Design of microprocessor-based hardware for number theoretic transform implementation

Shamim, Anwar Ahmed

DESIGN OF MICROPROCESSOR-BASED HARDWARE FOR NUMBER THEORETIC TRANSFORM IMPLEMENTATION

ANWAR A. SHAMIM

Vol II

FULLY DOCUMENTED PROGRAM LISTINGS APPEARING IN
THE APPENDICES A - E

Appendix-A

Modular arithmetic routines for the following microprocessors

i) TMS9900

ii) MC6809

iii) Z80

iv) 6502

32/16-bit division routine for the MC6809 microprocessor

ALL THE MODULAR ARITHMETIC ROUTINES IN APPENDIX-A CONSISTS OF UNSIGNED (16-BIT) ADDITION, SUBTRACTION AND MULTIPLICATION MODULO 65521. PLEASE REFER TO SECTION 3.6 IN THE THESIS FOR EXAMPLES. THESE BENCHMARK PROGRAMS WERE SPECIFICALLY WRITTEN TO TEST AND COMPARE THE PERFORMANCE OF THE FOLLOWING MICORPROCESSORS TMS9900, MC6809, Z80, 6502, FOR THE DESIGN OF THE PARALLEL MICROPROCESSOR SYSTEM. IN ADDITION A 32-BIT / 16-BIT UNSIGNED DIVIDE ROUTINE IS ALSO INCLUDED.

THE METHOD USED FOR MODULARISING THE 32-BIT UNSIGNED PRODUCT INTO 16-BIT PRODUCT REDUCED MODULO 65521, IS AS FOLLOWS. LET XXXX REPRESENT A 16-BIT UNSIGNED NUMBER, AND XX REPRESENT AN 8-BIT UNSIGNED NUMBER.

```
                                    XXXX
                                    XXXX
    ------------------------------------
          XX       XX               XXXX   32-BIT PRODUCT
        8-BITS   8-BITS           16-BITS
       (HI-MSB) (LO-MSB)           (LSB)
          |        |
          |        |
          |  MULTIPLY BY
          |  15 (MOD 65521)
          |  AND ADD (MOD 65521)->
          |
    MULTIPLY BY
    3840 (MOD 65521)
    AND ADD (MOD 65521)---------->
    ------------------------------------
                                   XXXX    16-BIT PRODUCT MOD 65521
```

```
*
*************************************************************
*  MODULAR ARITHMETIC ROUTINES FOR TMS9900 MICROPROCESSOR  *
*************************************************************
*
         OPTION XREF,SYMT
         AORG   >4000
*
*************************************************************
* ROUTINE : MODULAR ADDITION                                *
* PURPOSE : PERFORM UNSIGNED ADDITION MODULO 65521          *
*************************************************************
*
START    LWPI   WKS           LOAD WORKSPACE
         MOV    @AD1,R1       LOAD IST OPERAND
         MOV    @AD2,R2       LOAD 2ND OPERAND
         A      R1,R2         R2=R1+R2
         JNC    OVER          BRANCH TO OVER FOR NO CARRY
         AI     R2,15         ADD 15 IF A CARRY IS GENERATED
         JMP    OVR
OVER     CI     R2,65521      COMPARE R2 WITH 65521
         JL     OVR
         AI     R2,15         ADD 15 IF THE SUM EXCEEDS 65521
OVR      MOV    R2,@SUM       SAVE RESULT IN SUM
*
*************************************************************
* ROUTINE : MODULAR SUBRTRACTION                            *
* PURPOSE : PERFORM UNSIGNED SUBTRACTION MODULO 65521       *
*************************************************************
*
         MOV    @SUBT1,R1     LOAD IST OPERAND
         MOV    @SUBT2,R2     LOAD 2ND OPERAND
         MOV    R1,R3         COPY R1 INTO R3
         S      R2,R1         R1=R1-R2
         C      R3,R2         COMPARE R1 AND R2
         JHE    OVER1         IF SUBTRAHEND > MINUEND ADD
         AI     R1,65521      65521 TO IT, OTHERWISE IGNORE
OVER1    MOV    R1,@RES       SAVE RESULT IN RES
*
*************************************************************
* ROUTINE : MODULAR MULTIPLICATION                          *
* PURPOSE : REDUCE 32-BIT UNSIGNED PRODUCT MODULO 65521     *
*************************************************************
*
         MOV    @MPR,R1       MOVE MULTIPLIER TO R1
         MOV    @MPD,R2       MOVE MULTIPLICAND TO R2
         MPY    R1,R2         2-BIT PRODUCT IN R2:R3
         DIV    @MOD,R2       DIVIDE BY 65521
         MOV    R3,@PROD      REMAINDER IN R3
         B      @>0080        BRANCH TO MONITOR
*
WKS      BSS    32            WORKSPACE AREA
AD1      BSS    2
```

```
AD2      BSS    2
SUM      BSS    2
SUBT1    BSS    2
SUBT2    BSS    2
RES      BSS    2
MPR      BSS    2
MPD      BSS    2
PROD     BSS    2
MOD      DATA   65521
LAST     END    START
*
*
*
*************************************************************
*  MODULAR ARITHMETIC ROUTINES FOR MC6809 MICROPROCESSOR    *
*************************************************************
*
         NAM    M6809
         OPT    CRE,L,S,W,P
         ORG    $30
*
************************************************************
* ROUTINE : MODULAR ADDITION                               *
* PURPOSE : PERFORM UNSIGNED ADDITION MODULO 65521         *
************************************************************
*
START    LDX    #ADS          INTITIALISE THE INDEX REGISTER
         LDD    ,X++          FETCH FIRST OPERAND IN D
         ADDD   ,X++          ADD SECOND OPERAND
         BCS    SKIP
         CMPD   #65521        COMPARE SUM WITH 65521
         BLO    SKIP1
SKIP     ADDD   #15           ADD 15 IF SUM EXCEEDS 65521
SKIP1    STD    ,X            SAVE RESULTS
         JMP    OVER
ADS      FDB    0
         FDB    0
         FDB    0
*
************************************************************
* ROUTINE : MODULAR SUBTRACTION                            *
* PURPOSE : PERFORM UNSIGNED SUBTRACTION MODULO 65521      *
************************************************************
*
OVER     LDX    #SBTN         INITIALISE INDEX REGISTER
         LDD    ,X++          FETCH MINUEND
         SUBD   ,X++          SUBTRACT 2ND OPERAND
         BCC    SKIP2
         ADDD   #65521        ADD 65521 IF SUBTRAHEND WAS
*                             GREATER THAN MINUEND
SKIP2    STD    ,X            SAVE RESULTS
         JMP    OVER1
SBTN     FDB    0
         FDB    0
         FDB    0
*
***********************************************************
* ROUTINE : MODULAR MULTIPLICATION                        *
* PURPOSE : PERFORM UNSIGNED 16*16-BIT MULTIPLICATION AND *
*           REDUCE THE PRODUCT MODULO 65521               *
***********************************************************
*
OVER1    LDX    #MLTR
         LDY    #MLTN
         LDU    #PROD1
         CLR    0,U           CLEAR PROD1
         CLR    1,U           CLEAR PROD2
         LDA    1,X           LOAD LS BYTE OF MULTIPLIER
         LDB    1,Y           LOAD LS BYTE OF MULTIPLICAND
         MUL
         STD    2,U           SAVE 16-BIT PRODUCT IN PROD3:PROD4
         LDA    0,X           LOAD MS BYTE OF MULTIPLIER
         LDB    1,Y           LOAD LS BYTE OF MULTIPLICAND
         MUL
         ADDD   1,U           ADD PREVIOUS PARTIAL PRODUCT
*                             WHILE MAINTAINING THE SIGNIFICANCE
         STD    1,U           SAVE IN PROD2:PROD3
         BCC    SKIP3
         INC    0,U           INCREMENT PROD1 IF THERE WAS A CARRY
SKIP3    LDA    1,X           LOAD LS BYTE OF MULTIPLIER
         LDB    0,Y           LOAD MS BYTE OF MULTIPLICAND
         MUL
         ADDD   1,U           ADD PROD2:PROD3
         STD    1,U           SAVE IN PROD2:PROD3
         BCC    SKIP4
         INC    0,U           INCREMENT PROD1 IF THERE WAS A CARRY
SKIP4    LDA    0,X           LOAD MS BYTE OF MULTIPLIER
         LDB    0,Y           LOAD MS BYTE OF MULTIPLICAND
         MUL
         ADDD   0,U           ADD PROD1:PROD2
         STD    0,U           SAVE IN PROD1:PROD2
*
***********************************************************
* NOW THE 32-BIT UNSIGNED PRODUCT IS IN                   *
* PROD1:PROD2:PROD3:PROD4                                 *
***********************************************************
*
***********************************************************
* ROUTINE : REDUCE PRODUCT MODULO 65521                   *
* PURPOSE : ROUTINE FOR REDUCING 32-BIT UNSIGNED PRODUCT  *
*           MODULO 65521, BY USING MUL INSTRUCTION        *
***********************************************************
*
         LDA    1,U           GET PROD2
         BEQ    OMIT          IGNORE MULTIPLICATION IF PROD2 = 0
         LDB    #15
         MUL
         ADDD   2,U           ADD INTO PROD3:PROD4
         BCS    SKIP6
         CMPD   #65521
         BLO    SKIP7
SKIP6    ADDD   #15           ADD 15 IF SUM EXCEEDS 65521
SKIP7    STD    2,U
OMIT     LDA    0,U           GET PROD1
         BEQ    OMIT1         IGNORE MULTIPLICATION IF PROD1 = 0
SKIP8    LDY    #TEMP
         CLR    0,Y
         CLR    1,Y
         CLR    2,Y
         LDB    #15           MULTIPLY BY 3840 AND ADD IN PROD3:PROD4
         MUL                  (HEX EQUIVALENT OF 3840 $0F00)
SKIPA    STD    0,Y           REDUCE PRODUCT BY 3840 MODULO 65521
         LDA    0,Y
         BEQ    SKIPE
         LDB    #15
         MUL
         ADDD   1,Y
         BRA    SKIPD
SKIPE    LDD    1,Y
SKIPD    ADDD   2,U           ADD INTO PROD3:PROD4
         BCS    SKIPB
         CMPD   #65521
         BLO    SKIPC
SKIPB    ADDD   #15           ADD 15 IF SUM EXCEEDS 65521
SKIPC    STD    2,U
OMIT1    JMP    $F564         RETURN TO EXBUG
MLTR     FDB    0
MLTN     FDB    0
PROD1    FCB    0
PROD2    FCB    0
PROD3    FCB    0
PROD4    FCB    0
TEMP     FDB    0
         END
*
;
;
;
; **********************************************************
; *  MODULAR ARITHMETIC ROUTINES FOR Z80 MICROPROCESSOR    *
; **********************************************************
;
; **********************************************************
; * ROUTINE : MODULAR ADDITION                             *
; * PURPOSE : PERFORM UNSIGNED ADDITION MODULO 65521       *
; **********************************************************
         ORG    100H
START:   LD     HL,(ADD1)     ;LOAD CONTENTS OF ADD1 IN H:L
         LD     BC,(ADD2)     ;LOAD CONTENTS OF ADD2 IN B:C
         ADD    HL,BC         ;ADD B:C TO H:L RESULT IN H:L
         JP     C,OVER1       ;IF CARRY SET GOTO OVER1
         LD     A,255         ;LOAD REG A WITH FF
         CP     H             ;COMPARE WITH REG H
         JP     NZ,OVER       ;C=0 WHEN A>H
;                             ;C=1 WHEN A<H
;                             ;Z=0 WHEN A=H
         LD     A,241         ;LOAD REG A WITH F1
         CP     L             ;COMPARE WITH REG L
         JP     Z,OVER1
         JP     NC,OVER
OVER1:   LD     BC,15
         ADD    HL,BC         ;ADD 15 TO H:L
         LD     (SUM),HL      ;STORE RESULT IN SUM
OVER:    JP     SKIP
;
;        DATA FOR PROCESSING
;
ADD1:    DEFW   0
ADD2:    DEFW   0
SUM:     DEFW   0
;
; *********************************************************
; * ROUTINE : MODULAR SUBTRACTION                         *
; * PURPOSE : PERFORM UNSIGNED SUBTRACTION MODULO 65521   *
; *********************************************************
;
SKIP:    LD     HL,(SUBT1)    ;LOAD 1ST OPERAND IN H:L
         LD     DE,(SUBT3)    ;LOAD 2ND OPERAND IN D:E
         AND    A             ;CLEAR CARRY FLAG
         SBC    HL,DE         ;SUBTRACT D:E FROM H:L
         LD     A,(SUBT3)     ;LOAD REG A WITH LS BYTE
         LD     D,A           ;TRANSFER TO REG D
         LD     A,(SUBT1)     ;LOAD A WITH LS BYTE OF OTHER OPERAND
         CP     D             ;COMPARE REG A & REG D
         JP     NC,OVR        ;IF REG A > REG D GOTO OVR
         JP     Z,ZERO        ;IF A=D GOTO ZERO
BACK:    LD     BC,65521
         ADD    HL,BC
         JP     OVR
ZERO:    LD     A,(SUBT4)     ;LOAD A WITH MS BYTE
         LD     D,A           ;TRANSFER TO D
         LD     A,(SUBT2)     ;LOAD A WITH MS BYTE
         CP     D             ;COMPARE REG A & REG D
         JP     NC,OVR
         JP     Z,OVR
         JP     BACK
OVR:     LD     (RES),HL      ;STORE RESULT IN RES
         JP     SKIP2
;
;         DATA FOR SUBTRACTION
;
SUBT1:   DEFB   0
SUBT2:   DEFB   0
SUBT3:   DEFB   0
SUBT4:   DEFB   0
RES:     DEFW   0
```

```
;
; ************************************************************
; * ROUTINE : MODULAR MULTIPLICATION                          *
; * PURPOSE : REDUCE 32-BIT UNSIGNED PRODUCT MODULO 65521     *
; ************************************************************
;
SKIP2:   LD      A,(MPR1)        ;LOAD LS BYTE OF MULTIPLER
         LD      H,A
         LD      A,(MPD1)        ;LOAD LS BYTE OF MULTIPLICAND
         LD      E,A
         CALL    MULT            ;CALL SUBROUTINE MULT
         LD      (PROD3),HL      ;PROD3:PROD4 CONTAIN L*L
         LD      A,(MPR2)        ;LOAD MS BYTE OF MPR
         LD      H,A
         LD      A,(MPD2)        ;LOAD MS BYTE OF MPD
         LD      E,A
         CALL    MULT
         LD      (PROD1),HL      ;PROD1:PROD2 CONTAIN H*H
         LD      A,(MPR2)        ;LOAD LS BYTE OF MPR
         LD      H,A
         LD      A,(MPD1)        ;LOAD MS BYTE OF MPD
         LD      E,A
         CALL    MULT
         LD      (PROD5),HL      ;PROD5:PROD6 CONTAINS H*L
         LD      A,(MPR1)        ;LOAD MS BYTE OF MPR
         LD      H,A
         LD      A,(MPD2)        ;LOAD LS BYTE OF MPD
         LD      E,A
         CALL    MULT
         LD      DE,(PROD5)
         ADD     HL,DE           ;ADD PROD5:PROD6 TO H:L
         JP      NC,BAK
         LD      B,1
         LD      A,(PROD2)
         ADD     A,B
         LD      (PROD2),A
BAK:     LD      (PROD5),HL
         LD      A,(PROD4)
         LD      E,A
         LD      A,(PROD1)
         LD      D,A
         ADD     HL,DE
         JP      NC,BAK1
         LD      B,1
         LD      A,(PROD2)
         ADD     A,B
         LD      (PROD2),A
BAK1:    LD      (PROD5),HL
         LD      A,H
         LD      (PROD1),A
         LD      A,L
         LD      (PROD4),A
; ************************************************************
; * 32-BIT UNSIGNED PRODUCT NOW IN PROD1:PROD2:PROD3:PROD4*
; **********************************************************
;
; **********************************************************
; * NOW TO REDUCE 32-BIT UNSIGNED PRODUCT MODULO 65521     *
; **********************************************************
;
          LD     A,(PROD1)
          LD     H,A
          LD     E,15
          CALL   MULT
          LD     DE,(PROD3)
          ADD    HL,DE
          JP     NC,BAK2
          LD     BC,15
          ADD    HL,BC          ;ADD 15 IF CARRY FLAG SET
          LD     (PROD3),HL
          JP     BAK3
BAK2:     LD     (PROD3),HL
          LD     A,255          ;LOAD REG A WITH FF
          CP     H              ;COMPARE WITH H
          JP     NZ,BAK3
          LD     A,241          ;LOAD REG A WITH F1
          CP     L              ;COMPARE WITH REG L
          JP     Z,BAK6
          JP     NC,BAK3
BAK6:     LD     BC,15
          ADD    HL,BC          ;ADD 15 IF CARRY FLAG SET
          LD     (PROD3),HL
BAK3:     LD     A,(PROD2)
          LD     H,A
          LD     E,15
          CALL   MULT
          LD     A,L            ;LOAD LS BYTE OF PRODUCT IN A
          LD     (TMP2),A       ;STORE IN TMP2
;
;      H ALREADY CONTAINS MS BYTE OF PROD1*15
;
          LD     A,0
          LD     (TMP1),A       ;INITIALISING TMP1=0
          LD     E,15
          CALL   MULT
          LD     DE,(TMP1)
          ADD    HL,DE
          LD     DE,(PROD3)
          ADD    HL,DE
          JP     NC,BAK4
          LD     BC,15
          ADD    HL,BC          ;ADD 15 IF CARRY FLAG SET
          JP     BAK5
BAK4:     LD     (PROD3),HL
          LD     A,255          ;LOAD REG A WITH $FF
          CP     H              ;COMPARE WITH H
          JP     NZ,BAK5
          LD     A,241          ;LOAD A WITH $F1
```

```
         CP     L              ;COMPARE WITH L
         JP     Z,BAK7
         JP     NC,BAK5
BAK7:    LD     BC,15
         ADD    HL,BC          ;ADD 15 IF CARRY FLAG SET
BAK5:    LD     (PROD3),HL
         JP     0000H          ; JUMP TO MONITOR
;
; **********************************************************
; * ROUTINE : MULTIPLICATION
; * PURPOSE : SUBROUTINE FOR UNSIGNED MULTIPLICATION       *
; *           (8 * 8-BIT)                                  *
; **********************************************************
;
MULT:    LD     L,0
         LD     D,0
         LD     B,8
JUMP:    ADD    HL,HL
         JR     NC,NOADD
         ADD    HL,DE
NOADD:   DJNZ   JUMP
         RET                   ;RETURN FROM SUBROUTINE
MPD1:    DEFB   0
MPD2:    DEFB   0
MPR1:    DEFB   0
MPR2:    DEFB   0
PROD1:   DEFB   0
PROD2:   DEFB   0
PROD3:   DEFB   0
PROD4:   DEFB   0
PROD5:   DEFB   0
PROD6:   DEFB   0
TMP1:    DEFB   0
TMP2:    DEFB   0
         END
*
*
*
************************************************************
* MODULAR MULTIPLICATION ROUTINES FOR 6502 MICROPROCESSOR *
************************************************************
*
         NAM    M6502
         ORG    $1024
*
***********************************************************
* ROUTINE : MODULAR ADDITION                              *
* PURPOSE : PERFORM UNSIGNED ADDITION MODULO 65521        *
***********************************************************
*
START    LDX    #AD1          LOAD ADRESS OF OPERAND
         CLC                  CLEAR CARRY FLAG
         LDA    1,X           LOAD ACCUM WITH LOW ORDER BYTE
         ADC    3,X           ADD WITH CARRY LOW ORDER BYTE
         STA    5,X           STORE ACCUM IN LOW ORDER BYTE
         LDA    0,X           LOAD ACCUM WITH HIGH ORDER BYTE
         ADC    2,X           ADD WITH CARRY HIGH ORDER BYTE
         STA    4,X           STORE ACCUM IN HIGH ORDER BYTE
         BCS    OVR
         CMP    #$FF          COMPARE ACCUM WITH $FF
         BNE    SUBT1
         LDA    5,X           LOAD LOW ORDER BYTE OF SUM
         CMP    #$F1          COMPARE WITH $F1
         BEQ    SKIP1
         BMI    SUBT1
OVR      LDA    5,X
SKIP1    CLC
         ADC    #15
         STA    5,X
         LDA    #0
         ADC    4,X
         STA    4,X           RESULT STORED IN SUM:SUM1
SUBT1    JMP    SUBT
*
         ORG    $0023
AD1      FDB    0
AD2      FDB    0
SUM      FCB    0
SUM1     FCB    0
*
***********************************************************
* ROUTINE : MODULAR SUBTRACTION                           *
* PURPOSE : PERFORM UNSIGNED SUBTRACTION MODULO 65521     *
***********************************************************
*
SUBT     BRK                  SET BREAKPOINT
         ORG    $1024
         LDX    #SUB
         LDA    #0
         STA    CHECK
         LDA    0,X
         CMP    2,X
         BEQ    OMIT
         BCS    JMP
         INC    CHECK
JMP      LDA    1,X           LOAD LOW ORDER BYTE
JMP1     SEC                  SET CARRY FLAG
         SBC    3,X           SUBTRACT LOW ORDER BYTE
         STA    5,X           STORE IN SUB1
         LDA    0,X           LOAD ACCUM WITH HIGH ORDER OPERAND
         SBC    2,X           SUBTRACT HIGH ORDER OPERAND
         STA    4,X           STORE ACCUM IN SUB
*
*************************************************************
* IF CHECK=0 THEN SUBTRAHEND < MINUEND                      *
* CHECK NON ZERO WHEN SUBTRAHEND GREATER THAN MINUEND       *
*************************************************************
*
```

```
         LDA    CHECK
         BEQ    MULT1
         CLC                          CLEAR CARRY FLAG
         LDA    5,X
         ADC    #$F1                  ADD $FFF1 (65521) IS SUBTRAHEND
         STA    5,X                   IS GREATER THAN MINUEND
         LDA    4,X
         ADC    #$FF
         STA    4,X
MULT1    JMP    MULT
OMIT     LDA    1,X
         CMP    3,X
         BEQ    OMIT1
         BCS    JMP1
         INC    CHECK
         JMP    JMP1
OMIT1    LDA    #0
         STA    4,X
         STA    5,X
         JMP    MULT
         ORG    $0023
SUB      FDB    0
SUB1     FDB    0
SUB2     FCB    0
SUB3     FDB    0
CHECK    FCB    0
*
**********************************************************
* ROUTINE : MULTIPLICATION                               *
* PURPOSE : PERFORM UNSIGNED MULTIPLICATION MODULO 65521  *
*           (16 * 16-BIT)                                *
**********************************************************
*
MULT     BRK                          SET BREAKPOINT
         ORG    $1024
*
         LDX    #MPLR
         LDA    8,X                   LOAD LS BYTE OF MCND
         STA    2,X                   STORE IN MCND2
         LDA    6,X                   LOAD LS BYTE OF MPLR
         STA    0,X                   STORE IN MPLR
         JSR    SUBRT                 JUMP TO SUBROUTINE
         LDA    4,X                   PARTIAL PROD FROM SUBROUTINE IS
*                                     STORED IN LOCATION 3:4
         STA    16,X                  LOCATION 15:16 CONTAINS L*L
         LDA    3,X
         STA    15,X
         LDA    7,X                   LOAD HIGH ORDER BYTE OF MCND
         STA    2,X                   STORE IN MCND2
         LDA    6,X                   LOAD ACCUM LS BYTE OF MULTIPLIER
         STA    0,X                   STORE IN MPLR
         JSR    SUBRT
         LDA    4,X
         STA    14,X                  13:14 CONTAINS PRODUCT OF L*H
```

```
          LDA    3,X
          STA    13,X
          LDA    8,X            LOAD LS BYTE OF MND
          STA    2,X            STORE IN MCND2
          LDA    5,X            LOAD HIGH ORDER BYTE OF MPLR
          STA    0,X            STORE IN MPLR
          JSR    SUBRT
          LDA    4,X
          STA    12,X           11:12 CONTAIN PRODUCT OF H*L
          LDA    3,X
          STA    11,X
          LDA    7,X            LOAD MS BYTE OF MCND
          STA    2,X            STORE IN MCND2
*
          LDA    5,X            LOAD ACCUM WITH MS BYTE OF MULTIPLIER
          STA    0,X            STORE IN MPLR
          JSR    SUBRT
          LDA    4,X
          STA    10,X           9:10 CONTAIN PRODUCT OF H*H
          LDA    3,X
          STA    9,X
*
***************************************************************
* LOCATION 9:10,11:12,13:14,15:16 NOW CONTAIN FOUR            *
* PARTIAL PRODUCTS, ADDING UP PARTIAL PRODUCTS                *
***************************************************************
*
          CLC                   CLEAR CARRY FLAG
          LDA    14,X
          ADC    12,X
          STA    14,X
          LDA    13,X
          ADC    11,X
          STA    13,X
          LDA    #0
          ADC    9,X
          STA    9,X
          CLC
          LDA    15,X
          ADC    14,X
          STA    15,X
          LDA    13,X
          ADC    10,X
          STA    14,X
          LDA    #0
          ADC    9,X
          STA    13,X
*
***************************************************************
*   LOCATION 13:14:15:16 NOW CONTAIN 32 BIT PRODUCT           *
***************************************************************
*
***************************************************************
* NOW REDUCING THE 32-BIT UNSIGNED PRODUCT MODULO 65521       *
```

```
************************************************************
*
         LDA    15,X
         CMP    #$FF
         BNE    JMPA
         LDA    16,X
         CMP    #$F1
         BEQ    JMPB
         BCC    JMPA
JMPB     CLC
         ADC    #15
         STA    16,X
         LDA    #$0
         ADC    15,X
         STA    15,X
JMPA     LDA    14,X
         STA    2,X           STORE PROD2 IN MCND2
         LDA    #15
         STA    0,X           STORE 15 IN MPLR
         JSR    SUBRT         CALL SUBROUTINE
         CLC
         LDA    16,X
         ADC    4,X
         STA    16,X
         LDA    15,X
         ADC    3,X
         STA    15,X
         BCC    OVRA
         LDA    15,X
         CMP    #$FF
         BNE    OVRA
         LDA    16,X
         CMP    #$F1
         BEQ    JMPC
         BCC    OVRA
JMPC     CLC
         ADC    #15
         STA    16,X
         LDA    #$0
         ADC    15,X
         STA    15,X
OVRA     LDA    #0
         STA    17,X          CLEAR TMP1
         STA    18,X          CLEAR TMP2
         STA    19,X          CLEAR TMP3
         LDA    13,X
         STA    2,X           STORE PROD1 IN MCND2
         LDA    #15
         STA    0,X           STORE 15 IN MPLR
*
         JSR    SUBRT
         LDA    4,X
         STA    18,X
         LDA    3,X
         STA    2,X           STORE MS BYTE OF PARTIAL PROD IN MCND2
         LDA    #15
         STA    0,X
         JSR    SUBRT
         CLC
         LDA    4,X
         ADC    19,X
         STA    19,X
         LDA    3,X
         ADC    18,X
         STA    18,X
         CLC
         LDA    16,X
         ADC    19,X
         STA    16,X
         LDA    15,X
         ADC    18,X
         STA    15,X
         BCS    JUMPC
         CMP    #$FF
         BNE    OVER1
         LDA    16,X
         CMP    #$F1
         BEQ    JUMPC
         BCC    OVER1
JUMPC    CLC
         LDA    16,X
         ADC    #15
         STA    16,X
         LDA    #0
         ADC    15,X
         STA    15,X
OVER1    BRK
*
***********************************************************
* ROUTINE : 8 * 8-BIT MULTIPLICATION                      *
* PURPOSE : PERFORM UNSIGNED MULTIPLICATION USING SHIFT   *
*           AND ADD ALGORITHM                             *
***********************************************************
*
SUBRT    LDA    #0
         STA    1,X           CLEAR MCND1
         STA    3,X           CLEAR TEMP1
         STA    4,X           CLEAR TEMP2
         LDY    #8
         JMP    BAK
OVER     ASL    1,X
         ASL    2,X           SHIFT LEFT MCND2
         BCC    BAK
         LDA    #1
         ORA    1,X
         STA    1,X
BAK      CLC
         ROR    0,X
         BCC    BAK1
         CLC
         LDA    2,X           LOAD ACCUM WITH MCND2
         ADC    4,X           ADD TEMP2
         STA    4,X           STORE IN TEMP2
         LDA    1,X           LOAD ACCUM WITH MCND1
         ADC    3,X           ADD TEMP1
         STA    3,X           STORE IN TEMP1
BAK1     DEY
         BEQ    OUT
         JMP    OVER
OUT      RTS
         ORG    $0023         BYTE NUMBER
MPLR     FCB    0             0
MCND1    FCB    0             1
MCND2    FCB    0             2
TEMP1    FCB    0             3
TEMP2    FCB    0             4
MPR      FDB    0             5:6
MND      FDB    0             7:8
PROD1    FCB    0             9
PROD2    FCB    0             10
PROD3    FCB    0             11
PROD4    FCB    0             12
PROD5    FCB    0             13
PROD6    FCB    0             14
PROD7    FCB    0             15
PROD8    FCB    0             16
TMP1     FCB    0             17
TMP2     FCB    0             18
TMP3     FCB    0             19
         END    START
*
*
*************************************************************
*    DIVISION ROUTINE FOR THE MC6809 MICROPROCESSOR         *
*************************************************************
*

         NAM   DIVISION
*
*************************************************************
* ROUTINE : MULTIPLICATION / DIVISION ROUTINE               *
* PUTPOSE : FIRST PRODUCING A 32-BIT PRODUCT OF UNSIGNED    *
*           NUMBERS IN MLTN AND MLTR AND THEN DIVIDING      *
*           BY 65521 TO REDUCE 32-BIT UNSIGNED NUMBER INTO  *
*           A 16-BIT UNSIGNED NUMBER                        *
*************************************************************
*
         ORG   $0000
START    LDX   #MLTR
         LDY   #MLTN
         LDU   #PROD1
         CLR    ,U
```

```
          CLR   1,U
          LDA   1,X         GET LS BYTE OF MULTIPLIER
          LDB   1,Y         GET LS BYTE OF MULTIPLICAND
          MUL
          STD   2,U         SAVE PARTIAL PRODUCT IN PROD3:PROD4
          LDA    ,X         GET MS BYTE OF MULTIPLIER
          LDB   1,Y         GET LS BYTE OF MULTIPLICAND
          MUL
          ADDD  1,U         ADD PREVIOUS PARTIAL PRODUCT MAINTAINING
*                           THE SIGNIFICANCE
          STD   1,U         SAVE IN PROD2:PROD3
          BCC   SKIP3
          INC    ,U         INCREMENT PROD1 IF A CARRY IS GENERATED
SKIP3     LDA   1,X         GET LS BYTE OF MULTIPLIER
          LDB    ,Y         GET MS BYTE OF MULTIPLICAND
          MUL
          ADDD  1,U         ADD PROD2:PROD3
          STD   1,U         SAVE IN PROD2:PROD3
          BCC   SKIP4
          INC    ,U         INCREMENT PROD1 IF A CARRY IS GENERATED
SKIP4     LDA    ,X         GET MS BYTE OF MULTIPLIER
          LDB    ,Y         GET MS BYTE OF MULTIPLICAND
          MUL
          ADDD   ,U         ADD PROD1 INTO PARTIAL PRODUCT
          STD    ,U         SAVE IN PROD1
*
*************************************************************
*  PROD1:PROD2:PROD3:PROD4 NOW CONTAIN 32 BIT UNSIGNED      *
*  PRODUCT OF TWO 16-BIT NUMBERS IN MLTN AND MLTR           *
*************************************************************
*
************************************************************
* ROUTINE : DIVISION ROUTINE                               *
* PURPOSE : 32-BIT / 16-BIT UNSIGNED DIVIDE BY 65521       *
*           DIVISION ROUTINE                               *
************************************************************
*
          LDD   PROD1
          STD   DVND2
          LDD   PROD3
          STD   DVND4
          LDD   #0
          STA   DVND1       CLEAR DVND1 BEFORE STARTING THE DIVISION
          STD   QUOT1       QUOTIENT IF REQUIRED
          LDA   #16
          STA   COUNT
DIVIDE    ASL   DVND5       SHIFT DIVIDEND LEFT 1 BIT
          ROL   DVND4
          ROL   DVND3
          ROL   DVND2       (MSB OF DIVIDEND)
          ROL   DVND1
          LDA   DVND1       CHECK IF AFTER SHIFTING DIVIDEND LEFT ONE
          BNE   SKIP        BIT IS STILL ZERO OR NOT
          LDD   DVND2
          CMPD   DVSR2        IS TRIAL SUBTRACTION SUCCESSFUL?
          BCS    CHECK        IF CARRY SET BRANCH TO CHECK
SKIP      LDA    DVND3
          SUBA   DVSR3        YES, SUBTRACT AND SET BIT IN QUOTIENT
          STA    DVND3
          LDA    DVND2
          SBCA   DVSR2
          STA    DVND2
          LDA    DVND1
          SBCA   DVSR1
          STA    DVND1
          ASL    QUOT2
          ROL    QUOT1
          INC    QUOT2        INCREMENT QUOTIENT
CHECK     DEC    COUNT        DECREMENT COUNT
          BNE    DIVIDE
          LDD    DVND2
          STD    REM          STORE REMAINDER, QUOTIENT
          JMP    $D283        JUMP TO MONITOR
COUNT     FCB    00
DVSR1     FCB    00
DVSR2     FCB    00
DVSR3     FCB    00
REM       FCB    00
QUOT1     FCB    00
QUOT2     FDB    00
MLTR      FDB    00
MLTN      FCB    00
PROD1     FCB    00
PROD2     FCB    00
PROD3     FCB    00
PROD4     FCB    00
DVND1     FCB    00
DVND2     FCB    00
DVND3     FCB    00
DVND4     FCB    00
DVND5     FCB    00
          END
```

