ที่ใช้การการตรวจจับสัญญาณได้โดยผู้ใช้
5. สามารถกำหนดค่า THRESHOLD ที่ใช้การการตรวจจับสัญญาณได้โดยผู้ใช้

1.2.5 ขอบเขตของ Library สร้างสัญญาณ CPT

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. ตรวจจับสัญญาณ DTMF โดยอิงตามเอกสาร ITU-T Recommendation E.180/ Q.35
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard
4. สามารถกำหนดค่าตารางไซน์ที่ใช้ในการสร้างสัญญาณได้โดยผู้ใช้

1.2.6 ขอบเขตของ Library ตรวจจับสัญญาณ CPT

1. ทำงานที่ Sampling rate เท่ากับ 8000 Hz
2. ตรวจจับสัญญาณ DTMF โดยอิงตามเอกสาร ITU-T Recommendation E.180/ Q.35
3. พัฒนาตามรูปแบบ TMS320 DSP Algorithm Standard

1.3 วัตถุประสงค์ของโครงงาน

1. พัฒนา Library ในการสร้างและตรวจจับสัญญาณโทรศัพท์ DTMF, R2 MF, CPT บน DSP Chip TMS320C6713
2. ศึกษามาตรฐานในการสร้างและตรวจจับสัญญาณ DTMF, R2 MF, CPT
3. ศึกษาวิธีการพัฒนาอัลกอริทึม ที่ใช้ในการสร้างและตรวจจับสัญญาณ
4. ศึกษาการเขียนโปรแกรมในรูปแบบ TMS320 Algorithm Standard

1.4 เป้าหมายของโครงงาน

1. พัฒนาอัลกอริทึมในการสร้างและตรวจจับสัญญาณ DTMF, R2 MF, และ CPT บน PC โดยใช้ภาษา C Programming Language ให้สามารถทำการสร้างและตรวจจับสัญญาณให้ได้ตามมาตรฐาน
2. ตรวจสอบความถูกต้องของการสร้างและตรวจจับสัญญาณดังกล่าว(บน PC) และทำการปรับปรุงการทำงานของอัลกอริทึมให้เป็นไปตามมาตรฐาน

3. พัฒนาอัลกอริทึมในการสร้างและตรวจจับสัญญาณ DTMF, R2 MF, และ CPT บน DSP Chip โดยประยุกต์จากอัลกอริทึมที่ทำงานบน PC ให้เป็นไปตาม TMS320 Algorithm Standard และให้สามารถทำการสร้างและตรวจจับสัญญาณให้ได้ตามมาตรฐาน
4. ตรวจสอบความถูกต้องของการสร้างและตรวจจับสัญญาณดังกล่าว(บน DSP) และทำการปรับปรุงการทำงานของอัลกอริทึมให้เป็นไปตามมาตรฐาน
5. นำอัลกอริทึมที่ทำงานได้ถูกต้องแล้วบน DSP ไปใช้ในการพัฒนา Library ในการสร้างและตรวจจับสัญญาณโทรศัพท์ DTMF, R2 MF, CPT บน DSP Chip TMS320C6713 โดยยึดตามรูปแบบ TMS320 Algorithm Standard

บทที่ 2
ทฤษฎี สัญญาณทางโทรศัพท์

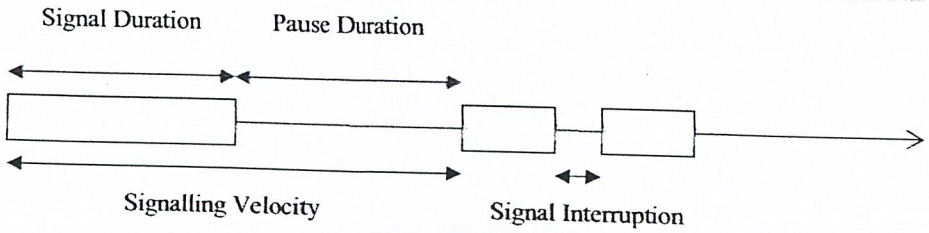
2.1 สัญญาณ DTMF

DTMF นั้น ย่อมาจาก Dual Tone Multi-Frequency โดยจะประกอบด้วยสัญญาณไซน์ (Sinusoid) จำนวน 2 สัญญาณ ที่มีความถี่ต่างกันมารวมกัน โดยจะมาจาก High-Frequency Group ได้แก่ 1209 Hz, 1336 Hz, 1477 Hz, 1633 Hz และ Low-Frequency Group ได้แก่ 697 Hz, 770 Hz, 852 Hz, 941 Hz ซึ่งถูกนำมาใช้มากในเครื่องโทรศัพท์ระบบกดปุ่ม ซึ่งจะใช้รหัส DTMF ในการระบุเลขหมายปลายทางไปยังชุมสายระบบอัตโนมัติ แทนที่จะส่ง Pulse ไปแบบเครื่องโทรศัพท์ระบบหมุน ซึ่งยุ่งยากและยังทำงานช้ากว่า อีกทั้งปัจจุบัน DTMF ยังถูกนำมาประยุกต์ใช้ในระบบสารสนเทศผ่านทางโทรศัพท์ โดยผู้ใช้งานส่งรหัส DTMF ไปยังระบบเพื่อใช้แทนการเลือกเมนูการทำงานต่างๆ เป็นต้น

	1209 Hz	**1336 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

ตารางที่ 2-1 คู่ความถี่ที่ใช้ในการสร้างสัญญาณ DTMF

ในการตรวจจับสัญญาณนั้นจะทำการตรวจสอบจังหวะของสัญญาณและระดับค่าพลังงานของสัญญาณ ให้เป็นไปตามมาตรฐาน AT&T ที่ระบุใน ITU-T Recommendation Q.24 ซึ่งเป็นข้อกำหนดสำหรับ Multifrequency Push-Button Signal Reception โดยมีข้อกำหนดที่สำคัญๆ ดังตารางด้านล่าง



รูปที่ 2-1 ความหมายของคำต่างๆ ที่ใช้ใน ITU-T Recommendation Q.24

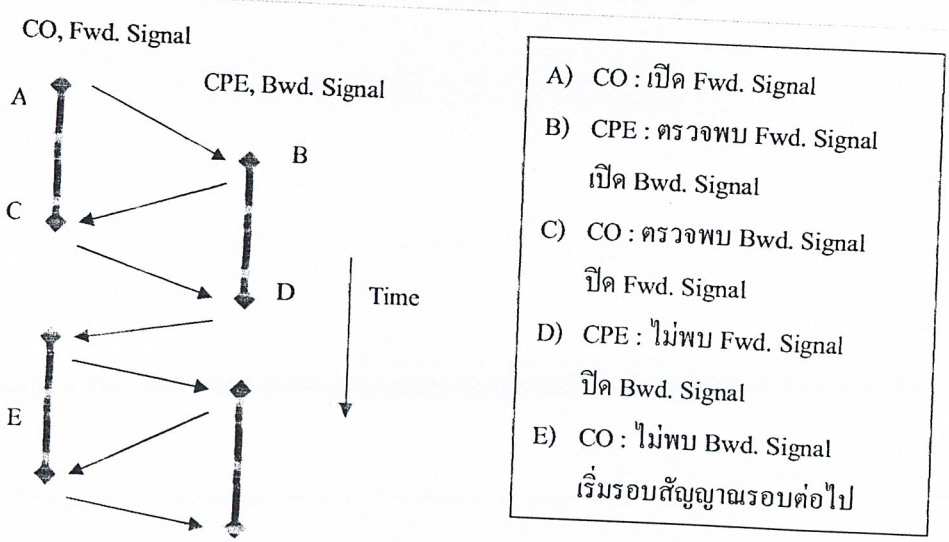
ข้อกำหนด	ITU-T Q.24, AT&T Values
Frequency Deviation	≤ ±1.5% must accept ≥ ±3.5% must reject
Signal Duration	≥ 40 mS must accept ≤ 23 mS must reject
Pause Duration	≥ 40 mS detect 2 digits ≤ 10 mS allowable Signal Interruption
Signalling Velocity	≥ 93 mS/digit
Accept Levels	0 to -25 dBm must accept ≤ -55 dBm reject

ตารางที่ 2-2 ข้อกำหนดในการตรวจสอบสัญญาณ DTMF

(adapted from ITU-T Recommendation Q.24 (AT&T))

2.2 สัญญาณ R2 MF

สัญญาณ R2 MF (Multi-Frequency) นั้นเป็นระบบการส่งสัญญาณระหว่างประเทศที่ใช้ในภูมิภาคยุโรป และภูมิภาคเอเชีย เพื่อใช้ในการส่งหมายเลข และข้อมูลที่สำคัญไปยังปลายทาง โดย R2 MF ที่ใช้ในการควบคุมการทำ Call Setup จะเรียกว่า Interregister Signals โดยสัญญาณจะถูกส่งระหว่าง Central Office (CO) และ Customer Premises Equipment (CPE) โดยสัญญาณที่ส่งจาก CO จะเรียกว่า Forward Signal และสัญญาณที่ส่งมาจาก CPE จะเรียกว่า Backward Signals โดยตัวอย่างของการสื่อสารจะแสดงดังภาพต่อไปนี้



รูปที่ 2-2 ตัวอย่างการส่งสัญญาณ R2 MF Interregister Signaling

โดยรหัสแต่ละตัวของสัญญาณ R2 MF นั้น จะสร้างมาจากคู่ความถี่ที่ต่างกัน โดยใช้ 6 ความถี่พื้นฐาน (1380, 1500, 1620, 1740, 1860, และ 1980 Hz) ใน Forward Signals และ อีก 6 ความถี่พื้นฐาน (1140, 1020, 900, 780, 660, และ 540 Hz) สำหรับ Backward Signals โดยมีความสัมพันธ์ดังตารางต่อไปนี้

	1500 Hz	**1620 Hz**	**1740 Hz**	**1860 Hz**	**1980 Hz**
1380 Hz	1	2	4	7	11 (A')
1500 Hz		3	5	8	12 (B')
1620 Hz			6	9	13 (C')
1740 Hz				10	14 (D')
1860 Hz					15 (E')

ตารางที่ 2-3 R2 MF, Forward Signals

	1020 Hz	**900 Hz**	**780 Hz**	**660 Hz**	**540 Hz**
1140 Hz	1	2	4	7	11 (A')
1020 Hz		3	5	8	12 (B')
900 Hz			6	9	13 (C')
780 Hz				10	14 (D')
660 Hz					15 (E')

ตารางที่ 2-4 R2 MF, Backward Signals

ในการตรวจจับสัญญาณนั้นจะทำการตรวจสอบจังหวะของสัญญาณ และ ระดับค่าพลังงานของสัญญาณ ให้เป็นไปตามมาตรฐานที่ระบุใน ITU-T Recommendation Q.455 ซึ่งเป็นข้อกำหนดสำหรับ Receiving Part of the Multifrequency Equipment โดยมีข้อกำหนดที่สำคัญๆ ดังตารางด้านล่าง

ข้อกำหนด		**CCITT/ITU Values (R2 MF)**
Frequency Deviation		≤ ± 10 Hz must accept
Signal Duration	Operation	≥ 7 mS
	Non-Operation	< 7 mS
Power Level	Operation	-5 to -35 dBm
	Non-Operation	Less than -38 dBm

ตารางที่ 2-5 ข้อกำหนดในการตรวจสอบสัญญาณ R2 MF

(adapted from ITU-T Recommendation Q.455))

2.3 สัญญาณ CPT

สัญญาณ CPT หรือ Call Progress Tone เป็นสัญญาณที่ส่งกลับมาจากชุมสายโทรศัพท์ ไปยังผู้ใช้งานเพื่อบ่งบอกสถานะ การเชื่อมต่อทางโทรศัพท์ในขณะนั้น โดยมีทั้งหมด 14 สัญญาณ ดังนี้

1. Dial Tone เป็นการบอกผู้ใช้ว่า ชุมสายพร้อมที่จะให้บริการ
2. Ring Tone เป็นการบอกผู้ใช้ว่า ได้ทำการติดต่อแล้ว และส่งสัญญาณไปที่ปลายทาง
3. Busy Tone เป็นการบอกผู้ใช้ว่า หมายเลขปลายทางสายไม่ว่าง
4. Congestion Tone เป็นการบอกผู้ใช้ว่า ช่องสัญญาณเต็ม
5. Special Information Tone เป็นการบอกผู้ใช้ว่า ไม่สามารถติดต่อหมายเลขปลายทางได้ ที่ไม่ใช้เหตุผลว่า Busy หรือ Congestion
6. Warning Tone เป็นการบอกผู้ใช้ว่า การสนทนานี้ถูกบันทึกอยู่
7. Payphone Tone (Payphone Recognition Tone) เป็นการบอก Operator ที่ชุมสายว่า ผู้โทร หรือปลายทาง เป็นเบอร์โทรของ Payphone
8. Call Waiting Tone ขณะที่ผู้ใช้กำลังใช้สายอยู่ สัญญาณ Call Waiting จะส่งมาบอกว่ามีคนกำลังโทรเข้า (เช่นสัญญาณเตือน สายเรียกซ้อน)
9. Recall Dial Tone เป็นการบอกผู้ใช้ว่า พร้อมที่จะเพิ่มการติดต่อกับเบอร์ปลายทาง ในขณะที่ คุยกับปลายทางอื่นอยู่
10. Intercept Tone เป็นการบอกผู้ใช้ว่า เบอร์ที่โทรออกไม่ถูกต้อง หรือบริการที่ขอใช้ไม่มีอยู่จริง
11. Confirmation Tone เป็นการบอกผู้ใช้ว่ายอมรับให้ผู้ใช้ ใช้ Feature เกี่ยวกับการ Active และการ Cancel
12. Comfort Tone เป็นการบอกผู้ใช้ว่าการ Call นั้นกำลังถูก Process อยู่ให้ผู้ใช้รอ
13. Busy Verification เป็นการเตือนผู้ใช้ที่คุยพร้อมกันหลายๆคนอยู่ว่าจะมีคู่สนทนาเพิ่มขึ้น
14. Caller Waiting Tone เป็นการบอกผู้ใช้ว่าหมายเลขปลายทางไม่ว่าง และ Call Waiting Service จะทำการ Active

สำหรับมาตรฐานที่ใช้ในการสร้างและตรวจจับสัญญาณนั้น จะยึดตามเอกสาร ITU-T Recommendation E.180/Q.35 โดยจะมีค่าความถี่ และจังหวะสัญญาณดังต่อไปนี้

2.3.1 มาตรฐาน CPT ฝั่งผู้ส่ง (Transmitter)

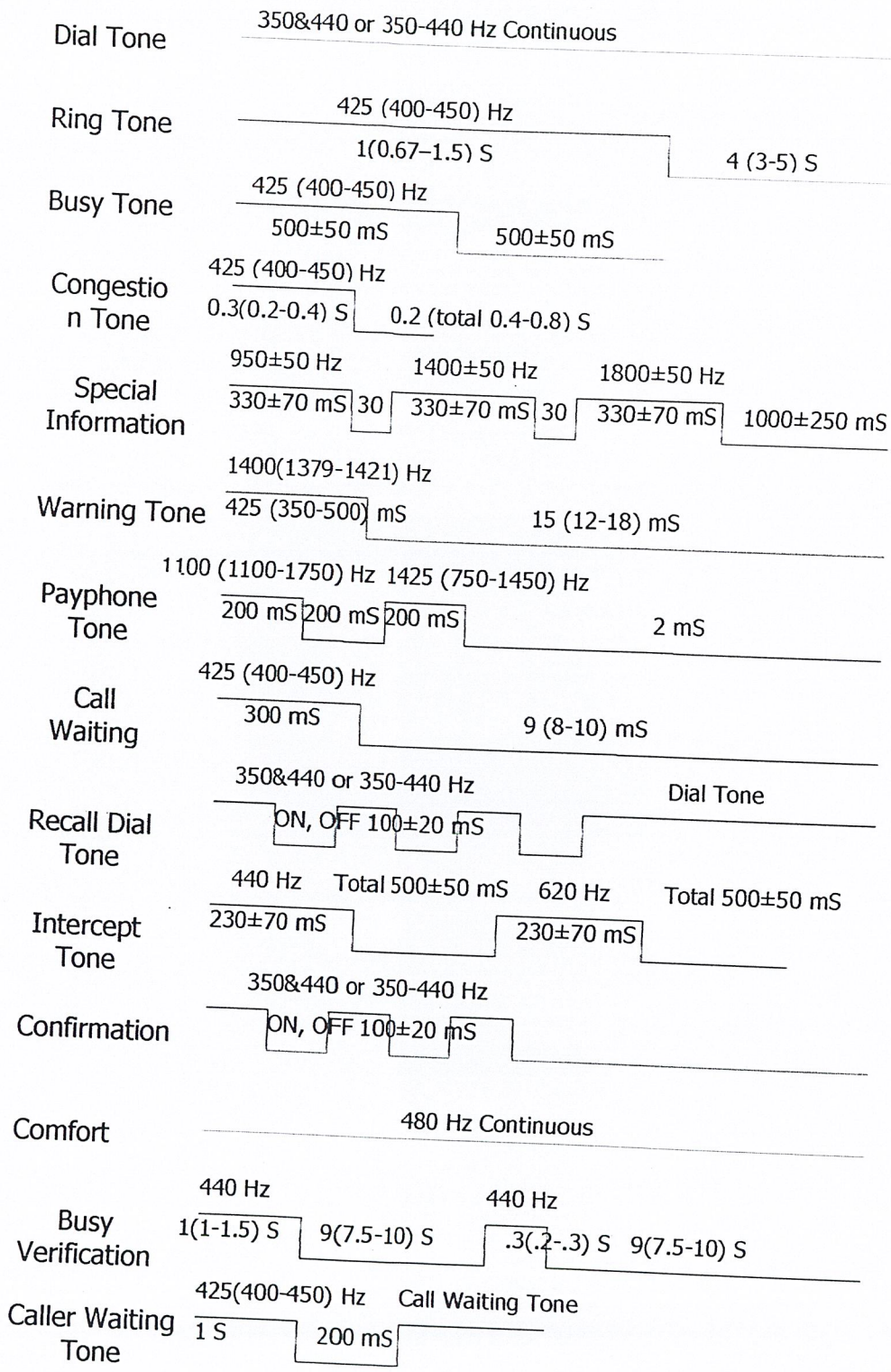
Tone ID	Tone Name	Frequencies	Cadence	Cycles
0	Dial tone	350 & 440 Hz	Continuous (Dual Tone)	N/A
1	Ring Tone	425 Hz	ON: 1 s, OFF: 4 s	Variable*
2	Busy Tone	425 Hz	ON:OFF:0.5 sec	Variable (3 cycles)
3	Congestion	Tone 425 Hz	ON:0.3sec, OFF:0.2sec	Variable (3 cycles)
4	Special Information	F1=950 Hz, F2=1400 Hz F3=1800 Hz	ON1: 330 ms, OFF1: 30 ms ON2: 330 ms, OFF2: 30 ms ON3: 330 ms, Silence: 1second (1 burst: 3 ON-OFF periods)	Variable (3 bursts)
5	Warning Tone	1400 Hz	ON: 425 ms, OFF:15 s	Variable (3 cycles)
6	Payphone Tone	F1=1100 Hz, F2=1425 Hz	ON1: 200 ms, OFF1: 200 ms ON2: 200 ms, Silence 2 seconds	Variable (3 cycles)
7	Call Waiting	425 Hz	ON:300 ms ,OFF 9 seconds	Variable (2 cycles)
8	Recall Dial Tone	350 & 440 Hz	ON, OFF: 100 ms (Dual tone: 3 cycles) + Dial	Fixed** (3 cycles)
9	Intercept Tone	F1: 440 Hz F2: 620 Hz	ON1:230 ms , Total1:500 ms ON2:230 ms , Total2:500 ms (1burst: ON1/OFF1+ON2/OFF2)	Variable (3 cycles)
10	Confirmation	350 & 440 Hz	ON: 100 ms, OFF: 100 ms (Dual Tone)	Fixed (3 cycles)
11	Comfort	480 Hz	Continuous	Tone N/A
12	Busy Verification	440 Hz	ON1: 1.5 s, OFF1: 9 s ON2: 300 ms, OFF2: 9 s	-
13	Caller Waiting	425 Hz	Ring (ON: 1000 ms +OFF: 200 ms) +Call waiting (2cycles)	3 cycles (including Ring Tone)
*The number of repetitions are programmable. Typical value is given brackets. **The number of repetitions are fixed as specified in brackets.				

ตารางที่ 2-6 มาตรฐาน CPT ฝั่งผู้ส่ง

2.3.2 มาตรฐาน CPT ฝั่งผู้รับ (Receive

Tone ID	Tone Name	Frequencies	Cadence
0	Dial Tone	F1=340-425Hz & F2=400-450Hz Or F=340-450 Hz	Continuous Tone
1	Ring Tone	F=400-450Hz	ON:0.67-1.5Sec,OFF:3-5Sec
2	Busy Tone	F=400-450Hz	ON:OFF:(500±50mSec)
3	Congestion Tone	F=400-450Hz	ON:0.2-0.4sec,Total:0.4-0.8(Sec)
4	Special Information	F1=950±50Hz, F2=1400±50Hz, F3=1800±50Hz	ON:330±70Sec, OFF:30msec
5	Warning Tone	F=1379-1421Hz	ON:350-500ms, OFF:12-18Sec
6	Payphone Tone	F1=1100-1750Hz F2=750-1450Hz	F1:200ms(ON1) : 200ms(OFF) : F2-200ms(ON2) : Silence of 2 seconds
7	Call Waiting	F=400-450Hz	ON:300-500ms,OFF:8-10Sec
8	Recall Dial	F1=340-425Hz & F2=400-450Hz	ON and OFF: 100±20ms followed by Dial Tone
9	Intercept Tone	F1-440 & F2-620Hz	ON:230±70ms Total:500±50ms
10	Confirmation	F1=340-425Hz & F2=400-450Hz	ON:100ms, OFF:100ms
11	Comfort	Tone F1=480Hz	Continuous
12	Busy Verification	F1=440Hz	ON1 1-1.5s, ON2 0.2-0.3Sec OFF 7.5-10Sec
13	Caller Waiting	F1=400-450Hz	ON1: 1000 ms OFF1: 0-200 ms ON2: Call Waiting

ตารางที่ 2-7 มาตรฐาน CPT ฝั่งผู้รับ



รูปที่ 2-3 ค่าความถี่ และจังหวะสัญญาณ ของ สัญญาณ CPT

โดยค่านอกวงเล็บคือค่าที่ใช้สร้างสัญญาณ และค่าในวงเล็บคือค่าจะที่ต้องตรวจจับให้ได้

บทที่ 3
ทฤษฎี การสร้างและการตรวจจับสัญญาณ

3.1 การสร้างสัญญาณ

ในการสร้างสัญญาณทั้ง DTMF, R2 MF, และ CPT นั้น จะสร้างจากการผสมสัญญาณไซน์ ที่มีความถี่ต่างกัน สมการต่อไปนี้แสดงการนำสัญญาณไซน์ 2 สัญญาณ ที่มีความถี่ต่างกัน (f1, f2) มารวมกันเพื่อสร้างแต่ละ Sample (F(t))ของสัญญาณที่ต้องการสร้างขึ้น

$$F(t) = A_1 \sin(2\P f_1 t) + A_2 \sin(2\P f_2 t)$$

F(t) : แต่ละ Sample ของสัญญาณที่ต้องการสร้าง,โดยที่ค่า t จะเริ่มตั้งแต่ {1/8000, 2/8000, ..., 1} จากนั้นก็จะวนไปที่ 1/8000 อีกครั้ง (กรณี Sampling rate = 8000 Hz)

A_1 : ค่าแอมปลิจูดสูงสุดของสัญญาณไซน์ สัญญาณที่ 1

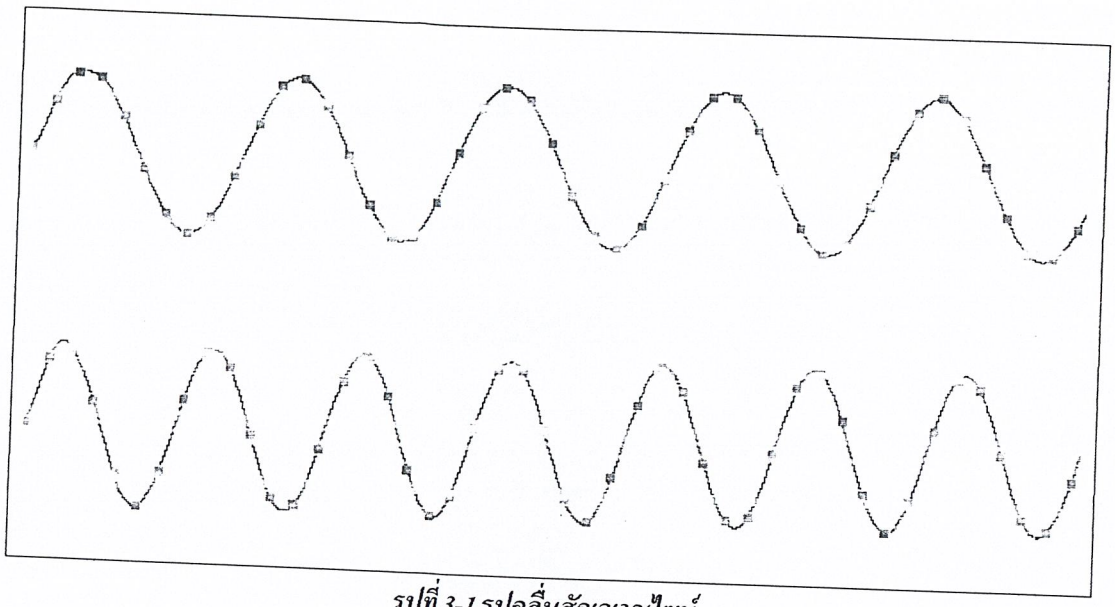
f_1 : ความถี่ของสัญญาณไซน์ สัญญาณที่ 1

A_2 : ค่าแอมปลิจูดสูงสุดของสัญญาณไซน์ สัญญาณที่ 2

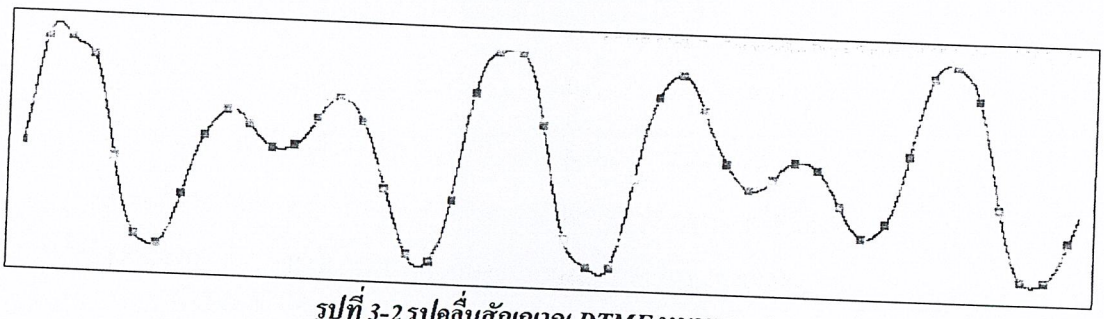
f_2 : ความถี่ของสัญญาณไซน์ สัญญาณที่ 2

ตารางที่ 3-1 สมการหลักที่ใช้ในการสร้างสัญญาณ

ในส่วนของการสร้างสัญญาณนั้น จะสร้างสัญญาณไซน์ส่วนประกอบขึ้นมาก่อน แล้วนำทั้งสองสัญญาณมารวมกันเพื่อสร้างสัญญาณที่ต้องการ เช่น ต้องการสร้างสัญญาณ DTMF หมายเลข 7 ก็จะทำการสร้างสัญญาณไซน์ความถี่ 1209 Hz, และ สร้างสัญญาณไซน์ความถี่ 852 Hz แล้วนำสัญญาณไซน์ส่วนประกอบทั้งสอง มารวมกัน



รูปที่ 3-1 รูปคลื่นสัญญาณไซน์
รูปบนเป็นสัญญาณไซน์ความถี่ 852 Hz, รูปล่างเป็นสัญญาณไซน์ความถี่ 1209 Hz



รูปที่ 3-2 รูปคลื่นสัญญาณ DTMF หมายเลข 7
เกิดจากการผสมคลื่นสัญญาณไซน์ความถี่ 852 Hz และ 1209 Hz เข้าด้วยกัน

3.2 การตรวจจับสัญญาณ

ในการตรวจสอบว่ามีสัญญาณไซน์ส่วนประกอบ (Sinusoid Composite Signal) ความถี่ใดบ้างที่ประกอบเข้ามาในสัญญาณนั้นสามารถตรวจสอบได้โดยทำการนำชุด Sample ของสัญญาณมาทำการแปลง จาก Time Domain ไปเป็น Frequency Domain โดยใช้การประมวลผลทางคณิตศาสตร์ ซึ่งในการพัฒนาส่วนตรวจจับสัญญาณนั้น จะใช้ Goertzel Algorithm ซึ่งได้รับการปรับปรุงมาจากการทำ Discrete Fourier Transform (DFT) โดยอาศัยคุณสมบัติ Periodicity ของ DFT เพื่อลดการคำนวณ เช่นเดียวกับที่ใช้ในการทำ Fast Fourier Transform (FFT) แต่เนื่องจากการตรวจสอบความถี่นั้น จะคำนึงแต่ค่า Magnitude ของสัญญาณ โดยไม่คำนึงถึง Phase ของสัญญาณ ดังนั้น ใน Goertzel Algorithm นั้นจะไม่นำ Phase ของสัญญาณมาทำการคำนวณ เพื่อลดเวลาที่ใช้ในการทำงาน

จากสมการ DFT

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi kn}{N}}$$

$$X(k)=e^{j\frac{2\pi kN}{N}}\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi kn}{N}} \quad , e^{j\frac{2\pi kN}{N}}=1$$

เขียนใหม่ได้ดังนี้

ให้ $h_k(N-n)=e^{-j\frac{2\pi k(n-N)}{N}}$

ได้ $$X(k)=\sum_{n=0}^{N-1}x(n)h_k(N-n)$$

Convolution $$y_k(m)=\sum_{n=0}^{N-1}x(n)h_k(m-n)$$

$$=x(m)*h(m)$$

เพราะฉะนั้น $$X_k(N)=y_k(m)\big|_{m=N}$$

หา Z-transform $Y(z)=X(z)\cdot H(z)$

$\alpha^n u(n)$ Z-transform $\frac{1}{1-\alpha z^{-1}}$ $\quad : \alpha=e^{\frac{j2\pi k}{N}}$

$$H(z)=\frac{1}{1-e^{j\frac{2\pi k}{N}}z^{-1}}$$

$$Y_k(z)=\frac{X(z)}{1-e^{j\frac{2\pi k}{N}}z^{-1}}$$

ต้องการหลีกเลี่ยงการคูณจำนวนเชิงซ้อน จึงคูณ Conjugate

$$=\frac{X(z)}{1-e^{j\frac{2\pi k}{N}}z^{-1}}\frac{(1-e^{-j\frac{2\pi k}{N}}z^{-1})}{(1-e^{-j\frac{2\pi k}{N}}z^{-1})}$$

ผลคูณของส่วน

$$(1-e^{j\frac{2\pi k}{N}}z^{-1})(1-e^{-j\frac{2\pi k}{N}}) = 1-e^{-j\frac{2\pi k}{N}}z^{-1}-e^{j\frac{2\pi k}{N}}z^{-1}+e^{j\frac{2\pi(k-k)}{N}}z^{-2}$$

$$= 1-(e^{j\frac{2\pi k}{N}}+e^{-j\frac{2\pi k}{N}})z^{-1}+z^{-2}$$

เมื่อ $e^{j\theta}+e^{-j\theta}=2\cos\theta$ $$= 1-2\cos(\frac{2\pi k}{N})z^{-1}+z^{-2}$$

จะได้

$$Y(z) = X(z)\frac{(1-e^{-j\frac{2\pi k}{N}}z^{-1})}{1-2\cos(\frac{2\pi k}{N})z^{-1}+z^{-2}}$$

$$\frac{Y(z)}{P(z)}\cdot\frac{P(z)}{X(z)} = \frac{(1-e^{-j\frac{2\pi k}{N}}z^{-1})}{1-2\cos(\frac{2\pi k}{N})z^{-1}+z^{-2}}$$

แบ่งเป็น 2 ส่วน คือ

1. $\frac{P(z)}{X(z)}$

$$P(z) = X(z)\frac{1}{1-2\cos(\frac{2\pi k}{N})z^{-1}+z^{-2}}$$

$$X(z) = P(z)-2\cos(\frac{2\pi k}{N})z^{-1}P(z)+z^{-2}P(z)$$

$$p(n) = x(n)+2\cos(\frac{2\pi k}{N})p(n-1)-p(n-2)$$

ให้ $p(n)=Q_n$

$$\boxed{Q_n = x(n)+2\cos(\frac{2\pi k}{N})Q_{n-1}-Q_{n-2}}$$

Feedback stage จะต้องทำซ้ำเป็นจำนวน N ครั้ง

2. $\frac{Y(z)}{P(z)}$

$$Y(z) = (1-e^{-j\frac{2\pi k}{N}}z^{-1})P(z)$$

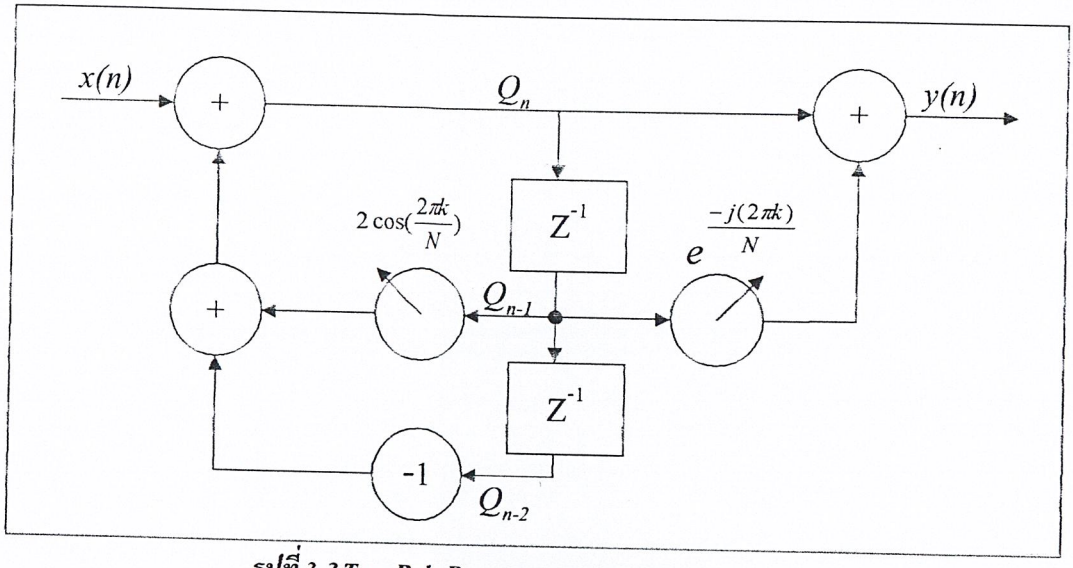
$$y(n) = p(n)-e^{-j\frac{2\pi k}{N}}p(n-1)$$

$$\boxed{y_k(n)\big|_{n=N} = X(k) = Q_N - Q_{N-1}e^{\frac{-j(2\pi k)}{N}},\ n=N}$$

Feedforward stage จะทำเพียงครั้งเดียว

และ $e^{-j\theta}$ แล้ว $$\boxed{|X(k)|^2 = Q_N^2 + Q_{N-1}^2 - 2\cos\left(\frac{2\pi k}{N}\right)Q_N Q_{N-1}}$$

จากสมการ Feedback และ Feedforward สามารถนำมาเขียนโครงสร้างของ Two-pole resonator เพื่อใช้คำนวณ DFT Sample อันดับที่ k ได้ดังนี้



รูปที่ 3-3 Two-Pole Resonator ของ Goertzel Algorithm

Feed back stage จะทำเป็นจำนวน N ครั้ง:

$$Q_n = x(n) + 2\cos(\frac{2\pi k}{N})Q_{n-1} - Q_{n-2} \quad \text{..... (1)}$$

Feed forward stage จะคิดแค่ครั้งเดียว:

$$y_k(n)\,|_{n=N} = X(k) = Q_N - Q_{N-1}e^{\frac{-j(2\pi k)}{N}},\ n=N \quad \text{..... (2)}$$

การตรวจจับนั้นสนใจค่า Magnitude แต่จะไม่สนใจค่า Phase จึงจะใช้แต่ค่า Magnitude ของสัญญาณ โดยใช้สมการที่ 3

$$|y_k(n)|^2 = |X(k)|^2 = Q^2_N + Q^2_{N-1} - 2\cos(\frac{2\pi k}{N})\,Q_N Q_{N-1} \quad \text{..... (3)}$$

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โดยค่า Coefficient, $2\cos(\frac{2\pi k}{N})$, นั้นสามารถหาได้จาก

$k = N(\frac{f_0}{f_s})$, โดยที่ f_0 = ความถี่ที่ต้องการจะตรวจจับ, f_S = Sampling Rate

จะได้ Coefficient, $2\cos(\frac{2\pi k}{N}) = 2\cos(\frac{2\pi f_0}{f_s})$

จากที่ได้อธิบายไปทั้งหมดในส่วนนี้ สามารถนำมาสรุปเป็นกลุ่มของสมการที่จะต้องนำไปใช้ในการตรวจจับสัญญาณได้ดังนี้

สมการหลักที่ใช้ในการตรวจจับ:

$$Q_n = x(n) + 2\cos(\frac{2\pi k}{N})Q_{n-1} - Q_{n-2} \qquad \text{(สมการ 3-1)}$$

$$|y_k(n)|^2 = |X(k)|^2 = Q^2_n + Q^2_{n-1} - 2\cos(\frac{2\pi k}{N})\, Q_n Q_{n-1} \qquad \text{(สมการ 3-2)}$$

สมการที่ใช้ในการหาค่า Coefficient เพื่อนใช้ในการตรวจจับแต่ละความถี่:

$$\text{Coefficient, } 2\cos(\frac{2\pi k}{N}) = 2\cos(\frac{2\pi f_0}{f_s}) \qquad \text{(สมการ 3-1)}$$

สมการ 3-1 จะถูกใช้ในการคำนวณกับแต่ละ Sample ของสัญญาณ

สมการ 3-2 จะถูกนำมาใช้คำนวณ หลังจากที่คำนวณ Sample สุดท้ายของสัญญาณด้วย สมการ 3-1 แล้ว

สมการ 3-3 จะถูกใช้ในการคำนวณหาค่า Coefficient สำหรับแต่ละความถี่ของสัญญาณไซน์ที่ต้องการทำการตรวจจับ โดยจะนำค่า Coefficient ที่ได้ ไปใช้ในสมการ 3-1 และ สมการ 3-2

Q_n : ผลลัพธ์ที่ได้จาก Sample ปัจจุบัน (Sample n)

x(n) : ค่าของ Sample ปัจจุบัน (Sample n)

$2\cos(\frac{2\pi k}{N})$: Coefficient, สามารถคำนวณได้จากสมการ 3-3

Q_{n-1} : ผลลัพธ์ที่ได้จาก Sample ก่อนหน้า ของ Sample **Q_n**

Q_{n-2} : ผลลัพธ์ที่ได้จาก Sample ก่อนหน้า ของ Sample **Q_{n-1}**

f_0 : ความถี่ที่ต้องการตรวจจับ

f_S : Sampling rate (Hz)

ตารางที่ 3-2 สมการจาก Goertzel Algorithm ที่ใช้ในการสร้างส่วนตรวจจับสัญญาณ

61481

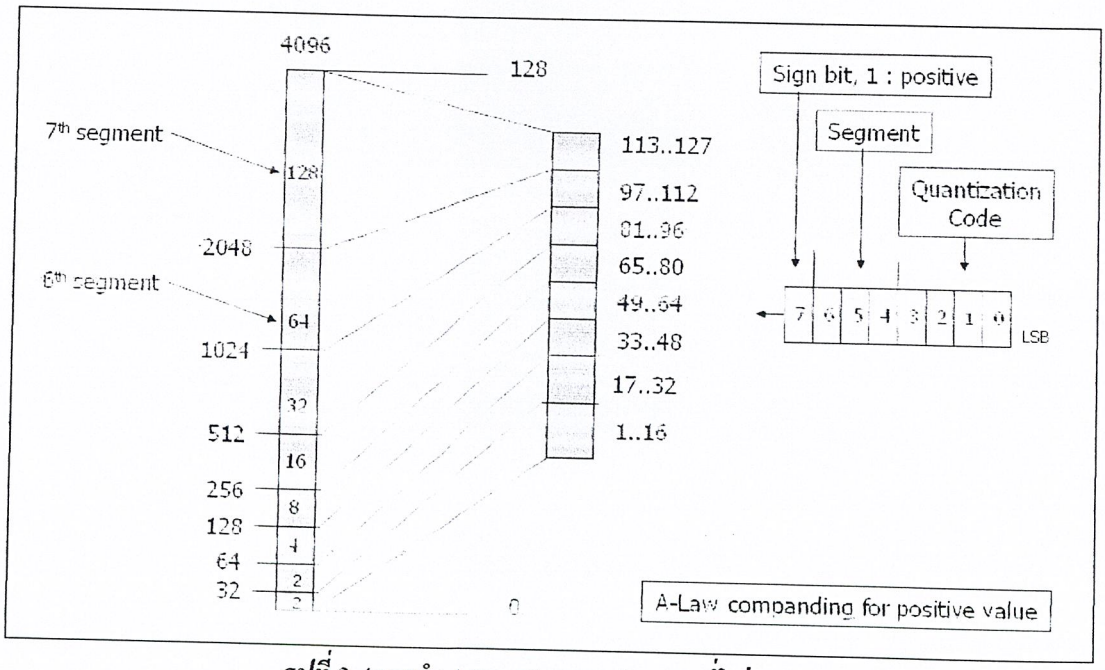
3.3 A-Law Companding

A-Law Companding (Compression and Expanding) เป็นรูปแบบการทำ Non-Uniform Quantization ที่ใช้ในการลดขนาดสัญญาณ (Compression) จาก 12 Magnitude Bits + 1 Sign Bit (±4095) ไปเป็น 7 Magnitude Bits + 1 Sign Bits (±127) และขยายกลับมา (Expanding)

การทำ Companding เช่น A-Law Companding นี้ได้ถูกนำมาใช้ในการส่งข้อมูลผ่าน Trunk ระหว่างประเทศ (ยกเว้นบางประเทศ เช่น ประเทศในทวีปอเมริกาเหนือ และ ญี่ปุ่น เป็นต้น ซึ่งจะใช้ การ Companding แบบ μ-Law Companding) เพื่อประโยชน์หลายๆด้าน เช่น ลดปัญหา Cross-Talk ในระบบ FDM เนื่องจากขนาดแอมปลิจูดของสัญญาณที่สูงเกินไป, ปรับปรุง SNR ให้ดีขึ้น เป็นต้น โดยมาตรฐานของ A-Law Companding นั้น จะใช้ตามเอกสาร ITU-T Recommendation G.711 "Pulse Code Modulation (PCM) Of Voice Frequencies"

เมื่อได้ Sample ของสัญญาณที่ผ่านการบีบอัดแล้ว ก่อนทำการส่ง (Transmit) Sample นั้นออกไป จะต้องทำการ Invert Even Bit ของ Sample นั้นๆก่อนโดยการ XOR ด้วย 0x55 และเมื่อได้รับ Sample ใดๆ เข้ามา ก่อนการทำ Expanding นั้น จะต้อง XOR ด้วย 0x55 กลับคืน ก่อนที่จะทำการ Expand ด้วยเช่นกัน

จากรูป แสดงการทำ A-Law Companding ทางฝั่งค่าบวก หากต้องการจะทำกับทางฝั่งค่าติดลบ ก็ให้เปลี่ยน Sign bit จาก 1 (ค่าบวก) ไปเป็น 0 (ค่าติดลบ)



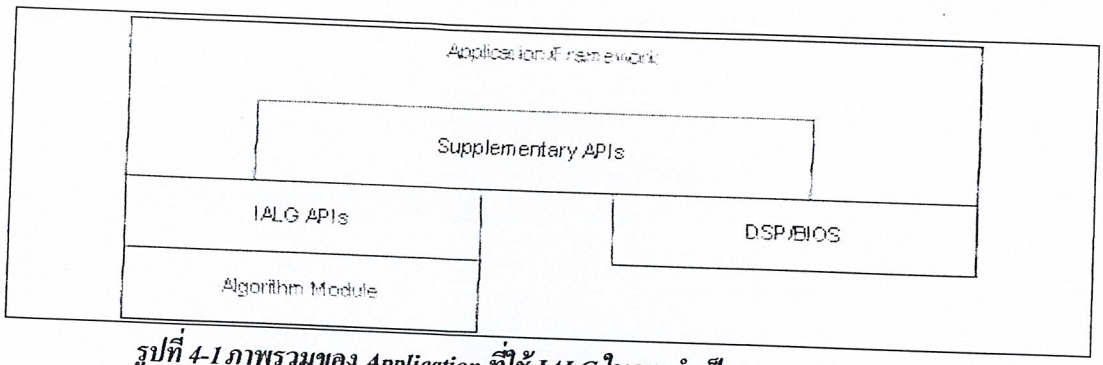
รูปที่ 3-4 การทำ A-Law Companding ทางฝั่งค่าบวก

ในการพัฒนา Library การสร้างสัญญาณโทรศัพท์ จะกำหนดให้แต่ละ Sample ที่ได้จากการสร้างนั้นมีค่าอยู่ในช่วง -4095 ... +4095 เพื่อที่ว่า หากจะนำสัญญาณที่ได้ ไปทำการลดขนาดสัญญาณด้วย A-Law Companding ก็จะสามารถนำค่า Sample นั้นๆ ไปใช้ได้ทันที และในส่วนของการพัฒนา Library การตรวจจับสัญญาณโทรศัพท์นั้น ก็จะกำหนดให้แต่ละ Sample ที่จะส่งให้กับ Library จะต้องมีค่าอยู่ในช่วง -4095 ... +4095 ด้วย เพื่อที่ว่า หากสัญญาณที่รับเข้ามา ได้รับการขยายสัญญาณด้วย A-Law Companding ก็จะสามารถนำ Sample ที่ได้จากการขยายสัญญาณนั้น ไปทำการตรวจจับได้ทันที

บทที่ 4

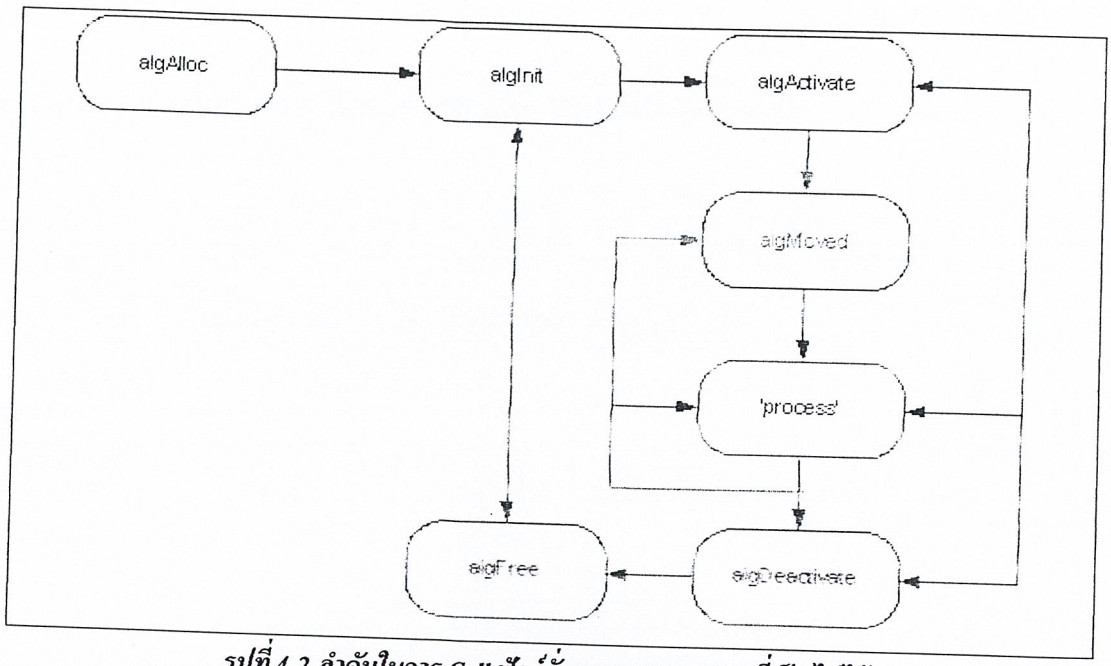
ทฤษฎี TMS320 DSP Algorithm Standard

การพัฒนาอัลกอริทึม ที่ใช้ทั่วไปบน DSP นั้นให้เป็น Algorithm Standard นั้นเราจะอาศัย Interface ชื่อว่า IALG (Instance Algorithm Interface) ในการทำให้เป็น Standard เพื่อง่ายต่อการใช้งาน คือ ทุก Algorithm ที่ทำเป็น Standard นั้นจะมีแนวทางและ Function ที่จำเป็นต่อการใช้เหมือนกัน ทำให้ผู้ที่นำไปใช้ (Integrator) สามารถนำอัลกอริทึมหลายๆ อัลกอริทึม ที่พัฒนาโดยทีมงานคนละทีมกัน มาใช้งานร่วมกันได้อย่างไม่มีปัญหา



