เอกสารอ้างอิง

1. Knowlton, K., "Progressive Transmission of Grey-scale and binary pictures by simple, efficient, and loseless encoding schemes", Proc. IEEE, 68, pp. 885-896, 1980.
2. Takaikawa, K., "Fast Progressive Reconstruction of a Transformed Image", IEEE Trans.Info.Theory, IT-30, pp.111-117, 1984.
3. Chen, W.H., and C.H. Smith, "Adaptive coding of monochrome and color images", IEEE Trans.Commun., COM-25, pp.1285-1292, 1977.
4. Burt, P.J., and E.H. Adelson, "The Laplacian pyramid as a compact image code", IEEE Trans. Commun., COM-31, pp. 532-540, 1983.
5. Woods, J.W., and S.P. O'Neil, "Subband Coding of Images", IEEE Trans. ASSP, ASSP-34, pp. 1278-1288, 1986.
6. Hofmann, W.D., and D.E. Troxel, "Making Progressive Transmission Adaptive", IEEE Trans.Commun., COM-34, pp.806-813, 1986.
7. Lohscheller, H., "A subjectively adapted image communication system", IEEE Trans.Commun., COM-32, pp.1316-1322, 1984.
8. Chitraprasert, B., and K.R. Rao, "Human Visual Weighted Progressive Image Transmission", IEEE Trans. Commun., COM-38 , pp.1040-1044, 1990.
9. Hudson, G.P.,"Photovideotex Image Compression Algorithms towards International Standardisation", ESPRIT 563, 5-16, Brussel, 1987.
10. ISO, "Adaptive Discrete Cosine Transform Coding Scheme for Still Image Telecommunication Services",ISO/TC97/SC2/WG8-ADCTG , 1988.

11. Hudson, G.P., H. Yasuda, and I. Sebestyen, "The International Standardisation of A Still Picture Compression Technique", GLOBECOM '88, pp. 1016-1021, Florida, 1988.

12. ISO, "Jpeg Technincal Specification, rev.5 ", ISO/IEC JTC1/SC2/ WG8, 1990.

13. Kamanagar, F.A., and K.R. Rao, "Fast Algorithms for the 2-D Discrete Cosine Transform", IEEE Trans. Comput., C-31, 899-847, 1982.

14. IBM, Technical Reference, Personal Computer XT, 1983.

15. Ahmed, N., J. Natarajan, and K. Rao, "Discrete Cosine Transform" , IEEE Trans. Comput., C-23, 90-93, 1974.

16. Chen, W.H., C.H. Smith, and S.C. Fralick, "A fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans.Commun., COM-25, pp. 1004-1009, 1977.

17. Yip, Y., and K.R. Rao, "A fast Computational Algorithm for the Discrete Sine Transform", IEEE Trans. Commun., COM-28, 304-307, 1980.

18. Eggerton, J.P., and M.D. Srinath, " A Visually Weighted Quantization Scheme for Image Bandwidth Compression at Low Data Rates", IEEE Trans. Commun., COM-34, 840-847, 1986.

19. Mannos, J.L., and D.J. Sakrison, "The effects of a Visual Fidelity Criterion on the Encoding of Images", IEEE Trans. Info. Theory, IT-20, 525-536, 1974.

20. Kunt, M., A Ikonomopoulos, and M. Kocher, "Second-generation Image-coding techniques", Proc. IEEE, 73, pp. 549-574, 1985.

21. Texas Instruments, TMS32010 User's Guide, 1983.

22. Texas Instruments, TMS32010 Assembly Language Programmer's Guide , 1983.

23. Texas Instruments, Digital Signal Processing Applications with the TMS 320 Family, 1986.

24. บุญช่วย ทรัพย์มนชัย, "การส่งภาพนิ่งแบบโปรเกรสซีฟ", การประชุมวิชาการทางวิศวกรรมไฟฟ้า 9 สถาบันอุดมศึกษา ครั้งที่ 12, หน้า 223-232, มหาวิทยาลัยเกษตรศาสตร์, 2532.

25. Clarke, R.J., Transform Coding of Image, Academic Press, Florida , 1985.

26. Jain, A.K., Fundamentals of Digital Image Processing, pp. 49-59, 150-151, New Jersey, 1989.

27. Lim, J.S., Two Dimensional Signal and Image Processing, Prentice Hall, New Jersey, 1990.

28. Yasuda, Y., "Progressive Coding of Still Images", International Work Shop on Image Coding, pp. 129-170, Korea, 1987.

29. Eggebrecht, L.C., Interfacing to the IBM Personal Computer, Howard W. Sams, Inc. Co., Indianapolis, 1983.

ภาคผนวก ก

Discrete Cosine Transform

Discrete Cosine Transform (DCT) เป็นออร์โทโกนัลทรานส์ฟอร์มแบบหนึ่ง ซึ่งถูกแนะนำเป็นครั้งแรก โดย Ahmed & Rao [15] ในปีค.ศ.1974 นับตั้งแต่นั้นเป็นต้นมา DCT ก็เป็นที่สนใจและถูกนำไปใช้ในงานด้านการประมวลผลภาพ (Digital Signal Processing) กันอย่างกว้างขวาง เนื่องจากว่า DCT จะให้ผลของการทรานส์ฟอร์มดีใกล้เคียงกับ KLT (Karnuhen-Loeve Transform) มาก เมื่อภาพมีความสัมพันธ์ระหว่างกัน (correlation) สูง

DCT เป็นการทรานส์ฟอร์มค่าจริง (Real Transform) ซึ่งกำเนิดขึ้นมา เพื่อแก้ปัญหาที่เกิดขึ้นเนื่องจากการทำ DFT (Discrete Fourier Transform) ต่อข้อมูลใดๆ ในการตัดตอนของข้อมูลที่ถูกสุ่มดังรูปที่ ก.1 (ก)-(ข) เมื่อนำไปทำ DFT จะต้องมีการซ้ำ (repeat) ค่าดังกล่าวเพื่อให้สเปคตรัมของฟูเรียร์อยู่ในสภาพ Discrete และมีการซ้ำเป็นรายคาบ ดังรูปที่ ก.1 (ค)

จากสภาพดังกล่าวทำให้เกิดความไม่ต่อเนื่องขึ้นที่บริเวณขอบของคาบ อันจะก่อให้เกิด Spurious Spectral ขึ้นเนื่องจากปรากฏการณ์ของกิบส์ (Gibbs phenomena) DCT จะสามารถกำจัดปัญหานี้ไปโดยการพับข้อมูลกลับ ทำให้บริเวณขอบต่อเนื่องดีขึ้น จุดที่พับกลับอาจจะให้ไม่ทับกันดังรูปที่ ก.1 (ง) หรือ ทับกันดังรูปที่ ก.1 (จ) ก็ได้ แต่ในกรณีที่จุดกลางทับกันจะเป็นที่นิยมมากกว่า เนื่องจากมีจุดเป็นจำนวนคู่ ซึ่งง่ายต่อการสร้าง fast transform

ก.1 สมการของ DCT

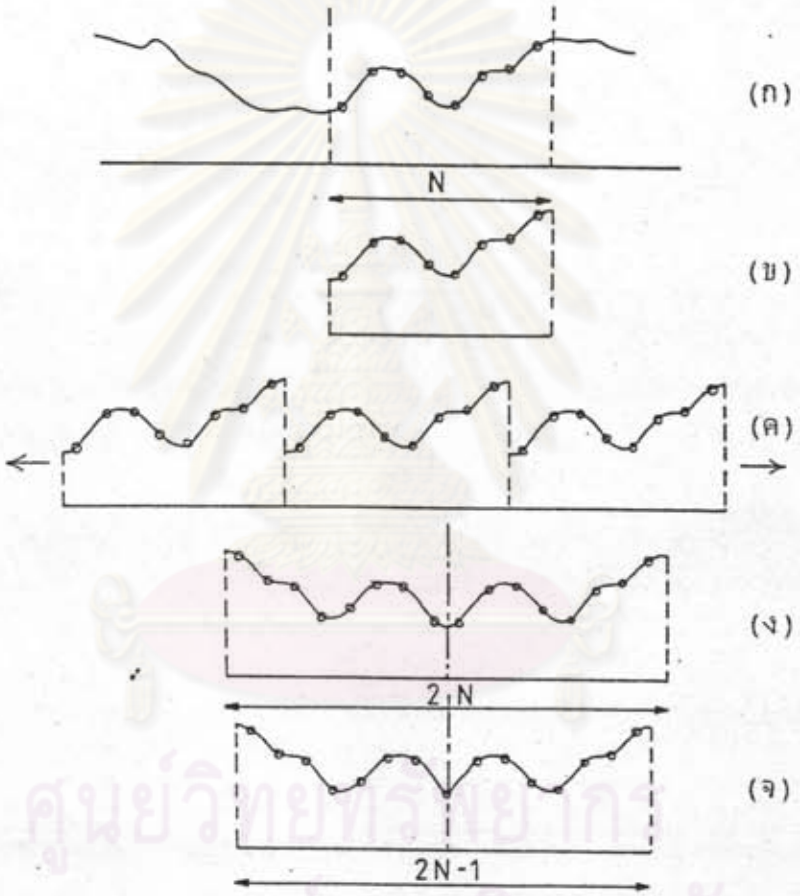
เมื่อมอง DCT ที่มีจุดเป็นจำนวนคู่ดังรูปที่ ก.1 (ง) เป็นฟูเรียร์ทรานส์ฟอร์มที่มีขนาดเป็น 2N จุด เราจะพบว่า สัมประสิทธิ์ของฟูเรียร์จะสามารถจัดให้อยู่ในรูปจำนวนจริงได้ เนื่องจากเป็นฟังก์ชันคู่ คือ

$$\bar{Y} = \bar{C}\bar{X}$$

Y เป็นสเปคตรัม y(k); 0 < k < N-1, X เป็นข้อมูล x(n); 0 < n < N-1 และ C เป็นเมตริกซ์สัมประสิทธิของการทรานส์ฟอร์ม = {c(k,n)} ซึ่ง

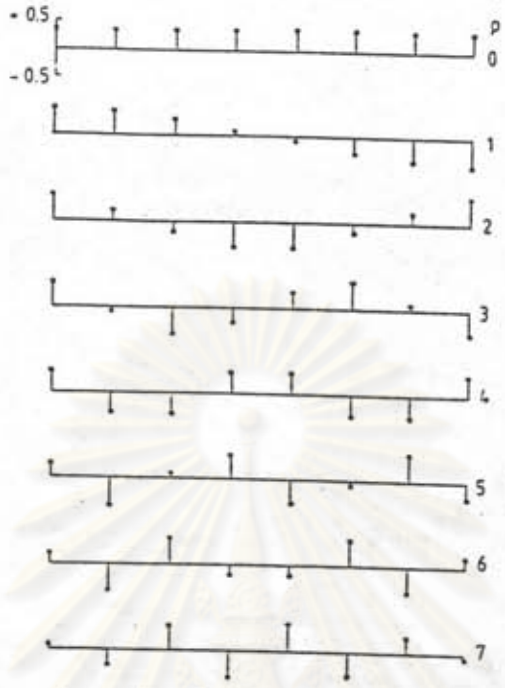
$$c(k,n) = 1 / \sqrt{N} \quad ,k = 0;\ 0 < n < N-1$$

$$= \sqrt{2 / N}\ \cos[\pi(2n+1)k/2N] \quad ,1 < k < N-1;\ 0 < n < N-1$$



รูปที่ ก.1 ข้อมูลที่ถูกสุ่ม และ ซ้ำเป็นรายคาบในการทรานส์ฟอร์ม FFT และ DCT

รูปที่ ก.2 แสดงถึงเวกเตอร์พื้นฐาน (basis vector) c(k) ที่ได้ เมื่อมีจุดต่อคาบ N = 8 จะสังเกตได้ว่า เวกเตอร์พื้นฐานแต่ละตัวจะมาจากการสุ่มรูปคลื่นโคซายน์ที่มีความถี่เพิ่มขึ้นตามลำดับ



รูปที่ ก.2 เวกเตอร์พื้นฐานของ DCT 1 มิติ ขนาด 8 จุด

ดังนั้นสเปคตรัม Y ของ DCT จากข้อมูล X จะได้

$$y(k) = a(k) \sum_{n=0}^{N-1} x(n)\cos[\pi(2n+1)k/2N] \quad , 0 < k < N-1$$

โดยที่

$$a(0) = 1 / \sqrt{N}$$

$$a(k) = \sqrt{2 / N} \quad , 1 < k < N-1$$

ทำนองเดียวกัน การทรานส์ฟอร์มกลับเพื่อหาค่า X ก็จะได้

$$x(k) = \sum_{k=0}^{N-1} a(k)y(k)\cos[\pi(2n+1)k/2N] \quad , 0 < n < N-1$$

โดยที่ ค่าของ a(k) จะเหมือนกับข้างต้น

ก.2 คุณสมบัติบางประการของ DCT

1. DCT เป็นการทรานส์ฟอร์มค่าจริง (Real transform) และ เป็นออร์โทโกนัลทรานส์ฟอร์มแบบหนึ่งด้วย

2. DCT ไม่ใช่ ส่วนจริง (Real part) ของ DFT แต่เป็นค่าที่ได้จากการทำ DFT ของ สัญญาณที่ถูกขยายให้สมมาตรทั้งสองด้านดังที่ได้กล่าวมาแล้ว

3. DCT สามารถถูกสร้างให้เป็น Fast Transform ได้เช่นเดียวกับ DFT ซึ่งมีผู้เสนอขั้นตอนวิธีในการสร้างเป็น Fast Transform มากมาย

4. DCT มีการอัดแน่นของพลังงานของข้อมูลสูงมาก สำหรับข้อมูลที่มีความสัมพันธ์ระหว่างกันสูง

ภาคผนวก ข

อัลกอริทึมสำหรับการสร้าง fast transform ของ DCT

จากคุณสมบัติของ DCT ที่สามารถสร้างเป็น fast transform ได้ทำให้มีผู้คิดค้นอัลกอริทึมสำหรับการสร้าง fast transform กันอย่างมากมาย ซึ่งอัลกอริทึมที่นิยมกันอย่างแพร่หลาย คือ อัลกอริทึม ของ Chen & Smith [16] สำหรับการทรานส์ฟอร์มใน 2 มิติ ก็จะใช้คุณสมบัติของ separable function คือ ทำการทรานส์ฟอร์ม 1 มิติ ทีละแกน อย่างไรก็ตามอัลกอริทึมของ Chen & Smith ก็ยังคงมีจุดอ่อนอยู่ตรงที่มีการคูณอยู่มาก ทำให้ต้องเสียเวลาในการคำนวณมากโดยเฉพาะ เมื่อทำการคูณด้วยซอฟต์แวร์

Rao [17] ได้เสนอหลักการสร้าง fast transform ของ DCT อีกแบบหนึ่ง โดยวิธีการเพิ่มมิติของข้อมูลที่ใช้ในการทรานส์ฟอร์ม ซึ่งมีผลทำให้ลดจำนวนการคูณได้มาก ในขณะที่จำนวนบวกคงเดิม

ข.1 <u>DCT 2 มิติ</u>

สำหรับ DCT ขนาด 2 มิติสามารถเขียนเป็นสมการได้ คือ

$$y(u,v) = c(u)c(v)/\sqrt{MN} \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} x(i,j)\cos[(2i+1)u\pi/2M]\cos[(2j+1)v\pi/2N]$$

$$u = 0, 1, \ldots, M-1$$

$$v = 0, 1, \ldots, N-1$$

ทำนองเดียวกันการทรานส์ฟอร์มกลับจะได้

$$x(i,j) = 1/\sqrt{MN} \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} c(u)c(v)y(u,v)\cos[(2i+1)u\pi/2M]\cos[(2j+1)v\pi/2N]$$

$$i = 0, 1, \ldots, M-1$$

$$j = 0, 1, \ldots, N-1$$

โดยที่ $c(l) = 1 \quad , \quad l = 0$

$= \sqrt{2} \quad , \quad l \neq 0$

หรือเขียนได้เป็น

$$[Y] = [T][X]$$

จากสูตรข้างต้น ถ้าเราจัดเรียงข้อมูลที่เข้าออกใหม่ คือ ให้

$$[X_L] = [x(0,0),x(0,1),\ldots,x(0,N-1),x(1,0),x(1,1),\ldots,x(M-1,N-1)]^T$$
$$[Y_L] = [y(0,0),y(0,1),\ldots,y(0,N-1),y(1,0),y(1,1),\ldots,y(M-1,N-1)]^T$$

จะได้ว่า

$$[Y_L] = [D(M,N)][X_L]$$

$[D(M,N)]$ คือ เมตริกซ์ขนาด MN x MN ซึ่งได้จากผลคูณ Kronecker

$$[D(M,N)] = [L(M)] \times [L(N)]$$

เมื่อพิจารณาเมตริกซ์ D เสียใหม่ให้อยู่ในรูปเมตริกซ์ H ที่ถูกจัดแถว และ แนวตั้งของเมตริกซ์เสียใหม่ โดยการคูณด้วยเมตริกซ์สลับที่ (permutation matrix) คือ

$$[D(M,N)] = [P_1(M,N)][H(M,N)][P_2(M,N)]$$

เมื่อแทน D ลงไปจะได้ว่า

$$[Y_{PL}] = [H(M,N)][X_{PL}]$$

เมื่อ [Y_{PL}] และ [X_{PL}] เป็นข้อมูลด้านออก และ ด้านเข้าที่ถูกสลับตำแหน่งใหม่ด้วยเมตริกซ์สลับที่ P1 และ P2 ตามลำดับ ในกรณีนี้เราจะสามารถแยก [H(M,N)] ออกเป็นเมตริกซ์ย่อยๆได้ เป็น ขั้นๆต่อไป คือ

$$[H(M,N)] = \begin{bmatrix} H(M/2,N) & 0 \\ 0 & G(M/2,N) \end{bmatrix} [A(MN)]$$

เมื่อ

$$[A(MN)] = \begin{bmatrix} I(MN/2) & I'(MN/2) \\ I'(MN/2) & -I(MN/2) \end{bmatrix}$$

I(MN/2) เป็น เมตริกซ์เอกลักษณ์ (identity matrix) , I'(MN/2) เป็น เมตริกซ์เอกลักษณ์ที่ทแยงด้านตรงข้าม (opposite diagonal identity matrix) เช่น

$$I'(2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

ซึ่ง [H(M/2,N)] ก็จะสามารถแยกย่อยต่อไปได้อีก

$$[H(M/2,N)] = \begin{bmatrix} H(M/2,N/2) & 0 \\ 0 & G(M/2,N/2) \end{bmatrix} [A(MN/2)]$$

ทำนองเดียวกันเมตริกซ์ประเภท H ก็จะสามารถแยกย่อยไปได้เรื่อยๆในลักษณะเดียวกัน ซึ่งเป็นลักษณะการเกิดซ้ำ (Recursive) เพราะ เมตริกซ์มิติที่สูงกว่าจะประกอบด้วยเมตริกซ์ที่มีมิติน้อยกว่าได้

ส่วน G(M/2,N) ก็จะสามารถแยกย่อยไปได้ต่างหากเป็น

$$[G(M/2,N)] = \begin{bmatrix} G_1(M/2,N/2) & 0 \\ 0 & G_2(M/2,N/2) \end{bmatrix} [A(MN/2)]$$

ซึ่งเมตริกซ์ประเภท G แต่ละเมตริกซ์ก็จะมีการแยกเมตริกซ์ที่มีแบบเฉพาะลงไปเรื่อยๆ ในกรณีของ DCT 2 มิติ ขนาด 8 x 8 จุดจึงได้เป็น

$$[Y_{P64}] = [H(8,8)][X_{P64}]$$

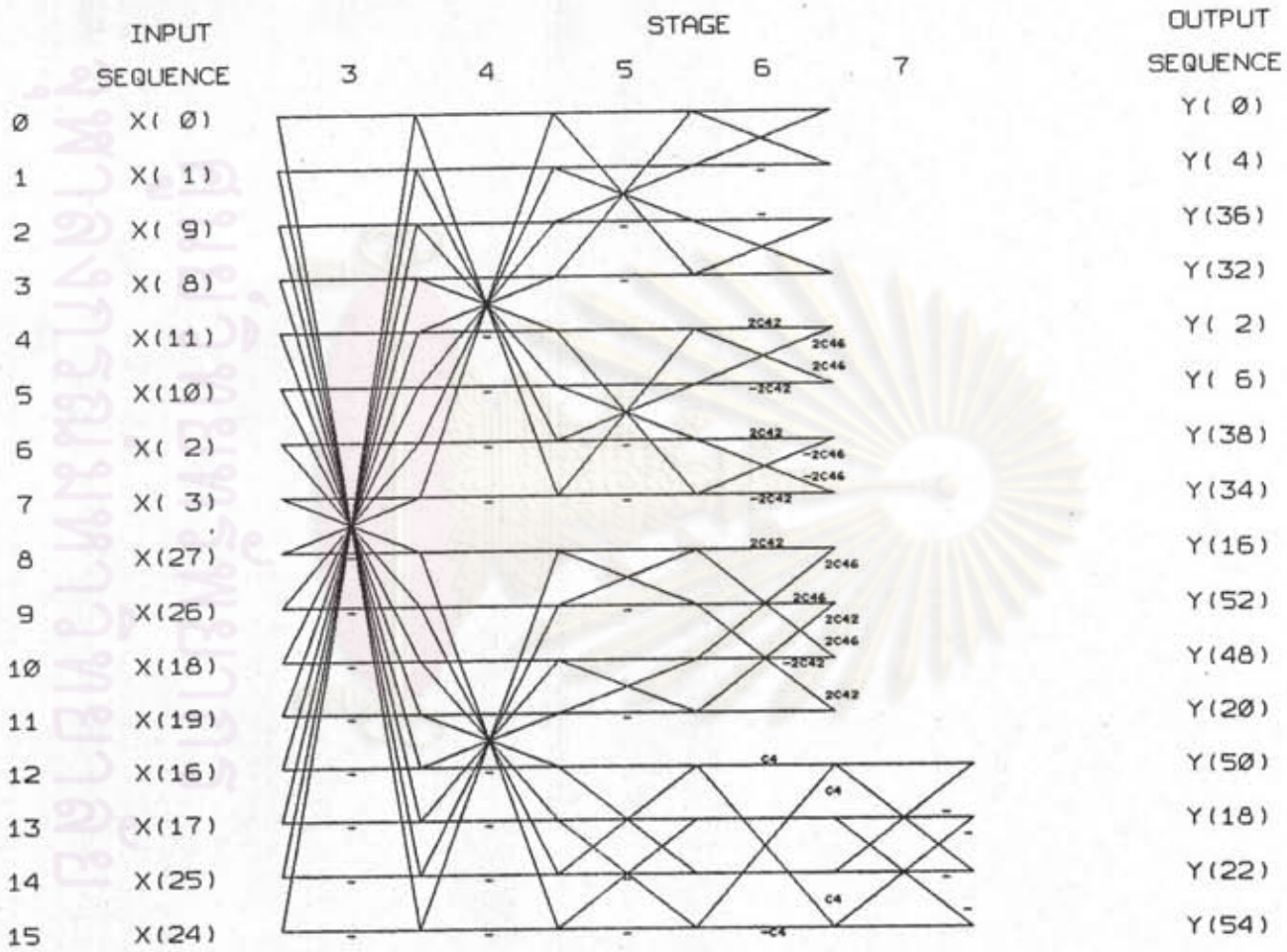
[H(8,8)] จะมีขนาดเป็น 64 ซึ่งจะแยกย่อยได้เป็น

$$[H(8,8)] = \begin{bmatrix} H(4,8) & 0 \\ 0 & G(4,8) \end{bmatrix} [A(64)]$$

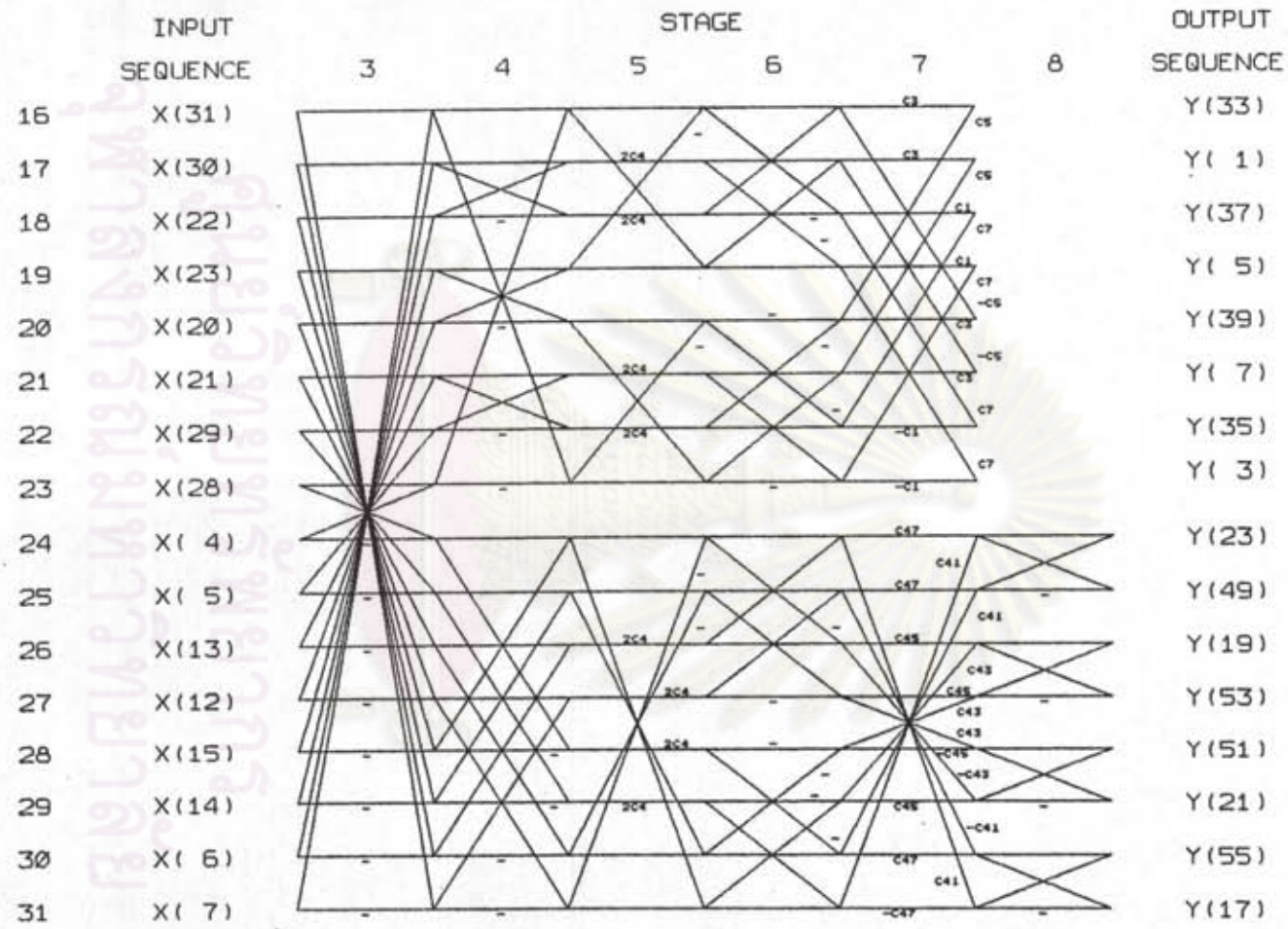
$$[H(4,8)] = \begin{bmatrix} H(4,4) & 0 \\ 0 & G(4,4) \end{bmatrix} [A(32)]$$

$$[G(4,8)] = \begin{bmatrix} G_1(4,4) & 0 \\ 0 & G_2(4,4) \end{bmatrix} [A(32)]$$