Appendix-B

Assembler program source listing for a 15-point WFTA (TMS9900)

FORTRAN program source listing for a 15-point WFTA

```
*
************************************************************
* 15-POINT WINOGRAD FOURIER TRANSFORM ALGORITHM FOR THE    *
* TMS9900 MICROPROCESSOR USING MULTIPLY/DIVIDE INSTRUCTIONS*
************************************************************
*
         IDT    'WINO15'
         OPTION XREF,SYMT
         AORG   >6000
START    LWPI   WSP
         LI     R4,YREG     INPUT DATA IN YREG ARRAY
         LI     R5,XREG     XREG ARRAY USED FOR INTERMEDIATE
*                           COMPUTATIONS
*
***********************************************************
* PERFORM THE INPUT SHUFFLE AND MOVE DATA FROM YREG INTO  *
* XREG                                                    *
***********************************************************
*
         MOV    *R4,*R5
         MOV    @6(R4),@2(R5)
         MOV    @12(R4),@4(R5)
         MOV    @18(R4),@6(R5)
         MOV    @24(R4),@8(R5)
         MOV    @10(R4),@10(R5)
         MOV    @16(R4),@12(R5)
         MOV    @22(R4),@14(R5)
         MOV    @28(R4),@16(R5)
         MOV    @4(R4),@18(R5)
         MOV    @20(R4),@20(R5)
         MOV    @26(R4),@22(R5)
         MOV    @2(R4),@24(R5)
         MOV    @8(R4),@26(R5)
         MOV    @14(R4),@28(R5)
*
************************************************************
* NOW TO PERFORM THE FIVE 3-POINT PRE-WEAVES IN XREG       *
************************************************************
*
LOOP1    MOV    @10(R5),R0        R5 CONTAINS ADDRESS OF XEG
         MOV    @20(R5),R1
         BL     @ADDSUB
         MOV    R2,@10(R5)
         MOV    R3,@20(R5)
         MOV    *R5,R3
         BL     @ADD
         MOV    R3,*R5
*
         MOV    @12(R5),R0
         MOV    @22(R5),R1
         BL     @ADDSUB
         MOV    R2,@12(R5)
         MOV    R3,@22(R5)
```

```
          MOV    @2(R5),R3
          BL     @ADD
          MOV    R3,@2(R5)
*
          MOV    @14(R5),R0
          MOV    @24(R5),R1
          BL     @ADDSUB
          MOV    R2,@14(R5)
          MOV    R3,@24(R5)
          MOV    @4(R5),R3
          BL     @ADD
          MOV    R3,@4(R5)
*
          MOV    @16(R5),R0
          MOV    @26(R5),R1
          BL     @ADDSUB
          MOV    R2,@16(R5)
          MOV    R3,@26(R5)
          MOV    @6(R5),R3
          BL     @ADD
          MOV    R3,@6(R5)
*
          MOV    @18(R5),R0
          MOV    @28(R5),R1
          BL     @ADDSUB
          MOV    R2,@18(R5)
          MOV    R3,@28(R5)
          MOV    @8(R5),R3
          BL     @ADD
          MOV    R3,@8(R5)
*
*****************************************************************
* NOW PERFORM THREE 5-POINT PRE-WEAVES ON XREG AND MOVE         *

*****************************************************************
*
          LI     R6,ZREG
          MOV    @2(R5),R0
          MOV    @8(R5),R1
          BL     @ADDSUB
          MOV    R2,@2(R5)
          MOV    R3,@6(R6)
*
          MOV    @6(R5),R0
          MOV    @4(R5),R1
          BL     @ADDSUB
          MOV    R2,@4(R5)
          MOV    R3,@10(R6)
          MOV    @6(R6),R2
          BL     @ADD
          MOV    R3,@8(R6)
*
          MOV    @2(R5),R0
          MOV    @4(R5),R1
```

```
       BL    @ADDSUB
       MOV   R2,@2(R6)
       MOV   R3,@4(R6)
       MOV   *R5,R3
       BL    @ADD
       MOV   R3,*R6
*
*
       MOV   @12(R5),R0
       MOV   @18(R5),R1
       BL    @ADDSUB
       MOV   R2,@12(R5)
       MOV   R3,@18(R6)
*
       MOV   @16(R5),R0
       MOV   @14(R5),R1
       BL    @ADDSUB
       MOV   R2,@14(R5)
       MOV   R3,@22(R6)
       MOV   @18(R6),R2
       BL    @ADD
       MOV   R3,@20(R6)
*
       MOV   @12(R5),R0
       MOV   @14(R5),R1
       BL    @ADDSUB
       MOV   R2,@14(R6)
       MOV   R3,@16(R6)
       MOV   @10(R5),R3
       BL    @ADD
       MOV   R3,@12(R6)
*
       MOV   @22(R5),R0
       MOV   @28(R5),R1
       BL    @ADDSUB
       MOV   R2,@22(R5)
       MOV   R3,@30(R6)
*
       MOV   @26(R5),R0
       MOV   @24(R5),R1
       BL    @ADDSUB
       MOV   R2,@24(R5)
       MOV   R3,@34(R6)
       MOV   @30(R6),R2
       BL    @ADD
       MOV   R3,@32(R6)
*
       MOV   @22(R5),R0
       MOV   @24(R5),R1
       BL    @ADDSUB
       MOV   R2,@26(R6)
       MOV   R3,@28(R6)
       MOV   @20(R5),R3
       BL    @ADD
```

```
         MOV   R3,@24(R6)
*
************************************************************
* MULTIPLICATION WITH THE TRANSFORM COEFFICIENTS. THERE    *
* ARE TWO SETS OF MULTIPLIER COEFFICIENTS. THE FORWARD AND *
* AND THE INVERSE TRANSFORM COEFFICIENTS, THE CHOICE OF    *
* THE COEFFICIENTS DEPENDS UPON THE VALUE IN THE VARIABLE  *
* FWD. IF FWD=0 THE MULTIPLICATION ROUTINE USES FORWARD    *
* TRANSFORM COEFFICIENTS OTHERWISE INVERSE COEFFICIENTS    *
* ARE USED.                                                *
*  IF EXTERNAL HARDWARE MODULAR MULTIPLIER IS TO BE USED   *
* THEN REPLACE THE CODE FOR MULTIPLY (MPY) AND DIVIDE (DIV)*
* BY THE EQUIVALENT CODE GIVEN IN FIGURE (5.6)             *
************************************************************
*
         MOV   @FWD,R1
         JEQ   FRWD
         LI    R7,COEFR           LOAD ADDRESS OF INVERSE TRANSFORM
*                                 COEFFICIENTS
         JMP   OVER
FRWD     LI    R7,COEFF           LOAD ADDRESS OF FORWARD TRANSFORM
*                                 COEFFICIENTS
OVER     LI    R4,0               R4 USED AS INDEX FOR ADDRESSING THE
*                                 ZREG AND AS A LOOP COUNTER
         LI    R8,65521           LOAD THE DIVISOR IN R8
LOOP     MOV   *R7+,R1            SEQUENTIAL INDEXING INTO THE ARRAY
         MOV   @ZREG(R4),R2
         MPY   R1,R2
         DIV   R8,R2              R3 CONTAINS THE MODULAR PRODUCT
         MOV   R3,@ZREG(R4)
         INCT  R4
         CI    R4,36
         JNE   LOOP
*
**************************************************
* PERFORM THREE 5-POINT POST-WEAVE FROM ZREG     *
* INTO XREG                                      *
**************************************************
*
         MOV   *R6,R3
         MOV   R3,*R5
         MOV   @2(R6),R2
         BL    @ADD
         MOV   R3,@2(R6)
         MOV   @6(R6),R0
         MOV   @8(R6),R1
         BL    @SUB
         MOV   R3,@6(R6)
         MOV   @8(R6),R2
         MOV   @10(R6),R3
         BL    @ADD
         MOV   R3,@10(R6)
         MOV   @2(R6),R0
         MOV   @4(R6),R1
```

```
            BL      @ADDSUB
            MOV     R2,@2(R6)
            MOV     R3,@4(R6)
            MOV     @2(R6),R0
            MOV     @6(R6),R1
            BL      @ADDSUB
            MOV     R2,@2(R5)
            MOV     R3,@8(R5)
            MOV     @4(R6),R0
            MOV     @10(R6),R1
            BL      @ADDSUB
            MOV     R2,@4(R5)
            MOV     R3,@6(R5)
*
            MOV     @12(R6),R3
            MOV     R3,@10(R5)
            MOV     @14(R6),R2
            BL      @ADD
            MOV     R3,@14(R6)
            MOV     @18(R6),R0
            MOV     @20(R6),R1
            BL      @SUB
            MOV     R3,@18(R6)
            MOV     @20(R6),R2
            MOV     @22(R6),R3
            BL      @ADD
            MOV     R3,@22(R6)
            MOV     @14(R6),R0
            MOV     @16(R6),R1
            BL      @ADDSUB
            MOV     R2,@14(R6)
            MOV     R3,@16(R6)
            MOV     @14(R6),R0
            MOV     @18(R6),R1
            BL      @ADDSUB
            MOV     R2,@12(R5)
            MOV     R3,@18(R5)
            MOV     @16(R6),R0
            MOV     @22(R6),R1
            BL      @ADDSUB
            MOV     R2,@14(R5)
            MOV     R3,@16(R5)
*
            MOV     @24(R6),R3
            MOV     R3,@20(R5)
            MOV     @26(R6),R2
            BL      @ADD
            MOV     R3,@26(R6)
            MOV     @34(R6),R2
            MOV     @32(R6),R3
            BL      @ADD
            MOV     R3,@34(R6)
            MOV     @30(R6),R0
            MOV     @32(R6),R1
```

```
         BL    @SUB
         MOV   R3,@30(R6)
         MOV   @26(R6),R0
         MOV   @28(R6),R1
         BL    @ADDSUB
         MOV   R2,@26(R6)
         MOV   R3,@28(R6)
         MOV   @26(R6),R0
         MOV   @30(R6),R1
         BL    @ADDSUB
         MOV   R2,@22(R5)
         MOV   R3,@28(R5)
         MOV   @28(R6),R0
         MOV   @34(R6),R1
         BL    @ADDSUB
         MOV   R2,@24(R5)
         MOV   R3,@26(R5)
*
***************************************************
* PERFORM FIVE 3-POINT POST-WEAVE IN XREG         *
***************************************************
*
         MOV   *R5,R3
         MOV   @10(R5),R2
         BL    @ADD
         MOV   R3,@10(R5)
*
         MOV   @2(R5),R3
         MOV   @12(R5),R2
         BL    @ADD
         MOV   R3,@12(R5)
*
         MOV   @4(R5),R3
         MOV   @14(R5),R2
         BL    @ADD
         MOV   R3,@14(R5)
*
         MOV   @6(R5),R3
         MOV   @16(R5),R2
         BL    @ADD
         MOV   R3,@16(R5)
*
         MOV   @8(R5),R3
         MOV   @18(R5),R2
         BL    @ADD
         MOV   R3,@18(R5)
*
         MOV   @10(R5),R0
         MOV   @20(R5),R1
         BL    @ADDSUB
         MOV   R2,@10(R5)
         MOV   R3,@20(R5)
*
         MOV   @12(R5),R0
         MOV   @22(R5),R1
         BL    @ADDSUB
         MOV   R2,@12(R5)
         MOV   R3,@22(R5)
*
         MOV   @14(R5),R0
         MOV   @24(R5),R1
         BL    @ADDSUB
         MOV   R2,@14(R5)
         MOV   R3,@24(R5)
*
         MOV   @16(R5),R0
         MOV   @26(R5),R1
         BL    @ADDSUB
         MOV   R2,@16(R5)
         MOV   R3,@26(R5)
*
         MOV   @18(R5),R0
         MOV   @28(R5),R1
         BL    @ADDSUB
         MOV   R2,@18(R5)
         MOV   R3,@28(R5)
*
***************************************************
* PERFORM OUTPUT SHUFFLE ON XREG AND STORE THE    *
* FINAL RESULTS IN ZREG                           *
***************************************************
*
         MOV   *R5,*R6
         MOV   @12(R5),@2(R6)
         MOV   @24(R5),@4(R6)
         MOV   @6(R5),@6(R6)
         MOV   @18(R5),@8(R6)
         MOV   @20(R5),@10(R6)
         MOV   @2(R5),@12(R6)
         MOV   @14(R5),@14(R6)
         MOV   @26(R5),@16(R6)
         MOV   @8(R5),@18(R6)
         MOV   @10(R5),@20(R6)
         MOV   @22(R5),@22(R6)
         MOV   @4(R5),@24(R6)
         MOV   @16(R5),@26(R6)
         MOV   @28(R5),@28(R6)
*
         B     @>0800            BRANCH TO MONITOR
*
*************************************************************
* ROUTINE : ADDSUB                                          *
* PURPOSE : PERFORM ADDITION AND SUBTRACTION                *
*           MODULO 65521                                    *
*           PARAMETERS ARE PASSED TO THE SUBROUTINE         *
*           VIA R0 AND R2 AND THE MODULAR SUM AND           *
*           MODULAR SUBTRACT RESULTS RETURNED VIA           *
*           R2 AND R3 RESPECTIVELY                          *
```

```
************************************************************
*
************************************************************
*            SUBROUTINE ADDSUB                             *
************************************************************
*
ADDSUB  MOV   R1,R2        SAVE CONTENTS OF R1
        A     R0,R2        R2=R2+R0
        JOC   PLUS         ADD 15 IF A CARRY IS GENERATED
        CI    R2,65521     OTHERWISE COMPARE WITH 65521
        JL    SUB
PLUS    AI    R2,15        ADD 15 IF SUM > 65521
SUB     MOV   R0,R3
        S     R1,R3        R3=R3-R1
        C     R1,R0        COMPARE IF SUBTRAHEND > MINUEND
        JL    FIN
        AI    R3,65521     ADD 65521 IF SUBTRAHEND > MINUEND
FIN     RT                 RETURN FROM SUBROUTINE
*
***********************************************************
* ROUTINE : ADD                                           *
* PURPOSE : PERFORM ADDITION MODULO 65521                 *
*           THE PARAMETERS ARE PASSED TO THE SUBROUTINE   *
*           VIA R2 AND R3 AND THE MODULAR SUM IS          *
*           RETURNED IN R3                                *
***********************************************************
*
ADD     A     R2,R3        R3=R2+R3
        JOC   PLUS1        ADD 15 IF A CARRY IS GENERATED
        CI    R3,65521     COMPARE WITH 65521
        JL    TAG
PLUS1   AI    R3,15        ADD 15 IF SUM > 65521
TAG     RT                 RETURN FROM SUBROUTINE
*
*
***********************************************************
*      FORWARD TRANSFORM COEFFICIENTS                     *
***********************************************************
*
COEFF    DATA       1,16379,13376
         DATA   19136,18005,48647
         DATA   32759,8192,45457
         DATA   36817,5753,25311
         DATA   16087,29032,8748
         DATA   23174,43615,1465
*
***********************************************************
*      INVERSE TRANSFORM COEFFICIENTS                     *
***********************************************************
*
COEFR    DATA   61153,5460,18364
         DATA   46773,20640,5493
         DATA    6552,57331,37975
         DATA   28122,34561,24521
         DATA   29504,28641,12521
         DATA    5913,24748,21938
*
WSP      BSS   32          WORKSPACE AREA
YREG     BSS   30
XREG     BSS   30
ZREG     BSS   36
LIM      BSS   2
FWD      BSS   2
LAST     END   START
C
C
C
C
C
C
C   **************************************************************
C   *    PROGRAM FOR 15-POINT WINOGRAD FOURIER TRANSFORM          *
C   *    ALGORITHM (WFTA)                                         *
C   **************************************************************
C
C
      IMPLICIT REAL*8(A - H,O - Z)
      DIMENSION X(15), Y(15), COEF(18), Z(18), OUT(15), COEFR(18)
      INTEGER IRF(15), IRFI(15)
      REAL*8 MODO
C
C    INPUT SHUFFLE VECTORS CALCULATED USING CHINESE REMAINDER
C    THEOREM (CRT). SEE SECTION 4.2.1.
C
      DATA IRF /0, 3, 6, 9, 12, 5, 8, 11, 14, 2, 10, 13, 1, 4, 7/
      FRD = 0.0
C
C    OUTPUT SHUFFLE VECTORS CALCULATED USING CRT
C
      DATA IRFI /0, 6, 12, 3, 9, 10, 1, 7, 13, 4, 5, 11, 2, 8, 14/
C
C    FORWARD TRANSFORM COEFFICIENTS
C
      DATA COEF /1.D0, 16379.D0, 13376.D0, 19136.D0, 18005.D0,
     1     48647.D0, 32759.D0, 8192.D0, 45457.D0, 36817.D0, 5753.D0,
     2     25311.D0, 16087.D0, 29032.D0, 8748.D0, 23174.D0, 43615.D0,
     3     1465.D0/
C
C    INVERSE TRANSFORM COEFFICIENTS
C
      DATA COEFR /61153.D0, 5460.D0, 18364.D0, 46773.D0, 20640.D0,
     1     5493.D0, 6552.D0, 57331.D0, 37975.D0, 28122.D0, 34561.D0,
     2     24521.D0, 29504.D0, 28641.D0, 12521.D0, 5913.D0, 24748.D0,
     3     21938.D0/
C
C     READ INPUT DATA ARRAY IN ARRAY Y. PERFORM INPUT
C     SHUFFLE AND REARRANGE DATA INTO X ARRAY
C
```

```
C     READ VALUES IN FREE FORMAT
C
      READ (5,*) (Y(I),I=1,15)
C
C     CALL TIME ROUTINE TO MEASURE CPU TIME USED
C
      CALL TIME(0, -1)
C
C     PERFORM INPUT SHUFFLE FROM ARRAY Y INTO ARRAY X
C
      DO 10 I = 1, 15
   10 X(I) = Y(IRF(I) + 1)
C
C     PERFORM FIVE 3-POINT PRE-WEAVE
C
      DO 20 I = 1, 5
        T = MODO(X(5 + I) + X(10 + I))
        X(I) = MODO(X(I) + T)
        X(10 + I) = MODO(X(5 + I) - X(10 + I))
        X(5 + I) = T
   20 CONTINUE
C
C     PERFORM THREE 5-POINT PRE-WEAVE
C     MOVE THE DATA INTO ARRAY Z
C
      J = 1
      DO 30 I = 1, 3
        IND = 5 * (I - 1)
        S1 = MODO(X(IND + 2) + X(IND + 5))
        S2 = MODO(X(IND + 2) - X(IND + 5))
        S3 = MODO(X(IND + 4) + X(IND + 3))
        S4 = MODO(X(IND + 4) - X(IND + 3))
        S5 = MODO(S1 + S3)
        S6 = MODO(S1 - S3)
        S7 = MODO(S2 + S4)
        S8 = MODO(S5 + X(IND + 1))
        Z(J) = S8
        Z(J + 1) = S5
        Z(J + 2) = S6
        Z(J + 3) = S2
        Z(J + 4) = S7
        Z(J + 5) = S4
        J = J + 6
   30 CONTINUE
C
C    IF FRD = 0 PERFORM MODULAR MULTIPLICATIONS WITH FORWARD
C    TRANSFORM COEFFICIENTS, OTHERWISE PERFORM MODULAR
C    MULTIPLICATIONS WITH THE INVERSE TRANSFORM COEFFICIENTS
C
      IF (FRD .EQ. 1.D0) GO TO 50
      DO 40 I = 1, 18
   40 Z(I) = MODO(Z(I)*COEF(I))
      GO TO 70
   50 DO 60 I = 1, 18
   60 Z(I) = MODO(Z(I)*COEFR(I))
   70 J = 1
C
C     MODULAR MULTIPLICATIONS COMPLETE.
C     NOW PERFORM THREE 5-POINT POST-WEAVES  AND MOVE THE
C     DATA INTO ARRAY X
C
C     DO 80 I = 1, 3
        IND = 5 * (I - 1)
        S9 = MODO(Z(J) + Z(J + 1))
        S10 = MODO(S9 + Z(J + 2))
        S11 = MODO(S9 - Z(J + 2))
        S12 = MODO(Z(J + 3) - Z(J + 4))
        S13 = MODO(Z(J + 4) + Z(J + 5))
        S14 = MODO(S10 + S12)
        S15 = MODO(S10 - S12)
        S16 = MODO(S11 + S13)
        S17 = MODO(S11 - S13)
        X(IND + 1) = Z(J)
        X(IND + 2) = S14
        X(IND + 3) = S16
        X(IND + 4) = S17
        X(IND + 5) = S15
        J = J + 6
   80 CONTINUE
C
C     PERFORM FIVE 3-POINT POST-WEAVE
C     DATA STILL IN ARRAY X
C
      DO 90 I = 1, 5
        T = MODO(X(I) + X(5 + I))
        T2 = MODO(T + X(10 + I))
        X(10 + I) = MODO(T - X(10 + I))
        X(5 + I) = T2
   90 CONTINUE
C
C     PERFORM OUTPUT SHUFFLE AND MOVE DATA INTO ARRAY OUT
C
      DO 100 I = 1, 15
        OUT(IRFI(I) + 1) = X(I)
  100 CONTINUE
C
C     CALL TIME ROUTINE AND PRINT CPU SEC
C
      CALL TIME(15, -1, CPU)
C
C     WRITE OUT THE RESULTS
C
C     PRINT INPUT ARRAY
C
      WRITE (6,110) (Y(I),I=1,15)
  110 FORMAT (' ', 5F10.2)
      WRITE (6,120)
C
```

```
C     PRINT TRANSFORMED VALUES
C
  120 FORMAT (' ', //)
      WRITE (6,130) (OUT(I),I=1,15)
  130 FORMAT (' ', 5F10.2)
      STOP
      END
C
C     THIS FUNCTION PERFORMS ARITHEMETIC MOD 65521
C
      DOUBLE PRECISION FUNCTION MODO(F)
      REAL*8 F, MOD
      MOD = 65521.D0
      IF (F .LT. 0.0D0) GO TO 10
      MODO = DMOD(F,MOD)
      GO TO 20
   10 MODO = MOD - DMOD(-F,MOD)
   20 RETURN
      END
```