รูปที่ 4-1 ภาพรวมของ Application ที่ใช้ IALG ในการทำเป็น Algorithm Standard

IALG APIs นั้นเป็น ฟังก์ชั่น ที่เราต้องทำการ Implement ซึ่งเป็น Interface ไว้ติดต่อกับ Algorithm Module ของผู้พัฒนา ซึ่ง IALG APIs นั้นมี ฟังก์ชั่น ที่ยากต่อการเข้าใจ นิยมเขียน Supplementary APIs มาใช้งานเพื่อให้ง่ายต่อการเข้าใจ เพราะจะให้ Supplementary ติดต่อกับ IALG APIs แทน

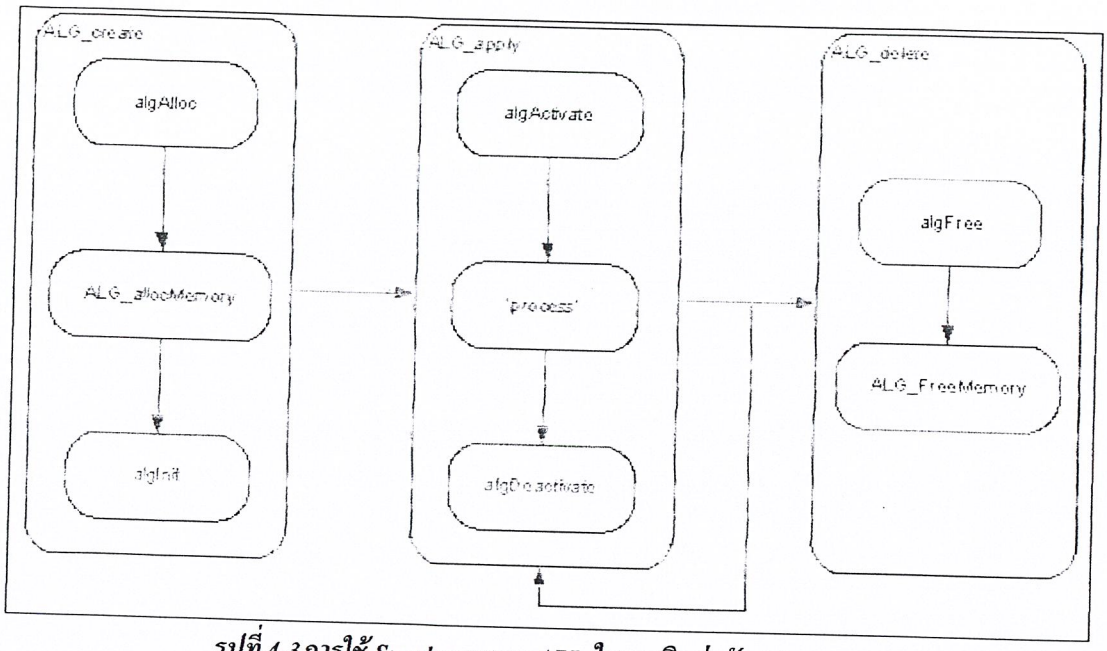


รูปที่ 4-2 ลำดับในการ Call ฟังก์ชั่นของ IALG APIs ที่เป็นไปได้

ใน IALG APIs นั้นลำดับการ Call ฟังก์ชั่นจะเป็นไปตามรูปภาพข้างบนก็คือ เริ่มที่ algAlloc ก็จะไปจองพื้นที่บนหน่วยความจำ และก็จะ call algInit เพื่อ Initialize ค่าเริ่มต้นให้แก่ อัลกอริทึม หลังจากนั้นก็จะเป็นส่วนการใช้ ก็คือ ถ้าต้องการที่จะเรียก อัลกอริทึม (ในที่นี้คือ Process) ขึ้นมาทำงานก็จะต้อง algActivate เพื่อ active ค่าต่างๆใน อัลกอริทึมแล้ว เรียก Process ขึ้นมาทำงาน หลังจากนั้นก็ Deactivate โดยใช้ algDeactivate เหตุผลที่ต้องมีการ Activate และ Deactivate นั้นก็เพราะว่า ในระบบ Real Time นั้น ในการ Call อัลกอริทึม ครั้งหนึ่งไม่สามารถประมวลผลเสร็จภายในครั้งเดียว ดังนั้นจึงจำเป็นที่จะต้องมีการ Save ตัวแปรของอัลกอริทึม ไว้เพื่อในการ Call ฟังก์ชั่นครั้งต่อจะได้ดำเนินการต่อได้ถูกต้อง เทียบกับการ Call ฟังก์ชั่นก็คือ algActive จะทำการ Restore ค่า State ที่เก็บบันทึกไว้ และ algDeactive นั้นจะเป็นการบันทึก ตัวแปรของอัลกอริทึม

ส่วน algMove นั้นมีไว้สำหรับในกรณี Integrator ได้ทำการเคลื่อนย้ายพื้นที่ไว้เก็บตัวแปรต่างๆ ที่จำเป็นในการ Restore และ Save ตัวแปรของอัลกอริทึม ไปไว้ที่อื่น Integrator จำเป็นต้องส่ง Pointer ที่ชี้ตำแหน่งใหม่ผ่าน algMove เพื่อให้ Algorithm รับรู้ถึงการเคลื่อนย้ายพื้นที่ในการทำงาน

และ algFree มีไว้สำหรับ ทำการคืนค่าพื้นที่ Memory ที่เคยถูกจองโดย Algorithm นี้ซึ่ง ฟังก์ชั่นนี้จะถูกเรียกใช้จาก Integrator เมื่อ อัลกอริทึม นี้ทำงานเสร็จเรียบร้อยแล้วไม่มีความจำเป็นที่จะต้องใช้ ณ ขณะนี้ ก็จะทำการเรียกฟังก์ชั่น algFree เพื่อคืนพื้นที่บนหน่วยความจำ ที่จองไว้ให้แก่ระบบ เพื่อนำไปใช้งานอย่างอื่นต่อไป



รูปที่ 4-3 การใช้ Supplementary APIs ในการติดต่อกับ IALG APIs

ถ้าใช้ Supplementary APIs นั้นแทน IALG APIs จะทำให้ง่ายต่อการใช้งานมากขึ้นก็คือตอนที่จะเริ่มใช้ ก็ทำการ Create โดยใช้ฟังก์ชั่น ALG_create เมื่อต้องการใช้ อัลกอริทึม ก็เพียงแต่ Call ฟังก์ชั่น ALG_apply ไปเรื่อยๆ พอทำเสร็จก็ Call ฟังก์ชั่น ALG_delete เพื่อคืน Memory ให้แก Application ซึ่งง่ายกว่าการใช้ IALG APIs โดยตรงมากทีเดียว

4.1 Reentrancy

จุดประสงค์สำคัญอีกอย่างของการทำให้เป็น Algorithm Standard คือ อัลกอริทึม นั้นจะต้องสามารถทำงานแบบ Multichannel ได้ หมายความว่าสามารถประมวลผลหลาย Input หรือ สร้าง Output ได้หลาย Output ในเวลาเดียวกัน โดยใช้ Instance Object ของ Code ชุดเดียวกัน (ใน Program Memory นั้นมีแค่ 1 Instance Object ของ Code แต่สามารถทำงานได้หลายงานพร้อมกัน) แต่การทำให้สิ่งที่กล่าวมาเป็นไปได้นั้นก็ต้องมีข้อกำหนด ดังนี้

- เนื่องจากการทำงานแบบ Preemptive (DSP ใช้ Schedule แบบ Preemptive) นั้นจะไม่สามารถทำการ Preempt ในขณะโปรแกรมกำลังใช้งานตัวแปร Global Variable ที่ไม่ใช่ Constance เพราะฉะนั้นเราควรหลีกเลี่ยงการใช้งานตัวแปร Global โดยตรงถ้าจะเป็นต้องใช้ก็ใช้เป็นแบบ Read Only (กำหนดให้เป็น Const)
- ณ เวลา หนึ่งจะมีการเข้าใช้งาน Resource ได้ทีละ 1 ผู้ใช้งานเพราะฉะนั้นถึงการทำ Multichannel จะเป็นการใช้ Code ร่วมกันแต่ในแต่ละ Channel จะมี Data Memory เป็นของตัวเอง

บทที่ 5
การสร้าง ส่วนสร้างสัญญาณโทรศัพท์

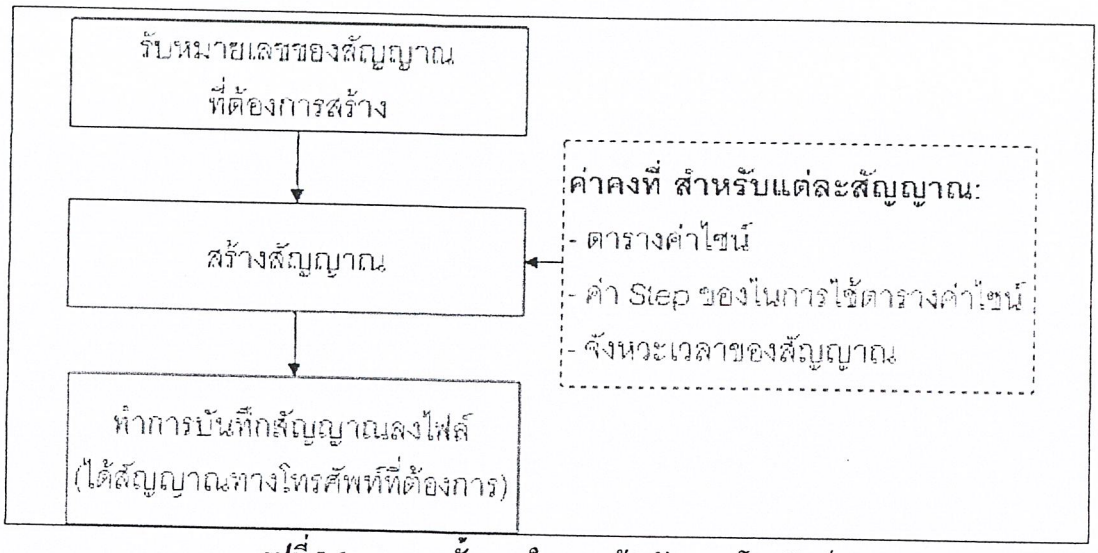
ในการสร้าง ส่วนสร้างสัญญาณโทรศัพท์นั้น จะทำการแบ่งออกเป็น 3 Library ย่อยๆ ได้แก่ [1] ส่วนสร้างสัญญาณ DTMF, [2] ส่วนสร้างสัญญาณ R2 MF, และ [3] ส่วนสร้างสัญญาณ CPT โดยที่แต่ละส่วนจะเป็น Library อิสระ ที่ไม่เกี่ยวข้องกัน และสามารถทำงานแยกกันได้ ในขั้นแรกจะทำการพัฒนา Library บน PC ก่อน โดยใช้โปรแกรม Visual C++ โดยมีข้อกำหนดในการสร้าง ดังนี้

1. ใช้ C Programming Language ในการพัฒนา เพราะจะทำให้สามารถ ย้ายฟังก์ชั่นการทำงานหลักๆ ไปใช้งานบน DSP ได้ง่ายขึ้น เนื่องจากโปรแกรมที่ใช้ในการพัฒนา Library บน DSP นั้นรองรับ C Programming Language และ Assembly Language (DSP Assembly)
2. ในการสร้างสัญญาณ/ตรวจจับสัญญาณ นั้น ใช้ *Sampling Rate = 8000 Hz*
3. ในการเก็บข้อมูลที่เป็นตัวเลข ใช้รูปแบบที่เป็นจำนวนเต็ม หากต้องการใช้ข้อมูลตัวเลขในรูปแบบที่เป็นทศนิยม ก็ต้องเก็บในรูปแบบ Fixed-Point เนื่องจากจะสามารถนำ Library ที่พัฒนาขึ้นนั้นไปใช้ได้ทั้งบน DSP Chip ที่รองรับ และที่ไม่รองรับการคำนวณ Floating-Point
4. ไม่สามารถใช้ฟังก์ชั่นทางคณิตศาสตร์มาตรฐาน ที่ใช้บน PC ได้ (เช่น ฟังก์ชั่นทางคณิตศาสตร์ ที่อยู่ใน math.h) เนื่องจากในการพัฒนาบน DSP นั้น ไม่มีฟังก์ชั่น เหล่านั้นให้ใช้
5. ค่าของแต่ละ Sample ของสัญญาณที่สร้างขึ้นมา จะมีค่าระหว่าง -4095 ... +4095
6. ทำการทดสอบส่วนสร้างสัญญาณ โดยรายละเอียดต่างๆในการทดสอบ จะอธิบายในบทที่เกี่ยวกับการทดสอบ เป็นลำดับถัดไป

เมื่อตัว Library สามารถทำงานได้บน PC แล้ว ก็จะทำการย้ายไปพัฒนากับโปรแกรมที่ใช้ในการพัฒนา Library บน DSP Chip เป็นลำดับต่อไป เพื่อให้สามารถทำงานได้บน DSP Chip และให้สามารถสร้างสัญญาณได้อย่างถูกต้อง

จากที่กล่าวมาแล้วว่าในการสร้างส่วนสร้างสัญญาณโทรศัพท์แต่ละประเภทสัญญาณนั้น จะทำแยกเป็น 3 Library โดยจะมีขั้นตอนในการสร้างเหมือนกัน ต่างกันที่ค่าความถี่ที่ใช้ และจังหวะที่ใช้ของแต่ละสัญญาณ ซึ่งจะถูกกำหนดเป็นค่าคงที่ของแต่ละสัญญาณ ในส่วนของการสร้างสัญญาณนั้นจะสร้างสัญญาณไซน์ส่วนประกอบขึ้นมาก่อน แล้วนำทั้ง 2 สัญญาณมารวมกันเพื่อสร้างสัญญาณที่ต้องการ เช่น ต้องการสร้างสัญญาณ DTMF หมายเลข 1 ก็จะทำการสร้างสัญญาณไซน์ความถี่ 1209 Hz, และ สร้างสัญญาณไซน์ความถี่ 697 Hz แล้วนำสัญญาณไซน์ส่วนประกอบทั้ง 2 มารวมกัน

การสร้างสัญญาณไซน์นั้นไม่สามารถใช้ฟังก์ชั่น sin() ของ math.h ได้ ดังนั้นจึงต้องสร้างตารางค่าไซน์ขึ้นมาไว้ก่อน แล้วประกาศเป็นค่าคงตัวเอาไว้ และในการเรียกใช้ค่าจากตารางค่าไซน์เพื่อสร้างสัญญาณไซน์ในความถี่ที่ต้องการนั้น ก็จะต้องมีการแปลงค่า Radian/Sample ไปเป็น Step Size/Sample โดยค่า Step Size นี้จะใช้เป็นตัวชี้ไปยังสมาชิกในอาร์เรย์ที่ใช้เก็บค่าไซน์ (ตารางค่าไซน์)



รูปที่ 5-1 แผนภาพ ขั้นตอนในการสร้างสัญญาณโทรศัพท์

5.1 ตารางค่าไซน์

ในการพัฒนาส่วนสร้างสัญญาณนี้จะใช้ตารางค่าไซน์ขนาด 512 ช่อง ในการแทนค่าตั้งแต่ sin(0) จนถึง sin(2¶) โดยมีคำสั่งที่ใช้ในการสร้าง Sine Table ดังนี้

SIN_TAB[i] = sin(i * 2 * PI /512) * 4095

i : *ตำแหน่งในอาร์เรย์ ที่ต้องการอ้างถึง, โดยมีค่าตั้งแต่ 0 ถึง 511*

PI : **¶** , ในที่นี้ใช้ค่าเท่ากับ 3.1415926535898

SIN_TAB : int SIN_TAB[512]

***4095** : คูณด้วย 4095 เพื่อรักษาค่าหลังจุดทศนิยม และให้ค่าสูงสุดไม่เกิน 4095, เนื่องจากต้องใช้ตัวแปรประเภทจำนวนเต็มเท่านั้น

ตารางที่ 5-1 คำสั่งในการสร้าง Sine Table

5.2 การใช้ตารางค่าไซน์ในการสร้างสัญญาณไซน์

ในการนำตารางค่าไซน์ไปใช้นั้น จำเป็นจะต้องหาค่า i เพื่อที่จะนำไปอ้างอิงตำแหน่งในตารางค่าไซน์ (0..511) โดยจะใช้ในรูปแบบของ การเปลี่ยนแปลงค่า i/Sample (Step_Size/Sample) ซึ่งหากเทียบกับแบบที่ใช้ฟังก์ชั่น sin() โดยตรงแล้วก็คือ ค่า Radian/Sample ซึ่งในการคำนวณค่า Step Size นั้น จะคำนวณจากสมการดังต่อไปนี้

```
stepf = (f * 512 / sampling_frq) <<10
```

stepf : Step size / Sample

f : ความถี่ของสัญญาณไซน์ ที่ต้องการสร้าง

512 : ขนาดของตารางค่าไซน์ (512 ค่า)

sampling_frq : Sampling rate = 8000 Hz

<<10 : คูณด้วย 1024 เพื่อรักษาค่าหลังจุดทศนิยม

ตารางที่ 5-2 คำสั่งในการคำนวณค่า Step Size / Sample

```
int num_of_sample = second * 8000;

int step1 = (512 * f1 / 8000) <<10;
int step2 = (512 * f2 / 8000) <<10;
int cs1=0, cs2=0;

for (i=0; i<num_of_sample; i++)
{
        sample =
        ( SIN_TAB[ (cs1>>10)%512 ] + SIN_TAB[ (cs2>>10)%512 ] ) / 2;

                // ...another code... //

        cs1 += step1;
        cs2 += step2;
}
```

second : ความยาวของสัญญาณที่ต้องการสร้าง

num_of_sample : จำนวน Sample ของสัญญาณทางโทรศัพท์ที่ต้องการสร้าง

step1, step2 : ค่า Step size ที่จะนำมาใช้ในการสร้างสัญญาณไซน์

f1, f2 : ความถี่ของสัญญาณไซน์ที่ต้องการนำมารวมกันเพื่อสร้างสัญญาณทางโทรศัพท์

cs1, cs2 : ผลรวมของค่า Step size ที่นำมาใช้ในการสร้างสัญญาณไซน์

sample : แต่ละ Sample ของสัญญาณทางโทรศัพท์ที่สร้างขึ้น

SIN_TAB : อาร์เรย์ขนาด 512 ช่อง เก็บค่าคงที่ของตารางค่าไซน์

>>10 : หารด้วย 1024 เพื่อคืนกลับมายังค่าจริง (เพราะมีการคูณด้วย 1024 เพื่อรักษาค่าหลังจุดทศนิยม ในขั้นตอนการคำนวณค่า Step size)

%512 : ทำการหาร ผลรวมของค่า Step size เพื่อใช้เศษที่ได้จากการหาร (2¶ = 512)

/2 : เพื่อป้องกันค่าแต่ละ Sample ของสัญญาณทางโทรศัพท์ที่สร้างขึ้นมีค่าเกิน 4095

ตารางที่ 5-3 ตัวอย่างคำสั่งภาษา C ในการสร้างสัญญาณทางโทรศัพท์ที่ต้องการ

5.3 การสร้าง ส่วนสร้างสัญญาณ DTMF (DTMFG)

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าคงที่ ที่จะใช้ในการสร้างสัญญาณ DTMF โดยค่าคงที่แต่ละตัวมีความหมายดังต่อไปนี้

1. Uns dtmfg_low_step[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ Low-Frequency Group (เป็นค่าที่คูณด้วย 1024 หรือ Left Shift แล้ว 10 Bits เพื่อรักษาค่าหลังจุดทศนิยม)
2. Uns dtmfg_hi_step[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ High-Frequency Group (เป็นค่าที่คูณด้วย 1024 หรือ Left Shift แล้ว 10 Bits เพื่อรักษาค่าหลังจุดทศนิยม)
3. Int *dtmfg_sine_tab เป็น Pointer ชี้ไปยังอาร์เรย์ที่เก็บตารางค่าไซน์
4. MdInt dtmfg_sine_tab_elem เป็นขนาดของอาร์เรย์ที่เก็บตารางค่าไซน์

```
typedef struct
{
        Uns dtmfg_low_step[16];     // unsigned int
        Uns dtmfg_hi_step[16];

        Int *dtmfg_sine_tab;  // int
        MdInt dtmfg_sine_tab_elem; //512

}DTMFG_KONST;
/*
Uns : 32-bit Unsigned Integer Value
Int : 32-bit Signed Integer Value
MdInt : 16-bit Signed Integer value */
```

ตารางที่ 5-4 Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนสร้างสัญญาณ DTMF

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าตัวแปรที่ใช้ในการสร้างสัญญาณ DTMF โดยตัวแปรแต่ละตัวมีความหมายดังต่อไปนี้

1. MdUns gen_dtmf_signal_length เก็บจำนวน Sample ที่ต้องการสร้างในส่วนของ Signal Duration (กำหนดได้โดยผู้ใช้ Library)
2. MdUns gen_dtmf_pause_length เก็บจำนวน Sample ที่ต้องการสร้างในส่วนของ Pause Duration (กำหนดได้โดยผู้ใช้ Library)
3. MdInt *gen_dtmf_targetarray เป็น Pointer ชี้ไปยังอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample ของสัญญาณที่ทำการสร้างขึ้น (กำหนดได้โดยผู้ใช้ Library)
4. MdUns gen_dtmf_array_element เก็บขนาดของอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample (กำหนดได้โดยผู้ใช้ Library)
5. MdUns gen_dtmf_arr_ptr เก็บ Index ของอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample (ตัวแปรใช้ภายใน Library)

6. Uns gen_dtmf_acs1 เก็บผลรวมของค่า Step size ที่นำมาใช้ในการสร้างสัญญาณไซน์ตัวที่ 1 (ตัวแปรใช้ภายใน Library)
7. Uns gen_dtmf_acs2 เก็บผลรวมของค่า Step size ที่นำมาใช้ในการสร้างสัญญาณไซน์ตัวที่ 2 (ตัวแปรใช้ภายใน Library)
8. Char gen_dtmf_currentDigit เก็บหมายเลขของ DTMF ที่กำลังทำการสร้างสัญญาณอยู่ในขณะนั้น (ตัวแปรใช้ภายใน Library)
9. MdInt gen_dtmf_signalGend เก็บจำนวน Sample ของ Signal Duration ที่ทำการสร้างไปแล้ว (ตัวแปรใช้ภายใน Library)
10. MdInt gen_dtmf_pauseGend เก็บจำนวน Sample ของ Pause Duration ที่ทำการสร้างไปแล้ว (ตัวแปรใช้ภายใน Library)
11. Char gen_dtmf_state เก็บ State ของการสร้างสัญญาณ โดยมีค่าที่เป็นไปได้ดังนี้

    0 – Idle State

    1 – Signal Generation State

    2 – Pause Generation State

    (ตัวแปรใช้ภายใน Library)
12. Char *gen_dtmf_genDTMFQueue เป็น Pointer ชี้ไปยังอาร์เรย์ที่ผู้ใช้ต้องการเก็บหมายเลข DTMF ที่ต้องการสร้างสัญญาณ โดยจะจัดการเป็นโครงสร้างข้อมูลแบบ Queue (กำหนดได้โดยผู้ใช้ Library)
13. MdUns gen_dtmf_genDTMFQueue_element เก็บขนาดของอาร์เรย์ที่ผู้ใช้ต้องการเก็บหมายเลข DTMF ที่ต้องการสร้างสัญญาณ (กำหนดได้โดยผู้ใช้ Library)
14. MdUns gen_dtmf_sPtr เก็บ Start Index ของ genDTMFQueue (ตัวแปรใช้ภายใน Library)
15. MdUns gen_dtmf_ePtr เก็บ End Index ของ genDTMFQueue (ตัวแปรใช้ภายใน Library)
16. Char gen_dtmf_enable เก็บ Flag ในการควบคุมการสร้างสัญญาณของ Library (กำหนดได้โดยผู้ใช้ Library)

```
typedef struct{

    MdUns gen_dtmf_signal_length;     //800  samples        // unsigned short
    MdUns gen_dtmf_pause_length;      //800

    MdInt *gen_dtmf_targetarray;                          //NULL //short
    MdUns gen_dtmf_array_element;              //0
    MdUns gen_dtmf_arr_ptr;

    Uns gen_dtmf_acs1;  //0              // unsigned int
    Uns gen_dtmf_acs2;  //0
    Char gen_dtmf_currentDigit; // short
    MdInt gen_dtmf_signalGend;
    MdInt gen_dtmf_pauseGend; // short
```

```
        Char gen_dtmf_state; //0              //MdInt

        Char *gen_dtmf_genDTMFQueue;          //NULL
        MdUns gen_dtmf_genDTMFQueue_element;//0

        MdUns gen_dtmf_sPtr;      //0
        MdUns gen_dtmf_ePtr;      //1

        Char gen_dtmf_enable;     //      0(disable, default), 1(enable)

} DTMFG_MBLK;

/*
Char : 8-bit Unsigned Integer Value
MdUns : 16-bit Unsigned Integer value */
```

ตารางที่ 5-5 Structure เพื่อใช้ในการเก็บตัวแปรของส่วนสร้างสัญญาณ DTMF

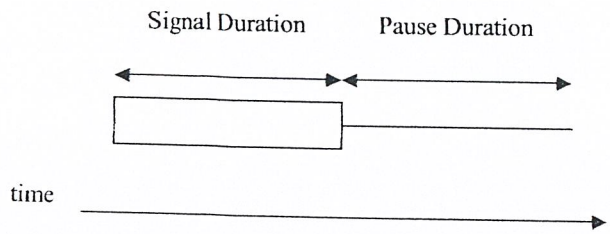
ในการทำงานนั้นเริ่มต้น ผู้ใช้จะต้องทำการตั้งค่าเริ่มต้นให้กับ Library โดยจะต้องทำการกำหนดค่าคงที่ต่างๆที่ต้องการใช้ และค่าตัวแปรบางตัว ดังที่ได้กล่าวไปแล้ว จากนั้นเมื่อต้องการสร้างสัญญาณก็จะต้องทำการเรียกใช้ฟังก์ชั่นที่ทำการเพิ่มหมายเลข DTMF ที่ต้องการสร้างสัญญาณ เข้าไปใน Generation Queue ก่อน โดยมีหมายเลขที่ใช้แทนหมายเลข DTMF จริง ดังตารางด้านล่าง เช่นหากต้องการให้สร้างสัญญาณ DTMF แทนหมายเลข *1001 ก็ทำการ เพิ่มค่า 10, 1, 0, 0, 1 เข้าไปใน Generation Queue ตามลำดับ จากนั้นจึงเรียกฟังก์ชั่นที่ใช้ในการสร้างสัญญาณอีกครั้งหนึ่ง

DTMF	0	1	2	3	4	5	6	7	8	9	*	#	A	B	C	D
ค่าที่ใช้	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

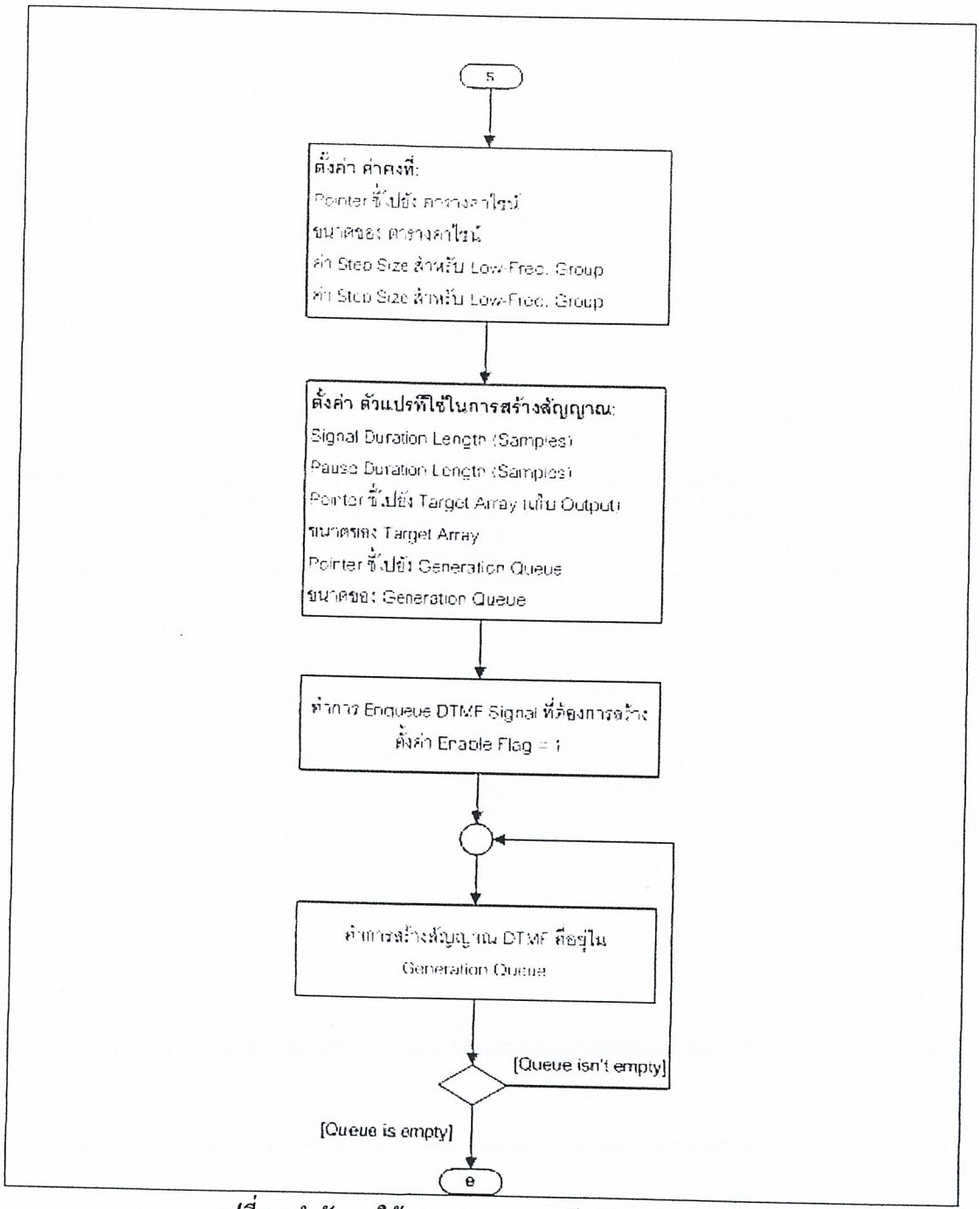
ตารางที่ 5-6 ค่าที่ใช้ในการสร้างสัญญาณ DTMF เทียบกับ หมายเลข DTMF จริง

ในการสร้างสัญญาณนั้น ตัวฟังก์ชั่นจะสร้างสัญญาณเป็นจำนวน Sample เท่ากับค่า gen_dtmf_array_element ที่กำหนดโดยผู้ใช้ และ Sample ของสัญญาณที่ได้จะถูกเขียนทับลงในอาร์เรย์ที่ตัวแปร *gen_dtmf_targetarray ชี้ไป (แต่ละ Sample มีค่าตั้งแต่ -4095 .. +4095) โดยจะ Return ค่าสถานะการสร้างสัญญาณในขณะนั้นด้วย ทำให้ผู้ใช้สามารถทราบได้ว่า หมายเลข DTMF ที่ต้องการสร้างสัญญาณ ใน Generation Queue นั้น ได้ถูกสร้างออกมาจนหมดแล้วหรือยัง

หมายเลข DTMF 1 หมายเลขที่ได้จาก Generation Queue จะทำให้ได้สัญญาณออกมาเป็นช่วง Signal Duration และช่วง Pause Duration ดังรูปต่อไปนี้



รูปที่ 5-2 สัญญาณที่ได้จากการสร้างสัญญาณ DTMF จำนวน 1 หมายเลข



รูปที่ 5-3 ลำดับการใช้งาน Library การสร้างสัญญาณ DTMF

5.3.1 อัลกอริทึมของการสร้างสัญญาณ DTMF

ในการสร้างสัญญาณ DTMF นั้น ประเด็นหลักที่ต้องคำนึงถึงก็คือ จังหวะของสัญญาณ (ประกอบด้วย Signal Duration และ Pause Duration) และ คู่ความถี่ที่ใช้ในการสร้างสัญญาณ ซึ่งจะมีผลอย่างมากในการตรวจจับที่ฝั่งผู้รับสัญญาณ โดยตัว Library ในการสร้างสัญญาณ DTMF นั้นอนุญาตให้ผู้ใช้ทำการกำหนด Signal Duration และ Pause Duration ที่ต้องการได้ รวมถึงการอนุญาตให้ผู้ใช้เลือกใช้ความถี่ของทั้ง Low Frequency Group และ High Frequency Group เองได้ โดยการส่งค่า Step Size ของแต่ละความถี่ที่ต้องการมาเป็นค่าคงตัวที่ใช้ในการสร้างสัญญาณ

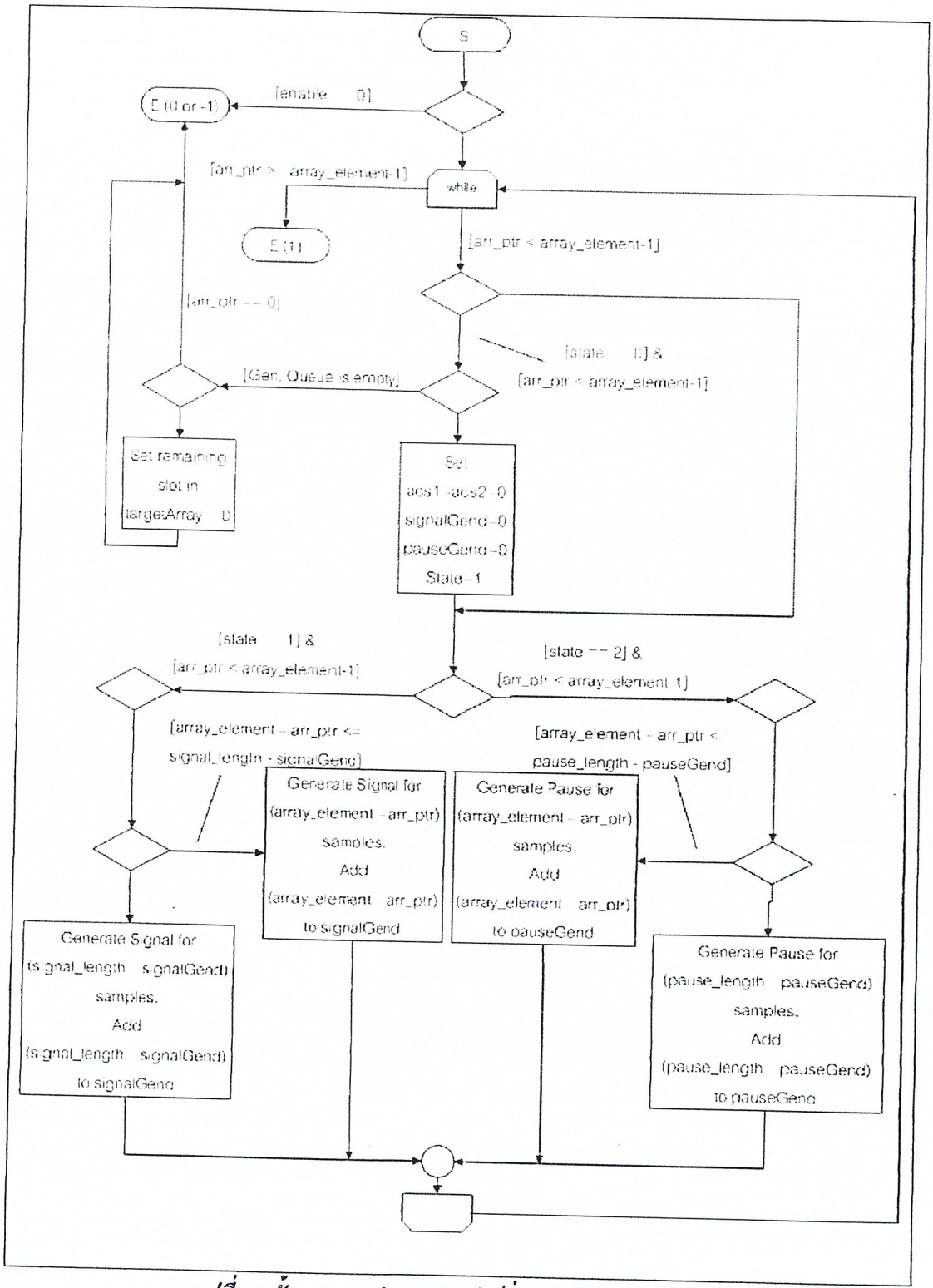
โดยในการเรียกฟังก์ชั่นที่ใช้ในการสร้างสัญญาณแต่ละครั้งนั้น ตัวฟังก์ชั่นจะทำการสร้างสัญญาณเป็นจำนวน Sample เท่ากับค่าของ gen_dtmf_array_element ที่กำหนดโดยผู้ใช้ การสร้างสัญญาณมีขั้นตอนดังต่อไปนี้

1) เมื่อเข้ามาในฟังก์ชั่นจะมีการตรวจก่อนว่า หาก enable flag == 0 ก็จะ return (-1) แล้วออกจากฟังก์ชั่น
2) จากนั้นจะดูว่าหากอยู่ใน Idle State (State 0) ก็จะพยายามดูว่าใน Generation Queue ยังมีสัญญาณที่ต้องสร้างเหลืออีกหรือไม่ หากยังมีก็จะทำการ Initialize ค่าตัวแปรภายในเพื่อเตรียมสร้างสัญญาณ หาก Generation Queue ว่าง ก็จะ return (0) แล้วออกจากฟังก์ชั่น
3) หากอยู่ใน State 1 (Signal Generation State) ก็จะสร้างสัญญาณ DTMF (การสร้างใช้ค่า Step Size ร่วมกับตารางค่าไซน์ ดังที่ได้อธิบายไปแล้ว)ให้เต็มช่องใน อาร์เรย์ targetArray หรือ สร้างจนครบจำนวน Sample ที่ผู้ใช้กำหนด (signal_length)
4) หากอยู่ใน State 2 (Pause Generation State) ก็จะสร้าง Pause ซึ่งก็คือ Sample ที่มีค่าเท่ากับศูนย์ให้เต็มช่องในอาร์เรย์ targetArray หรือสร้างจนครบจำนวน Sample ที่ผู้ใช้กำหนด (pause_length)

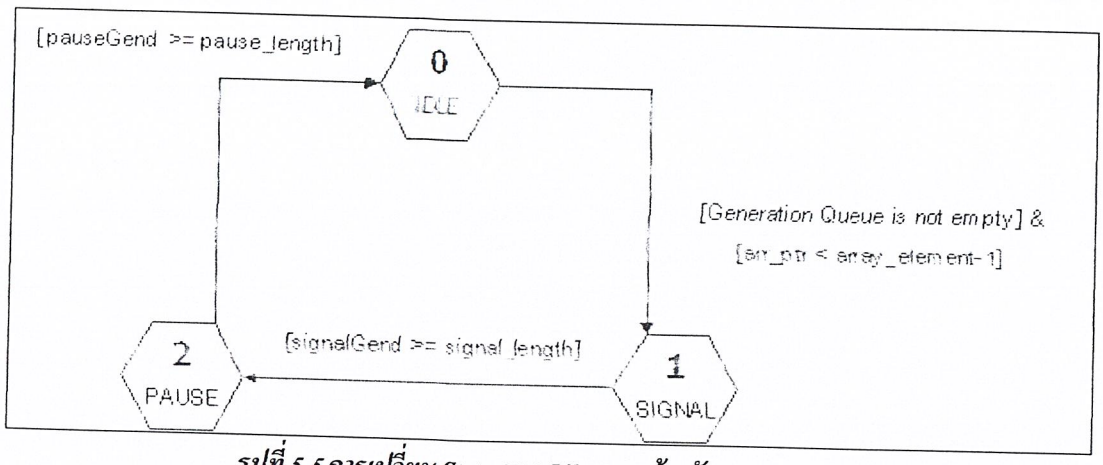
*ขั้นตอน 2..4 จะอยู่ใน while loop ที่จะวนไปจนกว่า arr_ptr >= array_element-1 ซึ่งก็คือ ทุกช่องใน targetArray ถูกป้อน Sample ของสัญญาณลงไปแล้ว โดย arr_ptr คือตัวชี้ช่องของอาร์เรย์ targetArray ที่จะถูกป้อนค่าลงไปในขณะนั้น

*หากช่องใดๆใน targetArray ถูกป้อน Sample ของสัญญาณลงไปแล้ว เมื่อการทำงานหลุดจาก while loop แล้ว จะ return (1) แล้วออกจากฟังก์ชั่น

*การเปลี่ยน State ของ Library ดูได้จากภาพประกอบ



รูปที่ 5-4 ขั้นตอนการทำงานของฟังก์ชั่นสร้างสัญญาณ DTMF



รูปที่ 5-5 การเปลี่ยน State ของ Library สร้างสัญญาณ DTMF

5.4 การสร้าง ส่วนสร้างสัญญาณ R2 MF (R2MFG)

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าคงที่ ที่จะใช้ในการสร้างสัญญาณ R2 MF โดยค่าคงที่แต่ละตัวมีความหมายดังต่อไปนี้

1. Uns r2mfg_fwd_a[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ที่1 ของ R2 MF Forward Signal
2. Uns r2mfg_fwd_b[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ที่2 ของ R2 MF Forward Signal
3. Uns r2mfg_bwd_a[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ที่1 ของ R2 MF Backward Signal
4. Uns r2mfg_bwd_b[16] เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่ที่2 ของ R2 MF Backward Signal
5. Int *r2mfg_sine_tab เป็น Pointer ชี้ไปยังอาร์เรย์ที่เก็บตารางค่าไซน์
6. MdInt r2mfg_sine_tab_elem เป็นขนาดของอาร์เรย์ที่เก็บตารางค่าไซน์

```
typedef struct
{
        Uns r2mfg_fwd_a[16];        //8kHz *1024  unsigned int
        Uns r2mfg_fwd_b[16];        //0(not used) 1..15

        Uns r2mfg_bwd_a[16];
        Uns r2mfg_bwd_b[16];

        Int *r2mfg_sine_tab;
        MdInt r2mfg_sine_tab_elem; //512

} R2MFG_KONST;
```

ตารางที่ 5-7 Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนสร้างสัญญาณ R2 MF

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าตัวแปรที่ใช้ในการสร้างสัญญาณ R2 MF โดยตัวแปรแต่ละตัวมีความหมายดังต่อไปนี้

1. MdInt *gen_r2mf_targetarray เป็น Pointer ชี้ไปยังอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample ของสัญญาณที่ทำการสร้างขึ้น (กำหนดได้โดยผู้ใช้ Library)

2. MdInt gen_r2mf_array_element เก็บขนาดของอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample (กำหนดได้โดยผู้ใช้ Library)
3. MdInt gen_r2mf_arr_ptr เก็บ Index ของอาร์เรย์ที่ผู้ใช้ต้องการเก็บ Output Sample (ตัวแปรใช้ภายใน Library)
4. gen_r2mf_percentpow ตัวคูณกับสัญญาณที่สร้างขึ้นเพื่อลดขนาดสัญญาณ เลือกได้ตั้งแต่ 0% - 100% , ค่า Default เท่ากับ 55% (กำหนดได้โดยผู้ใช้ Library)
5. Uns gen_r2mf_acs1 เก็บผลรวมของค่า Step size ที่นำมาใช้ในการสร้างสัญญาณไซน์ตัวที่ 1 (ตัวแปรใช้ภายใน Library)
6. Uns gen_r2mf_acs2 เก็บผลรวมของค่า Step size ที่นำมาใช้ในการสร้างสัญญาณไซน์ตัวที่ 2 (ตัวแปรใช้ภายใน Library)
7. Uns gen_r2mf_step1 เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่สัญญาณไซน์ตัวที่ 1 (ตัวแปรใช้ภายใน Library)
8. Uns gen_r2mf_step2 เป็นอาร์เรย์ที่เก็บค่า Step ของความถี่สัญญาณไซน์ตัวที่ 2 (ตัวแปรใช้ภายใน Library)
9. MdInt gen_r2mf_currentDigit เก็บหมายเลขของ R2 MF ที่กำลังทำการสร้างสัญญาณอยู่ในขณะนั้น (ตัวแปรใช้ภายใน Library)
10. MdInt gen_r2mf_state เก็บ State ของการสร้างสัญญาณ โดยมีค่าที่เป็นไปได้ดังนี้

    0 – Idle State

    1 – Signal Generation State

    (กำหนดได้โดยผู้ใช้ Library)

```
typedef struct{

        MdInt *gen_r2mf_targetarray;        //NULL short
        MdInt gen_r2mf_array_element;       //0
        MdInt gen_r2mf_arr_ptr;
        MdInt gen_r2mf_percentpow;                  //55

        // ---
        Uns gen_r2mf_acs1;          //0              unsigned int
        Uns gen_r2mf_acs2;          //0
        Uns gen_r2mf_step1;
        Uns gen_r2mf_step2;
        MdInt gen_r2mf_currentDigit;        //short

        MdInt gen_r2mf_state;                       //0      short
} R2MFG_MBLK;
```

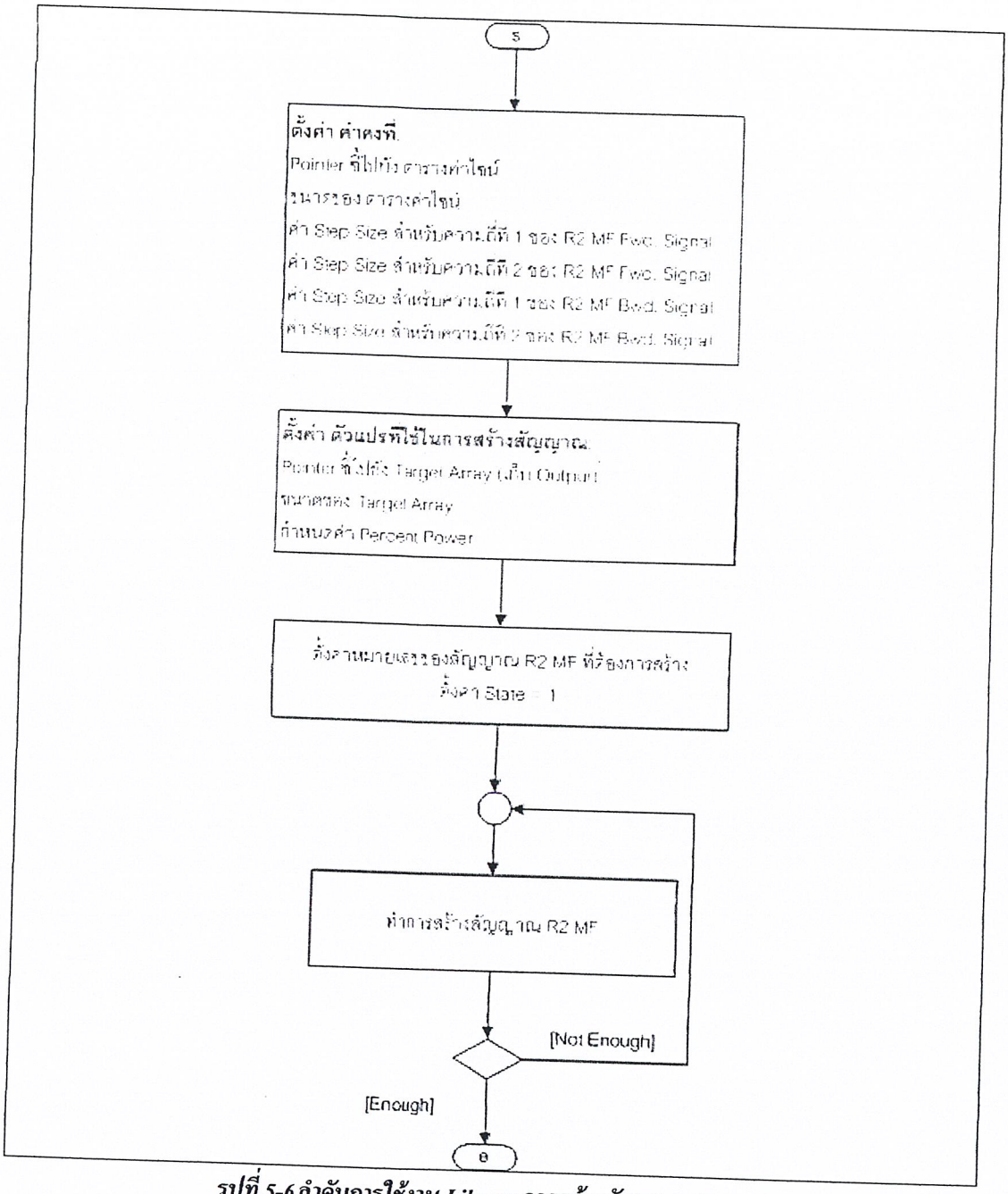
ตารางที่ 5-8 ***Structure เพื่อใช้ในการเก็บตัวแปรของส่วนสร้างสัญญาณ R2 MF***