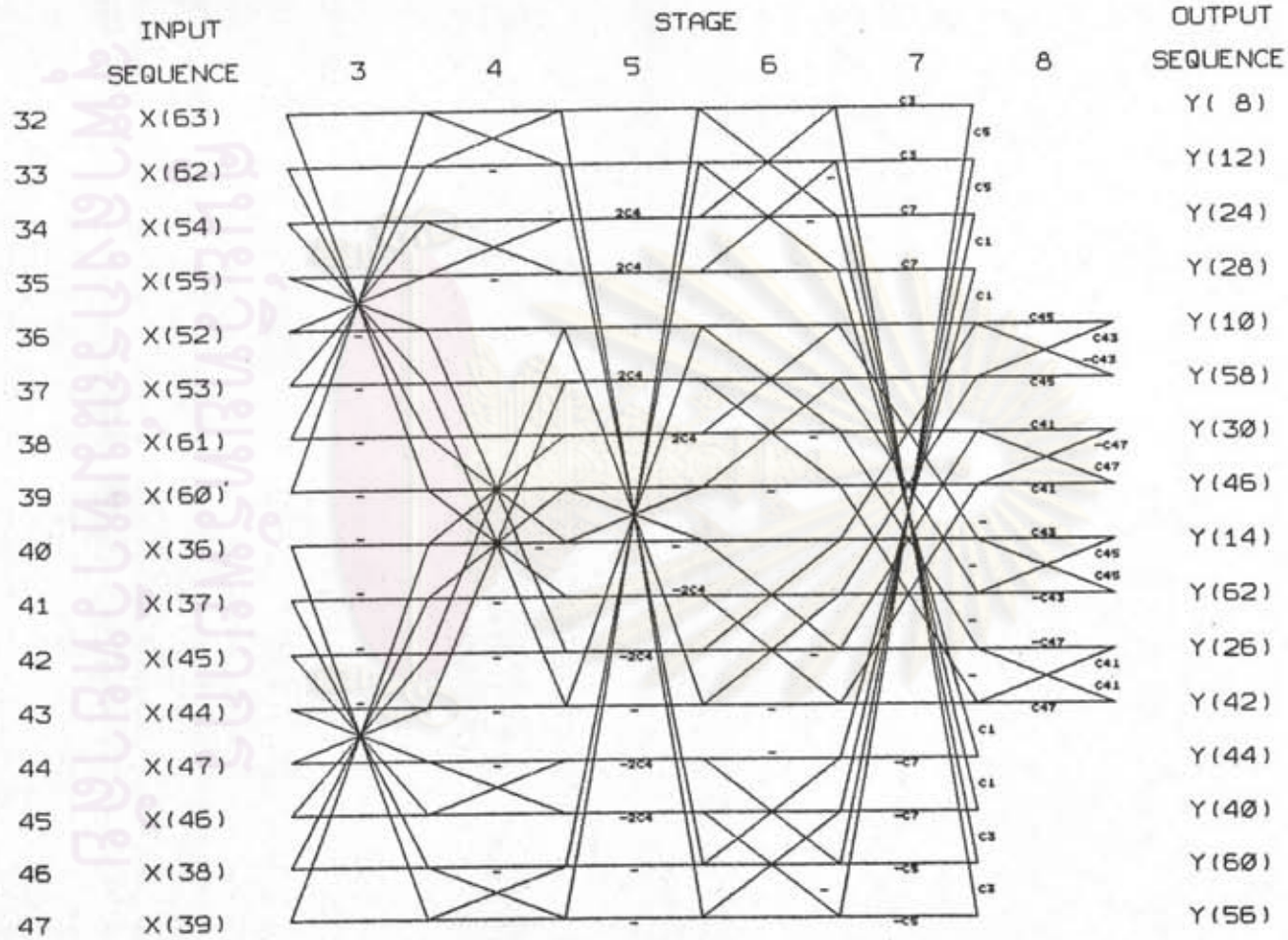
H(4,4), G(4,4), $G_1(4,4)$, $G_2(4,4)$ จะมีการแยกย่อยที่สามารถแสดงได้ด้วยแผนภูมิแบบผีเสื้อ (Butterfly chart) ได้ดังรูปที่ ข.1 ซึ่งบอกตำแหน่งการสลับข้อมูลด้านเข้าและ ด้านออกตามเมตริกซ์ของการสลับที่ P_1 และ P_2 ด้วย โดยละการประมวลผลในขั้นที่ 1 และ ขั้นที่ 2 ไป



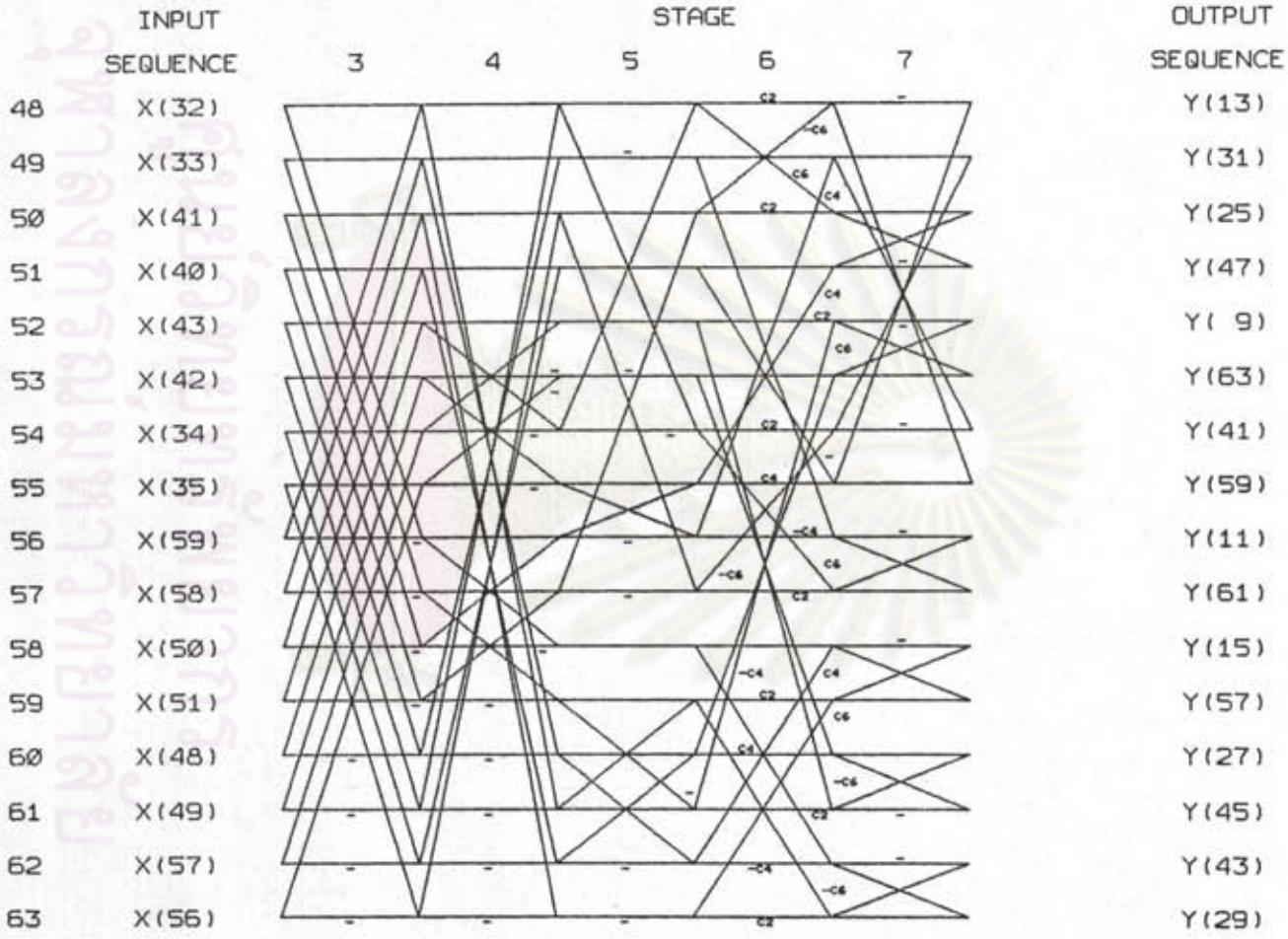
รูปที่ ข.1 แผนภูมิผีเสื้อของ fast transform สำหรับ DCT



รูปที่ ข.1 (ต่อ) แผนภูมิผีเสื้อของ fast transform สำหรับ DCT



รูปที่ ข.1 (ต่อ) แผนภูมิผีเสื้อของ fast transform สำหรับ DCT



รูปที่ ข.1 (ต่อ) แผนภูมิผีเสื้อของ fast transform สำหรับ DCT

ภาคผนวก ค
แบบจำลองการมองเห็นของมนุษย์
(Human Visual Model)

ในยุคแรก ๆ การลดข้อมูลภาพมักจะใช้วิธีการทางทฤษฎีข่าวสาร (Information Theory) ลดการซ้ำซ้อนกันของข้อมูลโดยอาศัยสถิติของข้อมูล เช่น วิธีการเข้ารหัสแบบเอนโทรปี แต่วิธีนี้ก็มีขีดจำกัดเนื่องจากว่า อัตราการลดข้อมูล และการเข้ารหัส จะขึ้นกับลักษณะของภาพต้นแบบอย่างมาก ภาพที่ต่างกันย่อมมีความน่าจะเป็นแตกต่างกัน หรือ แม้แต่ในภาพเดียวกันแต่ละบริเวณก็จะมีความน่าจะเป็นของการเกิดจุดแตกต่างกัน วิธีที่ใช้ได้ดีสำหรับภาพหนึ่ง ๆ อาจจะไม่ดีสำหรับภาพอื่นๆก็ได้ การลดข้อมูลในกรณีนี้จึงทำได้ไม่มากนัก

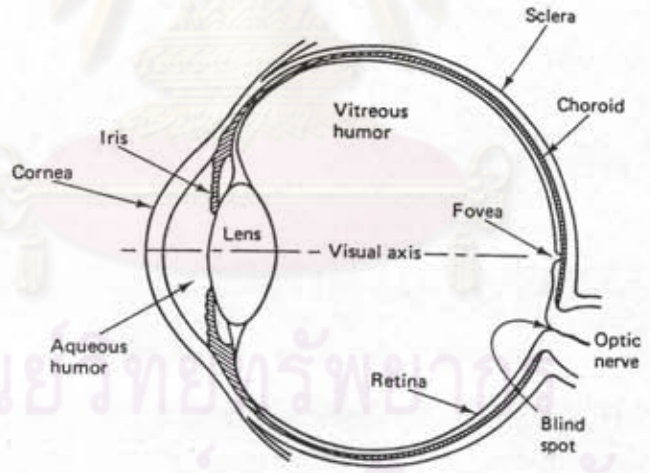
ภายหลังจึงได้มีผู้นำเอาผลตอบสนองของระบบการมองเห็นของมนุษย์ (Human visual response) มาร่วมใช้ในการเข้ารหัส เนื่องจากว่าผู้ใช้มักจะเป็นมนุษย์เสียส่วนใหญ่ การเข้ารหัสโดยวิธีนี้จะลดข้อมูลส่วนที่ไม่เกี่ยวข้องกับกลไกการมองเห็น (Psychovisual) ออกไปให้มากที่สุด โดยเหลือเฉพาะส่วนที่สำคัญสำหรับการมองก็พอ ได้มีผู้ที่สนใจทดลองสร้างแบบจำลองของระบบการมอง และ นำไปประยุกต์ใช้งานทางด้านประมวลผลภาพ ซึ่งให้ผลในการลดข้อมูลดีกว่าในกรณีแรกมากโดยเฉพาะที่อัตราของข้อมูลต่ำมาก [18] - [20]

ค.1 โครงสร้างทางกายภาพของตามนุษย์

ระบบการมองเห็นของมนุษย์ เป็นระบบประสาทที่ซับซ้อนที่สุดส่วนหนึ่งในร่างกาย อันเป็นเครื่องมือสำคัญในการให้ข้อมูลในการคิดของมนุษย์ ในระบบประสาทการมองเห็นจะประกอบด้วย ตา ซึ่งทำหน้าที่เปลี่ยนแสงสว่างให้เป็นสัญญาณประสาท (neural signal) และ ส่วนของสมอง ซึ่งจะรับเอาสัญญาณประสาทมาแยกแยะ แล้วประมวลผลต่อไป

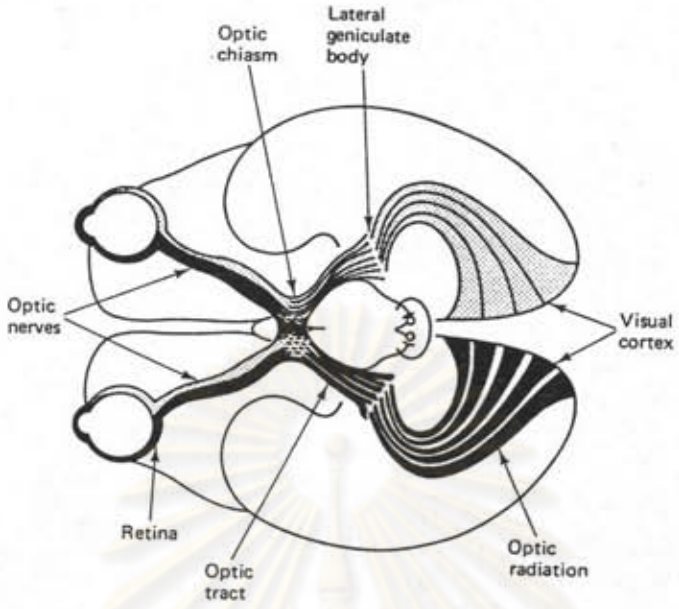
โครงสร้างของตาในส่วนที่ประกอบเป็นระบบการมองเห็น จะเริ่มต้นจาก Cornea ทำหน้าที่หักเหแสงให้มารวมกันคล้ายกับกระจกนูนในกล้องถ่ายรูป จาก Cornea แสงจะวิ่งผ่านของเหลวใสที่เรียกว่า Aqueous humor ไปสู่ ม่านตา (Iris) ม่านตาจะเป็นช่องเปิดของกล้ามเนื้อซึ่งจะทำหน้าที่ปรับปริมาณแสงสว่างที่จะเข้าสู่ภายในให้พอเหมาะ โดยการหดหรือขยายรูเปิด หลังม่านตาจะเป็นเลนส์ตาซึ่งสร้างจากเส้นใย (fibre) ที่โปร่งใสซ้อนกันเป็น

ชั้น ๆ มีรูปร่าง และ ขนาดคล้ายเม็ดถั่ว เลนส์ตาจะมีหน้าที่รวมแสงที่เข้ามาให้โฟกัสเข้าสู่ศูนย์กลางของลูกตา (fovea) โดยสามารถปรับรูปร่างของมันเองได้เพื่อปรับระยะโฟกัสให้ตกไปยังจอภาพ (Retina) ได้อย่างถูกต้อง ภายในลูกตาจะถูกบรรจุด้วยสารชนิดหนึ่งคล้ายเยลลี เรียกว่า Vitreous humor เพื่อรักษารูปร่างของตาไว้ เมื่อแสงวิ่งมากระทบกับจอภาพซึ่งจะกินเนื้อที่ประมาณ 65 % ของเนื้อที่ภายในลูกตา แสงจะไปกระตุ้นเซลรับแสง (light perception cell) 2 ประเภท คือ เซลรูปแท่ง (rod) และ เซลรูปกรวย (cone) ทำให้เกิดสัญญาณประสาทส่งไปยังสมอง เซลรูปกรวยจะมีจำนวนน้อยกว่า และ ไวต่อแสงน้อยกว่าเซลรูปแท่ง แต่เซลรูปกรวยมีข้อดีกว่าตรงที่สามารถให้สีในการมองได้ในที่ที่มีแสงสว่างมาก ๆ (photopic) ส่วนเซลรูปแท่งมักจะทำงานในที่มืด หรือ ในเวลากลางคืนได้ดี (scotopic)



รูปที่ ค.1 โครงสร้างจำลองของลูกตา

สัญญาณประสาทที่ออกจากจอภาพในตาจะถูกส่งไปยังสมอง บริเวณส่วนที่เรียกว่า Visual Cortex ซึ่งจะนำเอาสัญญาณที่ได้ไปประมวลผลต่อไป



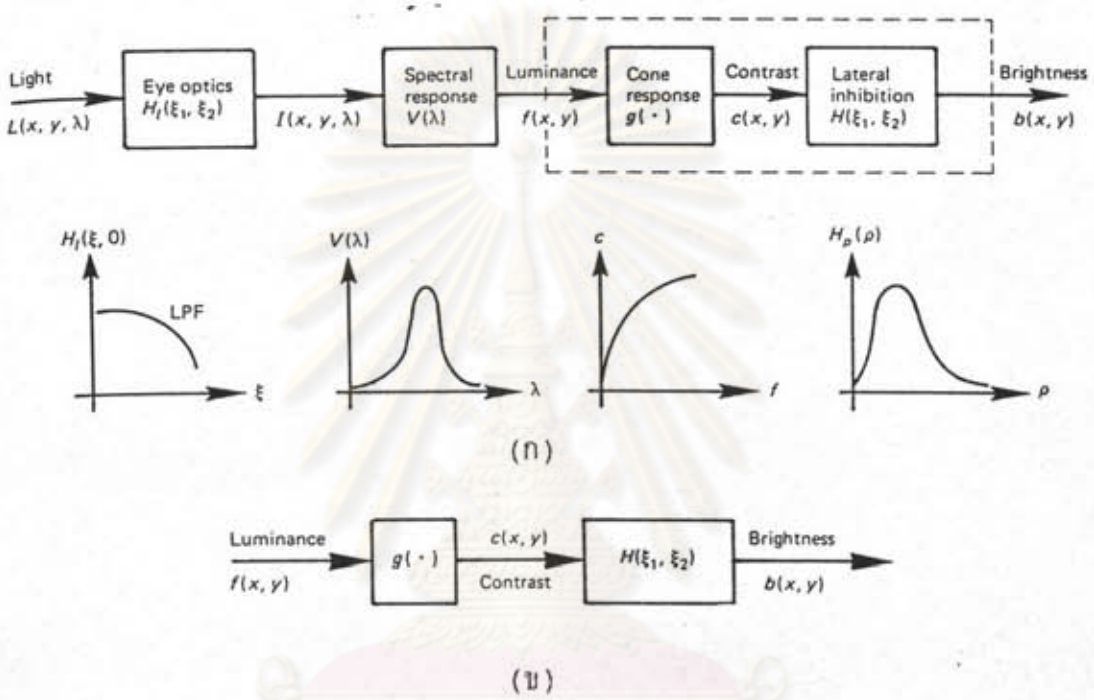
รูปที่ ค.2 ทางเดินของประสาทตาไปสู่สมอง

ค.2 แบบจำลองของระบบการมองเห็นภาพสีเดียว (Monochrome Vision Model)

จากโครงสร้างของตาดังที่ได้กล่าวมา สามารถนำสร้างเป็นฟังก์ชันจำลองการทำงานของอวัยวะที่สำคัญในเส้นทางที่แสงวิ่งผ่านไปสู่สมอง ซึ่งเราจะแยกการทำงานเป็นสองส่วน คือ ส่วนของตา และ ส่วน Cortex ในสมอง

ในส่วนของตา เมื่อแสงเข้าสู่ตาจะถูกกรองผ่านต่ำด้วยตัวกรอง ซึ่งมีผลตอบสนองเป็น $H_1(\xi_1, \xi_2)$ และ ผ่านฟังก์ชันที่เป็นผลตอบสนองต่อความถี่สว่างสัมพัทธ์ (relative luminance efficiency function) $V(\lambda)$ ซึ่งจะได้เป็นค่าการกระจาย luminance $f(x,y)$ ออกมา จากนั้นจะถูกประมวลผลด้วยเซลรูปแท่ง และ กรวยที่จอรับภาพ ซึ่งแทนได้ด้วยฟังก์ชันไม่เชิงเส้น $g(.)$ และ ผลตอบ $H(\xi_1, \xi_2)$ ที่เรียกว่า Modulation Transfer Function (MTF) สำหรับแบบจำลองที่มีสมมุติฐานว่าเป็น เชิงเส้น, ไอโซโทรปิค, และ ไม่แปรตามตำแหน่ง (spatial invariant) ผลตอบสุดท้ายที่ได้จะเป็น สัญญาณประสาทที่ถูกส่งไปยังสมอง ซึ่งจะทำการประมวลผลตามลักษณะย่อยของภาพที่ได้ การจำลองระบบการมองเห็นของภาพสีเดียวโดยรวมจะแสดงได้ดังรูปที่ ค.3 (ก)

แต่เนื่องจากพบว่า ตัวกรองผ่านต่ำ มีอัตราการลดต่ำลงน้อยมากเมื่อเทียบกับผลของฟังก์ชันอื่น ๆ จาก การรับภาพที่บริเวณจอภาพ ดังนั้นในส่วนของผลกระทบที่เกิดจากตัวกรองผ่านต่ำจึงอาจละเลยได้ ในกรณีนี้จะได้ แบบจำลองซึ่งมีลักษณะง่ายขึ้น คือเหลือเพียง การเปลี่ยนจาก luminance ไปเป็น ความสว่างเท่านั้นดังรูปที่ ค.3 (ข)



รูปที่ ค.3 แบบจำลองของระบบการมองเห็นภาพสีเดียว

ค.3 การประยุกต์ใช้แบบจำลองระบบการมองเห็น

การนำเอาผลของแบบจำลองระบบการมองเห็นของมนุษย์ไปใช้ สามารถทำได้ทั้งใน spatial domain และ spectral domain แต่โดยมากจะใช้กับวิธีหลัง เพราะอยู่ในรูปของความถี่อยู่แล้ว ทำให้การประยุกต์ทำได้ง่าย

แต่สำหรับ JPEG นำเอาผลที่ได้นี้ไปใช้โดยการใช้ค่าขีดเริ่มของการมองเห็น และควอนไตซ์ค่าความถี่ที่ได้จากการแปลง DCT ด้วยขนาดขั้นเดียวกัน ซึ่งเปรียบเสมือนการให้น้ำหนักสัมประสิทธิความถี่แต่ละความถี่ตามแบบจำลองภาพของตามนุษย์ โดยค่าขีดเริ่มที่ได้จะแตกต่างกันไปตามความถี่ซึ่งเป็นผลมาจากการตอบสนองของตานั่นเอง ค่าขีดเริ่มเหล่านี้ได้จาก

การทดลองที่เกี่ยวกับจิตวิทยาการมองเห็น (Psychovisual) ของผู้เชี่ยวชาญทางประมวลผลภาพ [8] ซึ่งได้ผลเป็นเมตริกซ์ค่าขัดเริ่มของการมองเห็น แยกเป็นเมตริกซ์ค่าขีดเริ่มของการมองเห็นของ Luminance และ เมตริกซ์ค่าขีดเริ่มของการมองเห็นของ Chrominance ซึ่งมีค่าต่างกันดังรูปที่ ค.4

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(ก)

17	18	24	47	66	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(ข)

รูปที่ ค.4 เมตริกซ์ค่าขีดเริ่มของการมองเห็น สำหรับ

(ก) Luminance

(ข) Chrominance

ภาคผนวก ง

TMS32010/TMS320E15

TMS32010 เป็นชิปตัวแรกในตระกูล TMS320 ซึ่งถูกออกแบบมาให้ใช้กับงานทางด้านการประมวลผลภาพโดยเฉพาะ โดยมีความเร็ว และ ความสามารถในการประมวลผลทางคณิตศาสตร์สูงมาก กล่าวคือ TMS32010 สามารถทำงานในโหมดของ Microprocessor ได้เร็วถึง 5 ล้านคำสั่งต่อวินาที โดยมีคำสั่งพิเศษ และ คำสั่งเฉพาะ สำหรับงานทางด้าน DSP เพื่อความสะดวกและรวดเร็ว จนในปัจจุบัน ได้มีชิปในตระกูลนี้ออกมาอีกหลายตัว และถูกใช้งานกันอย่างกว้างขวางในด้าน DSP เช่น การสร้างหุ่นยนต์ การสังเคราะห์ และจดจำเสียงพูด หรือ การควบคุมอุปกรณ์ทางกลต่าง ๆ ซึ่ง TMS320E15 ก็เป็นชิปในตระกูลนี้เช่นกัน

ง.1 <u>สถาปัตยกรรมภายใน (Architecture)</u>

สถาปัตยกรรมภายในของ TMS32010 จะเป็นแบบ Modified Harvard Architecture เพื่อเพิ่มความเร็ว และ ความคล่องตัวในการทำงาน ในสถาปัตยกรรมแบบ Harvard Architecture จะแยกบริเวณส่วนของโปรแกรม กับ ส่วนของข้อมูลออกจากกันโดยเด็ดขาด เพื่อให้ทำงานในลักษณะขนาน (parallel) ได้อย่างเต็มที่ แต่ใน TMS32010 ได้ดัดแปลงสถาปัตยกรรมแบบนี้เล็กน้อย โดยการยอมให้มีการแลกเปลี่ยนข้อมูลระหว่างส่วนของโปรแกรม กับ ส่วนของข้อมูลได้ ทำให้เพิ่มความยืดหยุ่นในการทำงาน เช่น สามารถที่จะย้ายค่าสัมประสิทธิ์ที่ถูกเก็บอยู่ในส่วนของความจำโปรแกรม ไปยังแรมข้อมูล ทำให้ไม่จำเป็นต้องมีรอมแยกต่างหากเพื่อเก็บค่าเหล่านี้ นอกจากนี้ยังทำให้ TMS32010 สามารถใช้คำสั่งประเภท Immediate ได้ และยังสามารถเรียกรูทีย่อยได้จากค่าที่ได้จากการคำนวณได้อีกด้วย

TMS32010 ยังถูกออกแบบมาให้ทำงานบางอย่างด้วยฮาร์ดแวร์ ในขณะที่ตัวประมวลผลอื่น ๆ ทำงานนั้นโดยใช้ซอฟต์แวร์ เช่น การคูณ TMS32010 มีตัวคูณโดยใช้ฮาร์ดแวร์ ซึ่งทำการคูณได้ในเวลา 1 รอบของการทำงาน สามารถเลื่อนข้อมูลก่อนที่จะเข้าสู่ ALU ได้ นอกจากนั้นยังมีฮาร์ดแวร์พิเศษที่สามารถทำให้เรจิสเตอร์พิเศษ (Auxillary register) เพิ่ม หรือ ลดค่าได้โดยอัตโนมัติภายในคำสั่งทำงานนั้น ๆ ด้วย [ง.1]

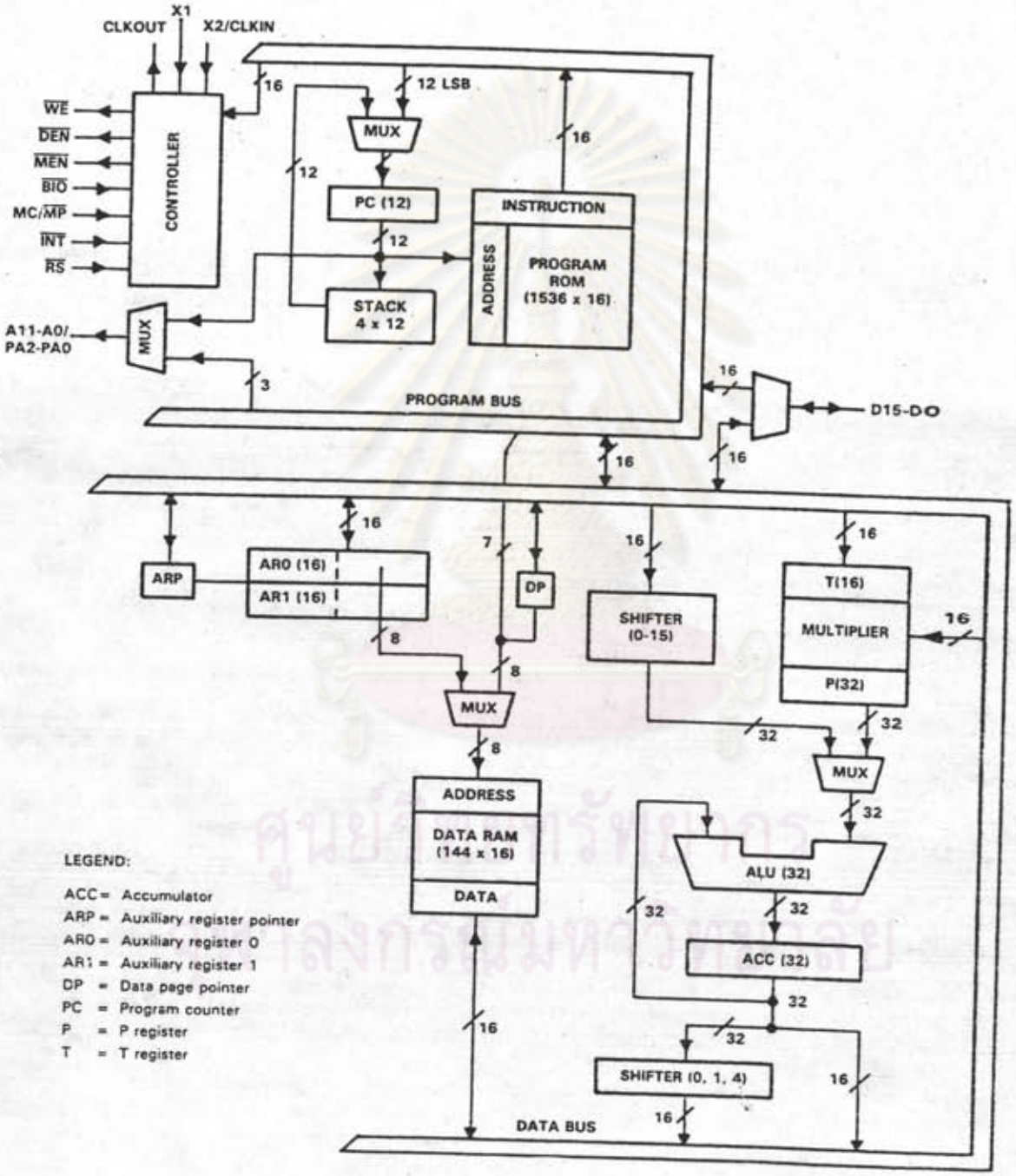
รูปที่ ง.1 แสดงตำแหน่ง และ หน้าที่ของขาต่าง ๆ ของ TMS32010/TMS320E15

PIN NOMENCLATURE

SIGNATURE	I/O	DEFINITION
A11-A0/ PA2-PA0	OUT	External address bus. I/O port address multiplexed over PA2-PA0.
BIO	IN	External polling input for bit test and jump operations.
CLKOUT	OUT	System clock output, ¼ crystal/CLKIN frequency.
D15-D0	I/O	16-bit data bus.
DEN	OUT	Data enable indicates the processor accepting input data on D15-D0.
INT	IN	Interrupt.
MC/MP	IN	Memory mode select pin. High selects microcomputer mode. Low selects microprocessor mode.
MEN	OUT	Memory enable indicates that D15-D0 will accept external memory instruction.
RS	IN	Reset used to initialize the device.
VCC	IN	Power.
VSS	IN	Ground.
WE	OUT	Write enable indicates valid data on D15-D0.
X1	IN	Crystal input.
X2/CLKIN	IN	Crystal input or external clock input.