Appendix-C

FORTH program source listing for a 60-point WFTA (TMS9900)

```
  ( TO FIND CURRENT WORKING BASE )      HEX
 : MG1 STRING $ HEX $ SAY ; : MG2 STRING $ DECIMAL $ SAY ;
 : MG3 STRING $ OCTAL $ SAY ; : MG4 STRING $ BINARY $ SAY ;
 : MSG CRLF STRING $ CURRENT BASE IS $ SAY BASE 1 -
       DUP F = IF MG1 DROP ELSE DUP 9 = IF MG2 DROP ELSE 7
       = IF MG3 ELSE MG4 THEN THEN THEN CRLF ;
   HEX  : ERASE 07 SEND 1B SEND 2B SEND .1S ;
   ( FOR PRINTING HEX BYTES )
 : ,, DUP 8 RIGHT DUP 10 < IF ' 0 ' SEND THEN , 08 SEND
      FF AND DUP 10 < IF ' 0 ' SEND THEN , 08 SEND ;
      ( FOR PRINTING ARRAYS )  0 INTEGER CNT
: PRT 0 CNT ! LIMITS DO I @ ,, 2 SPACES CNT @ 1 +
      CNT ! CNT @ 8 = IF CRLF 0 CNT ! THEN 2 +LOOP CRLF ;

:S
  ( THIS PROGRAM PERFORMS WINOGRAD LENGTH 60
    FORWARD AND INVERSE TRANSFORM )
  ( INPUT ARRAY IS Y AND THE RESULT OF TRANSFORM
    IS ALSO STORED IN ARRAY Y      )
  ( COMMANDS FOR PERFORMING FORWARD TRANSFORM
    AND INVERSE TRANSFORMS ARE 'FRD' AND 'REV'
    RESPECTIVELY)
 : MESAG  CRLF CRLF STRING $ INPUT AND OUTPUT ARRAY IS Y $ SAY
          CRLF STRING $ TYPE 'FRD' FOR FORWARD TRANSFORM $ SAY
          CRLF STRING $ & TYPE 'REV' FOR INVERSE TRANSFORM
          $ SAY CRLF ;



:S
  DECIMAL
  0 INTEGER S0   0 INTEGER S1   0 INTEGER S2   0 INTEGER S3
  0 INTEGER S4   0 INTEGER S5   0 INTEGER T1   0 INTEGER T2
  0 INTEGER T3   0 INTEGER T4   0 INTEGER T5   0 INTEGER TM0
  0 INTEGER TM1 0 INTEGER TM2 0 INTEGER TM3 0 INTEGER TM4
  0 INTEGER TM
  144 ARRAY FCOEF 144 ARRAY RCOEF 120 ARRAY X 144 ARRAY Y
  120 ARRAY RF 120 ARRAY RFI
: SINT 0 S0 ! 2 S1 ! 4 S2 ! 6 S3 ! 8 S4 ! 10 S5 ! ;
: INTZ 0 TM0 ! 2 TM ! 4 TM1 ! 6 TM2 ! 8 TM3 ! 10 TM4 ! ;
: 1CHG TM0 @ 10 + TM0 ! TM @ 10 + TM ! TM1 @ 10 + TM1 !
       TM2 @ 10 + TM2 ! TM3 @ 10 + TM3 ! TM4 @ 10 + TM4 ! ;
: 2CHG S0 @ 12 + S0 ! S1 @ 12 + S1 ! S2 @ 12 + S2
       ! S3 @ 12 + S3 ! S4 @ 12 + S4 ! S5 @ 12 + S5 ! ;

:S
   ( INPUT SHUFFLE VECTORS )     RF FILL
   0  72  24  96  48  90  42 114  66  18  60  12  84  36 108
  30 102  54   6  78  80  32 104  56   8  50   2  74  26  98
  20  92  44 116  68 110  62  14  86  38  40 112  64  16  88
  10  82  34 106  58 100  52   4  76  28  70  22  94  46 118
```

```
   ( OUTPUT SHUFFLE VECTORS )   RFI FILL
    0  24  48  72  96  30  54  78 102   6  60  84 108  12  36
   90 114  18  42  66  40  64  88 112  16  70  94 118  22  46
  100   4  28  52  76  10  34  58  82 106  80 104   8  32  56
  110  14  38  62  86  20  44  68  92 116  50  74  98   2  26


:S
 ( COEFFICIENTS FOR FORWARD TRANSFORM )
 FCOEF FILL
      1  16379  13376  64390  46385  48647
      1  16379  13376  64390  46385  48647
      1  16379  13376  64390  46385  48647
  41224  13991  53009  26608  10376  22681
  32759   8192  45457  34457  28704  25311
  32759   8192  45457  34457  28704  25311
  32759   8192  45457  34457  28704  25311
   3685  11774  18768  25609  49957  64260
  49434  36489  56773  45080  23174  64056
  49434  36489  56773  45080  23174  64056
  49434  36489  56773  45080  23174  64056
  33074  56939     32   5797  28796  17202

:S
( COEFFICIENTS FOR INVERSE TRANSFORM )
RCOEF FILL
  64429   1365   4591   9847   4687  50514
  64429   1365   4591   9847   4687  50514
  64429   1365   4591   9847   4687  50514
   3681  11779  30785  35388   4541  64807
   1638  30713  25874  17990  25730  55271
   1638  30713  25874  17990  25730  55271
   1638  30713  25874  17990  25730  55271
  27239  15092  52104  12439  25949   1071
  58145   9220  13250  44432  50619  27276
  58145   9220  13250  44432  50619  27276
  58145   9220  13250  44432  50619  27276
  50784   2041  30577  40308  60673  45578

:S
  ( MODULAR ADDITION )    HEX
CODE MOD 1 POP 2 POP 0 1 0 2 A FNC IF ELSE F 1 AI
        THEN FFF1 1 CI FH IF F 1 AI 1 PUSH ELSE
        1 PUSH THEN RETURN
  ( MODULAR MULTIPLICATION )
CODE D/ 7 POP 5 POP FFF1 4 LI 5 0 7 MPY
       5 0 4 DIV 6 PUSH RETURN
                    ( REG4 CONTAINS DIVISOR )
  ( MODULAR SUBTRACTION )
CODE SBT 2 POP 1 POP 0 3 0 1 MOV 0 1 0 2 S 0 2 0 3 C
     FLT IF FFF1 1 AI 1 PUSH ELSE 1 PUSH THEN RETURN
  ( MODULAR HARDWARE MULTPLIER )
HEX  CODE CREG 0 7 CLR 0 8 CLR 0 9 CLR RETURN
CODE ALOAD 3FF2 2 LI 3FF4 3 LI 3FF6 4 LI RETURN
CODE CALC 0 8 1 2 MOV 0 9 1 3 MOV 1 4 0 7 MOV 7 PUSH RETURN
:S
   ( 3 POINT TRANSFORM INPUT )  DECIMAL
: 3AD 40 0 DO I 40 + X + @ I 80 + X + @ OVER OVER MDD I
      40 + Y + ! SBT I 80 + Y + ! 2 +LOOP ;
: 3DAD 40 0 DO I 40 + Y + @ I X + @ MDD I Y +
      ! 2 +LOOP ;
: I3PT 3AD 3DAD ;

   ( 4 POINT TRANSFORM INPUT )
: 41AD 10 0 DO I Y + @ I 20 + Y + @ MDD I X + ! 2 +LOOP ;
: 42AD 10 0 DO I 10 + Y + @ I 30 + Y + @ OVER OVER MDD
       I 10 + X + ! SBT I 30 + X + ! 2 +LOOP ;
: 42SB 10 0 DO I Y + @ I 20 + Y + @ SBT I 20 + X + ! 2
       +LOOP ;
: 43AD 10 0 DO I X + @ I 10 + X + @ MDD TM ! I X + @ I 10 +
       X + @ SBT I 10 + X + ! TM @ I X + ! 2 +LOOP ;
:S
: 44AD 10 0 DO I 40 + Y + @ I 60 + Y + @ OVER OVER MDD
       I 40 + X + ! SBT I 60 + X + ! 2 +LOOP ;
: 45AD 10 0 DO I 50 + Y + @ I 70 + Y + @ OVER OVER MDD
       I 50 + X + ! SBT I 70 + X + ! 2 +LOOP ;
: 48AD 10 0 DO I 40 + X + @ I 50 + X + @ MDD TM ! I 40 + X +
       @ I 50 + X + @ SBT I 50 + X + ! TM @ I 40 + X + ! 2 +LOOP ;
: 49AD 10 0 DO I 80 + Y + @ I 100 + Y + @ OVER OVER MDD
       I 80 + X + ! SBT I 100 + X + ! 2  +LOOP ;
: 4AAD 10 0 DO I 90 + Y + @ I 110 + Y + @ OVER OVER MDD
       I 90 + X + ! SBT I 110 + X + ! 2 +LOOP ;
: 4DAD 10 0 DO I 80 + X + @ I 90 + X + @ MDD TM ! I 80 + X +
       @ I 90 + X + @ SBT I 90 + X + ! TM @ I 80 + X + ! 2 +LOOP ;
: I4PT 41AD 42AD 42SB 43AD 44AD 45AD 48AD 49AD 4AAD 4DAD ;


:S
  ( MULTIPLICATION WITH COEFFICIENTS )
  0 INTEGER FLAG
: FMULT 144 0 DO I FCOEF + @ I Y + @ D/ I Y + ! 2 +LOOP ;
: RMULT 144 0 DO I RCOEF + @ I Y + @ D/ I Y + ! 2 +LOOP ;
: MULT FLAG @ 0 =  IF FMULT ELSE RMULT THEN ;
  ( 5 POINT TRANSFORM INPUT )
: I15PT TM @ X + @ TM3 @ X + @ OVER OVER MDD S1 @ Y + !
        SBT S5 @ Y + ! ;
: I25PT TM1 @ X + @ TM2 @ X + @ MDD S2 @ Y + ! TM2 @ X
        + @ TM1 @ X + @ SBT S4 @ Y + ! ;
: I35PT S1 @ Y + @ S2 @ Y + @ OVER OVER MDD S1 @ Y + !
        SBT S2 @ Y + ! TM0 @ X + @ S1 @ Y + @ MDD S0 @ Y + ! ;
: I45PT S5 @ Y + @ S4 @ Y + @ MDD S3  @ Y + !  ;
: I5PT INTZ SINT 24 0 DO I15PT I25PT I35PT I45PT 2CHG
       1CHG 2 +LOOP ;
:S
```

```
  ( 5 POINT TRANSFORM OUTPUT )
: FVPT S0 @ Y + @ DUP S1 @ Y + @ MDD T1 ! TM0 @ X + ! ;
: 1FVPT S3 @ Y + @  S5 @ Y + @ MDD T5 ! ;
: 2FVPT S3 @ Y + @  S4 @ Y + @ SBT T3 ! ;
: 3FVPT T1 @ S2 @ Y + @ OVER OVER MDD T2 ! SBT T4 ! ;
: 4FVPT T2 @ T3 @ OVER OVER MDD TM @ X + ! SBT
        TM3 @ X + ! ;
: 5FVPT T4 @ T5 @ OVER OVER MDD TM1 @ X + ! SBT
        TM2 @ X + ! ;
: 05PT INTZ SINT 24 0 DO FVPT 1FVPT 2FVPT 3FVPT
        4FVPT 5FVPT 2CHG 1CHG 2 +LOOP ;




:S
  ( 4 POINT POINT TRANSFORM OUTPUT )
: 401 10 0 DO I X + @ I Y + ! 2 +LOOP ;
: 14D 10 0 DO I 20 + X + @ I 30 + X + @ OVER OVER
      MDD I 10 + Y + ! SBT I 30 + Y + ! I 10 + X + @ I 20
      + Y + ! 2 + LOOP ;
: 402 10  0 DO I 40 + X + @ I 40 + Y + ! 2 +LOOP ;
: 24D 10 0 DO I 60 + X + @ I 70 + X + @ OVER OVER MDD I
      50 + Y + ! SBT I 70 + Y + ! I 50 + X + @ I 60 + Y + ! 2
      + LOOP ;
: 403 10  0 DO I 80 + X + @ I 80 + Y + ! 2 +LOOP ;
: 34D 10  0 DO I 100 + X + @ I 110 + X + @ OVER OVER MDD I
      90 + Y + ! SBT I 110 + Y + ! I 90 + X + @
      I 100 + Y + ! 2 + LOOP ;
: 04PT 401 14D 402 24D 403 34D ;

:S
  ( 3 POINT TRANSFORM OUTPUT )
: 03PT 40 0 DO I Y + @ I 40 + Y + @ MDD
       I 80 + Y + @ OVER OVER MDD I 40 + X + ! SBT I 80
       + X + ! I Y + @ I X + ! 2 +LOOP ;
  ( INPUT RE-ORDERING VECTOR RF )
: IORD 120 0 DO I RF + @ Y + @ I X + ! 2 +LOOP ;
  ( OUTPUT RE-ORDERING VECTOR RFI )
: OORD 120 0 DO I X + @ I RFI + @ Y + ! 2 +LOOP ;
: TRANSFORM     IORD I3PT I4PT I5PT MULT 05PT
                04PT 03PT OORD Y PRT  ;
: REV 1 FLAG ! TRANSFORM ;
: FRD 0 FLAG ! TRANSFORM ;
      MESAG CRLF X EMPTY Y EMPTY  MSG


:S
```

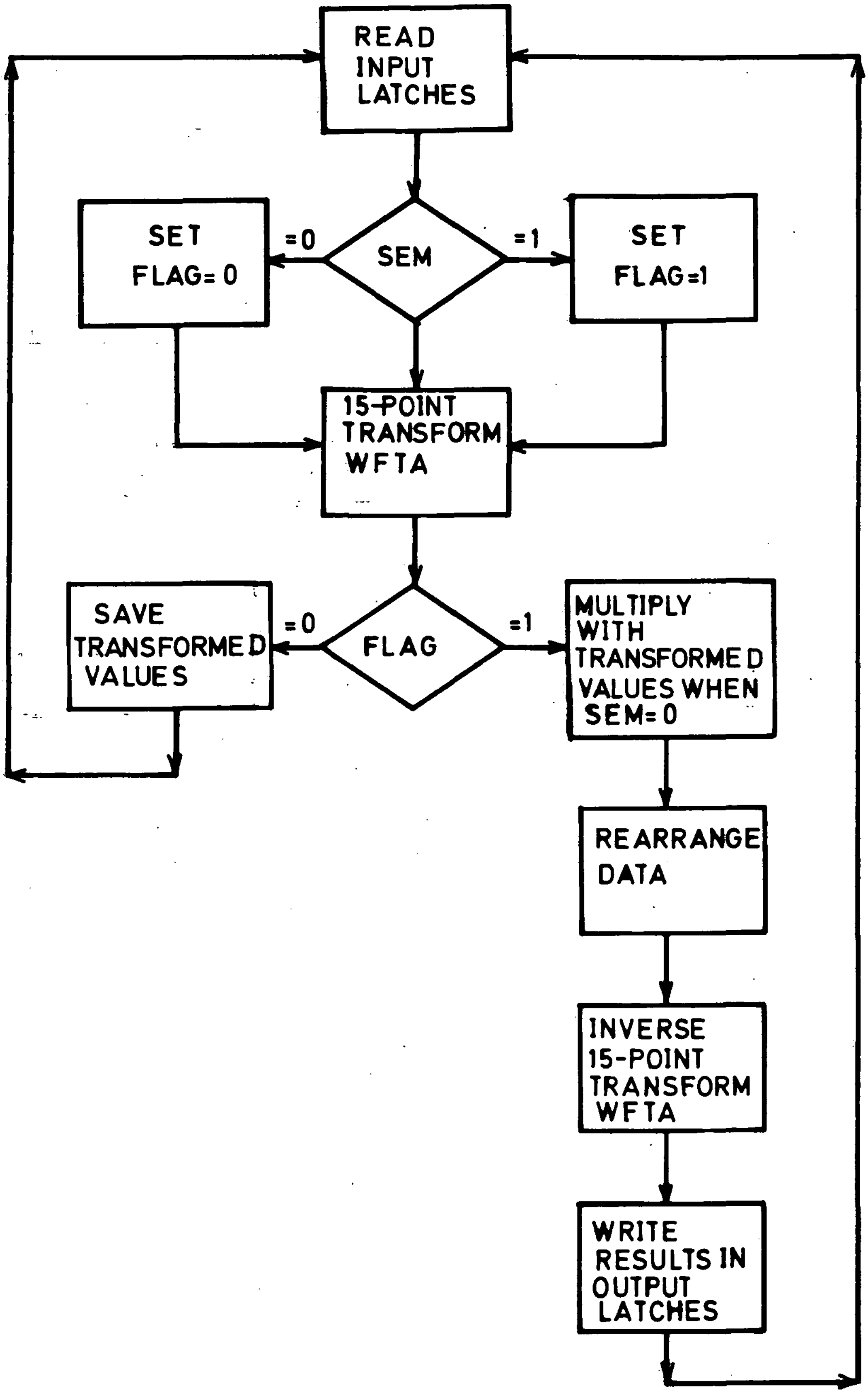
Appendix-D

Assembler program source listings for the slave microprocessors (1 to 18)

Assembler program source listing for the master microprocessor

Assembler program source listing for a 15-point WFTA (MC6809)