ในการทำงานนั้นเริ่มต้น ผู้ใช้จะต้องทำการตั้งค่าเริ่มต้นให้กับ Library โดยจะต้องทำการกำหนดค่าคงที่ต่างๆที่ต้องการใช้ และค่าตัวแปรบางตัว ดังที่ได้กล่าวไปแล้ว จากนั้นเมื่อต้องการสร้างสัญญาณก็จะต้องทำการเรียกใช้ฟังก์ชั่นที่ทำการตั้งค่าภายในเพื่อเตรียมสร้างสัญญาณ R2 MF ที่ต้องการ โดยมีรูปแบบหมายเลขที่ใช้แทนหมายเลข R2 MF จริง ดังตารางด้านล่าง เช่นหากต้องการให้สร้างสัญญาณ Forward R2 MF หมายเลข 1 ก็ทำการตั้งค่าด้วย 0x0001, หากต้องการให้สร้างสัญญาณ Forward R2 MF หมายเลข 13 ก็ทำการตั้งค่าด้วย 0x000D, หากต้องการให้สร้างสัญญาณ Backward R2 MF หมายเลข 10 ก็ทำการตั้งค่าด้วย 0x001A เป็นต้น จากนั้นจึงเรียกฟังก์ชั่นที่ใช้ในการสร้างสัญญาณอีกครั้งหนึ่ง

Set desired R2MF signal in this format 0x00[1][2].
[1] – 0 => forward signal, 1 => backward signal
[2] – <* as table below (in hexadecimal) *>

[2]	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
R2MF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ตารางที่ 5-9 การตั้งค่าที่ใช้ในการสร้างสัญญาณ R2 MF



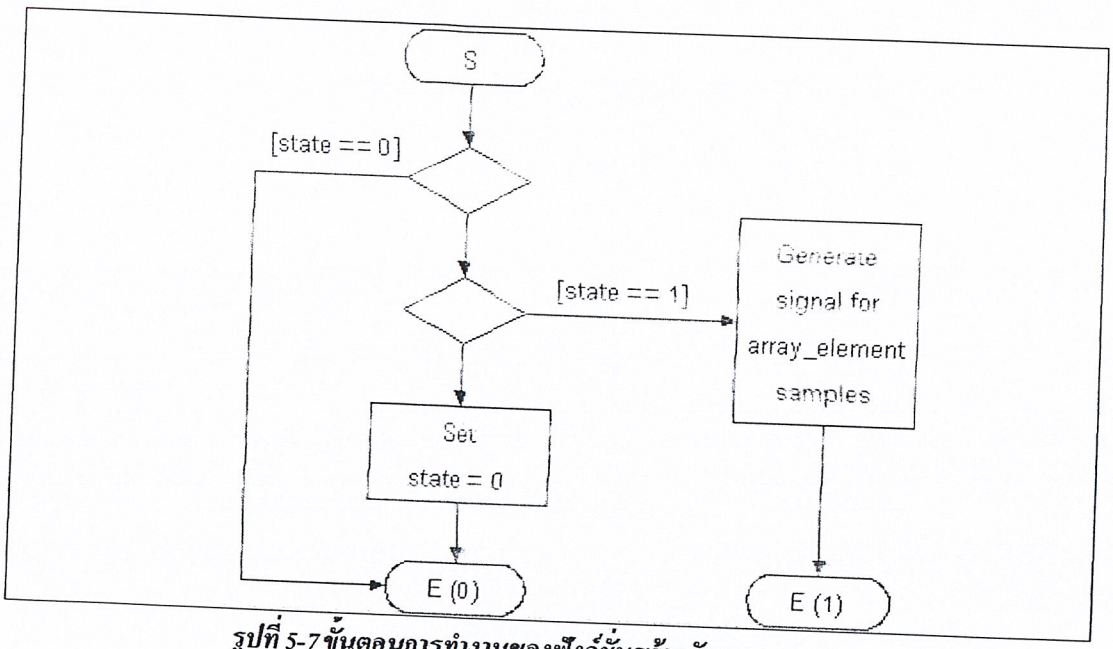
รูปที่ 5-6 ลำดับการใช้งาน Library การสร้างสัญญาณ R2 MF

5.4.1 อัลกอริทึมของการสร้างสัญญาณ R2 MF

ในการสร้างสัญญาณ R2 MF นั้น ประเด็นหลักที่ต้องคำนึงถึงก็คือ คู่ความถี่ที่ใช้ในการสร้างสัญญาณ ซึ่งจะมีผลอย่างมากในการตรวจจับที่ฝั่งผู้รับสัญญาณ โดยตัว Library ในการสร้างสัญญาณ R2 MF นั้นอนุญาตให้ผู้ใช้เลือกใช้ความถี่ของทั้ง R2 MF Forward Signal และ R2 MF Backward Signal เองได้ โดยการส่งค่า Step Size ของแต่ละความถี่ที่ต้องการมาเป็นค่าคงตัวที่ใช้ในการสร้างสัญญาณ

ในการเรียกฟังก์ชั่นที่ใช้ในการสร้างสัญญาณแต่ละครั้งนั้น ตัวฟังก์ชั่นจะทำการสร้างสัญญาณเป็นจำนวน Sample เท่ากับค่าของ gen_r2mf_array_element ที่เก็บโดยผู้ใช้ การสร้างสัญญาณมีขั้นตอนดังต่อไปนี้

1) เมื่อเข้ามาในฟังก์ชั่นจะมีการตรวจก่อนว่า หาก state == 0 ก็จะ return (0) แล้วออกจากฟังก์ชั่น
2) หากอยู่ใน State 1 (Signal Generation State) ก็จะสร้างสัญญาณ R2 MF (การสร้างใช้ค่า Step Size ร่วมกับตารางค่าไซน์ ดังที่ได้อธิบายไปแล้ว) ให้เต็มช่องใน อาร์เรย์ targetArray



รูปที่ 5-7 ขั้นตอนการทำงานของฟังก์ชั่นสร้างสัญญาณ R2 MF

5.5 การสร้าง ส่วนสร้างสัญญาณ CPT (CPTGEN)

การพัฒนาอัลกอริทึมให้เป็นไปตาม DSP Standard Algorithm นั้นจำเป็นที่จะต้องนำ Code จำลองการทำงานที่เขียนไว้บน PC ด้วยภาษา C++ มาปรับปรุงให้สามารถใช้ได้บนบอร์ด DSP ก่อนโดยไม่มีปัญหา และได้ผลลัพธ์เหมือนบน PC แต่ว่าการทำงานของ DSP มีข้อแตกต่างกับ PC ก็คือ PC จะเก็บ Sample ไว้บน File ทำแต่ละครั้งก่อนอ่าน Sample มาทั้งหมดของไฟล์มาเก็บไว้ที่ อาร์เรย์ แล้วค่อยนำมาประมวลผลซึ่งจะเป็นการทำงานแบบ Batch แต่การทำงานของ DSP นั้นเป็นแบบ Real Time ภายใต้ระบบปฏิบัติการแบบ Real Time ซึ่งจะมี Sample เข้ามาตลอดเวลาเพราะฉะนั้นวิธีการย่อมต่างกัน

5.5.1 อัลกอริทึมของการสร้างสัญญาณ CPT

การจะพัฒนา CPT Generation Algorithm ให้ทำงานแบบ Real Time นั้น ทำได้โดยให้ สร้างฟังก์ชั่น โดย ในการ Call ฟังก์ชั่น แต่ละครั้งให้ สร้าง Samples ออกมาตามค่าที่กำหนดไว้ (ค่านี้สามารถกำหนดได้ตามความเหมาะสม) อย่างเช่นการ Call ฟังก์ชั่น 1 ครั้งจะ สร้าง Samples ได้ 80 Samples ดังนั้นถ้าสัญญาณสัญญาณหนึ่งมี Samples 8000 ตัวก็จะต้อง Call Generation Function นี้ 100 ครั้ง เป็นต้น และ

เมื่อสร้าง ครบ 1 สัญญาณก็จะ Set Flag ตัวหนึ่งเพื่อบ่งบอกว่าสร้างสัญญาณนั้นเสร็จแล้ว ประโยชน์เพื่อที่จะเปลี่ยนสัญญาณที่จะสร้างเป็นตัวต่อไป

```
void GenCPT(CPT_GenParam*
param)
{
        switch(param->code)
        {
                case 0: ...
                case 1: ...
                ...
        }
}
```

ตารางที่ 5-10 ใช้ switch case ในการสร้างสัญญาณ

ซึ่ง Samples ที่ได้จากการสร้างโดย Generation Function นั้นจะเก็บอยู่ใน param->data ซึ่ง data มีชนิดเป็น pointer ของจำนวนเต็ม (Integer of Pointer)

ตัวอย่าง ต้องการ Generate สัญญาณ Busy Tone จะมีวิธีการดังนี้
กำหนดค่าตัวแปรใน Structure param->code = 2; //BUSY TONE
ซึ่งเป็น Parameter ใน Function

void GenCPT(CPT_GenParam* param)

ตัวแปร Structure param เป็นตัวแปรที่เก็บ state ในการสร้างสัญญาณ เนื่องจากในการสร้างสัญญาณ 1 สัญญาณไม่เสร็จภายในการ Call Function 1 ครั้ง จึงต้องมีการเก็บสถานะ เพื่อที่จะมาสร้างสัญญาณต่อได้ในการ Call Function ครั้งต่อไป แต่ถ้าสัญญาณถูกสร้างเสร็จ 1 สัญญาณก็จะ set ตัวแปรใน Structure param->completeCycle = 1; สิ่งที่ต้องทำก็คือ ตรวจสอบว่าตัวแปร completeCycle ใน Structure param มีค่าเป็น 1 รึเปล่าถ้ามีแสดงว่าสร้างสัญญาณ สัญญาณปัจจุบันเสร็จแล้ว ก็จะเปลี่ยน param->code ตัวต่อไปที่จะสร้าง รายละเอียดเกี่ยวกับ structure CPT_GenParam จะกล่าวในหัวข้อถัดไป

```
For(i = 0;i < orderlength; i++)
        {
                if(param.completeCycle == 1)
                {
                        param.code = order[i];
                        param.completeCycle = 0;
                }
                GenCPT(&param);
        }
```

ตารางที่ 5-11 ตัวอย่างการนำ อัลกอริทึม ไปใช้

ตัวแปรในการ Save State

```
typedef struct CPT_GenParam
{
        int *data;
        unsigned int  SampRate;
        unsigned char code;
        unsigned char amplitude;      //1.0 scale to 100
        unsigned char completeCycle;          //1 = completed cycle
//=========State Param==========
        unsigned int  StartSample;    //Start Sample of this call
        unsigned char state;          //1 = tone, 0 = pause
        unsigned int  step1;
        unsigned int  step2;
        unsigned char full;           //1 = full data in data
        unsigned int  count;
        int *SinTable;
} CPT_GenParam;
```

ตารางที่ 5-12 Structure ในที่ใช้ในการบันทึก state ในการสร้างสัญญาณ

ในการ Call Function แต่ละครั้งเมื่อสิ้นสุดการ Call มนแต่ละครั้งจะมีการ Save State ไว้ใน Structure ที่เป็น Instant ของ CPT_GenParam ซึ่งประกอบด้วย Member ดังรูปข้างบน ตัวแปร StartSample, step1, step2, full, count เป็นตัวแปรที่ใช้ Save State จริงๆ ซึ่งผู้ใช้ต้องไม่ไปเปลี่ยนแปลงค่าของตัวแปรเหล่านี้ เพราะว่าจะทำให้ Algorithm ทำงานไม่ถูกต้อง

ซึ่งตัวแปรที่จำเป็นสำหรับผู้ใช้ก็คือตัวแปรที่เหลือ มี

- data เป็น pointer ชี้ไปที่ Output Array ที่จะเก็บผลลัพธ์ซึ่งผู้นำไปใช้ต้องกำหนดให้
- code เป็นตัวแปรที่ใช้กำหนดว่า ณ ตอนนี้สร้างสัญญาณชนิดใดอยู่โดยมี code ดังนี้

  0 - Dial tone 1 - Ring tone 2 - Busy tone 3 - Congestion tone

  4 - Special information tone 5 - warning tone 6 - Pay phone tone

  7 - Call waiting tone 8 - Recall dial tone

9 - Intercept tone 10 - Confirmation tone 11 - Comfort tone

12 - Busy verification 13 - Caller waiting tone 20 - Pause 100 ms

ซึ่งค่าเหล่านี้ผู้ใช้จะทำการเปลี่ยนแปลงค่าได้เฉพาะตอนนี้ สัญญาณอันก่อนได้สร้างเสร็จไปแล้ว โดยตรวจสอบจาก completeCycle เท่านั้นไม่อย่างนั้น algorithm ก็จำทำงานผิดพลาด

- amplitude เป็นตัวแปรที่ใช้กำหนดค่า factor (มีค่าระหว่าง 0-1.0) ที่จะนำไปคูณกับ Amplitude ของสัญญาณได้จะมีผลต่อ Power ของสัญญาณโดยตรง โดย output ของการสร้างสัญญาณจะมีค่า ±4095 (12 bits + 1 signed bit) ซึ่งเป็น output ที่เหมาะสมในการทำ A-Law ได้เลยไม่ต้องมีการเปลี่ยนแปลงใดๆ
- completeCycle ไว้ใช้ตรวจสอบว่าสัญญาณปัจจุบันสร้างเสร็จหรือยัง ถ้ามีค่าเป็น 1 คือสร้างเสร็จแล้ว ซึ่งจะถูก set เป็น 1 เฉพาะในการ Call Function ครั้งที่มีการทำเสร็จ ในการ Call ครั้งต่อต่อค่านี้ก็จะถูกกำหนดให้เป็น 0 โดย algorithm

การสร้างสัญญาณช่วง Tone Duration

- ทำการสร้าง Sine table ไว้ก่อน เพื่อใช้ในการหาค่า Sine ที่จะนำมาสร้างสัญญาณ
- ใช้ Sine table ขนาด 512 ช่อง (2*PI)
- SIN_TAB i = sin(i * 2 * PI /512) * 4095 ; i= 0 .. 511
  คูณด้วย 4095 เพื่อให้ได้ค่าสูงสุดที่ 4095 และ ต่ำสุดที่ -4095
- ในการสร้างสัญญาณความถี่ใดๆ ทำได้โดยหาค่า step per sample
  $step_f = (f * 512 / sampling_frq) * 1000$

แนวคิดก็คือ หาว่าที่ความถี่ที่ต้องการนั้น 1 sample ที่ sampling rate ที่กำหนด นั้นเทียบเท่ากับกี่องศา (กี่ PI) และนำค่า องศาหรือ PI นั้นไปหาค่า sine มาซึ่งตารางที่เรากำหนดนั้นได้ทำการหาค่า sine * 4095 มาแล้วแต่หามาแค่ 512 ตัวจาก 2¶ หมายความว่า ค่า sine แต่ละตัวต่างกัน 2¶/512 radian (เพื่อความเข้าใจมากขึ้นควรอ่าน code ข้างบนประกอบและทำความเข้าใจเองอีกครั้ง)

การสร้างสัญญาณช่วง Pause Duration

สร้างสัญญาณที่มี Power ของสัญญาณน้อยกว่ามาตรฐานที่กำหนดไว้ ปกติก็จะกำหนดให้มีค่าเป็น ศูนย์ ไปเลยเพื่อความสะดวก

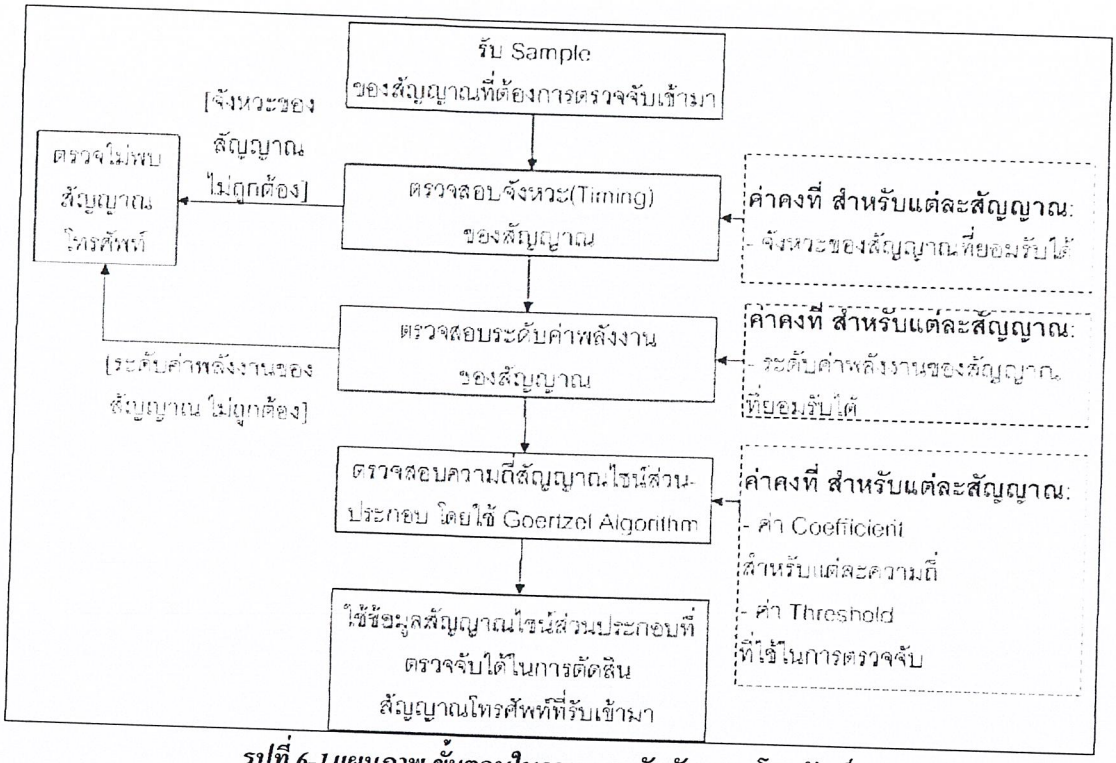
บทที่ 6
การสร้าง ส่วนตรวจจับสัญญาณโทรศัพท์

ในการสร้าง ส่วนตรวจจับสัญญาณโทรศัพท์นั้น จะทำการแบ่งออกเป็น 3 Library ย่อยๆ ได้แก่ [1] ส่วนตรวจจับสัญญาณ DTMF, [2] ส่วนตรวจจับสัญญาณ R2 MF, และ [3] ส่วนตรวจจับสัญญาณ CPT โดยที่แต่ละส่วนจะเป็น Library อิสระ ที่ไม่เกี่ยวข้องกัน และสามารถทำงานแยกกันได้ ในขั้นแรกจะทำการพัฒนา Library บน PC ก่อน โดยใช้โปรแกรม Visual C++ โดยมีข้อกำหนดในการสร้าง ดังนี้

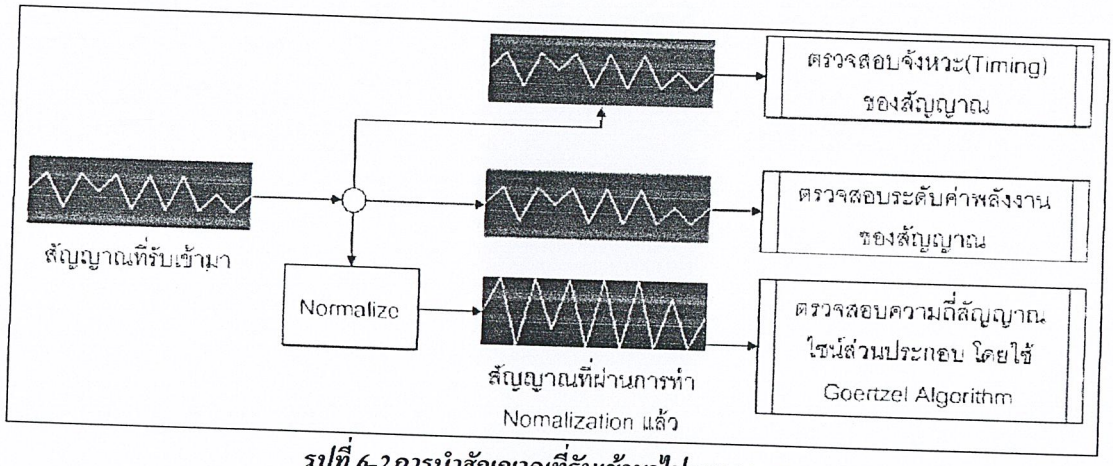
1. ใช้ C Programming Language ในการพัฒนา เพราะจะทำให้สามารถ ย้ายฟังก์ชั่นการทำงานหลักๆ ไปใช้งานบน DSP ได้ง่ายขึ้น เนื่องจากโปรแกรมที่ใช้ในการพัฒนา Library บน DSP นั้นรองรับ C Programming Language และ Assembly Language (DSP Assembly)
2. ในการสร้างสัญญาณ/ตรวจจับสัญญาณ นั้น ใช้ Sampling Rate = 8000 Hz
3. ในการเก็บข้อมูลที่เป็นตัวเลข ใช้รูปแบบที่เป็นจำนวนเต็ม หากต้องการใช้ข้อมูลตัวเลขในรูปแบบที่เป็นทศนิยม ก็ต้องเก็บในรูปแบบ Fixed-Point เนื่องจากจะสามารถนำ Library ที่พัฒนาขึ้นนั้นไปใช้ได้ทั้งบน DSP Chip ที่รองรับ และที่ไม่รองรับการคำนวณ Floating-Point
4. ไม่สามารถใช้ฟังก์ชั่นทางคณิตศาสตร์มาตรฐาน ที่ใช้บน PC ได้ (เช่น ฟังก์ชั่นทางคณิตศาสตร์ ที่อยู่ใน math.h) เนื่องจากในการพัฒนาบน DSP นั้น ไม่มีฟังก์ชั่น เหล่านั้นให้ใช้
5. ทำการทดสอบส่วนตรวจจับสัญญาณ โดยรายละเอียดต่างๆในการทดสอบ จะอธิบายในบทที่เกี่ยวกับการทดสอบ เป็นลำดับถัดไป
6. ค่าแต่ละ Sample ของสัญญาณที่รับเข้ามาจะต้องอยู่ในช่วง -4095 ... +4095
7. การตรวจจับสัญญาณ DTMF อิงตามเอกสาร ITU-T Recommendation Q.24 (AT&T)
8. การตรวจจับสัญญาณ R2 MF อิงตามเอกสาร ITU-T Recommendation Q.455
9. การตรวจจับสัญญาณ CPT อิงตามเอกสาร ITU-T Recommendation E.180/ Q.35

เมื่อตัว Library สามารถทำงานได้บน PC แล้ว ก็จะทำการย้ายไปพัฒนากับโปรแกรมที่ใช้ในการพัฒนา Library บน DSP Chip เป็นลำดับต่อไป เพื่อให้สามารถทำงานได้บน DSP Chip และให้สามารถตรวจจับสัญญาณได้อย่างถูกต้อง

ในการสร้างส่วนตรวจจับสัญญาณโทรศัพท์แต่ละประเภทนั้น จะทำแยกเป็น 3 Library โดยจะมีขั้นตอนในการตรวจจับคล้ายๆกัน ต่างกันที่การตรวจสอบจังหวะของสัญญาณ, การตรวจสอบระดับพลังงานของสัญญาณ, และการตรวจสอบความถี่ของสัญญาณ ซึ่งจะถูกกำหนดเป็นค่าคงที่ของแต่ละสัญญาณ



รูปที่ 6-1 แผนภาพ ขั้นตอนในการตรวจจับสัญญาณโทรศัพท์



รูปที่ 6-2 การนำสัญญาณที่รับเข้ามาไปตรวจสอบ

จากรูปที่ผ่านมาจะเห็นว่าสัญญาณที่รับเข้ามาจะถูกนำไปตรวจสอบยังส่วนต่างๆ ได้แก่ ตรวจสอบจังหวะของสัญญาณ, ตรวจสอบระดับค่าพลังงานของสัญญาณ, และตรวจสอบหาความถี่สัญญาณไซน์ส่วนประกอบด้วย Goertzel Algorithm ในการตรวจสอบด้วย Goertzel Algorithm นั้น จะมีการนำสัญญาณที่รับเข้ามาไปทำ Normalization ก่อน เพื่อขยายสัญญาณให้มีค่าสูงสุดที่ระดับ +4095 ซึ่งเหตุผลที่ต้องมีการทำ Normalization ก่อนนำสัญญาณไปตรวจสอบด้วย Goertzel Algorithm นั้น จะได้อธิบายในหัวข้อที่เกี่ยวกับการนำ Goertzel Algorithm มาประยุกต์ใช้

6.1 การตรวจสอบจังหวะของสัญญาณ

จังหวะของสัญญาณจำเป็นต้องให้รับการตรวจสอบให้เป็นไปตามมาตรฐานที่ได้กำหนดไว้ของแต่ละประเภทสัญญาณ ดังที่ระบุไว้ในส่วนของทฤษฎี

6.2 การตรวจสอบระดับค่าพลังงานของสัญญาณ

เนื่องจากในมาตรฐานของสัญญาณแต่ละประเภทนั้นกำหนดหน่วยของการวัดระดับค่าพลังงานของสัญญาณ ในหน่วย dBm ซึ่งเป็นหน่วย Decibel ที่อ้างอิงกับระดับ Power 1 mW ซึ่งมีสูตรการคำนวณดังต่อไปนี้

$$dBm = 10 \log (P/1\,mW)$$
$$dBm = 20 \log (rms/ref)$$
$$= 20 \log (\sqrt{avgsq} / ref) = 20 \log(\sqrt{avgsq}) - 20 \log (ref)$$
$$= 10 \log (avgsq) - m0 \quad ; m0 = 20 \log(\sqrt{avgsq(ref\text{-}samp)})$$

dBm : ระดับค่าพลังงาน ในหน่วย Decibel, อ้างอิงกับระดับค่าพลังงาน 1 mW

P : ระดับค่าพลังงาน ในหน่วย Watt

rms : Root Mean Square = $\sqrt{avgsq}$ = รากที่สองของ avgsq

avgsq : ค่าเฉลี่ยกำลังสอง = $(1/n) * \Sigma (Sample)^2$

ref : 1 mW

$\sqrt{avgsq}$: rms

ref-samp : Sample อ้างอิง ที่ใช้ในการคำนวณระดับค่าพลังงานอ้างอิง

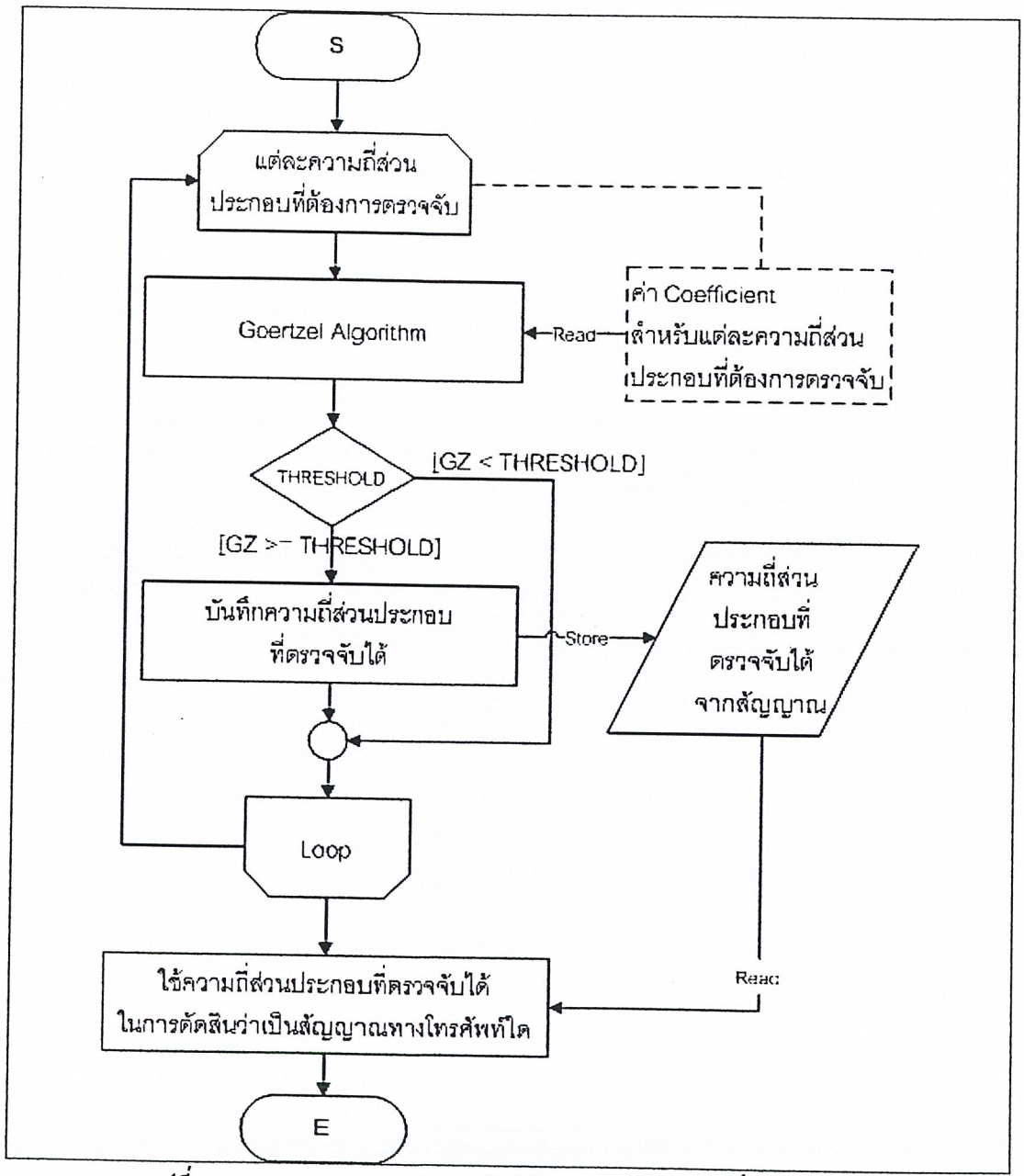
ตารางที่ 6-1 สูตรที่ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ

สำหรับระดับค่าพลังงานอ้างอิงนั้น ตามเอกสาร ITU-T Recommendation G.711 ได้ระบุให้ใช้ Samples ดังต่อไปนี้ ได้แก่ 1120, 2624, 2624, 1120, -1120, -2624, -2624, -1120 ในการคำนวณระดับค่าพลังงานอ้างอิง, 1mW ที่จะนำไปใช้ในการคำนวณระดับค่าพลังงาน (dBm) ของสัญญาณที่ผ่านการขยายด้วย A-Law Companding มาแล้ว ซึ่งสัญญาณที่ผ่านการขยายด้วย A-Law Companding นั้น แต่ละ Sample ของสัญญาณจะมีค่าอยู่ในช่วง -4095 ถึง +4095

6.3 การนำ Goertzel Algorithm มาประยุกต์ใช้

เมื่อส่วนตรวจสอบจังหวะของสัญญาณ และส่วนตรวจสอบระดับค่าพลังงานของสัญญาณ ได้รับรองแล้วว่าทั้ง จังหวะ และ ระดับค่าพลังงาน ของสัญญาณนั้นถูกต้อง เป็นไปตามมาตรฐานของแต่ละประเภทสัญญาณ ลำดับต่อไปก็จะเป็นการตรวจหาสัญญาณไซน์ส่วนประกอบ ของสัญญาณ เช่นถ้าเป็นการตรวจจับสัญญาณ DTMF ก็จะนำ Sample ในส่วนที่เป็น Signal Duration โดยจะใช้เพียง 250 Samples ท้ายของ Signal Duration มาทำการขยายสัญญาณ (Normalization) แล้วจากนั้นจึงจะทำการหาสัญญาณ

ไซน์ส่วนประกอบ โดยใช้ Goertzel Algorithm หรือหากถ้าเป็นการตรวจจับสัญญาณ R2 MF นั้นก็จะใช้ทีละ 250 Samples ของสัญญาณต่อเนื่องกันไป ซึ่งสาเหตุที่ต้องใช้จำนวน Sample ที่คงที่ และทำการขยายสัญญาณก่อนนั้น ก็เพื่อที่ให้ผลลัพธ์จาก Goertzel Algorithm สามารถใช้เปรียบเทียบกับค่า THRESHOLD ที่ถูกเลือกมาได้อย่างถูกต้อง แล้วจากนั้นจึงจะทำการหาสัญญาณไซน์ส่วนประกอบ โดยใช้ Goertzel Algorithm ตามรูปต่อไปนี้



รูปที่ 6-3 การนำ Goertzel Algorithm ไปใช้ในการตรวจจับความถี่ของสัญญาณ

ในการตรวจจับความถี่ของสัญญาณนั้น จำเป็นที่จะต้องคำนวณค่า Coefficient ของแต่ละความถี่ที่ต้องการตรวจจับ (เช่น 697Hz, 1209Hz, 1336Hz, เป็นต้น) โดยใช้สมการ 3-3 และเนื่องจาก Goertzel Algorithm นั้นสามารถตรวจจับได้ครั้งละ 1 ความถี่ใดๆเท่านั้น ดังนั้นจึงต้องมีการใช้การวนลูปเข้าจัดการในการตรวจจับแต่ละความถี่ที่ต้องการ เช่น ในการตรวจจับสัญญาณ DTMF ซึ่งมีความถี่ของ High-Frequency Group 4 ความถี่ และความถี่ของ Low-Frequency Group อีก 4 ความถี่ ดังนั้น ในการตรวจจับจะต้องวนลูป การตรวจจับ 8 ครั้ง โดยในแต่ละครั้ง ก็จะใช้ ค่า Coefficient ที่คำนวณมาจากแต่ละความถี่ที่ต้องการตรวจจับ เป็นต้น

ในแต่ล่ะ Sample ที่เข้ามา (1...250) นั้น จะทำการคำนวณด้วยสมการ 3-1 จนกระทั่งครบทุก Sample แล้วจะทำการคำนวณต่อด้วย 3-2 อีก 1 ครั้ง โดยผลลัพธ์ที่ได้ในการคำนวณครั้งสุดท้าย จากสมการ 3-2 นี้ จะถูกนำไปเทียบกับค่า THRESHOLD ที่เตรียมไว้แล้ว ว่ามีค่ามากกว่าค่า THRESHOLD หรือไม่ เพื่อใช้ในการตัดสินว่าสัญญาณนั้นๆ ประกอบด้วยความถี่ที่ต้องการที่ต้องการตรวจจับ หรือไม่ โดยการประเมินหาค่า THRESHOLD นั้นจะได้อธิบายในลำดับต่อไป

เมื่อทราบแล้วว่า สัญญาณที่รับเข้ามานั้น ประกอบด้วยความถี่ส่วนประกอบใดบ้าง ก็จะนำข้อมูลเหล่านี้ไปประกอบการพิจารณาว่า ความถี่ส่วนประกอบเหล่านั้น แทนด้วยหมายเลขใดของสัญญาณทางโทรศัพท์

6.4 การสร้าง ส่วนตรวจจับสัญญาณ DTMF (DTMFD)

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าคงที่ ที่จะใช้ในการตรวจจับสัญญาณ DTMF โดยค่าคงที่แต่ละตัวมีความหมายดังต่อไปนี้

1. Uns threshold_f เก็บค่า THRESHOLD ที่ใช้กับ High-frequency group
2. Uns threshold_g เก็บค่า THRESHOLD ที่ใช้กับ Low-frequency group
3. Uns threshold_v ใช้ในการเทียบค่าสูงสุดของสัญญาณ
4. MdInt sigOprMin เก็บระยะเวลาของ Signal Duration ต่ำสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน Sample)
5. MdInt sigPauseMin เก็บระยะเวลาของ Pause Duration ต่ำสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน Sample)
6. MdInt sigVelocity เก็บระยะเวลาของ Signal Duration + Pause Duration (Signal Velocity)ต่ำสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน Sample)
7. MdInt sigIntMax เก็บระยะเวลาของ Signal Interruption สูงสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน Sample)
8. Int coeff_dtmf[8] เก็บค่า Coefficient สำหรับ Goertzel Algorithm โดยเป็นค่า Coefficient ของความถี่ต่อไปนี้ตามลำดับ ได้แก่ 1209 Hz, 1336 Hz, 1477 Hz, 1633 Hz, 697 Hz, 770 Hz, 852 Hz, และ 941 Hz ซึ่งค่าของแต่ละตัวจะต้องถูก Shift left ไปแล้ว 17 บิต เพื่อรักษาความละเอียดหลังจุดทศนิยม

9. Int dBm0_OprMax เก็บค่าของระดับพลังงานของสัญญาณสูงสุดที่ยอมรับได้
   (มีหน่วยเป็น จำนวน dBm x 100)
10. Int dBm0_OprMin เก็บค่าของระดับพลังงานของสัญญาณต่ำสุดที่ยอมรับได้
    (มีหน่วยเป็น จำนวน dBm x100)
11. Int dBm0_NonOprMax เก็บค่าของระดับพลังงานของสัญญาณสูงสุดที่ไม่ยอมรับ หมายถึง หาก ระดับพลังงานของสัญญาณต่ำกว่าค่านี้ ก็จะไม่ทำการตรวจสอบสัญญาณให้
    (มีหน่วยเป็น จำนวน dBm x100)
12. Int log_scale ใช้ในการคำนวณค่า Log
    (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
13. Int dBm0_A_law ค่าระดับพลังงานอ้างอิง ที่ 1 dBm สำหรับ A-Law Companding
    (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
14. Int dBm0_Mju_law ค่าระดับพลังงานอ้างอิง ที่ 1 dBm สำหรับ μ-Law Companding
    (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
15. Int log10_2nm1[27] ใช้ในการคำนวณค่า Log
    (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
16. Int log10_2n[32] ใช้ในการคำนวณค่า Log
    (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)

```
typedef struct
{
        Uns threshold_f;        //to used with high-group 20M  6M
        Uns threshold_g;        //to used with low-group 8M //15M  //5M    //3M
        Uns threshold_v;
        MdInt sigOprMin;        // 320 samp = 40mS
        MdInt sigPauseMin;      // 320 samp = 40mS
        MdInt sigVelocity;      // 744 samp = 93mS
        MdInt sigIntMax;        // 80 samp = 10mS    ,allowed inturrupt period
        Int coeff_dtmf[8];

        Int dBm0_OprMax;                //0dBm=0
        Int dBm0_OprMin;                //-25dBm=-2500
        Int dBm0_NonOprMax;                     //-25dBm=-2500

        Int log_scale;                  //2^17
        Int dBm0_A_law;
        Int dBm0_Mju_law;
        Int log10_2nm1[27];
        Int log10_2n[32];

}DTMFD_KONST;
```

ตารางที่ 6-2 ***Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนตรวจจับสัญญาณ DTMF***

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าตัวแปรที่ใช้ในการตรวจจับสัญญาณ DTMF โดยตัวแปรแต่ละตัวมีความหมายดังต่อไปนี้

1. MdInt detectionBuffer[MaxQueueItem] เก็บ Sample ช่วงท้ายของ Signal Duration โดยจะเป็นโครงสร้างข้อมูลแบบ Circular Queue เพื่อนำไปตรวจสอบด้วย Goertzel Algorithm
2. MdInt sPtr เป็น Start Index ใช้ชี้ detectionBuffer
3. MdInt ePtr เป็น End Index ใช้ชี้ detectionBuffer
4. MdInt stateQueue[MaxStateQueueItem] ใช้ตรวจสอบสถานะของสัญญาณ ณ Sample ปัจจุบันว่าอยู่ในช่วง Signal Duration หรือ Pause Duration เพื่อนำข้อมูลไปใช้ตรวจสอบจังหวะของสัญญาณ โดยจะเป็นโครงสร้างข้อมูลแบบ Circular Queue
5. MdInt state_sPtr เป็น Start Index ใช้ชี้ stateQueue
6. MdInt state_ePtr เป็น End Index ใช้ชี้ stateQueue
7. MdInt lastItem[MaxLastItem] เก็บ Sample ช่วงท้ายของสัญญาณ (ไม่ว่าจะเป็น Signal Duration หรือ Pause Duration ก็ตาม) เพื่อใช้ในการตัดสินว่า Sample นั้นเป็น Pause Sample หรือ Non-pause Sample โดยจะเป็นโครงสร้างข้อมูลแบบ Circular Queue
8. MdInt lastItem_sPtr เป็น Start Index ใช้ชี้ lastItem
9. MdInt lastItem_ePtr เป็น End Index ใช้ชี้ lastItem
10. MdInt cntS เก็บจำนวน Sample ในช่วง Signal Duration (เพื่อนำข้อมูลไปใช้ตรวจสอบจังหวะของสัญญาณ)
11. MdInt cntZ เก็บจำนวน Sample ในช่วง Pause Duration (เพื่อนำข้อมูลไปใช้ตรวจสอบจังหวะของสัญญาณ)

```
#define MaxQueueItem 251  // e.g. 201 defined = 200 usable)
#define MaxStateQueueItem 91
#define MaxLastItem 8

typedef struct{
        MdInt detectionBuffer[MaxQueueItem];
        MdInt sPtr; //0
        MdInt ePtr; //1
        MdInt stateQueue[MaxStateQueueItem];
        MdInt state_sPtr; //0
        MdInt state_ePtr; //1
        MdInt lastItem[MaxLastItem];
        MdInt lastItem_sPtr; //0
        MdInt lastItem_ePtr; //1

        MdInt cntS;    //0, count for non-pause sample
        MdInt cntZ;    //0, count for pause sample
} DTMFD_MBLK; //, *dtmfd_handle;
```

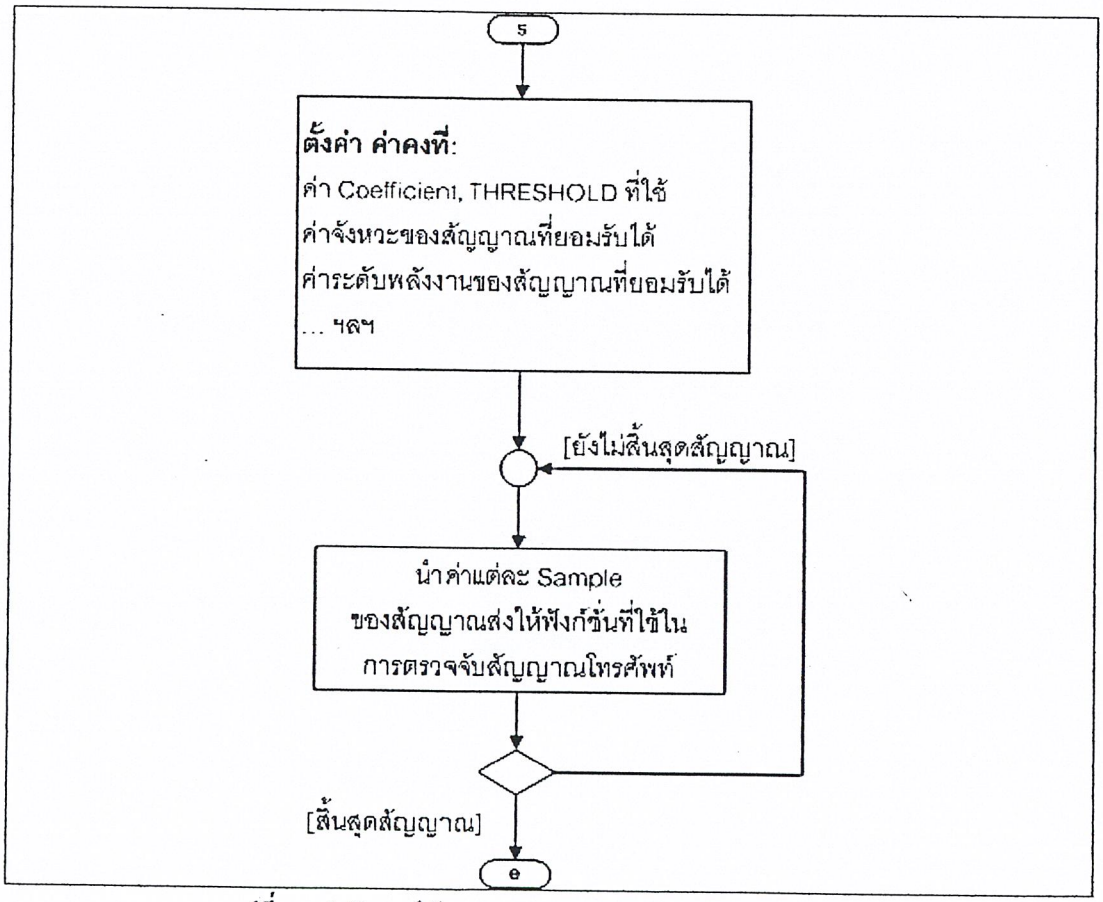
ตารางที่ 6-3 Structure เพื่อใช้ในการเก็บค่าตัวแปรของส่วนตรวจจับสัญญาณ DTMF

ในการทำงานนั้นเริ่มต้น ผู้ใช้จะต้องทำการตั้งค่าเริ่มต้นให้กับ Library โดยจะต้องทำการกำหนดค่าคงที่ต่างๆที่ต้องการใช้ จากนั้นเมื่อต้องการตรวจจับสัญญาณก็จะต้องทำการเรียกใช้ฟังก์ชั่นที่ใช้ในการตรวจจับสัญญาณ โดยส่ง Sample ของสัญญาณเข้าไป ครั้งละ 1 Sample ซึ่งฟังก์ชั่นที่ใช้ในการตรวจจับสัญญาณ จะคืนค่ากลับสู่ผู้ใช้ โดยมีความหมายเป็นสัญญาณ DTMF ดังนี้

Return Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**>= 100**
ความหมาย	0	1	2	3	4	5	6	7	8	9	*	#	A	B	C	D	(n/a)

ตารางที่ 6-4 ความหมายค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ DTMF

ค่าที่ได้จากการเรียกฟังก์ชั่นที่ใช้ในการตรวจจับสัญญาณ หากเท่ากับ 100 แล้ว จะหมายถึงยังไม่สามารถตรวจจับสัญญาณ DTMF ใดๆ ได้ในขณะนั้น ส่วนค่าอื่นๆที่เหลือนั้น ดูได้จากตาราง



รูปที่ 6-4 ลำดับการใช้งาน Library การตรวจจับสัญญาณ DTMF

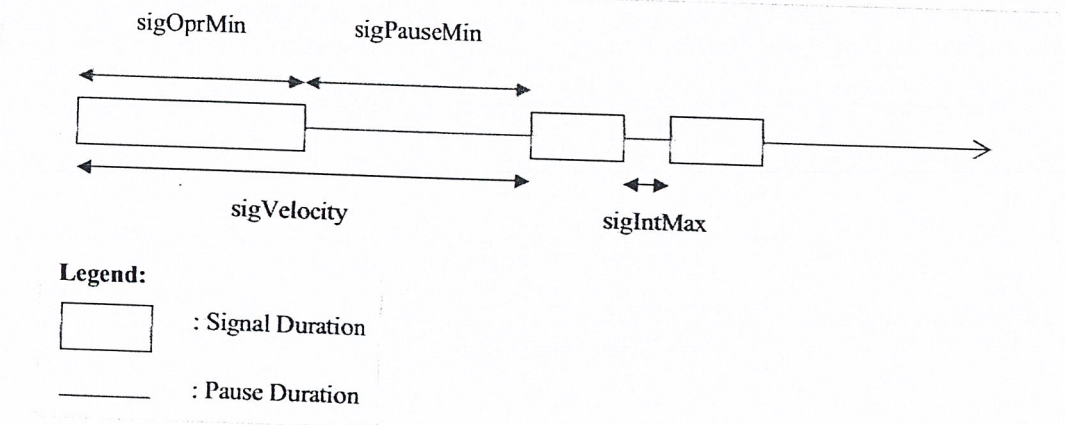
6.4.1 อัลกอริทึมของการตรวจจับสัญญาณ DTMF