รูปที่ ง.1 ตำแหน่ง และ หน้าที่ของขาต่าง ๆ ของ TMS32010/TMS320E15

ส่วนรูปที่ ง.2 แสดงให้เห็นสถาปัตยกรรมภายใน ดังมีรายละเอียดดังนี้



รูปที่ ง.2 โครงสร้างของสถาปัตยกรรมภายในของ TMS32010/TMS320E15

1. ALU/Accumulator TMS32010 และ TMS320E15 ประกอบด้วย ALU และ Accumulator ขนาด 32 บิต ซึ่งมีการคำนวณแบบ double precision ALU จะทำงานกับค่าข้อมูลขนาด 16 บิต ซึ่งอาจจะมาจากหน่วยความจำข้อมูล หรือ จากคำสั่งที่มีค่า Immediate และนอกจากจะสามารถทำงานที่เกี่ยวกับเลขคณิตแล้วยังสามารถทำงานประเภท Boolean ในระดับบิตได้อีกด้วย

2. ตัวเลื่อนข้อมูล ใน TMS32010 และ TMS320E15 จะมีตัวเลื่อนข้อมูลแบบ Barrel ที่สามารถเลื่อนข้อมูลได้ตั้งแต่ 0 ถึง 15 บิต ก่อนที่จะถูกบวกเข้ากับ ลบออกจาก หรือ โหลดเข้าสู่ Accumulator ลักษณะของการเลื่อนข้อมูลจะเป็นการขยายเครื่องหมาย (signed extended) และเติมศูนย์ (zero fills) ที่ตำแหน่งบิตที่ถูกเลื่อนไป ส่วนตัวเลื่อนข้อมูลอีกตัวจะใช้เลื่อนข้อมูล 16 บิตบน ก่อนที่จะออกจาก Accumulator ไปเก็บยังแรมข้อมูลได้ 0, 1, 4 บิต ตัวเลื่อนข้อมูลทั้งสองตัวนี้จะมีประโยชน์มากในการสเกลข้อมูล และ การแยกบิต (bit extraction)

3. ตัวคูณขนาน 16 x 16 บิต (16 x 16 bit Parallel Multiplier) สามารถทำการคูณตัวเลข 16 บิตสองตัวเข้าด้วยกันได้ในแบบ 2 complement โดยใช้เวลาเพียง 1 คำสั่งการทำงานของรอบนาฬิกาที่ 200 นาโนวินาทีเท่านั้น โดยตัวตั้งจะถูกโหลดเข้าไปในเรจิสเตอร์ T ก่อนที่จะถูกคูณด้วยตัวคูณจากค่าในหน่วยความจำข้อมูล หรือ จากคำสั่งของการคูณแบบ Immediate (MPYK) ก็ได้ โดยผลลัพธ์ที่ได้จะถูกเก็บไว้ในเรจิสเตอร์ P ขนาด 32 บิต ซึ่งค่าที่อยู่ในเรจิสเตอร์ P นี้จะต้องถูกย้ายไปยัง Accumulator และ ไปเก็บยังหน่วยความจำข้อมูลภายในต่อไป

4. หน่วยความจำโปรแกรม สำหรับ TMS32010 จะมีรอมของหน่วยความจำโปรแกรมที่สามารถโปรแกรมได้ขนาด 1536 คำ (word) ขนาด 16 บิต อยู่ภายใน ซึ่งอาจจะเพิ่มได้ถึง 4096 คำ ทางภายนอกที่ความเร็วสูงสุด หรือ อาจจะใช้หน่วยความจำโปรแกรมทั้ง 4096 คำ ต่ออยู่ภายนอกทั้งหมดก็ได้ โดยการเลือกค่าที่ขา MC/MP คือ

MC - 1536 คำ เป็นรอมภายใน และที่เหลือเป็นรอมภายนอก

MP - ทั้ง 4096 คำ เป็นรอมภายนอกทั้งหมด

ส่วน TMS320E15 จะมีข้อแตกต่างออกไปเล็กน้อย คือ

MC - ทั้ง 4096 คำ เป็นรอมภายในทั้งหมด

MP - เหมือนกับใน TMS32010

5. หน่วยความจำข้อมูล หน่วยความจำข้อมูลของ TMS32010 และ TMS320E15 จะเป็นแรมอยู่ภายในซึ่งไม่สามารถจะเข้าถึงได้จากภายนอก ใน TMS32010 จะมีหน่วยความจำข้อมูลอยู่ทั้งสิ้น 144 คำ ขนาดคำละ 16 บิต ส่วนใน TMS320E15 จะมีหน่วยความจำข้อมูลมากกว่าคือ 256 คำ

6. อินต์พุต/เอาต์พุต (Input/Output - I/O) ใน TMS32010 และ TMS320E15 จะมีอย่างละ 8 พอร์ต ทำให้สามารถส่งผ่านข้อมูลเข้าออกได้ถึง 40 ล้านบิตต่อวินาที เมื่อใช้วิธีการวนตรวจสอบขา $\overline{BIO}$ หรือ $\overline{INT}$ ซึ่งทำให้สามารถใช้หน่วยความจำภายนอกมาทำเป็นส่วนขยายหน่วยความจำข้อมูลเพิ่มได้

7. การอินเตอร์รัปต์/การเรียกรูทีนย่อย (Interrupt/Subroutine) ภายในตัว TMS32010 และ TMS320E15 จะมี stack ขนาด 12 บิตอยู่ 4 ระดับเพื่อใช้ในการเก็บค่าของ program counter ในระหว่างการอินเตอร์รัปต์ หรือ การเรียกใช้รูทีนย่อยการอินเตอร์รัปต์ หรือ การเรียกรูทีนย่อยซ้อนกันเป็นชั้น ๆ จึงทำได้รวมกันไม่เกิน 4 ระดับ ลักษณะของการอินเตอร์รัปต์ของ TMS32010/TMS320E15 ที่เกิดขึ้นจะเป็นลักษณะ maskable ทั้งหมดซึ่งจะควบคุมได้โดยใช้คำสั่ง EIN หรือ DIN

ง.2 ชุดคำสั่ง

ชุดคำสั่งของ TMS32010 และ TMS320E15 ประกอบด้วยคำสั่ง 2 ประเภทใหญ่ ๆ คือ ประเภทที่ทำงานเกี่ยวกับการคำนวณตัวเลขซึ่งใช้งานทางประมวลผลภาพโดยตรง เช่น การบวก, การลบ, การคูณ, การหาร กับ คำสั่งประเภทที่ใช้ในงานทั่วไป เช่น การแยก (branch), การรับเข้า/ส่งออกข้อมูล, การควบคุม เป็นต้น ในคำสั่งประเภทแรกจะมีรอบการทำงานเพียง 1 รอบของนาฬิกา ส่วนคำสั่งประเภทหลังมักจะมีรอบการทำงาน 2 รอบของนาฬิกา และบางส่วนเป็น 3 รอบของนาฬิกา จากคำสั่งทั้ง 2 ประเภทนี้ทำให้ TMS32010

ทำงานได้เร็วถึง 5 ล้านคำสั่งต่อวินาทีในกรณีที่มีคำสั่งประเภทที่ 2 อยู่น้อย

นอกจากนี้ในบางคำสั่งยังสามารถทำการเลื่อนข้อมูลได้ ภายในคำสั่งการทำงานนั้นด้วย การเลื่อนข้อมูลเหล่านี้จะเกิดขึ้นในรอบการทำงานเดียวกันกับคำสั่งการทำงานที่เกิดขึ้น ซึ่งจะมีประโยชน์มากในการสเกลข้อมูลในระหว่างเข้าออกจาก Accumulator

ชุดคำสั่งทั้งหมดของ TMS32010 และ TMS320E15 จะสามารถอ้างอิงแอดเดรส (addressing) ได้ทั้งหมด 3 โหมดด้วยกัน คือ

1. <u>การอ้างอิงแอดเดรสโดยตรง (Direct addressing)</u> ในชุดคำสั่งประเภทนี้ ภายในจะประกอบด้วยค่า pointer ซึ่งชี้ไปยังตำแหน่งของหน่วยความจำข้อมูลใน page นั้น ซึ่งการเลือก page จะถูกกำหนดด้วยค่าภายในเรจิสเตอร์ Data page (DP) ที่มีขนาดเพียง 1 บิต ใน page 0 จะมีหน่วยความจำอยู่ 128 คำ ส่วนใน page 1 จะมีหน่วยความจำเพียง 16 คำ สำหรับ TMS32010 และ 128 คำ สำหรับ TMS320E15 ซึ่งใน page 1 นี้มักจะใช้เป็นที่เก็บค่าที่เกี่ยวข้องกับการอินเตอร์รัปต์ การอ้างอิงแอดเดรสโดยตรงนี้สามารถใช้ได้กับทุกคำสั่งที่ต้องการข้อมูลในการทำงานที่ไม่ใช่ลักษณะ Immediate

2. <u>การอ้างอิงแอดเดรสโดยอ้อม (Indirect addressing)</u> การอ้างอิงแอดเดรสโดยอ้อมนี้จะอาศัยค่า 8 บิตล่างของรีจิสเตอร์พิเศษตัวใดตัวหนึ่งซึ่งถูกกำหนดโดยตัวชี้เรจิสเตอร์พิเศษ (Auxillary register pointer) ที่มีขนาด 1 บิต ซึ่งการอ้างอิงแบบนี้จะสามารถเพิ่ม หรือ ลดค่าภายในเรจิสเตอร์พิเศษได้ โดยอัตโนมัติไปพร้อมกับรอบการทำงานของคำสั่งนั้น

3. <u>การอ้างอิงแอดเดรสแบบ Immediate (Immediate Addressing)</u> นอกจากการอ้างอิงแอดเดรสทั้ง 2 วิธีข้างต้นแล้ว ยังมีชุดคำสั่งอีกกลุ่มที่สามารถแยกเอาข้อมูลที่อยู่ภายในคำสั่งนั้น มาใช้เป็นค่าในการทำงานได้อีกด้วย ซึ่งเรียกว่า การอ้างอิงแอดเดรสแบบ immediate โดยปกติการอ้างอิงแบบนี้จะใช้ในคำสั่งที่เกี่ยวกับการคำนวณทางตัวเลข เช่น การคูณด้วยค่าคงที่, การโหลด Accumulator ด้วยค่าคงที่, การโหลดค่าคงที่ให้กับเรจิสเตอร์พิเศษ เป็นต้น

สำหรับรายละเอียดของคำสั่งประเภทต่าง ๆ และ การใช้งานจะสามารถดูได้จาก [21]-[23]

ภาคผนวก จ

รายละเอียดของส่วนโปรแกรมภาษา C

```
>
/*************************************************/
/*                                               */
/*                   MAIN MENU                   */
/*                                               */
/*                     of                        */
/*                                               */
/*  PROGRESSIVE STILL IMAGE TRANSMISSION SYSTEM  */
/*                                               */
/*************************************************/

#include      <process.h>
#include      <conio.h>

void showheader(void);
void showmenu(void);

main()
{
       int            key;

       do{
          clrscr();
          showheader();
          showmenu();
          key = getch();
          switch(key) {
               case 'c' : /* CALL COMPRESSING MODULE */
               case 'C' : gotoxy(26,22);
                          cputs(".... loading please wait ....\n\r");
                          system("ttr.exe");
                          break;

               case 'r' : /* CALL RECEIVING MODULE */
               case 'R' : gotoxy(26,22);
                          cputs(".... loading please wait ....\n\r");
                          system("rec.exe");
                          break;

               case 't' : /* CALL TRANSMIT MODULE */
               case 'T' : gotoxy(26,22);
                          cputs(".... loading please wait ....\n\r");
                          system("tint.exe");
                          break;
          }
       } while ((key != 'q') && (key != 'Q'));
       clrscr();
}

void showheader(void)
{
     gotoxy(1,2);
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
     cputs("                                                    \n\r");
}
```

PROGRESSIVE STILL IMAGE TRANSMISSION SYSTEM
COMMUNICATION LAB.
Department of Electrical Engineering
CHULALONGKORN UNIVERSITY

```
void showmenu(void)
{
        gotoxy(1,14);
        cputs("                                    (C) ompress mode\n\r");
        cputs("                                    (T) ransmit mode\n\r");
        cputs("                                    (R) etrieve mode\n\r");
        cputs("                                    (Q) uit\n\r");
}
```

```
>

/***********************************************/
/*                                             */
/*            COMPRESSING MENU                 */
/*                                             */
/*    This module is for compressing image     */
/*   following scheme of JPEG recommendation   */
/***********************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <alloc.h>
#include <io.h>
#include <conio.h>
#include <fcntl.h>
#include <mem.h>
#include <math.h>
#include <sys\stat.h>
#include <time.h>
#include <bios.h>
#include "name.h"
#include "pack.h"
#include "intf1.h"
#include "\image\process\screen.h"

#define  SCREEN_POINTER         0xD000
#define  CONTROL_PORT           0x200
#define  SIZE_OF_PIC            0xFFFF
#define  MAXCODELENGTH          21
#define  BIAS                   255
#define  EOB                    0
#define  TRUE                   1
#define  FALSE                  0

/*       CHOICE OF PARAMETER       */

char *ans[2] = {"NO","YES"};
char *pands[2] = {"   Don't  Pause   ", "Pause & Show error"};

void set_bit_rate(void);
int getkey(void);  /* Uses the BIOS to read the next keyboard character */
void show_compress_menu(void);
void filter_sampling(BYTE far *source, int ps);
void interpolate(BYTE far *source, BYTE far *dest, int ps);
void savefile(BYTE *ch);
void reset(BYTE far *t_frame, BYTE far *ch,BYTE far *temp);
void adder14(char *source, BYTE far *op, BYTE far *result, int ps);
void subtractor14(BYTE far *op1, BYTE far *op2,char *dest, int ps);
void adder23(char *source, BYTE far *op, BYTE far *result, int ps);
void subtractor23(BYTE far *op1, BYTE far *op2,char *dest, int ps);
int encode_DC(struct HUFTAB *huf1, char *block, int nblock, int thres_factor);
int encode_AC(struct HUFTAB *huf1, char *block, int nblock);
void find_DCstat (char *block, int freq[], int nblock);
void find_ACstat (char *block, int freq[], int nblock);
void gen_hufftable(int *freq, struct HUFTAB *huf);
void min_max(struct HUFTAB *huf,int *mins, int *maxs);
void find_huffsize(int *freq, struct HUFTAB *huf);
void transform(void);
int coef_decode(int code, int temp);
int coef_code(int coef);
```

```
BYTE Instage(int stage);
int coefsize(int coef);
void showpic(BYTE far *source);
void loadpic(BYTE far *source);
void savepic(BYTE far *tpt);
void scanimage(BYTE far *image,BYTE far *dest,unsigned long  nbyte);
void swapbank();
unsigned long sumsqr(BYTE  *p);
double find_SNR(BYTE *pict,BYTE *t,unsigned long spwr);
void difference(BYTE *src, BYTE *dest);
void command_menu(void);
void writexy(int x,int y,char *s);


/*   GLOBAL VARIABLE  */

volatile int    filecount,error = 0;
BYTE far        *screen, *filebuffer,*fbuf;
int             stage ,pixsi,thres_factor,overhead_bit = 0;
unsigned long   bit_sent = 0,bit_sum = 0;
char            *ad1,*ad2;
unsigned        wc1,wc2;
double          snr[5];
float           bit_require[5] = {0, 0.08, 0.17, 0.5, 1.5};
void            (*adder)(),(*subtractor)();  /* pointer to function that used */
int             ps= 0,csnr=0,opt = 0;

int main()
{
        int                     i,st, pth,iterate;
        int                     nblock, size,freq[513];
        char                    *blockin,*blockout;
        struct HUFTAB           *huf1, *huf2;
        BYTE far                *t_frame, *pict, *temp0, *ch, *c1,*u1,*u2;
        char                    item, key, s[25], name[25], save;
        float                   bps ,error, rate;
        unsigned int            tseg,useg,m;
        unsigned long           addr1,addr2,spwr,wb;
        BYTE                    command=0;
        BUFFER                  cb;

        /*   make pointer to screen    */

        screen = MK_FP(SCREEN_POINTER,0);

        /*   memory allocation for buffer   */

        pict = farmalloc(0xFFFF);
        if (!pict) {
           cputs("   ! Not enough memory for picture storage !\n\r");
           exit(1);
        }

        tseg = FP_SEG(pict);
        m = ((0x0FFD - (0x0FFF & tseg))<<4);
        u1 = farmalloc(m);
        ad1 = farmalloc(0x10008L);

        t_frame = farmalloc(0xFFFF);
        if (!t_frame) {
           cputs("   ! Not enough memory for predictor !\n\r");
           exit(1);
```

```
}

temp0 = farmalloc(0xFFFF);
if (!temp0) {
   cputs("   ! Not enough memory for working area !\n\r");
   exit(1);
}

useg = FP_SEG(temp0);
m = ((0x0FFD - (0x0FFF & useg))<<4);
u2 = farmalloc(m);
ad2 = farmalloc(0x10008L);
farfree(u1);
farfree(u2);
blockin  = ad1 + 8;
blockout = ad2 + 8;
wc1 = wc2 = 0;
addr1 = abs_address(blockin);
addr2 = abs_address(blockout);
blockin  = MK_FP((addr1 >> 4),(addr1 & 0xFFFF));
blockout = MK_FP((addr2 >> 4),(addr2 & 0xFFFF));


ch = farmalloc(0xFFFF);
if (!ch) {
   cputs("   ! Not enough memory for channel buffer !\n\r");
   exit(1);
}

huf1 = farmalloc(sizeof(struct HUFTAB) * 512);
if(!huf1) {
   cputs("   ! Not enough memory for DC hufmann table !\n\r");
   exit(1);
}

huf2 = farmalloc(sizeof(struct HUFTAB) * 512);
if(!huf2) {
   cputs("   ! Not enough memory for AC hufmann table !\n\r");
   exit(1);
}


/*  begin process  */

rate = 1200.0;
filebuffer = ch;
show_compress_menu();
do {
   command_menu();
   item = getche();
   switch (item) {
        case 'L' : /* load picture */
        case 'l' : loadpic(pict);
                   showpic(pict);
                   if(csnr) spwr=sumsqr(pict);  /* compute image power */
                   break;

        case 'C' : /* compress picture */
        case 'c' : clearbuffer();
                   reset(t_frame,ch,temp0);
                   c1 = filebuffer;
                   cb.b = 0L;cb.s=0;
```

```
fullscreen;
writexy(28,12,"          ");
writexy(28,13,"          ");
writexy(28,14,"          ");

/* begin compress */
for(stage = 1;stage < 5;stage++) {
   infoscreen;
   clrscr();
   movmem(pict,temp0, 0xFFFF);
   pixsi = 1;
   st = 3 - stage;

   /* sampling and filtering for 3 stage */
   while (st > 0) {
      pixsi *= 2;
      filter_sampling(temp0, pixsi);
      st--;
   }
   nblock = (1024/pixsi/pixsi);
   command = Instage(stage);
   pth = thres_factor;
   iterate = 1;
   bps = bit_require[stage];

   /* subtract due to stage */
   (*subtractor)(temp0, t_frame, blockin, pixsi);
   do {
      movmem(blockin,blockout,wc2);
      *(blockout + wc2) = *(blockin + wc2);
      set_DMA_mode();
      DMA_setup(addr2,wc2,addr2,wc2);
      cputs("set_channel o.k.....\r");
      TMS_setup(command,(thres_factor - 128));
      transform();
      cputs(" process dc.....\r");
      resetbuffer(c1);
      setbuffer(cb);
      find_DCstat(blockout, freq, nblock);
      gen_hufftable(freq, huf1);
      overhead_bit = encode_DC(huf1,blockout,nblock,thres_factor);
      cputs(" process ac.....\r");
      find_ACstat(blockout, freq, nblock);
      gen_hufftable(freq, huf2);
      overhead_bit += encode_AC(huf2,blockout,nblock);
      bit_sent = filebuflength();
      cprintf("stage %d iteration %d -> ",stage,iterate);
      cprintf("%-4.3f bps ",(bit_sent)/65536);
      cprintf("threshold factor : %-3d\r",thres_factor);
      iterate++;
      error = fabs(bps - ((bit_sent)/65536));
      st = (int)(thres_factor*pow(((double)(bit_sent)/65536/bps),1.5));
      /* find next threshold factor */
      if(iterate > 1) {
         if(stage == 1) {
            if(st < 25)
               st = 25;
         } else {
            if(st < 5)
               st = 5;
         }
         if(st > 128)
```

```
                st = 128;
                if((st <= pth) && (pth < thres_factor))
                   st = (pth + thres_factor) / 2;
                if((st >= pth) && (pth > thres_factor))
                   st = (pth + thres_factor) / 2;
             }
             pth = thres_factor;
             thres_factor = st;
          } while ((st != pth) && (error > (.01 * bps)));
          bit_sum += bit_sent;
          cprintf("overhead %d   bit sent %lu   ",overhead_bit,bit_sent);
          cprintf('total %lu\n\r",bit_sum);

          /* move buffer pointer */
          cl += (filebuflength()/8);
          readbuffer(&cb);
          /* begin inverse transform */
          command |= 0x04;
          set_DMA_mode();
          DMA_setup(addr2,wc2,addr2,wc2);
          cputs(" set channel o.k.\r");
          TMS_setup(command,thres_factor-128);
          transform();
          (*adder)(blockout, t_frame, temp0, pixsi);

          /* interpolate predictor */
          if(pixsi != 1) {
             interpolate(temp0, t_frame, pixsi);
             pixsi /= 2;
          }
          else
             memmove(t_frame,temp0,0xFFFF);
          if(pixsi != 1)
             interpolate(t_frame,temp0,2);
          else
             memmove(temp0,t_frame,0xFFFF);
          showpic(temp0);

          /* do option */
          if(opt) {
             if(csnr) {
                cputs(" computing SNR....\r");
                snr[stage]=find_SNR(pict,temp0,spwr);
                cputs(" finish computing SNR \n\r");
             }
             if(ps){
                difference(pict,temp0);
                cputs(" press any key to see error\n\r");
                getch();
                showpic(temp0);
                cputs(" press any key to continue\n\r");
                getch();
                swapbank();
             }
          }
          fullscreen;
          gotoxy(28,12);cprintf("%d",stage);
          gotoxy(28,13);cprintf("%-5d",bit_sum/8);
          gotoxy(28,14);cprintf("%-4.3f",(double)bit_sum/65536);
          if(ps)
             gotoxy(28,5);cprintf("%-4.2f",snr[stage]);
       }
```

```
                        infoscreen;
                        clrscr();
                        cputs(" finish compressing picture ");
                        flushbuffer();
                        break;

                case 'S; : /* save code if want */
                case 's' : savefile(ch);
                        break;

                case 'O' : /* set option */
                case 'o' : opt = get_option();
                        break;

                case 'B' : /* set bit rate for each  stage */
                case 'b' : set_bit_rate();
                        break;

                case ' ' : swapbank();
                        break;
           };
        } while (!((item == 'x') || (item == 'X')));
}

void set_bit_rate(void)  /* set bit rate at each stage if don't want to follow JPEG */
{
        int             s,k='n';
        char            t[20];

        cputs("\n\r Default ? (y/n)");
        k = getch();
        if(k == 'y'){
           bit_require[1] = 0.08;
           bit_require[2] = 0.17;
           bit_require[3] = 0.50;
           bit_require[4] = 1.50;
        }else {
           for(s = 1;s <= 4;s++) {
                gotoxy(1,9);
                cputs  ("\r                              ");
                cprintf("\r stage %d = %4.3f :",s,bit_require[s]);
                (void)gets(t);
                if(strcmp(t,"")) {
                   bit_require[s] = atof(t);
                }
           }
        }
}

int getkey(void)  /* Uses the BIOS to read the next keyboard character */
{
     int key, lo, hi;

     key = bioskey(0);
     lo = key & 0x00FF;
     hi = (key & 0xFF00) >> 8;
     return((lo == 0) ? hi + 256 : lo);
} /* getkey */

void difference(BYTE *src, BYTE *dest)  /* find error picture power */
{
        long int                        i;
```