FLOW DIAGRAM FOR SLAVES



```
*
         NAM    68091
*
*      ****************************************************************
*      *                  PROCESSOR NUMBER 1                          *
*      ****************************************************************
*
OUTPUT   EQU    $0400         OUTPUT COMMUNICATION LATCH
STATUS   EQU    $0402         STATUS LATCH
T6       EQU    $0403         TRANSMIT DATA TO PROCESSOR NO 6
T2       EQU    $0405         TRANSMIT DATA TO PROCESSOR NO 2
INPUT    EQU    $0410         INPUT COMMUNICATION LATCH
R6       EQU    $0412         RECEIVE DATA FROM PROCESSOR NO 6
R2       EQU    $0414         RECEIVE DATA FROM PROCESSOR NO 2
SEM      EQU    $0416
*
*
         ORG     $F800
         NOP
         ORCC    #%01010000
         LDU     #PROD1
BEGIN    CLRA
         STA     FLAG
         LDA     SEM          SET FLAG=0 IF SEM=0
         BEQ     FRD
START    LDA     #1
         STA     FLAG         SET FLAG=1 IF SEM=1
FRD      LDY     #MCND
         LDX     #MLTFR
         LDA     #1
         STA     STATUS       SET STATUS LATCH=1
         SYNC                 WAIT FOR OTHER PROCESSORS
         CLRA
         STA     STATUS
         LDD     INPUT        READ INPUT LATCH
         BRA     OVER
NEXT     LDY     #MCND
         LDX     #MLTRR
         SYNC                 WAIT FOR OTHER PROCESSORS
         LDD     SAVE
*
OVER     SYNC
         SYNC
         ADDD   R6            ADD DATA RECEIVED FROM 6
         BCS    SKP12         *
         CMPD   #65521        *  MODULAR ADDITION
         BLO    SKP13         *
SKP12    ADDD   #15           *
```

```
SKP13     SYNC                  WAIT FOR OTHER PROCESSORS
          SYNC
          SYNC                  WAIT FOR DATA FROM 2
          ADDD   R2             ADD DATA RECIEVED FROM 2
          BCS    SKP14          *
          CMPD   #65521         * MODULAR ADDITION
          BLO    SKP15          *
SKP14     ADDD   #15            *
*
SKP15     STD    MCND           STORE RESULT IN MCND
          CLR     ,U            *
          CLR    1,U            *
          LDA    1,X            *
          LDB    1,Y            * MODULAR MULTIPLICATION
          MUL                   * WITH TRANSFORM COEFFICIENTS
          STD    2,U            *
          LDA     ,X            *
          LDB    1,Y            *
          MUL                   *
          ADDD   1,U            *
          STD    1,U            *
          BCC    SKP16          *
          INC     ,U            *
SKP16     LDA    1,X            *
          LDB     ,Y            *
          MUL                   *
          ADDD   1,U            *
          STD    1,U            *
          BCC    SKP19          *
          INC     ,U            *
SKP19     LDA     ,X            *
          LDB     ,Y            *
          MUL                   *
          ADDD    ,U            *
          STD     ,U            *
*                               *
          LDA    1,U            *
          LDB    #15            *
          MUL                   *
          ADDD   2,U            *
          BCS    SKP20          *
          CMPD   #65521         *
          BLO    SKP21          *
SKP20     ADDD   #15            *
SKP21     STD    2,U            *
*                               *
          LDA     ,U            *
          LDX    #TEMP          *
          CLR     ,X            *
          CLR    1,X            *
          CLR    2,X            *
          LDB    #15            *
          MUL                   *
          STD     ,X            *
          LDA    ,X            *
          LDB   #15            *
          MUL                  *
          ADDD  1,X            *
          ADDD  2,U            *
          BCS   SKP22          *
          CMPD  #65521         *
          BLO   SKP23          *
SKP22     ADDD  #15            *
SKP23     SYNC                 WAIT FOR OTHER PROCESSORS TO
*                              COMPLETE THE MULTIPLCATION
          STD   T2             SEND DATA TO 2
          SYNC                 WAIT
          SYNC                 WAIT
          SYNC                 WAIT
          STD   T6             SEND DATA TO 6
          SYNC                 WAIT
          SYNC                 WAIT
*
          STD    SAVE          SAVE RESULT
          LDA    FLAG          CHECK FLAG
          CMPA   #1
          BEQ    MULT
          CMPA   #2
          BEQ    CONV
          LDD    SAVE
          STD    RES
          LBRA   BEGIN
CONV      LDD    SAVE
          STD    OUTPUT        WRITE RESULT IN OUTPUT LATCH
          LBRA   BEGIN
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
MULT      INC    FLAG
          LDX    #SAVE         PERFORM MULTIPLICATION OF
          LDY    #RES          THE TWO TRANSFORMED VALUES
          CLR    ,U            *
          CLR   1,U            * MODULAR MULTIPLICATION
          LDA   1,X            *
          LDB   1,Y            *
          MUL                  *
          STD   2,U            *
          LDA    ,X            *
          LDB   1,Y            *
          MUL                  *
          ADDD  1,U            *
          STD   1,U            *
          BCC   LOP16          *
```

```
        INC     ,U          *
LOP16   LDA    1,X          *
        LDB     ,Y          *
        MUL                 *
        ADDD   1,U          *
        STD    1,U          *
        BCC    LOP19        *
        INC     ,U          *
LOP19   LDA     ,X          *
        LDB     ,Y          *
        MUL                 *
        ADDD    ,U          *
        STD     ,U          *
*                           *
        LDA    1,U          *
        LDB    #15          *
        MUL                 *
        ADDD   2,U          *
        BCS    LOP20        *
        CMPD   #65521       *
        BLO    LOP21        *
LOP20   ADDD   #15          *
LOP21   STD    2,U          *
*                           *
        LDA     ,U          *
        LDX    #TEMP        *
        CLR     ,X          *
        CLR    1,X          *
        CLR    2,X          *
        LDB    #15          *
        MUL                 *
        STD     ,X          *
        LDA     ,X          *
        LDB    #15          *
        MUL                 *
        ADDD   1,X          *
        ADDD   2,U          *
        BCS    LOP22        *
        CMPD   #65521       *
        BLO    LOP23        *
LOP22   ADDD   #15          *
LOP23   STD    SAVE         SAVE RESULT
*
**************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE                *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH          *
*  THE EXISTING COMMUNICATION LATCHES                        *
**************************************************************
*
        SYNC
        SYNC
        SYNC
        SYNC
        LBRA    NEXT        PERFORM INVERSE TRANSFORM
```

```
*
MLTFR    FDB    1           FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB    61153       INVERSE TRANSFORM COEFFICIENT
*
         ORG    $0000
MCND     FDB    0
PROD1    FCB    0
PROD2    FCB    0
PROD3    FCB    0
PROD4    FCB    0
TEMP     FCB    0
TEMP1    FCB    0
TEMP3    FCB    0
SAVE     FDB    0
FLAG     FCB    0
RES      FDB    0
*
         ORG    $FFFE
STRT     EQU    $F800
         END    BEGIN
*
         NAM    68092
*
*     ***************************************************************
*     *                  PROCESSOR NUMBER 2                         *
*     ***************************************************************
*
OUTPUT   EQU    $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU    $0402       STATUS LATCH
T7       EQU    $0403       TRANSMIT DATA TO PROCESSOR 7
T5       EQU    $0405       TRANSMIT DATA TO PROCESSOR 5
T3       EQU    $0407       TRANSMIT DATA TO PROCESSOR 3
T1       EQU    $0409       TRANSMIT DATA TO PROCESSOR 1
INPUT    EQU    $0410       INPUT COMMUNICATION LATCH
R7       EQU    $0412       RECEIVE DATA FROM PROCESSOR 7
R5       EQU    $0414       RECEIVE DATA FROM PROCESSOR 5
R3       EQU    $0416       RECEIVE DATA FROM PROCESSOR 3
R1       EQU    $0418       RECEIVE DATA FROM PROCESSOR 1
SEM      EQU    $041A
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG        SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG        SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS      SET STATUS LATCH=1
         SYNC                 WAIT FOR OTHER PROCESSORS
         CLRA
         STA    STATUS        SET STATUS LATCH=0
         LDD    INPUT         READ INPUT LATCH
         BRA    OVER
NEXT     LDY   #MCND
         LDX   #MLTRR
         SYNC                 WAIT FOR OTHER PROCESSORS
         LDD    SAVE
*
OVER     SYNC                 WAIT
         SYNC                 WAIT FOR DATA FROM PROCESSOR 7
         ADDD  R7             ADD DATA FROM PROCESSOR 7
         BCS   SKP12          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP13          *
SKP12    ADDD  #15            *
SKP13    STD   T5             TRANSMIT DATA TO PROCESSOR 5
         SYNC                 WAIT
         ADDD  R5             ADD DATA FROM PROCESSOR 5
         BCS   SKP14          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP15          *
SKP14    ADDD  #15            *
SKP15    STD   T3             TRANSMIT DATA TO PROCESSOR 3
         SYNC                 WAIT FOR DATA FROM PROCESSOR 3
         ADDD  R3             ADD DATA FROM PROCESSOR 3
         BCS   SKP16          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP17          *
SKP16    ADDD  #15            *
SKP17    STD   T1             TRANSMIT DATA TO PROCESSOR 1
         SYNC                 WAIT
         STD   MCND           SAVE RESULT IN MCND
         CLR    ,U            *
         CLR   1,U            *
         LDA   1,X            * MODULAR MULTIPLICATION
         LDB   1,Y            * WITH TRANSFORM COEFFICIENTS
         MUL                  *
         STD   2,U            *
         LDA    ,X            *
         LDB   1,Y            *
         MUL                  *
         ADDD  1,U            *
         STD   1,U            *
         BCC   SKP18          *
         INC    ,U            *
SKP18    LDA   1,X            *
         LDB    ,Y            *
         MUL                  *
         ADDD  1,U            *
         STD   1,U            *
         BCC   SKP21          *
         INC    ,U            *
```

```
SKP21       LDA      ,X         *
            LDB      ,Y         *
            MUL                 *
            ADDD     ,U         *
            STD      ,U         *
*                               *
            LDA     1,U         *
            LDB     #15         *
            MUL                 *
            ADDD    2,U         *
            BCS     SKP22       *
            CMPD    #65521      *
            BLO     SKP23       *
SKP22       ADDD    #15         *
SKP23       STD     2,U         *
*                               *
            LDA      ,U         *
            LDX     #TEMP       *
            CLR      ,X         *
            CLR     1,X         *
            CLR     2,X         *
            LDB     #15         *
            MUL                 *
            STD      ,X         *
            LDA      ,X         *
            LDB     #15         *
            MUL                 *
            ADDD    1,X         *
            ADDD    2,U         *
            BCS     SKP24       *
            CMPD    #65521      *
            BLO     SKP25       *
SKP24       ADDD    #15         *
SKP25       SYNC                WAIT FOR OTHER PROCESSORS TO
*                               COMPLETE MULTIPLICATION
            SYNC                WAIT FOR DATA FROM PROCESSOR 1
            ADDD    R1          ADD DATA FROM PROCESSOR 1
            BCS     SKP26       *
            CMPD    #65521      * MODULAR MULTIPLICATION
            BLO     SKP27       *
SKP26       ADDD    #15         *
SKP27       STD     T3          TRANSMIT DATA TO PROCESSOR 3
            SYNC                WAIT FOR DATA FROM PROCESSOR 3
            ADDD    R3          ADD DATA FROM PROCESSOR 3
            BCS     SKP28       *
            CMPD    #65521      * MODULAR ADDITION
            BLO     SKP29       *
SKP28       ADDD    #15         *
SKP29       STD     T5          TRANSMIT DATA TO PROCESSOR 5
            SYNC                WAIT FOR DATA FROM PROCESSOR 5
            ADDD    R5          ADD DATA FROM PROCESSOR 5
            BCS     SKP30       *
            CMPD    #65521      * MODULAR ADDITION
            BLO     SKP31       *
```

```
SKP30     ADDD   #15          *
SKP31     STD    T7           TRANSMIT DATA TO PROCESSOR 7
          SYNC
          SYNC
*
          STD    SAVE         SAVE RESULT
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
          LDA    FLAG
          CMPA   #1
          BEQ    MULT
          CMPA   #2
          BEQ    CONV
          LDD    SAVE
          STD    RES
          LBRA   BEGIN
CONV      LDD    SAVE
          STD    OUTPUT       WRITE RESULT IN OUTPUT LATCH
          LBRA   BEGIN
*
MULT      INC    FLAG
          LDX    #SAVE        PERFORM MULTIPLICATION OF THE TWO
          LDY    #RES         TRANSFORMED VALUES
LOP15     CLR    ,U           *
          CLR   1,U           *  MODULAR MULTIPLICATION
          LDA   1,X           *
          LDB   1,Y           *
          MUL                 *
          STD   2,U           *
          LDA    ,X           *
          LDB   1,Y           *
          MUL                 *
          ADDD  1,U           *
          STD   1,U           *
          BCC   LOP16         *
          INC    ,U           *
LOP16     LDA   1,X           *
          LDB    ,Y           *
          MUL                 *
          ADDD  1,U           *
          STD   1,U           *
          BCC   LOP19         *
          INC    ,U           *
LOP19     LDA    ,X           *
          LDB    ,Y           *
          MUL                 *
          ADDD   ,U           *
          STD    ,U           *
```

```
*                           *
         LDA    1,U         *
         LDB    #15         *
         MUL                *
         ADDD   2,U         *
         BCS    LOP20       *
         CMPD   #65521      *
         BLO    LOP21       *
LOP20    ADDD   #15         *
LOP21    STD    2,U         *
*                           *
         LDA     ,U         *
         LDX    #TEMP       *
         CLR     ,X         *
         CLR    1,X         *
         CLR    2,X         *
         LDB    #15         *
         MUL                *
         STD     ,X         *
         LDA     ,X         *
         LDB    #15         *
         MUL                *
         ADDD   1,X         *
         ADDD   2,U         *
         BCS    LOP22       *
         CMPD   #65521      *
         BLO    LOP23       *
LOP22    ADDD   #15         *
*
*************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                       *
*************************************************************
*
LOP23    STD    T3          TRANSMIT DATA TO PROCESSOR 3
         SYNC               WAIT FOR DATA FROM PROCESSOR 3
         LDD    R3          READ DATA AND
         STD    T5          TRANSMIT TO PROCESSOR 5
         SYNC               WAIT
         SYNC               WAIT
         SYNC               WAIT
         LDD    R3          RECEIVE DATA FROM PROCESSOR 3
         STD    SAVE        SAVE DATA IN SAVE
         LBRA   NEXT
*
MLTFR    FDB    16379       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB     5460       INVERSE TRANSFORM COEFFICIENT
*
         ORG    $0000
MCND     FDB    0
PROD1    FCB    0
PROD2    FCB    0
PROD3    FCB    0
```

```
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68093
*
*     ***************************************************************
*     *                 PROCESSOR NUMBER 3                          *
*     ***************************************************************
*
OUTPUT   EQU   $0400       OUTPUT COMMUMNICATION LATCH
STATUS   EQU   $0402       STATUS LATCH
T8       EQU   $0403       TRANSMIT DATA TO PROCESSOR 8
T4       EQU   $0405       TRANSMIT DATA TO PROCESSOR 4
T2       EQU   $0407       TRANSMIT DATA TO PROCESSOR 2
INPUT    EQU   $0410       INPUT COMMUNICATION LATCH
R8       EQU   $0412       RECEIVE DATA FROM PROCESSOR 8
R4       EQU   $0414       RECEIVE DATA FROM PROCESSOR 4
R2       EQU   $0416       RECEIVE DATA FROM PROCESSOR 2
SEM      EQU   $0418
*
         ORG   $F800
         NOP
         ORCC  #%01010000
         LDU   #PROD1
BEGIN    CLRA
         STA   FLAG        SET FLAG=0 IF SEM=0
         LDA   SEM
         BEQ   FRD
START    LDA   #1
         STA   FLAG        SET FLAG=1 IF SEM=1
FRD      LDY   #MCND
         LDX   #MLTFR
         LDA   #1
         STA   STATUS      SET STATUS LATCH=1
         SYNC              WAIT FOR OTHER PROCESSORS
         CLRA
         STA   STATUS      SET STATUS LATCH=0
         LDD   INPUT       READ INPUT LATCH
         BRA   OVER
NEXT     LDY   #MCND
         LDX   #MLTRR
         SYNC              WAIT
         LDD   SAVE
*
OVER     SYNC              WAIT
```

```
         SYNC                 WAIT FOR DATA FROM PROCESSOR 8
         ADDD   R8            ADD DATA FROM PROCESSOR 8
         BCS    SKP12         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP13         *
SKP12    ADDD   #15           *
SKP13    STD    T4            TRANSMIT DATA TO PROCESSOR 4
         SYNC                 WAIT DATA FROM PROCESSOR 4
         ADDD   R4            ADD DATA FROM PROCESSOR 4
         BCS    SKP14         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP15         *
SKP14    ADDD   #15           *
SKP15    STD    T2            TRANSMIT DATA TO PROCESSOR 2
         SYNC                 WAIT FOR DATA FROM PROCESSOR 2
         STD    SAVE          *
         LDD    R2            * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP16         *
         ADDD   #65521        *
SKP16    SYNC                 WAIT
*
         STD    MCND          *
         CLR     ,U           *  MODULAR MULTIPLICATION
         CLR    1,U           *  WITH TRANSFORM COEFFICIENTS
         LDA    1,X           *
         LDB    1,Y           *
         MUL                  *
         STD    2,U           *
         LDA     ,X           *
         LDB    1,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    SKP18         *
         INC     ,U           *
SKP18    LDA    1,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    SKP21         *
         INC     ,U           *
SKP21    LDA     ,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD    ,U           *
         STD     ,U           *
*                             *
         LDA    1,U           *
         LDB    #15           *
         MUL                  *
         ADDD   2,U           *
         BCS    SKP22         *
         CMPD   #65521        *
         BLO    SKP23         *
SKP22    ADDD   #15           *
SKP23    STD    2,U           *
*                             *
         LDA     ,U           *
         LDX    #TEMP         *
         CLR     ,X           *
         CLR    1,X           *
         CLR    2,X           *
         LDB    #15           *
         MUL                  *
         STD     ,X           *
         LDA     ,X           *
         LDB    #15           *
         MUL                  *
         ADDD   1,X           *
         ADDD   2,U           *
         BCS    SKP24         *
         CMPD   #65521        *
         BLO    SKP25         *
SKP24    ADDD   #15           *
SKP25    SYNC                 WAIT FOR OTHER PROCESSORS TO
*                             COMPLETE MULTIPLICATION
         SYNC
         STD    T2            TRANSMIT DATA TO PROCESSOR 2
         SYNC                 WAIT FOR DATA FROM PROCESSOR 2
         STD    SAVE          *
         LDD    R2            * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP26         *
         ADDD   #65521        *
SKP26    STD    T4            TRANSMIT DATA TO PROCESSOR 4
         SYNC                 WAIT FOR DATA FROM PROCESSOR 4
         ADDD   R4            ADD DATA FROM PROCESSOR 4
         BCS    SKP28         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP29         *
SKP28    ADDD   #15           *
SKP29    STD    T8            TRANSMIT DATA TO PROCESSOR 8
         SYNC                 WAIT
         SYNC                 WAIT
*
         STD     SAVE         SAVE RESULT
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA    FLAG
         CMPA   #1
```

```
          BEQ    MULT
          CMPA   #2
          BEQ    CONV
          LDD    SAVE
          STD    RES
          LBRA   BEGIN
CONV      LDD    SAVE
          STD    OUTPUT      WRITE RESULT IN OUTPUT LATCH
          LBRA   BEGIN
*
MULT      INC    FLAG
          LDX    #SAVE       PERFORM MULTIPLICATION OF THE
          LDY    #RES        THE TWO TRANSFORMED SEQUENCES
LOP15     CLR     ,U         *
          CLR    1,U         * MODULAR MULTIPLICATION
          LDA    1,X         *
          LDB    1,Y         *
          MUL                *
          STD    2,U         *
          LDA     ,X         *
          LDB    1,Y         *
          MUL                *
          ADDD   1,U         *
          STD    1,U         *
          BCC    LOP16       *
          INC     ,U         *
LOP16     LDA    1,X         *
          LDB     ,Y         *
          MUL                *
          ADDD   1,U         *
          STD    1,U         *
          BCC    LOP19       *
          INC     ,U         *
LOP19     LDA     ,X         *
          LDB     ,Y         *
          MUL                *
          ADDD    ,U         *
          STD     ,U         *
*                            *
          LDA    1,U         *
          LDB    #15         *
          MUL                *
          ADDD   2,U         *
          BCS    LOP20       *
          CMPD   #65521      *
          BLO    LOP21       *
LOP20     ADDD   #15         *
LOP21     STD    2,U         *
*                            *
          LDA     ,U         *
          LDX    #TEMP       *
          CLR     ,X         *
          CLR    1,X         *
          CLR    2,X         *
```

```
         LDB   #15          *
         MUL                *
         STD    ,X          *
         LDA    ,X          *
         LDB   #15          *
         MUL                *
         ADDD  1,X          *
         ADDD  2,U          *
         BCS   LOP22        *
         CMPD  #65521       *
         BLO   LOP23        *
LOP22    ADDD  #15          *
*
**************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                       *
**************************************************************
*
LOP23    STD   T2           TRANSMIT DATA TO PROCESSOR 2
         SYNC               WAIT FOR DATA FROM PROCESSOR 2
         LDD   R2           RECEICE DATA FROM PROCESSOR 2
         STD   SAVE         SAVE DATA
         SYNC               WAIT
         SYNC               WAIT FOR DATA FROM PROCESSOR 4
         LDD   R4           RECECIVE DATA FROM PROCESSOR 4
         STD   T2           TRANSMIT DATA TO PROCESSOR 2
         SYNC
         LBRA  NEXT
*
MLTFR    FDB   13376        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   18364        INVERSE TRANSFORM COEFFICIENT
*
         ORG    $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68094
*
*     ***************************************************************
*     *                  PROCESSOR NUMBER 4                         *
*         ****************************************************************
*
OUTPUT    EQU     $0400        OUTPUT COMMUNICATION LATCH
STATUS    EQU     $0402        STATUS LATCH
T9        EQU     $0403        TRANSMIT DATA TO PROCESSOR 9
T3        EQU     $0405        TRANSMIT DATA TO PROCESSOR 3
T16       EQU     $0407        TRANSMIT DATA TO PROCESSOR 16
INPUT     EQU     $0410        INPUT COMMUNICATION LATCH
R9        EQU     $0412        RECEIVE DATA FROM PROCESSOR 9
R3        EQU     $0414        RECEIVE DATA FROM PROCESSOR 3
R16       EQU     $0416        RECEIVE DATA FROM PROCESSOR 16
SEM       EQU     $0418
*
          ORG      $F800
          NOP
          ORCC     #%01010000
          LDU      #PROD1
BEGIN     CLRA
          STA      FLAG        SET FLAG=0 IF SEM=0
          LDA      SEM
          BEQ      FRD
START     LDA      #1
          STA      FLAG        SET FLAG=1 IF SEM=1
FRD       LDY      #MCND
          LDX      #MLTFR
          LDA      #1
          STA      STATUS      SET STATUS LATCH=1
          SYNC                 WAIT FOR OTHER PROCESSORS
          CLRA
          STA      STATUS      SET STATUS LATCH=0
          LDD      INPUT       READ INPUT LATCH
          BRA      OVER
NEXT      LDY      #MCND
          LDX      #MLTRR
          SYNC
          LDD      SAVE
*
OVER      SYNC
          SYNC                 WAIT FOR DATA FROM PROCESSOR 9
          ADDD   R9            ADD DATA FROM PROCESSOR 9
          BCS    SKP12         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP13         *
SKP12     ADDD   #15           *
SKP13     STD    T3            TRANSMIT DATA TO PROCESSOR 3
          SYNC                 WAIT FOR DATA FROM PROCESSOR 3
          SUBD   R3            *
          BCC    SKP14         * MODULAR SUBTRACTION
          ADDD   #65521        *
SKP14     STD    T16           TRANSMIT DATA TO PROCESSOR 16
          SYNC
          SYNC
*
          STD    MCND          *
```

```
        CLR   ,U        * MODULAR MULTIPLICATION
        CLR   1,U       * WITH TRANSFORM COEFFICIENTS
        LDA   1,X       *
        LDB   1,Y       *
        MUL             *
        STD   2,U       *
        LDA   ,X        *
        LDB   1,Y       *
        MUL             *
        ADDD  1,U       *
        STD   1,U       *
        BCC   SKP16     *
        INC   ,U        *
SKP16   LDA   1,X       *
        LDB   ,Y        *
        MUL             *
        ADDD  1,U       *
        STD   1,U       *
        BCC   SKP19     *
        INC   ,U        *
SKP19   LDA   ,X        *
        LDB   ,Y        *
        MUL             *
        ADDD  ,U        *
        STD   ,U        *
*                       *
        LDA   1,U       *
        LDB   #15       *
        MUL             *
        ADDD  2,U       *
        BCS   SKP20     *
        CMPD  #65521    *
        BLO   SKP21     *
SKP20   ADDD  #15       *
SKP21   STD   2,U       *
*                       *
        LDA   ,U        *
        LDX   #TEMP     *
        CLR   ,X        *
        CLR   1,X       *
        CLR   2,X       *
        LDB   #15       *
        MUL             *
        STD   ,X        *
        LDA   ,X        *
        LDB   #15       *
        MUL             *
        ADDD  1,X       *
        ADDD  2,U       *
        BCS   SKP22     *
        CMPD  #65521    *
        BLO   SKP23     *
SKP22   ADDD  #15       *
SKP23   SYNC            WAIT FOR OTHER PROCESSORS TO
```

```
*                               COMPLETE MULTIPLICATION
         SYNC                   WAIT FOR DATA FROM PROCESSOR 16
         ADDD   R16             ADD DATA FROM PROCESSOR 16
         BCS    SKP24           *
         CMPD   #65521          * MODULAR ADDITION
         BLO    SKP25           *
SKP24    ADDD   #15             *
SKP25    SYNC
         STD    T3              TRANSMIT DATA TO PROCESSOR 3
         SYNC                   WAIT FOR DATA FROM PROCESSOR 3
         STD    SAVE            *
         LDD    R3              * MODULAR SUBTRACTION
         SUBD   SAVE            *
         BCC    SKP26           *
         ADDD   #65521          *
SKP26    STD    T9              TRANSMIT DATA TO PROCESSOR 9
         SYNC
         SYNC
         STD     SAVE           SAVE RESULT
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT         WRITE RESULT IN OUTPUT LATCH
         LBRA    BEGIN
*
MULT     INC     FLAG
         LDX     #SAVE          PERFORM MULTIPLICATION OF THE
         LDY     #RES           TWO TRANSFORMED SEQUENCES
LOP15    CLR     ,U             *
         CLR    1,U             * MODULAR MULTIPLICATION
         LDA    1,X             *
         LDB    1,Y             *
         MUL                    *
         STD    2,U             *
         LDA     ,X             *
         LDB    1,Y             *
         MUL                    *
         ADDD   1,U             *
         STD    1,U             *
         BCC    LOP16           *
         INC     ,U           *
LOP16    LDA    1,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    LOP19         *
         INC     ,U           *
LOP19    LDA     ,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD    ,U           *
         STD     ,U           *
*                             *
         LDA    1,U           *
         LDB    #15           *
         MUL                  *
         ADDD   2,U           *
         BCS    LOP20         *
         CMPD   #65521        *
         BLO    LOP21         *
LOP20    ADDD   #15           *
LOP21    STD    2,U           *
*                             *
         LDA     ,U           *
         LDX    #TEMP         *
         CLR     ,X           *
         CLR    1,X           *
         CLR    2,X           *
         LDB    #15           *
         MUL                  *
         STD     ,X           *
         LDA     ,X           *
         LDB    #15           *
         MUL                  *
         ADDD   1,X           *
         ADDD   2,U           *
         BCS    LOP22         *
         CMPD   #65521        *
         BLO    LOP23         *
LOP22    ADDD   #15           *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
************************************************************
*
LOP23    SYNC
         SYNC                 WAIT FOR DATA FROM PROCESSOR 3
         STD    T3            ADD DATA FROM PROCESSOR 3
         LDD    R16           RECEIVE DATA FROM PROCESSOR 16
         STD    SAVE          SAVE RESULT
         SYNC
         SYNC
         LBRA  NEXT
*
MLTFR    FDB   48647       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB    5493       INVERSE TRANSFORM COEFFICIENT
*
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68095
*
*     ***************************************************************
*     *                 PROCESSOR NUMBER 5                          *
*     ***************************************************************
*
OUTPUT   EQU   $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402       STATUS LATCH
T10      EQU   $0403       TRANSMIT DATA TO PROCESSOR 10
T2       EQU   $0405       TRANSMIT DATA TO PROCESSOR 2
T16      EQU   $0407       TRANSMIT DATA TO PROCESSOR 16
INPUT    EQU   $0410       INPUT COMMUNICATION LATCH
R10      EQU   $0412       RECEIVE DATA FROM PROCESSOR 10
R2       EQU   $0414       RECEIVE DATA FROM PROCESSOR 2
R16      EQU   $0416       RECEIVE DATA FROM PROCESSOR 16
SEM      EQU   $0418
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG       SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG       SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS     SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS     SET STATUS LATCH=0
         LDD    INPUT      READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
*
OVER     SYNC
         SYNC              WAIT FOR DATA FROM PROCESSOR 10
         ADDD  R10         ADD DATA FROM PROCESSOR 10
         BCS   SKP12       *
         CMPD  #65521      * MODULAR ADDITION
         BLO   SKP13       *
SKP12    ADDD  #15         *
SKP13    STD   T2          TRANSMIT DATA TO PROCESSOR 2
         SYNC              WAIT FOR DATA FROM PROCESSOR 2
         STD   SAVE        *
         LDD   R2          * MODULAR SUBTRACTION
         SUBD  SAVE        *
         BCC   SKP14       *
         ADDD  #65521      *
SKP14    STD   T16         TRANSMIT DATA TO PROCESSOR 16
         SYNC
         SYNC
*
         STD   MCND        *
         CLR    ,U         * MODULAR MULTIPLICATION
         CLR   1,U         * WITH TRANSFORM COEFFICIENTS
         LDA   1,X         *
         LDB   1,Y         *
         MUL               *
         STD   2,U         *
         LDA    ,X         *
         LDB   1,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   SKP16       *
         INC    ,U         *
SKP16    LDA   1,X         *
         LDB    ,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   SKP19       *
         INC    ,U         *
SKP19    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
```

```
          STD    ,U          *
*                            *
          LDA   1,U          *
          LDB   #15          *
          MUL                *
          ADDD  2,U          *
          BCS   SKP20        *
          CMPD  #65521       *
          BLO   SKP21        *
SKP20     ADDD  #15          *
SKP21     STD   2,U          *
*                            *
          LDA    ,U          *
          LDX   #TEMP        *
          CLR    ,X          *
          CLR   1,X          *
          CLR   2,X          *
          LDB   #15          *
          MUL                *
          STD    ,X          *
          LDA    ,X          *
          LDB   #15          *
          MUL                *
          ADDD  1,X          *
          ADDD  2,U          *
          BCS   SKP22        *
          CMPD  #65521       *
          BLO   SKP23        *
SKP22     ADDD  #15          *
SKP23     SYNC               WAIT FOR OTHER PROCESSORS TO
*                            COMPLETE MULTIPLICATION
          SYNC               WAIT FOR DATA FROM PROCESSOR 16
          SUBD  R16          *
          BCC   SKP24        * MODULAR SUBTRACTION
          ADDD  #65521       *
SKP24     SYNC
          STD   T2           TRANSMIT DATA TO PROCESSOR 2
          SYNC               WAIT FOR DATA FROM PROCESSOR 2
          STD   SAVE         TRANSMIT DATA TO PROCESSOR 2
          LDD   R2           *
          SUBD  SAVE         * MODULAR SUBTRACTION
          BCC   SKP26        *
          ADDD  #65521       *
SKP26     STD   T10          TRANSMIT DATA TO PROCESSOR 10
          SYNC
          SYNC
*
          STD    SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA    FLAG
         CMPA   #1
         BEQ    MULT
         CMPA   #2
         BEQ    CONV
         LDD    SAVE
         STD    RES
         LBRA   BEGIN
CONV     LDD    SAVE
         STD    OUTPUT     WRITE RESULT IN OUTPUT LATCH
         LBRA   BEGIN
*
MULT     INC    FLAG
         LDX    #SAVE      PERFORM MULTIPLICATION OF THE
         LDY    #RES       TWO TRANSFORMED SEQUENCES
LOP15    CLR     ,U        *
         CLR    1,U        * MODULAR MULTIPLICATION
         LDA    1,X        *
         LDB    1,Y        *
         MUL               *
         STD    2,U        *
         LDA     ,X        *
         LDB    1,Y        *
         MUL               *
         ADDD   1,U        *
         STD    1,U        *
         BCC    LOP16      *
         INC     ,U        *
LOP16    LDA    1,X        *
         LDB     ,Y        *
         MUL               *
         ADDD   1,U        *
         STD    1,U        *
         BCC    LOP19      *
         INC     ,U        *
LOP19    LDA     ,X        *
         LDB     ,Y        *
         MUL               *
         ADDD    ,U        *
         STD     ,U        *
*                          *
         LDA    1,U        *
         LDB    #15        *
         MUL               *
         ADDD   2,U        *
         BCS    LOP20      *
         CMPD   #65521     *
         BLO    LOP21      *
LOP20    ADDD   #15        *
LOP21    STD    2,U        *
*                          *
         LDA     ,U        *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   LOP22       *
         CMPD  #65521      *
         BLO   LOP23       *
LOP22    ADDD  #15         *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
************************************************************
*
LOP23    STD   T16         TRANSMIT DATA TO PROCESSOR 16
         SYNC
         SYNC              WAIT FOR DATA FROM PROCESSOR 2
         LDD   R2          RECEIVE DATA FROM PROCESSOR 2
         STD   SAVE        SAVE RESULT3