การตรวจจับสัญญาณ DTMF นั้น จะประกอบด้วยการตรวจสอบจังหวะของสัญญาณ, การตรวจสอบระดับค่าพลังงานของสัญญาณ, และการตรวจสอบความถี่ส่วนประกอบของสัญญาณ ตามลำดับ

การตรวจสอบจังหวะของสัญญาณ นั้นจะใช้ค่าที่กำหนดมาจากผู้ใช้ 4 ค่า ในการประเมินและตัดสินว่า สัญญาณที่เข้ามานั้น เป็นไปตามจังหวะสัญญาณที่ยอมรับได้หรือไม่ ค่าที่ใช้ประกอบด้วย

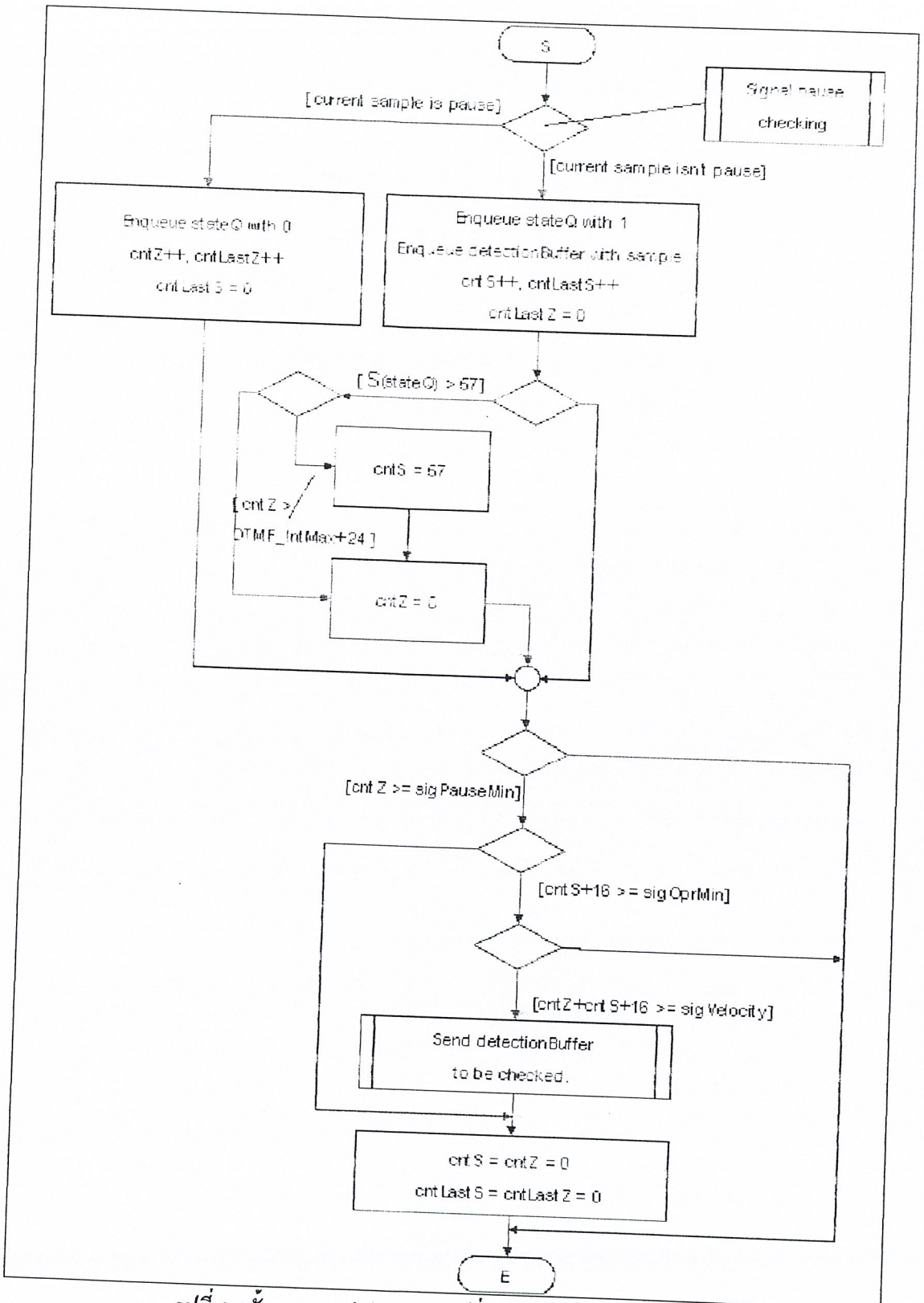
1. sigOprMin ซึ่งช่วง Signal Duration ของสัญญาณจะต้องมากกว่าค่านี้
2. sigPauseMin ซึ่งช่วง Pause Duration ของสัญญาณจะต้องมากกว่าค่านี้
3. sigVelocity ซึ่งช่วง Signal Duration + Pause Duration ของสัญญาณจะต้องมากกว่าค่านี้
4. sigIntMax ซึ่งช่วง Interrupt Duration ที่เกิดขึ้นระหว่างสัญญาณในช่วง Signal Duration จะต้องน้อยกว่าค่านี้

โดยสัญญาณ 1 สัญญาณ ของ DTMF นั้น จะประกอบด้วย Signal Duration และ Pause Duration (ซึ่งก็คือ Gap Duration ระหว่าง Signal Duration ของ DTMF สัญญาณแรก กับ Signal Duration ของ DTMF สัญญาณถัดไปนั้นเอง) การกำหนดค่า 4 ค่าที่กล่าวมานั้น ทำการกำหนดเป็นจำนวน Sample โดยคิดจากระยะเวลาที่ต้องการ คูณกับ Sampling Rate ที่ใช้ เช่น ต้องการ Signal Duration ไม่น้อยกว่า 40 mS ก็จะต้องกำหนดค่า sigOprMin เท่ากับ 40 mS x 8000 Hz = 320 Samples เป็นต้น



รูปที่ 6-5 ค่าคงที่ ที่ใช้ในการกำหนดจังหวะของสัญญาณ DTMF

เมื่อฟังก์ชั่นในการตรวจสอบสัญญาณ รับ Sample เข้ามาแล้ว ก็จะทำการเรียกฟังก์ชั่นย่อยที่ทำหน้าที่ตัดสินว่า Sample นั้นๆ เป็น Pause Sample หรือ Non-Pause Sample (Signal Sample) ซึ่งเป็นการยากที่ดูเพียงค่าของ Sample นั้นๆ เนื่องจากอาจมีสัญญาณรบกวนเข้ามาชั่วขณะ ที่จะทำให้การตัดสินผิดพลาดได้ ดังนั้นจึงมีการสร้างฟังก์ชั่นย่อยนี้ขึ้นมา โดยจะพิจารณาถึง Sample อื่นๆ ก่อนหน้า Sample ปัจจุบันด้วย (lastItem Queue) เมื่อพิจารณา Sample ว่าเป็นแบบใดแล้ว การทำงานต่อไปจะเป็นดังนี้



รูปที่ 6-6 ขั้นตอนการทำงานของฟังก์ชั่นตรวจจับสัญญาณ DTMF

- หากเป็น Non-Pause Sample: ก็จะทำการ Enqueue stateQueue ด้วย 1, Enqueue detectionBuffer ด้วย Sample นั้น, ทำการบวกความยาวของ Signal (cntS) ด้วย 1,ทำการบวกความยาวของ Signal แบบต่อเนื่อง(cntLastS) ด้วย 1, และ ตั้งค่าความยาวของ Pause แบบต่อเนื่อง(cntLastZ) ให้เป็น 0
- หากเป็น Pause Sample: ก็จะทำการ Enqueue stateQueue ด้วย, ทำการบวกความยาวของ Pause (cntZ) ด้วย 1,ทำการบวกความยาวของ Pause แบบต่อเนื่อง(cntLastZ) ด้วย 1, และ ตั้งค่าความยาวของ Signal แบบต่อเนื่อง(cntLastS) ให้เป็น 0

จากนั้น ก็จะทำการรวมค่าใน stateQueue (เป็นอาร์เรย์ขนาด 91 ช่อง) ซึ่งหาก

- หาก ผลรวมของค่าใน stateQueue มากกว่า 57 และค่า cntZ มากกว่า ค่า sigIntMax +24 แล้ว แสดงว่า เป็นตำแหน่งเริ่มต้นของสัญญาณ DTMF จะทำการตั้งค่า cntS = 57 และ cntZ = 0
- หาก ผลรวมของค่าใน stateQueue มากกว่า 57 และค่า cntZ น้อยกว่าหรือเท่ากับ ค่า sigIntMax +24 แล้ว แสดงว่า ช่วง Pause Duration ที่ผ่านไปล่าสุดเป็น Interrupt Duration จากนั้นจะทำการตั้งค่า cntZ = 0

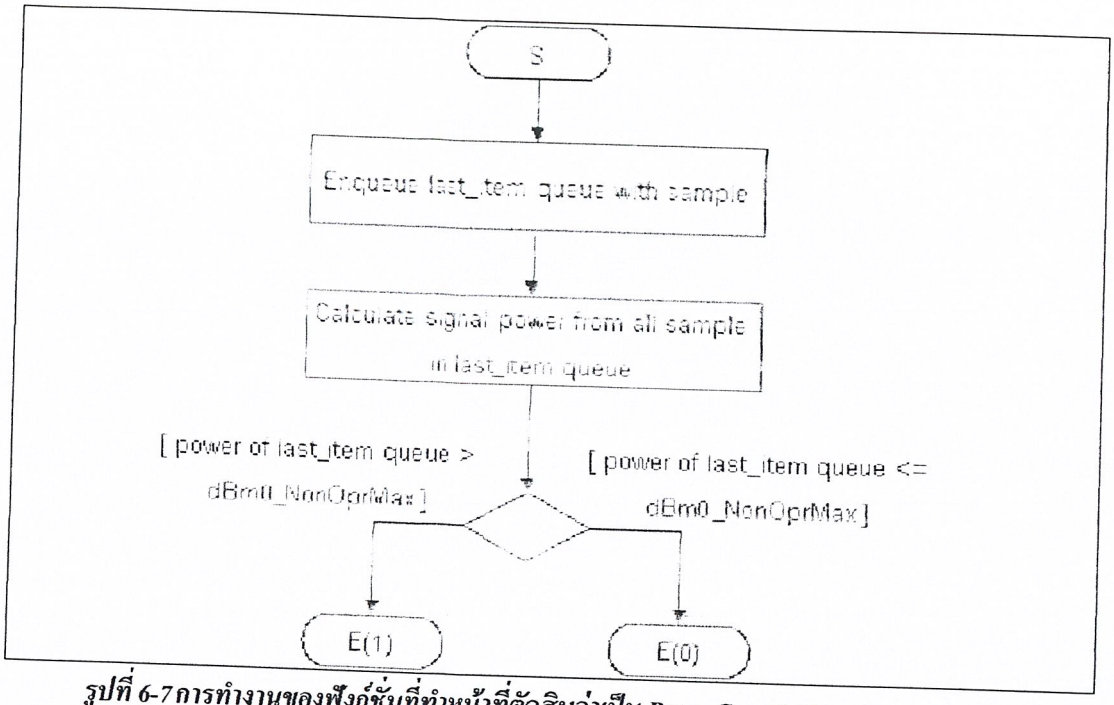
ต่อมา จะเทียบว่าค่า cntZ >= sigPauseMin หรือไม่ หากใช่ก็จะทำงานต่อไป หากไม่ก็จะออกจากฟังก์ชั่นไป

จากนั้นจะเทียบว่าค่า *cntS+16 >= sigOprMin* หรือไม่ หากใช่ก็จะทำงานต่อไป หากไม่ก็จะตั้งค่าตัวแปรภายในเพื่อเตรียมตรวจสอบสัญญาณ DTMF ตัวต่อไป และออกจากฟังก์ชั่น

สุดท้ายจะเทียนว่าค่า cntZ+cntS+16 >= sigVelocity หรือไม่ หากไม่ก็จะออกจากฟังก์ชั่น แต่ถ้าใช้ ก็จะทำขั้นตอนต่อไปนี้

- ส่งข้อมูลใน detectionBuffer ซึ่งก็คือช่วง 250 Samples สุดท้ายของ Signal Duration ไปทำการตรวจสอบระดับค่าพลังงานของสัญญาณ และตรวจสอบหาความถี่ส่วนประกอบ ตามลำดับ
- จะตั้งค่าตัวแปรภายในเพื่อเตรียมตรวจสอบสัญญาณ DTMF ตัวต่อไป

ก่อนที่จะอธิบายถึง การตรวจสอบระดับค่าพลังงานของสัญญาณ และตรวจสอบหาความถี่ส่วนประกอบ จะขออธิบายถึงฟังก์ชั่นย่อยที่ทำหน้าที่ตัดสินว่า Sample ที่เข้ามาเป็น Pause Sample หรือ Non-Pause Sample โดยฟังก์ชั่นนี้จะนำ Sample ที่เข้ามาเข้าไปใน lastItem queue (เป็นอาร์เรย์ขนาด 8 ช่อง) แล้วทำการคิดระดับค่าพลังงานของสัญญาณของทุก Sample ที่อยู่ใน lastItem queue จากนั้น นำค่าระดับพลังงานที่ได้ (ในหน่วย dBm) เทียบกับค่า dBm0_NonOprMax หากมากกว่าหรือเท่ากับ ก็จะตัดสินว่าเป็น Non-Pause Sample (Return(1))แต่ถ้าหากค่าระดับพลังงานที่ได้ น้อยกว่าค่า dBm0_NonOprMax ก็จะตัดสินว่าเป็น Pause Sample (Return(0))

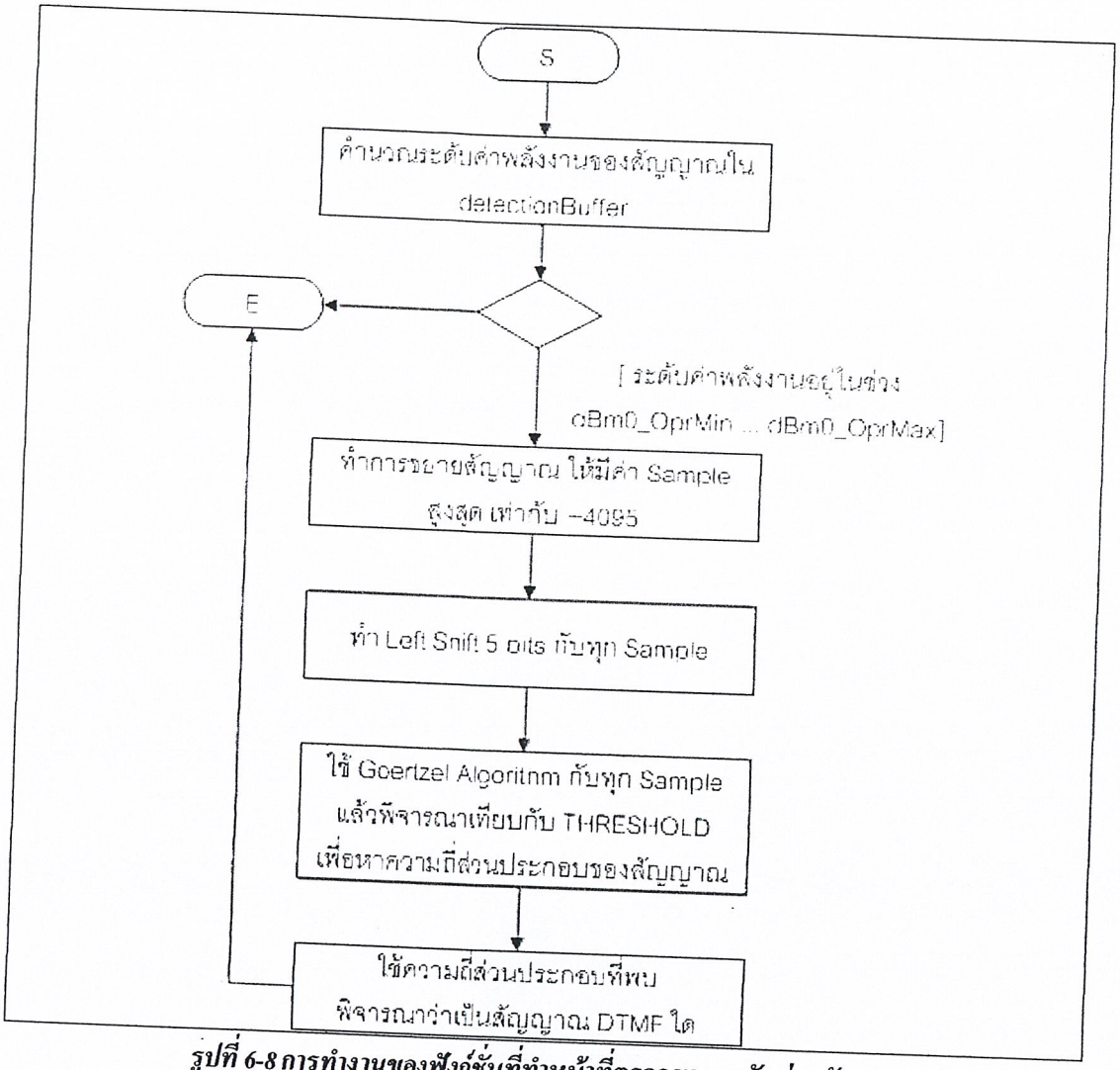


รูปที่ 6-7 การทำงานของฟังก์ชั่นที่ทำหน้าที่ตัดสินว่าเป็น Pause Sample/ Non-Pause Sample

เมื่อ detectionBuffer ซึ่งก็คือช่วง 250 Samples สุดท้ายของ Signal Duration ถูกส่งให้กับฟังก์ชั่นที่ใช้ในการตรวจสอบระดับค่าพลังงานของสัญญาณ และตรวจสอบหาความถี่ส่วนประกอบ ในการตรวจสอบระดับค่าพลังงานของสัญญาณนั้น จะใช้สูตรตามที่ได้อธิบายไปแล้วในส่วนของทฤษฎี จากนั้นหากระดับค่าพลังงานของสัญญาณเป็นไปตามที่มาตรฐานกำหนด ก็จะทำการหาค่าที่มากที่สุดที่อยู่ใน detectionBuffer เพื่อนำไปหาตัวคูณในการใช้คูณทุก Sample ใน detectionBuffer เพื่อขยายสัญญาณ ดังกรอบด้านล่าง จากนั้นนำค่าใน detectionBuffer ที่ทำการขยายแล้วไปทำการตรวจสอบด้วย Goertzel Algorithm โดยใช้ค่า Coefficient ที่กำหนดโดยผู้ใช้ ซึ่งก่อนนำการตรวจสอบด้วย Goertzel Algorithmนั้น จะทำการ Right Shift 5 bits เพื่อลดขนาดให้ค่าสูงสุดเป็น +127 เหตุที่ต้องขยายให้สัญญาณมีค่าสูงสุดที่ +4095 ก็เพื่อว่า หากสัญญาณที่เข้ามามีขนาดไม่สูง จะทำให้ค่าที่ได้จาก Goertzel Algorithm ไม่สามารถเทียบกับ THRESHOLD ที่ใช้ได้ (เพราะ THRESHOLD ที่ใช้ คำนวณจากขนาดสัญญาณที่มีค่า Sample สูงสุดเท่ากับ +4095) และเหตุที่ต้อง Right Shift 5 bits ก็เพราะหากค่า Sample มีขนาดสูงแล้ว ค่าที่ได้จาก Goertzel Algorithm จะมีขนาดใหญ่มากจนไม่สามารถเก็บในตัวแปรที่ใช้ได้

ตัวคูณ = 4095 x 1000 / (ค่าที่มากที่สุด)
x 1000 : เพื่อรักษาค่าหลังจุดทศนิยม เป็นจำนวน 3 ตำแหน่ง โดยเมื่อนำค่าตัวคูณ ไปคูณกับแต่ละ Sample ของสัญญาณแล้ว ก็จะทำการหารกลับด้วย 1000 อีกครั้งหนึ่ง

ตารางที่ 6-5 การหาค่าตัวคูณ เพื่อนำไปใช้ในการขยายสัญญาณ



รูปที่ 6-8 การทำงานของฟังก์ชั่นที่ทำหน้าที่ตรวจสอบระดับค่าพลังงาน และหาความถี่ส่วนประกอบ

6.5 การสร้าง ส่วนตรวจจับสัญญาณ R2 MF (R2MFD)

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าคงที่ ที่จะใช้ในการตรวจจับสัญญาณ R2 MF โดยค่าคงที่แต่ละตัวมีความหมายดังต่อไปนี้

1. Uns threshold_f เก็บค่า THRESHOLD ที่ใช้กับ Forward group
2. Uns threshold_g เก็บค่า THRESHOLD ที่ใช้กับ Backward group
3. Uns threshold_v ใช้ในการเทียบค่าสูงสุดของสัญญาณ
4. Int coeff_r2mf_fwd[6] เก็บค่า Coefficient สำหรับ Goertzel Algorithm โดยเป็นค่า Coefficient ของความถี่สำหรับ Forward R2 MF ได้แก่ 1380 Hz, 1500 Hz, 1620 Hz, 1740 Hz, 1860 Hz, และ 1980 Hz ซึ่งค่าของแต่ละตัวจะต้องถูก Shift left ไปแล้ว 17 บิต เพื่อรักษาความละเอียดหลังจุดทศนิยม

5. Int coeff_r2mf_bwd[6] เก็บค่า Coefficient สำหรับ Goertzel Algorithm โดยเป็นค่า Coefficient ของความถี่สำหรับ Backward R2 MF ได้แก่ 1140 Hz, 1020 Hz, 900 Hz, 780 Hz, 660 Hz, และ 540 Hz ซึ่งค่าของแต่ละตัวจะต้องถูก Shift left ไปแล้ว 17 บิต เพื่อรักษาความละเอียดหลังจุดทศนิยม
6. Int dBm0_OprMax เก็บค่าของระดับพลังงานของสัญญาณสูงสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน dBm x 100)
7. Int dBm0_OprMin เก็บค่าของระดับพลังงานของสัญญาณต่ำสุดที่ยอมรับได้ (มีหน่วยเป็น จำนวน dBm x 100)
8. Int log_scale ใช้ในการคำนวณค่า Log (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
9. Int dBm0_A_law ค่าระดับพลังงานอ้างอิง ที่ 1 dBm สำหรับ A-Law Companding (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
10. Int dBm0_Mju_law ค่าระดับพลังงานอ้างอิง ที่ 1 dBm สำหรับ μ-Law Companding (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
11. Int log10_2nm1[27] ใช้ในการคำนวณค่า Log (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)
12. Int log10_2n[32] ใช้ในการคำนวณค่า Log (ใช้ในการคำนวณค่าระดับพลังงานของสัญญาณ)

```
typedef struct
{
        Uns threshold_f;        //to used with fwd-group //fwd 8M  //15M  //5M
        Uns threshold_g;        //to used with bwd-group //bwd 8M //15M  //5M
        Uns threshold_v;        // against signal level

        Int coeff_r2mf_fwd[6];
        Int coeff_r2mf_bwd[6];

        Int dBm0_OprMax;                //0dBm=0
        Int dBm0_OprMin;                //-31.50dBm=-3150

        Int log_scale;                  //2^17
        Int dBm0_A_law;
        Int dBm0_Mju_law;

        Int log10_2nm1[27];
        Int log10_2n[32];

}R2MFD_KONST;
```

ตารางที่ 6-6 Structure เพื่อใช้ในการเก็บค่าคงที่ของส่วนตรวจจับสัญญาณ R2 MF

ทำการสร้าง Structure เพื่อใช้ในการเก็บค่าตัวแปรที่ใช้ในการตรวจจับสัญญาณ R2 MF โดยตัวแปรแต่ละตัวมีความหมายดังต่อไปนี้

1. MdInt r2mf_detect_detectionBuffer[R2MF_DETECT_MaxQueueItem] เก็บ Sample ช่วงท้ายของ Signal Duration โดยจะเป็นโครงสร้างข้อมูลแบบ อาร์เรย์
2. MdInt r2mf_detect_sPtr ใช้เป็นตัวชี้อาร์เรย์

```
#define R2MF_DETECT_MaxQueueItem 251
typedef struct{

        MdInt r2mf_detect_detectionBuffer[R2MF_DETECT_MaxQueueItem];
        MdInt r2mf_detect_sPtr;   //0

}R2MFD_MBLK;
```

ตารางที่ 6-7 Structure เพื่อใช้ในการเก็บค่าตัวแปรของส่วนตรวจจับสัญญาณ R2 MF

ในการทำงานนั้น ผู้ใช้จะต้องทำการตั้งค่าเริ่มต้นให้กับ Library โดยจะต้องทำการกำหนดค่าคงที่ต่างๆที่ต้องการใช้ จากนั้นเมื่อต้องการตรวจจับสัญญาณก็จะต้องทำการเรียกใช้ฟังก์ชั่นที่ใช้ในการตรวจจับสัญญาณ โดยส่ง Sample ของสัญญาณเข้าไป ครั้งละ 1 Sample ซึ่งฟังก์ชั่นที่ใช้ในการตรวจจับสัญญาณ จะคืนค่ากลับสู่ผู้ใช้ โดยมีความหมายเป็นสัญญาณ R2 MF ดังนี้

ความหมายของค่าที่ได้คืนจากการเรียกใช้ฟังก์ชั่นตรวจจับสัญญาณ R2 MF

0x[1][2]
[1] – R2MF backward signal
[2] – R2MF forward signal

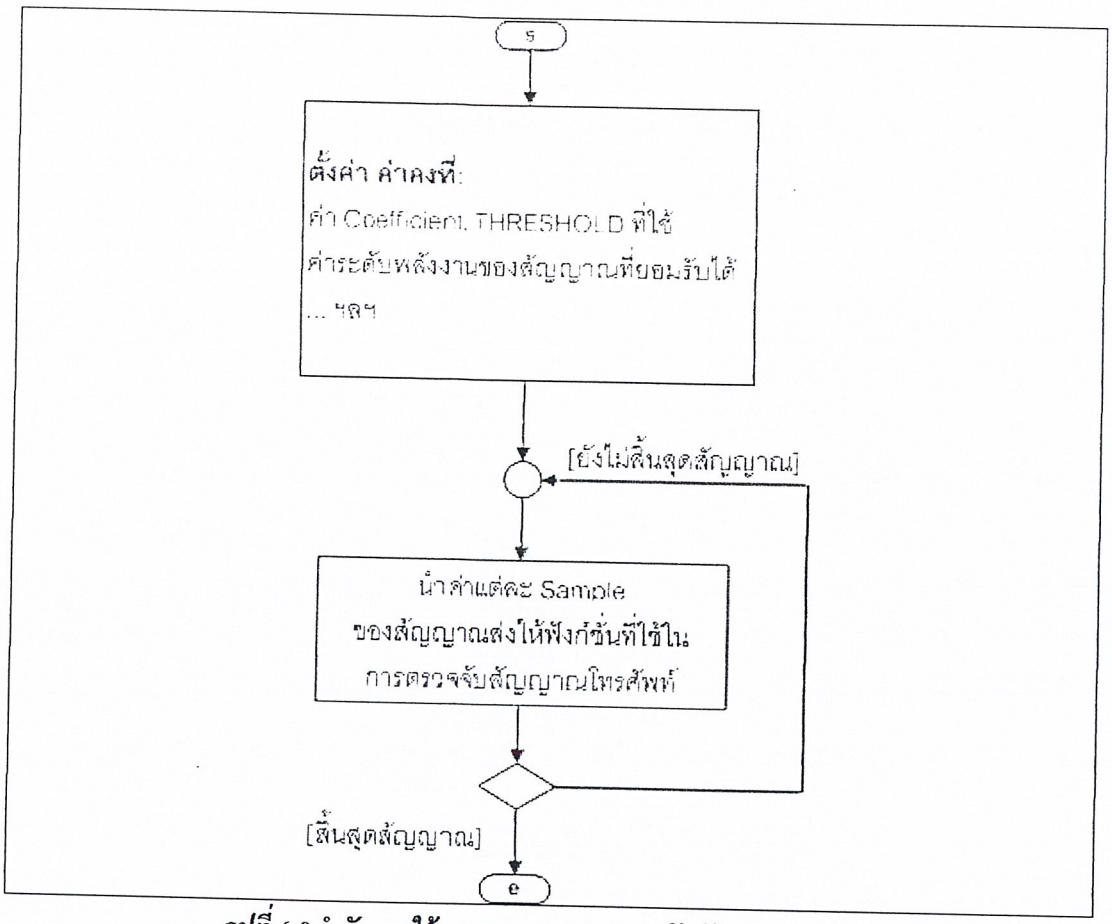
หมายเลขของสัญญาณ R2 MF ที่แทนด้วย Nibble [1] และ [2] (แสดงเป็นเลขฐาน 10)

Return Value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**0**
R2 MF	1	2	3	4	5	6	7	8	9	10	A'	B'	C'	D'	E'	unknown

ตารางที่ 6-8 ความหมายค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ R2 MF

Return Value	ความหมาย	หมายเหตุ
0x12	ตรวจพบ fwd:2 และ bwd:1	-
0x0F	ตรวจพบ fwd:E'	F in hex. = 15 in dec.
0xE5	ตรวจพบ fwd:5 และ bwd: D'	E in hex. = 14 in dec.

ตารางที่ 9-1 ตัวอย่างการตีความค่าที่คืนกลับจากการเรียกฟังก์ชั่นตรวจจับสัญญาณ R2 MF

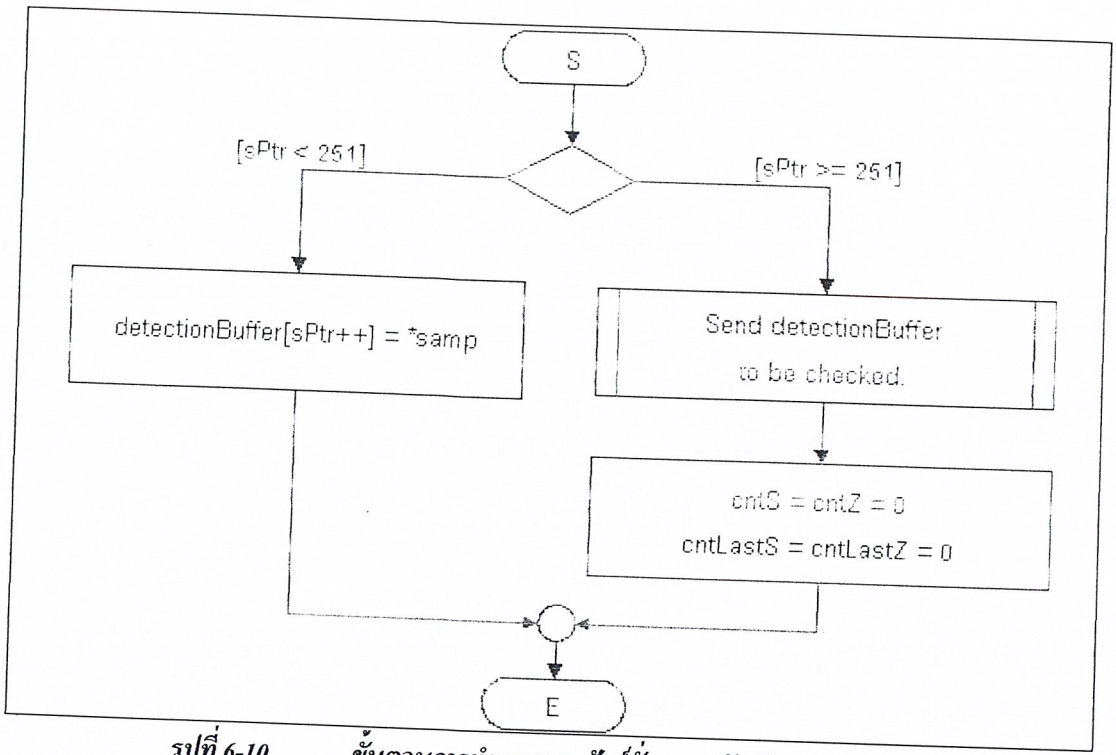


รูปที่ 6-9 ลำดับการใช้งาน Library การตรวจจับสัญญาณ R2 MF

6.5.1 อัลกอริทึมของการตรวจจับสัญญาณ R2 MF

การตรวจจับสัญญาณ R2 MF นั้น จะประกอบด้วยการตรวจสอบระดับค่าพลังงานของสัญญาณ, และการตรวจสอบความถี่ส่วนประกอบของสัญญาณ ตามลำดับ

เมื่อสัญญาณเข้ามานั้นจะดูว่าหาก detectionBuffer เต็ม ก็จะส่งข้อมูลใน detectionBuffer ซึ่งก็คือ ช่วง 250 Samples ของสัญญาณ ไปทำการตรวจสอบระดับค่าพลังงานของสัญญาณ และตรวจสอบหาความถี่ส่วนประกอบ ซึ่งขั้นตอนการทำงานนั้นก็คล้ายๆ กับส่วนตรวจสอบระดับค่าพลังงาน และตรวจสอบหาความถี่ส่วนประกอบ ของ การตรวจจับสัญญาณ DTMF ต่างกันที่ค่า dBm0_OprMin, ค่า dBm0_OprMax, ค่า Coefficient สำหรับ Goertzel Algorithm, และ ค่า THRESHOLD ที่ใช้



รูปที่ 6-10 ขั้นตอนการทำงานของฟังก์ชั่นตรวจจับสัญญาณ R2 MF

6.6 การสร้าง ส่วนตรวจจับสัญญาณ CPT (CPTDETECT)

การสร้าง CPT Detection Algorithm ให้สามารถทำงานแบบ Real Timeได้นั้นก็มีลักษณะคล้ายกับ การ Generation ก็คือ ในการ Call Function 1 ครั้งจะไม่สามารถตรวจจับสัญญาณได้ทั้งสัญญาณ ในการ Call Function ในการตรวจจับนั้นแต่ละครั้งก็จะส่ง samples เข้าไป block หนึ่ง พอ Call Function ในการตรวจจับครั้งถัดไปก็จะส่ง samples block ถัดไปเข้าไปตรวจสอบ เมื่อจำนวน samples ที่ถูกตรวจสอบเพียงพอต่อการระบุได้ว่าสัญญาณนั้นเป็นสัญญาณอะไร ก็จะมีการ set flag ตัวหนึ่งเพื่อบ่งบอกว่าสามารถตรวจจับสัญญาณได้แล้ว เพื่อการทำงานที่ถูกต้อง ในการ Call Function แต่ละครั้งเมื่อสิ้นสุดการ Call ครั้งนั้นก็จะมีการ save state ลงในตัวแปรเพื่อที่จะได้นำไปใช้ในการ Call Function นั้นครั้งถัดไปว่าได้ทำการตรวจจับไปถึงขั้นตอนไหนแล้ว

6.6.1 อัลกอรึทึมของการตรวจจับสัญญาณ CPT

```
typedef struct CPT_DetectParam
{
        int *data;
        unsigned int CntS;
        unsigned int CntZ;
        unsigned int InterruptCntS;
        unsigned int state;
        short PrevData[200];
```

```
        unsigned short isDetected;
        unsigned short isRejected;
        unsigned int result;
        int power;
        int CountPrev;
        /////use for Confirmation Tone and reject signal in state is not in START
state////
        unsigned short isConfirmation;        //for confirmation only
        unsigned short isSecondDetect;
        unsigned short isSecondReject;
        int SecondPower;
        unsigned short SecondResult;

        unsigned short StartKeep;
        unsigned short FirstStart;
        ////////////////////////////////////
} CPT_DetectParam;
```

ตารางที่ 6-9 ***Structure ที่ใช้ในการบันทึก State ของการทำ CPT Detection***

ขั้นตอนการตรวจสอบสัญญาณนั้นแบ่งเป็น 2 ขั้นตอน ดังนี้

1) ตรวจสอบจังหวะของสัญญาณ

```
CPTDETECT_Result DetectCPT(CPT_DetectParam* param)
{
        for(i=0;i<DETECT_LENGTH;i++)
        {
        int temp = param->data[i];
        if(isSilence(temp,CPTsilence) == 0)  //false(0) = not silence
        {

                if(param->CntZ <= CPT_INTR)
                {
                ...
                }
                else    //Complete 1 cycle signal
                {
                ...
                DetectSignal(param);
                ...
                }

                param->CntS++;

                if(param->CntS > DIAL_TONE_MIN_DURATION*SAMP_RATE)
                        //for detect dial tone duration
                {
                        DetectSignal(param);
```

```
            }

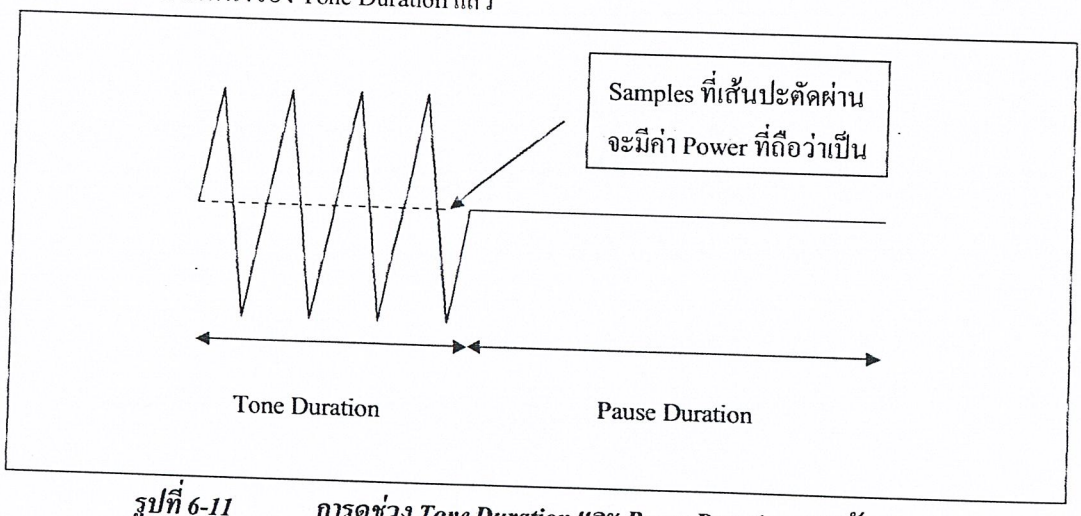
        }
        else    //silence
        {
                param->CntZ++;
        }

}
```

ตารางที่ 6-10 Source code อย่างย่อในการตรวจสอบ Timing ของ CPT

ใน Source Code ข้างบนจะเป็นวิธีการ แยกแยะระหว่างส่วนที่เป็น Tone Duration และ Pause Duration ซึ่งมีข้อกำหนดดังนี้

- Tone Duration คือส่วนของสัญญาณที่มีการเปลี่ยนแอมปลิจูด ของสัญญาณ และสัญญาณที่มีค่าระดับพลังงานของสัญญาณน้อยกว่าที่มาตรฐานกำหนดไว้แต่จำนวน Samples เป็นไปตามที่มาตรฐานระบุว่าเป็นสัญญาณรบกวนจะนับเป็นส่วนหนึ่งของ Tone Duration ด้วย ในกรณีนี้จะรวม Sample ที่มีค่าเป็น 0 และมีระดับพลังงานของสัญญาณน้อยกว่ามาตรฐานเป็นส่วนหนึ่งของ Tone Duration แล้ว



รูปที่ 6-11 การดูช่วง Tone Duration และ Pause Duration ของสัญญาณ

- Pause Duration คือส่วนของสัญญาณที่มีค่าระดับพลังงานของสัญญาณน้อยกว่าที่มาตรฐานกำหนดไว้ และในการตรวจจับจุดสิ้นสุดของสัญญาณ Pause Duration นั้นจะใช้การตรวจจับจากถ้ามี Pause Duration ไปเรื่อยๆแล้วเจอสัญญาณที่มีค่าระดับพลังงานของสัญญาณมากกว่า Pause จะถือว่ากำลังเข้าสู่ช่วง Tone Duration ถัดไป

หลังจากที่ตรวจจับได้หนึ่งช่วง Tone Duration และ หนึ่งช่วง Pause Duration แล้วจะส่งไปตรวจสอบว่าเป็นสัญญาณใดโดยใช้ ฟังก์ชั่น DetectSignal

```
bool DetectSignal(CPT_DetectParam *param)
{
        switch(param->state)
        {
          case START:...
          case .....
        }
}
```

ตารางที่ 6-11 Source Code อย่างย่อในการใช้หลักการ State ในการตรวจสอบสัญญาณ

เนื่องจาก CPT นั้นความแตกต่างหลักอยู่ที่ จังหวะของสัญญาณ จะใช้ความแตกต่างนี้เป็นหลักในการจำแนกแต่ละสัญญาณ โดยใช้หลักการ State เข้ามาช่วยเนื่องจากสัญญาณ 1 สัญญาณอาจประกอบด้วยมากกว่า 1 Tone Duration และ 1 Pause Duration ซึ่งมีวิธีการดังนี้

1. State เริ่มแรก คือ "START" State ซึ่งสัญญาณที่มีเพียง 1 Tone Duration กับ 1 Pause Duration (มี Ring Tone, Busy Tone, Congestion Tone, Warning Tone และ Call Waiting) จะสามารถ ตรวจพบ ได้ที่ State นี้เลยโดยไม่มีการเปลี่ยน State สำหรับสัญญาณที่มีแต่ Tone Duration (Ring Tone และ Comfort Tone) นั้นเราจะใช้วิธีดูว่า Tone Duration ของสัญญาณไหนมี Duration มากที่สุด จะกำหนดว่าถ้ามี Tone Duration ที่มี Duration ที่มากกว่า Duration ที่ยาวที่สุดให้นำไป ตรวจสอบความถี่ของสัญญาณไซน์ส่วนประกอบว่า เป็น Ring Tone หรือ Comfort Tone หรือไม่
2. ในกรณีที่สัญญาณมีประกอบด้วยมากกว่า 1 Tone Duration และ 1 Pause Duration นั้นจะต้องมีการเปลี่ยน State อย่างเช่น Special Information ประกอบด้วย 3 Tone Durations และ 3 Pause Durations ขั้นตอนการตรวจจับจะเป็นดังนี้ เริ่มที่ START State พอ ตรวจพบสัญญาณ Tone Duration แรกกับ Pause Duration แรกได้แล้วจะทำการเปลี่ยน State เป็น "SPECIAL1" ต่อไปถ้า ตรวจพบสัญญาณลูกต่อไปจะดู State ว่าถ้าเป็น STATE นี้ต่อไปต้องเป็น "SPECIAL2" แต่ถ้าพบว่าสัญญาณลูกต่อไปไม่ใช่ สัญญาณที่ต้องการ ก็จะทำการ Reject สัญญาณลูกก่อนหน้านี้แล้ว เริ่ม State "START" ใหม่ แต่ถ้าใช่ "SPECIAL2" ก็จะตรวจพบสัญญาณ Tone Duration กับ Pause Duration อันสุดท้ายถ้าใช่ ก็จะ Return Result ว่าเป็นสัญญาณ Special Information และจะเปลี่ยน State ไปที่ "START" อีกครั้ง โดย State ทั้งหมดมีดังนี้

    1. START
    2. SPECIAL1
    3. SPECIAL2

4. PAYPHONE1
5. RECALL_CONFIRM1
6. RECALL_CONFIRM2
7. RECALL_CONFIRM3
8. INTERCEPT1
9. BUSY_VERIFICATION1
10. CALLER_WAITING1
11. CALLER_WAITING2

แต่จะมี CPT อยู่ 2 สัญญาณที่มีปัญหาก็คือ Recall Dial Tone กับ Confirmation จะเห็นว่า จังหวะสัญญาณ เหมือนกันถึง 3 Tone Durations และ 3 Pause Duration จะแตกต่างกันตรง Confirmation ไม่มี Tone Duration ที่ 4 แต่ Recall Dial Tone มี Tone Duration ที่ 4 นั้นก็คือ Ring Tone เลยดังนั้นถ้าเรา Detect ได้ว่า Tone ที่ 4 เป็น Ring Tone ก็จะสรุปว่าเป็น Recall Dial Tone แต่ถ้าไม่ใช่ก็ให้บอกว่าเป็น Confirmation Tone และ Set State เป็น START และส่งสัญญาณ Tone Duration และ Pause Duration ล่าสุดไป ตรวจสอบ ใหม่ว่าเป็นสัญญาณอะไร

มาตรฐานเพิ่มเติมที่ใช้

เนื่องจากสัญญาณ CPT นั้นจะข้อกำหนดที่ไม่เหมือนกันในแต่ละสัญญาณ เพราะฉะนั้นที่จะตรวจสอบความถูกต้องก็จะมี จังหวะของสัญญาณ และความถี่ ซึ่งดูแล้วอาจจะไม่ค่อยดีเท่าที่ควรดังนั้นจึงมีการเพิ่มเติมข้อกำหนดเข้าไปอีกอย่างก็คือ Minimum Interrupt Duration เหมือนใน DTMF และ R2 MF ก็คือถ้ามี Interrupt เข้ามาน้อยกว่าค่าที่กำหนดจะถือเป็นส่วนหนึ่งของ Tone Duration และนำเอาการคิดเกี่ยวกับ ระดับค่าพลังงานของสัญญาณมาใช้ตัดสินช่วง Tone และ Pause ของสัญญาณที่นำมาจาก มาตรฐานของ DTMF ซึ่งน่าจะทำให้การตรวจจับ นั้นมีประสิทธิภาพมากขึ้น

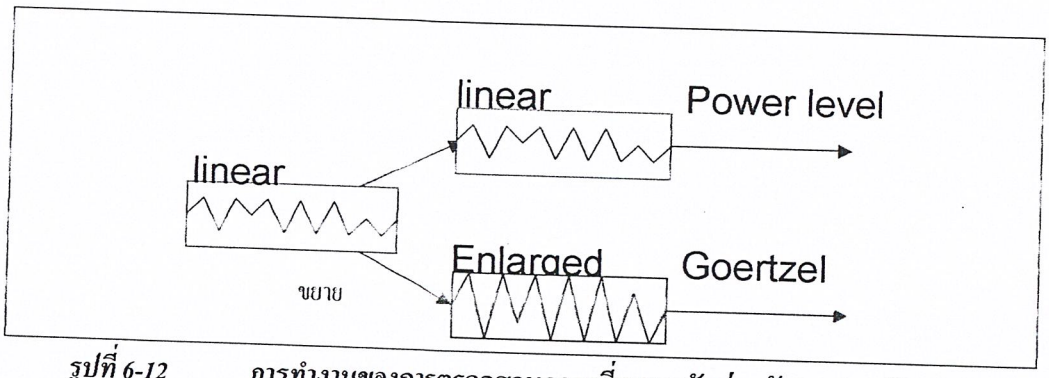
มาตรฐานเพิ่มเติมที่ใช้		**CPT**
Signal Interruption		Max. 10 mS
Power Level	Operation	0 to -25 dBm
	Non-Operation	Less than -55 dBm

ตารางที่ 6-12 มาตรฐานของ CPT ที่นำมาจากบางส่วนของ DTMF

2) การตรวจสอบความถี่

หลังจาก ตรวจจับ จังหวะของสัญญาณ ได้ว่าตรงตามสัญญาณไหนแล้วต่อไปก็จะนำมาคิดว่าใช่ความถี่ของสัญญาณนั้นรึเปล่า โดยใช้ Goertzel Algorithm

จะเห็นว่าการที่เรา ตรวจจับ จังหวะของสัญญาณก่อนนั้นจะทำให้เรารู้ว่าสัญญาณนั้นน่าจะ เป็นสัญญาณอะไร เราแค่ก็ ทำการตรวจสอบ ว่าใช่ความถี่ของสัญญาณนั้นหรือไม่ (ตรวจสอบแค่ไม่กี่ความถี่เท่านั้น) ในการตรวจสอบความถี่นั้นจะตรวจสอบ ระดับค่าพลังงานของสัญญาณก่อน เนื่องจากการตรวจสอบ ระดับค่าพลังงานของสัญญาณ นั้นตรวจสอบง่ายกว่า มีการคำนวณที่ซับซ้อนน้อยกว่าการตรวจสอบความถี่