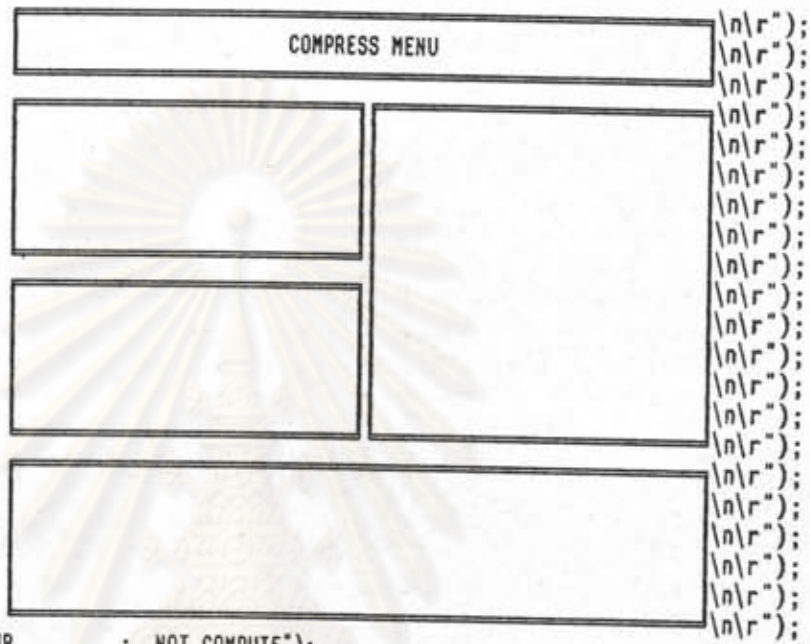
```
        int                             temp;

        for(i = 0;i < 0xFFFF;i++) {
           temp = abs(*src++ - *dest);
           *dest++ = (temp > 255)? 255 : temp;
        }
}

void show_compress_menu(void)
{
        fullscreen;
        clrscr();
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        cputs("
        writexy(13,5,"SNR          :  NOT COMPUTE");
        writexy(13,6,"Don't Pause");
        writexy(13,11,"FILENAME    :");
        writexy(13,12,"STAGE       :");
        writexy(13,13,"TOTAL BYTE  :");
        writexy(13,14,"BIT RATE    :");
}

int get_option(void)  /* set option (see below) */
{
        static int              o = FALSE;    /* option changed? */
        static int              p = 0;        /* cursor movement */
        int                     key;          /* response        */

        comdscreen;
        clrscr();
        cprintf("  compute SNR : %s\n\r",ans[csnr]);
        cprintf("  %s",pands[ps]);
        fullscreen;
        do {
           switch(p) {
                case 0 : gotoxy(58,5);    /* compute SNR of reconstructed image */
                         key = get_choice(&csnr,ans,2);
                         if(csnr)
                              o = TRUE;
                         break;
                case 1 : gotoxy(44,6);    /* pause and show error picture */
                         key = get_choice(&ps,pands,2);
                         if(ps)
```



```
                        o = TRUE;
                    break;
            }
            switch(key) {
                case UPKEY   : p--;
                               if(p < 0)
                                   p = 1;
                               break;
                case DOWNKEY : p++;
                               if(p > 1)
                                   p = 0;
                               break;
            }
        } while(key != ESC);
        if(csnr)
            writexy(28,5,"          ");
        else
            writexy(28,5,"NOT COMPUTE");
        writexy(13,6,pands[ps]);
        return o;
}

int get_choice(int *c,char **w,int n)   /* get setting of the parameter */
{
        int                     x,y;    /* position of cursor */
        int                     k;      /* key pressed */

        x = wherex(); y = wherey();
        textbackground(WHITE);
        textcolor(BLACK);
        cprintf("%-10s",w[*c]);
        while(((k = getkey()) != UPKEY) && (k != DOWNKEY) && (k != ESC)) {
            switch(k) {
              case LEFTKEY  : (*c)++;
                              if(*c == n)
                                 *c = 0;
                              break;
                case RIGHTKEY : (*c)--;
                              if(*c  < 0)
                                 *c = n - 1;
                              break;
            }
            gotoxy(x,y);
            cprintf("%-10s",w[*c]);
        }
        normvideo();
        gotoxy(x,y);
        cprintf("%-10s",w[*c]);
        return k;
}

void writexy(int x, int y, char *s)   /* write string at position X,Y */
{
        gotoxy(x,y);
        cputs(s);
}

void command_menu(void)
{
        comdscreen;
        clrscr();
        cputs("   1. load picture\n\r");
```

```
        cputs("   c. compress picture\n\r");
        cputs("   o. set option\n\r");
        cputs("   b. set bit rate\n\r");
        cputs("   s. save code\n\r");
        cputs("   x. exit\r\n");
        cputs("   space     swapbank\n\r");
        cputs("                      \r");
        cputs(" choose item: ");

}

unsigned long sumsqr(BYTE  *p)     /* find image signal power */
{
        long                    i;
        unsigned                d;
        unsigned long           temp = 0;

        infoscreen;
        clrscr();
        cputs("   please wait for calculation ...  \n\r");
        for(i = 0;i <= 65536;i++,p++) {
           d = *p++;
           temp += d * d;
        }
        cputs("         finish calculation          ");
        return temp;
}

double find_SNR(BYTE *pict,BYTE *t,unsigned long signal)
{
        long                    i;
        long int                temp;
        unsigned long           noise=0;

        for(i = 0;i <= 65535;i++) {
           temp = ((long)(*pict++) - *t++);
           noise += (temp * temp);
        }
        return (log10((double)signal/noise) * 10);
}

int encode_DC(struct HUFTAB *huf1, char *block, int nblock, int thres_factor)
{ /*  encode DC coefficient */
        int                     i,j, diff,overhead_bit=0;
        int                     temp=0;

        pack(nblock,11);
        pack(thres_factor,8);
        for (j = 0;j < 512;j++) {
           if(huf1[j].sizes == 0)
                continue;
           pack(huf1[j].sizes, 4);
           pack(j, 9);
           pack(huf1[j].codes, huf1[j].sizes);
           overhead_bit += (huf1[j].sizes + 13);
        }
        pack(0,4);

        for(i = 0;i < nblock;i++) {
           diff = (int)*block - temp;
           temp += diff;
           pack(huf1[diff+8IAS].codes,huf1[diff+8IAS].sizes);
```

```
            block += 64;
        }
        return (overhead_bit + 23);
}

int encode_AC(struct HUFTAB *huf2,char *block, int nblock)
{ /* encode AC coefficient */
        int                     i,j, diag, overhead_bit=0;
        int                     temp,distemp, run;

        for (j = 0;j < 512;j++) {
           if(huf2[j].sizes != 0) {
                pack(huf2[j].sizes, 4);
                pack(j, 9);
                pack(huf2[j].codes, huf2[j].sizes);
                overhead_bit += (huf2[j].sizes + 13);
           }
        }
        pack(0,4);

        for(i = 0;i < nblock;i++) {
           block++;
           distemp = 0;
           for (j = 1;j < BLOCKSIZE;j++){
                if(*block != 0) {
                   temp = coefsize(*block);
                   run = (distemp << 3) + temp;
                   pack(huf2[run].codes,huf2[run].sizes);
                   pack(coef_code(*block++),temp);
                   distemp = 0;
                   continue;
                }
                distemp++;
                block++;
           }
           pack(huf2[EOB].codes,huf2[EOB].sizes);
        }
        return(overhead_bit+4);
}

void find_DCstat (char *block, int freq_dc[] , int nblock)
{  /* find DC statistic for construction of HUFMANN table */
        int                     i,j,temp,diffdc;

        memset(freq_dc,0,(sizeof(int) * 512));
        temp = 0;
        for (i = 0;i < nblock;i++) {
           diffdc = *block - temp;
           temp += diffdc;
           freq_dc[diffdc + BIAS]++;
           block += 64;
        }
}

void find_ACstat (char *block,int freq_ac[], int nblock)
{  /* find AC statistic for construction of HUFMANN table */
        int                     i,j,temp,distemp,diag;

        memset(freq_ac,0,(sizeof(int) * 512));
        for(i = 0;i < nblock;i++) {
           distemp = 0;
           block++;
```

```
            for (j = 1;j < BLOCKSIZE;j++){
                if(*block != 0) {
                   freq_ac[(distemp << 3) + coefsize(*block++)]++;
                   distemp = 0;
                   continue;
                }
                distemp++;
                block++;
            }
            freq_ac[EOB]++;
        }
}

void gen_hufftable(int *freq, struct HUFTAB huf[])
{  /* generate HUFMANN table from statistic */
        int                     i,j,si, point,code;
        int                     bits,minsi,maxsi;

        memset(huf,0,(sizeof(struct HUFTAB) * 512));
        find_huffsize(freq, huf);
        min_max(huf,&minsi,&maxsi);

        code = 0;i = 0;
        si = bits = minsi;
        while(huf[i].sizes != bits) i++;
        huf[i++].codes = 0;
        do {
           while((huf[i].sizes != bits) && (bits <= maxsi)) {
                if(i < 512)
                   i++;
                else {
                   i = 0;
                   bits++;
                }
           }

           if(bits <= maxsi) {
                while((code & 1) != 0) {
                   code >>= 1;
                   si--;
                }
                code |= 1;
                while (si != bits) {
                   code <<= 1;
                   si++;
                }
                huf[i++].codes = code;
           }
        } while (bits <= maxsi);
}

void min_max(struct HUFTAB *huf,int *mins, int *maxs)
{  /* find minimum and maximum */
        int                     i,j;

        *mins = 21;
        *maxs = 0;
        for(i = 0;i < 512;i++) {
           if(huf[i].sizes != 0) {
                if(huf[i].sizes < *mins)
                   *mins = huf[i].sizes;
                if(huf[i].sizes > *maxs)
```

```
                *maxs = huf[i].sizes;
            }
        }
}

void find_huffsize(int *freq, struct HUFTAB *huf)
{  /* find huffman codesize of each code */
        register int            i,min1,min2;
        int                     *others;

        others = farmalloc(1024);
        memset(others,0xFF,1024);
        freq[512] = 32767;
        do {
           /* find 2 minimum of coefficient */
           min1 = 0; min2 = 512;
           while(freq[min1] == 0) min1++;
           for(i = min1 + 1;i < 512;i++) {
                if(freq[i] == 0)
                   continue;
                if(freq[i] < freq[min2]) {
                   if(freq[i] < freq[min1]) {
                        min2 = min1;
                        min1 = i;
                        continue;
                   }
                   min2 = i;
                }
           }
           if(min2 == 512)
                break;

           freq[min1] += freq[min2];
           freq[min2] = 0;
           huf[min1].sizes++;
           while (others[min1] >= 0){
                min1 = others[min1];
                huf[min1].sizes++;
           } ;

           others[min1] = min2;
           huf[min2].sizes++;
           while (others[min2] >= 0) {
                min2 = others[min2];
                huf[min2].sizes++;
           };
        } while(min2 != 512);
        farfree(others);
}

void reset(BYTE far *t_frame, BYTE far *ch,BYTE far *temp0)
{
        memset(t_frame, 0x80, 0xFFFF);
        memset(temp0,0,0xFFFF);
        memset(ch,0,0x7FFF);
        bit_sum = 0;
        thres_factor = 50;
        overhead_bit = 0;
}

/* adder and subtractor for the first and fourth stage */
```

```
void adder14(char *src, BYTE far *op, BYTE far *dest, int ps)
{
        int                     i,j,k,l,bpl;
        int                     temp;
        BYTE far                *p, *q, *r, *x, *y, *z;

        bpl = 32/ps;
        for(i = 0;i < bpl;i++) {
           r = op;
           z = dest;
           for(j = 0;j < bpl;j++) {
                p = r;
                x = z;
                for(k = 0;k < 8;k++) {
                   q = p;
                   y = x;
                   for(l = 0;l < 8;l++) {
                        temp = *src++ + *q++;
                        temp = (temp > 255)? 255:temp; /* check overflow */
                        *y++ = (temp < 0)? 0:temp;  /* check underflow  */
                   }
                   p += 256;
                   x += 256;
                }
                r += 8;
                z += 8;
           }
           op += 2048;
           dest += 2048;
        }
}

void subtractor14(BYTE far *op1,BYTE far *op2, char *dest, int ps)
{
        int                     i,j,k,l,temp,bpl;
        BYTE far                *p, *q, *r, *x, *y, *z;

        bpl = 32/ps;
        for(i = 0;i < bpl;i++) {
           r = op1;
           z = op2;
           for(j = 0;j < bpl;j++) {
                p = r;
                x = z;
                for(k = 0;k < 8;k++) {
                   q = p;
                   y = x;
                   for(l = 0;l < 8;l++){
                        *dest++ = (*q++ - *y++);
                   }
                   p += 256;
                   x += 256;
                }
                r += 8;
                z += 8;
           }
           op1 += 2048;
           op2 += 2048;
        }
}

/* adder and subtractor of the second and third stage */
```

```
void adder23(char *src, BYTE far *op, BYTE far *dest, int ps)
{
        int                     i,j,k,l,bpl;
        int                     temp;
        BYTE far                *p, *q, *r, *x, *y, *z;

        bpl = 32/ps;
        for(i = 0;i < bpl;i++) {
           r = op;
           z = dest;
           for(j = 0;j < bpl;j++) {
                p = r;
                x = z;
                for(k = 0;k < 8;k++) {
                   q = p;
                   y = x;
                   for(l = 0;l < 8;l++) {
                        temp = (*src++ * 2) + *q++;
                        temp = (temp > 255)? 255:temp; /* check overflow */
                        *y++ = (temp < 0)? 0:temp; /* check underflow */
                   }
                   p += 256;
                   x += 256;
                }
                r += 8;
                z += 8;
           }
           op += 2048;
           dest += 2048;
        }
}

void subtractor23(BYTE far *op1,BYTE far *op2, char *dest, int ps)
{
        int                     i,j,k,l,temp,bpl;
        BYTE far                *p, *q, *r, *x, *y, *z;

        bpl = 32/ps;
        for(i = 0;i < bpl;i++) {
           r = op1;
           z = op2;
           for(j = 0;j < bpl;j++) {
                p = r;
                x = z;
                for(k = 0;k < 8;k++) {
                   q = p;
                   y = x;
                   for(l = 0;l < 8;l++) {
                        *dest++ = ((int)*q++ - *y++)/2;
                   }
                   p += 256;
                   x += 256;
                }
                r += 8;
                z += 8;
           }
           op1 += 2048;
           op2 += 2048;
        }
}
```

```
void interpolate(BYTE far *source, BYTE far *dest, int ps)
{  /* interpolate to twice of input image */
        int                     i,j,fsize;
        BYTE far                *a, *b, *c, *d, *l1, *l2, *src, *de;

        fsize = 256/ps;
        src = source;
        de  = dest;
        for(i=0;i < fsize-1;i++) {
           a = src;
           b = a + 1;
           c = src + 256;
           d = c + 1;
           l1 = de;
           l2 = de + 256;
           for(j=0;j < fsize-1;j++) {
                *l1++ = *a;
                *l1++ = ((int)*a + *b)/2;
                *l2++ = ((int)*a + *c)/2;
                *l2++ = ((int)*a++ + *b++ + *c++ + *d++) / 4;
           }
           *l1++ = *a;
           *l1 = *a;
           *l2++ = ((int)*a + *c)/2;
           *l2 = *(l2-1);
           src += 256;
           de += 512;
        }
        l1 = de, l2 = de + 256;
        a = src, b = a + 1;
        for(j=0;j < fsize -1;j++) {
           *l1++ = *l2++ = *a;
           *l1++ = *l2++ = ((int)*a++ + *b++)/2;
        }
        *l1 = *(l1+1) = *l2 = *(l2+1) = *a;
}

void filter_sampling(BYTE far *source, int ps)
{  /* filter and sampling */
        int             i,j,sum,fsize;
        BYTE far        *p,*q,*r,*s,*t;

        fsize = 256/ps;
        r = source;
        for (i = 0;i < (2*fsize);i++) {
           p = q = r;
           sum = *q++ * 5;
           sum += *q;
           *p++ = (BYTE) (sum/6);
           for(j = 1;j < fsize;j++) {
                sum = *q++;
                sum += ((int)*q++ * 4);
                sum += *q;
                *p++ = (BYTE)(sum/6);
           }
           r += 256;
        }

        p = q = t = source;
        r = t + 256;
        for(j = 0;j < fsize;j++)
           *p++ = (BYTE)((((int)*q++ * 5) + *r++) / 6);
```

```
        memset(p,0,(256-fsize));
        t += 256, source += 256;
        for(i = 1;i < fsize;i++) {
           p = source;
           q = t;
           r = t + 256;
           s = t + 512;
           for(j = 0;j < fsize;j++) {
                *p++ = (BYTE)(((int)*q++ + ((int)*r++ * 4) + *s++) / 6);
           }
           memset(p,0,(256-fsize));
           t += 512;
           source += 256;
        }
        memset(source, 0, (256 - fsize) * 256);
}

void showpic(BYTE far *source)
{
        movmem(source, screen, 0xFFFF);
        swapbank();
}


int coef_decode(int code, int temp)
{
        int             val,sign;

        sign = code >> (temp - 1);
        val = (1 << (temp - 1)) | code;
        if(sign == 1)
           val = -val;
        return(val);
}

int coef_code(int coef)
{
        int             val,sign,mask;

        sign = (coef < 0)? 1: 0;
        coef = abs(coef);
        val = coef;
        if ((val>>=1) == 0)
           mask = 0;
        else if ((val>>=1) == 0)
           { mask = 1; sign <<= 1; }
        else if ((val>>=1) == 0)
           { mask = 3; sign <<= 2; }
        else if ((val>>=1) == 0)
           { mask = 7; sign <<= 3; }
        else if ((val>>=1) == 0)
           { mask = 15;sign <<= 4; }
        else if ((val>>=1) == 0)
           { mask = 31;sign <<= 5; }
        else
           { mask = 63;sign <<= 6; }
        return((coef & mask) + sign);
}

int coefsize(int coef)
{
        coef = abs(coef);
```

```
        if ((coef>>=1) == 0)
           return(1);
        else if ((coef>>=1) == 0)
           return(2);
        else if ((coef>>=1) == 0)
           return(3);
        else if ((coef>>=1) == 0)
           return(4);
        else if ((coef>>=1) == 0)
           return(5);
        else if ((coef>>=1) == 0)
           return(6);
        else
           return(7);
}


void transform(void)  /* begin transform */
{
        (void)inportb(DMA_STATUS);
        open_DMA_gate();
        clear_mask_reg();
        while(((inportb(DMA_STATUS) & 0x08) == 0) && !kbhit());
        close_DMA_gate();
        set_mask_reg();
        cputs(" transform complete.....\r");
}


void swapbank()
{
        outportb(CONTROL_PORT,0xC7);
}

BYTE    Instage(int stage)  /* give some parameter of stage */
{
        BYTE    b;

        if(stage == 1)
           {b = 0x00;wc1 = wc2 = 0x0FFF;pixsi = 4;
            adder = adder14;subtractor = subtractor14;}
        else if(stage == 2)
           {b = 0x01;wc1 = wc2 = 0x3FFF;pixsi = 2;
            adder = adder23;subtractor = subtractor23;}
        else if(stage == 3)
           {b = 0x02;wc1 = wc2 = 0xFFFF;pixsi = 1;
            adder = adder23;subtractor = subtractor23;}
        else
           {b = 0x02;wc1 = wc2 = 0xFFFF;pixsi = 1;
            adder = adder14;subtractor = subtractor14;}
        return b;
}

void savefile(BYTE *ch)  /* save code to hard disk */
{
     char              s[40],t[40]="data\\";   /* path of data directory */
     unsigned int      bytesent,count; /* count of save byte */
     BYTE              *m;
     int               handle,i;

     cputs("\n\r Enter file number :");
     (void)gets(s);
```

```
    if (strcmp(s,"")) {  /* check file s */
        strcat(t,s);
      i = 'Y';
        handle = _open(t,O_WRONLY | O_BINARY);
      if (handle != -1) {
          cprintf("\rOverwrite %s (Y/N)",s);
        do
              i = getche();
        while((i != 'y') && (i != 'Y') && (i != 'n') && (i != 'N'));
      }
      else {
        handle = creat(s,S_IREAD | S_IWRITE);
      }
        if ((i == 'y') || (i == 'Y')) {
            infoscreen;
            clrscr();
            cputs("   saving file.....");
            bytesent = (unsigned int) (bit_sum /8);
            if ((bit_sent % 8) != 0) bytesent++;
            bytesent+=3;
            _write(handle,&bytesent,sizeof(unsigned int));
            count = _write(handle,(BYTE far *)ch,bytesent);
            close(handle);
            cputs("\n\r saving o.k. ");
      }
    }
}

void loadpic(BYTE far *source)  /* load image to compress */
{
      char          s[40],t[40]="..\\pic\\";  /* path of picture database */
      int           handle;

      cputs("\r Enter filename to load\n\r  : ");
      (void)gets(s);
      infoscreen;
      clrscr();
      if (strcmp(s,"")) {
        strcat(t,s);
        strcat(t,".pic");
        handle = _open(t,O_RDONLY | O_BINARY);
        if (handle == -1) {
            cprintf("\007 file %s not found\n",s);
            getch();
        } else {
            cputs("    loading please wait .....\r");
            _read(handle,(BYTE far*)source,SIZE_OF_PIC);
            close(handle);
            cputs("    loading o.k.          ");
        }
      }
      fullscreen;
      gotoxy(28,11);
      cputs("        ");
      gotoxy(28,11);
      cprintf("%s",s);
}

void savepic(BYTE far *tpt)  /* save image to hard disk */
{
    char            s[40],t[40]="..\\pic\\";  /* path of picture storage */
    unsigned int    i;
```

```
        int             handle,err;

        cputs("\rEnter file number :");
        (void)gets(s);
        if (strcmp(s,"")) {
           strcat(t,s);
           strcat(t,".pic");
           i = 'Y';
           handle = _open(t,O_WRONLY | O_BINARY);
           if (handle != -1) {
                cprintf("\rOverwrite %s (Y/N)",s);
                do
                    i = getche();
                while((i != 'y') && (i != 'Y') && (i != 'n') && (i != 'N'));
           } else {
                handle = creat(t,S_IREAD | S_IWRITE);
           }
           if ((i == 'y') || (i == 'Y')) {
                err = _write(handle,tpt,0xFFFF);
                if(err == -1) {
                   cputs(" write error\n\r");
                   getch();
                }
                close(handle);
           }
        }
}
```

```
>

/*************************************************/
/*                                               */
/*              RECEIVE MODULE                   */
/*                                               */
/*    This module is to retrieve image from      */
/*    sources that will be LOCAL or REMOTE       */
/*************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <alloc.h>
#include <io.h>
#include <conio.h>
#include <dos.h>
#include <fcntl.h>
#include <mem.h>
#include <math.h>
#include <sys\stat.h>
#include <time.h>
#include "name.h"
#include "pack.h"
#include "intpoll.h"
#include "intfl.h"
#include "comm.h"

#define  CONTROL_PORT          0x200
#define  SCREEN_POINTER        0xD000
#define  BIAS                  255
#define  EOB                   0
#define  TRUE                  1