         SYNC
         SYNC
         LBRA  NEXT
*
MLTFR    FDB   19136       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   46773       INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68096
*
```

```
*       **************************************************************
*       *                   PROCESSOR NUMBER 6                       *
*       **************************************************************
*
OUTPUT   EQU    $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU    $0402       STATUS LATCH
T11      EQU    $0403       TRANSMIT DATA TO PROCESSOR 11
T1       EQU    $0405       TRANSMIT DATA TO PROCESSOR 1
T7       EQU    $0407       TRANSMIT DATA TO PROCESSOR 7
INPUT    EQU    $0410       INPUT COMMUNICATION LATCH
R11      EQU    $0412       RECEIVE DATA FROM PROCESSOR 11
R1       EQU    $0414       RECEIVE DATA FROM PROCESSOR 1
R7       EQU    $0416       RECEIVE DATA FROM PROCESSOR 7
SEM      EQU    $0418
*
         ORG     $F800
         NOP
         ORCC    #%01010000
         LDU     #PROD1
BEGIN    CLRA
         STA     FLAG       SET FLAG=0 IF SEM=0
         LDA     SEM
         BEQ     FRD
START    LDA     #1
         STA     FLAG       SET FLAG=1 IF SEM=1
FRD      LDY     #MCND
         LDX     #MLTFR
         LDA     #1
         STA     STATUS     SET STATUS LATCH=1
         SYNC
         CLRA
         STA     STATUS     SET STATUS LATCH=0
         LDD     INPUT
         BRA     OVER
NEXT     LDY     #MCND
         LDX     #MLTRR
         SYNC
         LDD     SAVE
*
OVER     STD    T11         TRANSMIT DATA TO PROCESSOR 11
         SYNC               WAIT FOR DATA FROM PROCESSOR 11
         ADDD   R11         ADD DATA FROM PROCESSOR 11
         BCS    SKP12       *
         CMPD   #65521      * MODULAR ADDITION
         BLO    SKP13       *
SKP12    ADDD   #15         *
SKP13    STD    T1          TRANSMIT DATA TO PROCESSOR 1
         SYNC               WAIT
         SYNC               WAIT
         SYNC               WAIT
         SYNC               WAIT FOR DATA FROM PROCESSOR 7
         ADDD   R7          ADD DATA FROM PROCESSOR 7
         BCS    SKP14       * MODULAR ADDITION
         CMPD   #65521      *
```

```
          BLO     SKP15        *
SKP14     ADDD    #15
*
SKP15     STD     MCND         *
          CLR      ,U          * MODULAR MULTIPLICATION
          CLR     1,U          * WITH TRANSFORM COECCIENTS
          LDA     1,X          *
          LDB     1,Y          *
          MUL                  *
          STD     2,U          *
          LDA      ,X          *
          LDB     1,Y          *
          MUL                  *
          ADDD    1,U          *
          STD     1,U          *
          BCC     SKP16        *
          INC      ,U          *
SKP16     LDA     1,X          *
          LDB      ,Y          *
          MUL                  *
          ADDD    1,U          *
          STD     1,U          *
          BCC     SKP19        *
          INC      ,U          *
SKP19     LDA      ,X          *
          LDB      ,Y          *
          MUL                  *
          ADDD     ,U          *
          STD      ,U          *
*                              *
          LDA     1,U          *
          LDB     #15          *
          MUL                  *
          ADDD    2,U          *
          BCS     SKP20        *
          CMPD    #65521       *
          BLO     SKP21        *
SKP20     ADDD    #15          *
SKP21     STD     2,U          *
*                              *
          LDA      ,U          *
          LDX     #TEMP        *
          CLR      ,X          *
          CLR     1,X          *
          CLR     2,X          *
          LDB     #15          *
          MUL                  *
          STD      ,X          *
          LDA      ,X          *
          LDB     #15          *
          MUL                  *
          ADDD    1,X          *
          ADDD    2,U          *
          BCS     SKP22        *
```

```
         CMPD   #65521        *
         BLO    SKP23         *
SKP22    ADDD   #15           *
SKP23    SYNC                 WAIT FOR OTHER PROCESSORS TO
*                             COMPLETE MULTIPLICATION
         STD    T7            TRANSMIT DATA TO PROCESSOR 7
         SYNC                 WAIT
         SYNC                 WAIT
         SYNC                 WAIT
         SYNC                 WAIT FOR DATA FROM PROCESSOR 1
         ADDD   R1            ADD DATA FROM PROCESSOR 1
         BCS    SKP24         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP25         *
SKP24    ADDD   #15           *
SKP25    STD    T11           TRANSMIT DATA TO PROCESSOR 11
         SYNC                 WAIT FOR DATA FROM PROCESSOR 11
         ADDD   R11           ADD DATA FROM PROCESSOR 11
         BCS    SKP26         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP27         *
SKP26    ADDD   #15           *
*
SKP27    STD     SAVE
*
*  CHECK FLAG
* IF   FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*      AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*      THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT       WRITE DATA IN OUTPUT LATCH
         LBRA    BEGIN
*
MULT     INC     FLAG
         LDX     #SAVE        PERFORM MULTIPLICATION OF THE
         LDY     #RES         TWO TRANSFORMED SEQUENCES
LOP15    CLR     ,U           *
         CLR    1,U           * MODULAR MULTIPLICATION
         LDA    1,X           *
         LDB    1,Y           *
         MUL                  *
         STD    2,U           *
         LDA     ,X           *
         LDB    1,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    LOP16         *
         INC     ,U           *
LOP16    LDA    1,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    LOP19         *
         INC     ,U           *
LOP19    LDA     ,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD    ,U           *
         STD     ,U           *
*                             *
         LDA    1,U           *
         LDB    #15           *
         MUL                  *
         ADDD   2,U           *
         BCS    LOP20         *
         CMPD   #65521        *
         BLO    LOP21         *
LOP20    ADDD   #15           *
LOP21    STD    2,U           *
*                             *
         LDA     ,U           *
         LDX    #TEMP         *
         CLR     ,X           *
         CLR    1,X           *
         CLR    2,X           *
         LDB    #15           *
         MUL                  *
         STD     ,X           *
         LDA     ,X           *
         LDB    #15           *
         MUL                  *
         ADDD   1,X           *
         ADDD   2,U           *
         BCS    LOP22         *
         CMPD   #65521        *
         BLO    LOP23         *
LOP22    ADDD   #15           *
*
*****************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE       *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH *
*  THE EXISTING COMMUNICATION LATCHES               *
*****************************************************
*
LOP23    STD    T11           TRANSMIT DATA TO PROCESSOR 11
```

```
          SYNC                    WAIT FOR DATA FROM PROCESSOR 11
          LDD    R11              RECEIVE DATA FROM PROCESSOR 11
          STD    SAVE             SAVE RESULT
          SYNC
          SYNC
          SYNC
          LBRA   NEXT
*
MCTFR     FDB    32759            FORWARD TRANSFORM COEFFICIENT
MLTRR     FDB     6552            INVERSE TRANSFORM COEFFICIENT
*
          ORG    $0000
MCND      FDB    0
PROD1     FCB    0
PROD2     FCB    0
PROD3     FCB    0
PROD4     FCB    0
TEMP      FCB    0
TEMP1     FCB    0
TEMP3     FCB    0
SAVE      FDB    0
FLAG      FCB    0
RES       FDB    0
*
          ORG    $FFFE
STRT      EQU    $F800
          END    BEGIN
*
          NAM    68097
*
*     ***************************************************************
*     *                  PROCESSOR NUMBER 7                         *
*     ***************************************************************
*
OUTPUT    EQU    $0400            OUTPUT COMMUNICATION LATCH
STATUS    EQU    $0402            STATUS LATCH
T12       EQU    $0403            TRANSMIT DATA TO PROCESSOR 12
T2        EQU    $0405            TRANSMIT DATA TO PROCESSOR 2
T10       EQU    $0407            TRANSMIT DATA TO PROCESSOR 10
T8        EQU    $0409            TRANSMIT DATA TO PROCESSOR 8
T6        EQU    $040B            TRANSMIT DATA TO PROCESSOR 6
INPUT     EQU    $0410            INPUT COMMUNICATION LATCH
R12       EQU    $0412            RECEIVE DATA FROM PROCESSOR 12
R2        EQU    $0414            RECEIVE DATA FROM PROCESSOR 2
R10       EQU    $0416            RECEIVE DATA FROM PROCESSOR 10
R8        EQU    $0418            RECEIVE DATA FROM PROCESSOR 8
R6        EQU    $041A            RECEIVE DATA FROM PROCESSOR 6
SEM       EQU    $041C
*
          ORG     $F800
          NOP
          ORCC    #%01010000
          LDU     #PROD1
BEGIN     CLRA
```

```
         STA    FLAG          SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG          SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS        SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS        SET STATUS LATCH=0
         LDD    INPUT         READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC                 WAIT FOR OTHER PROCESSORS
         LDD    SAVE
*
OVER     STD   T12            TRANSMIT DATA TO PROCESSOR 12
         SYNC                 WAIT FOR DATA FROM PROCESSOR 12
         ADDD  R12            ADD DATA FROM PROCESSOR 12
         BCS   SKP12          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP13          *
SKP12    ADDD  #15            *
SKP13    STD   T2             TRANSMIT DATA TO PROCESSOR 2
         SYNC
         STD   T10            TRANSMIT DATA TO PROCESSOR 10
         SYNC                 WAIT FOR DATA FROM PROCESSOR 10
         ADDD  R10            ADD DATA FROM PROCESSOR 10
         BCS   SKP14          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP15          *
SKP14    ADDD  #15            *
SKP15    STD   T8             TRANSMIT DATA TO PROCESSOR 8
         SYNC                 WAIT FOR DATA FROM PROCESSOR 8
         ADDD  R8             ADD DATA FROM PROCESSOR 8
         BCS   SKP16          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP17          *
SKP16    ADDD  #15            *
SKP17    STD   T6             TRANSMIT DATA TO PROCESSOR 6
         SYNC
*
         STD   MCND           *
         CLR    ,U            * MODULAR MULTIPLICATION
         CLR   1,U            * WITH TRANSFORM COEFFICIENTS
         LDA   1,X            *
         LDB   1,Y            *
         MUL                  *
         STD   2,U            *
         LDA    ,X            *
         LDB   1,Y            *
```

```
          MUL                  *
          ADDD   1,U           *
          STD    1,U           *
          BCC    SKP18         *
          INC     ,U           *
SKP18     LDA    1,X           *
          LDB     ,Y           *
          MUL                  *
          ADDD   1,U           *
          STD    1,U           *
          BCC    SKP21         *
          INC     ,U           *
SKP21     LDA     ,X           *
          LDB     ,Y           *
          MUL                  *
          ADDD    ,U           *
          STD     ,U           *
*                              *
          LDA    1,U           *
          LDB    #15           *
          MUL                  *
          ADDD   2,U           *
          BCS    SKP22         *
          CMPD   #65521        *
          BLO    SKP23         *
SKP22     ADDD   #15           *
SKP23     STD    2,U           *
*                              *
          LDA     ,U           *
          LDX    #TEMP         *
          CLR     ,X           *
          CLR    1,X           *
          CLR    2,X           *
          LDB    #15           *
          MUL                  *
          STD     ,X           *
          LDA     ,X           *
          LDB    #15           *
          MUL                  *
          ADDD   1,X           *
          ADDD   2,U           *
          BCS    SKP24         *
          CMPD   #65521        *
          BLO    SKP25         *
SKP24     ADDD   #15           *
SKP25     SYNC                 WAIT FOR OTHER PROCESSORS TO
*                              COMPLETE MULTIPLICATION
          SYNC                 WAIT FOR DATA FROM PROCESSOR 6
          ADDD   R6            ADD DATA FROM PROCESSOR 6
          BCS    SKP26         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP27         *
SKP26     ADDD   #15           *
SKP27     STD    T8            TRANSMIT DATA FROM PROCESSOR 8
```

```
         SYNC                    WAIT FOR DATA FROM PROCESSOR 8
         ADDD   R8               ADD DATA FROM PROCESSOR 8
         BCS    SKP28            *
         CMPD   #65521           * MODULAR ADDITION
         BLO    SKP29            *
SKP28    ADDD   #15              *
SKP29    STD    T10              TRANSMIT DATA TO PROCESSOR 10
         SYNC                    WAIT FOR DATA FROM PROCESSOR 10
         ADDD   R10              ADD DATA FROM PROCESSOR 10
         BCS    SKP30            *
         CMPD   #65521           * MODULAR ADDITION
         BLO    SKP31            *
SKP30    ADDD   #15              *
SKP31    SYNC                    WAIT FOR DATA FROM PROCESSOR 2
         ADDD   R2               ADD DATA FROM PROCESSOR 2
         BCS    SKP32            *
         CMPD   #65521           * MODULAR ADDITION
         BLO    SKP33            *
SKP32    ADDD   #15              *
SKP33    STD    T12              TRANSMIT DATA TO PROCESSOR 12
         SYNC                    WAIT FOR DATA FROM PROCESSOR 12
         ADDD   R12              ADD DATA FROM PROCESSOR 12
         BCS    SKP34            *
         CMPD   #65521           * MODULAR ADDITION
         BLO    SKP35            *
SKP34    ADDD   #15              *
*
SKP35    STD     SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT          WRITE IN OUTPUT LATCH
         LBRA    BEGIN
*
MULT     INC     FLAG
         LDX     #SAVE           PERFORM MULTIPLICATION OF THE TWO
         LDY     #RES            TRANSFORMED SEQUENCES
LOP15    CLR     ,U              *
         CLR    1,U              * MODULAR MULTIPLICATION
         LDA    1,X              *
         LDB   1,Y         *
         MUL               *
         STD   2,U         *
         LDA    ,X         *
         LDB   1,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP16       *
         INC    ,U         *
LOP16    LDA   1,X         *
         LDB    ,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP19       *
         INC    ,U         *
LOP19    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
         STD    ,U         *
*                          *
         LDA   1,U         *
         LDB   #15         *
         MUL               *
         ADDD  2,U         *
         BCS   LOP20       *
         CMPD  #65521      *
         BLO   LOP21       *
LOP20    ADDD  #15         *
LOP21    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   LOP22       *
         CMPD  #65521      *
         BLO   LOP23       *
LOP22    ADDD  #15         *
*
*************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                               *
*********************************************************************
*
LOP23    STD   T12          TRANSMIT DATA TO PROCESSOR 12
         SYNC               WAIT FOR DATA FROM PROCESSOR 12
         LDD   R12          RECEIVE DATA FROM PROCESSOR 12
         STD   T8           TRANSMIT DATA TO PROCESSOR 8
         SYNC               WAIT FOR DATA FROM PROCESSOR 8
         LDD   R8           RECEIVE DATA FROM PROCESSOR 8
         STD   T12          TRANSMIT DATA TO PROCESSOR 12
         SYNC               WAIT FOR DATA FROM PROCESSOR 12
         LDD   R12          RECEIVE DATA FROM PROCESSOR 12
         STD   SAVE         SAVE RESULT
         SYNC
         LBRA  NEXT
*
MLTFR    FDB    8192        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   57331        INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68098
*
*     ******************************************************************
*     *                     PROCESSOR NUMBER 8                         *
*     ******************************************************************
*
OUTPUT   EQU   $0400        OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402        STATUS LATCH
T13      EQU   $0403        TRANSMIT DATA TO PROCESSOR 13
T3       EQU   $0405        TRANSMIT DATA TO PROCESSOR 3
T9       EQU   $0407        TRANSMIT DATA TO PROCESSOR 9
T7       EQU   $0409        TRANSMIT DATA TO PROCESSOR 7
INPUT    EQU   $0410        INPUT COMMUNICATION LATCH
R13      EQU   $0412        RECEIVE DATA FROM PROCESSOR 13
R3       EQU   $0414        RECEIVE DATA FROM PROCESSOR 3
R9       EQU   $0416        RECEIVE DATA FROM PROCESSOR 9
R7       EQU   $0418        RECEIVE DATA FROM PROCESSOR 7
SEM      EQU   $041A
```

```
*
         ORG     $F800
         NOP
         ORCC    #%01010000
         LDU     #PROD1
BEGIN    CLRA
         STA     FLAG          SET FLAG=0 IF SEM=0
         LDA     SEM
         BEQ     FRD
START    LDA     #1
         STA     FLAG          SET FLAG=1 IF SEM=1
FRD      LDY     #MCND
         LDX     #MLTFR
         LDA     #1
         STA     STATUS        SET STATUS LATCH=1
         SYNC
         CLRA
         STA     STATUS        SET STATUS LATCH=0
         LDD     INPUT         READ INPUT LATCH
         BRA     OVER
NEXT     LDY     #MCND
         LDX     #MLTRR
         SYNC
         LDD     SAVE
*
OVER     STD    T13            TRANSMIT DATA TO PROCESSOR 13
         SYNC                  WAIT FOR DATA FROM PROCESSOR 13
         ADDD   R13            ADD DATA FROM PROCESSOR 13
         BCS    SKP12          *
         CMPD   #65521         * MODULAR ADDITION
         BLO    SKP13          *
SKP12    ADDD   #15            *
SKP13    STD    T3             TRANSMIT DATA TO PROCESSOR 3
         SYNC
         STD    T9             TRANSMIT DATA TO PROCESSOR 9
         SYNC                  WAIT FOR DATA FROM PROCESSOR 9
         ADDD   R9             ADD DATA FROM PROCESSOR 9
         BCS    SKP14          *
         CMPD   #65521         * MODULAR ADDITION
         BLO    SKP15          *
SKP14    ADDD   #15            *
SKP15    STD    T7             TRANSMIT DATA TO PROCESSOR 7
         SYNC                  WAIT FOR DATA FROM PROCESSOR 7
         STD    SAVE           *
         LDD    R7             * MODULAR SUBTRACTION
         SUBD   SAVE           *
         BCC    SKP16          *
         ADDD   #65521         *
SKP16    SYNC
*
         STD    MCND           *
         CLR     ,U            * MODULAR MULTIPLICATION
         CLR    1,U            * WITH TRANSFORM COEFFICIENTS
         LDA    1,X            *
         LDB   1,Y         *
         MUL               *
         STD   2,U         *
         LDA    ,X         *
         LDB   1,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   SKP18       *
         INC    ,U         *
SKP18    LDA   1,X         *
         LDB    ,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   SKP21       *
         INC    ,U         *
SKP21    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
         STD    ,U         *
*                          *
         LDA   1,U         *
         LDB   #15         *
         MUL               *
         ADDD  2,U         *
         BCS   SKP22       *
         CMPD  #65521      *
         BLO   SKP23       *
SKP22    ADDD  #15         *
SKP23    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   SKP24       *
         CMPD  #65521      *
         BLO   SKP25       *
SKP24    ADDD  #15         *
SKP25    SYNC              WAIT FOR OTHER PROCESSORS TO
*                          COMPLETE MULTIPLICATION
         SYNC              WAIT FOR DATA FROM PROCESSOR 7
         STD   T7          TRANSMIT DATA TO PROCESSOR 7
         SYNC                 WAIT FOR DATA FROM PROCESSOR 7
         STD   SAVE           *
         LDD   R7             * MODULAR SUBTRACTION
         SUBD  SAVE           *
         BCC   SKP26          *
         ADDD  #65521         *
SKP26    STD   T9             TRANSMIT DATA TO PROCESSOR 9
         SYNC                 WAIT FOR DATA FROM PROCESSOR 9
         ADDD  R9             ADD DATA FROM PROCESSOR 9
         BCS   SKP28          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP29          *
SKP28    ADDD  #15            *
SKP29    SYNC                 WAIT FOR DATA FROM PROCESSOR 3
         ADDD  R3             ADD DATA FROM PROCESSOR 3
         BCS   SKP30          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP31          *
SKP30    ADDD  #15            *
SKP31    STD   T13            TRANSMIT DATA TO PROCESSOR 13
         SYNC                 WAIT FOR DATA FROM PROCESSOR 13
         ADDD  R13            ADD DATA FROM PROCESSOR 13
         BCS   SKP32          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP33          *
SKP32    ADDD  #15            *
*
SKP33    STD    SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA    FLAG
         CMPA   #1
         BEQ    MULT
         CMPA   #2
         BEQ    CONV
         LDD    SAVE
         STD    RES
         LBRA   BEGIN
CONV     LDD    SAVE
         STD    OUTPUT        WRITE DATA IN OUTPUT LATCH
         LBRA   BEGIN
*
MULT     INC    FLAG
         LDX    #SAVE         PERFORM MULTIPLICATION OF THE TWO
         LDY    #RES          TRANSFORMED SEQUENCES
LDP15    CLR     ,U           *
         CLR   1,U            * MODULAR MULTIPLICATION
         LDA   1,X            *
         LDB   1,Y        *
         MUL              *
         STD   2,U        *
         LDA    ,X        *
         LDB   1,Y        *
         MUL              *
         ADDD  1,U        *
         STD   1,U        *
         BCC   LOP16      *
         INC    ,U        *
LOP16    LDA   1,X        *
         LDB    ,Y        *
         MUL              *
         ADDD  1,U        *
         STD   1,U        *
         BCC   LOP19      *
         INC    ,U        *
LOP19    LDA    ,X        *
         LDB    ,Y        *
         MUL              *
         ADDD   ,U        *
         STD    ,U        *
*                         *
         LDA   1,U        *
         LDB   #15        *
         MUL              *
         ADDD  2,U        *
         BCS   LOP20      *
         CMPD  #65521     *
         BLO   LOP21      *
LOP20    ADDD  #15        *
LOP21    STD   2,U        *
*                         *
         LDA    ,U        *
         LDX   #TEMP      *
         CLR    ,X        *
         CLR   1,X        *
         CLR   2,X        *
         LDB   #15        *
         MUL              *
         STD    ,X        *
         LDA    ,X        *
         LDB   #15        *
         MUL              *
         ADDD  1,X        *
         ADDD  2,U        *
         BCS   LOP22      *
         CMPD  #65521     *
         BLO   LOP23      *
LOP22    ADDD  #15        *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                        *
************************************************************
*
LOP23    STD   T13         TRANSMIT DATA TO PROCESSOR 13
         SYNC              WAIT FOR DATA FROM PROCESSOR 9
         LDD   R9          RECEIVE DATA FROM PROCESSOR 9
         STD   T7          TRANSMIT DATA TO PROCESSOR 7
         LDD   R13         RECEIVE DATA FROM PROCESSOR 13
         STD   T9          TRANSMIT DATA TO PROCESSOR 9
         SYNC              WAIT FOR DATA FROM PROCESSOR 7
         LDD   R7          RECEIVE DATA FROM PROCESSOR 7
         STD   SAVE        SAVE RESULT
         SYNC
         SYNC
         LBRA   NEXT
*
MLTFR    FDB   45457       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   37975       INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   68099
*
*    **************************************************************
*     *                PROCESSOR NUMBER 9                          *
*    **************************************************************
*
OUTPUT   EQU   $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402       STATUS LATCH
T14      EQU   $0403       TRANSMIT DATA TO PROCESSOR 14
T4       EQU   $0405       TRANSMIT DATA TO PROCESSOR 4
T8       EQU   $0407       TRANSMIT DATA TO PROCESSOR 8
T17      EQU   $0409       TRANSMIT DATA TO PROCESSOR 17
INPUT    EQU   $0410       INPUT COMMUNICATION LATCH
R14      EQU   $0412       RECEIVE DATA FROM PROCESSOR 14
R4       EQU   $0414       RECEIVE DATA FROM PROCESSOR 4
R8       EQU   $0416       RECEIVE DATA FROM PROCESSOR 8
R17      EQU   $0418       RECEIVE DATA FROM PROCESSOR 17
SEM      EQU   $041A
```

```
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG       SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG       SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS     SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS     SET STATUS LATCH=0
         LDD    INPUT      READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
*
OVER     STD   T14          TRANSMIT DATA TO PROCESSOR 14
         SYNC               WAIT FOR DATA FROM PROCESSOR 14
         ADDD  R14          ADD DATA FROM PROCESSOR 14
         BCS   SKP12        *
         CMPD  #65521       * MODULAR ADDITION
         BLO   SKP13        *
SKP12    ADDD  #15          *
SKP13    STD   T4           TRANSMIT DATA TO PROCESSOR 4
         SYNC
         STD   T8           TRANSMIT DATA TO PROCESSOR 8
         SYNC               WAIT FOR DATA FROM PROCESSOR 8
         SUBD  R8           *
         BCC   SKP14        * MODULAR SUBTRACTION
         ADDD  #65521       *
SKP14    STD   T17          TRANSMIT DATA TO PROCESSOR 17
         SYNC
         SYNC
*
         STD   MCND         *
         CLR    ,U          * MODULAR MULTIPLICATION
         CLR   1,U          * WITH TRANSFORM COEFFICIENTS
         LDA   1,X          *
         LDB   1,Y          *
         MUL                *
         STD   2,U          *
         LDA    ,X          *
         LDB   1,Y          *
         MUL                *
         ADDD  1,U          *
         STD    1,U        *
         BCC    SKP16      *
         INC     ,U        *
SKP16    LDA    1,X        *
         LDB     ,Y        *
         MUL               *
         ADDD   1,U        *
         STD    1,U        *
         BCC    SKP19      *
         INC     ,U        *
SKP19    LDA     ,X        *
         LDB     ,Y        *
         MUL               *
         ADDD    ,U        *
         STD     ,U        *
*                          *
         LDA    1,U        *
         LDB    #15        *
         MUL               *
         ADDD   2,U        *
         BCS    SKP20      *
         CMPD   #65521     *
         BLO    SKP21      *
SKP20    ADDD   #15        *
SKP21    STD    2,U        *
*                          *
         LDA     ,U        *
         LDX    #TEMP      *
         CLR     ,X        *
         CLR    1,X        *
         CLR    2,X        *
         LDB    #15        *
         MUL               *
         STD     ,X        *
         LDA     ,X        *
         LDB    #15        *
         MUL               *
         ADDD   1,X        *
         ADDD   2,U        *
         BCS    SKP22      *
         CMPD   #65521     *
         BLO    SKP23      *
SKP22    ADDD   #15        *
SKP23    SYNC              WAIT FOR OTHER PROCESSORS TO
*                          COMPLETE MULTIPLICATION
         SYNC              WAIT FOR DATA FROM PROCESSOR 17
         ADDD   R17        ADD DATA FROM PROCESSOR 17
         BCS    SKP24      *
         CMPD   #65521     * MODULAR ADDITION
         BLO    SKP25      *
SKP24    ADDD   #15        *
SKP25    SYNC
         STD    T8         TRANSMIT DATA TO PROCESSOR 8
         SYNC              WAIT FOR DATA FROM PROCESSOR 8
```

```
          STD    SAVE            *
          LDD    R8              * MODULAR SUBTRACTION
          SUBD   SAVE            *
          BCC    SKP26           *
          ADDD   #65521          *
SKP26     SYNC                   WAIT FOR DATA FROM PROCESSOR 4
          ADDD   R4              ADD DATA FROM PROCESSOR 4
          BCS    SKP28           *
          CMPD   #65521          * MODULAR ADDITION
          BLO    SKP29           *
SKP28     ADDD   #15             *
SKP29     STD    T14             TRANSMIT DATA TO PROCESSOR 14
          SYNC                   WAIT FOR DATA FROM PROCESSOR 14
          ADDD   R14             ADD DATA FROM PROCESSOR 14
          BCS    SKP30           *
          CMPD   #65521          * MODULAR ADDITION
          BLO    SKP31           *
SKP30     ADDD   #15             *
*
SKP31     STD     SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
          LDA     FLAG
          CMPA    #1
          BEQ     MULT
          CMPA    #2
          BEQ     CONV
          LDD     SAVE
          STD     RES
          LBRA    BEGIN
CONV      LDD     SAVE
          STD     OUTPUT       WRITE RESULT INTO OUTPUT LATCH
          LBRA    BEGIN
*
MULT      INC     FLAG
          LDX     #SAVE        PERFORM MULTIPLICATION OF THE TWO
          LDY     #RES         TRANSFORMED SEQUENCES
LOP15     CLR     ,U           *
          CLR    1,U           * MODULAR MULTIPLICATION
          LDA    1,X           *
          LDB    1,Y           *
          MUL                  *
          STD    2,U           *
          LDA     ,X           *
          LDB    1,Y           *
          MUL                  *
          ADDD   1,U           *
          STD    1,U           *
          BCC   LOP16         *
          INC    ,U           *
LOP16     LDA   1,X           *
          LDB    ,Y           *
          MUL                 *
          ADDD  1,U           *
          STD   1,U           *
          BCC   LOP19         *
          INC    ,U           *
LOP19     LDA    ,X           *
          LDB    ,Y           *
          MUL                 *
          ADDD   ,U           *
          STD    ,U           *
*                             *
          LDA   1,U           *
          LDB   #15           *
          MUL                 *
          ADDD  2,U           *
          BCS   LOP20         *
          CMPD  #65521        *
          BLO   LOP21         *
LOP20     ADDD  #15           *
LOP21     STD   2,U           *
*                             *
          LDA    ,U           *
          LDX   #TEMP         *
          CLR    ,X           *
          CLR   1,X           *
          CLR   2,X           *
          LDB   #15           *
          MUL                 *
          STD    ,X           *
          LDA    ,X           *
          LDB   #15           *
          MUL                 *
          ADDD  1,X           *
          ADDD  2,U           *
          BCS   LOP22         *
          CMPD  #65521        *
          BLO   LOP23         *
LOP22     ADDD  #15           *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
************************************************************
*
LOP23     STD   T8            TRANSMIT DATA TO PROCESSOR 8
          SYNC
          SYNC                WAIT FOR DATA FROM PROCESSOR 17
          LDD   R17           RECEIVE DATA FROM PROCESSOR 17
          STD   T14           TRANSMIT DATA TO PROCESSOR 14
```

```
         LDD   R8           RECEIVE DATA FROM PROCESSOR 8
         STD   T17          TRANSMIT DATA FROM PROCESSOR 17
         SYNC               WAIT FOR DATA FROM PROCESSOR 14
         LDD   R14          RECEIVE DATA FROM PROCESSOR 14
         STD   SAVE         SAVE RESULT
         SYNC
         LBRA  NEXT
*
MLTFR    FDB   25311        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   24521        INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680910
*
*     ***************************************************************
*     *                 PROCESSOR NUMBER 10                         *
*     ***************************************************************
*
OUTPUT   EQU   $0400        OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402        STATUS LATCH
T15      EQU   $0403        TRANSMIT DATA TO PROCESSOR 15
T5       EQU   $0405        TRANSMIT DATA TO PROCESSOR 5
T7       EQU   $0407        TRANSMIT DATA TO PROCESSOR 7
T17      EQU   $0409        TRANSMIT DATA TO PROCESSOR 17
INPUT    EQU   $0410        INPUT COMMUNICATION LATCH
R15      EQU   $0412        RECEIVE DATA FROM PROCESSOR 15
R5       EQU   $0414        RECEIVE DATA FROM PROCESSOR 5
R7       EQU   $0416        RECEIVE DATA FROM PROCESSOR 7
R17      EQU   $0418        RECEIVE DATA FROM PROCESSOR 17
SEM      EQU   $041A
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG        SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG          SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS        SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS        SET STATUS LATCH=0
         LDD    INPUT         READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
*
OVER     STD   T15            TRANSMIT DATA TO PROCESSOR 15
         SYNC                 WAIT FOR DATA FROM PROCESSOR 15
         ADDD  R15            ADD DATA FROM PROCESSOR 15
         BCS   SKP12          *
         CMPD  #65521         * MODULAR ADDITION
         BLO   SKP13          *
SKP12    ADDD  #15            *
SKP13    STD   T5             TRANSMIT DATA FROM PROCESSOR 5
         SYNC                 WAIT FOR DATA FROM PROCESSOR 7
         STD   T7             TRANSMIT DATA TO PROCESSOR 7
         SYNC                 WAIT FOR DATA FROM PROCESSOR 7
         STD   SAVE           *
         LDD   R7             * MODULAR SUBTRACTION
         SUBD  SAVE           *
         BCC   SKP14          *
         ADDD  #65521         *
SKP14    STD   T17            TRANSMIT DATA TO PROCESSOR 17
         SYNC
         SYNC
*
         STD   MCND
         CLR    ,U            *
         CLR   1,U            * MODULAR MULTIPLICATION
         LDA   1,X            * WITH TRANSFORM COEFFICIENTS
         LDB   1,Y            *
         MUL                  *
         STD   2,U            *
         LDA    ,X            *
         LDB   1,Y            *
         MUL                  *
         ADDD  1,U            *
         STD   1,U            *
         BCC   SKP16          *
         INC    ,U            *
SKP16    LDA   1,X            *
         LDB    ,Y            *
         MUL                  *
```

```
          ADDD    1,U             *
          STD     1,U             *
          BCC     SKP19           *
          INC      ,U             *
SKP19     LDA      ,X             *
          LDB      ,Y             *
          MUL                     *
          ADDD     ,U             *
          STD      ,U             *
*                                 *
          LDA     1,U             *
          LDB     #15             *
          MUL                     *
          ADDD    2,U             *
          BCS     SKP20           *
          CMPD    #65521          *
          BLO     SKP21           *
SKP20     ADDD    #15             *
SKP21     STD     2,U             *
*                                 *
          LDA      ,U             *
          LDX     #TEMP           *
          CLR      ,X             *
          CLR     1,X             *
          CLR     2,X             *
          LDB     #15             *
          MUL                     *
          STD      ,X             *
          LDA      ,X             *
          LDB     #15             *
          MUL                     *
          ADDD    1,X             *
          ADDD    2,U             *
          BCS     SKP22           *
          CMPD    #65521          *
          BLO     SKP23           *
SKP22     ADDD    #15             *
*
SKP23     SYNC
          SYNC                    WAIT FOR DATA FROM PROCESSOR 17
          SUBD    R17             *
          BCC     SKP24           * MODULAR SUBTRACTION
          ADDD    #65521          *
SKP24     SYNC                    WAIT FOR DATA FROM PROCESSOR 7