รูปที่ 6-12 การทำงานของการตรวจสอบความถี่และระดับค่าพลังงาน ของสัญญาณ

เริ่มจากการนำสัญญาณไปหาระดับค่าพลังงานของสัญญาณก่อนเพราะตรวจสอบง่ายกว่า (แค่ตรวจสอบโดยใช้ if..else...) หลังจากนั้นถ้าตรวจสอบแล้วสัญญาณมี ระดับค่าพลังงานของสัญญาณเป็นไปตามมาตราฐาน แล้วก็จะทำการ Normalize สัญญาณให้มีค่าสูงสุดอยู่ที่ ±4095 และนำไปคำนวณโดยใช้ Goertzel Algorithm สาเหตุที่ต้องขยายสัญญาณให้เต็มนั้นก็เพราะ เพื่อความแม่นยำในการตรวจสอบด้วย เพราะถ้าค่าระดับพลังงาน ของสัญญาณไม่เท่ากัน ถึงจะเป็นความถี่เดียวกันค่าที่ได้มาจาก Goertzel Algorithm ก็จะได้ไม่เท่ากัน ซึ่งเมื่อใช้ THRESHOLD ตัดสินนั้นก็จะไม่สามารถกำหนดช่วงความถี่ที่ต้องการได้อย่างแม่นยำ เช่น ตั้ง THRESHOLD ไว้ที่ 1,000,000 มีความถี่หนึ่งมาผ่าน Goertzel Algorithm ได้ค่า 1,100,000 ก็จะผ่านเนื่องจากมีค่ามากกว่า THRESHOLD แต่ถ้าเกิด ค่าระดับพลังงานของสัญญาณลดลงค่า THRESHOLD ที่ได้ก็จะน้อยลง เมื่อผ่าน Goertzel Algorithm แล้วเหลือ 900,000 จะทำให้ความถี่นั้นมีค่าน้อยกว่า THRESHOLD ก็จะไม่ผ่าน ซึ่งเป็นความถี่เดียวกันแค่ต่างกันที่ ค่าระดับพลังงาน ซึ่งเหตุการณ์อย่างนี้ไม่ควรจะเกิดขึ้น

```
int DetectFV2(short* data, int SampCount, const int* DetectedFreq,const int* Threshold,const int
FreqCount, int* Power,const int* Db)
{
        for(a=0;a<SampCount;a++)
        {
                sum2 += data[a]*data[a];

                if(data[a] > max)
                {
                        max = data[a];
                }
        }
        p = calc_dBm0(sum2/SampCount)/100;

        if(p > Db[0] || p < Db[1])
        {
                return 0;
        }
        FullScaleFact = 4095*1000/max;

        for(i=0;i<FreqCount;i++)
        {
                power = GoertzelPower()
                if(power > Threshold[i])
                {
                        return DetectedFreq[i];
                }
        }
        return 0;
}
```

คำนวณ Power ของสัญญาณ และหา Amplitude สูงสุดของ

Check ว่าเป็นไปตาม STD

คำนวณหา Goertzel

ขยายสัญญาณและหาGoertzel Powerและ check ว่ามากกว่า

ตรวจสอบโดย เปรียบเทียบกับค่า

ตารางที่ 6-13 ฟังก์ชันในการตรวจสอบ Power และความถี่ของสัญญาณ

ค่า THRESHOLD ที่ตั้งไว้นั้นได้มาจากการทำการทดลองหาค่าที่เหมาะสมซึ่งดูได้จากผลการทดลองของ CPT ว่าสามารถตรวจสอบสัญญาณแต่ละสัญญาณ อยู่ในความถี่ช่วงไหนบ้าง

```
u0 = u1 = 0;
        co = calcCoeff(DetectedFreq[i]);
        in = t = 0;

        for(x=0;x<SampCount;x++)
        {
                in = ((data[x]*FullScaleFact)/1000)/divx;
                t = u0;
                u0 = in + ((co*u0)/1000)-u1;
                u1 = t;
        }
        power = (u0*u0 + u1*u1 - ((co*u0*u1)/1000));
```

ตารางที่ 6-14 ***Source code ในการคำนวณหาค่าจากสมการ Goertzel Algorithm***

Source code ที่ผ่านมาเป็นวิธีการคำนวณหา Goertzel Algorithm จากสมการที่ได้กล่าวไปแล้วซึ่งจะมาแสดงให้ดูอีกครั้งเพื่อความสะดวกในการเปรียบเทียบกับ Source code

บทที่ 7
การพัฒนา Library ของส่วนสร้างสัญญาณโทรศัพท์

7.1 การพัฒนา Library ของส่วนสร้างสัญญาณ DTMF (DTMFG)

ในการพัฒนา Library ของส่วนสร้างสัญญาณ DTMF ให้เป็นรูปแบบ TMS320 Algorithm Standard นั้น จะต้องเริ่มจากการรวมตัวแปร และค่าคงตัว ไว้เป็น Structure โดยจะต้องมีการระบุส่วนของหน่วยความจำที่จะเก็บ Structure เหล่านั้น จากนั้น ทำการใช้เครื่องมือของตัวโปรแกรมที่ใช้ในการพัฒนา Library บน DSP ในการสร้างเค้าโครง (Template) ของตัว Library จากนั้นทำการแก้ไขการทำงานของฟังก์ชั่นต่างๆที่กำหนดไว้โดยตัว Algorithm Standard

7.1.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

ดังที่ได้อธิบายไปแล้วในส่วนของการสร้างส่วนสร้างสัญญาณ DTMF ว่าได้ทำการเก็บค่าตัวแปรและค่าคงที่เป็น Structure และเมื่อนำมาใช้ในรูปแบบ Algorithm Standard จะต้องกำหนดประเภทและส่วนของหน่วยความจำที่จะใช้ โดยได้ทำการกำหนด memTab ไว้ 3 ชุด ดังนี้

```
/* Request memory for DTMFG object */
    memTab[0].size = sizeof(DTMFG_KASATKA_Obj);
    memTab[0].alignment = (0 * 8) / CHAR_BIT;
    memTab[0].space = IALG_EXTERNAL;
    memTab[0].attrs = IALG_PERSIST;

// mem. tab for history var.
    memTab[1].size = sizeof(DTMFG_MBLK);
    memTab[2].alignment = 0;
    memTab[3].space = IALG_EXTERNAL;
    memTab[4].attrs = IALG_PERSIST;

// mem. tab for field var.
    memTab[1].size = sizeof(DTMFG_MBLK);
    memTab[2].alignment = 0;
    memTab[3].space = IALG_DARAM0;
    memTab[4].attrs = IALG_SCRATCH;
```

ตารางที่ 7-1 Code ที่ใช้ในการกำหนด memTab ของ Library DTMFG

จะเห็นว่า memTab[0] จะระบุถึงหน่วยความจำที่ใช้เก็บ Object ของ Library ว่าเป็นหน่วยความจำภายนอก และเป็นแบบ Persistence โดยมีขนาดเท่ากับตัว Object ของ Library, memTab[1] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนสร้างสัญญาณ DTMF ว่าเป็นหน่วยความจำ

ภายนอก มีขนาดเท่ากับ Structure DTMFG_MBLK และเป็นแบบ Persistence โดยในส่วนนี้ใช้เก็บค่าถาวรของตัวแปร, memTab[2] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนสร้างสัญญาณ DTMF ว่าเป็นหน่วยความจำภายใน แบบ Dual-Accessible มีขนาดเท่ากับ Structure DTMFG_MBLK และเป็นแบบ Scratch โดยในส่วนนี้ใช้เก็บค่าของตัวแปร ในขณะที่ Library ทำงาน

Structure ของ memTab[1] และ memTab[2] นั้นเป็นแบบเดียวกัน ซึ่งสาเหตุที่ต้องจองพื้นที่ 2 ที่นั้น ก็เพื่อให้ memTab[1] ทำการเก็บค่าตัวแปรถาวรของ Object (ในที่นี้จะเรียกว่า History) ซึ่งจะอยู่บน External Memory และเป็นประเภท Persistence คือ Object อื่นๆที่ทำงานอยู่บนระบบเดียวกันจะไม่สามารถเข้ามาเขียนทับบนพื้นที่ส่วนนี้ได้ ส่วน memTab[2] นั้นทำการเก็บค่าตัวแปรขณะทำงานของ Object (ในที่นี้จะเรียกว่า Field) ซึ่งจะอยู่บน Internal Memory ที่เป็นแบบ Dual-Accessible และเป็นประเภท Scratch ซึ่งคือพื้นที่ประเภทที่ Object อื่นๆที่ทำงานอยู่บนระบบเดียวกัน สามารถมาใช้พื้นที่ในส่วนนี้ซ้ำได้ หากว่า Object ใดๆนั้นทำงานอยู่ เนื่องจาก Internal Memory นั้นมีความเร็วสูง และมีขนาดที่น้อยกว่า External Memory จึงอาจไม่สามารถใช้เป็น Persistence Memory สำหรับทุก Object ในระบบได้ ดังนั้นจึงต้องใช้เป็น Scratch Memory เพื่อให้ทุก Object สามารถใช้งานได้ขณะที่ Object นั้นกำลังทำงานอยู่ (Execution)

ก่อนทำงานฟังก์ชั่นใดๆของ Library จะต้องทำการคัดลอกข้อมูลที่อยู่ในส่วน History ไปอยู่ในส่วน Field ก่อน จากนั้นจึงทำงานโดยอ้างตัวแปรจากส่วน Field จากนั้นเมื่อทำงานเสร็จแล้ว ก็จะทำการคัดลอกข้อมูลในส่วน Field กลับไปยังส่วน History เพื่อรักษาค่าตัวแปรเอาไว้

7.1.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

จะเป็นฟังก์ชั่นในการกำหนด memTab ดังที่ได้อธิบายไปข้างต้นโดยเมื่อมีการสร้าง Object ของ Library, ฟังก์ชั่นนี้จะถูกเรียกโดย ALGRF_create เพื่อดูความต้องการของการใช้พื้นที่หน่วยความจำ เพื่อจะได้จองเนื้อที่หน่วยความจำให้เป็นไปตามความต้องการของ Library

7.1.3 กำหนดการทำงานของฟังก์ชั่น algInit

จะเป็นฟังก์ชั่นในการตั้งค่าตัวแปร Pointer ให้ชี้ไปยังตำแหน่งในหน่วยความจำที่ได้รับการจองพื้นที่ให้ (ที่ทำการขอพื้นที่โดยใช้ memTab) จากนั้นก็จะ Initialize ค่าตัวแปรต่างๆ ของ Object ที่ทำการสร้างขึ้น ซึ่งฟังก์ชั่นนี้จะถูกเรียกโดย ALGRF_create ภายหลังจากที่ได้ทำการจองพื้นที่ในหน่วยความจำให้กับ Object เป็นที่เรียบร้อยแล้ว

7.1.4 กำหนดการทำงานของฟังก์ชั่น algActivate

เป็นฟังก์ชั่นที่จะต้องถูกเรียกก่อนการทำงานของฟังก์ชั่นของ Library โดยฟังก์ชั่นนี้จะทำการคัดลอกข้อมูลจากพื้นที่ส่วน Persistence Memory ไปยัง Scratch memory

7.1.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate

เป็นฟังก์ชั่นที่จะต้องถูกเรียกหลังการทำงานของฟังก์ชั่นของ Library โดยฟังก์ชั่นนี้จะทำการคัดลอกข้อมูลจากพื้นที่ส่วน Scratch Memory ไปยัง Persistence Memory

7.1.6 กำหนดการทำงานของฟังก์ชั่น algMoved

เป็นฟังก์ชั่นที่จะทำการปรับค่าของตัวแปร Pointer ที่ชี้ไปยังตำแหน่งในหน่วยความจำที่ได้รับการจองพื้นที่ให้ (ที่ทำการขอพื้นที่โดยใช้ memTab) ให้ชี้ไปยังตำหน่งใหม่ โดยฟังก์ชั่นนี้จะถูกเรียกเมื่อระบบทำการย้าย Object ของ Library ไปยังตำแหน่งอื่น

7.1.7 กำหนดการทำงานของฟังก์ชั่น algFree

ฟังก์ชั่นนี้จะถูกเรียกโดย ALGRF_delete เพื่อดูข้อมูลการใช้พื้นที่หน่วยความจำ เพื่อจะได้ทำการ Free เนื้อที่หน่วยความจำให้เป็นไปตามข้อมูลการใช้พื้นที่หน่วยความจำของ Library

7.1.8 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_apply

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการสร้างสัญญาณ ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ DTMF

7.1.9 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_enqueue

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการเพิ่มหมายเลขสัญญาณ DTMF เข้าไปใน Generation Queue ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ DTMF

7.1.10 กำหนดการทำงานของฟังก์ชั่น DTMFG_KASATKA_setEnable

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการตั้งค่า Enable Flag ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ DTMF

7.2 การพัฒนา Library ของส่วนสร้างสัญญาณ R2 MF (R2MFG)

ในการพัฒนา Library ของส่วนสร้างสัญญาณ R2 MF ให้เป็นรูปแบบ TMS320 Algorithm Standard นั้น จะต้องเริ่มจากการรวมตัวแปร และค่าคงตัว ไว้เป็น Structure โดยจะต้องมีการระบุส่วนของหน่วยความจำที่จะเก็บ Structure เหล่านั้น จากนั้น ทำการใช้เครื่องมือของตัวโปรแกรมที่ใช้ในการพัฒนา Library บน DSP ในการสร้างเค้าโครง (Template) ของตัว Library จากนั้นทำการแก้ไขการทำงานของฟังก์ชั่นต่างๆที่กำหนดไว้โดยตัว Algorithm Standard

7.2.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

ดังที่ได้อธิบายไปแล้วในส่วนของการสร้างส่วนสร้างสัญญาณ R2 MF ว่าได้ทำการเก็บค่าตัวแปรและค่าคงที่เป็น Structure และเมื่อนำมาใช้ในรูปแบบ Algorithm Standard จะต้องกำหนดประเภทและส่วนของหน่วยความจำที่จะใช้ โดยได้ทำการกำหนด memTab ไว้ 3 ชุด ดังนี้

```
/* Request memory for R2MFG object */
   memTab[0].size = sizeof(R2MFG_KASATKA_Obj);
   memTab[0].alignment = (0 * 8) / CHAR_BIT;
   memTab[0].space = IALG_EXTERNAL;
   memTab[0].attrs = IALG_PERSIST;

// mem tab for history var.
   memTab[HISTORY_TABN].size = sizeof(R2MFG_MBLK);
   memTab[HISTORY_TABN].alignment = 0;
   memTab[HISTORY_TABN].space = IALG_EXTERNAL;
   memTab[HISTORY_TABN].attrs = IALG_PERSIST;

// mem tab for field var.

   memTab[FIELD_TABN].size = sizeof(R2MFG_MBLK);
   memTab[FIELD_TABN].alignment = 0;
   memTab[FIELD_TABN].space = IALG_DARAM0;
   memTab[FIELD_TABN].attrs = IALG_SCRATCH;
```

ตารางที่ 7-2 Code ที่ใช้ในการกำหนด memTab ของ Library R2MFG

จะเห็นว่า memTab[0] จะระบุถึงหน่วยความจำที่ใช้เก็บ Object ของ Library ว่าเป็นหน่วยความจำภายนอก และเป็นแบบ Persistence โดยมีขนาดเท่ากับตัว Object ของ Library, memTab[1] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนสร้างสัญญาณ R2 MF ว่าเป็นหน่วยความจำภายนอก มีขนาดเท่ากับ Structure R2MFG_MBLK และเป็นแบบ Persistence โดยในส่วนนี้ใช้เก็บค่าถาวรของตัวแปร, memTab[2] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนสร้างสัญญาณ R2 MF ว่าเป็นหน่วยความจำภายใน แบบ Dual-Accessible มีขนาดเท่ากับ Structure R2MFG_MBLK และเป็นแบบ Scratch โดยในส่วนนี้ใช้เก็บค่าของตัวแปร ในขณะที่ Library ทำงาน

7.2.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.3 กำหนดการทำงานของฟังก์ชั่น algInit

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.4 กำหนดการทำงานของฟังก์ชั่น algActivate

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.6 กำหนดการทำงานของฟังก์ชั่น algMoved

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.7 กำหนดการทำงานของฟังก์ชั่น algFree

เช่นเดียวกับการกำหนดใน Library DTMFG

7.2.8 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_apply

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการสร้างสัญญาณ ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ R2 MF

7.2.9 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_setDigit

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการระบุสัญญาณ R2 MF ที่ต้องการสร้างสัญญาณ ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ R2 MF

7.2.10 กำหนดการทำงานของฟังก์ชั่น R2MFG_KASATKA_setState

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการตั้งค่า State การทำงานของ Library ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนสร้างสัญญาณ R2 MF

7.3 การพัฒนา Library ของส่วนสร้างสัญญาณ CPT (CPTGEN)

หลังจากที่ใช้ Tool ในการสร้าง Template ของ Standard Algorithm ใช้ชื่อ Algorithm ว่า "CPTGEN" ใช้ชื่อ Vendor ว่า "KMITL" (วิธีการสร้าง Template ดูได้จากคู่มือ Programmer) แล้วก็จะนำมาแก้ไขโดยมีรายละเอียดดังนี้

รายละเอียดในไฟล์ CPTGEN_KMITL_ialgvt.c

```
#define IALGFXNS \
    &CPTGEN_KMITL_IALG,     /* module ID                                */ \
    NULL,                   /* activate   (NULL => not suported)        */ \
    CPTGEN_KMITL_alloc,     /* algAlloc                                 */ \
    CPTGEN_KMITL_control,   /* control    (NULL => not suported)        */ \
    NULL,                   /* deactivate (NULL => not suported)        */ \
```

ตารางที่ 7-3 ฟังก์ชั่น ที่จะถูก Implement ใน vector table

ถ้าอันไหนเป็น NULL นั้นหมายความว่าไม่ได้จัดทำไว้ใน Algorithm นี้ซึ่งใน CPTGEN Algorithm นั้นไม่ Implement 3 ฟังก์ชั่น ในการ Activate, Deactivate, และ memAlloc เพราะว่าจะใช้ของ ALG ที่มีจัดไว้ให้อยู่แล้ว ส่วน field ที่มีชื่อนั้นหมายความว่า

- CPTGEN_KMITL_alloc ใช้แทน algAlloc ใน IALG APIs
- CPTGEN_KMITL_control ใช้แทน algControl ใน IALG APIs
- CPTGEN_KMITL_free ใช้แทน algFree ใน IALG APIs
- CPTGEN_KMITL_initObj ใช้แทน algInit ใน IALG APIs
- CPTGEN_KMITL_moved ใช้แทน algMoved ใน IALG APIs

การใช้งานแต่ละฟังก์ชั่น ให้ดูในส่วน ทฤษฎี

7.3.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

ดูรายละเอียดในไฟล์ icptgen.h

```
typedef struct CPT_GenParam
{
        int *data;
        unsigned char state;                    //1 = tone, 0 = pause
        unsigned int  SampRate;
        int code;
        int amplitude;                                  //1.0 scale to 100
        unsigned int  StartSample;              //Start Sample of this call
        unsigned int  step1;
        unsigned int  step2;
        unsigned char full;                             //1 = full data in
data[GEN_LENGTH]
        unsigned char completeCycle;            //1 = completed cycle
        unsigned int  count;
        unsigned int  frameSize;                //number of sample per 1 call
        const int *SinTable;
} CPT_GenParam;
```

ตารางที่ 7-4 Structure ที่ไว้บันทึก State ใน Generation Algorithm

เพิ่ม Structure ลงในไฟล์นี้ซึ่งใช้ในการบันทึก State การทำงานในแต่ละครั้งที่มีการ Call ฟังก์ชั่น apply และต้องการให้ Instance ของ Structure ไปอยู่ในส่วน Persistence Memory ซึ่งจะเป็นส่วนที่ Algorithm อื่นไม่สามารถเข้ามาเขียนได้ จึงต้องใช้การกำหนดไว้ใน memTab ซึ่งเป็นตัวแปรที่จะบ่งบอกถึงสถานะของตัวแปรว่าจะเก็บอย่างไรและอยากจะเก็บไว้ที่ไหน ซึ่งจะกำหนดไว้ใน ฟังก์ชั่น CPTGEN_KMITL_alloc ซึ่งอยู่ในไฟล์ cptgen_kmitl_ialg.cไว้ดังนี้

7.3.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

แก้ไขในไฟล์ cptgen_kmitl_ialg.c

```
Int CPTGEN_KMITL_alloc(const IALG_Params *CPTGENParams, IALG_Fxns **fxns,
IALG_MemRec memTab[])
{
    const ICPTGEN_Params *params = (Void *)CPTGENParams;

    /* TODO: implement algAlloc */

    if (params == NULL) {
        params = &ICPTGEN_PARAMS;  /* set default parameters */
    }
    /* Request memory for CPTGEN object */
    memTab[0].size = sizeof(CPTGEN_KMITL_Obj);
    memTab[0].alignment = (0 * 8) / CHAR_BIT;
    memTab[0].space = IALG_DARAM0;
    memTab[0].attrs = IALG_PERSIST;

    memTab[1].size = sizeof(CPT_GenParam);
    memTab[1].alignment = 0;
    memTab[1].space = IALG_DARAM0;
    memTab[1].attrs = IALG_PERSIST;

    return (MTAB_NRECS);
}
```

ตารางที่ 7-5 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algAlloc

ในส่วนของ memTab[0] นั้นเป็นส่วนที่ทุก อัลกอรึทึม ต้องมีซึ่ง Code ที่ถูก Generate ,จะมีอยู่แล้ว ซึ่งจะมีค่าต่างตอนที่เรา Create อัลกอรึทึม เก็บอยู่อย่างเช่น ขนาดของเฟรมที่จะกระทำในการ Call ฟังก์ชั่น หนึ่งครั้ง ในส่วนถัดไปนั้นเราต้องทำการเพิ่ม memTab[1] เข้าไปเองใน field “space” นั้นก็คืออยากได้พื้นที่ส่วนไหน (Internal Ram, External Ram หรืออื่นๆ รายละเอียดให้ดูในคู่มือที่กล่าวไปข้างต้น) และ filed “attrs” บอกว่าเป็น หน่วยความจำ ประเภทใด (IALG_SCRATCH หรือ IALG_PERSIST) ในที่นี้กำหนดให้เป็น IALG_PERSIST เพราะว่าต้องการใช้เก็บ State จึงจำเป็นต้องเก็บในพื้นที่ถาวรไม่มีใครมาแก้ไขได้ และสุดท้ายสิ่งที่ Return ออกมาจาก ฟังก์ชั่น ก็คือจำนวน memTab ที่ต้องการสร้างซึ่งในที่นี้

ต้องการสร้าง 2 memTab จึงต้องแก้ไข MTAB_NRECS ให้มีค่าเป็น 2 ซึ่งค่านี้จะถูกประกาศอยู่ในไฟล์เดียวกันนี้เอง

ในไฟล์นี้ก็ยังมีจุดที่ต้องแก้ไขอีกหลายจุดดังนี้

```
typedef struct CPTGEN_KMITL_Obj {
   IALG_Obj       alg;   /* MUST be first field of all CPTGEN objs */
   XDAS_UInt32    framesizeOut0;
   const XDAS_Int32 *sinTable;

   CPT_GenParam *param;
} CPTGEN_KMITL_Obj;
```

ตารางที่ 7-6 Structure ที่ใช้เป็นตัวแปร Global เพื่อให้ อัลกอริทึม ทำงานได้หลาย Channel

สร้าง Pointer ไว้ชี้ CPT_GenParam Instance ไว้ใน Structure เพราะว่า Structure นี้จะเป็นตัวแปร Global ของแต่ละ Algorithm Instance ที่จะมีไม่เหมือนกัน

7.3.3 กำหนดการทำงานของฟังก์ชั่น algInit

```
Int CPTGEN_KMITL_initObj(IALG_Handle handle,
                const IALG_MemRec memTab[], IALG_Handle p, const IALG_Params
*CPTGENParams)
{
   CPTGEN_KMITL_Obj *CPTGEN = (Void *)handle;
   const ICPTGEN_Params *params = (Void *)CPTGENParams;
   if(params == NULL){
      params = &ICPTGEN_PARAMS; /* set default parameters */
   }
   CPTGEN->framesizeOut0 = params->framesizeOut0;
   CPTGEN->param = memTab[1].base;
   CPTGEN->param->SinTable = params->sinTable;
        CPTGEN->param->SampRate = 8000;
        CPTGEN->param->count = 0;
        CPTGEN->param->completeCycle = 0;
        CPTGEN->param->StartSample = 0;
        CPTGEN->param->step1 = 0;
        CPTGEN->param->step2 = 0;
        CPTGEN->param->state = 1;
        CPTGEN->param->full = 0;
        CPTGEN->param->frameSize = CPTGEN->framesizeOut0;
return (IALG_EOK);
```

ตารางที่ 7-7 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algInit

ในส่วน CPTGEN_KMITL_initObj นั้นก็คือ algInit ของ IALG APIs นั้นเองซึ่งจะถูกเรียกใช้ตอนที่อัลกอริทึม ถูกสร้างและถูกเรียกใช้เพียงครั้งเดียว เพื่อทำการกำหนดค่าเริ่มต้นต่างๆ และยังกำหนดให้ตัวแปร Global ที่ถูก Pass เป็น Parameter นั้น (บรรทัดแรกของ Hilight สีเหลือง) ที่มี CPT_GenParam เป็น Structure ไว้เก็บ State เป็น Pointer ชี้ไปที่ memTab[1] ซึ่งเราได้ทำการจองพื้นที่ไว้สำหรับตัวแปรที่ใช้เก็บ State โดยเฉพาะ

7.3.4 กำหนดการทำงานของฟังก์ชั่น algMoved

```
Void CPTGEN_KMITL_moved(IALG_Handle handle,
                const IALG_MemRec memTab[], IALG_Handle p, const IALG_Params
*CPTGENParams)
{
   CPTGEN_KMITL_Obj *CPTGEN = (Void *)handle;
   const ICPTGEN_Params *params = (Void *)CPTGENParams;

   CPTGEN->param = memTab[1].base;
}
```

ตารางที่ 7-8 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algMoved

ส่วน ฟังก์ชั่น การ Moved นั้น Integrator เป็นคนเคลื่อนย้ายพื้นที่จัดเก็บและ Call ฟังก์ชั่น นี้เพื่อส่งค่า Pointer ของตำแหน่งใหม่ถูกเก็บตัวแปรนี้มาให้ อัลกอริทึม

7.3.5 กำหนดการทำงานของฟังก์ชั่น algFree

```
Int CPTGEN_KMITL_free(IALG_Handle handle, IALG_MemRec memTab[])
{
   Int n;
   CPTGEN_KMITL_Obj *CPTGEN = (Void *)handle;

   n = CPTGEN_KMITL_alloc(NULL, NULL, memTab);

   memTab[0].base = handle;
   memTab[1].base = CPTGEN->param;

   return (n);
}
```

ตารางที่ 7-9 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algFree

ใน ฟังก์ชั่น นี้เป็นการ Return Pointer ของตัวแปรที่ถูก alloc ไว้โดย CPTGEN_KMITL_alloc ซึ่ง memTab[0] จะมีอยู่แล้วก็เพิ่ม memTab[1] ลงไปเพื่อบอกตำแหน่งของ CPT_GenParam ที่เป็น Instance ก็คือ Param ที่เป็น Member ของ Structure

7.3.6 กำหนดการทำงานของฟังก์ชั่น apply

และ Function CPTGEN_KMITL_appy นั้นเป็น ฟังก์ชั่น ที่เรา Implement โดยการนำ อัลกอริทึม ที่เขียนไว้แล้วมาใส่ลงไปโดยมีการปรับเปลี่ยนตัวแปร โดยใช้ตัวแปรที่ใช้บันทึก State แทนตัวแปร Global ที่เคยใช้ ก็คือนำ CPTGEN->param ที่ชี้ไปยัง memTab[0] มาใช้นั้นเอง

และยังมีไฟล์ 2 ไฟล์ที่เพิ่มเติมมาซึ่งก็คือ cptgen_kmitl_priv.h และ cptgen_kmitl_priv.c ซึ่งเป็นไฟล์เกี่ยวกับ ฟังก์ชั่น ภายในที่ไม่เกี่ยวข้องกับผู้เรียกใช้ (Integrator)

บทที่ 8
การพัฒนา Library ของส่วนตรวจจับสัญญาณโทรศัพท์

8.1 การพัฒนา Library ของส่วนตรวจจับสัญญาณ DTMF (DTMFD)

ในการพัฒนา Library ของส่วนตรวจจับสัญญาณ DTMF ให้เป็นรูปแบบ TMS320 Algorithm Standard นั้น จะต้องเริ่มจากการรวมตัวแปร และค่าคงตัว ไว้เป็น Structure โดยจะต้องมีการระบุส่วนของหน่วยความจำที่จะเก็บ Structure เหล่านั้น จากนั้น ทำการใช้เครื่องมือของตัวโปรแกรมที่ใช้ในการพัฒนา Library บน DSP ในการสร้างเค้าโครง (Template) ของตัว Library จากนั้นทำการแก้ไขการทำงานของฟังก์ชั่นต่างๆที่กำหนดไว้โดยตัว Algorithm Standard

8.1.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

ดังที่ได้อธิบายไปแล้วในส่วนของการสร้างส่วนตรวจจับสัญญาณ DTMF ว่าได้ทำการเก็บค่าตัวแปร และค่าคงที่เป็น Structure และเมื่อนำมาใช้ในรูปแบบ Algorithm Standard จะต้องกำหนดประเภทและส่วนของหน่วยความจำที่จะใช้ โดยได้ทำการกำหนด memTab ไว้ 3 ชุด ดังนี้

```
/* Request memory for DTMFD object */
   memTab[0].size = sizeof(DTMFD_KASATKA_Obj);
   memTab[0].alignment = (0 * 8) / CHAR_BIT;
   memTab[0].space = IALG_EXTERNAL;
   memTab[0].attrs = IALG_PERSIST;

// mem. tab for history var.,
   memTab[HISTORY_TABN].size = sizeof(DTMFD_MBLK);
   memTab[HISTORY_TABN].alignment = 0;
   memTab[HISTORY_TABN].space = IALG_EXTERNAL;
   memTab[HISTORY_TABN].attrs = IALG_PERSIST;

// mem. tab for field var.,
   memTab[FIELD_TABN].size = sizeof(DTMFD_MBLK);
   memTab[FIELD_TABN].alignment = 0;
   memTab[FIELD_TABN].space = IALG_DARAM0;
   memTab[FIELD_TABN].attrs = IALG_SCRATCH;
```

ตารางที่ 8-1 Code ที่ใช้ในการกำหนด memTab ของ Library DTMFD

จะเห็นว่า memTab[0] จะระบุถึงหน่วยความจำที่ใช้เก็บ Object ของ Library ว่าเป็นหน่วยความจำภายนอก และเป็นแบบ Persistence โดยมีขนาดเท่ากับตัว Object ของ Library, memTab[1] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนตรวจจับสัญญาณ DTMF ว่าเป็นหน่วยความจำ

ภายนอก มีขนาดเท่ากับ Structure DTMFD_MBLK และเป็นแบบ Persistence โดยในส่วนนี้ใช้เก็บค่าถาวรของตัวแปร, memTab[2] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนตรวจจับสัญญาณ DTMF ว่าเป็นหน่วยความจำภายใน แบบ Dual-Accessible มีขนาดเท่ากับ Structure DTMFD_MBLK และเป็นแบบ Scratch โดยในส่วนนี้ใช้เก็บค่าของตัวแปร ในขณะที่ Library ทำงาน

8.1.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.3 กำหนดการทำงานของฟังก์ชั่น algInit

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.4 กำหนดการทำงานของฟังก์ชั่น algActivate

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.6 กำหนดการทำงานของฟังก์ชั่น algMoved

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.7 กำหนดการทำงานของฟังก์ชั่น algFree

เช่นเดียวกับการกำหนดใน Library DTMFG

8.1.8 กำหนดการทำงานของฟังก์ชั่น DTMFD_KASATKA_apply

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการตรวจจับสัญญาณ ดังที่ได้อธิบายไปในส่วนของการสร้าง ส่วนตรวจจับสัญญาณ DTMF

8.2 การพัฒนา Library ของส่วนตรวจจับสัญญาณ R2 MF (R2MFD)

ในการพัฒนา Library ของส่วนตรวจจับสัญญาณ R2 MF ให้เป็นรูปแบบ TMS320 Algorithm Standard นั้น จะต้องเริ่มจากการรวมตัวแปร และค่าคงตัว ไว้เป็น Structure โดยจะต้องมีการระบุส่วนของหน่วยความจำที่จะเก็บ Structure เหล่านั้น จากนั้น ทำการใช้เครื่องมือของตัวโปรแกรมที่ใช้ในการพัฒนา Library บน DSP ในการสร้างเค้าโครง (Template) ของตัว Library จากนั้นทำการแก้ไขการทำงานของฟังก์ชั่นต่างๆที่กำหนดไว้โดยตัว Algorithm Standard

8.2.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

ดังที่ได้อธิบายไปแล้วในส่วนของการสร้างส่วนตรวจจับสัญญาณ R2 MF ว่าได้ทำการเก็บค่าตัวแปร และค่าคงที่เป็น Structure และเมื่อนำมาใช้ในรูปแบบ Algorithm Standard จะต้องกำหนดประเภทและส่วนของหน่วยความจำที่จะใช้ โดยได้ทำการกำหนด memTab ไว้ 3 ชุด ดังนี้

```
/* Request memory for R2MFD object */
   memTab[0].size = sizeof(R2MFD_KASATKA_Obj);
   memTab[0].alignment = (0 * 8) / CHAR_BIT;
   memTab[0].space = IALG_EXTERNAL;
   memTab[0].attrs = IALG_PERSIST;

// mem tab for history var.
   memTab[HISTORY_TABN].size = sizeof(R2MFD_MBLK);
   memTab[HISTORY_TABN].alignment = 0;
   memTab[HISTORY_TABN].space = IALG_EXTERNAL;
   memTab[HISTORY_TABN].attrs = IALG_PERSIST;

// mem tab for field var.
   memTab[FIELD_TABN].size = sizeof(R2MFD_MBLK);
   memTab[FIELD_TABN].alignment = 0;
   memTab[FIELD_TABN].space = IALG_DARAM0;
   memTab[FIELD_TABN].attrs = IALG_SCRATCH;
```

ตารางที่ 8-2 Code ที่ใช้ในการกำหนด memTab ของ Library R2MFD

จะเห็นว่า memTab[0] จะระบุถึงหน่วยความจำที่ใช้เก็บ Object ของ Library ว่าเป็นหน่วยความจำภายนอก และเป็นแบบ Persistence โดยมีขนาดเท่ากับตัว Object ของ Library, memTab[1] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนตรวจจับสัญญาณ R2 MF ว่าเป็นหน่วยความจำภายนอก มีขนาดเท่ากับ Structure R2MFD_MBLK และเป็นแบบ Persistence โดยในส่วนนี้ใช้เก็บค่าถาวรของตัวแปร, memTab[2] จะระบุถึงหน่วยความจำที่ใช้เก็บ Structure ตัวแปรของส่วนตรวจจับสัญญาณ R2 MF ว่าเป็นหน่วยความจำภายใน แบบ Dual-Accessible มีขนาดเท่ากับ Structure R2MFD_MBLK และเป็นแบบ Scratch โดยในส่วนนี้ใช้เก็บค่าของตัวแปร ในขณะที่ Library ทำงาน

8.2.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.3 กำหนดการทำงานของฟังก์ชั่น algInit

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.4 กำหนดการทำงานของฟังก์ชั่น algActivate

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.5 กำหนดการทำงานของฟังก์ชั่น algDeactivate

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.6 กำหนดการทำงานของฟังก์ชั่น algMoved

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.7 กำหนดการทำงานของฟังก์ชั่น algFree

เช่นเดียวกับการกำหนดใน Library DTMFG

8.2.8 กำหนดการทำงานของฟังก์ชั่น R2MFD_KASATKA_apply

ฟังก์ชั่นนี้จะเป็นฟังก์ชั่นที่ผู้ใช้ทำการเรียก เพื่อให้ทำการตรวจจับสัญญาณ ดังที่ได้อธิบายไปใน *ส่วนของการสร้าง ส่วนตรวจจับสัญญาณ R2 MF*

8.3 การพัฒนา Library ของส่วนตรวจจับสัญญาณ CPT (CPTDETECT)

ในการทำให้ Detection Algorithm เป็นไปตาม Standard นั้นก็จะมีวิธีคล้ายๆกับ การทำ Generation Algorithm ให้เป็น Standard ซึ่งเรื่องที่กล่าวมาแล้วในหัวข้อ Generation Algorithm จะไม่กล่าวซ้ำในหัวข้อนี้ และชื่อ Algorithm ของการ Detection คือ "CPTDETECT" และชื่อ Vendor ก็เหมือนเดิมคือ "KMITL"

ดูในไฟล์ cptdetect_kmitl_ialgvt.c

```
#define IALGFXNS \
  &CPTDETECT_KMITL_IALG,      /* module ID                              */ \
  NULL,                       /* activate  (NULL => not suported)       */ \
  CPTDETECT_KMITL_alloc,      /* algAlloc                               */ \
  CPTDETECT_KMITL_control,    /* control   (NULL => not suported)       */ \
  NULL,                       /* deactivate (NULL => not suported)      */ \
```

ตารางที่ 8-3 ฟังก์ชั่น ที่ถูก Implement ใน Vector Table

เหมือนการ Generation Algorithm มี 3 Function ที่ไม่ได้ถูก Implement

8.3.1 กำหนดพื้นที่หน่วยความจำสำหรับ Structure

```
typedef struct CPT_DetectParam
{
        int *data;
        unsigned int CntS;
        unsigned int CntZ;
        unsigned int InterruptCntS;
        unsigned int state;
        short PrevData[200];
        unsigned short isDetected;
        unsigned short isRejected;
        unsigned int result;
        int power;
        int CountPrev;
        /////use for Confirmation Tone and reject signal in state is not in START
state////
        unsigned short isConfirmation;        //for confirmation only
        unsigned short isSecondDetect;
        unsigned short isSecondReject;
        int SecondPower;
        unsigned short SecondResult;

        unsigned short StartKeep;
        unsigned short FirstStart;
        ////////////////////////////////////////
} CPT_DetectParam;
```

ตารางที่ 8-4 Structure ที่ใช้บันทึก state ของ CPT Detection Algorithm

8.3.2 กำหนดการทำงานของฟังก์ชั่น algAlloc

```
Int CPTDETECT_KMITL_alloc(const IALG_Params *CPTDETECTParams, IALG_Fxns
**fxns, IALG_MemRec memTab[])
{
   const ICPTDETECT_Params *params = (Void *)CPTDETECTParams;

   /* TODO: implement algAlloc */

   if (params == NULL) {
      params = &ICPTDETECT_PARAMS;  /* set default parameters */
   }

   /* Request memory for CPTDETECT object */
   memTab[0].size = sizeof(CPTDETECT_KMITL_Obj);
   memTab[0].alignment = (0 * 8) / CHAR_BIT;
   memTab[0].space = IALG_DARAM0;
   memTab[0].attrs = IALG_PERSIST;
```

```
    // NOTE: Some algorithms may be better suited to use an equation to define
memtab
    //      size(s). User may want to modify the size field as required by his algorithm.

    /* Request memory for param array */
    memTab[PARAM].size = sizeof(CPT_DetectParam);
    memTab[PARAM].alignment = 0;
    memTab[PARAM].space = IALG_DARAM0;
    memTab[PARAM].attrs = IALG_PERSIST;

    return (MTAB_NRECS);
}
```

ตารางที่ 8-5 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algAlloc

คล้ายกับ CPT Generation Algorithm โดยในที่นี้ PARAM มีค่าเท่ากับ 1 ก็คือจองเนื้อที่ไว้เก็บตัวแปรที่ใช้บันทึก state ซึ่งก็คือ Instance ของ Structure CPT_DetectParam และไปกำหนดค่า MTAB_NRECS ให้มีค่าเป็น 2 ด้วย

```
typedef struct CPTDETECT_KMITL_Obj {
    IALG_Obj        alg;   /* MUST be first field of all CPTDETECT objs */
    XDAS_UInt32     framesizeIn0;
    CPT_DetectParam *param;

    /* TODO: add custom fields here */
} CPTDETECT_KMITL_Obj;
```

ตารางที่ 8-6 Structure ที่ใช้เป็นตัวแปร Global เพื่อให้ อัลกอริทึม ทำงานได้หลาย Channel

เพิ่ม Pointer ที่ใช้เพื่อชี้ไปยัง Structure Instance ที่จะใช้บันทึก State

8.3.3 กำหนดการทำงานของฟังก์ชั่น algInit

```
Int CPTDETECT_KMITL_initObj(IALG_Handle handle,
                const IALG_MemRec memTab[], IALG_Handle p, const IALG_Params
*CPTDETECTParams)
{
    CPTDETECT_KMITL_Obj *CPTDETECT = (Void *)handle;
    const ICPTDETECT_Params *params = (Void *)CPTDETECTParams;

    if(params == NULL){
        params = &ICPTDETECT_PARAMS; /* set default parameters */
    }
```

```
        CPTDETECT->framesizeIn0 = params->framesizeIn0;
        CPTDETECT->param = memTab[PARAM].base;

         CPTDETECT->param->CntS = 0;
        CPTDETECT->param->CntZ = 0;
        CPTDETECT->param->InterruptCntS = 0;
        CPTDETECT->param->isDetected = 0;
        CPTDETECT->param->isRejected = 0;
        CPTDETECT->param->power = 0;
        CPTDETECT->param->result = 99;
        CPTDETECT->param->CountPrev = 0;
        CPTDETECT->param->state = START;
        CPTDETECT->param->isConfirmation = 0;
        CPTDETECT->param->isSecondDetect = 0;
        CPTDETECT->param->isSecondReject = 0;
        CPTDETECT->param->SecondPower = 0;
        CPTDETECT->param->SecondResult =  99;
        CPTDETECT->param->StartKeep = 1;
        CPTDETECT->param->FirstStart = 1;

    return (IALG_EOK);
}
```

ตารางที่ 8-7 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algInit

คล้ายกับ CPT Generation Algorithm ก็คือ มีการกำหนดค่าเริ่มต้นของ Algorithm และกำหนดให้ CPT_DetectParam pointer ชี้ไปยัง MemTab ที่จัดเตรียมไว้เพื่อใช้บันทึก State ในการทำงานแบบ Real Time

8.3.4 กำหนดการทำงานของฟังก์ชั่น algMoved

```
Void CPTDETECT_KMITL_moved(IALG_Handle handle,
                const IALG_MemRec memTab[], IALG_Handle p, const IALG_Params
*CPTDETECTParams)
{
    CPTDETECT_KMITL_Obj *CPTDETECT = (Void *)handle;
    const ICPTDETECT_Params *params = (Void *)CPTDETECTParams;

    CPTDETECT->param = memTab[PARAM].base;
}
```

ตารางที่ 8-8 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algMoved

8.3.5 กำหนดการทำงานของฟังก์ชั่น algFree

```
Int CPTDETECT_KMITL_free(IALG_Handle handle, IALG_MemRec memTab[])
{
    Int n;
    CPTDETECT_KMITL_Obj *CPTDETECT = (Void *)handle;

    n = CPTDETECT_KMITL_alloc(NULL, NULL, memTab);

    memTab[0].base = handle;
    memTab[PARAM].base = CPTDETECT->param;

    return (n);
}
```

ตารางที่ 8-9 ฟังก์ชั่น ที่ถูกเรียกใช้แทน algFree

8.3.6 กำหนดการทำงานของฟังก์ชั่น apply

และสุดท้ายก็คือ Function CPTDETECT_KMITL_apply ซึ่งก็จะนำ Detection Algorithm ที่ได้พัฒนาไว้แล้วมาดัดแปลงใส่ลงไปโดยใช้ตัวแปรในการบันทุก state คือ pointer ที่ชี้ไปยัง MemTab[1] นั้นก็คือ CPTDETECT->param นั้นเอง

บทที่ 9
การทดลอง เพื่อประเมินค่า THRESHOLD ที่จะนำมาใช้

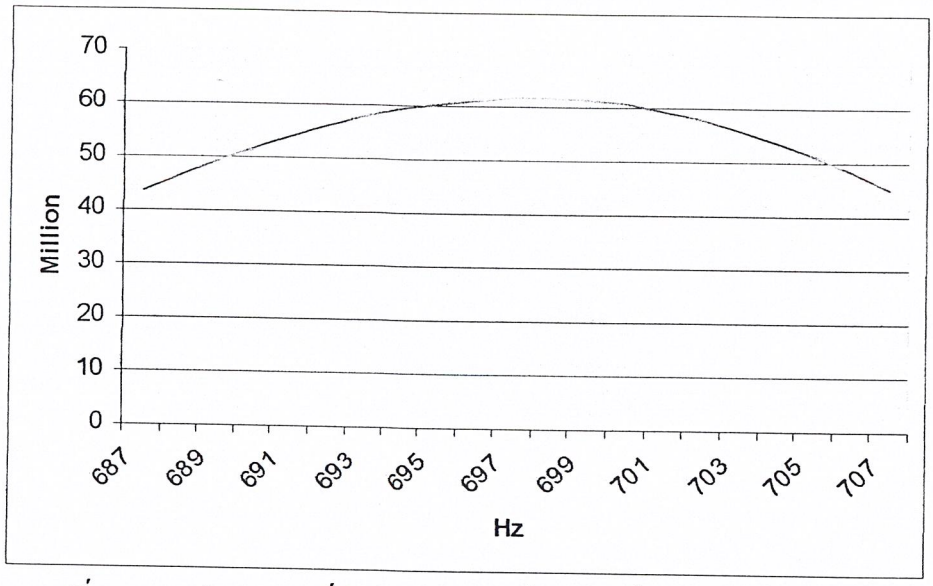
9.1 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น

ในการจะประเมินหาค่า THRESHOLD ที่จะนำไปใช้นั้น จะต้องคำนึงถึงมาตรฐานของแต่ละประเภทสัญญาณที่กำหนดไว้ในประเด็น Frequency Deviation ซึ่งเมื่อความถี่ส่วนประกอบที่เข้ามากับสัญญาณ เบี่ยงเบนไกลออกไปจากความถี่ที่ต้องการตรวจจับมากเท่าไร ค่าที่ได้จากสมการ 3-2 ก็จะน้อยลงเรื่อยๆ ดังนั้นการพิจารณาเลือกใช้ค่า THRESHOLD ที่เหมาะสมจึงเป็นเรื่องที่สำคัญ โดยจะทำการทดลองโดยการสร้างสัญญาณ ด้วยการนำความถี่ต่างๆมารวมกันซึ่งจะเป็นทั้งความถี่ที่ตรงตามมาตรฐาน และความถี่ที่เบี่ยงเบนออกไป จากนั้นนำสัญญาณต่างๆเหล่านั้นมาทำการประมวลผลด้วย Goertzel Algorithm (ใช้เพียง 250 Samples และทำการขยายสัญญาณก่อน เช่นเดียวกับขั้นตอนการตรวจจับสัญญาณ) แล้วทำการบันทึกค่าสุดท้ายที่ได้จากสมการ 3-2 เอาไว้ใช้ในการพิจารณาเลือกค่า THRESHOLD ที่ทำให้ Algorithm ทำงานได้ตามมาตรฐานในประเด็นของ Frequency Deviation

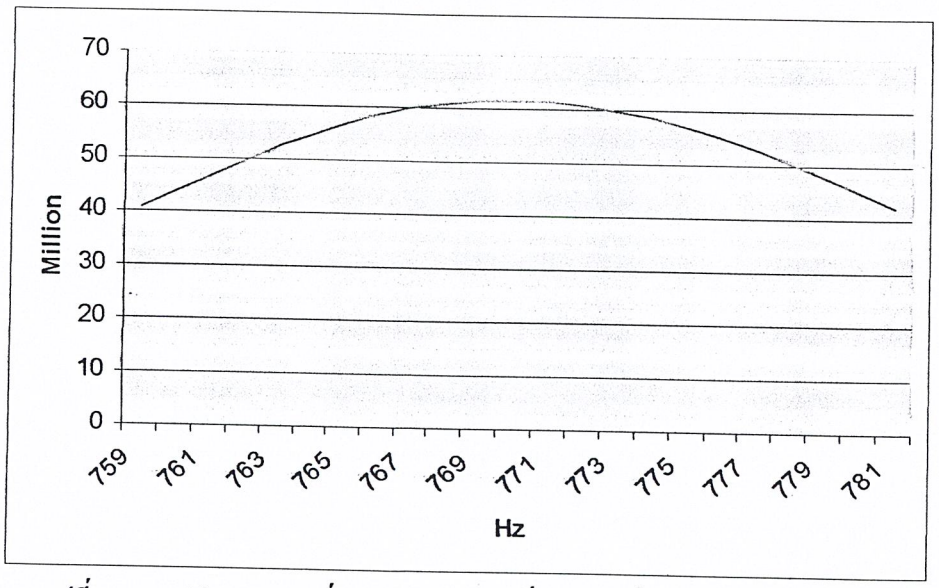
ตัวอย่างเช่น ในการประเมินค่า THRESHOLD เริ่มต้นที่จะนำไปใช้สำหรับการตรวจจับสัญญาณ DTMF นั้น จะทำการสร้างสัญญาณไซน์ ที่ความถี่ต่างๆกัน ตั้งแต่ ความถี่ปกติ – 1.5% ไปจนถึง ความถี่ปกติ + 1.5% โดยทำการเพิ่มความถี่ขึ้นทีละ 1 Hz จากนั้นก็จะนำแต่ละสัญญาณไซน์ไปประมวลผลด้วย Goertzel Algorithm โดยใช้ค่า Coefficient ที่คำนวณจากความถี่ปกติ แล้วทำการบันทึกค่าที่ได้จากสมการ 3-2 ของ Goertzel Algorithm เพื่อนำมาใช้ในการเลือกค่า THRESHOLD เริ่มต้น