#define  FALSE                 0

void show_rec_menu(void);
void getDC(struct HUFTAB *huftable,int *dcbuf,int noblk);
void getAC(struct HUFTAB *huftable,char *line,int bpl);
void gettable(struct HUFTAB *huftable);
int getcommand(void);
int coef_decode(int code, int temp);
void savepic(BYTE far *source,int *fn,char *ext,unsigned int fs);
void reset(BYTE far *screen,int *thres_factor,struct HUFTAB *huftable);
void swapbank();
void adder14(char *line,BYTE far *frame,int bpl);
void adder23(char *line,BYTE far *frame,int bpl);
void update_line(void);
void fill_DC(char *line,int *dc,int bpl);
void writexy(int x,int y,char *s);
void showpic(BYTE far *frame);
void set_mode(void);

/*  GLOBAL VARIABLE   */

volatile int    filecount = -1,error;
int             stage = 1,pixsi,command,bit_sent,mode = FALSE;
BYTE far        *scrptr,*updptr,*intptr,*addptr,*filebuffer,*screen;
char            *line0,*line1,*linein;
void            (*addline_to_predictor)();
int             filesize;

int main()
```

```
{
   int                  i,j,k,r,quit,run,handle,fileno;
   int                  nblock, thres_factor,st,bpl;
   struct HUFTAB        *huftable; /* hufmann table */
   BYTE far             *r_frame;
   char                 item, s[40] = "", t[25],*wline;
   int                  *dcbuf,oneline;
   unsigned             wc;
   unsigned char        b;
   char                 *at,*au,*av;
   unsigned long        ada,adb,adc,wad;
   float                ta[5],t1,t2;

   /*  make pointer to screen  */

   screen = MK_FP(SCREEN_POINTER,0);

   /*  allocate memory   */

   dcbuf= farmalloc(sizeof(int) * 1024);
   if(!dcbuf) {
      cputs("   ! Not enough memory for DC buffer !\n\r");
      exit(1);
   }

   r_frame = farmalloc(0xFFFF);
   if (!r_frame) {
      cputs("   ! Not enough memory for r_frame !\n\r");
      exit(1);
   }

   filebuffer = farmalloc(25000L);
   if (!filebuffer) {
      cputs("   ! Not enough memory for filebuffer !\n\r");
      exit(1);
   }

   huftable = farmalloc(sizeof(struct HUFTAB)*512);
   if(!huftable) {
      cputs("   ! Not enough memory for hufmann table !\n\r");
      exit(1);
   }

   at = farmalloc(2048);
   ada = abs_address(at);
   if((ada & 0x0FFFFL) > 0xF800L) {
      line0 = farmalloc(2048);
      if(!line0) {
          cputs("   ! Not enough memory for 1st line buffer !\n\r");
          exit(1);
      }
      farfree(at);
   } else
      line0 = at;
   au = farmalloc(2048);
   adb = abs_address(au);
   if((adb & 0x0FFFFL) > 0xF800L) {
      line1 = farmalloc(2048);
      if(!line1) {
          cputs("   ! Not enough memory for 2nd line buffer !\n\r");
          exit(1);
      }
```

```
   farfree(au);
} else
   line1 = au;
av = farmalloc(2048);
adc = abs_address(av);
if((adb & 0x0FFFFL) > 0xF800L) {
   linein = farmalloc(2048);
   if(!linein) {
        cputs("   ! Not enough memory for linein buffer !\n\r");
        exit(1);
   }
   farfree(av);
} else
   linein = av;

ada = abs_address(linein);
adb = abs_address(line0);
adc = abs_address(line1);

/*  now begin process  */

thres_factor = 50;
clearscreen(screen);
show_rec_menu();
do {
   comdscreen;
   clrscr();
   cputs("  a. retrieving picture\n\r");
   cputs("  c. communication setup\n\r");
   cputs("  r. reset receiver\n\r");
   cputs("  m. set mode\n\r");
   cputs("  s. save received picture\n\r");
   cputs("  t. time used\n\r");
   cputs("  x. exit\n\r");
   cputs("                  \r");
   cputs(" select one : ");
   item = getche();
   switch(item) {
        case 'A' :  /*  retrieving image  */
        case 'a' : memset(r_frame,0x80,0xFFFF);
                   clearscreen(screen);
                   clearbuffer();
                   resetbuffer(filebuffer);
                   quit = 0;
                   cputs("\r enter file number : ");
                   scanf("%d",&fileno);
                   cputs("\r  press 'q' to quit\n\r");
                   cputs("  press 'p' to pause");
                   fullscreen;
                   gotoxy(27,12);
                   cprintf("%-5d",fileno);
                   if(mode) {
                      send_fileno(fileno);
                      filesize = receive_data();
                   }else {
                      inportb(0x3FD);
                      error = 0;
                      filesize = loadcode(itoa(fileno,s,10),filebuffer);
                   }
                   if(filesize == -1)  break;

                   /* begin retrieve */
```

```
t1 = clock()/CLK_TCK;

/* clear DMA status */
(void)inportb(DMA_STATUS);
for(stage = 1;stage <= 4;stage++){
   fullscreen;
   gotoxy(27,14);
   cprintf("%-3d",stage);
   infoscreen;
   clrscr();
   nblock = extract(11);
   thres_factor = extract(8);
   cprintf(" nblock %d   factor %d\n\r",nblock,thres_factor);
   scrptr = screen;
   updptr = intptr = addptr = r_frame;
   gettable(huftable);
   pixsi = Instage(stage);
   oneline = (256 * 8 * pixsi);
   bpl = 32/pixsi;
   wc = (2048 / pixsi)-1;
   getDC(huftable,dcbuf,nblock);
   gettable(huftable);
   set_mask_reg();
   set_DMA_mode();
   TMS_setup(command,thres_factor-128);
   open_DMA_gate();
   cprintf("start processing\n\r");

   /* process first line */
   memset(linein,0,2048);
   fill_DC(linein,&dcbuf[0],bpl);
   getAC(huftable,linein,bpl);
   wad = adb;  /* set input address to b line */
   DMA_setup(ada,wc,wad,wc);
   clear_mask_reg();
   /* wait if TMS not finish */
   while((((b = inportb(DMA_STATUS)) & 0x08) == 0) && !kbhit());

   /* process second line */
   memset(linein,0,2048);
   fill_DC(linein,&dcbuf[bpl],bpl);
   getAC(huftable,linein,bpl);
   wad = adc; /* set input address to c line */
   DMA_setup(ada,wc,wad,wc);
   clear_mask_reg();
   wline = line0; /* set output address to line0 */
   /* get first line result to continue processing */
   (*addline_to_predictor)(wline,addptr,bpl);
   addptr += oneline;
   /* wait if TMS not finish yet */
   while((((b = inportb(DMA_STATUS)) & 0x08) == 0) && !kbhit());

   /* process third line */
   memset(linein,0,2048);
   fill_DC(linein,&dcbuf[2*bpl],bpl);
   getAC(huftable,linein,bpl);
   wad = adb;
   DMA_setup(ada,wc,wad,wc);
   clear_mask_reg();
   wline = line1;
   /* get second line result */
   (*addline_to_predictor)(wline,addptr,bpl);
```

```
addptr += oneline;
cputs("interpolate");
if(pixsi == 4){
   interpolate4(intptr);
}
else if(pixsi == 2)
   interpolate2(intptr);
intptr += oneline;
movmem(updptr,scrptr,oneline);
swapbank();
while((((b = inportb(DMA_STATUS)) & 0x08) == 0) && !kbhit());

/* process other lines */
for(i = 3;(i < bpl) && (!quit);i++){
   memset(linein,0,2048);
   fill_DC(linein,&dcbuf[i*bpl],bpl);
   getAC(huftable,linein,bpl);
   wad = (i % 2)? adc:adb;
   DMA_setup(ada,wc,wad,wc);
   clear_mask_reg();
   wline = (i % 2)? line0:line1;
   /* process last line result */
   (*addline_to_predictor)(wline,addptr,bpl);
   addptr += oneline;
   if(pixsi == 4){
      interpolate4(intptr);
   }
   else if(pixsi == 2)
      interpolate2(intptr);
   intptr += oneline;
   cprintf("\n\r update screen %d",i);
   movmem(updptr,scrptr,oneline * 2);
   swapbank();
   updptr += oneline;
   scrptr += oneline;
   if(kbhit())
      quit = getcommand();  /* get command if kbhit() */

   /*  check if TMS finish computing  */
   while((((b = inportb(DMA_STATUS)) & 0x08) == 0) && !kbhit());
}
if(quit) break;

/* process last line result */
wline = line1;
cputs(" add last line\n\r");
(*addline_to_predictor)(wline,addptr,bpl);
cputs("\n\rlast add");
if(pixsi == 4) {
   interpolate4(intptr);
   intptr += oneline;
   interlast4(intptr);
}
if(pixsi == 2){
   interpolate2(intptr);
   intptr += oneline;
   interlast2(intptr);
}
cputs("\n\r update screen");
movmem(updptr,scrptr,oneline * 3);
swapbank();
updptr += oneline;
```

```
                        scrptr += oneline;
                        delay(120);
                        cputs("\n\r update last screen");
                        movmem(updptr,scrptr,oneline*2);
                        close_DMA_gate();
                        set_mask_reg();
                        t2 = clock()/CLK_TCK;
                        ta[stage] = t2 - t1;
                        cprintf("\n\r time used %10.4f\n\r",ta[stage]);
                    }
                    clear_receive();
                    close_DMA_gate();
                    set_mask_reg();
                    break;

            case 'c' : /*  communication set up  */
            case 'C' : fullscreen;
                    set_comm_setup();
                    write_comm_setup();
                    break;

            case 'S' : /* save option */
            case 's' : cputs("\n\r (P)icture (C)ode (B)oth");
                    k = getch();
                    switch(tolower(k)) {
                        case 'p' : savepic(screen,&fileno,".pic",0xFFFF);
                                   break;
                        case 'c' : savepic(filebuffer,&fileno,"",filecount);
                                   break;
                        case 'b' : savepic(screen,&fileno,".pic",0xFFFF);
                                   savepic(filebuffer,&fileno,"",filecount);
                                   break;
                    }
                    break;

            case 'R' : /* reset retriever */
            case 'r' : reset(screen,&thres_factor,huftable);
                    break;

            case 'M' : /* LOCAL or REMOTE source */
            case 'm' : set_mode();
                    break;

            case 'T' : /* time used in each stage */
            case 't' : for(i = 0;i < 5;i++){
                        cprintf(" time used in stage %d = %10.4f\n\r",i,ta[i]);
                    }
                    getch();
                    break;
        }
    }while(item != 'x');
}

void show_rec_menu(void)
{
    clrscr();
    cputs("                                                                \n\r");
    cputs("                          RETRIEVE MENU                         \n\r");
    cputs("                                                                \n\r");
    cputs("                                                                \n\r");
    cputs("                                                                \n\r");
    cputs("                                                                \n\r");
```

```
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        cputs("                                                                    \n\r");
        writexy(13,5,"BAUD RATE   :");
        writexy(13,6,"DATA BIT    :");
        writexy(13,7,"STOP BIT    :");
        writexy(13,8,"PARITY      :");
        writexy(13,11,"MODE        : LOCAL");
        writexy(13,12,"FILE NUMBER :");
        writexy(13,13,"FILE SIZE   :");
        writexy(13,14,"STAGE       :");
        read_comm_setup();
        print_comm_setup();
}

void set_mode(void)  /* set LOCAL or REMOTE mode for retrieving image */
{
        fullscreen;
        mode = (mode)? FALSE : TRUE;
        if(mode)
           writexy(27,11,"REMOTE");
        else
           writexy(27,11,"LOCAL ");
}

void adder14(char *line,BYTE far *ptr,int bpl)  /* adder of stage 1 and 4 */
{
        int                     i,j,k,l,m,temp;
        char                    *x;

        for(i = 0;i < 8;i++) {
           for(m = 0;m < pixsi;m++) {
                x = line;
                for(j = 0;j < bpl;j++){
                   for(k = 0;k < 8;k++){
                        for(l = 0;l < pixsi;l++){
                           temp = *ptr + *x;
                           temp = (temp > 255)? 255:temp; /* check overflow */
                           *ptr++ = (temp < 0)? 0:temp; /*  check underflow */
                        }
                        x++;
                   }
                   x += 56;
                }
           }
           line += 8;
        }
}

void adder23(char *line,BYTE far *ptr,int bpl)  /* adder of stage 2 and 3 */
```

```
{
        int                     i,j,k,l,m,temp;
        char                    *x;

        for(i = 0;i < 8;i++) {
           for(m = 0;m < pixsi;m++) {
                x = line;
                for(j = 0;j < bpl;j++){
                   for(k = 0;k < 8;k++){
                        for(l = 0;l < pixsi;l++){
                           temp = *ptr + *x + *x;
                           temp = (temp > 255)? 255:temp;
                           *ptr++ = (temp < 0)? 0:temp;
                        }
                        x++;
                   }
                   x += 56;
                }
           }
           line += 8;
        }
}

void update_line()   /* update screen buffer */
{
        int                     oneline;

        oneline = (256 * 8 * pixsi);
        movmem(updptr,scrptr,oneline * 2);
        swapbank();
        updptr += oneline;
        scrptr += oneline;
}

int getcommand(void)  /* get user command */
{
        int             key;  /* response */

        key = tolower(getch());
        if(key == 'q') {
           cputs(" Press any key to return to menu...\n\r");
           if(mode) outportb(0x3F8,'~');   /* send end of transmission to sender */
           getch();
           return 1;
        }
        else {
           cputs(" Press any key to continue...\n\r");
           getch();
           return 0;
        }
}


void getDC(struct HUFTAB *huftable,int *dcbuf,int noblk)  /* get DC coefficient */
{
        int             i,dc = 0;

        for(i = 0;i<noblk;i++) {
           dc += (decode(huftable) - BIAS);
           dcbuf[i] = dc;
        }
}
```

```
void fill_DC(char *line,int *dc,int bpl)  /* fill DC to block */
{
        int                     i;

        for(i = 0;i < bpl;i++,line+=64)
           *line = *dc++;
}

void getAC(struct HUFTAB *huftable,char *line,int bpl)  /* get AC coefficient */
{
        int                     i,index = 0;
        int                     temp, distemp, run;

        for(i = 0;i < bpl;i++,line += 64) {
           index = 0;
           while (((run = decode(huftable)) != EOB) && (index < 64)) {
                distemp = run >> 3;
                temp = run % 8;
                index += (distemp + 1);
                line[index] = coef_decode(extract(temp), temp);
           }
         }
}

void savepic(BYTE far *picture,int *fn,char *ext,unsigned int fs)
{  /* save retrieved image to Hard disk */
     char                       s[45];
     unsigned int               i;
     int                        handle=-1;

     do {
          strcat(itoa(*fn,s,10),ext);
          handle = _open(s,O_WRONLY | O_BINARY);

          /* check file s */
          if (handle != -1) {
             cprintf("\rOverwrite %s (Y/N) ",s);
             do
                  i = toupper(getche());
             while((i != 'Y') && (i != 'N'));
             if (i == 'Y') {
                  infoscreen;
                  clrscr();
                  cputs("    saving..... ");
                  _write(handle,picture,0xFFFF);
                  close(handle);
                  cputs("\r    finish saving    \n\r");
             } else {
                  cputs("\rEnter file number :");
                  scanf("%d",fn);
             }
           }else {
              infoscreen;
              clrscr();
              cputs("    saving..... ");
              handle = creat(s,S_IREAD | S_IWRITE);
              _write(handle,picture,fs);
              close(handle);
              cputs("\r    finish saving    \n\r");
           }
       }while(handle == -1);
```

```
}

void gettable(struct HUFTAB *huftable)  /* get table from stream */
{
        int             index,size;

        memset(huftable,0,sizeof(struct HUFTAB)*512);
        while((size = extract(4)) != 0) {
           index = extract(9);
           huftable[index].codes = extract(size);
           huftable[index].sizes = size;
        }
}

int Instage(int stage)  /* give useful parameter */
{
        if(stage == 4) {
           command = 0x06;
           addline_to_predictor = adder14;
           return (1);
        }
        else if(stage == 3) {
           command = 0x06;
           addline_to_predictor = adder23;
           return (1);
        }
        else if(stage == 2){
           command = 0x05;
           addline_to_predictor = adder23;
           return (2);
        }
        else {
           command = 0x04;
           addline_to_predictor = adder14;
           return (4);
        }
}

int coef_decode(int code, int temp)
{
        int             val,sign;

        sign = code >> (temp - 1);
        val = (1 << (temp - 1)) | code;
        if(sign == 1)
           val = -val;
        return(val);
}

void reset(BYTE far *screen,int *thres_factor,struct HUFTAB *huftable)
{
        clearscreen(screen);
        clearbuffer();
        resetbuffer(filebuffer);
        stage = 1;
        *thres_factor = 50;
        memset(huftable,0,sizeof(struct HUFTAB)*512);
}

void swapbank()
{
        outportb(CONTROL_PORT,0xC7);
```

```
}

void showpic(BYTE far *source)  /* show picture to screen */
{
        movmem(source, screen, 0xFFFF);
        swapbank();
        getch();
}
```

```
>

/************************************************/
/*   MODULE : PACK.C                            */
/*   This module is for packing, unpacking code */
/*   decode and encode data using table given,  */
/*  extract data from packed code ,etc.         */
/************************************************/

#include <io.h>
#include <conio.h>

#define  EOB                    0
#define  BLOCKSIZE              64

typedef  unsigned char          BYTE;
typedef  struct {
               unsigned long    b;
               int              s;
         } BUFFER;

/* structure used for huffmann table */

struct HUFTAB {
       int   codes;
       int   sizes;
};

void readbuffer(BUFFER *cb);
void setbuffer(BUFFER cb);
void fillbuffer(void);
void flushbuffer(void);
void clearfbuffer(void);
void resetfilebuf(BYTE far *filebuf);
int decode(struct HUFTAB *huftable);
unsigned long filebuflength(void);
int extract(int bit);
void pack(int code, int size);
int unpack(int bit);

/* internal GLOBAL VARIABLE */

static unsigned long            buffer = 0L;
static int                      bufsize = 0;
static int                      bytecount = 0;
static BYTE far                 *fbuf;

/* shared variable from other module */

volatile extern int             error,filecount;

void pack(int code, int size)  /* pack data together */
{
       if(size > 0) {
          bufsize += size;
          buffer = (buffer << size) + code;
          while(bufsize >= 8) {
               bufsize -= 8;
               *fbuf++ = (BYTE) (buffer >> bufsize);  /* out data to buffer */
               bytecount++;
          }
       }
```

```
}

int unpack(int bit)  /* unpack data */
{
        int             code;

        code = (int) (buffer >> (32 - bit));
        buffer <<= bit;
        bufsize -= bit;
        return code;
}

int decode(struct HUFTAB *huftable)  /* decode data from packed code using given table */
{
        int             i;

        for(i = 0;i < 512;i++) {
           if(huftable[i].sizes == 0)
                continue;
           if(bufsize < huftable[i].sizes)
                fillbuffer();
           if((int)(buffer >> (32 - huftable[i].sizes)) == huftable[i].codes) {
                (void)unpack(huftable[i].sizes);
                return i;
           }
        }
        return -1;
}

int extract(int bit)  /* extract data */
{
        if(bufsize <  bit)
           fillbuffer();
        return unpack(bit);
}

void fillbuffer(void)  /* fill buffer for next decode or extract */
{
        while(bufsize <= 24){
           while((bytecount == filecount) && !error);
           if(error){ /* check error */
                cprintf(" error %x\n\r",error);
                exit(1);
           }
           buffer += ((unsigned long) (*fbuf++) << (24 - bufsize));
           bufsize += 8;
           bytecount++;
        };
}

void flushbuffer(void)   /* clear residual data to channel buffer */
{
        *fbuf++ = (BYTE)(buffer << (8 - bufsize));
        buffer = 0L;
        bufsize = 0;
        bytecount++;
}

void clearbuffer(void)  /* clear buffer */
{
        buffer = 0L;
        bufsize = 0;
```

```
}

void resetbuffer(BYTE far *filebuf)  /* reset buffer to file buffer pointer */
{
        bytecount = 0;
        fbuf = filebuf;
}

void readbuffer(BUFFER *cb)  /* get last status of buffer  */
{
        cb->b = buffer;
        cb->s = bufsize;
}

void setbuffer(BUFFER cb)  /* set buffer to status given */
{
        buffer = cb.b;
        bufsize = cb.s;
}

unsigned long filebuflength(void)  /* give size of channel buffer */
{
        return ((unsigned long)(bytecount)* 8 + bufsize);
}
```

```
>

/************************************************/
/*                                              */
/*              TRANSMIT MODULE                 */
/*                                              */
/*     This module is to transmit image to      */
/*     remote user due to user request          */
/************************************************/

#include <conio.h>
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <alloc.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "name.h"
#include "comm.h"
#include "\image\process\screen.h"

#define IRQ4                    0x0C
#define LINE_CONTROL_REG        0x3F8
#define LINE_STATUS_REG         0x3FD
#define MODEM_CONTROL_REG       0x3FC
#define BAUD_RATE_LO            0x3F8
#define BAUD_RATE_HI            0x3F9
#define RX_BUF_REG              0x3F8
#define TX_BUF_REG              0x3F8
#define OCW1                    0x021
#define OCW2                    0x020

/* GLOBAL VARIABLE */

BYTE                    *filebuffer;
int                     filecount,error;

void show_transmit_menu(void);
void showbuf(BYTE *buf);
int get_choice(int c,char **w,int n);
int loadcode(char *s,BYTE *ch);

main()
{
        int             key,b;
        char            s[40];
        int             fileno;

        /*  allocate memory  */

        filebuffer = farmalloc(0x5FFF);
        if(!filebuffer) {
           cputs(" not enough memory for filebuffer!\n\r");
           exit(1);
        }

        show_transmit_menu();
        do {
           comdscreen;
```