          STD     T7              TRANSMIT DATA TO PROCESSOR 7
          SYNC                    WAIT FOR DATA FROM PROCESSOR 7
          STD     SAVE            *
          LDD     R7              * MODULAR SUBTRACTION
          SUBD    SAVE            *
          BCC     SKP26           *
          ADDD    #65521          *
SKP26     SYNC                    WAIT FOR DATA FOM PROCESSOR 7
          ADDD    R5              ADD DATA FROM PROCESSOR 5
          BCS     SKP28           *
```

```
         CMPD   #65521      * MODULAR ADDITION
         BLO    SKP29       *
SKP28    ADDD   #15         *
SKP29    STD    T15         TRANSMIT DATA TO PROCESSOR 15
         SYNC               WAIT FOR DATA FROM PROCESSOR 15
         ADDD   R15         ADD DATA FROM PROCESSOR 15
         BCS    SKP30       *
         CMPD   #65521      * MODULAR ADDITION
         BLO    SKP31       *
SKP30    ADDD   #15         *
*
SKP31    STD     SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT
         LBRA    BEGIN
*
MULT     INC     FLAG
         LDX     #SAVE      PERFORM MULTIPLICATION OF THE TWO
         LDY     #RES       TRANSFORMED SEQUENCES
LOP15    CLR      ,U        *
         CLR    1,U         * MODULAR MULTIPLICATION
         LDA    1,X         *
         LDB    1,Y         *
         MUL                *
         STD    2,U         *
         LDA     ,X         *
         LDB    1,Y         *
         MUL                *
         ADDD   1,U         *
         STD    1,U         *
         BCC    LOP16       *
         INC     ,U         *
LOP16    LDA    1,X         *
         LDB     ,Y         *
         MUL                *
         ADDD   1,U         *
         STD    1,U         *
         BCC    LOP19       *
         INC     ,U          *
LOP19    LDA     ,X          *
         LDB     ,Y          *
         MUL                 *
         ADDD    ,U          *
         STD     ,U          *
*                            *
         LDA   1,U           *
         LDB   #15           *
         MUL                 *
         ADDD  2,U           *
         BCS   LOP20         *
         CMPD  #65521        *
         BLO   LOP21         *
LOP20    ADDD  #15           *
LOP21    STD   2,U           *
*                            *
         LDA     ,U          *
         LDX   #TEMP         *
         CLR     ,X          *
         CLR   1,X           *
         CLR   2,X           *
         LDB   #15           *
         MUL                 *
         STD     ,X          *
         LDA     ,X          *
         LDB   #15           *
         MUL                 *
         ADDD  1,X           *
         ADDD  2,U           *
         BCS   LOP22         *
         CMPD  #65521        *
         BLO   LOP23         *
LOP22    ADDD  #15           *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
************************************************************
*
LOP23    STD   T17           TRANSMIT DATA TO PROCESSOR 17
         SYNC
         SYNC
         SYNC
         SYNC
         LDD    R17          RECEIVE DATA FROM PROCESSOR 17
         STD    SAVE
         LBRA   NEXT
*
MLTFR    FDB   36817         FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   28122         INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
```

```
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680911
*
*     **************************************************************
*     *                 PROCESSOR NUMBER 11                        *
*     **************************************************************
*
OUTPUT   EQU   $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402       STATUS LATCH
T6       EQU   $0403       TRANSMIT DATA TO PROCESSOR 6
T12      EQU   $0405       TRANSMIT DATA TO PROCESSOR 12
INPUT    EQU   $0410       INPUT COMMUNICATION LATCH
R6       EQU   $0412       RECEIVE DATA FROM PROCESSOR 6
R12      EQU   $0414       RECEIVE DATA FROM PROCESSOR 12
SEM      EQU   $0416
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG       SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG       SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS     SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS     SET STATUS LATCH=0
         LDD    INPUT      READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
```

```
*
OVER      STD    T6            TRANSMIT DATA TO PROCESSOR 6
          SYNC                 WAIT FOR DATA FROM PROCESSOR 6
          STD    SAVE          *
          LDD    R6            * MODULAR SUBTRACTION
          SUBD   SAVE          *
          BCC    SKP12         *
          ADDD   #65521        *
SKP12     SYNC
          SYNC
          SYNC
          SYNC                 WAIT FOR DATA FROM PROCESSOR 12
          ADDD   R12           ADD DATA FROM PROCESSOR 12
          BCS    SKP14         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP15         *
SKP14     ADDD   #15           *
*
SKP15     STD    MCND
          CLR     ,U           *
          CLR    1,U           * MODULAR MULTIPLICATION
          LDA    1,X           * WITH TRANSFORM COEFFICIENTS
          LDB    1,Y           *
          MUL                  *
          STD    2,U           *
          LDA     ,X           *
          LDB    1,Y           *
          MUL                  *
          ADDD   1,U           *
          STD    1,U           *
          BCC    SKP16         *
          INC     ,U           *
SKP16     LDA    1,X           *
          LDB     ,Y           *
          MUL                  *
          ADDD   1,U           *
          STD    1,U           *
          BCC    SKP19         *
          INC     ,U           *
SKP19     LDA     ,X           *
          LDB     ,Y           *
          MUL                  *
          ADDD    ,U           *
          STD     ,U           *
*                              *
          LDA    1,U           *
          LDB    #15           *
          MUL                  *
          ADDD   2,U           *
          BCS    SKP20         *
          CMPD   #65521        *
          BLO    SKP21         *
SKP20     ADDD   #15           *
SKP21     STD    2,U           *
```

```
*                           *
         LDA     ,U         *
         LDX    #TEMP       *
         CLR     ,X         *
         CLR    1,X         *
         CLR    2,X         *
         LDB    #15         *
         MUL                *
         STD     ,X         *
         LDA     ,X         *
         LDB    #15         *
         MUL                *
         ADDD   1,X         *
         ADDD   2,U         *
         BCS    SKP22       *
         CMPD   #65521      *
         BLO    SKP23       *
SKP22    ADDD   #15         *
SKP23    SYNC
*
         STD    T12         TRANSMIT DATA TO PROCESSOR 12
         SYNC
         SYNC
         SYNC
         SYNC               WAIT FOR DATA FROM PROCESSOR 6
         STD    T6          TRANSMIT DATA TO PROESSOR 6
         SYNC               WAIT
         STD    SAVE        *
         LDD    R6          * MODULAR SUBTRACTION
         SUBD   SAVE        *
         BCC    SKP24       *
         ADDD   #65521      *
*
SKP24    STD     SAVE
*
*  CHECK FLAG
* IF   FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*      AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*      THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT
         LBRA    BEGIN
*
```

```
MULT      INC     FLAG      *
          LDX     #SAVE     PERFORM MULTIPLICATION OF THE TWO
          LDY     #RES      TRANSFORMED SEQUENCES
LOP15     CLR      ,U       *
          CLR     1,U       * MODULAR MULTIPICATION
          LDA     1,X       *
          LDB     1,Y       *
          MUL               *
          STD     2,U       *
          LDA      ,X       *
          LDB     1,Y       *
          MUL               *
          ADDD    1,U       *
          STD     1,U       *
          BCC     LOP16     *
          INC      ,U       *
LOP16     LDA     1,X       *
          LDB      ,Y       *
          MUL               *
          ADDD    1,U       *
          STD     1,U       *
          BCC     LOP19     *
          INC      ,U       *
LOP19     LDA      ,X       *
          LDB      ,Y       *
          MUL               *
          ADDD     ,U       *
          STD      ,U       *
*                           *
          LDA     1,U       *
          LDB     #15       *
          MUL               *
          ADDD    2,U       *
          BCS     LOP20     *
          CMPD    #65521    *
          BLO     LOP21     *
LOP20     ADDD    #15       *
LOP21     STD     2,U       *
*                           *
          LDA      ,U       *
          LDX     #TEMP     *
          CLR      ,X       *
          CLR     1,X       *
          CLR     2,X       *
          LDB     #15       *
          MUL               *
          STD      ,X       *
          LDA      ,X       *
          LDB     #15       *
          MUL               *
          ADDD    1,X       *
          ADDD    2,U       *
          BCS     LOP22     *
          CMPD    #65521    *
```

```
         BLO   LOP23       *
LOP22    ADDD  #15         *
*
*************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                       *
*************************************************************
*
LOP23    STD   T6          TRANSMIT DATA TO PROCESSOR 6
         SYNC              WAIT FOR DATA FROM PROCESSOR 6
         LDD   R6          RECEIVE DATA FROM PROCESSOR 6
         STD   SAVE        SAVE RESULT
         SYNC
         SYNC
         SYNC
         LBRA   NEXT
*
MLTFR    FDB   16087       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   29504       INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680912
*
*      ***************************************************************
*      *                 PROCESSOR NUMBER 12                          *
*      ***************************************************************
*
OUTPUT   EQU   $0400       OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402       STATUS LATCH
T7       EQU   $0403       TRANSMIT DATA TO PROCESSOR 7
T15      EQU   $0405       TRANSMIT DATA TO PROCESSOR 15
T13      EQU   $0407       TRANSMIT DATA TO PROCESSOR 13
T11      EQU   $0409       TRANSMIT DATA TO PROCESSOR 11
INPUT    EQU   $0410       INPUT COMMUNICATION LATCH
R7       EQU   $0412       RECEIVE DATA FROM PROCESSOR 7
R15      EQU   $0414       RECEIVE DATA FROM PROCESSOR 15
R13      EQU   $0416       RECEIVE DATA FROM PROCESSOR 13
```

```
R11       EQU    $0418        RECEIVE DATA FROM PROCESSOR 11
SEM       EQU    $041A
*
          ORG     $F800
          NOP
          ORCC    #%01010000
          LDU     #PROD1
BEGIN     CLRA
          STA     FLAG        SET FLAG=0 IF SEM=0
          LDA     SEM
          BEQ     FRD
START     LDA     #1
          STA     FLAG        SET FLAG=1 IF SEM=1
FRD       LDY     #MCND
          LDX     #MLTFR
          LDA     #1
          STA     STATUS      SET STATUS LATCH=1
          SYNC
          CLRA
          STA     STATUS      SET STATUS LATCH=0
          LDD     INPUT       READ INPUT LATCH
          BRA     OVER
NEXT      LDY     #MCND
          LDX     #MLTRR
          SYNC
          LDD     SAVE
*
OVER      STD    T7           TRANSMIT DATA TO PROCESSOR 7
          SYNC                WAIT FOR DATA FROM PROCESSOR 7
          STD    SAVE         *
          LDD    R7           * MODULAR SUBTRACTION
          SUBD   SAVE         *
          BCC    SKP12        *
          ADDD   #65521       *
SKP12     SYNC
          STD    T15          TANSMIT DATA TO PROCESSOR 15
          SYNC                WAIT FOR DATA FROM PROCESSOR 15
          ADDD   R15          ADD DATA FROM PROCESSOR 15
          BCS    SKP14        *
          CMPD   #65521       *
          BLO    SKP15        *
SKP14     ADDD   #15          *
SKP15     STD    T13          TRANSMIT DATA TO PROCESSOR 13
          SYNC                WAIT FOR DATA FROM PROCESSOR 13
          ADDD   R13          ADD DATA FROM PROCESSOR 13
          BCS    SKP16        *
          CMPD   #65521       * ADD DATA FROM PROCESSOR 13
          BLO    SKP17        *
SKP16     ADDD   #15          *
SKP17     STD    T11          TRANSMIT DATA TO PROCESSOR 11
          SYNC
*
          STD    MCND
          CLR     ,U          *
```

```
         CLR     1,U          * MODULAR MULTIPLICATION
         LDA     1,X          * WITH TRANSFORM COEFFICIENTS
         LDB     1,Y          *
         MUL                  *
         STD     2,U          *
         LDA      ,X          *
         LDB     1,Y          *
         MUL                  *
         ADDD    1,U          *
         STD     1,U          *
         BCC     SKP18        *
         INC      ,U          *
SKP18    LDA     1,X          *
         LDB      ,Y          *
         MUL                  *
         ADDD    1,U          *
         STD     1,U          *
         BCC     SKP21        *
         INC      ,U          *
SKP21    LDA      ,X          *
         LDB      ,Y          *
         MUL                  *
         ADDD     ,U          *
         STD      ,U          *
*                             *
         LDA     1,U          *
         LDB     #15          *
         MUL                  *
         ADDD    2,U          *
         BCS     SKP22        *
         CMPD    #65521       *
         BLO     SKP23        *
SKP22    ADDD    #15          *
SKP23    STD     2,U          *
*                             *
         LDA      ,U          *
         LDX     #TEMP        *
         CLR      ,X          *
         CLR     1,X          *
         CLR     2,X          *
         LDB     #15          *
         MUL                  *
         STD      ,X          *
         LDA      ,X          *
         LDB     #15          *
         MUL                  *
         ADDD    1,X          *
         ADDD    2,U          *
         BCS     SKP24        *
         CMPD    #65521       *
         BLO     SKP25        *
SKP24    ADDD    #15          *
SKP25    SYNC
*
```

```
          SYNC                 WAIT FOR DATA FROM PROCESSOR 11
          ADDD   R11           ADD DATA FROM PROCESSOR 11
          BCS    SKP26         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP27         *
SKP26     ADDD   #15           *
SKP27     STD    T13           TRANSMIT DATA TO PROCESSOR 13
          SYNC                 WAIT FOR DATA FROM PROCESSOR 13
          ADDD   R13           ADD DATA FROM PROCESSOR 13
          BCS    SKP28         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP29         *
SKP28     ADDD   #15           *
SKP29     STD    T15           TRANSMIT DATA TO PROCESSOR 15
          SYNC                 WAIT FOR DATA FROM PROCESSOR 15
          ADDD   R15           ADD DATA FROM PROCESSOR 15
          BCS    SKP30         *
          CMPD   #65521        * MODULAR ADDITION
          BLO    SKP31         *
SKP30     ADDD   #15           *
SKP31     SYNC
          STD    T7            TRANSMIT DATA TO PROCESSOR 7
          SYNC                 WAIT FOR DATA FROM PROCESSOR 7
          STD    SAVE          *
          LDD    R7            * MODULAR SUBTRACTION
          SUBD   SAVE          *
          BCC    SKP32         *
          ADDD   #65521
*
SKP32     STD     SAVE
*
*  CHECK FLAG
* IF   FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*         AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF   FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*        THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
          LDA     FLAG
          CMPA    #1
          BEQ     MULT
          CMPA    #2
          BEQ     CONV
          LDD     SAVE
          STD     RES
          LBRA    BEGIN
CONV      LDD     SAVE
          STD     OUTPUT
          LBRA    BEGIN
*
MULT      INC     FLAG
          LDX     #SAVE        PERFORM MULTIPLICTION OF THE TWO
          LDY     #RES         TRANSFORMED SEQUENCES
LOP15     CLR     ,U           *
```

```
         CLR   1,U         * MODULAR MULTIPLICATION
         LDA   1,X         *
         LDB   1,Y         *
         MUL               *
         STD   2,U         *
         LDA    ,X         *
         LDB   1,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP16       *
         INC    ,U         *
LOP16    LDA   1,X         *
         LDB    ,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP19       *
         INC    ,U         *
LOP19    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
         STD    ,U         *
*                          *
         LDA   1,U         *
         LDB   #15         *
         MUL               *
         ADDD  2,U         *
         BCS   LOP20       *
         CMPD  #65521      *
         BLO   LOP21       *
LOP20    ADDD  #15         *
LOP21    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   LOP22       *
         CMPD  #65521      *
         BLO   LOP23       *
LOP22    ADDD  #15         *
*
************************************************************
```

```
*  RESHUFFLING THE DATA BEFORE PERFORMING THE                 *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH           *
*  THE EXISTING COMMUNICATION LATCHES                         *
*************************************************************
*
LOP23    STD   T7           TRANSMIT DATA TO PROCESSOR 7
         SYNC               WAIT FOR DATA FROM PROCESSOR 7
         LDD   R7           RECEIVE DATA FROM PROCESSOR 7
         STD   T13          TRANSMIT DATA TO PROCESSOR 13
         SYNC               WAIT FOR DATA FROM PROCESSOR 13
         LDD   R13          RECEIVE DATA FROM PROCESSOR 13
         STD   T7           TRANSMIT DATA TO PROCESSOR 7
         SYNC               WAIT FOR DATA FROM PROCESSOR 7
         LDD   R7           RECEIVE DATA FROM PROCESSOR 7
         STD   SAVE         SAVA RESULT
         SYNC
         LBRA   NEXT
*
MLTFR    FDB   29032        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   28641        INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
RES      FDB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680913
*
*     ***************************************************************
*     *                      PROCESSOR NUMBER 13                    *
*     ***************************************************************
*
OUTPUT   EQU   $0400        OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402        STATUS LATCH
T8       EQU   $0403        TRANSMIT DATA TO PROCESSOR 8
T14      EQU   $0405        TRANSMIT DATA TO PROCESSOR 14
T12      EQU   $0407        TRANSMIT DATA TO PROCESSOR 12
INPUT    EQU   $0410        INPUT COMMUNICATION LATCH
R8       EQU   $0412        RECEIVE DATA FROM PROCESSOR 8
R14      EQU   $0414        RECEIVE DATA FROM PROCESSOR 14
R12      EQU   $0416        RECEIVE DATA FROM PROCESSOR 12
SEM      EQU   $0418
```

```
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG         SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG         SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS       SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS       SET STATUS LATCH=0
         LDD    INPUT        READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
*
OVER     STD   T8            TRANSMIT DATA TO PROCESSOR 8
         SYNC                WAIT FOR DATA FROM PROCESSOR 8
         STD   SAVE          *
         LDD   R8            * MODULAR SUBTRACTION
         SUBD  SAVE          *
         BCC   SKP12         *
         ADDD  #65521        *
SKP12    SYNC                WAIT FOR DATA FROM PROCESSOR 14
         STD   T14           TRANSMIT DATA TO PROCESSOR 14
         SYNC                WAIT FOR DATA FROM PROCESSOR 14
         ADDD  R14           ADD DATA FROM PROCESSOR 14
         BCS   SKP14         *
         CMPD  #65521        * MODULAR ADDITION
         BLO   SKP15         *
SKP14    ADDD  #15           *
SKP15    STD   T12           TRANSMIT DATA TO PROCESSOR 12
         SYNC                WAIT FOR DATA FROM PROCESSOR 12
         STD   SAVE          *
         LDD   R12           * MODULAR SUBTRACTION
         SUBD  SAVE          *
         BCC   SKP16         *
         ADDD  #65521        *
SKP16    SYNC
*
         STD   MCND          *
         CLR    ,U           * MODULAR MULTIPLICATION
         CLR   1,U           * WITH TRANSFORM COEFFICIENTS
         LDA   1,X           *
         LDB   1,Y           *
         STD    SAVE          *
         LDD    R12           * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP26         *
         ADDD   #65521        *
SKP26    STD    T14           TRANSMIT DATA TO PROCESSOR 14
         SYNC                 WAIT FOR DATA FROM PROCESSOR 14
         ADDD   R14           ADD DATA FROM PROCESSOR 14
         BCS    SKP28         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP29         *
SKP28    ADDD   #15           *
SKP29    SYNC
         STD    T8            TRANSMIT DATA TO PROCESSOR 8
         SYNC                 WAIT FOR DATA FROM PROCESSOR 8
         STD    SAVE          *
         LDD    R8            * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP30         *
         ADDD   #65521        *
*
SKP30    STD     SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA     FLAG
         CMPA    #1
         BEQ     MULT
         CMPA    #2
         BEQ     CONV
         LDD     SAVE
         STD     RES
         LBRA    BEGIN
CONV     LDD     SAVE
         STD     OUTPUT       WRITE RESULT IN THE OUTPUT LATCH
         LBRA    BEGIN
*
MULT     INC     FLAG
         LDX     #SAVE        PERFORM MULTIPICATION OF THE TWO
         LDY     #RES         TRANSFORMED SEQUENCES
LOP15    CLR     ,U           *
         CLR    1,U           * MODULAR MULTIPLICATION
         LDA    1,X           *
         LDB    1,Y           *
         MUL                  *
         STD    2,U           *
         LDA     ,X           *
         LDB    1,Y           *
         MUL                  *
```

```
          MUL                *
          STD   2,U          *
          LDA    ,X          *
          LDB   1,Y          *
          MUL                *
          ADDD  1,U          *
          STD   1,U          *
          BCC   SKP18        *
          INC    ,U          *
SKP18     LDA   1,X          *
          LDB    ,Y          *
          MUL                *
          ADDD  1,U          *
          STD   1,U          *
          BCC   SKP21        *
          INC    ,U          *
SKP21     LDA    ,X          *
          LDB    ,Y          *
          MUL                *
          ADDD   ,U          *
          STD    ,U          *
*                            *
          LDA   1,U          *
          LDB   #15          *
          MUL                *
          ADDD  2,U          *
          BCS   SKP22        *
          CMPD  #65521       *
          BLO   SKP23        *
SKP22     ADDD  #15          *
SKP23     STD   2,U          *
*                            *
          LDA    ,U          *
          LDX   #TEMP        *
          CLR    ,X          *
          CLR   1,X          *
          CLR   2,X          *
          LDB   #15          *
          MUL                *
          STD    ,X          *
          LDA    ,X          *
          LDB   #15          *
          MUL                *
          ADDD  1,X          *
          ADDD  2,U          *
          BCS   SKP24        *
          CMPD  #65521       *
          BLO   SKP25        *
SKP24     ADDD  #15          *
SKP25     SYNC
*
          SYNC
          STD   T12          TRANSMIT DATA TO PROCESSOR 12
          SYNC               WAIT FOR DATA FROM PROCESSOR 12
          ADDD  1,U          *
          STD   1,U          *
          BCC   LOP16        *
          INC    ,U          *
LOP16     LDA   1,X          *
          LDB    ,Y          *
          MUL                *
          ADDD  1,U          *
          STD   1,U          *
          BCC   LOP19        *
          INC    ,U          *
LOP19     LDA    ,X          *
          LDB    ,Y          *
          MUL                *
          ADDD   ,U          *
          STD    ,U          *
*                            *
          LDA   1,U          *
          LDB   #15          *
          MUL                *
          ADDD  2,U          *
          BCS   LOP20        *
          CMPD  #65521       *
          BLO   LOP21        *
LOP20     ADDD  #15          *
LOP21     STD   2,U          *
*                            *
          LDA    ,U          *
          LDX   #TEMP        *
          CLR    ,X          *
          CLR   1,X          *
          CLR   2,X          *
          LDB   #15          *
          MUL                *
          STD    ,X          *
          LDA    ,X          *
          LDB   #15          *
          MUL                *
          ADDD  1,X          *
          ADDD  2,U          *
          BCS   LOP22        *
          CMPD  #65521       *
          BLO   LOP23        *
LOP22     ADDD  #15          *
*
*************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                       *
*************************************************************
*
LOP23     STD   T8          TRANSMIT DATA TO PROCESSOR 8
          SYNC              WAIT FOR DATA FROM PROCESSOR 8
          LDD   R8          RECEIVE DATA FROM PROCESSOR 8
```

```
         STD    T14          TRANSMIT DATA TO PROCESSOR 14
         LDD    R14          RECEIVE DATA FROM PROCESSOR 14
         STD    T12          TRANSMIT DATA TO PROCESSOR 12
         SYNC                WAIT FOR DATA FROM PROCESSOR 12
         LDD    R12          RECEIVE DATA FROM PROCESSOR 12
         STD    SAVE         SAVE RESULT
         SYNC
         SYNC
         LBRA    NEXT
*
MLTFR    FDB     8748        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB    12521        INVERSE TRANSFORM COEFFICIENT
*
         ORG    $0000
MCND     FDB    0
PROD1    FCB    0
PROD2    FCB    0
PROD3    FCB    0
PROD4    FCB    0
TEMP     FCB    0
TEMP1    FCB    0
TEMP3    FCB    0
SAVE     FDB    0
FLAG     FCB    0
RES      FDB    0
*
         ORG    $FFFE
STRT     EQU    $F800
         END    BEGIN
*
         NAM    680914
*
*     **************************************************************
*     *                  PROCESSOR NUMBER 14                       *
*     **************************************************************
*
OUTPUT   EQU    $0400        OUTPUT COMMUNICATION LATCH
STATUS   EQU    $0402        STATUS LATCH
T9       EQU    $0403        TRANSMIT DATA TO PROCESSOR 9
T13      EQU    $0405        TRANSMIT DATA TO PROCESSOR 13
T18      EQU    $0407        TRANSMIT DATA TO PROCESSOR 18
INPUT    EQU    $0410        INPUT COMMUNICATION LATCH
R9       EQU    $0412        RECEIVE DATA FROM PROCESSOR 9
R13      EQU    $0414        RECEIVE DATA FROM PROCESSOR 13
R18      EQU    $0416        RECEIVE DATA FROM PROCESSOR 18
SEM      EQU    $0418
*
         ORG     $F800
         NOP
         ORCC    #%01010000
         LDU     #PROD1
BEGIN    CLRA
         STA     FLAG        SET FLAG=0 IF SEM=0
         LDA     SEM
         BEQ     FRD
START    LDA     #1
         STA     FLAG          SET FLAG=1 IS SEM=1
FRD      LDY     #MCND
         LDX     #MLTFR
         LDA     #1
         STA     STATUS        SET STATUS LATCH=1
         SYNC
         CLRA
         STA     STATUS        SET STATUS LATCH=0
         LDD     INPUT         READ INPUT LATCH
         BRA     OVER
NEXT     LDY     #MCND
         LDX     #MLTRR
         SYNC
         LDD     SAVE
*
OVER     STD     T9            TRANSMIT DATA TO PROCESSOR 9
         SYNC                  WAIT FOR DATA FROM PROCESSOR 9
         STD     SAVE          *
         LDD     R9            * MODULAR SUBTRACTION
         SUBD    SAVE          *
         BCC     SKP12         *
         ADDD    #65521        *
SKP12    SYNC
         STD     T13           TRANSMIT DATA TO PROCESSOR 13
         SYNC                  WAIT FOR DATA FROM PROCESSOR 13
         SUBD    R13           *
         BCC     SKP14         * MODULAR SUBTRACTION
         ADDD    #65521        *
SKP14    STD     T18           TRANSMIT DATA TO PROCESSOR 18
         SYNC
         SYNC
*
         STD     MCND
         CLR      ,U           *
         CLR     1,U           * MODULAR MULTIPLICATION
         LDA     1,X           * WITH TRANSFORM COEFFICIENTS
         LDB     1,Y           *
         MUL                   *
         STD     2,U           *
         LDA      ,X           *
         LDB     1,Y           *
         MUL                   *
         ADDD    1,U           *
         STD     1,U           *
         BCC     SKP16         *
         INC      ,U           *
SKP16    LDA     1,X           *
         LDB      ,Y           *
         MUL                   *
         ADDD    1,U           *
         STD     1,U           *
         BCC     SKP19         *
```

```
          INC    ,U          *
SKP19     LDA    ,X          *
          LDB    ,Y          *
          MUL                *
          ADDD   ,U          *
          STD    ,U          *
*                            *
          LDA   1,U          *
          LDB   #15          *
          MUL                *
          ADDD  2,U          *
          BCS   SKP20        *
          CMPD  #65521       *
          BLO   SKP21        *
SKP20     ADDD  #15          *
SKP21     STD   2,U          *
*                            *
          LDA    ,U          *
          LDX   #TEMP        *
          CLR    ,X          *
          CLR   1,X          *
          CLR   2,X          *
          LDB   #15          *
          MUL                *
          STD    ,X          *
          LDA    ,X          *
          LDB   #15          *
          MUL                *
          ADDD  1,X          *
          ADDD  2,U          *
          BCS   SKP22        *
          CMPD  #65521       *
          BLO   SKP23        *
SKP22     ADDD  #15          *
SKP23     SYNC
*
          SYNC               WAIT FOR DATA FROM PROCESSOR 18
          ADDD  R18          ADD DATA FROM PROCESSOR 18
          BCS   SKP24        *
          CMPD  #65521       * MODULAR ADDITION
          BLO   SKP25        *
SKP24     ADDD  #15          *
SKP25     SYNC
          STD   T13          TRANSMIT DATA TO PROCESSOR 13
          SYNC               WAIT FOR DATA FROM PROCESSOR 13
          STD   SAVE         *
          LDD   R13          * MODULAR SUBTRACTION
          SUBD  SAVE         *
          BCC   SKP26        *
          ADDD  #65521       *
SKP26     SYNC               WAIT FOR DATA FROM PROCESSOR 9
          STD   T9           TRANSMIT DATA TO PROCESSOR 9
          SYNC
          STD   SAVE
          LDD    R9          RECEIVE DATA FROM PROCESSOR 9
          SUBD   SAVE        *
          BCC    SKP28       * MODULAR SUBTRACTION
          ADDD   #65521      *
*
SKP28     STD     SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
          LDA     FLAG
          CMPA    #1
          BEQ     MULT
          CMPA    #2
          BEQ     CONV
          LDD     SAVE
          STD     RES
          LBRA    BEGIN
CONV      LDD     SAVE
          STD     OUTPUT     WRITE DATA IN OUTPUT LATCH
          LBRA    BEGIN
*
MULT      INC     FLAG
          LDX     #SAVE      PERFORM MULTIPLICATION OF THE TWO
          LDY     #RES       TRANSFORMED SEQUENCES
LOP15     CLR      ,U        *
          CLR    1,U         * MODULAR MULTIPLICATION
          LDA    1,X         *
          LDB    1,Y         *
          MUL                *
          STD    2,U         *
          LDA     ,X         *
          LDB    1,Y         *
          MUL                *
          ADDD   1,U         *
          STD    1,U         *
          BCC    LOP16       *
          INC     ,U         *
LOP16     LDA    1,X         *
          LDB     ,Y         *
          MUL                *
          ADDD   1,U         *
          STD    1,U         *
          BCC    LOP19       *
          INC     ,U         *
LOP19     LDA     ,X         *
          LDB     ,Y         *
          MUL                *
          ADDD    ,U         *
          STD     ,U         *
```

```
*                            *
          LDA   1,U          *
          LDB   #15          *
          MUL                *
          ADDD  2,U          *
          BCS   LOP20        *
          CMPD  #65521       *
          BLO   LOP21        *
LOP20     ADDD  #15          *
LOP21     STD   2,U          *
*                            *
          LDA    ,U          *
          LDX   #TEMP        *
          CLR    ,X          *
          CLR   1,X          *
          CLR   2,X          *
          LDB   #15          *
          MUL                *
          STD    ,X          *
          LDA    ,X          *
          LDB   #15          *
          MUL                *
          ADDD  1,X          *
          ADDD  2,U          *
          BCS   LOP22        *
          CMPD  #65521       *
          BLO   LOP23        *
LOP22     ADDD  #15          *
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
************************************************************
*
LOP23     STD    T13        TRANSMIT DATA TO PROCESSOR 13
          SYNC
          SYNC              WAIT FOR DATA FROM PROCESSOR 13
          LDD    R13        RECEIVE DATA FROM PROCESSOR 13
          STD    T18        TRANSMIT DATA TO PROCESSOR 18
          LDD    R18        RECEIVE DATA FROM PROCESSOR 18
          STD    T9         TRANSMIT DATA TO PROCESSOR 9
          SYNC              WAIT FOR DATA FROM PROCESSOR 9
          LDD    R9         RECEIVE DATA FROM PROCESSOR 9
          STD    SAVE       SAVE RESULT
          SYNC
          LBRA    NEXT
*
MLTFR     FDB    1465       FORWARD TRANSFORM COEFFICIENT
MLTRR     FDB   21938       INVERSE TRANSFORM COEFFICIENT
*
          ORG   $0000
MCND      FDB   0
PROD1     FCB   0
```

```
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB    0
RES      FDB    0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680915
*
*     ***************************************************************
*     *                  PROCESSOR NUMBER 15                        *
*     ***************************************************************
*
OUTPUT   EQU   $0400        OUTPUT COMMUNICATION LATCH
STATUS   EQU   $0402        STATUS LATCH
T10      EQU   $0403        TRANSMIT DATA TO PROCESSOR 10
T12      EQU   $0405        TRANSMIT DATA TO PROCESSOR 12
T18      EQU   $0407        TRANSMIT DATA TO PROCESSOR 18
INPUT    EQU   $0410        INPUT COMMUNICATION LATCH
R10      EQU   $0412        RECEIVE DATA FROM PROCESSOR 10
R12      EQU   $0414        RECEIVE DATA FROM PROCESSOR 12
R18      EQU   $0416        RECEIVE DATA FROM PROCESSOR 18
SEM      EQU   $0418
         ORG   $F800
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG        SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG        SET FLAG=1 IS SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         LDA    #1
         STA    STATUS      SET STATUS LATCH=1
         SYNC
         CLRA
         STA    STATUS      SET STATUS LATCH=0
         LDD    INPUT       READ INPUT LATCH
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
         SYNC
         LDD    SAVE
*
OVER     STD    T10           TRANSMIT DATA TO PROCESSOR 10
         SYNC                 WAIT FOR DATA FROM PROCESSOR 10
         STD    SAVE          *
         LDD    R10           * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP12         *
         ADDD   #65521        *
SKP12    SYNC
         STD    T12           TRANSMIT DATA TO PROCESSOR 12
         SYNC                 WAIT FOR DATA FROM PROCESSOR 12
         STD    SAVE          *
         LDD    R12           * MODULAR SUBTRACTION
         SUBD   SAVE          *
         BCC    SKP14         *
         ADDD   #65521        *
SKP14    STD    T18           TRANSMIT DATA TO PROCESSOR 18
         SYNC
         SYNC
*
         STD    MCND
         CLR     ,U           *
         CLR    1,U           * MODULAR MULTIPICATION
         LDA    1,X           * WITH TRANSFORM COEFFICIENTS
         LDB    1,Y           *
         MUL                  *