แต่ค่า THRESHOLD เริ่มต้น ที่ได้จากการประเมินในขั้นนี้ ยังไม่สามารถนำไปใช้ได้โดยทันที เพียงแต่นำไปเป็นค่าเริ่มต้นก่อนเท่านั้น โดยจะต้องมีการปรับค่า THRESHOLD อีกในขั้นตอนการทดสอบการตรวจจับสัญญาณที่มีการเบี่ยงเบนความถี่ต่างๆกัน จากการทดลองในขั้นนี้ จะได้ค่า THRESHOLD ที่จะสามารถนำไปใช้งานได้

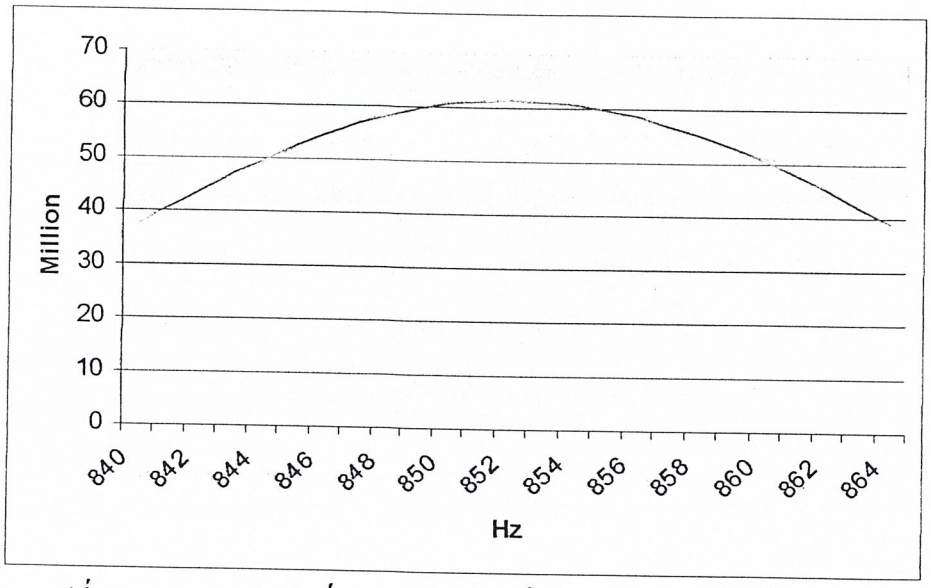
9.1.1 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น สำหรับสัญญาณ DTMF



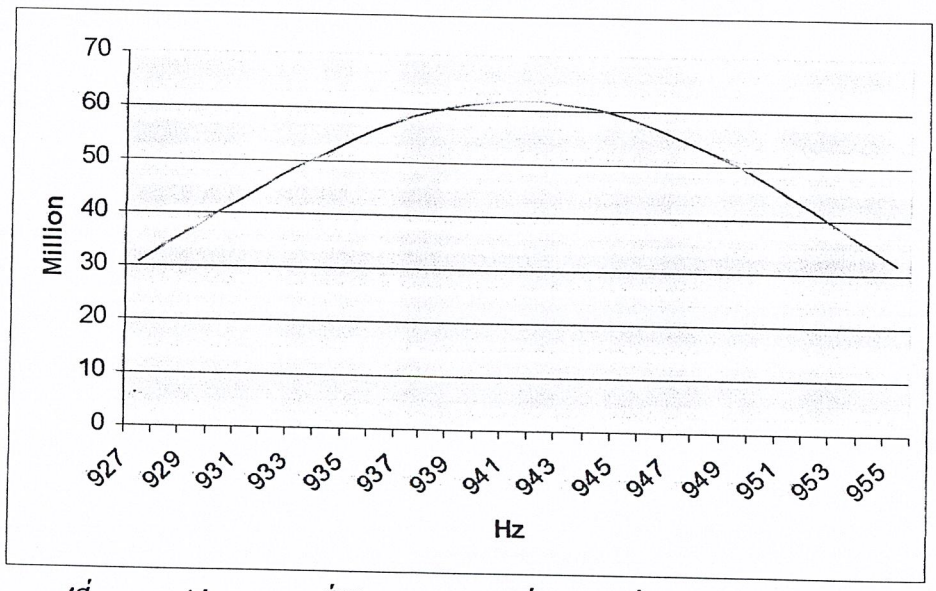
รูปที่ 9-1 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 697 Hz ใช้ 250 Samples



รูปที่ 9-2 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 770 Hz ใช้ 250 Samples



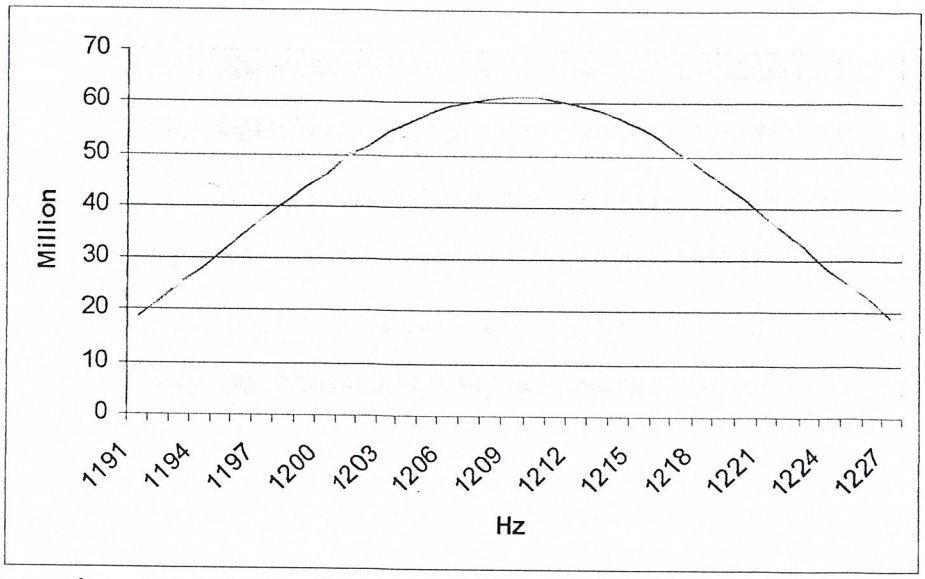
รูปที่ 9-3 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 852 Hz ใช้ 250 Samples



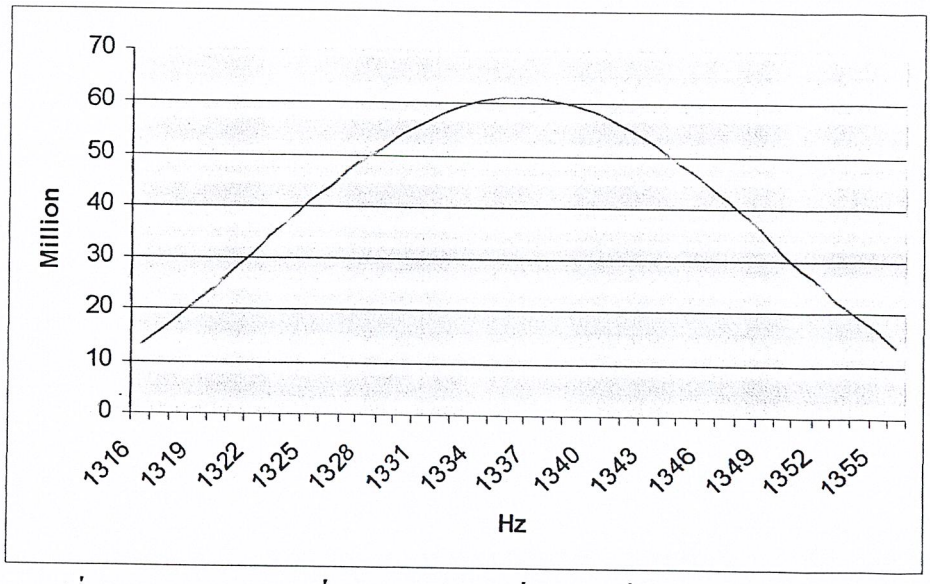
รูปที่ 9-4 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 941 Hz ใช้ 250 Samples

จาก 4 รูปที่ผ่านมา แสดงถึงค่าที่ได้จาก Goertzel Algorithm โดยทำการสร้างสัญญาณแต่ละความถี่ใน Low-Frequency Group ของสัญญาณ DTMF โดยสร้างจนครอบคลุม ±1.5% Frequency Deviation แล้วทำการใช้ค่า Coefficient ที่คำนวณจากความถี่ปกติ (±0%) ในการประมวลผลด้วย Goertzel Algorithm

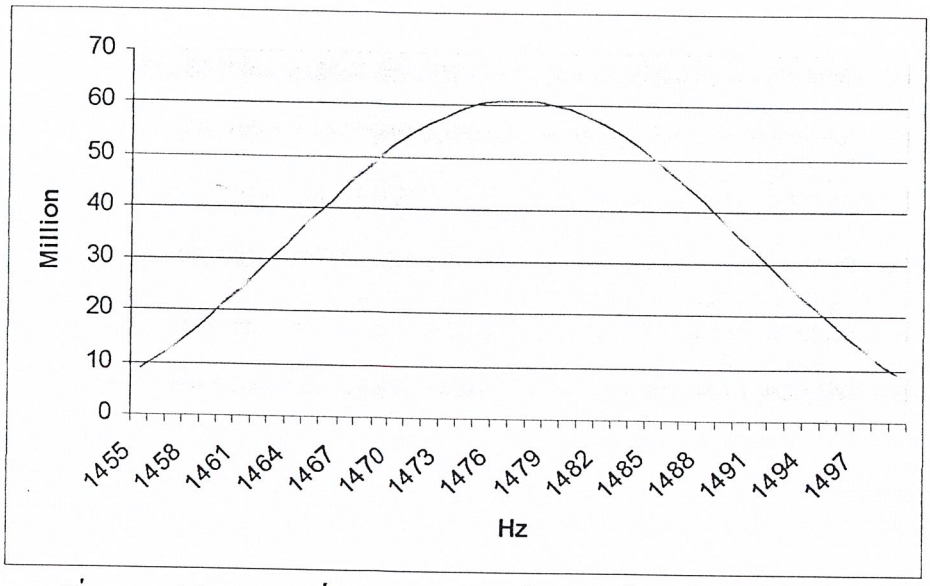
จะเห็นว่าค่า THRESHOLD ที่เหมาะจะนำมาใช้สำหรับการตรวจจับ Low-Frequency Group อยู่ที่ประมาณ 25 ล้าน



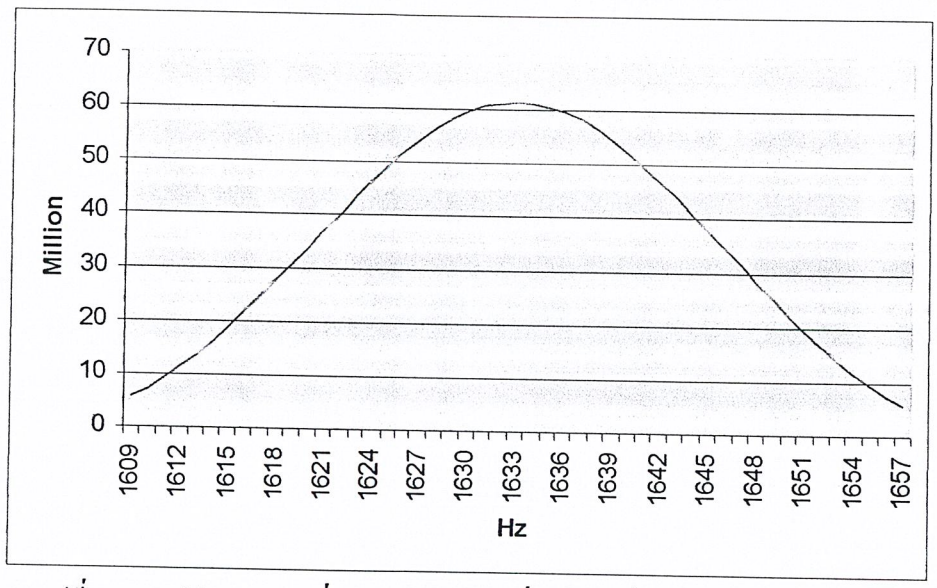
รูปที่ 9-5 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1209 Hz ใช้ 250 Samples



รูปที่ 9-6 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1336 Hz ใช้ 250 Samples



รูปที่ 9-7 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1477 Hz ใช้ 250 Samples

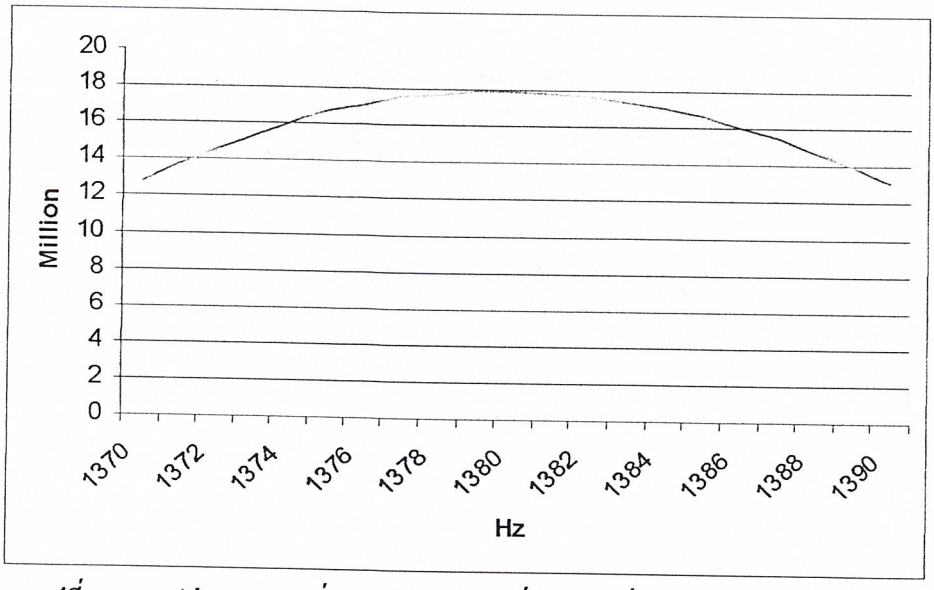


รูปที่ 9-8 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1633 Hz ใช้ 250 Samples

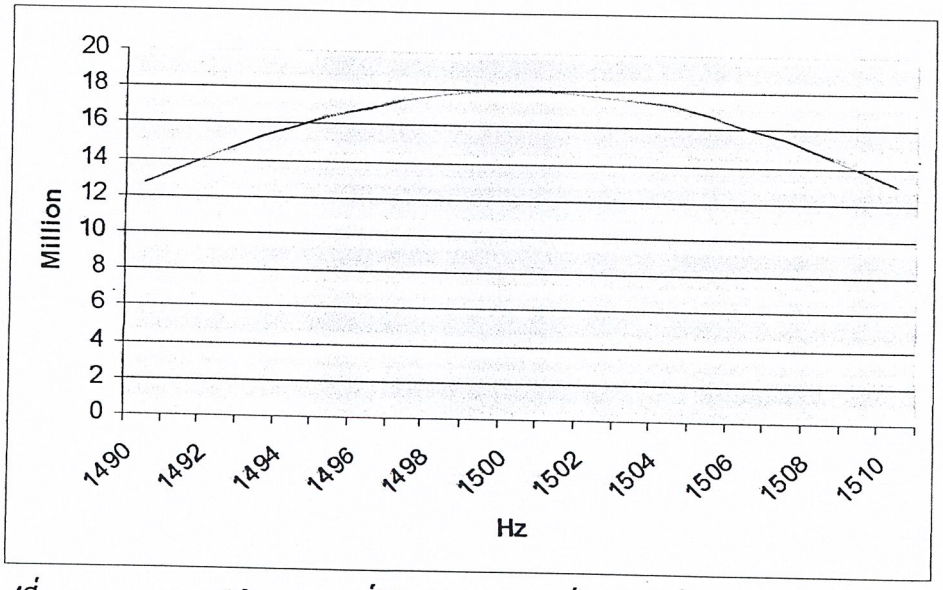
จาก 4 รูปที่ผ่านมา แสดงถึงค่าที่ได้จาก Goertzel Algorithm โดยทำการสร้างสัญญาณแต่ละความถี่ใน High-Frequency Group ของสัญญาณ DTMF โดยสร้างจนครอบคลุม ±1.5% Frequency Deviation แล้วทำการใช้ค่า Coefficient ที่คำนวณจากความถี่ปกติ (±0%) ในการประมวลผลด้วย Goertzel Algorithm

จะเห็นว่าค่า THRESHOLD ที่เหมาะจะนำมาใช้สำหรับการตรวจจับ High-Frequency Group อยู่ที่ประมาณ 7

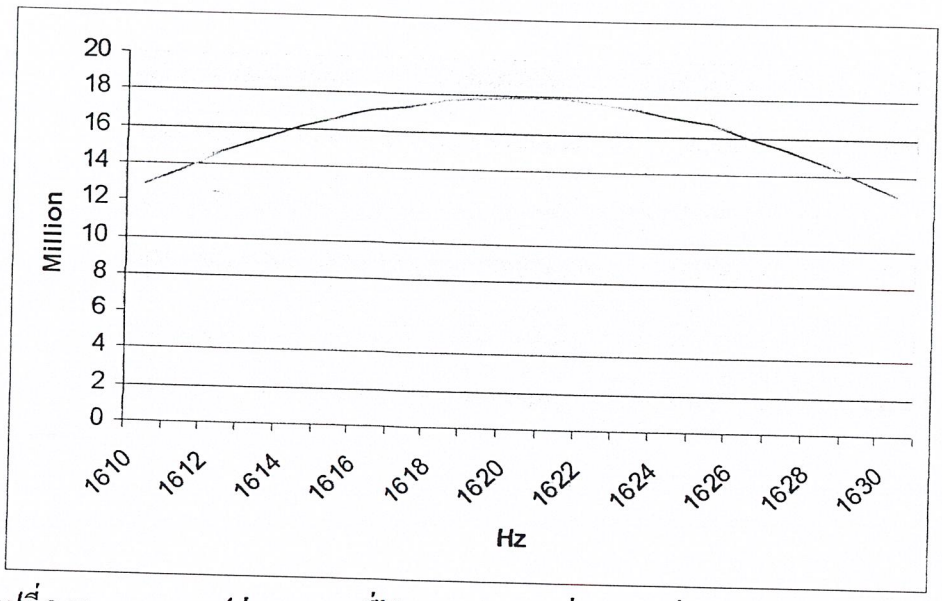
9.1.2 การพิจารณาประเมินค่า THRESHOLD เริ่มต้น สำหรับสัญญาณ R2 MF



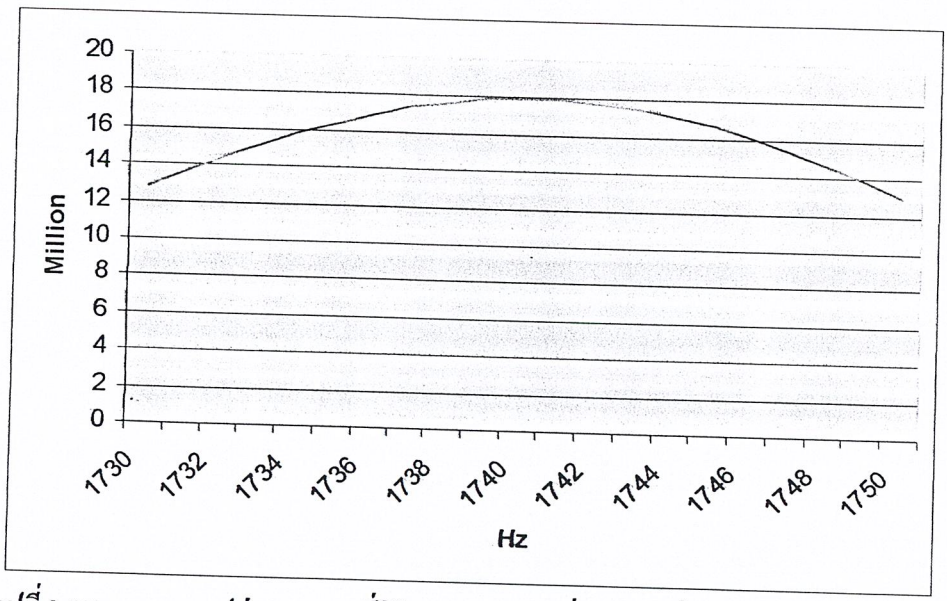
รูปที่ 9-9 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1380 Hz ใช้ 250 Samples



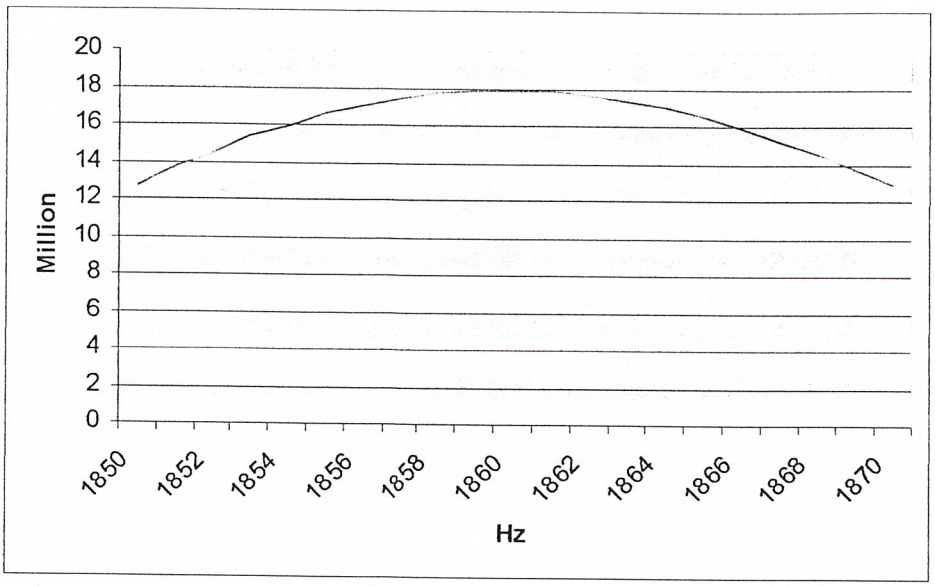
รูปที่ 9-10 กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1500 Hz ใช้ 250 Samples



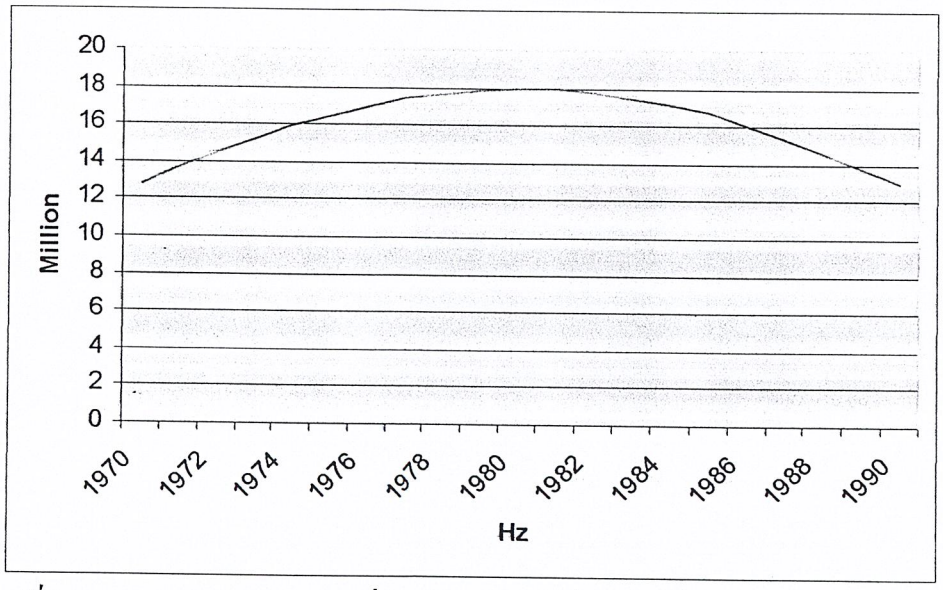
รูปที่ 9-11 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1620 Hz ใช้ 250 Samples***



รูปที่ 9-12 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1740 Hz ใช้ 250 Samples***



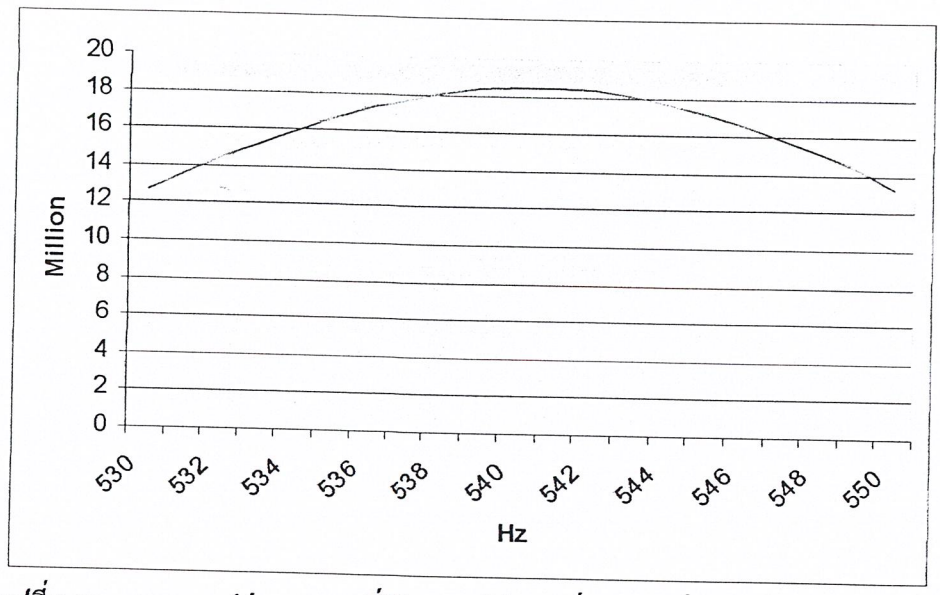
รูปที่ 9-13 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1860 Hz ใช้ 250 Samples***



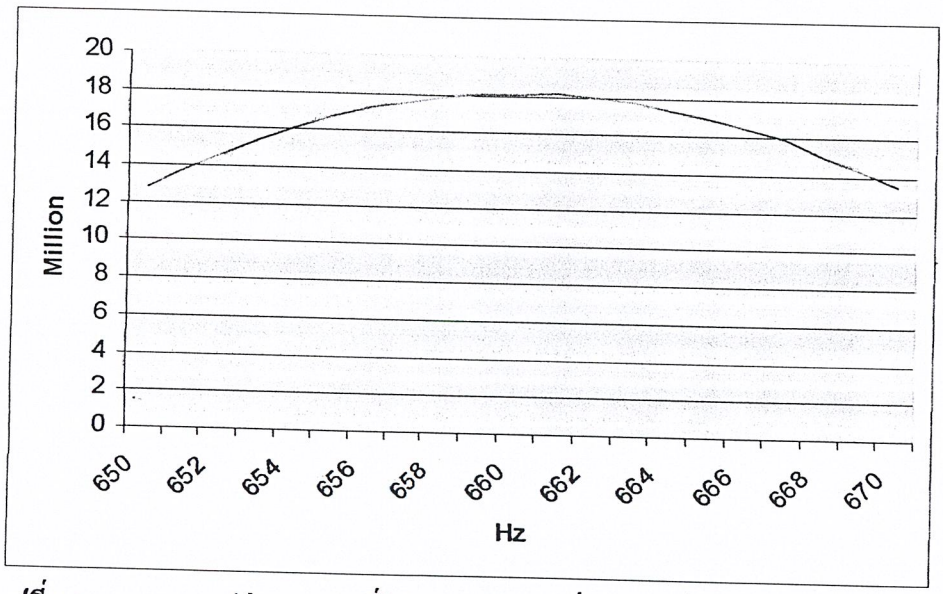
รูปที่ 9-14 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1980 Hz ใช้ 250 Samples***

จาก 6 รูปที่ผ่านมา แสดงถึงค่าที่ได้จาก Goertzel Algorithm โดยทำการสร้างสัญญาณแต่ละความถี่ของ Forward R2 MF โดยสร้างจนครอบคลุม ±10 Hz Frequency Deviation แล้วทำการใช้ค่า Coefficient ที่คำนวณจากความถี่ปกติ (±0 Hz) ในการประมวลผลด้วย Goertzel Algorithm

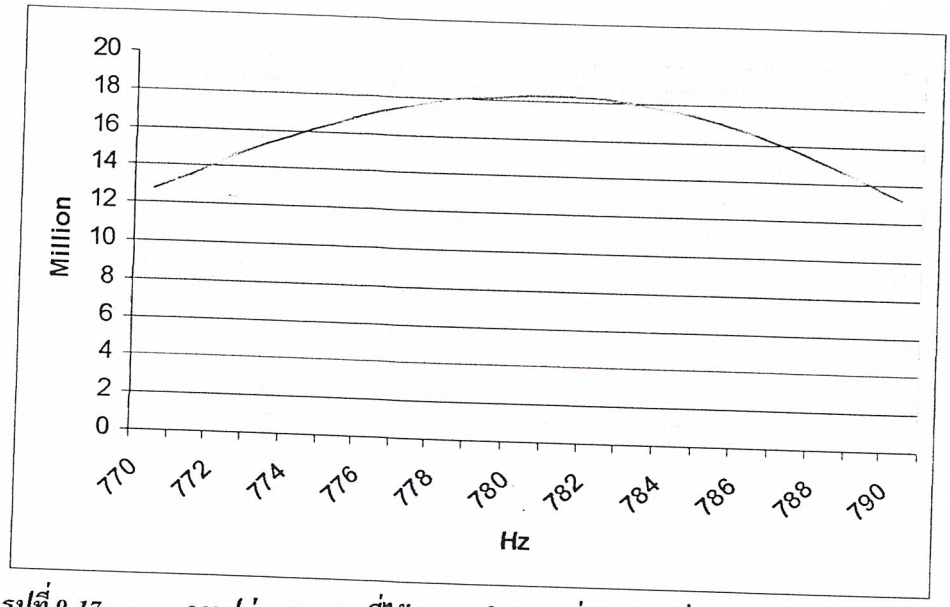
จะเห็นว่าค่า THRESHOLD ที่เหมาะจะนำมาใช้สำหรับการตรวจจับสัญญาณ Forward R2 MF อยู่ที่ประมาณ 12 ล้าน



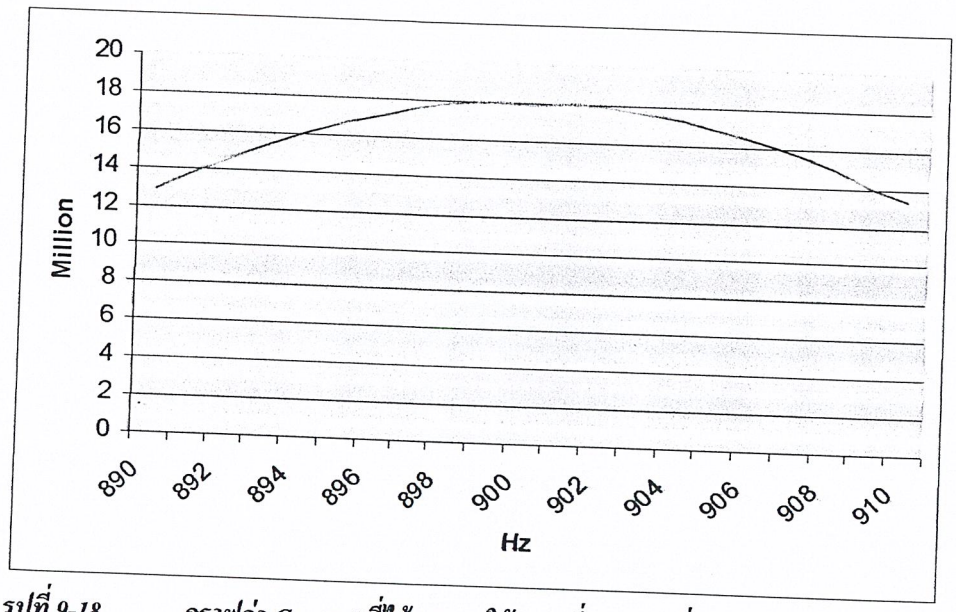
รูปที่ 9-15 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 540 Hz ใช้ 250 Samples***



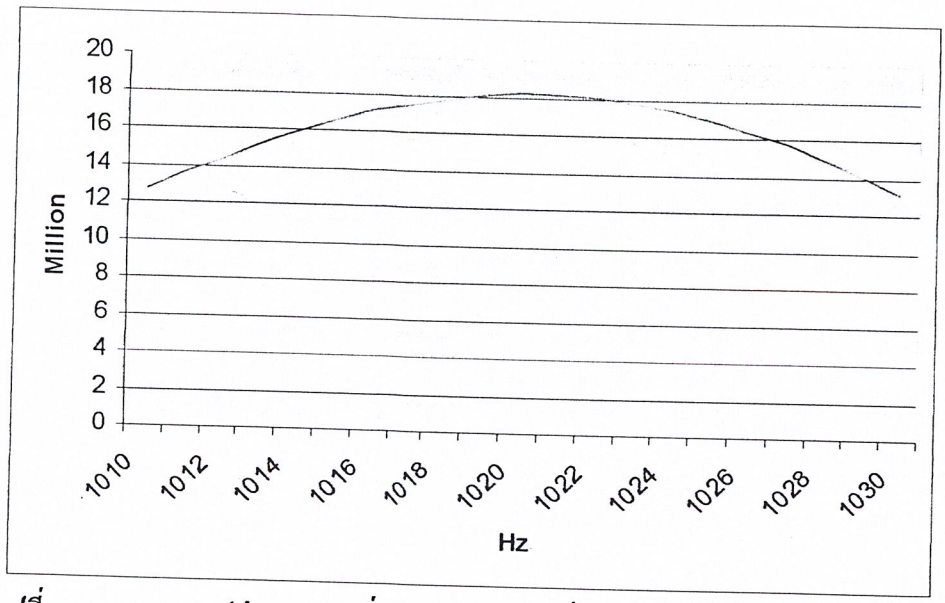
รูปที่ 9-16 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 660 Hz ใช้ 250 Samples***



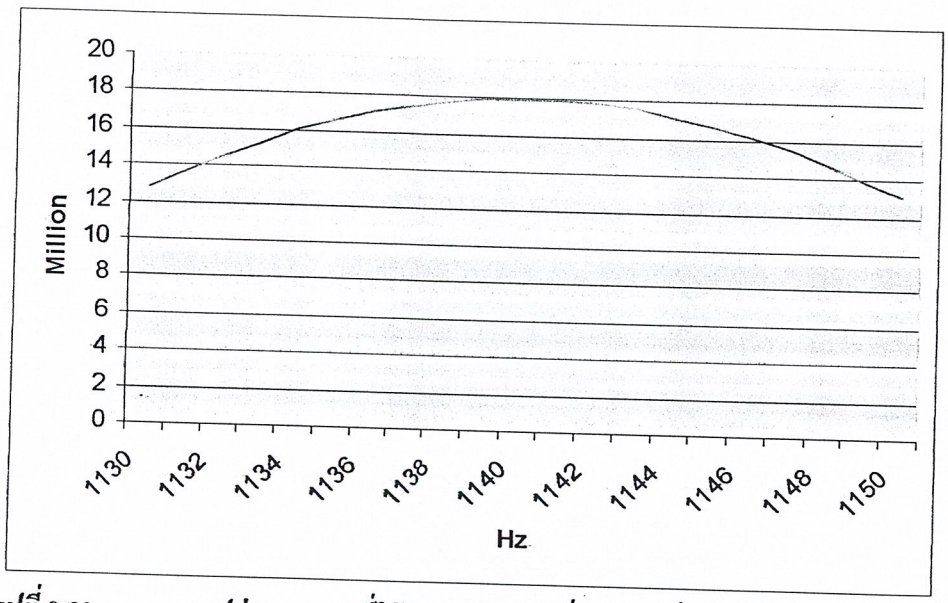
รูปที่ 9-17 กราฟค่า *Goertzel* ที่ได้จากการใช้ความถี่ตรวจจับที่ *780 Hz* ใช้ *250 Samples*



รูปที่ 9-18 กราฟค่า *Goertzel* ที่ได้จากการใช้ความถี่ตรวจจับที่ *900 Hz* ใช้ *250 Samples*



รูปที่ 9-19 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1020 Hz ใช้ 250 Samples***



รูปที่ 9-20 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1140 Hz ใช้ 250 Samples***

จาก 6 รูปที่ผ่านมา แสดงถึงค่าที่ได้จาก Goertzel Algorithm โดยทำการสร้างสัญญาณแต่ละความถี่ของ Backward R2 MF โดยสร้างจนครอบคลุม ±10 Hz Frequency Deviation แล้วทำการใช้ค่า Coefficient ที่คำนวณจากความถี่ปกติ (±0 Hz) ในการประมวลผลด้วย Goertzel Algorithm

จะเห็นว่าค่า THRESHOLD ที่เหมาะจะนำมาใช้สำหรับการตรวจจับสัญญาณ Backward R2 MF อยู่ที่ประมาณ 12 ล้าน

9.1.3 ค่า THRESHOLD เริ่มต้น ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF

ที่ต้องทำการกำหนดค่า THRESHOLD แยกเป็น 2 ประเภทสำหรับ สัญญาณ DTMF นั้น ก็เพราะที่อัตรา Frequency Deviation ที่เท่ากัน เช่น 1.5% แต่ปริมาณความถี่ที่เปลี่ยนไปจะไม่เท่ากัน โดย High-Frequency Group จะเกิดการเบี่ยงเบนมากกว่า Low-Frequency Group จึงต้องทำการใช้ THRESHOLD แยกกัน แต่กับสัญญาณ R2 MF แม้จะทำการแยก THRESHOLD ที่ใช้สำหรับ R2 MF Forward, และ R2 MF Backward เอาไว้ แต่ในขั้นแรกจะทำตั้งค่า THRESHOLD เอาไว้เท่ากัน ในขั้นแรกนี้ทำให้ได้ค่า THRESHOLD เริ่มต้น ดังนี้

	DTMF		R2 MF	
	High-Frq. Group	Low-Frq. Group	Forward R2 MF	Backward R2 MF
THRESHOLD (ล้าน)	7	25	12	12

ตารางที่ 9-2 ค่า THRESHOLD เริ่มต้น ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF

9.2 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ DTMF

	การเบี่ยงเบนความถี่	ITU-T Q.24	ผลการตรวจจับ	ปัญหาที่พบ	การแก้ไข
1	±0%_high ±0.5%_low	ต้องตรวจจับได้	ผ่าน	-	-
2	±0%_high ±1.0%_low	ต้องตรวจจับได้	ผ่าน	-	-
3	±0%_high ±1.5%_low	ต้องตรวจจับได้	ผ่าน	-	-
4	±0%_high ±2.0%_low	ไม่สนใจ	ผ่าน	-	-
5*	±0%_high ±3.5%_low	ต้องตรวจจับไม่ได้	ไม่ผ่าน	ตรวจจับบางหมายเลขได้	ปรับเพิ่ม T/H Low-Frq. เป็น 27 ล้าน
6*	(ซ้ำ 5)	ต้องตรวจจับไม่ได้	ผ่าน	-	-
7*	±0%_high ±4.0%_low	ต้องตรวจจับไม่ได้	ผ่าน	-	-

8	±0.5%_high ±0%_low	ต้องตรวจจับได้	ผ่าน	-	-
9	±1.0%_high ±0%_low	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับบาง หมายเลข ไม่ได้	ปรับลด T/H High-Frq. เป็น 6 ล้าน
10	(ซ้ำ 9)	ต้องตรวจจับได้	ผ่าน	-	-
11*	±1.5%_high ±0%_low	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับบาง หมายเลข ไม่ได้	ปรับลด T/H Low-Frq. เป็น 13 ล้าน
12	(ซ้ำ 11)	ต้องตรวจจับได้	ผ่าน	-	-
13	±2.0%_high ±0%_low	ไม่สนใจ	ผ่าน	-	-
14	±3.5%_high ±0%_low	ต้องตรวจจับไม่ได้	ผ่าน	-	-
15	±4.0%_high ±0%_low	ต้องตรวจจับไม่ได้	ไม่ผ่าน	-	-
16	±1.5%_high ±1.5%_low	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับบาง หมายเลข ไม่ได้	ปรับลด T/H High-Frq. เป็น 5.8 ล้าน
17	(ซ้ำ 16)	ต้องตรวจจับได้	ผ่าน	-	-
18	±3.5%_high ±3.5%_low	ต้องตรวจจับไม่ได้	ผ่าน	-	-

ตารางที่ 9-3 สรุปผลการทดลอง การประเมินค่า THRESHOLD สำหรับสัญญาณ DTMF

ในการทดลองนั้น สัญญาณ DTMF ที่นำมาทดสอบแต่ละหัวข้อนั้นคือสัญญาณ DTMF ที่ประกอบด้วยสัญญาณทั้งหมดของ DTMF ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, และ D โดยใช้ Signal Duration เท่ากับ Pause Duration เท่ากับ 50 mS และทำการเบี่ยงเบนความถี่ที่ใช้จากความถี่มาตรฐานตามอัตราส่วนร้อยละ ที่กำหนดให้แต่ละหัวข้อ

ในหัวข้อที่ 11 เมื่อทำการปรับลดค่า THRESHOLD แล้ว จะทำให้หัวข้อที่ 11 นั้นผ่าน แต่จะมีผลกระทบกับหัวข้อ 5, 6, และ 7 ที่ได้ทำผ่านมาแล้ว โดยจะทำให้สามารถตรวจพบบางสัญญาณ แต่เนื่องจากหัวข้อ 11 นั้น จะเป็น ±1.5% Frequency Deviation ซึ่งต้องตรวจจับให้ได้ ในขณะที่ หัวข้อ 5, 6, และ 7 นั้น เป็นช่วง Frequency Deviation ที่จะต้องตรวจจับไม่พบ โดยในที่นี้จะให้ความสำคัญกับช่วง Frequency Deviation ที่ต้องตรวจจับให้ได้ก่อน ดังนั้นจึงยอมที่จะใช้ค่า THRESHOLD ดังกล่าว

9.3 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ R2 MF

	การเบี่ยงเบนความถี่	ITU-T Q.24	ผลการตรวจจับ	ปัญหาที่พบ	การแก้ไข
1	±5_a ±0_b	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับสัญญาณผิด	-
2	±10_a ±0_b	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับสัญญาณผิด	ปรับลด T/H Forward เป็น 8 ล้าน
3	(ซ้ำ 2)	ต้องตรวจจับได้	ผ่าน	-	-
4	±0_a ±5_b	ต้องตรวจจับได้	ผ่าน	-	-
5	±0_a ±10_b	ต้องตรวจจับได้	ไม่ผ่าน	ตรวจจับสัญญาณผิด	ปรับลด T/H Backward เป็น 10 ล้าน
6	(ซ้ำ 5)	ต้องตรวจจับได้	ผ่าน	-	-

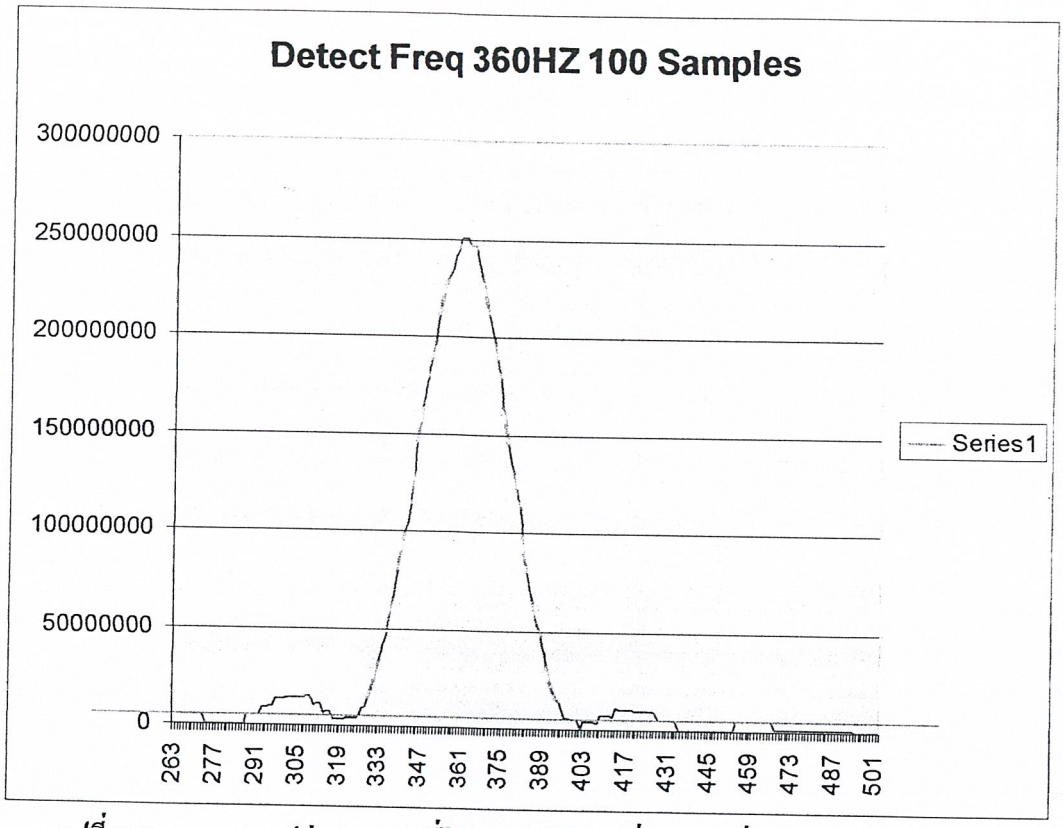
ตารางที่ 9-4 สรุปผลการทดลอง การประเมินค่า THRESHOLD สำหรับสัญญาณ R2 MF

ในการทดลองนั้น สัญญาณ R2 MF ที่นำมาทดสอบแต่ละหัวข้อนั้นคือสัญญาณ R2 MF ที่ประกอบด้วยสัญญาณทั้งหมดของ R2 MF ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 (A'), 12 (B'), 13 (C'), 14 (D'), และ 15 (E') ทั้ง Forward, Backward โดยทำการเบี่ยงเบนความถี่ที่ใช้ตามที่กำหนดให้ในแต่ละหัวข้อ

9.4 การทดลอง เพื่อประเมินค่า THRESHOLD สำหรับสัญญาณ CPT

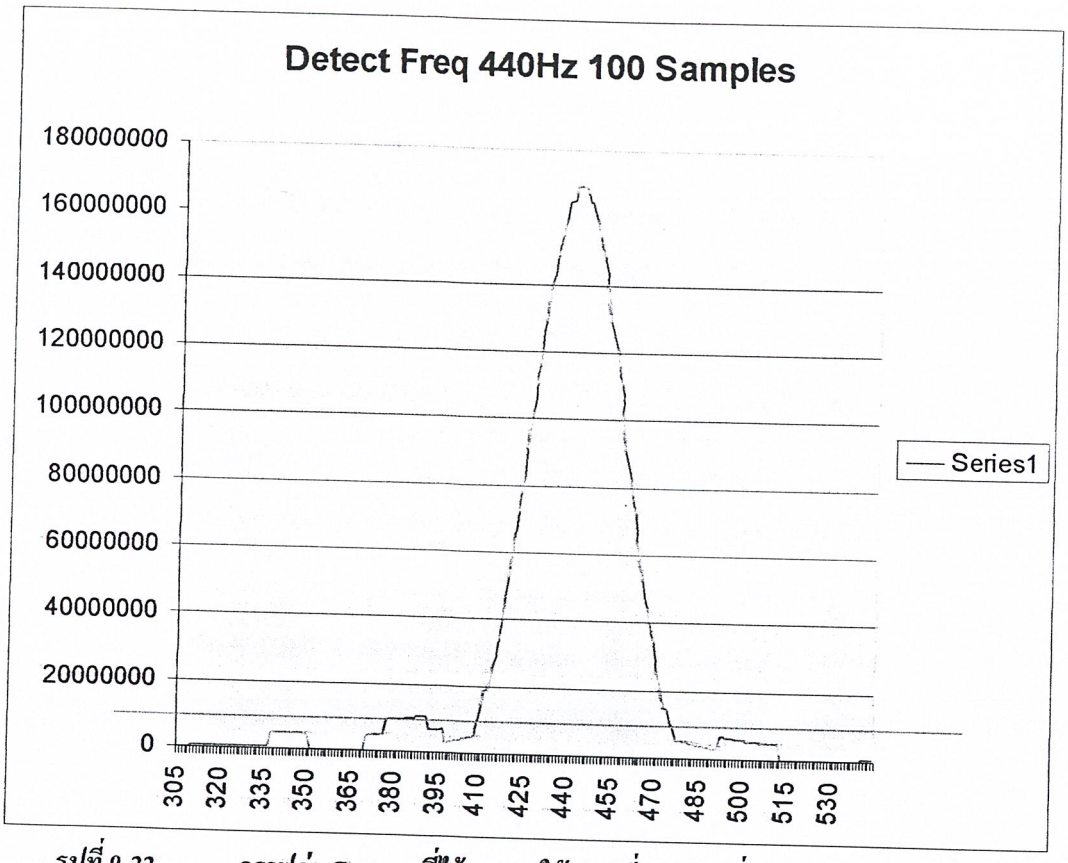
ในการประเมินค่า THRESHLOD ของ CPT นั้น จะใช้วิธีสร้างกราฟ โดยแกน Y เป็นค่าที่ได้จากสมการ Goertzel และแกน X เป็นความถี่ที่นำมาตรวจจับ ส่วนความถี่ที่ใช้ตรวจจับและจำนวน Sample ที่ใช้ตรวจจับนั้นจะอยู่ที่หัวเรื่องของกราฟนั้น