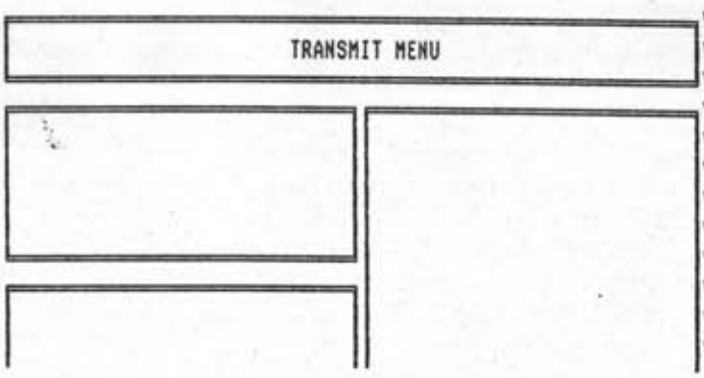
```
            cputs("\n\r");
            cputs("  c. communication setup\n\r");
            cputs("  s. sender\n\r");
            cputs("  x. exit\n\r");
            cputs("\n\r");
            cputs("             \r");
            cputs("  choose : ");
            key = getche();
            cputs("\n\r");
            fullscreen;
            switch(key) {
                 case 'c' : /* communication setup */
                 case 'C' : set_comm_setup();
                            write_comm_setup();
                            break;

                 case 's' : /* send file due to user request */
                 case 'S' : do{
                              infoscreen;
                              /* clear input data */
                              while(((inportb(LINE_STATUS_REG) & 0x01) == 0) && !kbhit());

                              /* check sender break */
                              if(kbhit()){
                                 key = getch();
                                 cputs("  sender break!            \n\r");
                                 break;
                              }
                              if((b=inportb(RX_BUF_REG)) != '?') {
                                 cprintf("  line error   %c          \n\r",b);
                                 break;
                              }

                              /* send acknowledge to receiver */
                              outportb(TX_BUF_REG,'~');
                              receive_fileno(s);
                              filecount = loadcode(s,filebuffer);
                              send_code(filecount,filebuffer);
                              infoscreen;
                              clrscr();
                              cputs("    finish sending ");
                            } while(key != 'q');
                            break;
            }
       }while(key != 'x');
}

void show_transmit_menu(void)
{
       clrscr();
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
       cputs("
```




```
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
\n\r");
```

```
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        cputs("                                                                \n\r");
        writexy(13,5,"BAUD RATE   :");
        writexy(13,6,"DATA BIT    :");
        writexy(13,7,"STOP BIT    :");
        writexy(13,8,"PARITY      :");
        writexy(13,11,"STATUS      : IDLE");
        writexy(13,12,"FILE NUMBER :");
        writexy(13,13,"FILE SIZE   :");
        writexy(13,14,"SEND COUNT  :");
        read_comm_setup();
        print_comm_setup();
}
```

```
}

/*************************************************/
/*   MODULE : COMM.C                             */
/*   This module collect function related to     */
/*   communication channel include HANDSHAKE     */
/*   between sender and receiver.                */
/*************************************************/

#include <conio.h>
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <alloc.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "name.h"
#include "\image\process\screen.h"

#define IRQ4                    0x0C
#define LINE_CONTROL_REG        0x3F8
#define LINE_STATUS_REG         0x3FD
#define MODEM_CONTROL_REG       0x3FC
#define BAUD_RATE_LO            0x3F8
#define BAUD_RATE_HI            0x3F9
#define RX_BUF_REG              0x3F8
#define TX_BUF_REG              0x3F8
#define OCW1                    0x021
#define OCW2                    0x020

/* structure used in communication setup */

struct COMM_SETUP {
        int             wl;
        int             stb;
        int             pen;
        int             eps;
        int             sp;
        int             sb;
        int             dbg;
};

struct LINE_STATUS {
        int             databit;
        int             stopbit;
        int             parity;
        int             rate;
};

/* PARAMETER FOR COMMUNICATION SETUP */

char *wordlength[] = {"5 bits","6 bits","7 bits","8 bits"};
char *no_of_stb[]  = {"1 bit","1-1/2 bits","2 bits"};
char *type_par[]   = {"none","odd","even"};
char *speed[]      = {"600","1200","1800","2400","3600","4800","7200","9600"};
char *general[]    = {"no","yes"};
char *(*line[])[]  = {wordlength,no_of_stb,type_par,speed};

int  baudrate[8]   = {0x0C0,0x060,0x040,0x030,0x020,0x018,0x010,0x00C};
```

```
/* GLOBAL VARIABLE */

void interrupt          (*oldvect)();
struct COMM_SETUP       line_control;
struct LINE_STATUS      lset;
static BYTE far         *fbuf;
extern BYTE far         *filebuffer;
volatile extern int     filecount,error;


void far restore_IRQ4();
void far disable_IRQ4();
void far enable_IRQ4();
void far set_IRQ4();
void set_comm_setup(void);
void read_comm_setup(void);
void print_comm_setup(void);
void write_comm_setup(void);
void send_file(char *s,BYTE *f);
void receive_file(char *s);
int get_fileno(void);
int loadcode(char *s, BYTE *ch);
void writexy(int x,int y,char *s);
void receive_fileno();
void send_fileno(int fn);
int receive_data();
void receive_fileno(char *s);
void send_code(int filecount,BYTE *f);
void clear_receive();
int get_choice(int *c,char **w,int n);

int get_fileno(void)   /* get file number request from user */
{
        int     fn;

        cputs(" input file number you want : ");
        scanf("%d",&fn);
        return fn;
}

void send_fileno(int fn)  /* send file number to sender */
{
        char            b;

        /* clear 8250 receive register */
        while((inportb(LINE_STATUS_REG) & 0x01) != 0)
          inportb(RX_BUF_REG);

        infoscreen;
        /* send request to sender */
        outportb(TX_BUF_REG,'?');

        /* wait for response */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0)
           cputs(" no acknowledge\r");
        if((b = inportb(RX_BUF_REG)) != '~'){
           cprintf(" line error    %c  \n\r",b);
        }
        cputs(" receiver ackknowledge\n\r");

        /* send file number */
```

```
        outportb(TX_BUF_REG,(char) (fn & 0xFF));  /* file number low byte */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0)
           cputs(" no acknowledge\r");
        if((b = inportb(RX_BUF_REG)) != '?'){
           cprintf(" line error    %c  \n\r",b);
        }
        outportb(TX_BUF_REG,(fn >> 8) & 0xFF);    /* file number high byte */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0)
           cputs(" no acknowledge\r");
        if((b = inportb(RX_BUF_REG)) != '~'){
           cprintf(" line error    %c  \n\r",b);
        }
        cputs(" send filenumber ready \n\r");
        cputs(" waiting data ....\r");
}

int receive_data(void)  /*  get data from channel */
{
        int             c= ' ',filesize;

        /* get filesize from 8250 receiving register */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0);  /* get low byte */
        filesize = inportb(RX_BUF_REG);
        outportb(TX_BUF_REG,'?');
        while((inportb(LINE_STATUS_REG) & 0x01) == 0);  /* get high byte */
        filesize += (inportb(RX_BUF_REG) << 8);

        /* check if file number requested valid */
        if(filesize == -1) {
           cprintf(" file not found %d \n\r",filesize);
           outportb(TX_BUF_REG,'~');
           return filesize;
        }
        fullscreen;
        gotoxy(27,13);
        cprintf("%-5d\r",filesize);

        /* ready to receive*/

        filecount = 0;
        fbuf = filebuffer;

        /* clear rx buffer*/

        (void)inportb(RX_BUF_REG);
        error = 0;

        set_IRQ4();
        outportb(0x3FC,0x08);
        outportb(0x3F9,0x01);
        enable_IRQ4();
        /* send ready signal */
        outportb(TX_BUF_REG,'~');
}

void receive_fileno(char *s)  /* sender get request file number */
{
        int             fn;

        fullscreen;
        writexy(27,11,"ACTIVE");
```

```
        /* get file number from channel */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0);
        fn = inportb(RX_BUF_REG);
        outportb(TX_BUF_REG,'?');
        while((inportb(LINE_STATUS_REG) & 0x01) == 0);
        fn += inportb(RX_BUF_REG) << 8;
        outportb(TX_BUF_REG,'~');
        gotoxy(27,12);
        cprintf("%-5d",fn);

        itoa(fn,s,10);
}

int loadcode(char *s, BYTE *ch)   /* load file number requested from database */
{
        int                     handle;         /* file parameter */
        int             count, filesize;
        char                    ss[40]="data\\"; /* path of image database */

        infoscreen;
        clrscr();
        cprintf(" loading %s\n\r",s);
        strcat(ss,s);
        handle = _open(ss,O_RDONLY | O_BINARY);

        /* check file number */
        if (handle == -1) {
           cprintf(" file %s not found\n\r",s);
           cputs("\0x07");
           return -1;
        }
        else {
           _read(handle,&filesize,sizeof(unsigned int));
           count = (unsigned int)_read(handle,(BYTE far*)ch,filesize);
           close(handle);
        }
        fullscreen;
        gotoxy(27,13);
        cprintf("%d",count);
        return(count);
}

void send_code(int filecount,BYTE *f)  /* send code to retriever */
{
        int sendcount = 1, key = ' ',b;

        infoscreen;
        outportb(TX_BUF_REG,filecount & 0x00FF);

        /* wait for sender ready signal */
        while((inportb(LINE_STATUS_REG) & 0x01) == 0);
        if((b = inportb(RX_BUF_REG)) != '?'){
           cputs(" no acknowledge\r");
        }
        outportb(TX_BUF_REG,(filecount & 0xFF00) >> 8);
        while((inportb(LINE_STATUS_REG) & 0x01) == 0)
           cputs(" no acknowledge\r");
        if((b = inportb(RX_BUF_REG)) != '~'){
           cprintf(" line error    %c\n\r",b);
        }

        /* exit if file number not valid */
```

```
        if(filecount == -1) {
           cputs("  file not available \n\r");
           return;
        }
        cprintf(" filecount o.k. %d\n\r",filecount);
        cputs(" sending .....\n\r");
        fullscreen;

        /* begin send code */
        while((sendcount <= filecount) && (key != '~')) {
           while((((b = inportb(LINE_STATUS_REG)) & 0x20) == 0) && (!kbhit()));
           outportb(TX_BUF_REG,*f++);
           gotoxy(27,14);
           cprintf("%-5d\r",sendcount);
           sendcount++;   /* increase count */
           if((b & 0x01) != 0){
                key = inportb(RX_BUF_REG);
           }

           /* check command for sender side break */
           if(kbhit()) {
                if(((key = getch()) == 'q') || (key == 'Q')){
                   infoscreen;
                   cputs("   ...please wait until finish sending this picture...\n\r");
                   cputs("                    ...then press any key....");
                   fullscreen;
                }
           }
        };
        writexy(27,11,"IDLE     ");
}

void clear_receive()  /* clear 8250 */
{
        if(!error)
           cputs("\n file receive o.k.\n\r");
        else
           cprintf("\n file receive error %x\r\n",error);
        outportb(0x3F9,0x00);
        outportb(0x3FC,0x00);
        disable_IRQ4();
        restore_IRQ4();
}

/* interrupt service routine for receiving data from channel */

void interrupt comm(bp,di,si,ds,es,dx,cx,bx,ip,cs,flags)
{
        enable();
        if(((error = inportb(LINE_STATUS_REG)) & 0x01) != 0){
           *fbuf++ = inportb(RX_BUF_REG);
           filecount++;
           error &= 0x1E;  /* check error */
        }
        outportb(OCW2,0x20);  /* send end of interrupt to 8259 */
}

void read_comm_setup()  /* read parameter set up */
{
        BYTE            b;
        int             i,t;
```

```
        b = inportb(LINE_CONTROL_REG);
        line_control.wl  = (b & 0x03);          /* data word length */
        line_control.stb = ((b & 0x04) >> 2);   /* length of stop bit */
        line_control.pen = ((b & 0x08) >> 3);   /* parity enable */
        line_control.eps = ((b & 0x10) >> 4);   /* even or odd parity */
        line_control.sp  = ((b & 0x20) >> 5);
        line_control.sb  = ((b & 0x40) >> 6);
        lset.databit = line_control.wl;
        if(line_control.wl == 0)
           lset.stopbit = line_control.stb;
        else
           lset.stopbit = line_control.stb + 1;
        if(line_control.pen == 0)
           lset.parity = 0;
        else
           lset.parity = line_control.pen + 1;
        outportb(LINE_CONTROL_REG,(b | 0x80));
        t = inportb(BAUD_RATE_LO);
        line_control.dbg = (inportb(BAUD_RATE_HI) << 8) + t;
        outportb(LINE_CONTROL_REG,(b & 0x7F));
        lset.rate = 0;
        while(baudrate[lset.rate] != line_control.dbg) lset.rate++;
}

void writexy(int x,int y,char *s)  /* write string at position X,Y */
{
        gotoxy(x,y);
        cputs(s);
}

void print_comm_setup(void)  /* show communication setup */
{
        writexy(27,5,speed[lset.rate]);
        writexy(27,6,wordlength[lset.databit]);
        writexy(27,7,no_of_stb[lset.stopbit]);
        writexy(27,8,type_par[lset.parity]);
}

void write_comm_setup(void)  /* write communication setup to 8250 */
{
        int                     temp;

        line_control.wl = lset.databit;
        if(lset.databit == 0)
           line_control.stb = lset.stopbit;
        else
           line_control.stb = lset.stopbit - 1;
        if(lset.parity == 0)
           line_control.pen = 0;
        else{
           line_control.pen = 1;
           line_control.eps = lset.parity - 1;
        }
        temp = (line_control.eps << 4) + (line_control.pen << 3) + (line_control.stb << 2 )
                + (line_control.wl) + (1 << 7);

        outportb(LINE_CONTROL_REG,(BYTE)temp);
        outportb(BAUD_RATE_LO,(baudrate[lset.rate] & 0x00FF));
        outportb(BAUD_RATE_HI,(baudrate[lset.rate] & 0xFF00) >> 8);
        outportb(LINE_CONTROL_REG,(BYTE)(temp & 0x7F));
}
```

```
void set_comm_setup(void)  /* set up communication parameter */
{
        static int                      p = 0;
        int                             key;

        do {
           switch(p) {
                case 0 : gotoxy(27,5);  /* set data bit rate */
                         key = get_choice(&lset.rate,speed,8);
                         break;

                case 1 : gotoxy(27,6);  /* set data word length */
                         key = get_choice(&lset.databit,wordlength,4);
                         break;

                case 2 : gotoxy(27,7);  /* set length of stop bit */
                         key = get_choice(&lset.stopbit,no_of_stb,3);
                         break;

                case 3 : gotoxy(27,8);  /* set parity */
                         key = get_choice(&lset.parity,type_par,3);
                         break;
           }
           switch(key) {
                case UPKEY   : p--;
                               if(p < 0)
                                  p = 3;
                               break;
                case DOWNKEY : p++;
                               if(p > 3)
                                  p = 0;
                               break;
           }
        } while(key != ESC);
}

int get_choice(int *c,char **w,int n)  /* get user choice */
{
        int                     x,y;  /* last position print */
        int                     k;

        x = wherex(); y = wherey();
        textbackground(WHITE);
        textcolor(BLACK);
        cprintf("%-10s",w[*c]);  /* print last parameter chosen */
        while(((k = getkey()) != UPKEY) && (k != DOWNKEY) && (k != ESC)) {
           switch(k) {
              case LEFTKEY  : (*c)++;
                              if(*c == n)
                                 *c = 0;
                              break;
              case RIGHTKEY : (*c)--;
                              if(*c  < 0)
                                 *c = n - 1;
                              break;
           }
           gotoxy(x,y);
           cprintf("%-10s",w[*c]);  /* print new parameter */
        }
        normvideo();
        gotoxy(x,y);
        cprintf("%-10s",w[*c]);
```

```
        return k;
}

int getkey(void)  /* Uses the BIOS to read the next keyboard character */
{
     int               key, lo, hi;

     key = bioskey(0);
     lo = key & 0x00FF;
     hi = (key & 0xFF00) >> 8;
     return((lo == 0) ? hi + 256 : lo);
} /* getkey */


void far set_IRQ4()  /* install interrupt service routine to IRQ4 */
{
        oldvect=getvect(IRQ4);
        setvect(IRQ4,comm);
}

void far restore_IRQ4()  /* restore old IRQ4 */
{
        setvect(IRQ4,oldvect);
}

void far enable_IRQ4()  /* enable IRQ4 */
{
        outportb(OCW1,(inportb(OCW1) & 0xEF));
}

void far disable_IRQ4()  /* disable IRQ4 */
{
        outportb(OCW1,(inportb(OCW1) | 0x10));
}
```

```
>

/************************************************/
/*   MODULE : INTF.C                            */
/*   This module control interfacing between    */
/*   microcomputer and TMS320E15.  There are    */
/*   functions to set up 8237 DMA controller    */
/*   and TMS before communicating to each other */
/************************************************/

#include <conio.h>
#include "name.h"

void set_mask_reg(void);
void resetFF(void);
void read_current(void);
unsigned read_addr(int chno);
unsigned read_count(int chno);
void write_addr(int chno,unsigned long abs_addr);
void write_count(int chno,unsigned count);

void TMS_setup(BYTE command,int th)  /* reset then send command and parameters to TMS */
{
        outportb(IBM_CONTROL_PORT,TMS_RESET);
        delay(20);
        outportb(IBM_CONTROL_PORT,TMS_IDLE);
        while((inportb(TMS_STATUS) & 0x40) == 0);
        outportb(PORT_DATA,command);     /* send command to TMS */
        while((inportb(TMS_STATUS) & 0x40) == 0);
        outportb(PORT_DATA,(BYTE)(th));  /* send threshold to TMS */
        if((inportb(TMS_STATUS) & 0x80) != 0) {
           /*      clear output tms port */
           (void)inportb(PORT_DATA);
        }
}

void DMA_setup(unsigned long addr1,unsigned wc1,unsigned long addr2,unsigned wc2)
{  /* prepare 8237 DMA controller before communicating to TMS */
        write_addr(CHANNEL1,addr1);  /* set address of DMA channel 1 */
        write_addr(CHANNEL3,addr2);  /* set address of DMA channel 3 */
        write_count(CHANNEL1,wc1);   /* set data count of DMA channel 1 */
        write_count(CHANNEL3,wc2);   /* set data count of DMA channel 3 */
}

void read_current(void)  /* show current address and count of DMA channel */
{
        cprintf("channel 1");
        cprintf("  CURRENT ADDRESS : %x",read_addr(CHANNEL1));
        cprintf(" CURRENT WORD COUNT : %x\n\r",read_count(CHANNEL1));
        cprintf("channel 3");
        cprintf("  CURRENT ADDRESS : %x",read_addr(CHANNEL3));
        cprintf(" CURRENT WORD COUNT : %x\n\r",read_count(CHANNEL3));
}

unsigned read_addr(int chno)  /* read current address of DMA channel */
{
        unsigned char           addr;

        resetFF();   /* clear first last flipflop */
        addr = inportb(chno);
        addr += ((unsigned)inportb(chno) << 8);
        return (addr);
```

```
}

unsigned read_count(int chno)  /* read current word count of DMA channel */
{
        unsigned                count;

        resetFF();   /* clear first last flipflop */
        count = inportb(chno + 1);
        count += ((unsigned) inportb(chno + 1) << 8);
        return (count);
}

void write_addr(int chno,unsigned long abs_addr)  /* write address to 8237 */
{
        resetFF();   /* clear first last flipflop */
        outportb(chno,(unsigned char)(abs_addr & 0x00FF));
        outportb(chno,(unsigned char)((abs_addr >> 8) & 0x00FF));

        /* write DMA page register */
        if(chno == CHANNEL1)
           outportb(DMA_PAGE_REG1,(unsigned char)(abs_addr >>16) & 0x000F);
        else
           outportb(DMA_PAGE_REG3,(unsigned char)(abs_addr >>16) & 0x000F);
}

void write_count(int chno,unsigned count) /* write word count to 8237 */
{
        resetFF();   /* clear first last flipflop */
        outportb(chno + 1,(unsigned char)(count & 0xFF));
        outportb(chno + 1,(unsigned char)(count >> 8));
}

void set_DMA_mode(void)  /* set mode DMA */
{
        set_mask_reg();
        outportb(DMA_MODE_REG,0x49);  /* set channel 1 for read */
        outportb(DMA_MODE_REG,0x47);  /* set channel 3 for write */
}

unsigned long abs_address(void *pointer)  /* compute absolute address */
{
        return (((unsigned long)FP_SEG(pointer) << 4) + FP_OFF(pointer));
}

unsigned char read_8237_status(void)  /* read 8237 status */
{
        return (inportb(DMA_STATUS));
}

void open_DMA_gate(void)  /* open DMA gate */
{
        outportb(IBM_CONTROL_PORT,DMA_GATE);
}

void close_DMA_gate(void) /* close DMA gate */
{
        outportb(IBM_CONTROL_PORT,TMS_IDLE);
}

void set_mask_reg(void)   /* set 8237 mask register */
{
        outportb(DMA_MASK_REG,0x0E);
```

```
}

void clear_mask_reg(void)  /* clear 8237 mask register */
{
        outportb(DMA_MASK_REG,0x04);
}

void resetFF(void)  /* clear first-last flipflop for read or write */
{
        outportb(CLEAR_FF,0x00);
}
```

```
>

/************************************************/
/*   MODULE : INTPOL1.C                         */
/*   This module collects functions dealed with */
/*   interpolation for line and screen.         */
/************************************************/

#include <conio.h>
#include "name.h"
#include "intpol1.h"

void interlast4(BYTE *frame)  /* interpolate last line of stage 1 */
{
        int             i;

        for(i = 0;i < 7;i++,frame += 1024) {
           interline4(frame);
        }
        interx4(frame);
}

void interpolate4(BYTE *frame)  /* interpolate line of stage 1 */
{
        int             i;

        for(i = 0;i < 8;i++,frame += 1024) {
           interline4(frame);
        }
}

void interline4(BYTE far *s)    /* interpolate subline of stage 1 */
{
        int             j;

        for(j = 0;j < 63;j++,s+=4) {
           interpixel4(s);
        }
        intery4(s);
}

void intery4(BYTE far *s)  /* interpolate last Y axis for stage 1 */
{
        int             a,d,temp;

        a = *s; d = *(s + 1024);
        temp = (a + d)/2;
        temp = (temp >255)? 255:temp;   /* check for saturation */
        s += 512;
        *s = *(s+1) = *(s+2) = *(s + 3) = (BYTE)(temp); s += 256;
        *s = *(s+1) = *(s+2) = *(s + 3) = (BYTE)(temp);
}

void interx4(BYTE far *s)  /* interpolate last X axis for stage 1 */
{
        int             i,a,b,temp;

        for(i = 0;i < 63;i++) {
           a = *s; b = *(s + 4);
           temp = (a + b)/2;
           temp = (temp > 255)? 255:temp;  /* check for saturation */
           *(s+2) = *(s+3) = (BYTE)(temp);
```

```
            movmem(s,s+256,4);
            movmem(s,s+512,4);
            movmem(s,s+768,4);
            s += 4;
         }
}

void interpixel4(BYTE *pos)  /* interpolate subpoint of stage 1 */
{
        int                     a,b,c,d,temp;
        BYTE far                *r,*s;

        a = *pos;b = *(pos+4);d = *(pos + 1024);c = *(pos + 1028);
        s = pos+2 ;r = pos + 258;
        temp = (a + b)/2;
        temp = (temp > 255)? 255:temp; /* check for saturation */
        *s++ = *r++ = (BYTE)(temp);
        *s++ = *r++ = (BYTE)(temp);
        s += 508,r += 508;
        temp = (a + d)/2;
        temp = (temp > 255)? 255:temp; /* check for saturation */
        *s++ = *r++ = (BYTE)(temp);
        *s++ = *r++ = (BYTE)(temp);
        temp = (a + b + c + d)/4;
        temp = (temp > 255)? 255:temp; /* check for saturation */
        *s++ = *r++ = (BYTE)(temp);
        *s++ = *r++ = (BYTE)(temp);
}

void interlast2(BYTE *frame)   /* interpolate last line of stage 2 */
{
        int             i;

        for(i = 0;i < 7;i++,frame += 512) {
           interline2(frame);
        }
        interx2(frame);
}

void interpolate2(BYTE *frame)   /* interpolate line of stage 2 */
{
        int             i;

        for(i = 0;i < 8;i++,frame += 512) {
           interline2(frame);
        }
}

void interline2(BYTE far *s)  /* interpolate subline of stage 2 */
{
        int             j;

        for(j = 0;(j < 127);j++,s+=2) {
           interpixel2(s);
        }
        intery2(s);
}

void intery2(BYTE far *s)  /* interpolate last Y axis for stage 2 */
{
        int             temp;
```

```
        temp = (*s + *(s + 512))/2;
        temp = (temp > 255)? 255:temp;  /* check for saturation */
        s += 256;
        *s = *(s+1) = (BYTE)(temp);
}

void interx2(BYTE far *s)  /* interpolate last subline for stage 2 */
{
        int             i,a,b,temp;

        for(i = 0;i < 127;i++) {
           temp = (*s + *(s + 2))/2;
           temp = (temp > 255)? 255:temp;  /* check for saturation */
           *(s+1) = *(s+257) = (BYTE)(temp);
           s += 2;
        }
}

void interpixel2(BYTE *pos)   /* interpolate subpoint of stage 2 */
{
        int             a,b,c,d,temp;
        BYTE far        *s;

        a = *pos;b = *(pos+2);d = *(pos + 512);c = *(pos + 514);
        s = ++pos;
        temp = (a + b)/2;
        temp = (temp >255)? 255:temp;  /* check for saturation */
        *s++ = (BYTE)(temp);
        s += 254;
        temp = (a + d)/2;
        temp = (temp >255)? 255:temp;
        *s++ = (BYTE)(temp);
        temp = (a + b + c + d)/4;
        temp = (temp >255)? 255:temp;
        *s++ = (BYTE)(temp);
}

void update(int *blk,BYTE far *r_frame,BYTE far *screen,int iblk,int pixsi)
{   /* update screen buffer */
        int             nx,ny,ox,oy;
        int             xblk,yblk;

      xblk = 32/pixsi;
        ny = iblk / xblk;
        nx = iblk % xblk;
        writeblock(r_frame,blk,nx,ny,pixsi);
        copyblock(r_frame,screen,nx,ny,pixsi);
        if(iblk != 0){
           oy = (iblk - 1) / xblk;
           ox = (iblk - 1) % xblk;
           copyblock(r_frame,screen,ox,oy,pixsi);
        }
        swapbank();
        delay(25);
        if((ny == (xblk-1)) && (nx == (xblk-1)))
           copyblock(r_frame,screen,nx,ny,pixsi);
}

void writeblock(BYTE far *r_frame,char *blk,int x,int y,int pixsi)
{
        BYTE far        *p,*q,*r;
        int             i,j,k,l,v,temp;
```

```
        p = r_frame + (y * 2048 * pixsi) + (x * 8 * pixsi);
        for(i = 0;i < 8;i++) {
           q = p;
           for(j = 0;j < 8;j++) {
                r = q;
                v = *blk++ + *r;
                for(k = 0;k < pixsi;k++) {
                   for(l = 0;l < pixsi;l++){
                        v = (v > 255)? 255 : v;
                        *r++ = (BYTE)((v < 0)? 0 : v);
                   }
                   r += (256 - pixsi);
                }
                q += pixsi;
           }
           p += (256 * pixsi);
        }
}

void copyblock(BYTE far *r_frame,BYTE far *screen,int x,int y,int pixsi)
{
        BYTE far        *q,*r;
        int             i,j,k,offset;

        k = 8*pixsi;
        if((y == 63) && (x == 63))
        cprintf(" ny %d , nx %d \r",y,x);
        offset = (y * 2048 * pixsi) + (x * 8 *pixsi);
        r = r_frame + offset;
        q = screen + offset;
        for(i = 0;i < k;i++) {
           for(j = 0;j < k;j++)
                *q++ = *r++;
           r += (256-k);q += (256-k);
        }
}

void clearscreen(BYTE far *screen)
{
        memset(screen,0,0xFFFF);
        swapbank();
        delay(20);
        memset(screen,0,0xFFFF);
}
```

```
}

/*************************************************/
/*   HEADER : COMM.H                             */
/*************************************************/

void far restore_IRQ3();
void far disable_IRQ3();
void far enable_IRQ3();
void far set_IRQ3();
void set_comm_setup(void);
void read_comm_setup(void);
void print_comm_setup(void);
void write_comm_setup(void);
void writexy(int x,int y,char *s);
int get_fileno(void);
int loadcode(char *s, BYTE *ch);
void receive_fileno();
void send_fileno(int fn);
int receive_data();
void receive_fileno(char *s);
int OpenCodeFile(char *s);
void send_code(int filecount,BYTE *f);
void clear_receive();
int get_choice(int c,char **w,int n);


/*************************************************/
/*   HEADER : INTF1.H                            */
/*************************************************/

void open_DMA_gate(void);
void close_DMA_gate(void);
void set_mask_reg(void);
void clear_mask_reg(void);
unsigned char read_8237_status(void);
void TMS_setup(BYTE command,int th);
void DMA_setup(unsigned long addr1,unsigned wc1,unsigned long addr2,unsigned wc2);
void set_DMA_mode(void);
void read_current(void);
unsigned long abs_address(void *pointer);


/*************************************************/
/*   HEADER : INTPOL1.H                          */
/*************************************************/

void writeblock(BYTE far *r_frame,char *blk,int x,int y,int pixsi);
void copyblock(BYTE far *r_frame,BYTE far *screen,int x,int y,int pixsi);
void update(int *blk,BYTE far *r_frame,BYTE far *screen,int iblk,int pixsi);
void clearscreen(BYTE far *screen);
void interpolate4(BYTE far *s);
void interpixel4(BYTE far *pos);
void interx4(BYTE far *s);
void intery4(BYTE far *s);
void interline4(BYTE far *s);
void interpolate2(BYTE far *s);
void interpixel2(BYTE far *pos);
void interx2(BYTE far *s);
void intery2(BYTE far *s);
void interline2(BYTE far *s);
```

```
/************************************************/
/*   HEADER : PACK.H                            */
/************************************************/

#define  EOB                    0
#define  BLOCKSIZE              64

typedef  struct {
               unsigned long          b;
               int                    s;
         } BUFFER;

struct HUFTAB {
       int   codes;
       int   sizes;
};

void readbuffer(BUFFER *cb);
void setbuffer(BUFFER cb);
void fillbuffer(void);
void flushbuffer(void);
void clearfbuffer(void);
void resetfilebuf(BYTE far *filebuf);
unsigned long filebuflength(void);
int decode(struct HUFTAB *huftable);
int extract(int bit);
void pack(int code, int size);
int unpack(int bit);


/************************************************/
/*   HEADER : NAME.H                            */
/*   MACRO DEFINITION for TMS-DMA interface     */
/************************************************/

#include <dos.h>

#define CHANNEL1                0x02
#define CHANNEL3                0x06
#define DMA_PAGE_REG1           0x083
#define DMA_PAGE_REG3           0x082
#define DMA_MASK_REG            0x00F
#define DMA_MODE_REG            0x008
#define DMA_STATUS              0x008
#define CLEAR_FF                0x00C
#define IBM_CONTROL_PORT        0x33E
#define PORT_DATA               0x33D
#define TMS_STATUS              0x33C
#define TMS_RESET               0x10
#define TMS_IDLE                0x11
#define DMA_GATE                0x17
#define DATASIZE                1024
#define BLOCKSIZE               64

#define fullscreen              window(1,1,80,25)
#define infoscreen              window(13,17,68,20)
#define comdscreen              window(42,5,68,14)

typedef unsigned char           BYTE;
```