         STD    2,U           *
         LDA     ,X           *
         LDB    1,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    SKP16         *
         INC     ,U           *
SKP16    LDA    1,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    SKP19         *
         INC     ,U           *
SKP19    LDA     ,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD    ,U           *
         STD     ,U           *
*                             *
         LDA    1,U           *
         LDB    #15           *
         MUL                  *
         ADDD   2,U           *
         BCS    SKP20         *
         CMPD   #65521        *
         BLO   SKP21       *
SKP20    ADDD  #15         *
SKP21    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   SKP22       *
         CMPD  #65521      *
         BLO   SKP23       *
SKP22    ADDD  #15         *
SKP23    SYNC
*
         SYNC              WAIT FOR DATA FROM PROCESSOR 18
         SUBD  R18         *
         BCC   SKP24       * MODULAR SUBTRACTION
         ADDD  #65521      *
SKP24    SYNC
         STD   T12         TRANSMIT DATA TO PROCESSOR 12
         SYNC              WAIT FOR DATA FROM PROCESSOR 12
         STD   SAVE        *
         LDD   R12         * MODULAR SUBTRACTION
         SUBD  SAVE        *
         BCC   SKP26       *
         ADDD  #65521      *
SKP26    SYNC
         STD   T10         TRANSMIT DATA TO PROCESSOR 10
         SYNC              WAIT FOR DATA FROM PROCESSOR 10
         STD   SAVE        *
         LDD   R10         * MODULAR SUBTRACTION
         SUBD  SAVE        *
         BCC   SKP28       *
         ADDD  #65521      *
*
SKP28    STD    SAVE
*
*  CHECK FLAG
* IF  FLAG=0 PERFORM FORWARD TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=1 PERFORM FORWARD TRANSFORM OF SECOND SEQUENCE
*     AND MULTIPLY WITH TRANSFORM OF FIRST SEQUENCE
* IF  FLAG=2 PERFORM INVERSE TRANSFORM OF THE PRODUCT OF THE
*     THE TWO TRANSFORMS AND STORE RESULT IN THE OUTPUT LATCH
*
         LDA    FLAG
         CMPA   #1
         BEQ    MULT
         CMPA   #2
         BEQ    CONV
         LDD    SAVE
         STD    RES
         LBRA   BEGIN
CONV     LDD    SAVE
         STD    OUTPUT     WRITE DATA IN OUTPUT LATCH
         LBRA   BEGIN
*
MULT     INC    FLAG
         LDX    #SAVE      PERFORM MULTIPLICATION OF THE TWO
         LDY    #RES       TRANSFORMED SEQUENCES
LOP15    CLR    ,U         *
         CLR   1,U         * MODULAR MULTIPLICATION
         LDA   1,X         *
         LDB   1,Y         *
         MUL               *
         STD   2,U         *
         LDA    ,X         *
         LDB   1,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP16       *
         INC    ,U         *
LOP16    LDA   1,X         *
         LDB    ,Y         *
         MUL               *
         ADDD  1,U         *
         STD   1,U         *
         BCC   LOP19       *
         INC    ,U         *
LOP19    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
         STD    ,U         *
*                          *
         LDA   1,U         *
         LDB   #15         *
         MUL               *
         ADDD  2,U         *
         BCS   LOP20       *
         CMPD  #65521      *
         BLO   LOP21       *
LOP20    ADDD  #15         *
LOP21    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
```

```
          CLR    2,X          *
          LDB    #15          *
          MUL                 *
          STD     ,X          *
          LDA     ,X          *
          LDB    #15          *
          MUL                 *
          ADDD   1,X          *
          ADDD   2,U          *
          BCS    LOP22        *
          CMPD   #65521       *
          BLO    LOP23        *
LOP22     ADDD   #15          *
*
****************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE                  *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH            *
*  THE EXISTING COMMUNICATION LATCHES                          *
****************************************************************
*
LOP23     STD     T18         TRANSMIT DATA TO PROCESSOR 18
          SYNC
          SYNC
          SYNC
          SYNC                WAIT FOR DATA FROM PROCESSOR 18
          LDD     R18         RECEIVE DATA FROM PROCESSOR 18
          STD     SAVE
          LBRA    NEXT
*
MLTFR     FDB    23174        FORWARD TRANSFORM COEFFICIENT
MLTRR     FDB     5913        INVERSE TRANSFORM COEFFICIENT
*
          ORG    $0000
MCND      FDB    0
PROD1     FCB    0
PROD2     FCB    0
PROD3     FCB    0
PROD4     FCB    0
TEMP      FCB    0
TEMP1     FCB    0
TEMP3     FCB    0
SAVE      FDB    0
FLAG      FCB    0
RES       FDB    0
*
          ORG    $FFFE
STRT      EQU    $F800
          END    BEGIN
*
          NAM    680916
*
*     **************************************************************
*     *                    PROCESSOR NUMBER 16                     *
*     **************************************************************
```

```
*
T4       EQU    $0410         TRANSMIT DATA TO PROCESSOR 4
T5       EQU    $0412         TRANSMIT DATA TO PROCESSOR 5
R4       EQU    $0414         RECEIVE DATA FROM PROCESSOR 4
R5       EQU    $0416         RECEIVE DATA FROM PROCESSOR 5
SEM      EQU    $0418
*
         ORG    $F800
         NOP
         ORCC   #%01010000
         LDU    #PROD1
BEGIN    CLRA
         STA    FLAG          SET FLAG=0 IF SEM=0
         LDA    SEM
         BEQ    FRD
START    LDA    #1
         STA    FLAG          SET FLAG=1 IF SEM=1
FRD      LDY    #MCND
         LDX    #MLTFR
         BRA    OVER
NEXT     LDY    #MCND
         LDX    #MLTRR
OVER     SYNC
         SYNC
         SYNC
         SYNC
         SYNC                 WAIT FOR DATA FROM PROCESSOR 4
         LDD    R4            RECEIVE DATA FROM PROCESSOR 4
         ADDD   R5            ADD DATA FROM PROCESSOR 5
         BCS    SKP12         *
         CMPD   #65521        * MODULAR ADDITION
         BLO    SKP13         *
SKP12    ADDD   #15           *
SKP13    SYNC
*
         STD    MCND
         CLR     ,U           *
         CLR    1,U           * MODULAR MULTIPLICATION
         LDA    1,X           * WITH TRANSFORM COEFFICIENTS
         LDB    1,Y           *
         MUL                  *
         STD    2,U           *
         LDA     ,X           *
         LDB    1,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC    SKP14         *
         INC     ,U           *
SKP14    LDA    1,X           *
         LDB     ,Y           *
         MUL                  *
         ADDD   1,U           *
         STD    1,U           *
         BCC   SKP17       *
         INC    ,U         *
SKP17    LDA    ,X         *
         LDB    ,Y         *
         MUL               *
         ADDD   ,U         *
         STD    ,U         *
*                          *
         LDA   1,U         *
         LDB   #15         *
         MUL               *
         ADDD  2,U         *
         BCS   SKP18       *
         CMPD  #65521      *
         BLO   SKP19       *
SKP18    ADDD  #15         *
SKP19    STD   2,U         *
*                          *
         LDA    ,U         *
         LDX   #TEMP       *
         CLR    ,X         *
         CLR   1,X         *
         CLR   2,X         *
         LDB   #15         *
         MUL               *
         STD    ,X         *
         LDA    ,X         *
         LDB   #15         *
         MUL               *
         ADDD  1,X         *
         ADDD  2,U         *
         BCS   SKP20       *
         CMPD  #65521      *
         BLO   SKP21       *
SKP20    ADDD  #15         *
SKP21    SYNC              WAIT FOR OTHER PROCESSORS TO
*                          COMPLETE MULTIPLICATION
         STD   T5          TRANSMIT DATA TO PROCESSOR 5
         STD   T4          TRANSMIT DATA TO PROCESSOR 4
         SYNC
         SYNC
         SYNC
         SYNC
         SYNC
         LDA    FLAG
         CMPA   #1
         BEQ    SKP
         LBRA   BEGIN
SKP      INC    FLAG
*
************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE              *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH        *
*  THE EXISTING COMMUNICATION LATCHES                      *
```

```
****************************************************************
*
         SYNC               WAIT FOR DATA FROM PROCESSOR 5
         LDD    R5          RECEIVE DATA FROM PROCESSOR 5
         STD    T4          TRANSMIT DATA TO PROCESSOR 4
         SYNC
         SYNC
         SYNC
         LBRA   NEXT
*
MLTFR    FDB    18005       FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB     5493       INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
         NAM   680917
*
*     **************************************************************
*     *                   PROCESSOR NUMBER 17                      *
*     **************************************************************
*
T9       EQU   $0410        TRANSMIT DATA TO PROCESSOR 9
T10      EQU   $0412        TRANSMIT DATA TO PROCESSOR 10
R9       EQU   $0414        RECEIVE DATA FROM PROCESSOR 9
R10      EQU   $0416        RECEIVE DATA FROM PROCESSOR 10
SEM      EQU   $0418
*
         ORG   $F800
         NOP
         ORCC  #%01010000
         LDU   #PROD1
BEGIN    CLRA
         STA   FLAG         SET FLAG=0 IF SEM=0
         LDA   SEM
         BEQ   FRD
START    LDA   #1
         STA   FLAG         SET FLAG=1 IF SEM=1
FRD      LDY   #MCND
         LDX   #MLTFR
         BRA   OVER
```

```
NEXT      LDY    #MCND
          LDX    #MLTRR
OVER      SYNC
          SYNC
          SYNC
          SYNC
          SYNC                    WAIT FOR DATA FROM PROCESSOR 9
          LDD    R9               RECEIVE DATA FROM PROCESSOR 9
          ADDD   R10              ADD DATA FROM PROCESSOR 10
          BCS    SKP12            *
          CMPD   #65521           * MODULAR ADDITION
          BLO    SKP13            *
SKP12     ADDD   #15
SKP13     SYNC
*
          STD    MCND
          CLR     ,U              *
          CLR    1,U              * MODULAR MULTIPLICATION
          LDA    1,X              * WITH TRANSFORM COEFFICIENTS
          LDB    1,Y              *
          MUL                     *
          STD    2,U              *
          LDA     ,X              *
          LDB    1,Y              *
          MUL                     *
          ADDD   1,U              *
          STD    1,U              *
          BCC    SKP14            *
          INC     ,U              *
SKP14     LDA    1,X              *
          LDB     ,Y              *
          MUL                     *
          ADDD   1,U              *
          STD    1,U              *
          BCC    SKP17            *
          INC     ,U              *
SKP17     LDA     ,X              *
          LDB     ,Y              *
          MUL                     *
          ADDD    ,U              *
          STD     ,U              *
*                                 *
          LDA    1,U              *
          LDB    #15              *
          MUL                     *
          ADDD   2,U              *
          BCS    SKP18            *
          CMPD   #65521           *
          BLO    SKP19            *
SKP18     ADDD   #15              *
SKP19     STD    2,U              *
*                                 *
          LDA     ,U              *
          LDX    #TEMP            *
          CLR      ,X        *
          CLR     1,X        *
          CLR     2,X        *
          LDB     #15        *
          MUL                *
          STD      ,X        *
          LDA      ,X        *
          LDB     #15        *
          MUL                *
          ADDD    1,X        *
          ADDD    2,U        *
          BCS     SKP20      *
          CMPD    #65521     *
          BLO     SKP21      *
SKP20     ADDD    #15        *
SKP21     SYNC
*
          STD     T10             TRANSMIT DATA TO PROCESSOR 10
          STD     T9              TRANSMIT DATA TO PROCESSOR 9
          SYNC
          SYNC
          SYNC
          SYNC
          SYNC
          LDA      FLAG
          CMPA     #1
          BEQ      SKP
          LBRA     BEGIN
SKP       INC      FLAG
*
******************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE                    *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH              *
*  THE EXISTING COMMUNICATION LATCHES                            *
******************************************************************
*
          SYNC
          LDD      R10            RECEIVE DATA FROM PROCESSOR 10
          STD      T9             TRANSMIT DATA TO PROCESSOR 9
          SYNC
          SYNC                    WAIT FOR DATA FROM PROCESSOR 9
          LDD      R9             RECEIVE DATA FROM PROCESSOR 9
          STD      T10            TRANSMIT DATA TO PROCESSOR 10
          SYNC
          LBRA     NEXT
*
MLTFR     FDB      5753           FORWARD TRANSFORM COEFFICIENT
MLTRR     FDB     34561           INVERSE TRANSFORM COEFFICIENT
*
          ORG     $0000
MCND      FDB     0
PROD1     FCB     0
PROD2     FCB     0
PROD3     FCB     0
```

```
PROD4     FCB   0
TEMP      FCB   0
TEMP1     FCB   0
TEMP3     FCB   0
SAVE      FDB   0
FLAG      FCB   0
*
          ORG   $FFFE
STRT      EQU   $F800
          END   BEGIN
*
          NAM   680918
*
*       ************************************************************
*       *                   PROCESSOR NUMBER 18                    *
*       ************************************************************
*
T14       EQU   $0410        TRANSMIT DATA TO PROCESSOR 14
T15       EQU   $0412        TRANSMIT DATA TO PROCESSOR 15
R14       EQU   $0414        RECEIVE DATA FROM PROCESSOR 14
R15       EQU   $0416        RECEIVE DATA FROM PROCESSOR 15
SEM       EQU   $0418
*
          ORG   $F800
          NOP
          ORCC  #%01010000
          LDU   #PROD1
BEGIN     CLRA
          STA   FLAG         SET FLAG=0 IF SEM=0
          LDA   SEM
          BEQ   FRD
START     LDA   #1
          STA   FLAG         SET FLAG=1 IF SEM=1
FRD       LDY   #MCND
          LDX   #MLTFR
          BRA   OVER
NEXT      LDY   #MCND
          LDX   #MLTRR
OVER      SYNC
          SYNC
          SYNC
          SYNC
          SYNC               WAIT FOR DATA FROM PROCESSOR 14
          LDD   R14          RECEIVE DATA FROM PROCESSOR 14
          ADDD  R15          ADD DATA FROM PROCESSOR 15
          BCS   SKP12        *
          CMPD  #65521       * MODULAR ADDITION
          BLO   SKP13        *
SKP12     ADDD  #15          *
SKP13     SYNC
*
          STD   MCND
          CLR   ,U           *
          CLR   1,U          * MODULAR MULTIPLICATION
```

```
         LDA   1,X          * WITH TRANSFORM COEFFICIENTS
         LDB   1,Y          *
         MUL                *
         STD   2,U          *
         LDA    ,X          *
         LDB   1,Y          *
         MUL                *
         ADDD  1,U          *
         STD   1,U          *
         BCC   SKP14        *
         INC    ,U          *
SKP14    LDA   1,X          *
         LDB    ,Y          *
         MUL                *
         ADDD  1,U          *
         STD   1,U          *
         BCC   SKP17        *
         INC    ,U          *
SKP17    LDA    ,X          *
         LDB    ,Y          *
         MUL                *
         ADDD   ,U          *
         STD    ,U          *
*                           *
         LDA   1,U          *
         LDB   #15          *
         MUL                *
         ADDD  2,U          *
         BCS   SKP18        *
         CMPD  #65521       *
         BLO   SKP19        *
SKP18    ADDD  #15          *
SKP19    STD   2,U          *
*                           *
         LDA    ,U          *
         LDX   #TEMP        *
         CLR    ,X          *
         CLR   1,X          *
         CLR   2,X          *
         LDB   #15          *
         MUL                *
         STD    ,X          *
         LDA    ,X          *
         LDB   #15          *
         MUL                *
         ADDD  1,X          *
         ADDD  2,U          *
         BCS   SKP20        *
         CMPD  #65521       *
         BLO   SKP21        *
SKP20    ADDD  #15          *
SKP21    SYNC               WAIT FOR OTHER PROCESSORS TO
*                           COMPLETE MULTIPLICATION
         STD   T15          TRANSMIT DATA TO PROCESSOR 15
         STD   T14          TRANSMIT DATA TO PROCESSOR 14
         SYNC
         SYNC
         SYNC
         SYNC
         SYNC
         LDA    FLAG
         CMPA   #1
         BEQ    SKP
         LBRA   BEGIN
SKP      INC    FLAG
*
**************************************************************
*  RESHUFFLING THE DATA BEFORE PERFORMING THE               *
*  INVERSE TRANSFORM. THE DATA IS EXCHANGED THROUGH         *
*  THE EXISTING COMMUNICATION LATCHES                       *
**************************************************************
*
         SYNC               WAIT FOR DATA FROM PROCESSOR 15
         LDD    R15         RECEIVE DATA FROM PROCESSOR 15
         STD    T14         TRANSMIT DATA TO PROCESSOR 14
         SYNC
         SYNC               WAIT FOR DATA FROM PROCESSOR 14
         LDD    R14         RECEIVE DATA FROM PROCESSOR 14
         STD    T15         TRANSMIT DATA TO PROCESSOR 15
         SYNC
         LBRA   NEXT
*
MLTFR    FDB   43615        FORWARD TRANSFORM COEFFICIENT
MLTRR    FDB   24748        INVERSE TRANSFORM COEFFICIENT
*
         ORG   $0000
MCND     FDB   0
PROD1    FCB   0
PROD2    FCB   0
PROD3    FCB   0
PROD4    FCB   0
TEMP     FCB   0
TEMP1    FCB   0
TEMP3    FCB   0
SAVE     FDB   0
FLAG     FCB   0
*
         ORG   $FFFE
STRT     EQU   $F800
         END   BEGIN
*
*        NAM   WINO15
*
*    *****************************************************
*    *    WINOGRAD'S 15 POINT ALGORITHM                  *
*    *   Performs 15 point transform on array in 'ARYIN'  *
*    *****************************************************
*
```

```
         ORG    $E000
WIN15    LDX    #AX           LOAD ADDRESS OF AX IN INDEX REG X
         LDY    #ARYIN        LOAD ADDRESS OF ARYIN IN INDEX REG Y
*
*    ************************************************************
*    *              REORDERING THE INPUT ARRAY                  *
*    ************************************************************
*
         LDD      ,Y
         STD      ,X
         LDD     6,Y
         STD     2,X
         LDD    12,Y
         STD     4,X
         LDD    18,Y
         STD     6,X
         LDD    24,Y
         STD     8,X
         LDD    10,Y
         STD    10,X
         LDD    16,Y
         STD    12,X
         LDD    22,Y
         STD    14,X
         LDD    28,Y
         STD    16,X
         LDD     4,Y
         STD    18,X
         LDD    20,Y
         STD    20,X
         LDD    26,Y
         STD    22,X
         LDD     2,Y
         STD    24,X
         LDD     8,Y
         STD    26,X
         LDD    14,Y
         STD    28,X
*
*    ************************************************************
*    *              3 POINT PREWEAVE                            *
*    ************************************************************
*
*                             INDEX REGISTER STILL
SKP2     LDD    10,X          CONTAINS ADDRESS OF ARRAY AX
         ADDD   20,X          **********************************
         BCS    JMP1          *                                *
         CMPD   #65521        *      MODULAR ADDITION          *
         BLO    JMP2          *                                *
JMP1     ADDD   #15           **********************************
JMP2     STD    TMP1
         ADDD     ,X
         BCS    JMP3
         CMPD   #65521
```

```
          BLO    JMP4
JMP3      ADDD   #15
JMP4      STD      ,X
          LDD    10,X
          SUBD   20,X          ********************************
          BCC    JMP5          *     MODULAR SUBTRACTION      *
          ADDD   #65521        ********************************
JMP5      STD    20,X
          LDD    TMP1
          STD    10,X
          LDD    12,X
          ADDD   22,X
          BCS    JMP6
          CMPD   #65521
          BLO    JMP7
JMP6      ADDD   #15
JMP7      STD    TMP1
          ADDD   2,X
          BCS    JMP8
          CMPD   #65521
          BLO    JMP9
JMP8      ADDD   #15
JMP9      STD    2,X
          LDD    12,X
          SUBD   22,X
          BCC    JMP10
          ADDD   #65521
JMP10     STD    22,X
          LDD    TMP1
          STD    12,X
          LDD    14,X
          ADDD   24,X
          BCS    JMP11
          CMPD   #65521
          BLO    JMP12
JMP11     ADDD   #15
JMP12     STD    TMP1
          ADDD   4,X
          BCS    JMP13
          CMPD   #65521
          BLO    JMP14
JMP13     ADDD   #15
JMP14     STD    4,X
          LDD    14,X
          SUBD   24,X
          BCC    JMP15
          ADDD   #65521
JMP15     STD    24,X
          LDD    TMP1
          STD    14,X
          LDD    16,X
          ADDD   26,X
          BCS    JMP16
          CMPD   #65521
```

```
         BLO   JMP17
JMP16    ADDD  #15
JMP17    STD   TMP1
         ADDD  6,X
         BCS   JMP18
         CMPD  #65521
         BLO   JMP19
JMP18    ADDD  #15
JMP19    STD   6,X
         LDD   16,X
         SUBD  26,X
         BCC   JMP20
         ADDD  #65521
JMP20    STD   26,X
         LDD   TMP1
         STD   16,X
         LDD   18,X
         ADDD  28,X
         BCS   JMP21
         CMPD  #65521
         BLO   JMP22
JMP21    ADDD  #15
JMP22    STD   TMP1
         ADDD  8,X
         BCS   JMP23
         CMPD  #65521
         BLO   JMP24
JMP23    ADDD  #15
JMP24    STD   8,X
         LDD   18,X
         SUBD  28,X
         BCC   JMP25
         ADDD  #65521
JMP25    STD   28,X
         LDD   TMP1
         STD   18,X
*
*    *******************************************************
*    *    5 POINT PREWEAVE                                 *
*    *******************************************************
*
         LDY   #Z
         LDD   2,X
         ADDD  8,X
         BCS   JMP26
         CMPD  #65521
         BLO   JMP27
JMP26    ADDD  #15
JMP27    STD   2,Y
         LDD   2,X
         SUBD  8,X
         BCC   JMP28
         ADDD  #65521
JMP28    STD   6,Y
```

```
          LDD    4,X
          ADDD   6,X
          BCS    JMP29
          CMPD   #65521
          BLO    JMP30
JMP29     ADDD   #15
JMP30     STD    4,Y
          LDD    6,X
          SUBD   4,X
          BCC    JMP31
          ADDD   #65521
JMP31     STD    10,Y
          ADDD   6,Y
          BCS    JMP32
          CMPD   #65521
          BLO    JMP33
JMP32     ADDD   #15
JMP33     STD    8,Y
          LDD    4,Y
          ADDD   2,Y
          BCS    JMP34
          CMPD   #65521
          BLO    JMP35
JMP34     ADDD   #15
JMP35     STD    TMP1
          ADDD     ,X
          BCS    JMP36
          CMPD   #65521
          BLO    JMP37
JMP36     ADDD   #15
JMP37     STD      ,Y
          LDD     2,Y
          SUBD    4,Y
          BCC    JMP38
          ADDD   #65521
JMP38     STD     4,Y
          LDD    TMP1
          STD     2,Y
*
*
*
          LDD    12,X
          ADDD   18,X
          BCS    JMP39
          CMPD   #65521
          BLO    JMP40
JMP39     ADDD   #15
JMP40     STD    14,Y
          LDD    12,X
          SUBD   18,X
          BCC    JMP41
          ADDD   #65521
JMP41     STD    18,Y
          LDD    14,X
```

```
         ADDD  16,X
         BCS   JMP42
         CMPD  #65521
         BLO   JMP43
JMP42    ADDD  #15
JMP43    STD   16,Y
         LDD   16,X
         SUBD  14,X
         BCC   JMP44
         ADDD  #65521
JMP44    STD   22,Y
         ADDD  18,Y
         BCS   JMP45
         CMPD  #65521
         BLO   JMP46
JMP45    ADDD  #15
JMP46    STD   20,Y
         LDD   16,Y
         ADDD  14,Y
         BCS   JMP47
         CMPD  #65521
         BLO   JMP48
JMP47    ADDD  #15
JMP48    STD   TMP1
         ADDD  10,X
         BCS   JMP49
         CMPD  #65521
         BLO   JMP50
JMP49    ADDD  #15
JMP50    STD   12,Y
         LDD   14,Y
         SUBD  16,Y
         BCC   JMP51
         ADDD  #65521
JMP51    STD   16,Y
         LDD   TMP1
         STD   14,Y
*
*
*
         LDD   22,X
         ADDD  28,X
         BCS   JMP52
         CMPD  #65521
         BLO   JMP53
JMP52    ADDD  #15
JMP53    STD   26,Y
         LDD   22,X
         SUBD  28,X
         BCC   JMP54
         ADDD  #65521
JMP54    STD   30,Y
         LDD   24,X
         ADDD  26,X
         BCS   JMP56
         CMPD  #65521
         BLO   JMP57
JMP56    ADDD  #15
JMP57    STD   28,Y
         LDD   26,X
         SUBD  24,X
         BCC   JMP58
         ADDD  #65521
JMP58    STD   34,Y
         ADDD  30,Y
         BCS   JMP59
         CMPD  #65521
         BLO   JMP60
JMP59    ADDD  #15
JMP60    STD   32,Y
         LDD   26,Y
         ADDD  28,Y
         BCS   JMP61
         CMPD  #65521
         BLO   JMP62
JMP61    ADDD  #15
JMP62    STD   TMP1
         ADDD  20,X
         BCS   JMP63
         CMPD  #65521
         BLO   JMP64
JMP63    ADDD  #15
JMP64    STD   24,Y
         LDD   26,Y
         SUBD  28,Y
         BCC   JMP65
         ADDD  #65521
JMP65    STD   28,Y
         LDD   TMP1
         STD   26,Y
*
*
*    ********************************************************
*    *            START OF MULTIPLICATION                   *
*    ********************************************************
*
*     ROUTINE FOR 16*16-BIT UNSIGNED MULTIPICATION
*
         CLRA
         STA   IND               INDEX FOR MULTIPLIER
         LDS   #Z
LOOP     LDA   FRD
         BEQ   OVER1             CHECK FRD
         LDY   #COEFR            LOAD INVERSE TRANSFORM COEFFICIENTS
         BRA   OVER2
OVER1    LDY   #COEFF            LOAD FORWARD TRANSFORM COEFFICIENTS
OVER2    LDA   IND
         LDD   A,Y
         STD   MLTR
         LDD     ,S
         STD   MLTN
         LDX   #MLTR
         LDY   #MLTN
         LDU   #PROD1
         CLR     ,U
         CLR    1,U
         LDA    1,X
         LDB    1,Y
         MUL
         STD    2,U
         LDA     ,X
         LDB    1,Y
         MUL
         ADDD   1,U
         STD    1,U
         BCC   SKIP3
         INC     ,U
SKIP3    LDA    1,X
         LDB     ,Y
         MUL
         ADDD  1,U
         STD   1,U
         BCC   SKIP4
         INC     ,U
SKIP4    LDA     ,X
         LDB     ,Y
         MUL
         ADDD    ,U
         STD     ,U
*
* MODULARISING THE 32-BIT PRODUCT
*
         LDA    1,U
         LDB   #15
         MUL
         ADDD   2,U
         BCS   SKIP6
         CMPD  #65521
         BLO   SKIP7
SKIP6    ADDD  #15
SKIP7    STD    2,U
         LDA     ,U
         LDY   #TEMP
         CLR     ,Y
         CLR    1,Y
         CLR    2,Y
         LDB   #15
         MUL
SKIPA    STD     ,Y
         LDA     ,Y
         BEQ   SKIPE
         LDB   #15
         MUL
         ADDD   1,Y
         BRA   SKIPD
SKIPE    LDD    1,Y
SKIPD    ADDD   2,U
         BCS   SKIPB
         CMPD  #65521
         BLO   SKIPC
SKIPB    ADDD  #15
SKIPC    STD     ,S++
         LDA   IND
         ADDA  #2
         STA   IND
         CMPA  #34
         LBLS  LOOP
*
*    ********************************************************
*    *      5-POINT POSTWEAVE                               *
*    ********************************************************
*
         LDX   #AX
         LDY   #Z
         LDD     ,Y
         STD     ,X
         ADDD   2,Y
         BCS   JMP67
         CMPD  #65521
         BLO   JMP68
JMP67    ADDD  #15
JMP68    STD   2,X
         LDD   8,Y
         ADDD  10,Y
         BCS   JMP69
         CMPD  #65521
         BLO   JMP70
JMP69    ADDD  #15
JMP70    STD   10,Y
         LDD   6,Y
         SUBD  8,Y
         BCC   JMP71
         ADDD  #65521
JMP71    STD   8,X
         LDD   2,X
         ADDD  4,Y
         BCS   JMP72
         CMPD  #65521
         BLO   JMP73
JMP72    ADDD  #15
JMP73    STD   TMP1
         LDD   2,X
         SUBD  4,Y
         BCC   JMP74
         ADDD  #65521
JMP74    STD   4,X
```

```
         SUBD   10,Y
         BCC    JMP75
         ADDD   #65521
JMP75    STD    6,X
         LDD    TMP1
         STD    2,X
         LDD    4,X
         ADDD   10,Y
         BCS    JMP76
         CMPD   #65521
         BLO    JMP77
JMP76    ADDD   #15
JMP77    STD    4,X
         LDD    2,X
         ADDD   8,X
         BCS    JMP78
         CMPD   #65521
         BLO    JMP79
JMP78    ADDD   #15
JMP79    STD    TMP1
         LDD    2,X
         SUBD   8,X
         BCC    JMP80
         ADDD   #65521
JMP80    STD    8,X
         LDD    TMP1
         STD    2,X
*
         LDD    12,Y
         STD    10,X
         ADDD   14,Y
         BCS    JUP67
         CMPD   #65521
         BLO    JUP68
JUP67    ADDD   #15
JUP68    STD    12,X
         LDD    20,Y
         ADDD   22,Y
         BCS    JUP69
         CMPD   #65521
         BLO    JUP70
JUP69    ADDD   #15
JUP70    STD    22,Y
         LDD    18,Y
         SUBD   20,Y
         BCC    JUP71
         ADDD   #65521
JUP71    STD    18,X
         LDD    12,X
         ADDD   16,Y
         BCS    JUP72
         CMPD   #65521
         BLO    JUP73
JUP72    ADDD   #15
```

```
JUP73    STD   TMP1
         LDD   12,X
         SUBD  16,Y
         BCC   JUP74
         ADDD  #65521
JUP74    STD   14,X
         SUBD  22,Y
         BCC   JUP75
         ADDD  #65521
JUP75    STD   16,X
         LDD   TMP1
         STD   12,X
         LDD   14,X
         ADDD  22,Y
         BCS   JUP76
         CMPD  #65521
         BLO   JUP77
JUP76    ADDD  #15
JUP77    STD   14,X
         LDD   12,X
         ADDD  18,X
         BCS   JUP78
         CMPD  #65521
         BLO   JUP79
JUP78    ADDD  #15
JUP79    STD   TMP1
         LDD   12,X
         SUBD  18,X
         BCC   JUP80
         ADDD  #65521
JUP80    STD   18,X
         LDD   TMP1
         STD   12,X
*
         LDD   24,Y
         STD   20,X
         ADDD  26,Y
         BCS   SKP67
         CMPD  #65521
         BLO   SKP68
SKP67    ADDD  #15
SKP68    STD   22,X
         LDD   32,Y
         ADDD  34,Y
         BCS   SKP69
         CMPD  #65521
         BLO   SKP70
SKP69    ADDD  #15
SKP70    STD   34,Y
         LDD   30,Y
         SUBD  32,Y
         BCC   SKP71
         ADDD  #65521
SKP71    STD   28,X
```

```
        LDD   22,X
        ADDD  28,Y
        BCS   SKP72
        CMPD  #65521
        BLO   SKP73
SKP72   ADDD  #15
SKP73   STD   TMP1
        LDD   22,X
        SUBD  28,Y
        BCC   SKP74
        ADDD  #65521
SKP74   STD   24,X
        SUBD  34,Y
        BCC   SKP75
        ADDD  #65521
SKP75   STD   26,X
        LDD   TMP1
        STD   22,X
        LDD   24,X
        ADDD  34,Y
        BCS   SKP76
        CMPD  #65521
        BLO   SKP77
SKP76   ADDD  #15
SKP77   STD   24,X
        LDD   22,X
        ADDD  28,X
        BCS   SKP78
        CMPD  #65521
        BLO   SKP79
SKP78   ADDD  #15
SKP79   STD   TMP1
        LDD   22,X
        SUBD  28,X
        BCC   SKP80
        ADDD  #65521
SKP80   STD   28,X
        LDD   TMP1
        STD   22,X
*
*
*   **********************************************
*   *   THREE POINT POST-WEAVE                   *
*   **********************************************
*
        LDD     ,X
        ADDD  10,X
        BCS   JMP81
        CMPD  #65521
        BLO   JMP82
JMP81   ADDD  #15
JMP82   STD   10,X
        LDD    2,X
        ADDD  12,X
```

```
         BCS    JMP83
         CMPD   #65521
         BLO    JMP84
JMP83    ADDD   #15
JMP84    STD    12,X
         LDD     4,X
         ADDD   14,X
         BCS    JMP85
         CMPD   #65521
         BLO    JMP86
JMP85    ADDD   #15
JMP86    STD    14,X
         LDD     6,X
         ADDD   16,X
         BCS    JMP87
         CMPD   #65521
         BLO    JMP88
JMP87    ADDD   #15
JMP88    STD    16,X
         LDD     8,X
         ADDD   18,X
         BCS    JMP89
         CMPD   #65521
         BLO    JMP90
JMP89    ADDD   #15
JMP90    STD    18,X
         LDD    10,X
         ADDD   20,X
         BCS    JMP91
         CMPD   #65521
         BLO    JMP92
JMP91    ADDD   #15
JMP92    STD    TMP1
         LDD    10,X
         SUBD   20,X
         BCC    JMP911
         ADDD   #65521
JMP911   STD    20,X
         LDD    TMP1
         STD    10,X
         LDD    12,X
         ADDD   22,X
         BCS    JMP922
         CMPD   #65521
         BLO    JMP93
JMP922   ADDD   #15
JMP93    STD    TMP1
         LDD    12,X
         SUBD   22,X
         BCC    JMP94
         ADDD   #65521
JMP94    STD    22,X
         LDD    TMP1
         STD    12,X
```

```
          LDD    14,X
          ADDD   24,X
          BCS    JMP95
          CMPD   #65521
          BLO    JMP96
JMP95     ADDD   #15
JMP96     STD    TMP1
          LDD    14,X
          SUBD   24,X
          BCC    JMP97
          ADDD   #65521
JMP97     STD    24,X
          LDD    TMP1
          STD    14,X
          LDD    16,X
          ADDD   26,X
          BCS    JMP98
          CMPD   #65521
          BLO    JMP99
JMP98     ADDD   #15
JMP99     STD    TMP1
          LDD    16,X
          SUBD   26,X
          BCC    JMP100
          ADDD   #65521
JMP100    STD    26,X
          LDD    TMP1
          STD    16,X
          LDD    18,X
          ADDD   28,X
          BCS    JMP101
          CMPD   #65521
          BLO    JMP102
JMP101    ADDD   #15
JMP102    STD    TMP1
          LDD    18,X
          SUBD   28,X
          BCC    JMP103
          ADDD   #65521
JMP103    STD    28,X
          LDD    TMP1
          STD    18,X
*
*    ***********************************************************
*    *                  OUTPUT REORDERING                      *
*    ***********************************************************
*
          LDX    #AX
          LDY    #OUT
          LDD      ,X
          STD      ,Y
          LDD    12,X
          STD     2,Y
          LDD    24,X
```

```
         STD     4,Y
         LDD     6,X
         STD     6,Y
         LDD    18,X
         STD     8,Y
         LDD    20,X
         STD    10,Y
         LDD     2,X
         STD    12,Y
         LDD    14,X
         STD    14,Y
         LDD    26,X
         STD    16,Y
         LDD     8,X
         STD    18,Y
         LDD    10,X
         STD    20,Y
         LDD    22,X
         STD    22,Y
         LDD     4,X
         STD    24,Y
         LDD    16,X
         STD    26,Y
         LDD    28,X
         STD    28,Y
*
*    FORWARD TRANSFORM MULTIPLIER COEFFICIENTS
*
COEFF    FDB    1,16379,13376,19136,18005
         FDB    48647,32759,8192,45457
         FDB    36817,5753,25311,16087,29032
         FDB    8748,23174,43615,1465
*
*    INVERSE TRANSFORM MULTIPLIER COEFFICIENTS
*
COEFR    FDB    61153,5460,18364,46773,20640
         FDB    5493,6552,57331,37975,28122,
         FDB    34561,24521,29504,28641,12521
         FDB    5913,24748,21938
*
*************************************************************
*     RAM START ADDRESS $0000 - $3FFF                       *
*         Stack is from $FF to $00                          *
*************************************************************
*
         ORG    $0100
STACK    RMB    1             ADDRESS OF THE START OF STACK
MLTR     FDB    0             MULTIPLIER DOUBLE BYTE
MLTN     FDB    0             MULTIPLICAND DOUBLE BYTE
PROD1    FCB    0             32 BIT PRODUCT IS STORED IN
PROD2    FCB    0             PROD1:PROD2:PROD3:PROD4
PROD3    FCB    0
PROD4    FCB    0
TEMP     FDB    0             TEMPORARY MEMORY LOCATION FOR
```

```
TEMP1    FCB    0              HOLDING INTERMEDIATE VALUES
TEMP2    FCB    0
FRD      FCB    0              IF FRD=0 FORWARD OTHERWISE INVERSE TRANSFORM
TMP1     RMB    2
TMP2     RMB    1
CNT      RMB    1              CNT,CNT1,CNT2 ARE USED AS COUNTERS
CNT1     RMB    1
CNT2     RMB    1
IND      RMB    1              USED FOR INDEXING PURPOSES
STATUS   RMB    1              CONTAINS COPY OF CONTRL LATCH
AX       RMB    1              CONTAINS INTERMEDIATE RESULTS
Z        RMB    36             HOLDS INTERMEDIATE RESULTS
OUT      RMB    30             RESULT OF 15 POINT TRANSFORM FOR VERIFYING
ARRAY    RMB    30             CONTAINS COPY OF PARALLEL TRANSFORM O/P
PAD1     RMB    2
ARYIN    RMB    30             READS INPUT ARRAY FROM VDU INTO THIS ARRAY
PAD2     RMB    2
         END   STRT
```