สัญญาณ CPT นั้นหลายสัญญาณใช้ความถี่เดียวกัน จะค่อยๆกล่าวถึงทีละความถี่ดังนี้



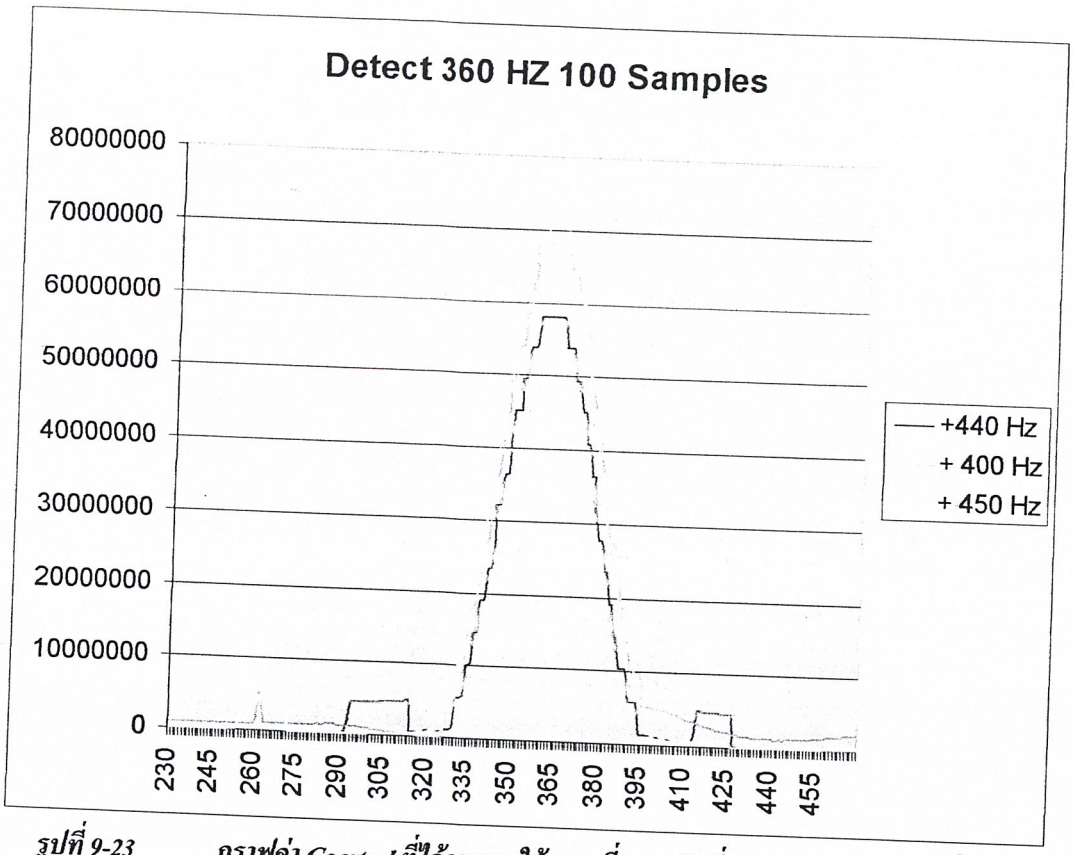
รูปที่ 9-21 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 360 Hz ใช้ 100 Samples***

สัญญาณที่มาตรฐานกำหนดว่าต้องตรวจจับได้ในช่วง 340-425 Hz นั้นจะใช้ความถี่ 360 Hz ในการคำนวณหาค่า Goertzel ซึ่งจากกราฟจะเห็นว่า เราเลือกเส้นสีแดงเป็น THRESHOLD คือ 5,500,000 ซึ่งดูแล้วอาจไม่ค่อยดีนักเพราะด้านซ้ายและขวามีความถี่ที่ไม่ต้องการผ่านค่า THRESHOLD แต่ก็ใกล้เคียงกับที่มาตรฐานต้องการมากที่สุดแล้ว ซึ่งที่ความถี่ 360 นั้นมีสัญญาณ CPT ดังนี้ Dial Tone, Recall Dial Tone, Confirmation



รูปที่ 9-22 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 440 Hz ใช้ 100 Samples***

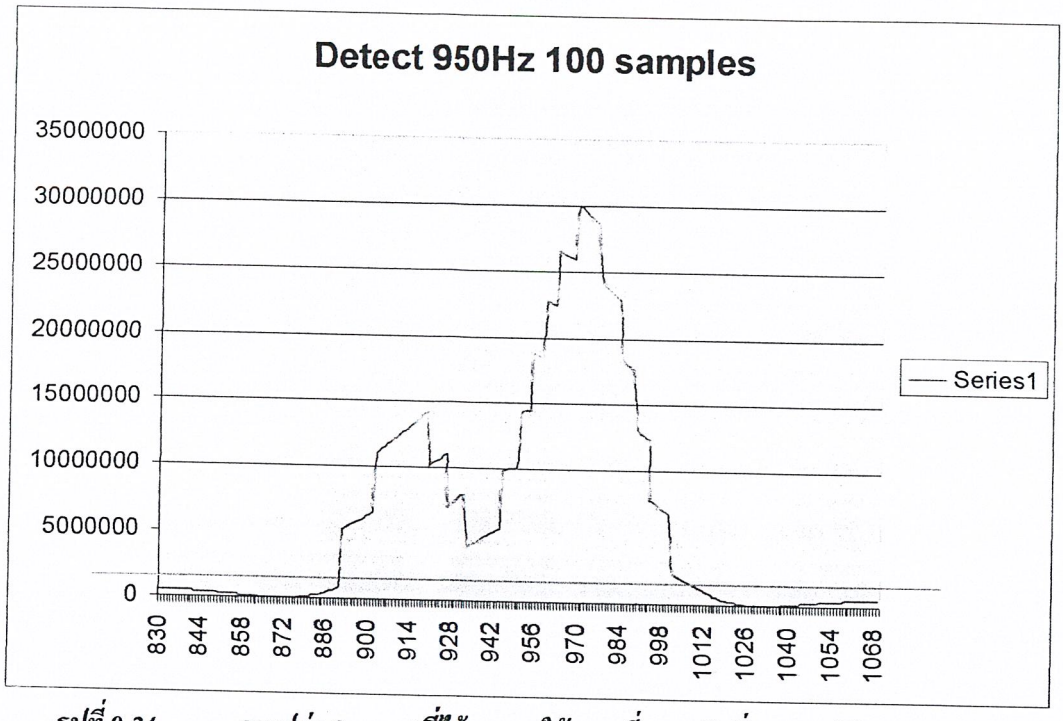
ความถี่ 440 Hz นั้นก็เลือกใช้ THRESHOLD 9,500,000 เพราะถ้าเพิ่มมากกว่านี้ความถี่ที่ตรวจจับได้จะมีช่วงที่แคบเกินไป สัญญาณ CPT ที่ใช้ความถี่นี้มีดังนี้ Dial Tone, Ring Tone, Busy Tone, Congestion Tone, Call Waiting, Recall Dial Tone, Confirmation, Busy Verification และ Caller Waiting



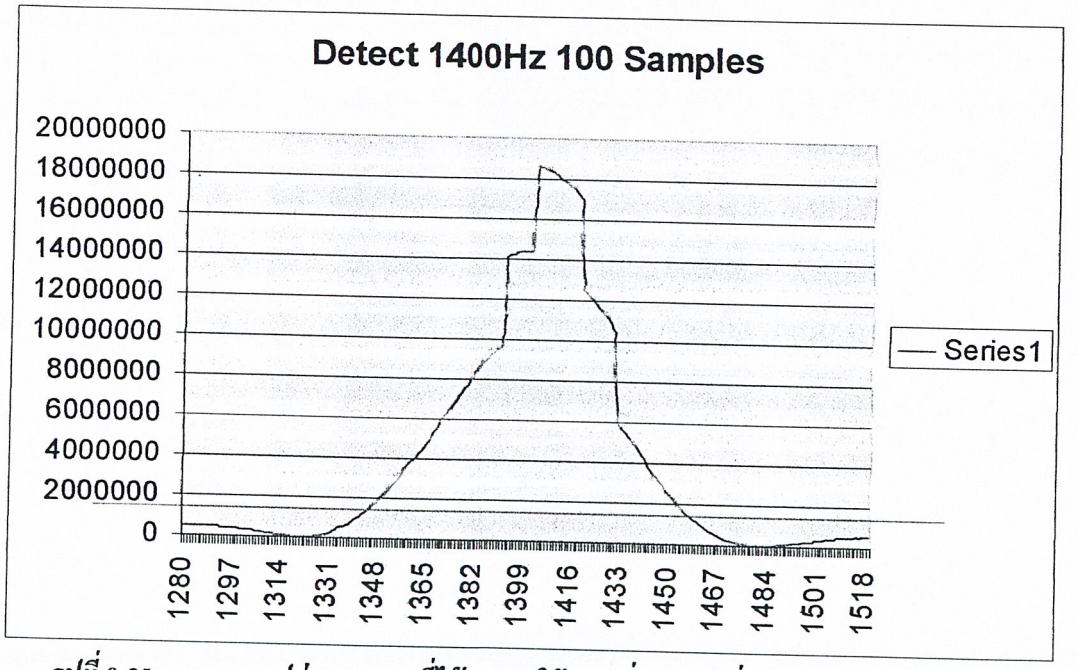
รูปที่ 9-23 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 360 Hz รวมกับความถี่ต่างๆ ใช้ 100 Samples***

กราฟนี้เป็นการทดสอบของสัญญาณ Dial Tone และ Confirmation ที่เกิดจากการผสมกันระหว่าง 2 ความถี่จะเห็นว่าการที่ผสมกับความถี่สูงขึ้นค่าที่ได้จาก Goertzel จะมีค่าสูงขึ้นตามไปด้วย ในบางกรณีความถี่ต่ำกว่าค่า THRESHOLD เล็กน้อยเมื่อผสมกับความถี่สูงก็อาจจะสามารถตรวจจับได้เนื่องจากความถี่เพิ่มขึ้น

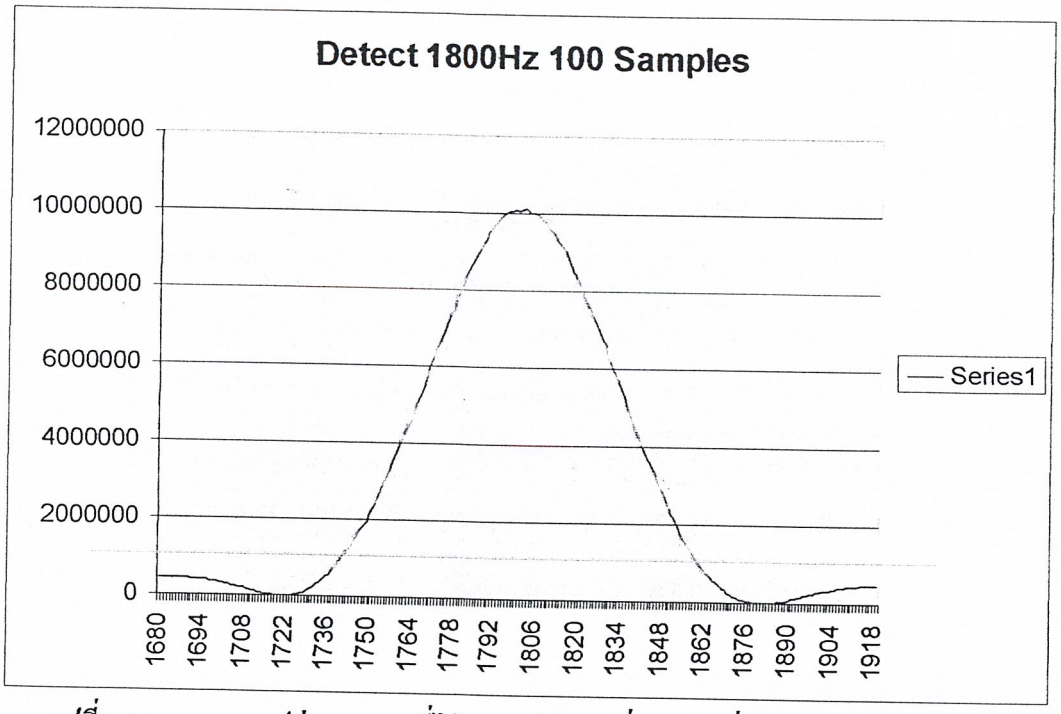
3 ความถี่จากนี้เป็นของ Special Information



รูปที่ 9-24 *กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 950 Hz ใช้ 100 Samples*



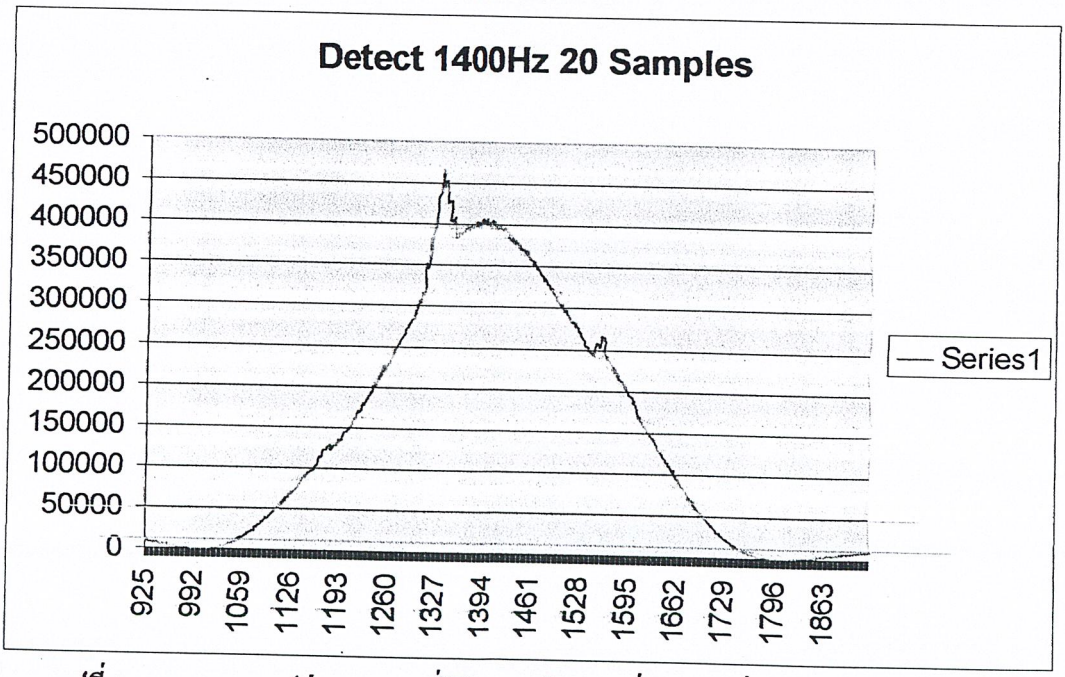
รูปที่ 9-25 *กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1400 Hz ใช้ 100 Samples*



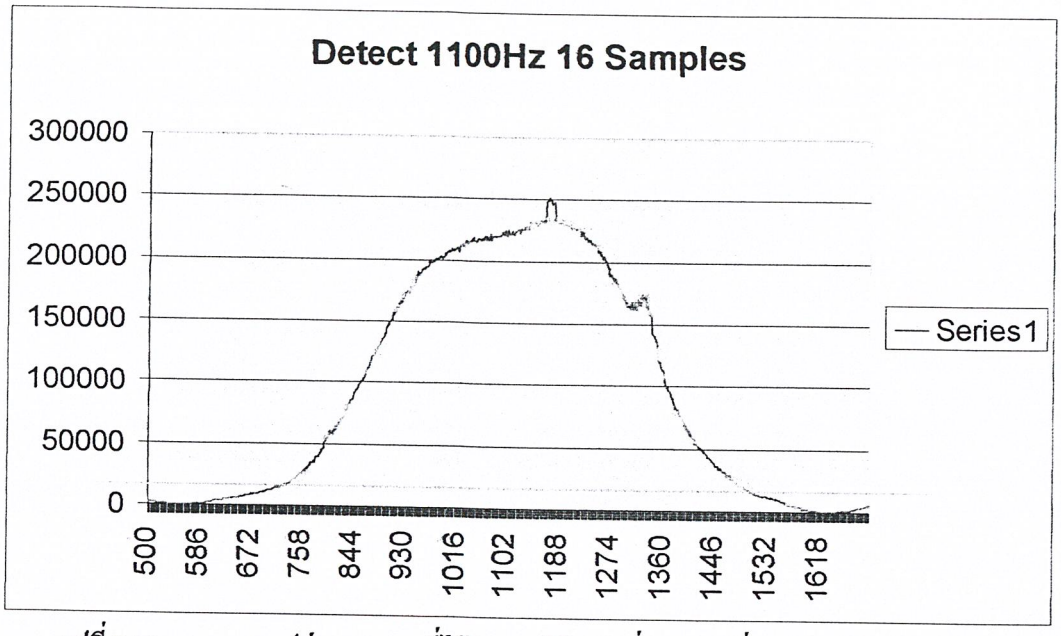
รูปที่ 9-26 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1800 Hz ใช้ 100 Samples***

สำหรับ Special Information ทั้ง 3 ความถี่โดยเลือก THRESHOLD ดังนี้ 600,000(950 Hz), 2,000,000(1,400 Hz), 1,500,000(1,800 Hz)

หลังจากนี้อีก 2 ความถี่เป็นการประเมินของ Payphone Tone

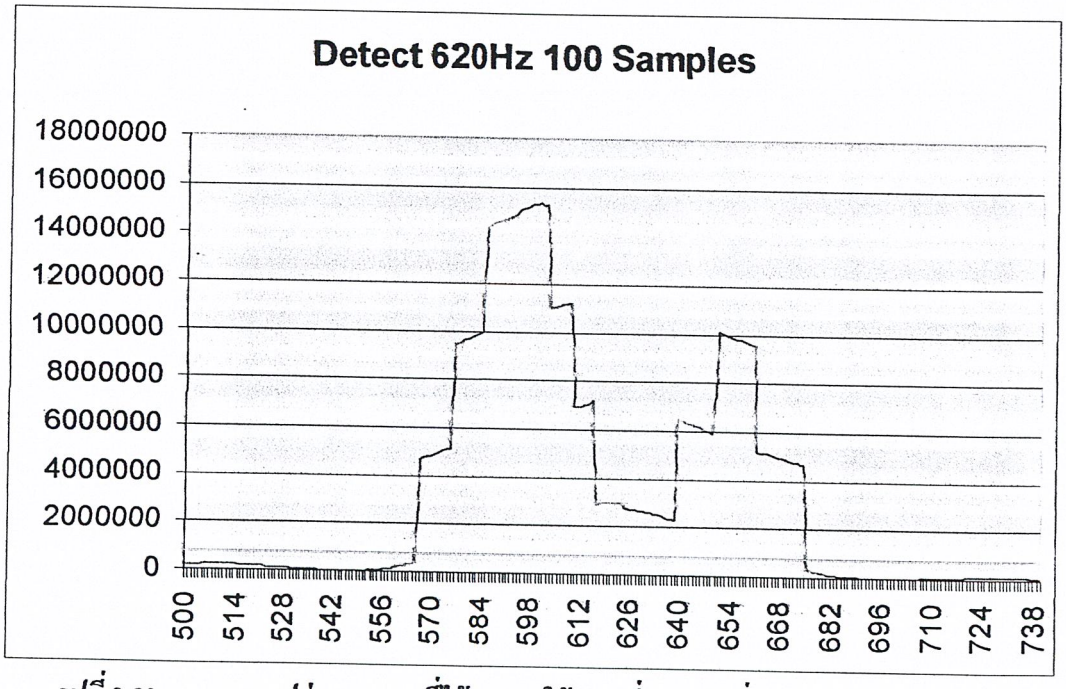


รูปที่ 9-27 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1400 Hz ใช้ 20 Samples***



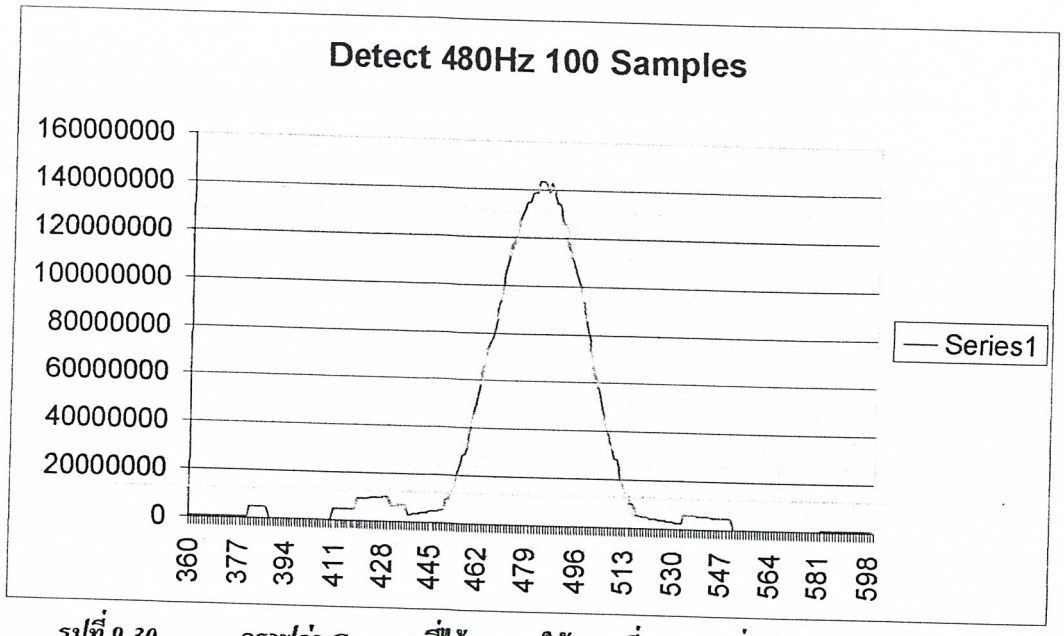
รูปที่ 9-28 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 1100 Hz ใช้ 16 Samples***

2 สัญญาณนี้เป็นของ Payphone Tone ใช้ค่า THRESHOLD 40,000(1,400 Hz), 20,000(1,100 Hz)



รูปที่ 9-29 ***กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 620 Hz ใช้ 100 Samples***

ความถี่นี้ของ Intercept Tone เลือกค่า THRESHOLD ที่ 5,000,000



รูปที่ 9-30 *กราฟค่า Goertzel ที่ได้จากการใช้ความถี่ตรวจจับที่ 480 Hz ใช้ 100 Samples*

สุดท้ายเป็นของ Comfort เลือก THRESHOLD 10,000,000

9.5 สรุปค่า THRESHOLD ที่จะนำไปใช้ ของ DTMF, R2MF

	DTMF		R2 MF	
	High-Frq. Group	**Low-Frq. Group**	**Forward R2 MF**	**Backward R2 MF**
THRESHOLD (ล้าน)	5.8	13	8	10

ตารางที่ 9-5 *ค่า THRESHOLD ที่ปรับแล้ว ของส่วนตรวจจับสัญญาณ DTMF, และ R2 MF*

บทที่ 10

การทดสอบ การทำงานของ Library

10.1 การทดสอบ การทำงานของ Library สร้างสัญญาณ DTMF

การทดสอบ การทำงานของ Library สร้างสัญญาณ DTMF ทำโดยการตั้งค่า จังหวะของสัญญาณ, และค่าความถี่ที่ใช้ (กำหนดโดยใช้ค่า Step Size) ให้เป็นไปตามมาตรฐานที่กำหนดใน ITU-T Recommendation Q.24 (AT&T) โดยมีการตั้งค่าจังหวะของสัญญาณ, และค่าความถี่ที่ใช้ ดังนี้

การตั้งค่า	ค่าที่ใช้
Signal Duration	50 mS
Pause Duration	50 mS
ความถี่ที่ใช้	ใช้ความถี่ที่อ้างอิงตามบทที่ 2

ตารางที่ 10-1 การตั้งค่าในการทดสอบ การทำงานของ Library สร้างสัญญาณ DTMF

จากนั้นทำการสร้างสัญญาณทุกตัวของ DTMF ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, และ D แล้วนำสัญญาณที่ได้ไปตรวจสอบด้วย โปรแกรมตรวจจับสัญญาณ DTMF ที่พัฒนาขึ้นบน PC

ผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.2 การทดสอบ การทำงานของ Library สร้างสัญญาณ R2 MF

การทดสอบ การทำงานของ Library สร้างสัญญาณ R2 MF ทำโดยการตั้งค่าความถี่ที่ใช้ (กำหนดโดยใช้ค่า Step Size) ให้เป็นไปตามมาตรฐานที่กำหนดใน บทที่ 2 โดยมีการตั้งค่าต่างๆ ดังนี้

การตั้งค่า	ค่าที่ใช้
Signal Duration	> 7 mS
Percent Power	55% (ระดับค่าของ Sample สูงสุด ไม่เกิน 55% x 4095)
ความถี่ที่ใช้	ใช้ความถี่ที่อ้างอิงตามบทที่ 2

ตารางที่ 10-2 การตั้งค่าในการทดสอบ การทำงานของ Library สร้างสัญญาณ R2 MF

จากนั้นทำการสร้างสัญญาณทุกตัวของ R2 MF ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 (A'), 12 (B'), 13 (C'), 14 (D'), และ 15 (E') ทั้ง Forward, Backward แล้วนำสัญญาณที่ได้ไปตรวจสอบด้วย โปรแกรมตรวจจับสัญญาณ R2 MF ที่พัฒนาขึ้นบน PC

ผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.3 การทดสอบ การทำงานของ Library สร้างสัญญาณ CPT

ทดสอบการสร้างสัญญาณโดยสร้างสัญญาณตามมาตรฐานที่กล่าวไปแล้วในบทที่ 2 ในหัวข้อสัญญาณ CPT ซึ่งผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.4 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ DTMF

การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ DTMF ประกอบด้วยการทดสอบต่างๆ ดังนี้ [1] การทดสอบทั่วไป, [2] การทดสอบการเบี่ยงเบนความถี่ (Frequency Deviation), และ [3] การทดสอบการเปลี่ยนแปลงจังหวะสัญญาณ

10.4.1 การทดสอบทั่วไป ของการตรวจจับสัญญาณ DTMF

ทำการสร้างสัญญาณ DTMF บน PC โดยใช้ค่า Signal Duration เท่ากับ Pause Duration เท่ากับ 50 mS และใช้ความถี่มาตรฐานตามที่อ้างอิงในบทที่ 2 ซึ่งสัญญาณ DTMF ที่ทำการสร้างขึ้นนั้นประกอบด้วย หมายเลขดังนี้ ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, และ D จากนั้นนำสัญญาณที่สร้างขึ้นมาตรวจจับด้วย Library การตรวจจับสัญญาณ DTMF บน DSP

ผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.4.2 การทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ DTMF

ทำการสร้างสัญญาณ DTMF บน PC โดยใช้ค่า Signal Duration เท่ากับ Pause Duration เท่ากับ 50 mS ซึ่งสัญญาณ DTMF ที่ทำการสร้างขึ้นนั้นประกอบด้วย หมายเลขดังนี้ ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, และ D โดยทำการเบี่ยงเบนความถี่ที่ใช้ไปจากความถี่มาตรฐาน จากนั้นนำสัญญาณที่สร้างขึ้น มาตรวจจับด้วย Library การตรวจจับสัญญาณ DTMF บน DSP

	การเบี่ยงเบนความถี่	ITU-T Q.24	ผลการตรวจจับ	ปัญหาที่พบ
1	±0%_high ±0.5%_low	ต้องตรวจจับได้	ผ่าน	-
2	±0%_high ±1.0%_low	ต้องตรวจจับได้	ผ่าน	-
3	±0%_high ±1.5%_low	ต้องตรวจจับได้	ผ่าน	-
4	±0%_high ±2.0%_low	ไม่สนใจ	ผ่าน	-

5	±0%_high ±3.5%_low	ต้องตรวจจับไม่ได้	ไม่ผ่าน*	ตรวจจับพบบางสัญญาณ
6	±0%_high ±4.0%_low	ต้องตรวจจับไม่ได้	ไม่ผ่าน*	
7	±0.5%_high ±0%_low	ต้องตรวจจับได้	ผ่าน	-
8	±1.0%_high ±0%_low	ต้องตรวจจับได้	ผ่าน	-
9	±1.5%_high ±0%_low	ต้องตรวจจับได้	ผ่าน	-
10	±2.0%_high ±0%_low	ไม่สนใจ	ผ่าน	-
11	±3.5%_high ±0%_low	ต้องตรวจจับไม่ได้	ผ่าน	-
12	±4.0%_high ±0%_low	ต้องตรวจจับไม่ได้	ผ่าน	-
13	±1.5%_high ±1.5%_low	ต้องตรวจจับได้	ผ่าน	-
14	±3.5%_high ±3.5%_low	ต้องตรวจจับไม่ได้	ผ่าน	-

ตารางที่ 10-3 ผลการทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ DTMF

ผลการทดลองปรากฏว่ามีกรณีที่ไม่ผ่าน โดยเป็นกรณีของ การเบี่ยงเบน Low-Frequency Group ไป ±3.5%, และ ±4.0% ของค่าความถี่มาตรฐาน ซึ่งอาจจะตรวจจับพบบางสัญญาณ โดยสาเหตุนั้นเกิดมาจากการเลือกค่า *THRESHOLD ที่ใช้ โดยสามารถอ่านรายละเอียดได้จาก บทที่ 9*

10.4.3 การทดสอบการเปลี่ยนแปลงจังหวะสัญญาณ ของการตรวจจับสัญญาณ DTMF

ทำการสร้างสัญญาณ DTMF บน PC โดยใช้ค่าความถี่มาตรฐานตามที่อ้างอิงในบทที่ 2 ซึ่งสัญญาณ DTMF ที่ทำการสร้างขึ้นนั้นประกอบด้วย หมายเลขดังนี้ ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, #, A, B, C, และ D โดยทำการเปลี่ยนแปลงจังหวะสัญญาณในรูปแบบต่างๆ จากนั้นนำสัญญาณที่สร้างขึ้น มาตรวจจับด้วย Library การตรวจจับสัญญาณ DTMF บน DSP

	สัญญาณ	ITU-T Q.24	ผลการตรวจจับ	ปัญหาที่พบ
1	20-10-20 60 (mS)	ต้องตรวจจับได้	ไม่ผ่าน*	G/Z ตรวจไม่พบบางสัญญาณ
2	20-11-20 60 (mS)	ต้องตรวจจับไม่ได้	ไม่ผ่าน*	ระยะ Interrupt Time ที่กระชั้น
3	20-15-20 60 (mS)	ต้องตรวจจับไม่ได้	ผ่าน	-
4	40-10-40 60 (mS)	ต้องตรวจจับได้	ผ่าน	-
5	40-60-40 60 (mS)	ต้องตรวจจับได้ 2 สัญญาณ	ผ่าน	-
6	23 60 (mS)	ต้องตรวจจับไม่ได้	ผ่าน	-
7	40 60 (mS)	ต้องตรวจจับได้	ผ่าน	-
8	40 40 velocity = 80 (mS)	ต้องตรวจจับไม่ได้	ผ่าน	-
9	43 50 velocity = 93 (mS)	ต้องตรวจจับได้	ผ่าน	-

ตารางที่ 10-4 ผลการทดสอบเปลี่ยนแปลงจังหวะสัญญาณ ของการตรวจจับสัญญาณ DTMF

ผลการทดลองปรากฏว่ามีกรณีที่ไม่ผ่านเพียง 2 กรณี โดยเป็นกรณีของการมี Interrupt เข้ามาในส่วนของ Signal Duration เป็นเวลา 10 mS ซึ่งถ้าอิงตามเอกสาร ITU-T Recommendation Q.24 แล้ว ควรจะต้องตรวจจับได้ทุกสัญญาณ แต่ในการทดลองพบว่ามีบางสัญญาณตรวจจับไม่ได้ เนื่องจาก Interrupt ที่เกิดขึ้นทำให้ค่าที่ได้จะทำให้ค่าที่ได้จาก Goertzel Algorithm มีขนาดลดลงจนไม่ผ่านค่า THRESHOLD ที่ตั้งเอาไว้ และกรณีที่สองนั้น คือกรณีของการมี Interrupt เข้ามาในส่วนของ Signal Duration เป็นเวลา 11 mS ซึ่งถ้าอิงตามเอกสาร ITU-T Recommendation Q.24 แล้ว ควรจะต้องตรวจจับไม่ได้ แต่เนื่องจากเป็นค่าที่ก้ำกึ่งระหว่าง 10 และ 11 mS จึงอาจทำให้มีบางสัญญาณถูกตรวจจับได้บ้าง

10.5 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ R2 MF

การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ R2 MF ประกอบด้วยการทดสอบต่างๆ ดังนี้ [1] การทดสอบทั่วไป, และ [2] การทดสอบการเบี่ยงเบนความถี่ (Frequency Deviation)

10.5.1 การทดสอบทั่วไป ของการตรวจจับสัญญาณ R2 MF

ทำการสร้างสัญญาณ R2 MF บน PC โดยใช้ความถี่มาตรฐานตามที่อ้างอิงในบทที่ 2 ซึ่งสัญญาณ R2 MF ที่ทำการสร้างขึ้นนั้นประกอบด้วย หมายเลขดังนี้ ได้แก่ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 (A'),

12 (B'), 13 (C'), 14 (D'), และ 15 (E') ทั้ง Forward, Backward จากนั้นนำสัญญาณที่สร้างขึ้นมาตรวจจับด้วย Library การตรวจจับสัญญาณ R2 MF บน DSP

ผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.5.2 การทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ R2 MF

ทำการสร้างสัญญาณ R2 MF บน PC โดยทำการเบี่ยงเบนความถี่ที่ใช้ไปจากความถี่มาตรฐานที่อ้างอิงในบทที่ 2 จากก็จะนำสัญญาณที่สร้างขึ้น มาทำการตรวจจับด้วย Library การตรวจจับสัญญาณ R2 MF บน DSP

	การเบี่ยงเบนความถี่	ITU-T Q.24	ผลการตรวจจับ	ปัญหาที่พบ
1	±5_a ±0_b	ต้องตรวจจับได้	ผ่าน	-
2	±10_a ±0_b	ต้องตรวจจับได้	ผ่าน	-
3	±0_a ±5_b	ต้องตรวจจับได้	ผ่าน	-
4	±0_a ±10_b	ต้องตรวจจับได้	ผ่าน	-

ตารางที่ 10-5 ผลการทดสอบการเบี่ยงเบนความถี่ ของการตรวจจับสัญญาณ R2 MF

ผลการทดลองปรากฏว่าสามารถตรวจจับสัญญาณ R2 MF ทุกตัวที่ทำการสร้างขึ้นได้ครบถ้วน

10.6 การทดสอบ การทำงานของ Library ตรวจจับสัญญาณ CPT

การทดสอบ CPT นั้นต้องใช้เวลามากเนื่องจาก แต่ละสัญญาณไม่ได้ใช้ Standard เดียวกัน เพราะฉะนั้นจึงจำเป็นต้องแยกการทดสอบเป็นสัญญาณ สัญญาณไป โดยแต่ละสัญญาณจะแบ่งเป็นการทดสอบ 3 หัวข้อ คือ

1. การทดสอบ Frequency (Frequency Testing)
2. การทดสอบ Timing (Timing Testing)
3. การทดสอบ Signal Power (Signal Power Testing)

10.6.1 Dial Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ : 360 and 440 Hz or 400 Hz

Timing: ไม่มี Timing

Amplitude: 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
329,440 Hz	ไม่สามารถตรวจจับได้
330,440 Hz	สามารถตรวจจับได้
350,440 Hz	สามารถตรวจจับได้
394,440 Hz	สามารถตรวจจับได้
394,407 Hz	สามารถตรวจจับได้
394,407 Hz	สามารถตรวจจับได้
330,439 Hz	สามารถตรวจจับได้
330,438 Hz	ไม่สามารถตรวจจับได้
330,407 Hz	ไม่สามารถตรวจจับได้
394,475 Hz	สามารถตรวจจับได้
394,476 Hz	ไม่สามารถตรวจจับได้
329,475 Hz	สามารถตรวจจับได้
360,400 Hz	สามารถตรวจจับได้
360	ไม่สามารถตรวจจับได้
361	สามารถตรวจจับได้
447	สามารถตรวจจับได้
448	ไม่สามารถตรวจจับได้

ตารางที่ 10-6 ผลทดสอบความถี่ของสัญญาณ Dial Tone

เนื่องจากสัญญาณ Dial Tone สามารถสร้างได้ 2 แบบ คือ เกิดจากการผสมระหว่าง 2 ความถี่ คือ 360 กับ 440 Hz หรือ สัญญาณความถี่เดียว ที่ 400 Hz จึงมีการทดสอบความถี่เยอะกว่าสัญญาณ อื่นและในบางกรณี ความถี่ต่ำที่ไม่สามารถตรวจจับได้อย่าง 329 Hz รวมกับความถี่ไม่สูงมากอย่าง 440 Hz ไม่สามารถตรวจจับได้ แต่ ถ้า 329 Hz ไปผสมกับ 475 Hz นั้น 475 Hz จะทำให้ค่า THRESHOLD ของ 329 Hz เพิ่มขึ้นและผ่าน ค่า THRESHOLD ที่อัลกอริทึมได้กำหนดไว้ จึงสามารถตรวจจับได้

การทดสอบ Timing

ไม่ได้ทำการทดสอบเนื่องจากเป็นสัญญาณแบบต่อเนื่อง

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 350,440 Hz

ความถี่ที่ใช้ตรวจจับ = 360,440 Hz or 400 Hz

Timing = ไม่มีทาง Timing

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.53	-5.06 / -4.92	ไม่สามารถตรวจจับได้
0.40	-7.50 / -7.35	สามารถตรวจจับได้
0.30	-10.01 / -9.85	สามารถตรวจจับได้
0.17	-14.94 / -14.78	สามารถตรวจจับได้
0.16	-15.46 / -15.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-7 ผลทดสอบ Power ของสัญญาณ Dial Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
360	100	5,500,000
440	100	5,000,000

ตารางที่ 10-8 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Dial Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=340-425Hz F2=400-450Hz	F1 = 330-393 Hz & F2 = 407-440 Hz
	340-450 Hz	361-447 Hz
Timing	ไม่ได้ทำการทดสอบเนื่องจากเป็นสัญญาณแบบต่อเนื่อง	ไม่ได้ทำการทดสอบเนื่องจากเป็นสัญญาณแบบต่อเนื่อง
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-9 สรุปการทดลองการตรวจจับสัญญาณ Dial Tone

10.6.2 Ring Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 1 Sec OFF: 4 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
408	ไม่สามารถตรวจจับได้
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้
478	ไม่สามารถตรวจจับได้

ตารางที่ 10-10 ผลทดสอบความถี่ของสัญญาณ Ring Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.66 Sec	4 Sec	ไม่สามารถตรวจจับได้
0.67 Sec	4 Sec	สามารถตรวจจับได้
1 Sec	4 Sec	สามารถตรวจจับได้
1.5 Sec	4 Sec	สามารถตรวจจับได้
1.6 Sec	4 Sec	ไม่สามารถตรวจจับได้
1 Sec	5 Sec	สามารถตรวจจับได้
1 Sec	5.1 Sec	ไม่สามารถตรวจจับได้

ตารางที่ 10-11 ผลทดสอบ Timing ของสัญญาณ Ring Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 1 Sec OFF: 4 Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.86 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.10 / -5.05	สามารถตรวจจับได้
0.30	-7.34 / -7.28	สามารถตรวจจับได้
0.14	-14.00 / -13.96	สามารถตรวจจับได้
0.13	-14.61 / -14.57	สามารถตรวจจับได้
0.12	-15.32 / -15.27	ไม่สามารถตรวจจับได้

ตารางที่ 10-12 ผลทดสอบ Power ของสัญญาณ Ring Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-13 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Ring Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	400-450 Hz	410-476 Hz
Timing	ON:0.67-1.5Sec OFF:3-5Sec	ON:0.67-1.5Sec OFF:3-5Sec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-14 สรุปการทดลองการตรวจจับสัญญาณ Ring Tone

10.6.3 Busy Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 500 mSec OFF: 500 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
408	ไม่สามารถตรวจจับได้
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้
478	ไม่สามารถตรวจจับได้

ตารางที่ 10-15 ผลทดสอบความถี่ของสัญญาณ Busy Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
449 mSec	500 mSec	ไม่สามารถตรวจจับได้
450 mSec	500 mSec	สามารถตรวจจับได้
500 mSec	500 mSec	สามารถตรวจจับได้
550 mSec	500 mSec	สามารถตรวจจับได้
551 mSec	500 mSec	ไม่สามารถตรวจจับได้
500 mSec	449 mSec	ไม่สามารถตรวจจับได้
500 mSec	551 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-16 ผลทดสอบ Timing ของสัญญาณ Busy Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.86 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.10 / -5.05	สามารถตรวจจับได้
0.30	-7.34 / -7.28	สามารถตรวจจับได้
0.14	-14.00 / -13.96	สามารถตรวจจับได้
0.13	-14.61 / -14.57	สามารถตรวจจับได้
0.12	-15.32 / -15.27	ไม่สามารถตรวจจับได้

ตารางที่ 10-17 ผลทดสอบ Power ของสัญญาณ Busy Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-18 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Busy Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F=400-450Hz	F = 410-476 Hz
Timing	ON:OFF:500±50mSec	ON:OFF:500±50mSec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-19 สรุปการทดลองการตรวจจับสัญญาณ Busy Tone

10.6.4 Congestion Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON:0.3 Sec OFF: 0.2 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
408	ไม่สามารถตรวจจับได้
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้
478	ไม่สามารถตรวจจับได้

ตารางที่ 10-20 ผลทดสอบความถี่ของสัญญาณ Congestion Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
200 mSec	199 mSec	ไม่สามารถตรวจจับได้
200 mSec	200 mSec	สามารถตรวจจับได้
300 mSec	200 mSec	สามารถตรวจจับได้
400 mSec	200 mSec	สามารถตรวจจับได้
401 mSec	200 mSec	ไม่สามารถตรวจจับได้
400 mSec	400 mSec	สามารถตรวจจับได้
401 mSec	400 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-21 ผลทดสอบ Timing ของสัญญาณ Congestion Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.86 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.10 / -5.05	สามารถตรวจจับได้
0.30	-7.34 / -7.28	สามารถตรวจจับได้
0.14	-14.00 / -13.96	สามารถตรวจจับได้
0.13	-14.61 / -14.57	สามารถตรวจจับได้
0.12	-15.32 / -15.27	ไม่สามารถตรวจจับได้

ตารางที่ 10-22 ผลทดสอบ Power ของสัญญาณ Congestion Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-23 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Congestion Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F=400-450Hz	F = 410-476 Hz
Timing	ON:OFF:500±50mSec	ON:OFF:500±50mSec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-24 สรุปการทดลองการตรวจจับสัญญาณ Congestion Tone

10.6.5 Special Information

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 960 Hz

Timing = ON: 330 mSec OFF: 30 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
897	ไม่สามารถตรวจจับได้
898	สามารถตรวจจับได้
950	สามารถตรวจจับได้
1024	สามารถตรวจจับได้
1025	ไม่สามารถตรวจจับได้

ตารางที่ 10-25 ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 960 Hz

ความถี่ที่ใช้ตรวจจับ = 1400 Hz

Timing = ON: 330 mSec OFF: 30 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
1349	ไม่สามารถตรวจจับได้
1350	สามารถตรวจจับได้
1400	สามารถตรวจจับได้
1452	สามารถตรวจจับได้
1453	ไม่สามารถตรวจจับได้

ตารางที่ 10-26 ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 1400 Hz

ความถี่ที่ใช้ตรวจจับ = 1800 Hz

Timing = ON: 330 mSec OFF: 1 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
1745	ไม่สามารถตรวจจับได้

1746	สามารถตรวจจับได้
1800	สามารถตรวจจับได้
1855	สามารถตรวจจับได้
1856	ไม่สามารถตรวจจับได้

ตารางที่ 10-27 ผลทดสอบความถี่ของสัญญาณ Special Information ที่ความถี่ 1800 Hz

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

First Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
259 mSec	30 mSec	ไม่สามารถตรวจจับได้
260 mSec	30 mSec	สามารถตรวจจับได้
400 mSec	30 mSec	สามารถตรวจจับได้
330 mSec	30 mSec	สามารถตรวจจับได้
401 mSec	30 mSec	ไม่สามารถตรวจจับได้
330 mSec	24 mSec	ไม่สามารถตรวจจับได้
330 mSec	25 mSec	สามารถตรวจจับได้
330 mSec	35 mSec	สามารถตรวจจับได้
330 mSec	36 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-28 ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 1

Second Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
259 mSec	30 mSec	ไม่สามารถตรวจจับได้
260 mSec	30 mSec	สามารถตรวจจับได้
400 mSec	30 mSec	สามารถตรวจจับได้

330 mSec	30 mSec	สามารถตรวจจับได้
401 mSec	30 mSec	ไม่สามารถตรวจจับได้
330 mSec	24 mSec	ไม่สามารถตรวจจับได้
330 mSec	25 mSec	สามารถตรวจจับได้
330 mSec	35 mSec	สามารถตรวจจับได้
330 mSec	36 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-29 ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 2

Third Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
259 mSec	30 mSec	ไม่สามารถตรวจจับได้
260 mSec	30 mSec	สามารถตรวจจับได้
400 mSec	30 mSec	สามารถตรวจจับได้
330 mSec	30 mSec	สามารถตรวจจับได้
401 mSec	30 mSec	ไม่สามารถตรวจจับได้
330 mSec	24 mSec	ไม่สามารถตรวจจับได้
330 mSec	25 mSec	สามารถตรวจจับได้
330 mSec	35 mSec	สามารถตรวจจับได้
330 mSec	36 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-30 ผลทดสอบ Timing ของสัญญาณ Special Information ช่วงที่ 3

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 950,1400,1800 Hz

ความถี่ที่ใช้ตรวจจับ = 960,1400,1800 Hz

Timing = ON:330,330,330 mSec OFF: 30,30,1000 mSec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.80 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.05 / -5.05	สามารถตรวจจับได้
0.30	-7.28 / -7.30	สามารถตรวจจับได้

0.13	-14.61 / -14.56	สามารถตรวจจับได้
0.12	-15.27 / -14.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-31 ผลทดลอบ Power ของสัญญาณ Special Information

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
960	100	600,000
1400	100	2,000,000
1800	100	1,500,000

ตารางที่ 10-32 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Special Information

Frequency	F1=950±50Hz, F2=1400±50Hz, F3=1800±50Hz	F1=898-1024Hz F2=1350-1452Hz F3=1746-1855Hz
Timing	ON:330±70Sec, OFF:30±5msec	ON:330±70Sec, OFF:30±5msec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-33 สรุปการทดลองการตรวจจับสัญญาณ Special Information

10.6.6 Warning Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 1400 Hz

Timing = ON: 425 mSec OFF: 15 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
1349	ไม่สามารถตรวจจับได้
1350	สามารถตรวจจับได้
1400	สามารถตรวจจับได้
1452	สามารถตรวจจับได้
1453	ไม่สามารถตรวจจับได้
1349	ไม่สามารถตรวจจับได้