ภาคผนวก ฉ

รายละเอียดของส่วนโปรแกรมภาษาแอสเซมบลีของ TMS320E15

```
>
*************************************************************
*                                                           *
*          TMS320E15 ASSEMBLY LANGUAGE MODULE               *
*                                                           *
*                        for                                *
*                                                           *
*    2-D FAST ADAPTIVE DISCRETE COSINE TRANSFORM 8x8        *
*                                                           *
*************************************************************

              IDT     'CODEC'

C1            EQU           4017          * These coefficients are scaled up
C2            EQU           3784          * by 4096
C3            EQU           3406
C4            EQU           2896
C5            EQU           2276
C6            EQU           1567
C7            EQU           799
C41           EQU           2841
C43           EQU           2408
C45           EQU           1609
C47           EQU           565
C42           EQU           2676
C46           EQU           1108

*---------------------------------- DATA AREA -------------------------------*

**********************  FIRST ARRAY  *********************

FIR0          EQU           0
FIR1          EQU           1
FIR2          EQU           2
FIR3          EQU           3
FIR4          EQU           4
FIR5          EQU           5
FIR6          EQU           6
FIR7          EQU           7
FIR8          EQU           8
FIR9          EQU           9
FIR10         EQU           10
FIR11         EQU           11
FIR12         EQU           12
FIR13         EQU           13
FIR14         EQU           14
FIR15         EQU           15
FIR16         EQU           16
FIR17         EQU           17
FIR18         EQU           18
FIR19         EQU           19
FIR20         EQU           20
FIR21         EQU           21
FIR22         EQU           22
FIR23         EQU           23
FIR24         EQU           24
FIR25         EQU           25
FIR26         EQU           26
FIR27         EQU           27
FIR28         EQU           28
FIR29         EQU           29
FIR30         EQU           30
```

```
FIR31           EQU             31
FIR32           EQU             32
FIR33           EQU             33
FIR34           EQU             34
FIR35           EQU             35
FIR36           EQU             36
FIR37           EQU             37
FIR38           EQU             38
FIR39           EQU             39
FIR40           EQU             40
FIR41           EQU             41
FIR42           EQU             42
FIR43           EQU             43
FIR44           EQU             44
FIR45           EQU             45
FIR46           EQU             46
FIR47           EQU             47
FIR48           EQU             48
FIR49           EQU             49
FIR50           EQU             50
FIR51           EQU             51
FIR52           EQU             52
FIR53           EQU             53
FIR54           EQU             54
FIR55           EQU             55
FIR56           EQU             56
FIR57           EQU             57
FIR58           EQU             58
FIR59           EQU             59
FIR60           EQU             60
FIR61           EQU             61
FIR62           EQU             62
FIR63           EQU             63

********************** SECOND  ARRAY ********************

SEC0            EQU             64
SEC1            EQU             65
SEC2            EQU             66
SEC3            EQU             67
SEC4            EQU             68
SEC5            EQU             69
SEC6            EQU             70
SEC7            EQU             71
SEC8            EQU             72
SEC9            EQU             73
SEC10           EQU             74
SEC11           EQU             75
SEC12           EQU             76
SEC13           EQU             77
SEC14           EQU             78
SEC15           EQU             79
SEC16           EQU             80
SEC17           EQU             81
SEC18           EQU             82
SEC19           EQU             83
SEC20           EQU             84
SEC21           EQU             85
SEC22           EQU             86
SEC23           EQU             87
SEC24           EQU             88
SEC25           EQU             89
```

```
SEC26           EQU             90
SEC27           EQU             91
SEC28           EQU             92
SEC29           EQU             93
SEC30           EQU             94
SEC31           EQU             95
SEC32           EQU             96
SEC33           EQU             97
SEC34           EQU             98
SEC35           EQU             99
SEC36           EQU             100
SEC37           EQU             101
SEC38           EQU             102
SEC39           EQU             103
SEC40           EQU             104
SEC41           EQU             105
SEC42           EQU             106
SEC43           EQU             107
SEC44           EQU             108
SEC45           EQU             109
SEC46           EQU             110
SEC47           EQU             111
SEC48           EQU             112
SEC49           EQU             113
SEC50           EQU             114
SEC51           EQU             115
SEC52           EQU             116
SEC53           EQU             117
SEC54           EQU             118
SEC55           EQU             119
SEC56           EQU             120
SEC57           EQU             121
SEC58           EQU             122
SEC59           EQU             123
SEC60           EQU             124
SEC61           EQU             125
SEC62           EQU             126
SEC63           EQU             127

********************  DEFINED VALUE ********************

ZERO            EQU             0               store zero
ONE             EQU             1               store one
STATUS          EQU             2               status of TMS flag
COMMND          EQU             3               command from IBM
BLKIN           EQU             4               define no. of block
BLKOUT          EQU             5
FACTOR          EQU             6               threshold factor
COUNT           EQU             7
OFFSET          EQU             8
TMSSTA          EQU             9
ROFF            EQU             10              round off value
ROFINV          EQU             138
NOWIN           EQU             11
NOWOUT          EQU             12
AR00            EQU             13
AR01            EQU             14
ACH             EQU             15
ACL             EQU             16
MATX0           EQU             192
MATX63          EQU             255
```

```
********************** DEFINED PORT **********************

REQ             EQU             PA0
INPORT          EQU             PA1
OUTPOR          EQU             PA2
STAWRT          EQU             PA3

********************** DEFINED MACRO **********************

* This macro is to sum 2 operands with LOGICAL shift right *

SLSR            $MACRO  OP1,OP2,RES,SH
                $ASG    16-SH.V TO SH.V
                LAC     :OP1:,:SH.V:
                ADD     :OP2:,:SH.V:
                SACH    :RES:,0
                $END    SLSR

* This macro is to subtract 2 operands with LOGICAL shift right *

DLSR            $MACRO  OP1,OP2,RES,SH
                $ASG    16-SH.V TO SH.V
                LAC     :OP1:,:SH.V:
                SUB     :OP2:,:SH.V:
                SACH    :RES:,0
                $END    DLSR

* This macro is to sum 2 operands with LOGICAL shift left *

SLSL            $MACRO  OP1,OP2,RES,SH
                $IF     SH.A&$PCALL
                LAC     :OP1:,:SH:
                ADD     :OP2:,:SH:
                $ELSE
                LAC     :OP1:
                ADD     :OP2:
                $ENDIF
                SACL    :RES:
                $END    SLSL

* This macro is for subtract 2 operands with LOGICAL shift left *

DLSL            $MACRO  OP1,OP2,RES,SH
                $IF     SH.A&$PCALL
                LAC     :OP1:,:SH:
                SUB     :OP2:,:SH:
                $ELSE
                LAC     :OP1:
                SUB     :OP2:
                $ENDIF
                SACL    :RES:
                $END    DLSL

* This macro is for add 2 operands at MSB of accumulator *

SUMH            $MACRO  OP1,OP2,RES,RSH
                ZALH    :OP1:
                ADDH    :OP2:
                $IF     RSH.A&$PCALL
                SACH    :RES:
                ADDH    :RES:
                ADDH    *                       round off
```

```
                $ENDIF
                SACH    :RES:
                $END    SUMH

* This macro is for subtract 2 operands at MSB of accumulator *

DIFH            $MACRO  OP1,OP2,RES,RSH
                ZALH    :OP1:
                SUBH    :OP2:
                $IF     RSH.A&$PCALL
                SACH    :RES:
                ADDH    :RES:
                ADDH    *                       round off
                $ENDIF
                SACH    :RES:
                $END    DIFH

*-------------------------------- PROGRAM AREA -----------------------------*

                AORG    0
                B       INIT
                B       INTR

********************* INTERRUPT ROUTINE ******************

INTR            SST     STATUS
                LDPK    1
                SACH    ACH
                SACL    ACL
                SAR     0,AR00
                SAR     1,AR01

                LAC     BLKOUT
                SUB     BLKIN
                BZ      D00                     BLKIN = BLKOUT
                SUB     ONE
                BZ      D02                     wait for OUT to IBM
                LACK    4                       DMA error!
                B       STORE
D00             LAC     BLKOUT
                BNZ     D01
                LACK    0
                B       STORE                   finish transform
D01             LACK    1
                B       STORE                   going to IN next block
D02             LAC     BLKIN
                BNZ     D03
                LACK    2                       IN all blocks but still OUT last block
                B       STORE
D03             LACK  3                         going to OUT next block
STORE           ADD     ONE,3                   set bit 3 for indicate interrupt reply
                SACL    TMSSTA
                LAC     TMSSTA,8                shift to high bit before out
                SACL    TMSSTA
                OUT     TMSSTA,STAWRT           out to status register to I8M

                LAR     0,AR00                  restore register
                LAR     1,AR01
                ZALH    ACH
                ADDS    ACL
                LST     STATUS
                EINT
```

```
                RET

********************** INITIALIZATION ******************

INIT            DINT
                SOVM
                LDPK    1
                LACK    1
                SACL    ONE

                LAC     ONE,15          open 8IO gate for OUT
                ADD     ONE,12          not open IBM interrupt gate
                SACL    NOWOUT

                LAC     ONE,14          open 8IO gate fo IN
                ADD     ONE,12          not open IBM interrupt gate
                SACL    NOWIN

                ZAC
                SACL    ZERO

                LAC     ONE,14
                SACL    OFFSET          OFFSET for threshold factor = 128

                IN      COMMND,INPORT   flush input buffer
                OUT     NOWIN,STAWRT
                EINT

START           LDPK    1               use page 1 to be processing area
WAIT1           BIOZ    WAIT1
                IN      COMMND,INPORT   input command from IBM
                LAC     COMMND          add command to tell status
                ADD     NOWOUT
                SACL    NOWOUT          status during out buffer
                LAC     COMMND
                ADD     NOWIN
                SACL    NOWIN           status during in buffer

WAIT2           BIOZ    WAIT2
                IN      FACTOR,INPORT   input threshold factor from IBM
                LAC     FACTOR,15
                ADDH    OFFSET
                SACH    FACTOR          this is true value of threshold factor
                LAC     ONE,9
                AND     COMMND          test bit 1
                BNZ     B1024           if set goto B1024
                LAC     ONE,8
                AND     COMMND          test bit 0
                BNZ     B256            if set goto B256
                LACK    64              if not set store 64 to noblk
                B       TINV
B1024           LAC     ONE,10          store 1024 to noblk
                B       TINV
B256            LAC     ONE,8           store 256 to noblk
TINV            SACL    BLKIN
                SACL    BLKOUT

                LARK    AR0,MATX0       load destination for threshold value
                LARK    AR1,63          total 64 value in matrix
                LT      ONE
                LAC     ONE,10
                AND     COMMND          test bit 2
```

```
                BNZ     INVM            if = 1 goto inverse
                MPYK    THLD            set pntr to threshold matrix
                PAC
                B       TABLE
INVM            MPYK    ITHLD           set pntr to inverse threshold matrix
                PAC
TABLE           LARP    AR0
                TBLR    *+,AR1          read value to store at AR0
                ADD     ONE             increment pointer
                BANZ    TABLE

                LARK    AR0,MATX0       position of matrix element
                LARK    AR1,63          loop count
                LAC     ONE,10
                AND     COMMND          test bit 2
                BZ      DIV

                LT      FACTOR          load multiplicand
MUL             LARP    AR0
                MPY     *               multiply with threshold matrix
                PAC
                SACH    *+,1,AR1        store at the same location in 11.5
                BANZ    MUL
                LAC     ONE,7
                SACL    ROFF            round off before OUT when finished
                B       IFDCT

DIV             SAR     AR1,COUNT       store loop count
                LARK    AR1,14          another loop count for dividing
                LARP    AR0
                LAC     *,11,AR1        load nominator
KPDVNG  SUBC    FACTOR                  divide by denum
                BANZ    KPDVNG          get 15 decimal digits
                LARP    AR0
                SACL    *+,0,AR1        store the result at same position
                LAR     AR1,COUNT       in form Q15
                BANZ    DIV             decrement loop count
                LAC     ONE,4
                SACL    ROFF            round off before output
                B       FDCT

*********************************************
*       VISIBILITY THRESHOLD MATRIX         *
*********************************************

* this INVERSE matrix is scaled up 8192 and ZIGZAG *

ITHLD           DATA    2621,1802,1966,2294,1966,1638,2621,2294
                DATA    2130,2294,2949,2785,2621,3113,3932,6554
                DATA    4260,3932,3604,3604,3932,8028,5734,6062
                DATA    4751,6554,9503,8356,9994,9830,9339,8356
                DATA    9175,9011,10486,11796,15073,12780,10486,11141
                DATA    14254,11305,9011,9175,13107,17859,13271,14254
                DATA    15565,16056,16876,17039,16876,10158,12616,18514
                DATA    19825,18350,16384,19661,15073,16548,16876,16220

* this matrix is scaled up 4096 and zigzag *

THLD            DATA    12800,18618,17067,14629,17067,20480,12800,14629
                DATA    15754,14629,11378,12047,12800,10779,8533,5120
                DATA    7877,8533,9309,9309,8533,4180,5851,5535
                DATA    7062,5120,3531,4016,3357,3413,3593,4016
```

```
                DATA    3657,3724,3200,2844,2226,2626,3200,3012
                DATA    2354,2968,3724,3657,2560,1879,2528,3593
                DATA    2156,2090,1988,1969,1988,3303,2660,1812
                DATA    1693,1829,2048,1707,2226,2028,1988,2069

*------------------------------BEGIN TRANSFORM-----------------------------*

FDCT            LARK    AR0,FIR0                start addr for input data
                LARK    AR1,63                  count for input
                OUT     NOWIN,STAWRT            now open gate for in data
WAIT3           BIOZ    WAIT3                   now in
                LARP    AR0
                IN      *+,INPORT,AR1
                BANZ    WAIT3

                DINT                            now decrease block count
                LAC     BLKIN
                SUB     ONE
                SACL    BLKIN
                EINT
                LDPK    0

******************      state 1     ******************

                SLSR    FIR0,FIR56,SEC0,6       *inputs are in Q8 so translate
                SLSR    FIR1,FIR57,SEC1,6        to Q0
                SLSR    FIR9,FIR49,SEC2,6
                SLSR    FIR8,FIR48,SEC3,6
                SLSR    FIR11,FIR51,SEC4,6
                SLSR    FIR10,FIR50,SEC5,6
                SLSR    FIR2,FIR58,SEC6,6
                SLSR    FIR3,FIR59,SEC7,6

                SLSR    FIR27,FIR35,SEC8,6
                SLSR    FIR26,FIR34,SEC9,6
                SLSR    FIR18,FIR42,SEC10,6
                SLSR    FIR19,FIR43,SEC11,6
                SLSR    FIR16,FIR40,SEC12,6
                SLSR    FIR17,FIR41,SEC13,6
                SLSR    FIR25,FIR33,SEC14,6
                SLSR    FIR24,FIR32,SEC15,6

                SLSR    FIR31,FIR39,SEC16,6
                SLSR    FIR30,FIR38,SEC17,6
                SLSR    FIR22,FIR46,SEC18,6
                SLSR    FIR23,FIR47,SEC19,6
                SLSR    FIR20,FIR44,SEC20,6
                SLSR    FIR21,FIR45,SEC21,6
                SLSR    FIR29,FIR37,SEC22,6
                SLSR    FIR28,FIR36,SEC23,6

                SLSR    FIR4,FIR60,SEC24,6
                SLSR    FIR5,FIR61,SEC25,6
                SLSR    FIR13,FIR53,SEC26,6
                SLSR    FIR12,FIR52,SEC27,6
                SLSR    FIR15,FIR55,SEC28,6
                SLSR    FIR14,FIR54,SEC29,6
                SLSR    FIR6,FIR62,SEC30,6
                SLSR    FIR7,FIR63,SEC31,6

                DLSR    FIR7,FIR63,SEC32,6
                DLSR    FIR6,FIR62,SEC33,6
```

```
        DLSR    FIR14,FIR54,SEC34,6
        DLSR    FIR15,FIR55,SEC35,6
        DLSR    FIR12,FIR52,SEC36,6
        DLSR    FIR13,FIR53,SEC37,6
        DLSR    FIR5,FIR61,SEC38,6
        DLSR    FIR4,FIR60,SEC39,6

        DLSR    FIR28,FIR36,SEC40,6
        DLSR    FIR29,FIR37,SEC41,6
        DLSR    FIR21,FIR45,SEC42,6
        DLSR    FIR20,FIR44,SEC43,6
        DLSR    FIR23,FIR47,SEC44,6
        DLSR    FIR22,FIR46,SEC45,6
        DLSR    FIR30,FIR38,SEC46,6
        DLSR    FIR31,FIR39,SEC47,6

        DLSR    FIR24,FIR32,SEC48,6
        DLSR    FIR25,FIR33,SEC49,6
        DLSR    FIR17,FIR41,SEC50,6
        DLSR    FIR16,FIR40,SEC51,6
        DLSR    FIR19,FIR43,SEC52,6
        DLSR    FIR18,FIR42,SEC53,6
        DLSR    FIR26,FIR34,SEC54,6
        DLSR    FIR27,FIR35,SEC55,6

        DLSR    FIR3,FIR59,SEC56,6
        DLSR    FIR2,FIR58,SEC57,6
        DLSR    FIR10,FIR50,SEC58,6
        DLSR    FIR11,FIR51,SEC59,6
        DLSR    FIR8,FIR48,SEC60,6
        DLSR    FIR9,FIR49,SEC61,6
        DLSR    FIR1,FIR57,SEC62,6
        DLSR    FIR0,FIR56,SEC63,6

********************    state 2    ********************

        SLSL    SEC0,SEC31,FIR0
        SLSL    SEC1,SEC30,FIR1
        SLSL    SEC2,SEC29,FIR2
        SLSL    SEC3,SEC28,FIR3
        SLSL    SEC4,SEC27,FIR4
        SLSL    SEC5,SEC26,FIR5
        SLSL    SEC6,SEC25,FIR6
        SLSL    SEC7,SEC24,FIR7

        SLSL    SEC8,SEC23,FIR8
        SLSL    SEC9,SEC22,FIR9
        SLSL    SEC10,SEC21,FIR10
        SLSL    SEC11,SEC20,FIR11
        SLSL    SEC12,SEC19,FIR12
        SLSL    SEC13,SEC18,FIR13
        SLSL    SEC14,SEC17,FIR14
        SLSL    SEC15,SEC16,FIR15

        DLSL    SEC15,SEC16,FIR16
        DLSL    SEC14,SEC17,FIR17
        DLSL    SEC13,SEC18,FIR18
        DLSL    SEC12,SEC19,FIR19
        DLSL    SEC11,SEC20,FIR20
        DLSL    SEC10,SEC21,FIR21
        DLSL    SEC9,SEC22,FIR22
        DLSL    SEC8,SEC23,FIR23
```

```
        DLSL    SEC7,SEC24,FIR24
        DLSL    SEC6,SEC25,FIR25
        DLSL    SEC5,SEC26,FIR26
        DLSL    SEC4,SEC27,FIR27
        DLSL    SEC3,SEC28,FIR28
        DLSL    SEC2,SEC29,FIR29
        DLSL    SEC1,SEC30,FIR30
        DLSL    SEC0,SEC31,FIR31

        SLSL    SEC32,SEC63,FIR32
        SLSL    SEC33,SEC62,FIR33
        SLSL    SEC34,SEC61,FIR34
        SLSL    SEC35,SEC60,FIR35
        SLSL    SEC36,SEC59,FIR36
        SLSL    SEC37,SEC58,FIR37
        SLSL    SEC38,SEC57,FIR38
        SLSL    SEC39,SEC56,FIR39

        SLSL    SEC40,SEC55,FIR40
        SLSL    SEC41,SEC54,FIR41
        SLSL    SEC42,SEC53,FIR42
        SLSL    SEC43,SEC52,FIR43
        SLSL    SEC44,SEC51,FIR44
        SLSL    SEC45,SEC50,FIR45
        SLSL    SEC46,SEC49,FIR46
        SLSL    SEC47,SEC48,FIR47

        DLSL    SEC47,SEC48,FIR48
        DLSL    SEC46,SEC49,FIR49
        DLSL    SEC45,SEC50,FIR50
        DLSL    SEC44,SEC51,FIR51
        DLSL    SEC43,SEC52,FIR52
        DLSL    SEC42,SEC53,FIR53
        DLSL    SEC41,SEC54,FIR54
        DLSL    SEC40,SEC55,FIR55

        DLSL    SEC39,SEC56,FIR56
        DLSL    SEC38,SEC57,FIR57
        DLSL    SEC37,SEC58,FIR58
        DLSL    SEC36,SEC59,FIR59
        DLSL    SEC35,SEC60,FIR60
        DLSL    SEC34,SEC61,FIR61
        DLSL    SEC33,SEC62,FIR62
        DLSL    SEC32,SEC63,FIR63

********************    state 3    ********************

        SLSL    FIR0,FIR15,SEC0
        SLSL    FIR1,FIR14,SEC1
        SLSL    FIR2,FIR13,SEC2
        SLSL    FIR3,FIR12,SEC3
        SLSL    FIR4,FIR11,SEC4
        SLSL    FIR5,FIR10,SEC5
        SLSL    FIR6,FIR9,SEC6
        SLSL    FIR7,FIR8,SEC7

        DLSL    FIR7,FIR8,SEC8
        DLSL    FIR6,FIR9,SEC9
        DLSL    FIR5,FIR10,SEC10
        DLSL    FIR4,FIR11,SEC11
        DLSL    FIR3,FIR12,SEC12
```

```
DLSL    FIR2,FIR13,SEC13
DLSL    FIR1,FIR14,SEC14
DLSL    FIR0,FIR15,SEC15

SLSL    FIR16,FIR31,SEC16
SLSL    FIR17,FIR30,SEC17
SLSL    FIR18,FIR29,SEC18
SLSL    FIR19,FIR28,SEC19
SLSL    FIR20,FIR27,SEC20
SLSL    FIR21,FIR26,SEC21
SLSL    FIR22,FIR25,SEC22
SLSL    FIR23,FIR24,SEC23

DLSL    FIR23,FIR24,SEC24
DLSL    FIR22,FIR25,SEC25
DLSL    FIR21,FIR26,SEC26
DLSL    FIR20,FIR27,SEC27
DLSL    FIR19,FIR28,SEC28
DLSL    FIR18,FIR29,SEC29
DLSL    FIR17,FIR30,SEC30
DLSL    FIR16,FIR31,SEC31

SLSL    FIR32,FIR39,SEC32
SLSL    FIR33,FIR38,SEC33
SLSL    FIR34,FIR37,SEC34
SLSL    FIR35,FIR36,SEC35
DLSL    FIR35,FIR36,SEC36
DLSL    FIR34,FIR37,SEC37
DLSL    FIR33,FIR38,SEC38
DLSL    FIR32,FIR39,SEC39

DLSL    FIR47,FIR40,SEC40
DLSL    FIR46,FIR41,SEC41
DLSL    FIR45,FIR42,SEC42
DLSL    FIR44,FIR43,SEC43
SLSL    FIR44,FIR43,SEC44
SLSL    FIR45,FIR42,SEC45
SLSL    FIR46,FIR41,SEC46
SLSL    FIR47,FIR40,SEC47

SLSL    FIR48,FIR56,SEC48
SLSL    FIR49,FIR57,SEC49
SLSL    FIR50,FIR58,SEC50
SLSL    FIR51,FIR59,SEC51
SLSL    FIR52,FIR60,SEC52
SLSL    FIR53,FIR61,SEC53
SLSL    FIR54,FIR62,SEC54
SLSL    FIR55,FIR63,SEC55

DLSL    FIR48,FIR56,SEC56
DLSL    FIR49,FIR57,SEC57
DLSL    FIR50,FIR58,SEC58
DLSL    FIR51,FIR59,SEC59
DLSL    FIR52,FIR60,SEC60
DLSL    FIR53,FIR61,SEC61
DLSL    FIR54,FIR62,SEC62
DLSL    FIR55,FIR63,SEC63

********************    state 4    ********************

SLSL    SEC0,SEC7,FIR0
SLSL    SEC1,SEC6,FIR1
```

```
SLSL    SEC2,SEC5,FIR2
SLSL    SEC3,SEC4,FIR3
DLSL    SEC3,SEC4,FIR4,1
DLSL    SEC2,SEC5,FIR5,1
DLSL    SEC1,SEC6,FIR6,1
DLSL    SEC0,SEC7,FIR7,1

SLSL    SEC8,SEC15,FIR8,1
SLSL    SEC9,SEC14,FIR9,1
SLSL    SEC10,SEC13,FIR10,1
SLSL    SEC11,SEC12,FIR11,1
DLSL    SEC11,SEC12,FIR12
DLSL    SEC10,SEC13,FIR13
DLSL    SEC9,SEC14,FIR14
DLSL    SEC15,SEC8,FIR15

SLSL    SEC16,SEC23,FIR16
SLSL    SEC17,SEC18,FIR17,1
DLSL    SEC17,SEC18,FIR18,1
SLSL    SEC19,SEC20,FIR19
DLSL    SEC19,SEC20,FIR20
SLSL    SEC21,SEC22,FIR21,1
DLSL    SEC22,SEC21,FIR22,1
DLSL    SEC16,SEC23,FIR23

SLSL    SEC24,SEC28,FIR24
SLSL    SEC25,SEC29,FIR25
SLSL    SEC26,SEC30,FIR26,1
SLSL    SEC27,SEC31,FIR27,1
DLSL    SEC24,SEC28,FIR28,1
DLSL    SEC29,SEC25,FIR29,1
DLSL    SEC26,SEC30,FIR30
DLSL    SEC27,SEC31,FIR31

SLSL    SEC32,SEC33,FIR32
DLSL    SEC32,SEC33,FIR33
SLSL    SEC34,SEC35,FIR34,1
DLSL    SEC34,SEC35,FIR35,1
SLSL    SEC36,SEC42,FIR36
SLSL    SEC37,SEC43,FIR37,1
SLSL    SEC38,SEC40,FIR38,1
SLSL    SEC39,SEC41,FIR39

DLSL    SEC38,SEC40,FIR40
DLSL    SEC41,SEC39,FIR41,1
DLSL    SEC42,SEC36,FIR42,1
DLSL    SEC37,SEC43,FIR43
DLSL    SEC44,SEC45,FIR44,1
SLSL    SEC45,SEC44,FIR45,1
DLSL    SEC47,SEC46,FIR46
SLSL    SEC47,SEC46,FIR47

SLSL    SEC48,SEC61,FIR48
SLSL    SEC49,SEC60,FIR49
SLSL    SEC50,SEC63,FIR50
SLSL    SEC51,SEC62,FIR51
SLSL    SEC52,SEC54,FIR52
SLSL    SEC53,SEC55,FIR53
DLSL    SEC52,SEC54,FIR54
DLSL    SEC53,SEC55,FIR55

SLSL    SEC56,SEC58,FIR56
```

```
                SLSL    SEC57,SEC59,FIR57
                DLSL    SEC56,SEC58,FIR58
                DLSL    SEC57,SEC59,FIR59
                DLSL    SEC49,SEC60,FIR60
                DLSL    SEC48,SEC61,FIR61
                DLSL    SEC51,SEC62,FIR62
                DLSL    SEC50,SEC63,FIR63

**********          state 5     **********

*In this state coef in position 17,18,21,22,26-29,34,35,37,38,41,42,44,45
* 49,51,53,58 will not be process *

                SLSL    FIR0,FIR3,SEC0
                SLSL    FIR1,FIR2,SEC1
                DLSL    FIR1,FIR2,SEC2
                DLSL    FIR0,FIR3,SEC3
                SLSL    FIR4,FIR7,SEC4
                SLSL    FIR5,FIR6,SEC5
                DLSL    FIR6,FIR5,SEC6
                DLSL    FIR7,FIR4,SEC7

                SLSL    FIR8,FIR9,SEC8
                DLSL    FIR8,FIR9,SEC9
                SLSL    FIR10,FIR11,SEC10
                DLSL    FIR10,FIR11,SEC11
                SLSL    FIR12,FIR14,SEC12
                DLSL    FIR15,FIR13,SEC13
                DLSL    FIR12,FIR14,SEC14
                SLSL    FIR13,FIR15,SEC15

                DLSL    FIR16,FIR19,SEC16
                SLSL    FIR16,FIR19,SEC19
                DLSL    FIR20,FIR23,SEC20
                SLSL    FIR20,FIR23,SEC23

                DLSL    FIR24,FIR31,SEC24
                DLSL    FIR25,FIR30,SEC25
                SLSL    FIR25,FIR30,SEC30
                SLSL    FIR24,FIR31,SEC31

                SLSL    FIR32,FIR47,SEC32
                SLSL    FIR33,FIR46,SEC33
                SLSL    FIR36,FIR43,SEC36
                SLSL    FIR39,FIR40,SEC39

                DLSL    FIR39,FIR40,SEC40
                DLSL    FIR36,FIR43,SEC43
                DLSL    FIR33,FIR46,SEC46
                DLSL    FIR32,FIR47,SEC47

                SLSL    FIR48,FIR54,SEC48
                SLSL    FIR50,FIR57,SEC50
                SLSL    FIR52,FIR63,SEC52
                DLSL    FIR48,FIR54,SEC54
                SLSL    FIR55,FIR56,SEC55

                DLSL    FIR55,FIR56,SEC56
                DLSL    FIR50,FIR57,SEC57
                SLSL    FIR59,FIR61,SEC59
                SLSL    FIR60,FIR62,SEC60
                DLSL    FIR61,FIR59,SEC61
```

```
                DLSL    FIR60,FIR62,SEC62
                DLSL    FIR52,FIR63,SEC63

******************      state 6     ******************

* the first 12 coef will be reversely processed at the end of this state *

                LDPK    1                       Change page to get ONE
                LT      ONE
                MPYK    C4
                PAC
                LDPK    0                       Restore working page
                SACL    FIR0                    use FIR0 as temporary storage
                LT      FIR0                    to load T register with C4

                MPY     SEC15
                PAC
                MPY     SEC12
                APAC
                SACH    SEC12,4                 for C4 is in Q12 form

                PAC
                MPY     SEC15
                SPAC
                SACH    SEC15,4

                MPY     FIR18                   last state did not do anything
                LAC     SEC16,12                scale up to be in Q12
                SPAC
                SACH    SEC18,4

                LAC     SEC16,12
                APAC
                SACH    SEC16,4

                MPY     FIR17                   last state did not do anything
                PAC
                ADD     SEC19,12
                SACH    SEC17,4

                LAC     SEC19,12
                SPAC
                SACH    SEC19,4

                MPY     FIR22                   last state did not do anything
                PAC
                ADD     SEC20,12
                SACH    SEC22,4

                PAC
                SUB     SEC20,12
                SACH    SEC20,4

                MPY     FIR21                   last state did not do anything
                PAC
                ADD     SEC23,12
                SACH    SEC21,4

                PAC
                SUB     SEC23,12
                SACH    SEC23,4
```

```
MPY     FIR34                   last state did not do anything
LAC     SEC32,12
SPAC
SACH    SEC34,4

LAC     SEC32,12
APAC
SACH    SEC32,4

MPY     FIR35                   last state did not do anything
LAC     SEC33,12
APAC
SACH    SEC35,4

LAC     SEC33,12
SPAC
SACH    SEC33,4

MPY     FIR44                   last state did not do anything
LAC     SEC46,12
SPAC
SACH    SEC44,4

LAC     SEC46,12
APAC
SACH    SEC46,4

MPY     FIR45                   last state did not do anything
LAC     SEC47,12
SPAC
SACH    SEC45,4

LAC     SEC47,12
APAC
SACH    SEC47,4

MPY     SEC56
PAC
ADD     FIR49,12                by pass from last state
SACH    SEC56,4

PAC
SUB     FIR49,12
SACH    SEC49,4

MPY     SEC55
PAC
SUB     FIR51,12                by pass from last state
SACH    SEC55,4

PAC
ADD     FIR51,12
SACH    SEC51,4

MPY     SEC60
PAC
SUB     FIR53,12                by pass from last state
SACH    SEC60,4

ZAC
SUB     FIR53,12
```

```
SPAC
SACH    SEC53,4

MPY     SEC62
LAC     FIR58,12            by pass from last state
SPAC
SACH    SEC62,4

LAC     FIR58,12
LTA     SEC50
SACH    SEC58,4

MPYK    C6
PAC
LT      SEC48
MPYK    C2
SPAC
SACH    SEC48,4

MPYK    C6
PAC
LT      SEC50
MPYK    C2
LTA     SEC52               load T register for next mpy
SACH    SEC50,4

MPYK    C6
PAC
LT      SEC61
MPYK    C2
SPAC
SACH    SEC61,4

MPYK    C6
PAC
LT      SEC52
MPYK    C2
LTA     SEC57               load T register for next mpy
SACH    SEC52,4

MPYK    C6
PAC
LT      SEC54
MPYK    C2
SPAC
SACH    SEC54,4

MPYK    C6
PAC
LT      SEC57
MPYK    C2
LTA     SEC59               load T register for next mpy
SACH    SEC57,4

MPYK    C6
PAC
LT      SEC63
MPYK    C2
SPAC
SACH    SEC63,4

MPYK    C6
```

```
        PAC
        LT      SEC59
        MPYK    C2
        LTA     FIR0                    that is loading C4
        SACH    SEC59,4

* These coefs will have 1 another state to do *

        MPY     FIR26                   last state did not do anything