Appendix-E

Backplane wiring connections for the parallel microprocessor system

BOARD NO B - (PROCESSOR 4 5 16)

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA IN	4 5
9-12	CLOCK	4 5
13-20	DATA OUT	4 5
21-24	OUTPUT ENABLE	4 5
25-32	TRANSMIT	4 -> 9 4 -> 3
33-36	CLOCK	4 -> 9 4 -> 3
37-44	RECEIVE	9 -> 4
45-46	CLOCK	9 -> 4
47-54	RECEIVE	3 -> 4
55-56	CLOCK	3 -> 4
57-64	TRANSMIT	5 -> 10 5 -> 2
65-68	CLOCK	5 -> 10 5 -> 2
69-76	RECEIVE	10 -> 5
77-78	CLOCK	10 -> 5
79-86	RECEIVE	2 -> 5
87-88	CLOCK	2 -> 5

SIDE B

PIN#	FUNCTION
74	STATUS OUT
75	SYNC OUT
76	SYNC IN
77	SYSTEM CLOCK
78	HALT
79	RESET
29-61-93	+VCC
32-64-96	GROUND

BOARD NO C - (PROCESSOR 6 7 8)

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA IN	6 7 8
9-14	CLOCK	6 7 8
15-22	DATA OUT	6 7 8
23-28	DATA OUT ENABLE	6 7 8
29-36	TRANSMIT	6 -> 11 6 -> 1
37-40	CLOCK	6 -> 11 6 -> 1
41-48	RECEIVE	11 -> 6
49-50	CLOCK	11 -> 6
51-58	RECEIVE	1 -> 6
59-60	CLOCK	1 -> 6
61-68	TRANSMIT	7 -> 12 7 -> 2 7 -> 10
69-74	CLOCK	7 -> 12 7 -> 2 7 -> 10
75-52	RECEIVE	12 -> 7

83-84	CLOCK	12 -> 7
85-92	RECEIVE	2 -> 7
93-94	CLOCK	2 -> 7

SIDE B

PIN#	FUNCTION	PROCESSOR#
1-8	RECEIVE	10 -> 7
9-10	CLOCK	10 -> 7
11-18	TRANSMIT	8 -> 13 8 -> 3 8 -> 9
19-24	CLOCK	8 -> 13 8 -> 3 8 -> 9
25-32	RECEIVE	13 -> 8
33-34	CLOCK	13 -> 8
35-42	RECEIVE	3 -> 8
43-44	CLOCK	3 -> 8
45-52	RECEIVE	9 -> 8
53-54	CLOCK	9 -> 8
74	STATUS OUT	
75	SYNC OUT	
76	SYNC IN	
77	SYSTEM CLOCK	
78	HALT	
79	RESET	
29-61-93	+VCC	
32-64-96	GROUND	

BOARD NO D - (PROCESSOR 9 10 17)

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA IN	9 10
9-12	CLOCK	9 10
13-20	DATA OUT	9 10
21-24	OUTPUT ENABLE	9 10
25-32	TRANSMIT	9 -> 14 9 -> 4 9 -> 8
33-38	CLOCK	9 -> 14 9 -> 4 9 -> 8
39-46	RECEIVE	14 -> 9
47-48	CLOCK	14 -> 9
49-56	RECEIVE	4 -> 9
57-58	CLOCK	4 -> 9
59-66	RECEIVE	8 -> 9
67-68	CLOCK	8 -> 9
69-76	TRANSMIT	10 -> 15 10 -> 5 10 -> 7
77-82	CLOCK	10 -> 15 10 -> 5 10 -> 7
83-90	RECEIVE	15 -> 10
91-92	CLOCK	15 -> 10

SIDE B

PIN#	FUNCTION	PROCESSOR#
1-8	RECEIVE	5 -> 10
9-10	CLOCK	5 -> 10

11-18	RECEIVE	7 -> 10
19-20	CLOCK	7 -> 10
74	STATUS OUT	
75	SYNC OUT	
76	SYNC IN	
77	SYSTEM CLOCK	
78	HALT	
79	RESET	
29-61-93	+VCC	
32-64-96	GROUND	

BOARD NO E - (PROCESSOR 11 12 13)

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA IN	11 12 13
9-14	CLOCK	11 12 13
15-22	DATA OUT	11 12 13
23-28	OUTPUT ENABLE	11 12 13
29-36	TRANSMIT	11 -> 6
37-38	CLOCK	11 -> 6
39-46	RECEIVE	6 -> 11
47-48	CLOCK	6 -> 11
49-56	TRANSMIT	12 -> 7 12 -> 15
57-60	CLOCK	12 -> 7 12 -> 15
61-68	RECEIVE	7 -> 12
69-70	CLOCK	7 -> 12
71-78	RECEIVE	15 -> 12
79-80	CLOCK	15 -> 12
81-88	TRANSMIT	13 -> 8 13 -> 14
89-92	CLOCK	13 -> 8 13 -> 14

SIDE B

PIN#	FUNCTION	PROCESSOR#
1-8	RECEIVE	8 -> 13
9-10	CLOCK	8 -> 13
11-18	RECEIVE	14 -> 13
19-20	CLOCK	14 -> 13
74	STATUS OUT	
75	SYNC OUT	
76	SYNC IN	
77	SYSTEM CLOCK	
78	HALT	
79	RESET	
29-61-93	+VCC	
32-64-96	GROUND	

BOARD NO F - (PROCESSOR 14 15 18)

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA IN	14 15
9-12	CLOCK	14 15
13-20	DATA OUT	14 15
21-24	OUTPUT ENABLE	14 15
25-32	TRANSMIT	14 -> 9 14 -> 13
33-36	CLOCK	14 -> 9 14 -> 13
37-44	RECEIVE	9 -> 14
45-46	CLOCK	9 -> 14
47-54	RECEIVE	13 -> 14
55-56	CLOCK	13 -> 14
57-64	TRANSMIT	15 -> 10 15 -> 12
65-68	CLOCK	15 -> 10 15 -> 12
69-76	RECEIVE	10 -> 15
77-78	CLOCK	10 -> 15
79-86	RECEIVE	12 -> 15
87-88	CLOCK	12 -> 15

SIDE B

PIN#	FUNCTION
74	STATUS OUT
75	SYNC OUT
76	SYNC IN
77	SYSTEM CLOCK
29-61-93	+VCC
32-64-96	GROUND

:::

CONTROL BOARD

SIDE A

PIN#	FUNCTION	PROCESSOR#
1-8	DATA OUT	
17-18	CLOCK	1
19-20	CLOCK	4
21-22	CLOCK	7
23-24	CLOCK	10
25-26	CLOCK	13
27-28	CLOCK	6
29-30	CLOCK	9
31-32	CLOCK	12
33-34	CLOCK	15
35-36	CLOCK	3
37-38	CLOCK	11

39-40	CLOCK	14
41-42	CLOCK	2
43-44	CLOCK	5
45-46	CLOCK	8
47-54	DATA IN	
63-64	OUTPUT ENABLE	1
65-66	OUTPUT ENABLE	7
67-68	OUTPUT ENABLE	13
69-70	OUTPUT ENABLE	4
71-72	OUTPUT ENABLE	10
73-74	OUTPUT ENABLE	11
75-76	OUTPUT ENABLE	2
77-78	OUTPUT ENABLE	8
79-80	OUTPUT ENABLE	14
81-82	OUTPUT ENABLE	5
83-84	OUTPUT ENABLE	6
85-86	OUTPUT ENABLE	12
87-88	OUTPUT ENABLE	3
89-90	OUTPUT ENABLE	9
91-92	OUTPUT ENABLE	15

SIDE B

PIN#	FUNCTION
1-6	STATUS IN
7-12	SYNC IN
13	SYNC OUT
14-19	SYSTEM CLOCK OUTPUT TO PROCESSORS
20	RESET TO OTHER BOARDS
21	HALT TO OTHER BOARDS
27	-9V FOR RS-232 RECEIVER
29-61-93	+VCC 5V POWER FOR ALL BOARDS
32-64-96	GROUND