ตารางที่ 10-34 ผลทดสอบความถี่ของสัญญาณ Warning Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 1400 Hz

ความถี่ที่ใช้ตรวจจับ = 1400 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
259 mSec	15 Sec	ไม่สามารถตรวจจับได้
260 mSec	15 Sec	สามารถตรวจจับได้
330 mSec	15 Sec	สามารถตรวจจับได้
300 mSec	15 Sec	สามารถตรวจจับได้
301 mSec	15 Sec	ไม่สามารถตรวจจับได้
330 mSec	11.99 Sec	ไม่สามารถตรวจจับได้
330 mSec	12 Sec	สามารถตรวจจับได้
330 mSec	15 Sec	สามารถตรวจจับได้
330 mSec	18 Sec	สามารถตรวจจับได้
330 mSec	18.01 Sec	ไม่สามารถตรวจจับได้

ตารางที่ 10-35 ผลทดสอบ Timing ของสัญญาณ Warning Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 1400 Hz

ความถี่ที่ใช้ตรวจจับ = 1400 Hz

Timing = ON: 425 mSec OFF: 15 Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.80 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.05 / -5.05	สามารถตรวจจับได้
0.30	-7.28 / -7.30	สามารถตรวจจับได้
0.13	-14.61 / -14.56	สามารถตรวจจับได้
0.12	-15.27 / -14.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-36 ผลทดสอบ Power ของสัญญาณ Warning Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
1400	100	2,000,000

ตารางที่ 10-37 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Warning Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F=1379-1421Hz	F=1350-1452Hz
Timing	ON:350-500ms, OFF:12-18Sec	ON:350-500ms, OFF:12-18Sec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-38 สรุปการทดลองการตรวจจับสัญญาณ Warning Tone

10.6.7 Payphone Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 1400 Hz

Timing =ON: 200 mSec OFF: 200 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
1098	ไม่สามารถตรวจจับได้
1099	สามารถตรวจจับได้
1100	สามารถตรวจจับได้
1710	สามารถตรวจจับได้
1711	ไม่สามารถตรวจจับได้

ตารางที่ 10-39 ผลทดสอบความถี่ของสัญญาณ Payphone Tone ที่ 1400 Hz

ความถี่ที่ใช้ตรวจจับ = 1100 Hz

Timing = ON: 200 mSec OFF: 2 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
735	ไม่สามารถตรวจจับได้
736	สามารถตรวจจับได้
1100	สามารถตรวจจับได้
1491	สามารถตรวจจับได้
1492	ไม่สามารถตรวจจับได้

ตารางที่ 10-40 ผลทดสอบความถี่ของสัญญาณ Payphone Tone ที่ 1100 Hz

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 1400, 1100 Hz

ความถี่ที่ใช้ตรวจจับ = 1400, 1100 Hz

Amplitude = 0.3

First Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
149 mSec	200 mSec	ไม่สามารถตรวจจับได้
150 mSec	200 mSec	สามารถตรวจจับได้
200 mSec	200 mSec	สามารถตรวจจับได้
250 mSec	200 mSec	สามารถตรวจจับได้
251 mSec	200 mSec	ไม่สามารถตรวจจับได้
200 mSec	149 mSec	ไม่สามารถตรวจจับได้
200 mSec	150 mSec	สามารถตรวจจับได้
200 mSec	200 mSec	สามารถตรวจจับได้
200 mSec	250 mSec	สามารถตรวจจับได้
200 mSec	251 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-41 ผลทดสอบ Timing ของสัญญาณ Payphone Tone ช่วงที่ 1

Second Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
149 mSec	2000 mSec	ไม่สามารถตรวจจับได้
150 mSec	2000 mSec	สามารถตรวจจับได้
200 mSec	2000 mSec	สามารถตรวจจับได้
250 mSec	2000 mSec	สามารถตรวจจับได้
251 mSec	2000 mSec	ไม่สามารถตรวจจับได้
200 mSec	1799 mSec	ไม่สามารถตรวจจับได้
200 mSec	1800 mSec	สามารถตรวจจับได้
200 mSec	2000 mSec	สามารถตรวจจับได้
200 mSec	2200 mSec	สามารถตรวจจับได้
200 mSec	2201 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-42 ผลทดสอบ Timing ของสัญญาณ Payphone Tone ช่วงที่ 2

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 1400, 1100 Hz

ความถี่ที่ใช้ตรวจจับ = 1400, 11000 Hz

Timing = ON: 200,200 mSec OFF: 200, 2000 mSec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.71	0.14 / 0.14	ไม่สามารถตรวจจับได้
0.70	0.07 / 0.00	สามารถตรวจจับได้
0.30	-7.32 / -7.33	สามารถตรวจจับได้
0.08	-18.79 / -18.85	สามารถตรวจจับได้
0.07	-19.99 / -20.01	ไม่สามารถตรวจจับได้

ตารางที่ 10-43 ผลทดสอบ Power ของสัญญาณ Payphone Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
1400	20	40,000
1100	16	20,000

ตารางที่ 10-44 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Payphone Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=1100-1750Hz F2=750-1450Hz	F1=1099-1710Hz F2=736-1491Hz
Timing	ON1: 200, ON2: 200 mSec OFF1: 200, OFF2: 2000 mSec	ON1: 200±50, ON2: 200±50 mSec OFF1: 200±50, OFF2: 2000±200 mSec
Signal Power(dBm)	-0 to -20	-0 to -20

ตารางที่ 10-45 สรุปการทดลองการตรวจจับสัญญาณ Payphone Tone

เนื่องจาก มาตรฐานเกี่ยวกับTiming ของ Payphone Tone นั้นมีข้อจำกัดมากเกินไป อัลกอริทึมที่พัฒนาขึ้นจะยืดหยุ่นจากมตราฐานอีกเล็กน้อย

10.6.8 Call Waiting

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 300 mSec OFF: 9 Sec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
408	ไม่สามารถตรวจจับได้
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้

ตารางที่ 10-46 ผลทดสอบความถี่ของสัญญาณ Call Waiting

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
299 Sec	9 Sec	ไม่สามารถตรวจจับได้
300 Sec	9 Sec	สามารถตรวจจับได้
499 Sec	9 Sec	สามารถตรวจจับได้
500 Sec	9 Sec	สามารถตรวจจับได้
300 Sec	7.99 Sec	ไม่สามารถตรวจจับได้
300 Sec	8 Sec	สามารถตรวจจับได้
300 Sec	10 Sec	สามารถตรวจจับได้
300 Sec	10.1 Sec	ไม่สามารถตรวจจับได้

ตารางที่ 10-47 ผลทดสอบ Timing ของสัญญาณ Call Waiting

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.40	-4.86 / -4.81	ไม่สามารถตรวจจับได้
0.39	-5.10 / -5.05	สามารถตรวจจับได้
0.30	-7.34 / -7.28	สามารถตรวจจับได้
0.14	-14.00 / -13.96	สามารถตรวจจับได้
0.13	-14.61 / -14.57	สามารถตรวจจับได้
0.12	-15.32 / -15.27	ไม่สามารถตรวจจับได้

ตารางที่ 10-48 ผลทดสอบ Power ของสัญญาณ Call Waiting

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-49 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Call Waiting

	มาตรฐาน CPT	ที่ทำได้
Frequency	F=400-450Hz	F=410-476 Hz
Timing	ON:300-500ms,OFF:8-10Sec	ON:300-500ms,OFF:8-10Sec
Signal Power(dBm)	-5 to -15	-5 to -15

ตารางที่ 10-50 สรุปการทดลองการตรวจจับสัญญาณ Call Waiting

10.6.9 Recall Dial Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 360, 440 Hz

Timing = ON: 100 OFF:100 Sec Follow by Dial Tone

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
329,440 Hz	ไม่สามารถตรวจจับได้
330,440 Hz	สามารถตรวจจับได้
350,440 Hz	สามารถตรวจจับได้
393,440 Hz	สามารถตรวจจับได้
393,460 Hz	สามารถตรวจจับได้
330,460 Hz	สามารถตรวจจับได้
393,461 Hz	ไม่สามารถตรวจจับได้
330,463 Hz	สามารถตรวจจับได้
330,464 Hz	ไม่สามารถตรวจจับได้
330,439 Hz	สามารถตรวจจับได้
330,438 Hz	ไม่สามารถตรวจจับได้
329,460 Hz	ไม่สามารถตรวจจับได้
360,400 Hz	สามารถตรวจจับได้

ตารางที่ 10-51 ผลทดสอบความถี่ของสัญญาณ Recall Dial Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 350, 440 Hz

ความถี่ที่ใช้ตรวจจับ = 360, 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.66 Sec	4 Sec	ไม่สามารถตรวจจับได้
0.67 Sec	4 Sec	สามารถตรวจจับได้
1 Sec	4 Sec	สามารถตรวจจับได้

1.5 Sec	4 Sec	สามารถตรวจจับได้
1.6 Sec	4 Sec	ไม่สามารถตรวจจับได้
1 Sec	5 Sec	สามารถตรวจจับได้
1 Sec	5.1 Sec	ไม่สามารถตรวจจับได้

ตารางที่ 10-52 ผลทดสอบ Timing ของสัญญาณ Recall Dial Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 350, 440 Hz

ความถี่ที่ใช้ตรวจจับ = 360, 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	**Power T/R (dBm)**	**ผลการตรวจจับโดยใช้ CPT Detection Algorithm**
0.94	-0.09 / 0.08	ไม่สามารถตรวจจับได้
0.93	-0.19 / -0.02	สามารถตรวจจับได้
0.30	-10.01 / -9.85	สามารถตรวจจับได้
0.17	-14.94 / -14.78	สามารถตรวจจับได้
0.16	-15.46 / -15.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-53 ผลทดสอบ Power ของสัญญาณ Recall Dial Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	**จำนวน Sample ที่ใช้คำนวณ Goertzel**	**THRESHOLD**
360	100	5,500,000
440	100	5,000,000

ตารางที่ 10-54 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Recall Dial Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=340-425Hz F2=400-450Hz	F1=330-393Hz F2=400-460Hz
Timing	ON and OFF: 100±20ms followed by Dial Tone	ON and OFF: 100±20ms followed by Dial Tone
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-55 สรุปการทดลองการตรวจจับสัญญาณ Recall Dial Tone

10.6.10 Intercept Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440, 620 Hz

Timing = ON:230 mSec Total: 500 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้างสัญญาณ (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
406, 620 Hz	ไม่สามารถตรวจจับได้
407, 620 Hz	สามารถตรวจจับได้
440, 620 Hz	สามารถตรวจจับได้
480, 620 Hz	สามารถตรวจจับได้
481, 620 Hz	ไม่สามารถตรวจจับได้
440, 562 Hz	ไม่สามารถตรวจจับได้
440, 563 Hz	สามารถตรวจจับได้
440, 684 Hz	สามารถตรวจจับได้
440, 685 Hz	ไม่สามารถตรวจจับได้
407, 684 Hz	สามารถตรวจจับได้

ตารางที่ 10-56 ผลทดสอบความถี่ของสัญญาณ Intercept Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 440, 620 Hz

ความถี่ที่ใช้ตรวจจับ = 440, 620 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
159 mSec	270 mSec	ไม่สามารถตรวจจับได้
160 mSec	270 mSec	ไม่สามารถตรวจจับได้
180 mSec	270 mSec	สามารถตรวจจับได้
230 mSec	270 mSec	สามารถตรวจจับได้
280 mSec	270 mSec	สามารถตรวจจับได้
281 mSec	270 mSec	ไม่สามารถตรวจจับได้

301 mSec	249 mSec	ไม่สามารถตรวจจับได้
300 mSec	250 mSec	สามารถตรวจจับได้
300 mSec	249 mSec	ไม่สามารถตรวจจับได้
219 mSec	230 mSec	ไม่สามารถตรวจจับได้
220 mSec	230 mSec	สามารถตรวจจับได้

ตารางที่ 10-57 ผลทดสอบ Timing ของสัญญาณ Intercept Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 440, 620 Hz

ความถี่ที่ใช้ตรวจจับ = 440, 620 Hz

Timing = ON:230 mSec Total: 500 mSec

Amplitude	**Power T/R (dBm)**	**ผลการตรวจจับโดยใช้ CPT Detection Algorithm**
0.70	0.04 / 0.03	ไม่สามารถตรวจจับได้
0.69	-0.08 / -0.08	สามารถตรวจจับได้
0.30	-7.30 / -7.30	สามารถตรวจจับได้
0.14	-13.96 / -13.96	สามารถตรวจจับได้
0.13	-14.57 / -14.57	สามารถตรวจจับได้
0.12	-15.27 / -15.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-58 ผลทดสอบ Power ของสัญญาณ Intercept Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	**จำนวน Sample ที่ใช้คำนวณ Goertzel**	**THRESHOLD**
440	100	4,500,000
620	100	5,000,000

ตารางที่ 10-59 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Intercept Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=440Hz F2=620Hz	F1=407-480HZ F2=563-620Hz
Timing	ON:230±70ms Total:500±50ms	ON:230±70ms Total:500±50ms
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-60 สรุปการทดลองการตรวจจับสัญญาณ Intercept Tone

จะเห็นว่าในมาตรฐานระบุแค่ให้ตรวจจับ 440 Hz ผสม 620 Hz เท่านั้นซึ่งดูแล้วมันใช้งานไม่ได้แน่ อัลกอริทึม ที่พัฒนาขึ้นมานี้จึงพยายามยืดหยุ่นความถี่ที่จะตรวจจับออกไปอีกเล็กน้อย

10.6.11 Confirmation Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440, 360 Hz

Timing = ON: 100 mSec OFF: 100 mSec

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
329,440 Hz	ไม่สามารถตรวจจับได้
330,440 Hz	สามารถตรวจจับได้
350,440 Hz	สามารถตรวจจับได้
393,440 Hz	สามารถตรวจจับได้
393,460 Hz	สามารถตรวจจับได้
330,460 Hz	สามารถตรวจจับได้
393,461 Hz	ไม่สามารถตรวจจับได้
330,463 Hz	สามารถตรวจจับได้
330,464 Hz	ไม่สามารถตรวจจับได้
330,439 Hz	สามารถตรวจจับได้
330,438 Hz	ไม่สามารถตรวจจับได้
329,460 Hz	ไม่สามารถตรวจจับได้
360,400 Hz	สามารถตรวจจับได้

ตารางที่ 10-61 ผลทดสอบความถี่ของสัญญาณ Confirmation Tone

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 440, 350 Hz

ความถี่ที่ใช้ตรวจจับ = 440, 360 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
79 mSec	100 mSec	ไม่สามารถตรวจจับได้
80 mSec	100 mSec	สามารถตรวจจับได้
100 mSec	100 mSec	สามารถตรวจจับได้

120 mSec	100 mSec	สามารถตรวจจับได้
121 mSec	100 mSec	ไม่สามารถตรวจจับได้
100 mSec	79 mSec	ไม่สามารถตรวจจับได้
100 mSec	80 mSec	สามารถตรวจจับได้
100 mSec	100 mSec	สามารถตรวจจับได้
100 mSec	120 mSec	สามารถตรวจจับได้
100 mSec	121 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-62 ผลทดสอบ Timing ของสัญญาณ Confirmation Tone

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	**Power T/R (dBm)**	**ผลการตรวจจับโดยใช้ CPT Detection Algorithm**
0.94	-0.09 / 0.08	ไม่สามารถตรวจจับได้
0.93	-0.19 / -0.02	สามารถตรวจจับได้
0.30	-10.01 / -9.85	สามารถตรวจจับได้
0.17	-14.94 / -14.78	สามารถตรวจจับได้
0.16	-15.46 / -15.28	ไม่สามารถตรวจจับได้

ตารางที่ 10-63 ผลทดสอบ Power ของสัญญาณ Confirmation Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	**จำนวน Sample ที่ใช้คำนวณ Goertzel**	**THRESHOLD**
360	100	5,500,000
440	100	5,000,000

ตารางที่ 10-64 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Confirmation Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=340-425Hz F2=400-450Hz	F1=330-393Hz F2=400-460Hz
Timing	ON:100ms, OFF:100ms	ON and OFF: 100±20ms
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-65 สรุปการทดลองการตรวจจับสัญญาณ Confirmation Tone

เนื่องจากตามมาตรฐานนั้น Timing ของสัญญาณ Confirmation นั้นไม่ได้ระบุมาชัดเจนเกินไปซึ่งไม่สามารถนำเอาไปใช้งานได้ อัลกอริทึมที่พัฒนาขึ้นมานี้จึงกำหนดให้การตรวจสอบ Timing เป็นไปตามมาตรฐาน Recall Dial Tone เนื่องจาก 2 สัญญาณนี่มีลักษณะเหมือนกันเกือบทั้งหมด Recall Dial Tone เกิดจากการนำเอาสัญญาณ Confirmation Tone มาต่อท้ายด้วย Dial Tone

10.6.12 Comfort Tone

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 480 Hz

Timing = Continuous

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
451	ไม่สามารถตรวจจับได้
452	สามารถตรวจจับได้
440	สามารถตรวจจับได้
480	สามารถตรวจจับได้
500	สามารถตรวจจับได้
514	สามารถตรวจจับได้
515	ไม่สามารถตรวจจับได้

ตารางที่ 10-66 ผลทดสอบความถี่ของสัญญาณ Comfort Tone

การทดสอบ Timing

ไม่ได้ทดสอบ Timing เนื่องจากเป็นสัญญาณแบบต่อเนื่อง

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON: 0.67-1.5Sec OFF: 3-5Sec

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.70	0.06 / 0.09	ไม่สามารถตรวจจับได้
0.69	-0.07 / -0.03	สามารถตรวจจับได้
0.30	-7.29 / -7.26	สามารถตรวจจับได้
0.14	-13.95 / -13.93	สามารถตรวจจับได้
0.13	-14.55 / -14.54	สามารถตรวจจับได้
0.12	-15.27 / -15.25	ไม่สามารถตรวจจับได้

ตารางที่ 10-67 ผลทดสอบ Timing ของสัญญาณ Comfort Tone

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-68 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Comfort Tone

	มาตรฐาน CPT	ที่ทำได้
Frequency	F=480 Hz	F=452-514Hz
Timing	-	ไม่ได้ทำการทดสอบเนื่องจากเป็นสัญญาณแบบต่อเนื่อง
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-69 สรุปการทดลองการตรวจจับสัญญาณ Comfort Tone

จะเห็นว่าในมาตรฐานระบุแค่ให้ตรวจจับ 480 Hz เท่านั้นซึ่งดูแล้วมันใช้งานไม่ได้แน่ อัลกอริทึม ที่พัฒนาขึ้นมานี้จึงพยายามยืดหยุ่นความถี่ที่จะตรวจจับออกไปอีกเล็กน้อย

10.6.13 Busy Verification

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON1: 1.5 s, OFF1: 9 s ON2: 300 ms, OFF2: 9 s

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้

ตารางที่ 10-70 ผลทดสอบความถี่ของสัญญาณ Busy Verification

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 440 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

First Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
999 mSec	9000 mSec	ไม่สามารถตรวจจับได้
1000 mSec	9000 mSec	สามารถตรวจจับได้
1500 mSec	9000 mSec	สามารถตรวจจับได้
1501 mSec	9000 mSec	ไม่สามารถตรวจจับได้
1000 mSec	7499 mSec	ไม่สามารถตรวจจับได้
1000 mSec	7500 mSec	สามารถตรวจจับได้
1000 mSec	10000 mSec	สามารถตรวจจับได้
1000 mSec	10001 mSec	สามารถตรวจจับได้

ตารางที่ 10-71 ผลทดสอบ Timing ของสัญญาณ Busy Verification ช่วงที่ 1

Second Tone and Pause Duration

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
199 mSec	9000 mSec	ไม่สามารถตรวจจับได้
200 mSec	9000 mSec	สามารถตรวจจับได้
300 mSec	9000 mSec	สามารถตรวจจับได้
301 mSec	9000 mSec	ไม่สามารถตรวจจับได้
300 mSec	7499 mSec	ไม่สามารถตรวจจับได้
300 mSec	7500 mSec	สามารถตรวจจับได้
300 mSec	10000 mSec	สามารถตรวจจับได้
300 mSec	10001 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-72 ผลทดสอบ Timing ของสัญญาณ Busy Verification ช่วงที่ 1

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 425 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = ON1: 1.5 s, OFF1: 9 s ON2: 300 ms, OFF2: 9 s

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.70	0.06 / 0.07	ไม่สามารถตรวจจับได้
0.69	0.07 / -0.05	สามารถตรวจจับได้
0.30	-7.28 / -7.28	สามารถตรวจจับได้
0.14	-13.94 / -13.92	สามารถตรวจจับได้
0.13	-14.54 / -14.53	สามารถตรวจจับได้
0.12	-1526 / -15.25	ไม่สามารถตรวจจับได้

ตารางที่ 10-73 ผลทดสอบ Power ของสัญญาณ Busy Verification

สรุปผลการทดลอง

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-74 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Busy Verification

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=440Hz	F=410~476 Hz
Timing	ON1: 1-1.5s, ON2 .2-.3 Sec, OFF 7.5-10 Sec	ON1: 1-1.5s, ON2 .2-.3 Sec, OFF 7.5-10 Sec
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-75 สรุปการทดลองการตรวจจับสัญญาณ Busy Verification

จะเห็นว่าในมาตรฐานระบุแค่ให้ตรวจจับ 440 Hz เท่านั้นซึ่งดูแล้วมันใช้งานไม่ได้แน่ อัลกอริทึม ที่พัฒนาขึ้นมานี้จึงพยายามยืดหยุ่นความถี่ที่จะตรวจจับออกไปอีกเล็กน้อย

10.6.14 Caller Waiting

การตรวจสอบความถี่

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = Ring (ON: 1000 ms +OFF: 200 ms) +Call waiting (2cycles)

Amplitude = 0.3

ความถี่ที่ใช้สร้าง (Hz)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
408	ไม่สามารถตรวจจับได้
409	ไม่สามารถตรวจจับได้
410	สามารถตรวจจับได้
425	สามารถตรวจจับได้
460	สามารถตรวจจับได้
476	สามารถตรวจจับได้
477	ไม่สามารถตรวจจับได้
478	ไม่สามารถตรวจจับได้

ตารางที่ 10-76 ผลทดสอบความถี่ของสัญญาณ Caller Waiting

การทดสอบ Timing

ความถี่ที่ใช้สร้าง = 440 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Amplitude = 0.3

ON	OFF	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
899 mSec	200 mSec	ไม่สามารถตรวจจับได้
900 mSec	200 mSec	สามารถตรวจจับได้
1000 mSec	200 mSec	สามารถตรวจจับได้
1100 mSec	200 mSec	สามารถตรวจจับได้
1101 mSec	200 mSec	ไม่สามารถตรวจจับได้
1000 mSec	179 mSec	ไม่สามารถตรวจจับได้
1000 mSec	180 mSec	สามารถตรวจจับได้

บรรณานุกรม

[1] Xiangdong Fu, and Zhaohong Zhang (2000) : "A Multichannel/Algorithm Implementation on the TMS320C6000 DSP" – SPRA556B , "Texas Instruments Inc.", February 2000.

[2] Murat Karaorman (2002) : "Techniques for Implementing Shared Relocatable Buffers Using the TMS320 DSP Algorithm Standard" – SPRA790 , "Texas Instruments Inc.", October 2002.

[3] Stig Torud (2002) : "A Technical Overview of eXpressDSP-Compliant Algorithms for DSP Software Producers" – SPRA579C , "Texas Instruments Inc.", September 2002.

[4] Texas Instruments Inc. : "TMS320 DSP Algorithm Standard Rules and Guidelines" – SPRU352E , "Texas Instruments Inc.", October 2002.

[5] Texas Instruments Inc. : "TMS320 DSP Algorithm Standard API Reference" – SPRU360C , "Texas Instruments Inc.", September 2002.

[6] International Telecommunication Union : "Multifrequency Push-Button Signal Reception" – ITU-T Recommendation Q.24 , "International Telecommunication Union".

[7] International Telecommunication Union : "The Receiving Part of the Multifrequency Equipment" – ITU-T Recommendation Q.455 , "International Telecommunication Union".

[8] International Telecommunication Union : "Technical Characteristics of Tones for the Telephone Service" – ITU-T Recommendation E.180/Q.35 , "International Telecommunication Union".

[9] Geoff Probert : "Compute Signal Power with Fixed-Point Calculators", http://www.embedded.com, September 2003.

1000 mSec	220 mSec	สามารถตรวจจับได้
1000 mSec	221 mSec	ไม่สามารถตรวจจับได้

ตารางที่ 10-77 ผลทดสอบ Timing ของสัญญาณ Caller Waiting

การทดสอบ Power ของสัญญาณ

ความถี่ที่ใช้สร้าง = 440 Hz

ความถี่ที่ใช้ตรวจจับ = 440 Hz

Timing = Ring (ON: 1000 ms +OFF: 200 ms) +Call waiting (2cycles)

Amplitude	Power T/R (dBm)	ผลการตรวจจับโดยใช้ CPT Detection Algorithm
0.70	0.01 / 0.05	ไม่สามารถตรวจจับได้
0.69	-0.12 / -0.06	สามารถตรวจจับได้
0.30	-7.34 / -7.28	สามารถตรวจจับได้
0.14	-14.00 / -13.96	สามารถตรวจจับได้
0.13	-14.61 / -14.57	สามารถตรวจจับได้
0.12	-15.32 / -15.27	ไม่สามารถตรวจจับได้

ตารางที่ 10-78 ผลทดสอบ Power ของสัญญาณ Caller Waiting

สรุปผลการทดลอง.

ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
440	100	9,500,000

ตารางที่ 10-79 ค่าต่างๆที่ใช้ในการตรวจจับสัญญาณ Caller Waiting

	มาตรฐาน CPT	ที่ทำได้
Frequency	F1=400-450Hz	F = 410-476 Hz
Timing	ON1: 1000 ms OFF1: 0-200 ms ON2: Call Waiting	ON1: 900-1100 ms OFF1: 180-220 ms ON2: Call Waiting
Signal Power(dBm)	0 to -15	0 to -15

ตารางที่ 10-80 สรุปการทดลองการตรวจจับสัญญาณ Caller Waiting

ตามมาตรฐานนั้น Timing ของสัญญาณ Caller Waiting นั้นดูไม่น่าใช้งานได้ ไม่สามารถนำเอาไปใช้งานได้ อัลกอริทึมที่พัฒนาขึ้นมานี้จึงกำหนดให้การตรวจสอบ Timing ขึ้นมาใหม่เองให้ยืดหยุ่นจากของเดิมเล็กน้อย

10.6.15 สรุปผลการทดลอง CPT

สรุปผลการทดสอบความถี่

ID	Tone Name	ความถี่ที่มาตรฐาน CPT ระบุให้ตรวจจับได้	ความถี่ที่อัลกอริทึมที่พัฒนาขึ้นมานี้สามารถตรวจจับได้	ความถี่ที่ใช้ตรวจจับ(Hz)	จำนวน Sample ที่ใช้คำนวณ Goertzel	THRESHOLD
0	Dial tone	F1=340-425Hz F2=400-450Hz	F1=330-393Hz F2=407-440Hz	360 440	100 100	5,500,000 5,000,000
		F=340-450 Hz	F=330-476Hz	400	40	15,000,000
1	Ring Tone	F=400-450Hz	F=410-476Hz	440	100	9,500,000
2	Busy Tone	F=400-450Hz	F=410-476Hz	440	100	9,500,000
3	Congestion	F=400-450Hz	F=410-476HZ	440	100	9,500,000
4	Special Information	F1=950±50Hz F2=1400±50Hz F3=1800±50Hz	F1=898-1024Hz F2=1350-1452Hz F3=1746-1855Hz	960 1400 1800	100 100 100	600,000 2,000,000 1,500,000
5	Warning Tone	F=1379-1421Hz	F=1350-1452Hz	1400	100	2,000,000
6	Payphone Tone	F1=1100-1750Hz F2=750-1450Hz	F1=1099-1710Hz F2=736-1491Hz	1400 1100	20 16	40,000 20,000
7	Call Waiting	F=400-450Hz	F=410-476HZ	440	100	9,500,000
8	Recall Dial Tone	F1=340-425Hz F2=400-450Hz	F1=330-393Hz F2=400-460Hz	360 440	100 100	5,500,000 9,500,000
9	Intercept Tone	F1=440HZ F2=620Hz	F1=407-480HZ F2=563-684Hz	440 620	100 100	4,500,000 5,000,000
10	Confirmation	F1=340-425Hz F2=400-450Hz	F1=330-393Hz F2=400-460Hz	360 440	100 100	5,500,000 9,500,000
11	Comfort	Tone F1=480Hz	F=452-514Hz	480	100	10,000,000
12	Busy Verification	F1=440Hz	F=410-476HZ	440	100	9,500,000
13	Caller Waiting	F1=400-450Hz	F=410-476HZ	440	100	9,500,000

ตารางที่ 10-81 สรุปผลการทดสอบความถี่ของ CPT

* ช่องที่ Highlight ด้วยสีเทาคือ อัลกอริทึมที่พัฒนามีความยืดหยุ่นมากกว่ามาตรฐานเนื่องจากมาตรฐานมีข้อจำกัดมากเกินไปไม่สามารถนำไปใช้งานจริงๆ ได้

สรุปผลการทดสอบ Timing

Tone Name	มาตรฐาน Timing ของ CPT	ที่อัลกอริทึมสามารถตรวจจับได้
Dial tone	Continuous Tone	Continuous Tone
Ring Tone	ON:0.67-1.5Sec, OFF:3-5Sec	ON:0.67-1.5Sec,OFF:3-5Sec
Busy Tone	ON:OFF:(500±50mSec)	ON:OFF:(500±50mSec)
Congestion	ON:0.2-0.4sec, Total:0.4-0.8(Sec)	ON:0.2-0.4sec,Total:0.4-0.8(Sec)
Special Information	ON:330±70Sec, OFF:30msec	ON:330±70Sec, OFF:30msec
Warning Tone	ON:350-500ms, OFF:12-18Sec	ON:350-500ms, OFF:12-18Sec
Payphone Tone	ON1: 200, ON2: 200mSec OFF1: 200, OFF2: 2000mSec	ON1: 200±50, ON2: 200±50mSec OFF1: 200±50, OFF2: 2000±200mSec
Call Waiting	ON:300-500ms, OFF:8-10Sec	ON:300-500ms, OFF:8-10Sec
Recall Dial Tone	ON and OFF: 100±20ms followed by Dial Tone	ON and OFF: 100±20ms followed by Dial Tone
Intercept Tone	ON:230±70ms Total:500±50ms	ON:230±70ms Total:500±50ms
Confirmation	ON:100ms, OFF:100ms	ON and OFF: 100±20ms
Comfort	Continuous	Continuous
Busy Verification	ON1 1-1.5s, ON2 0.2-0.3Sec OFF 7.5-10Sec	ON1 1-1.5s, ON2 0.2-0.3Sec OFF 7.5-10Sec
Caller Waiting	ON1: 1000 ms OFF1: 0-200 ms ON2: Call Waiting	ON1: 900-1100 ms OFF1: 180-220 ms ON2: Call Waiting

ตารางที่ 10-82 สรุปผลการทดสอบ Timing ของ CPT

* ช่องที่ Highlight ด้วยสีเทาคือ อัลกอริทึมที่พัฒนามีความยืดหยุ่นมากกว่ามาตรฐานเนื่องจากมาตรฐานมีข้อจำกัดมากเกินไปไม่สามารถนำไปใช้งานจริงๆ ได้

สรุปผลการทดสอบ Power ของสัญญาณ

Tone Name	มาตรฐาน Power ของ CPT (dBm)	ที่อัลกอริทึมสามารถตรวจจับได้ (dBm)	ค่า Factor สำหรับ Amplitude ที่ใช้ได้
Dial tone	-5 to -15	-5 to -15	0.17 – 0.52
Ring Tone	-5 to -15	-5 to -15	0.13 – 0.39
Busy Tone	-5 to -15	-5 to -15	0.13 – 0.39
Congestion	-5 to -15	-5 to -15	0.13 – 0.39
Special Information	-5 to -15	-5 to -15	0.13 – 0.39
Warning Tone	-5 to -15	-5 to -15	0.13 – 0.39
Payphone Tone	0 to -20	0 to -20	0.08 – 0.70
Call Waiting	-5 to -15	-5 to -15	0.13 – 0.39
Recall Dial Tone	0 to -15	0 to -15	0.17 – 0.93
Intercept Tone	0 to -15	0 to -15	0.13 – 0.69
Confirmation	0 to -15	0 to -15	0.17 – 0.93
Comfort	0 to -15	0 to -15	0.13 – 0.69
Busy Verification	0 to -15	0 to -15	0.13 – 0.69
Caller Waiting	0 to -15	0 to -15	0.13 – 0.69

ตารางที่ 10-83 สรุปผลการทดสอบ Power ของสัญญาณของ CPT

บทที่ 11
บทวิจารณ์และสรุป

11.1 เกี่ยวกับ Algorithm Standard

หากจะว่าไปแล้ว หลักการของ Algorithm Standard นั้น ก็คือการพยายามที่จะทำการพัฒนาอัลกอริทึมต่าง ให้อยู่ในรูปแบบที่คล้ายๆกับหลักการพัฒนาโปรแกรมแบบ Object-Oriented แต่ Algorithm Standard นั้นจะใช้ Syntax ของ ANSI C เป็นหลัก (ไม่ได้มีการเพิ่มคำสั่งเข้าไปเหมือนภาษา C++) โดยการที่ต้องรวม ตัวแปรที่ใช้ เข้าเป็น Structure แทนที่จะเป็น Global Variable ก็คล้ายๆกับหลักการ Encapsulation นั่นเอง

เมื่อทำการสร้าง Instance Object ของ Library ฟังก์ชั่นที่ใช้ในการสร้าง จะทำการจองพื้นที่ในหน่วยความจำตามที่ Library นั้นๆต้องการ โดย Instance Object ของส่วน Code นั้นจะมีเพียง Instance เดียวในระบบ แต่ Instance Object ของส่วนตัวแปรนั้น จะมีเป็นจำนวนเท่ากับ Object ที่ทำการสร้าง โดยเมื่อต้องการทำงานโดยใช้ Object ตัวใด ก็จะทำการ Pass ค่า Pointer ที่ชี้ไปยัง Instance Object ของส่วนตัวแปรของ Object ที่ต้องการทำงาน เพื่อให้ Instance Object ส่วน Code ทำการทำงานตามที่ Library นั้นๆ กำหนดไว้ โดยใช้ค่าตัวแปรที่อยู่ใน Structure ที่ถูกชี้โดย Pointer ที่ได้รับเข้ามา

ด้วยหลักการดังกล่าว จะเป็นการแบ่งแยกตัวแปรภายในของแต่ละ Library ออกจากกัน เพื่อลดความขัดแย้งที่อาจจะเกิดขึ้นระหว่าง Library ในระบบ อีกทั้งยังสามารถนำ Library ที่สร้างขึ้นไปใช้ในการพัฒนาระบบได้อย่างรวดเร็ว ตามหลักการ Component-Based Development

11.2 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ DTMF

ในส่วนระบบสร้างสัญญาณ DTMF นั้น จากการทดสอบการทำงาน สามารถสร้างสัญญาณได้อย่างถูกต้องตามที่ได้ออกแบบไว้ แต่ในส่วนของระบบตรวจจับสัญญาณ DTMF นั้น จากการทดสอบการทำงาน พบว่ายังมีปัญหาเล็กน้อยในบางส่วน โดยเป็นปัญหาในส่วนของการเลือกใช้ค่า THRESHOLD ที่จะนำมาใช้ โดยจะเป็นผลทำให้ระบบตรวจจับสัญญาณฯ อาจตรวจพบบางสัญญาณ ที่มีความเบี่ยงเบนของความถี่สัญญาณไซน์ส่วนประกอบ อยู่ในช่วง Reject แต่ก็เป็นเพียงส่วนน้อยเท่านั้น และปัญหาอีกอย่างก็คือหากจังหวะของสัญญาณที่รับเข้ามาตรวจสอบ มีความก้ำกึ่งกับจังหวะที่ทำการกำหนดเอาไว้สำหรับระบบตรวจจับสัญญาณฯ ก็อาจจะทำให้ผลการตรวจจับผิดพลาดได้

11.3 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ R2 MF

ในส่วนระบบสร้างสัญญาณ R2 MF นั้น จากการทดสอบการทำงาน สามารถสร้างสัญญาณได้อย่างถูกต้องตามที่ได้ออกแบบไว้ และในส่วนของระบบตรวจจับสัญญาณ R2 MF นั้น จากการทดสอบการทำงาน พบสามารถทำการตรวจจับสัญญาณได้อย่างถูกต้องเช่นกัน

11.4 วิจารณ์และสรุป ระบบสร้างและตรวจจับสัญญาณ CPT

โดยทั่วไปนั้น CPT ทั้ง Generation และ Detection Algorithm นั้นจะสามารถสร้างและตรวจจับสัญญาณได้เพียงแค่ 4 สัญญาณก็คือ

- Dial Tone
- Ring Tone
- Busy Tone
- Congestion Tone

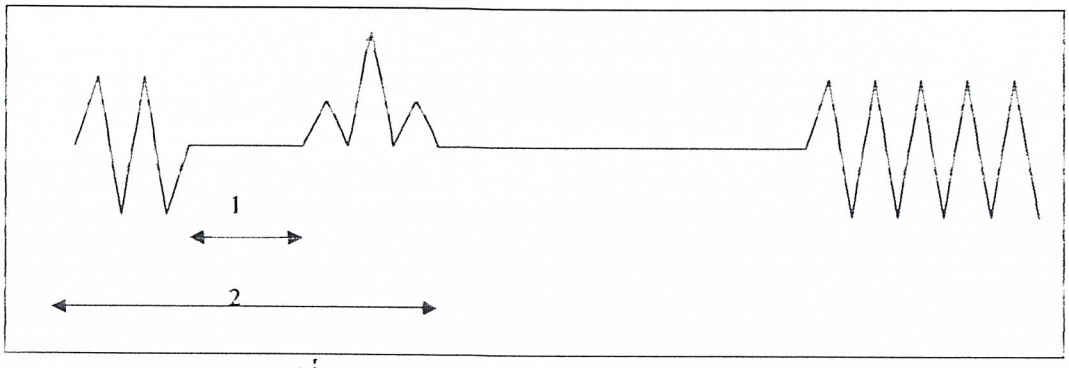
แตกต่างจาก Algorithm ที่ได้พัฒนาขึ้นมานี้ซึ่งสามารถ Generation และ Detection ได้ทั้งหมด 14 สัญญาณ ซึ่งเป็นสัญญาณที่มากที่สุดเท่าที่เราจะหาได้ แต่จะทำให้ Algorithm ช้าลงไปเพียงเล็กน้อย ก็เนื่องจากเป็นแค่เพิ่ม case ในคำสั่ง switch ให้มากขึ้นแค่นั้นเอง

ปัญหาที่สำคัญประการหนึ่งของการตรวจสอบก็ คือ การตรวจสอบความถี่ โดยใช้ Goertzel Algorithm ถึงจะเป็น Algorithm ที่มีประสิทธิภาพเนื่องจากทำงานง่าย ไม่ซับซ้อนมีแค่บวกลบ คูณหารง่าย จริงๆทำงานได้อย่างรวดเร็ว กว่าการใช้ Discrete Furious Transform ซึ่งจะเสียเวลากว่ามาก แต่ถ้าเกิดจากความถี่ผสมกันมากกว่า 1 ความถี่นั้นจะทำให้ค่า ที่หามาได้จาก Goertzel Algorithm มีค่าไม่เท่ากัน ตัวอย่างเช่น ต้องการหาค่า Goertzel Algorithm ของความถี่ 360 ที่เกิดจาก 360 Hz + 440 Hz จะได้ค่า Goertzel Algorithm มาไม่เท่ากับ ค่า Goertzel Algorithm ที่หาได้จากความถี่ 360 Hz + 497 Hz (ควบคุมให้แอมปลิจูด ทั้ง 2 มีค่าเท่ากัน) ซึ่งจะเป็นปัญหาต่อการที่จะระบุให้แน่ชัดว่าสามารถตรวจสอบความถี่เท่าไรถึงเท่าไร ซึ่งวิธีการแก้ปัญหาคือ หลีกเลี่ยงที่จะไม่สร้างสัญญาณที่มีความถี่ใกล้เคียงกับขอบๆของ range ที่จะตรวจสอบได้

ปัญหาที่สอง คือ ถ้า Amplitude ของสัญญาณทีค่าไม่เท่ากันถึงจะเป็นความถี่เดียวกัน ก็จะได้ค่า Goertzel Algorithm ไม่เท่ากันแต่ปัญหานี้ยังพอแก้ได้โดยการ Normalize สัญญาณให้มีค่า แอมปลิจูดสูงสุดตามที่เราต้องการได้

11.5 ข้อจำกัดในการใช้งาน ระบบสร้างและตรวจจับสัญญาณโทรศัพท์

1. สามารถทนสัญญาณรบกวนที่เกิดจากช่วงสัญญาณขาดหายไปในช่วง Tone Duration ซึ่งช่วงสัญญาณที่ขาดหายมีค่าไม่เกินมาตรฐานที่กำหนด
2. สามารถทนสัญญาณรบกวนในลักษณะ FIR (Finite Impulse Response) ที่เกิดขึ้นในช่วง Pause Duration ได้ แต่ถ้า FIR เกิดหลังจากหมดช่วง Tone Duration ไม่นาน (อัลกอริทึมจะมองว่าสัญญาณช่วงเงียบที่เกิดขึ้นก่อนหน้า FIR เป็นสัญญาณรบกวนในข้อ 1 แทน) จะไม่สามารถแก้ไขได้และอาจเกิดความผิดพลาดในส่วนของจังหวะสัญญาณได้



รูปที่ 11-1 การเกิดสัญญาณรบกวนแบบ FIR

ถ้าช่วงที่ 1 เป็นสัญญาณช่วงเงียบ (มีระดับค่าพลังงานของสัญญาณ น้อยกว่าที่มาตรฐานกำหนด) ที่มีค่าเป็นไปตาม Signal Interruption สัญญาณรบกวนนั้นจะถูกรวมเป็นส่วนหนึ่งของช่วง Tone Duration ดังช่วงที่ 2 ทำให้จังหวะสัญญาณของช่วง Tone Duration เกิดความผิดพลาดโดยช่วง Tone Duration มีค่ามากขึ้น แต่ถ้าช่วงที่ 1 สัญญาณช่วงเงียบไม่เป็นไปตาม Signal Interruption จะถือว่าเป็นสัญญาณรบกวนชนิด FIR ซึ่ง อัลกอริทึมที่พัฒนามานี้สามารถจัดการได้

3. อัลกอริทึมที่พัฒนานี้ไม่ได้ถูกออกแบบมาให้จัดการกับสัญญาณรบกวนประเภท ECHO ดังนั้น เพื่อให้เกิดประสิทธิภาพมากยิ่งขึ้นควรใช้ควบคู่กับ Echo Cancellation Algorithm
4. สัญญาณรบกวนที่อยู่ในช่วง Pause Duration ของ สัญญาณ DTMF, และ สัญญาณ CPT นั้น อาจทำให้ไปรบกวนการทำงานของส่วนตรวจจับจังหวะของสัญญาณ และอาจทำให้ไม่สามารถตรวจจับสัญญาณ DTMF หรือ สัญญาณ CPT ได้เลย เนื่องจากจะกลายเป็นว่าสัญญาณที่เข้ามามีจังหวะสัญญาณที่ไม่เป็นไปตามค่าที่ยอมรับได้ ที่ได้ทำการกำหนดเอาไว้ แต่ในส่วนของสัญญาณ R2 MF นั้น ไม่มีการกำหนดเกี่ยวกับช่วง Pause Duration และในการตรวจจับสัญญาณ R2 MF นั้นก็ไม่ได้มีการตรวจสอบจังหวะของสัญญาณ เพียงแต่นำสัญญาณมาตรวจสอบระดับค่าพลังงานของสัญญาณ และตรวจสอบความถี่สัญญาณไซน์ส่วนประกอบ เท่านั้น ดังนั้น สัญญาณรบกวนที่เข้ามาจึงมีผลน้อย เพราะก็เหมือนกับเป็นการเพิ่มความถี่ส่วนประกอบเข้ามา ซึ่งถ้าหากไม่ใช่ความถี่ในช่วงที่สัญญาณ R2 MF ใช้งานอยู่ ก็จะมีผลน้อยกับ Goertzel Algorithm ที่ได้รับการระบุค่าความถี่ที่ต้องการทำการตรวจสอบไว้แล้ว
5. สัญญาณรบกวนที่เกิดจากสื่อตัวนำมีคุณภาพไม่ดี (เช่น เสียง "ซ่า ซ่า" ที่ได้ยินตอนคุยโทรศัพท์) ไม่สามารถระบุได้แน่ชัดว่า อัลกอริทึม จะสามารถทนได้ขนาดไหน เนื่องจากสัญญาณรบกวนประเภทนี้ลักษณะจะไปทำให้ความถี่ของสัญญาณมีการเปลี่ยนแปลง อย่างเช่นเกิดความถี่ที่ไม่ต้องการเข้ามาผสมอยู่ทำให้ ค่าที่ได้จากสมการ Goertzel Algorithm มีค่าเปลี่ยนไป ซึ่งถ้าเปลี่ยนมากในทางที่ทำให้ค่าที่ได้จาก Goertzel Algorithm เพิ่มขึ้นก็จะทำให้ความถี่ที่ปกติตรวจจับไม่ไม่ได้กลับมาตรวจจับได้ เพราะค่าที่ได้จาก Goertzel Algorithm มีค่ามากกว่า THRESHOLD ที่ตั้งไว้ แต่ถ้าสัญญาณรบกวนทำให้ค่าที่ได้จาก Goetzel Algorithm มีค่าน้อยลงก็จะทำให้บางความถี่ที่เคยตรวจจับได้ กลับกลายเป็นตรวจจับไม่ได้เพราะมีค่าที่ได้จาก Goertzel Algorithm

น้อยกว่า THRESHOLD ที่ตั้งไว้ เพราะฉะนั้นการสร้างสัญญาณโดยเลือกความถี่ที่ทำให้เกิดค่า Goertzel Algorithm สูงนั้น(ค่า Step Size ในการสร้างสัญญาณ และค่า Coefficient ในการตรวจจับสัญญาณ ได้รับการคำนวณโดยใช้ค่าความถี่ที่เท่ากัน) จะมีผลทำให้ทนต่อสัญญาณรบกวนประเภทนี้ได้อย่างดีเพราะ ค่าที่ได้จาก Goertzel Algorithm มีค่าสูงอยู่แล้ว ถ้าสัญญาณรบกวนทำให้ค่าที่ได้จาก Goertzel Algorithm สูงขึ้นก็ยังตรวจจับได้อยู่ดี แต่ถ้าสัญญาณรบกวนทำให้ค่าที่ได้จาก Goertzel Algorithm มีค่าต่ำลง (ซึ่งส่วนใหญ่จะเป็นอย่างนั้น) ถ้าจะทำให้ตรวจจับไม่ได้นั้นก็ต้องใช้สัญญาณรบกวนที่แรง เพื่อที่จะรบกวนให้ค่า Goertzel Algorithm ที่มีค่าสูง กลับมามีค่าต่ำกว่า THRESHOLD

บรรณานุกรม

[1] Xiangdong Fu, and Zhaohong Zhang (2000) : "A Multichannel/Algorithm Implementation on the TMS320C6000 DSP" – SPRA556B , "Texas Instruments Inc.", February 2000.

[2] Murat Karaorman (2002) : "Techniques for Implementing Shared Relocatable Buffers Using the TMS320 DSP Algorithm Standard" – SPRA790 , "Texas Instruments Inc.", October 2002.

[3] Stig Torud (2002) : "A Technical Overview of eXpressDSP-Compliant Algorithms for DSP Software Producers" – SPRA579C , "Texas Instruments Inc.", September 2002.

[4] Texas Instruments Inc. : "TMS320 DSP Algorithm Standard Rules and Guidelines" – SPRU352E , "Texas Instruments Inc.", October 2002.

[5] Texas Instruments Inc. : "TMS320 DSP Algorithm Standard API Reference" – SPRU360C , "Texas Instruments Inc.", September 2002.

[6] International Telecommunication Union : "Multifrequency Push-Button Signal Reception" – ITU-T Recommendation Q.24 , "International Telecommunication Union".

[7] International Telecommunication Union : "The Receiving Part of the Multifrequency Equipment" – ITU-T Recommendation Q.455 , "International Telecommunication Union".

[8] International Telecommunication Union : "Technical Characteristics of Tones for the Telephone Service" – ITU-T Recommendation E.180/Q.35 , "International Telecommunication Union".

[9] Geoff Probert : "Compute Signal Power with Fixed-Point Calculators", http://www.embedded.com, September 2003.