        LAC     SEC24,12
        SPAC
        SACH    FIR26,4

        PAC
        ADD     SEC24,12
        SACH    FIR24,4

        MPY     FIR27                   last state did not do anything
        LAC     SEC25,12
        SPAC
        SACH    FIR27,4

        LAC     SEC25,12
        APAC
        SACH    FIR25,4

        MPY     FIR28                   last state did not do anything
        PAC
        SUB     SEC30,12
        SACH    FIR28,4

        PAC
        ADD     SEC30,12
        SACH    FIR30,4

        MPY     FIR29                   last state did not do anything
        LAC     SEC31,12
        SPAC
        SACH    FIR29,4

        LAC     SEC31,12
        APAC
        SACH    FIR31,4

        MPY     FIR38                   last state did not do anything
        LAC     SEC36,12
        APAC
        SACH    FIR36,4

        LAC     SEC36,12
        SPAC
        SACH    FIR38,4

        MPY     FIR37                   last state did not do anything
        PAC
        ADD     SEC39,12
        SACH    FIR37,4

        PAC
        SUB     SEC39,12
        SACH    FIR39,4
```

```
                MPY     FIR42                   last state did not do anything
                LAC     SEC40,12
                APAC
                SACH    FIR40,4

                LAC     SEC40,12
                SPAC
                SACH    FIR42,4

                MPY     FIR41                   last state did not do anything
                PAC
                ADD     SEC43,12
                SACH    FIR41,4

                PAC
                SUB     SEC43,12
                SACH    FIR43,4

****************    state 7    ****************

                LT      FIR24
                MPYK    C41
                PAC
                LT      FIR31
                MPYK    C47
                SPAC
                SACH    SEC31,4                 Store at SEC for next state

                MPYK    C41
                PAC
                LT      FIR24
                MPYK    C47
                LTA     FIR25
                SACH    SEC24,4                 Store at SEC for next state

                MPYK    C41
                PAC
                LT      FIR30
                MPYK    C47
                SPAC
                SACH    SEC30,4                 store at SEC for next state

                MPYK    C41
                PAC
                LT      FIR25
                MPYK    C47
                LTA     FIR29
                SACH    SEC25,4                 store at SEC for next state

                MPYK    C43
                PAC
                LT      FIR26
                MPYK    C45
                SPAC
                SACH    SEC26,4                 store at SEC for next state

                MPYK    C43
                PAC
                LT      FIR29
                MPYK    C45
                LTA     FIR27
                SACH    SEC29,4                 store at SEC for next state
```

```
        MPYK    C43
        PAC
        LT      FIR28
        MPYK    C45
        SPAC
        SACH    SEC28,4                 store at SEC for next state

        MPYK    C43
        PAC
        LT      FIR27
        MPYK    C45
        LTA     SEC4
        SACH    SEC27,4                 store at SEC for next state

        SLSL    FIR40,FIR36,SEC36
        SLSL    FIR41,FIR37,SEC37
        SLSL    FIR42,FIR38,SEC38
        SLSL    FIR43,FIR39,SEC39
        DLSL    FIR40,FIR36,SEC40
        DLSL    FIR41,FIR37,SEC41
        DLSL    FIR42,FIR38,SEC42
        DLSL    FIR43,FIR39,SEC43


**********          state 8     **********

* These coefs will be stored at the final position with permutation
* and zigzag scan but not scaling by 4 yet

        DLSL    SEC0,SEC1,FIR14
        SLSL    SEC0,SEC1,FIR0
        SLSL    SEC3,SEC2,FIR10
        DLSL    SEC3,SEC2,FIR39

        MPYK    C46
        PAC
        LT      SEC5
        MPYK    C42
        SPAC
        SACH    FIR27,4

        MPYK    C46
        PAC
        LT      SEC4
        MPYK    C42
        LTA     SEC7
        SACH    FIR5,4

        MPYK    C46
        PAC
        LT      SEC6
        MPYK    C42
        SPAC
        SACH    FIR52,4

        MPYK    C46
        PAC
        LT      SEC7
        MPYK    C42
        LTA     SEC8
        SACH    FIR23,4
```

```
MPYK    C46
PAC
LT      SEC10
MPYK    C42
SPAC
SACH    FIR21,4

MPYK    C46
PAC
LT      SEC8
MPYK    C42
LTA     SEC9
SACH    FIR3,4

MPYK    C42
PAC
LT      SEC11
MPYK    C46
SPAC
SACH    FIR25,4

MPYK    C42
PAC
LT      SEC9
MPYK    C46
LTA     SEC20
SACH    FIR50,4

SLSL    SEC12,SEC14,FIR37
DLSL    SEC13,SEC15,FIR12
DLSL    SEC12,SEC14,FIR41
SLSL    SEC15,SEC13,FIR59

MPYK    C3
PAC
LT      SEC16
MPYK    C5
SPAC
SACH    FIR54,4

MPYK    C3
PAC
LT      SEC20
MPYK    C5
LTA     SEC21
SACH    FIR19,4

MPYK    C3
PAC
LT      SEC17
MPYK    C5
SPAC
SACH    FIR28,4

MPYK    C3
PAC
LT      SEC21
MPYK    C5
LTA     SEC18
SACH    FIR1,4
```

```
MPYK    C7
PAC
LT      SEC22
MPYK    C1
SPAC
SACH    FIR32,4

MPYK    C7
PAC
LT      SEC18
MPYK    C1
LTA     SEC19
SACH    FIR45,4

MPYK    C7
PAC
LT      SEC23
MPYK    C1
SPAC
SACH    FIR6,4

MPYK    C7
PAC
LT      SEC19
MPYK    C1
LTA     SEC47
SACH    FIR15,4

ZAC
SUB     SEC31
SUB     SEC30
SACL    FIR8

SLSL    SEC26,SEC27,FIR17
DLSL    SEC28,SEC29,FIR30
DLSL    SEC24,SEC25,FIR34
SLSL    SEC24,SEC25,FIR43
SLSL    SEC28,SEC29,FIR47
DLSL    SEC26,SEC27,FIR56
DLSL    SEC31,SEC30,FIR61

MPYK    C3
PAC
LT      SEC32
MPYK    C5
SPAC
SACH    FIR35,4

MPYK    C3
PAC
LT      SEC47
MPYK    C5
LTA     SEC46
SACH    FIR2,4

MPYK    C3
PAC
LT      SEC33
MPYK    C5
SPAC
SACH    FIR57,4
```

```
MPYK    C3
PAC
LT      SEC46
MPYK    C5
LTA     SEC34
SACH    FIR16,4

MPYK    C1
PAC
LT      SEC45
MPYK    C7
SPAC
SACH    FIR20,4

MPYK    C1
PAC
LT      SEC34
MPYK    C7
LTA     SEC35
SACH    FIR9,4

MPYK    C1
PAC
LT      SEC44
MPYK    C7
SPAC
SACH    FIR46,4

MPYK    C1
PAC
LT      SEC35
MPYK    C7
LTA     SEC37
SACH    FIR31,4

MPYK    C45
PAC
LT      SEC36
MPYK    C43
SPAC
SACH    FIR48,4

MPYK    C45
PAC
LT      SEC37
MPYK    C43
LTA     SEC38
SACH    FIR7,4

MPYK    C41
PAC
LT      SEC39
MPYK    C47
SPAC
SACH    FIR44,4

MPYK    C41
PAC
LT      SEC38
MPYK    C47
LTA     SEC40
SACH    FIR55,4
```

```
                MPYK    C45
                PAC
                LT      SEC41
                MPYK    C43
                SPAC
                SACH    FIR62,4

                MPYK    C45
                PAC
                LT      SEC40
                MPYK    C43
                LTA     SEC43
                SACH    FIR29,4

                MPYK    C41
                PAC
                LT      SEC42
                MPYK    C47
                SPAC
                SACH    FIR18,4

                MPYK    C41
                PAC
                LT      SEC43
                MPYK    C47
                APAC
                SACH    FIR33,4

                SLSL    SEC55,SEC48,FIR26
                DLSL    SEC49,SEC54,FIR53
                SLSL    SEC50,SEC51,FIR11
                DLSL    SEC50,SEC51,FIR60
                DLSL    SEC53,SEC52,FIR4
                SLSL    SEC53,SEC52,FIR63
                SLSL    SEC49,SEC54,FIR22
                DLSL    SEC55,SEC48,FIR49

                SLSL    SEC57,SEC56,FIR13
                DLSL    SEC57,SEC56,FIR58
                DLSL    SEC59,SEC58,FIR42
                SLSL    SEC59,SEC58,FIR36
                DLSL    SEC60,SEC61,FIR24
                SLSL    SEC60,SEC61,FIR51

                ZAC
                SUB     SEC62
                SUB     SEC63
                SACL    FIR38

                DLSL    SEC62,SEC63,FIR40

*--------------------------------END TRANSFORM-----------------------------*

* now begin quantization by matrix that is already adjusted by threshold factor

                LDPK    1                       point to working area
                LARK    AR0,MATX63              point to coef and matrix
                LARK    AR1,FIR63
                LARP    AR1
QUAN            LT      *,AR0                   load matrix in Q15 form
                MPY     *-,AR1                  multiply with coef in Q15 form
```

```
                PAC
                BGEZ    CONT
                ADDH    ROFF                    add one if less than zero
CONT            SACH    *,4                     now store in Q8 form
                BANZ    QUAN

                LARK    AR0,FIR0
                LARK    AR1,63
                OUT     NOWOUT,STAWRT
WAIT4           BIOZ    WAIT4                   now out data to IBM
                LARP    AR0
                OUT     *+,OUTPOR,AR1
                BANZ    WAIT4

                DINT
                LAC     BLKOUT                  decrease block count
                SUB     ONE
                SACL    BLKOUT
                EINT
                BGZ     FDCT                    if not finished do again

                ZALS    NOWIN                   generate interrupt to IBM
                SUB     ONE,12                  after finished
                SACL    NOWIN
                OUT     NOWIN,STAWRT
                B       START

*--------------------------------END QUANTIZE-----------------------------*

*-----------------------------BEGIN  DEQUANTIZE---------------------------*

IFDCT           LARK    AR0,FIR0                now in data
                LARK    AR1,63
                OUT     NOWIN,STAWRT
WAIT5           BIOZ    WAIT5
                LARP    AR0
                IN      *+,INPORT,AR1
                BANZ    WAIT5

                DINT                            decrease block count
                LAC     BLKIN
                SUB     ONE
                SACL    BLKIN
                EINT

* now begin dequantize coefficients *

                LARK    AR0,MATX63
                LARK    AR1,FIR63
                LARP    AR1
DEQUAN          LT      *,AR0                   load matrix in 11.5 form
                MPY     *-,AR1                  multiply with coef in 8.8 form
                PAC
                ADD     ONE,11                  round off
                SACH    *,4                     now restore in 12.1 form
                BANZ    DEQUAN

                LARK    AR0,ROFINV
                LARP    AR0
                LDPK    0

*----------------------------BEGIN INVERSE TRANSFORM----------------------*
```

```
********************  state 1 ***************************

        SLSL    FIR0,FIR14,SEC0
        DLSL    FIR0,FIR14,SEC1
        DLSL    FIR10,FIR39,SEC2
        SLSL    FIR10,FIR39,SEC3

        LT      FIR5
        MPYK    C46
        PAC
        LT      FIR27
        MPYK    C42
        SPAC
        SACH    SEC5,4

        MPYK    C46
        PAC
        LT      FIR5
        MPYK    C42
        LTA     FIR23
        SACH    SEC4,4

        MPYK    C46
        PAC
        LT      FIR52
        MPYK    C42
        SPAC
        SACH    SEC6,4

        MPYK    C46
        PAC
        LT      FIR23
        MPYK    C42
        LTA     FIR3
        SACH    SEC7,4

        MPYK    C46
        PAC
        LT      FIR21
        MPYK    C42
        SPAC
        SACH    SEC10,4

        MPYK    C46
        PAC
        LT      FIR3
        MPYK    C42
        LTA     FIR50
        SACH    SEC8,4

        MPYK    C42
        PAC
        LT      FIR25
        MPYK    C46
        SPAC
        SACH    SEC11,4

        MPYK    C42
        PAC
        LT      FIR50
        MPYK    C46
```

```
LTA     FIR19
SACH    SEC9,4

SLSL    FIR37,FIR41,SEC12
SLSL    FIR59,FIR12,SEC13
DLSL    FIR37,FIR41,SEC14
DLSL    FIR59,FIR12,SEC15

MPYK    C3
PAC
LT      FIR54
MPYK    C5
SPAC
SACH    SEC16,4

MPYK    C3
PAC
LT      FIR19
MPYK    C5
LTA     FIR1
SACH    SEC20,4

MPYK    C3
PAC
LT      FIR28
MPYK    C5
SPAC
SACH    SEC17,4

MPYK    C3
PAC
LT      FIR1
MPYK    C5
LTA     FIR45
SACH    SEC21,4

MPYK    C7
PAC
LT      FIR32
MPYK    C1
SPAC
SACH    SEC22,4

MPYK    C7
PAC
LT      FIR45
MPYK    C1
LTA     FIR15
SACH    SEC18,4

MPYK    C7
PAC
LT      FIR6
MPYK    C1
SPAC
SACH    SEC23,4

MPYK    C7
PAC
LT      FIR15
MPYK    C1
LTA     FIR2
```

```
SACH    SEC19,4

SLSL    FIR43,FIR34,SEC24
DLSL    FIR43,FIR34,SEC25
SLSL    FIR17,FIR56,SEC26
DLSL    FIR17,FIR56,SEC27
SLSL    FIR47,FIR30,SEC28
DLSL    FIR47,FIR30,SEC29
SLSL    FIR61,FIR8,SEC30
DLSL    FIR61,FIR8,SEC31

MPYK    C3
PAC
LT      FIR35
MPYK    C5
SPAC
SACH    SEC32,4

MPYK    C3
PAC
LT      FIR2
MPYK    C5
LTA     FIR16
SACH    SEC47,4

MPYK    C3
PAC
LT      FIR57
MPYK    C5
SPAC
SACH    SEC33,4

MPYK    C3
PAC
LT      FIR16
MPYK    C5
LTA     FIR9
SACH    SEC46,4

MPYK    C1
PAC
LT      FIR20
MPYK    C7
SPAC
SACH    SEC45,4

MPYK    C1
PAC
LT      FIR9
MPYK    C7
LTA     FIR31
SACH    SEC34,4

MPYK    C1
PAC
LT      FIR46
MPYK    C7
SPAC
SACH    SEC44,4

MPYK    C1
PAC
```

```
LT      FIR31
MPYK    C7
LTA     FIR7
SACH    SEC35,4

MPYK    C45
PAC
LT      FIR48
MPYK    C43
SPAC
SACH    SEC36,4

MPYK    C45
PAC
LT      FIR7
MPYK    C43
LTA     FIR55
SACH    SEC37,4

MPYK    C41
PAC
LT      FIR44
MPYK    C47
SPAC
SACH    SEC39,4

MPYK    C41
PAC
LT      FIR55
MPYK    C47
LTA     FIR29
SACH    SEC38,4

MPYK    C45
PAC
LT      FIR62
MPYK    C43
SPAC
SACH    SEC41,4

MPYK    C45
PAC
LT      FIR29
MPYK    C43
LTA     FIR33
SACH    SEC40,4

MPYK    C41
PAC
LT      FIR18
MPYK    C47
SPAC
SACH    SEC42,4

MPYK    C41
PAC
LT      FIR33
MPYK    C47
APAC
SACH    SEC43,4

DLSL    FIR26,FIR49,SEC48
```

```
        SLSL    FIR53,FIR22,SEC49
        SLSL    FIR11,FIR60,SEC50
        DLSL    FIR11,FIR60,SEC51
        DLSL    FIR63,FIR4,SEC52
        SLSL    FIR63,FIR4,SEC53
        DLSL    FIR22,FIR53,SEC54
        SLSL    FIR49,FIR26,SEC55

        DLSL    FIR58,FIR13,SEC56
        SLSL    FIR13,FIR58,SEC57
        DLSL    FIR36,FIR42,SEC58
        SLSL    FIR36,FIR42,SEC59
        SLSL    FIR24,FIR51,SEC60
        DLSL    FIR51,FIR24,SEC61
        DLSL    FIR40,FIR38,SEC62
        SLSL    FIR40,FIR38,SEC63

**********      state 2     **********

        LDPK    1
        LT      ONE
        MPYK    C4
        PAC
        LDPK    0
        SACL    FIR0                    That is to load T with C4
        LT      FIR0

        MPY     SEC12
        PAC
        MPY     SEC15
        SPAC
        SACH    SEC15,4

        PAC
        MPY     SEC12
        APAC
        SACH    SEC12,4

        MPY     SEC16
        SLSL    SEC16,SEC18,SEC16
        PAC
        MPY     SEC18
        SPAC
        SACH    FIR18,4                 for by pass in state 4

        MPY     SEC19
        SLSL    SEC17,SEC19,SEC19
        ZAC
        SPAC
        MPY     SEC17
        APAC
        SACH    FIR17,4                 for by pass in state 4

        MPY     SEC20
        DLSL    SEC22,SEC20,SEC20
        PAC
        MPY     SEC22
        APAC
        SACH    FIR22,4                 for by pass in state 4

        MPY     SEC23
        DLSL    SEC21,SEC23,SEC23
```

```
PAC
MPY     SEC21
APAC
SACH    FIR21,4                 for by pass in state 4

MPY     SEC32
SLSL    SEC32,SEC34,SEC32
PAC
MPY     SEC34
SPAC
SACH    FIR34,4                 for by pass in state 4

MPY     SEC33
SLSL    SEC33,SEC35,SEC33
PAC
MPY     SEC35
SPAC
SACH    FIR35,4                 for by pass in state 4

MPY     SEC46
SLSL    SEC46,SEC44,SEC46
PAC
MPY     SEC44
SPAC
SACH    FIR44,4                 for by pass in state 4

MPY     SEC47
SLSL    SEC47,SEC45,SEC47
PAC
MPY     SEC45
SPAC
SACH    FIR45,4                 for by pass in state 4

MPY     SEC49
SLSL    SEC56,SEC49,FIR49
PAC
MPY     SEC56
SPAC
SACH    SEC56,4

MPY     SEC51
DLSL    SEC51,SEC55,FIR51
PAC
MPY     SEC55
APAC
SACH    SEC55,4

MPY     SEC53
SLSL    SEC60,SEC53,FIR53
PAC
MPY     SEC60
SPAC
SACH    SEC60,4

MPY     SEC58
SLSL    SEC58,SEC62,FIR58
PAC
MPY     SEC62
SPAC
SACH    SEC62,4

LT      SEC50
```

```
MPYK    C6
PAC
LT      SEC48
MPYK    C2
SPAC
SACH    SEC48,4

MPYK    C6
PAC
LT      SEC50
MPYK    C2
LTA     SEC52
SACH    SEC50,4

MPYK    C6
PAC
LT      SEC61
MPYK    C2
SPAC
SACH    SEC61,4

MPYK    C6
PAC
LT      SEC52
MPYK    C2
LTA     SEC57
SACH    SEC52,4

MPYK    C6
PAC
LT      SEC54
MPYK    C2
SPAC
SACH    SEC54,4

MPYK    C6
PAC
LT      SEC57
MPYK    C2
LTA     SEC63
SACH    SEC57,4

MPYK    C6
PAC
LT      SEC59
MPYK    C2
SPAC
SACH    SEC59,4

MPYK    C6
PAC
LT      SEC63
MPYK    C2
LTA     SEC24
SACH    SEC63,4

MPYK    C41
PAC
LT      SEC31
MPYK    C47
SPAC
SACH    FIR31,4
```

```
        MPYK    C41
        PAC
        LT      SEC24
        MPYK    C47
        LTA     SEC25
        SACH    FIR24,4

        MPYK    C47
        PAC
        LT      SEC30
        MPYK    C41
        SPAC
        SACH    FIR25,4

        MPYK    C47
        PAC
        LT      SEC25
        MPYK    C41
        LTA     SEC26
        SACH    FIR30,4

        MPYK    C45
        PAC
        LT      SEC29
        MPYK    C43
        SPAC
        SACH    FIR26,4

        MPYK    C45
        PAC
        LT      SEC26
        MPYK    C43
        LTA     SEC27
        SACH    FIR29,4

        MPYK    C43
        PAC
        LT      SEC28
        MPYK    C45
        SPAC
        SACH    FIR28,4

        MPYK    C43
        PAC
        LT      SEC27
        MPYK    C45
        LTA     FIR0            this still equals to C4
        SACH    FIR27,4

        DLSL    SEC36,SEC40,FIR36
        DLSL    SEC37,SEC41,FIR37
        DLSL    SEC38,SEC42,FIR38
        DLSL    SEC39,SEC43,FIR39
        SLSL    SEC36,SEC40,FIR40
        SLSL    SEC37,SEC41,FIR41
        SLSL    SEC38,SEC42,FIR42
        SLSL    SEC39,SEC43,FIR43

****************    state_3    ****************

        DLSL    FIR24,FIR26,SEC24
```

```
        SLSL    FIR25,FIR27,SEC25
        DLSL    FIR30,FIR28,SEC30
        SLSL    FIR31,FIR29,SEC31

        MPY     FIR24
        PAC
        MPY     FIR26
        APAC
        SACH    FIR26,4                 For by pass in state 4

        MPY     FIR25
        PAC
        MPY     FIR27
        SPAC
        SACH    FIR27,4

        MPY     FIR28
        PAC
        MPY     FIR30
        APAC
        SACH    FIR28,4

        MPY     FIR29
        PAC
        MPY     FIR31
        SPAC
        SACH    FIR29,4

        SLSL    FIR36,FIR38,SEC36
        DLSL    FIR37,FIR39,SEC39
        SLSL    FIR40,FIR42,SEC40
        DLSL    FIR41,FIR43,SEC43

        MPY     FIR37
        PAC
        MPY     FIR39
        APAC
        SACH    FIR37,4

        MPY     FIR36
        PAC
        MPY     FIR38
        SPAC
        SACH    FIR38,4

        MPY     FIR41
        PAC
        MPY     FIR43
        APAC
        SACH    FIR41,4

        MPY     FIR40
        PAC
        MPY     FIR42
        SPAC
        SACH    FIR42,4

**********      state 4     **********

        SLSL    SEC0,SEC3,FIR0
        SLSL    SEC1,SEC2,FIR1
        DLSL    SEC1,SEC2,FIR2
```

```
DLSL    SEC0,SEC3,FIR3
DLSL    SEC4,SEC7,FIR4,1
SLSL    SEC5,SEC6,FIR5,1
DLSL    SEC5,SEC6,FIR6,1
SLSL    SEC4,SEC7,FIR7,1

SLSL    SEC8,SEC9,FIR8,1
DLSL    SEC8,SEC9,FIR9,1
SLSL    SEC10,SEC11,FIR10,1
DLSL    SEC10,SEC11,FIR11,1
SLSL    SEC12,SEC14,FIR12
DLSL    SEC15,SEC13,FIR13
DLSL    SEC12,SEC14,FIR14
SLSL    SEC15,SEC13,FIR15

SLSL    SEC16,SEC19,FIR16
DLSL    SEC19,SEC16,FIR19
SLSL    SEC20,SEC23,FIR20
DLSL    SEC23,SEC20,FIR23

SLSL    SEC24,SEC31,FIR24
SLSL    SEC25,SEC30,FIR25
DLSL    SEC30,SEC25,FIR30
DLSL    SEC31,SEC24,FIR31

SLSL    SEC32,SEC47,FIR32
SLSL    SEC33,SEC46,FIR33
SLSL    SEC36,SEC43,FIR36
SLSL    SEC39,SEC40,FIR39

DLSL    SEC39,SEC40,FIR40
DLSL    SEC36,SEC43,FIR43
DLSL    SEC33,SEC46,FIR46
DLSL    SEC32,SEC47,FIR47

SLSL    SEC48,SEC54,FIR48
SLSL    SEC50,SEC57,FIR50
SLSL    SEC52,SEC63,FIR52
DLSL    SEC48,SEC54,FIR54
SLSL    SEC55,SEC56,FIR55

DLSL    SEC55,SEC56,FIR56
DLSL    SEC50,SEC57,FIR57
SLSL    SEC59,SEC61,FIR59
DLSL    SEC62,SEC60,FIR60
DLSL    SEC61,SEC59,FIR61
SLSL    SEC60,SEC62,FIR62
DLSL    SEC52,SEC63,FIR63

********************    state 5    ********************

SLSL    FIR0,FIR7,SEC0
SLSL    FIR1,FIR6,SEC1
SLSL    FIR2,FIR5,SEC2
SLSL    FIR3,FIR4,SEC3
DLSL    FIR3,FIR4,SEC4
DLSL    FIR2,FIR5,SEC5
DLSL    FIR1,FIR6,SEC6
DLSL    FIR0,FIR7,SEC7

DLSL    FIR8,FIR15,SEC8
SLSL    FIR9,FIR14,SEC9
```

```
SLSL    FIR10,FIR13,SEC10
SLSL    FIR11,FIR12,SEC11
DLSL    FIR11,FIR12,SEC12
DLSL    FIR10,FIR13,SEC13
DLSL    FIR9,FIR14,SEC14
SLSL    FIR8,FIR15,SEC15

SLSL    FIR16,FIR23,SEC16
SLSL    FIR17,FIR18,SEC17,1
DLSL    FIR17,FIR18,SEC18,1
SLSL    FIR19,FIR20,SEC19
DLSL    FIR19,FIR20,SEC20
DLSL    FIR21,FIR22,SEC21,1
SLSL    FIR21,FIR22,SEC22,1
DLSL    FIR16,FIR23,SEC23

LAC     FIR24
ADD     FIR28,1
SACL    SEC24

LAC     FIR25
ADD     FIR29,1
SACL    SEC25

LAC     FIR26,1
ADD     FIR30
SACL    SEC26

LAC     FIR27,1
ADD     FIR31
SACL    SEC27

LAC     FIR24
SUB     FIR28,1
SACL    SEC28

LAC     FIR25
SUB     FIR29,1
SACL    SEC29

LAC     FIR26,1
SUB     FIR30
SACL    SEC30

LAC     FIR27,1
SUB     FIR31
SACL    SEC31

SLSL    FIR32,FIR33,SEC32
DLSL    FIR32,FIR33,SEC33
DLSL    FIR34,FIR35,SEC34,1
SLSL    FIR35,FIR34,SEC35,1

LAC     FIR36
SUB     FIR42,1
SACL    SEC36

LAC     FIR37,1
ADD     FIR43
SACL    SEC37

LAC     FIR38,1
```

```
                ADD     FIR40
                SACL    SEC38

                LAC     FIR39
                SUB     FIR41,1
                SACL    SEC39

                LAC     FIR38,1
                SUB     FIR40
                SACL    SEC40

                LAC     FIR39
                ADD     FIR41,1
                SACL    SEC41

                LAC     FIR36
                ADD     FIR42,1
                SACL    SEC42

                LAC     FIR37,1
                SUB     FIR43
                SACL    SEC43

                SLSL    FIR45,FIR44,SEC44,1
                DLSL    FIR45,FIR44,SEC45,1
                DLSL    FIR47,FIR46,SEC46
                SLSL    FIR46,FIR47,SEC47

                SLSL    FIR48,FIR61,SEC48
                DLSL    FIR60,FIR49,SEC49
                SLSL    FIR50,FIR63,SEC50
                DLSL    FIR51,FIR62,SEC51
                SLSL    FIR52,FIR54,SEC52
                DLSL    FIR55,FIR53,SEC53
                DLSL    FIR52,FIR54,SEC54
                ZAC
                SUB     FIR53
                SUB     FIR55
                SACL    SEC55

                SLSL    FIR56,FIR58,SEC56
                DLSL    FIR57,FIR59,SEC57
                DLSL    FIR56,FIR58,SEC58
                SLSL    FIR57,FIR59,SEC59
                ZAC
                SUB     FIR49
                SUB     FIR60
                SACL    SEC60
                DLSL    FIR48,FIR61,SEC61
                SLSL    FIR51,FIR62,SEC62
                DLSL    FIR50,FIR63,SEC63

********************    state 6    ********************

                SLSL    SEC0,SEC15,FIR0,3
                SLSL    SEC1,SEC14,FIR1,3
                SLSL    SEC2,SEC13,FIR2,3
                SLSL    SEC3,SEC12,FIR3,3
                SLSL    SEC4,SEC11,FIR4,3
                SLSL    SEC5,SEC10,FIR5,3
                SLSL    SEC6,SEC9,FIR6,3
                SLSL    SEC7,SEC8,FIR7,3
```

```
DLSL    SEC7,SEC8,FIR8,3
DLSL    SEC6,SEC9,FIR9,3
DLSL    SEC5,SEC10,FIR10,3
DLSL    SEC4,SEC11,FIR11,3
DLSL    SEC3,SEC12,FIR12,3
DLSL    SEC2,SEC13,FIR13,3
DLSL    SEC1,SEC14,FIR14,3
DLSL    SEC0,SEC15,FIR15,3

SLSL    SEC16,SEC31,FIR16,3
SLSL    SEC17,SEC30,FIR17,3
SLSL    SEC18,SEC29,FIR18,3
SLSL    SEC19,SEC28,FIR19,3
SLSL    SEC20,SEC27,FIR20,3
SLSL    SEC21,SEC26,FIR21,3
SLSL    SEC22,SEC25,FIR22,3
SLSL    SEC23,SEC24,FIR23,3

DLSL    SEC23,SEC24,FIR24,3
DLSL    SEC22,SEC25,FIR25,3
DLSL    SEC21,SEC26,FIR26,3
DLSL    SEC20,SEC27,FIR27,3
DLSL    SEC19,SEC28,FIR28,3
DLSL    SEC18,SEC29,FIR29,3
DLSL    SEC17,SEC30,FIR30,3
DLSL    SEC16,SEC31,FIR31,3

SLSL    SEC32,SEC39,FIR32,3
SLSL    SEC33,SEC38,FIR33,3
SLSL    SEC34,SEC37,FIR34,3
SLSL    SEC35,SEC36,FIR35,3
DLSL    SEC35,SEC36,FIR36,3
DLSL    SEC34,SEC37,FIR37,3
DLSL    SEC33,SEC38,FIR38,3
DLSL    SEC32,SEC39,FIR39,3

DLSL    SEC47,SEC40,FIR40,3
DLSL    SEC46,SEC41,FIR41,3
DLSL    SEC45,SEC42,FIR42,3
DLSL    SEC44,SEC43,FIR43,3
SLSL    SEC44,SEC43,FIR44,3
SLSL    SEC45,SEC42,FIR45,3
SLSL    SEC46,SEC41,FIR46,3
SLSL    SEC47,SEC40,FIR47,3

SLSL    SEC48,SEC56,FIR48,3
SLSL    SEC49,SEC57,FIR49,3
SLSL    SEC50,SEC58,FIR50,3
SLSL    SEC51,SEC59,FIR51,3
SLSL    SEC52,SEC60,FIR52,3
SLSL    SEC53,SEC61,FIR53,3
SLSL    SEC54,SEC62,FIR54,3
SLSL    SEC55,SEC63,FIR55,3

DLSL    SEC48,SEC56,FIR56,3
DLSL    SEC49,SEC57,FIR57,3
DLSL    SEC50,SEC58,FIR58,3
DLSL    SEC51,SEC59,FIR59,3
DLSL    SEC52,SEC60,FIR60,3
DLSL    SEC53,SEC61,FIR61,3
DLSL    SEC54,SEC62,FIR62,3
```

```
                DLSL    SEC55,SEC63,FIR63,3

********************    state 7    ********************

                SUMH    FIR0,FIR31,SEC0
                SUMH    FIR1,FIR30,SEC1
                SUMH    FIR2,FIR29,SEC2
                SUMH    FIR3,FIR28,SEC3
                SUMH    FIR4,FIR27,SEC4
                SUMH    FIR5,FIR26,SEC5
                SUMH    FIR6,FIR25,SEC6
                SUMH    FIR7,FIR24,SEC7

                SUMH    FIR8,FIR23,SEC8
                SUMH    FIR9,FIR22,SEC9
                SUMH    FIR10,FIR21,SEC10
                SUMH    FIR11,FIR20,SEC11
                SUMH    FIR12,FIR19,SEC12
                SUMH    FIR13,FIR18,SEC13
                SUMH    FIR14,FIR17,SEC14
                SUMH    FIR15,FIR16,SEC15

                DIFH    FIR15,FIR16,SEC16
                DIFH    FIR14,FIR17,SEC17
                DIFH    FIR13,FIR18,SEC18
                DIFH    FIR12,FIR19,SEC19
                DIFH    FIR11,FIR20,SEC20
                DIFH    FIR10,FIR21,SEC21
                DIFH    FIR9,FIR22,SEC22
                DIFH    FIR8,FIR23,SEC23

                DIFH    FIR7,FIR24,SEC24
                DIFH    FIR6,FIR25,SEC25
                DIFH    FIR5,FIR26,SEC26
                DIFH    FIR4,FIR27,SEC27
                DIFH    FIR3,FIR28,SEC28
                DIFH    FIR2,FIR29,SEC29
                DIFH    FIR1,FIR30,SEC30
                DIFH    FIR0,FIR31,SEC31

                SUMH    FIR32,FIR63,SEC32
                SUMH    FIR33,FIR62,SEC33
                SUMH    FIR34,FIR61,SEC34
                SUMH    FIR35,FIR60,SEC35
                SUMH    FIR36,FIR59,SEC36
                SUMH    FIR37,FIR58,SEC37
                SUMH    FIR38,FIR57,SEC38
                SUMH    FIR39,FIR56,SEC39

                SUMH    FIR40,FIR55,SEC40
                SUMH    FIR41,FIR54,SEC41
                SUMH    FIR42,FIR53,SEC42
                SUMH    FIR43,FIR52,SEC43
                SUMH    FIR44,FIR51,SEC44
                SUMH    FIR45,FIR50,SEC45
                SUMH    FIR46,FIR49,SEC46
                SUMH    FIR47,FIR48,SEC47

                DIFH    FIR47,FIR48,SEC48
                DIFH    FIR46,FIR49,SEC49
                DIFH    FIR45,FIR50,SEC50
                DIFH    FIR44,FIR51,SEC51
```

```
          DIFH    FIR43,FIR52,SEC52
          DIFH    FIR42,FIR53,SEC53
          DIFH    FIR41,FIR54,SEC54
          DIFH    FIR40,FIR55,SEC55

          DIFH    FIR39,FIR56,SEC56
          DIFH    FIR38,FIR57,SEC57
          DIFH    FIR37,FIR58,SEC58
          DIFH    FIR36,FIR59,SEC59
          DIFH    FIR35,FIR60,SEC60
          DIFH    FIR34,FIR61,SEC61
          DIFH    FIR33,FIR62,SEC62
          DIFH    FIR32,FIR63,SEC63

********************    state 8    ********************

          SUMH    SEC0,SEC63,FIR0,R
          SUMH    SEC1,SEC62,FIR1,R
          SUMH    SEC2,SEC61,FIR9,R
          SUMH    SEC3,SEC60,FIR8,R
          SUMH    SEC4,SEC59,FIR11,R
          SUMH    SEC5,SEC58,FIR10,R
          SUMH    SEC6,SEC57,FIR2,R
          SUMH    SEC7,SEC56,FIR3,R

          SUMH    SEC8,SEC55,FIR27,R
          SUMH    SEC9,SEC54,FIR26,R
          SUMH    SEC10,SEC53,FIR18,R
          SUMH    SEC11,SEC52,FIR19,R
          SUMH    SEC12,SEC51,FIR16,R
          SUMH    SEC13,SEC50,FIR17,R
          SUMH    SEC14,SEC49,FIR25,R
          SUMH    SEC15,SEC48,FIR24,R

          SUMH    SEC16,SEC47,FIR31,R
          SUMH    SEC17,SEC46,FIR30,R
          SUMH    SEC18,SEC45,FIR22,R
          SUMH    SEC19,SEC44,FIR23,R
          SUMH    SEC20,SEC43,FIR20,R
          SUMH    SEC21,SEC42,FIR21,R
          SUMH    SEC22,SEC41,FIR29,R
          SUMH    SEC23,SEC40,FIR28,R

          SUMH    SEC24,SEC39,FIR4,R
          SUMH    SEC25,SEC38,FIR5,R
          SUMH    SEC26,SEC37,FIR13,R
          SUMH    SEC27,SEC36,FIR12,R
          SUMH    SEC28,SEC35,FIR15,R
          SUMH    SEC29,SEC34,FIR14,R
          SUMH    SEC30,SEC33,FIR6,R
          SUMH    SEC31,SEC32,FIR7,R

          DIFH    SEC31,SEC32,FIR63,R
          DIFH    SEC30,SEC33,FIR62,R
          DIFH    SEC29,SEC34,FIR54,R
          DIFH    SEC28,SEC35,FIR55,R
          DIFH    SEC27,SEC36,FIR52,R
          DIFH    SEC26,SEC37,FIR53,R
          DIFH    SEC25,SEC38,FIR61,R
          DIFH    SEC24,SEC39,FIR60,R

          DIFH    SEC23,SEC40,FIR36,R
```

```
          DIFH    SEC22,SEC41,FIR37,R
          DIFH    SEC21,SEC42,FIR45,R
          DIFH    SEC20,SEC43,FIR44,R
          DIFH    SEC19,SEC44,FIR47,R
          DIFH    SEC18,SEC45,FIR46,R
          DIFH    SEC17,SEC46,FIR38,R
          DIFH    SEC16,SEC47,FIR39,R

          DIFH    SEC15,SEC48,FIR32,R
          DIFH    SEC14,SEC49,FIR33,R
          DIFH    SEC13,SEC50,FIR41,R
          DIFH    SEC12,SEC51,FIR40,R
          DIFH    SEC11,SEC52,FIR43,R
          DIFH    SEC10,SEC53,FIR42,R
          DIFH    SEC9,SEC54,FIR34,R
          DIFH    SEC8,SEC55,FIR35,R

          DIFH    SEC7,SEC56,FIR59,R
          DIFH    SEC6,SEC57,FIR58,R
          DIFH    SEC5,SEC58,FIR50,R
          DIFH    SEC4,SEC59,FIR51,R
          DIFH    SEC3,SEC60,FIR48,R
          DIFH    SEC2,SEC61,FIR49,R
          DIFH    SEC1,SEC62,FIR57,R
          DIFH    SEC0,SEC63,FIR56,R

*------------------------ END OF INVERSE TRANSFORM ----------------------*

          LDPK    1                       switch to working page
          LARK    AR0,FIR0                count and address for out data
          LARK    AR1,63
          OUT     NOWOUT,STAWRT
WAIT6     BIOZ    WAIT6                   now out to IBM
          LARP    AR0
          OUT     *+,OUTPOR,AR1
          BANZ    WAIT6

          DINT                            decrease block count
          LAC     BLKOUT
          SUB     ONE
          SACL    BLKOUT
          EINT
          BGZ     IFDCT                   do next block

          ZALS    NOWIN                   generate interrupt to IBM
          SUB     ONE,12                  after finished
          SACL    NOWIN
          OUT     NOWIN,STAWRT
          B       START

          END
```

ประวัติผู้เขียน

นาย บุญช่วย ทรัพย์มนชัย เกิดเมื่อวันที่ 16 พฤษภาคม พ.ศ. 2506 ที่ อ. สัมพันธวงศ์ กรุงเทพมหานครฯ สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า (เกียรตินิยมอันดับ 2) จาก คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2529 จากนั้นก็เข้าศึกษาต่อในระดับปริญญามหาบัณฑิต สาขาวิศวกรรมไฟฟ้า สังกัดห้องปฏิบัติการวิจัยไฟฟ้าสื่อสาร คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย โดยมีความสนใจทางด้าน Digital Signal Processing, การประมวลผลภาพ และ การสื่อสารข้อมูล มีบทความเคยได้รับการตีพิมพ์ในการประชุมทางวิศวกรรมไฟฟ้าระดับอุดมศึกษา 9 สถาบันในปี 2532 เรื่อง การส่งภาพนิ่งแบบโปรเกรสซีฟ