SGP MEMO NO. 5

VLA COMPUTER MEMORANDUM NO. 158

GRIDDER SYSTEM

W. N. Brouw

September 15, 1981

Part 1 - Array Processor Programs

1. INTRODUCTION

The array processor (AP) package consists of 3 programs:

TRAL: Data selection, gain corrections, gridding. (Tralie=grid in Dutch)
F1FT: First phase FFT, output of gridded U,V-plane
F2FT: Second phase FFT, output of maps.

Visibility data is input into TRAL (i.e., AP1), sent via an IOP-link to F1FT (i.e., AP2) and sent to the transpose memory (TM). F2FT (i.e., AP3) reads the data from the transpose memory and outputs the maps to the user.

The package will work in a 2-AP environment, in which case TRAL and F1FT run in the same AP, and send data to TM. The second AP will run F2FT. Since F2FT runs after F1FT, it could, in principle, run again in the same AP.

To accomodate the 2/3 AP difference, the actual routine names are:

for the 3-AP case: TRAL2
F1FT2
F2FT,
for the 2-AP case: TRAL1
F1FT1
F2FT.

All AP subroutines start with 2 initial letters:

TR: TRAL
TV: TRAL-acting on "raw" visibility records
F1: F1FT
F2: F2FT

All AP subroutines have the extension .APS, and reside in [302,20]. They can be compiled by: APA @name, which produces a [302,21]name .APO object module. An AP-load module can be generated by:

```
@TR2:  TRAL2.BIN and TRAL2.FTN in [302,21]
@TR1:  TRAL1
@F12:  F1FT2
@F11:  F1FT1
@F21:  F2FT
```

In addition 2 libraries of general routines exist: QFFT.APS and QLIB.APS, which can be compiled by APA @QFFT, resp. APA @QLIB.

Each program is steered by data in the AP-memory. This memory area should be filled before the program is run, and should also be formatted for efficient use. The formatting routines are separate entry points in the 3 load modules, and are:

```
TRPM:  TRAL general memory area
TRPV:  TRAL visibility selection area
TVPM:  TRAL gain lists
F1PM:  F1FT general memory area
F2PM:  F2FT general memory area.
```

The content of the various areas is described in Appendix A.

2. TRAL

Before running TRAL, the following steps should be taken:

a. Reserve 64-word work area in AP memory at DPAPTR.
b. Write general memory area (see Appendix A) into AP-memory at MEMPTR.
c. CALL TRPM (DPAPTR,MEMPTR,CONVSC,TAPSC,GAINPTR)

```
where: CONVSC=log2 (convolution function points per grid point)
       TAPSC =log2 (taper function points per grid point)
       GAINPTR=address in AP-memory of gain memory area
```

The convolution and taper functions should have a power of 2 number of points between grid points.

NOTE: All memory areas and function tables must reside in page 0 of the AP memory.

d. Write value and map tables (see App. A) into AP-memory at VALPTR,resp. MAPPTR.
e. CALL TRPV(DPAPTR, MAPPTR, VALPTR, VALLEN, BUFLEN)

```
where: VALLEN=length of one entry in value table
       BUFLEN=length of one input buffer.
```

f. Write gain tables and gain memory area into AP
g. CALL TVPM (DPAPTR, BPCORSC, XINXPTR, XINXLEN, GAINSC)

where: BPCORSC=27-log2 (number of real points in bandpass correction table per correlator) (for line only).
XINXPTR=pointer to .INX overflow area (see later)
XINXL=length of .INX overflow area
GAINSC=27-log2 (number of real points in gain correction table per antenna).

TRPM should be called only once per run. TRPV and TVPM can be called as often as a change in the tables is necessary. In practice TRPV and TVPM will also be called only once per run.

The program is started by:

CALL TRAL2 (DPAPTR, TYPE, WEIGHT, INXTYP)

or

CALL TRAL1 (DPAPTR, TYPE, WEIGHT, INXTYP)

```
where: TYPE= 1   :Produce Antenna pattern
            = 2   :Produce Map
            = 3   :Produce Map and Antenna pattern
       WEIGHT= 0  :Natural weight
            = 1   :Uniform weight
       INXTYP= 1  :2-IF continuum (A,C)
            = 2   :4-IF continuum (A, C, B, D)
            = 3   :1-IF line (AA, CC, BB, or DD)
            = 4   :2-IF line (AA, CC, or BB, DD)
            = 5   :line polarization
```

TRAL starts off by suppressing parity error detection. Although this makes the running of APTEST on a regular basis necessary, it turns out that the AP generates parity error detections (although no actual parity error) if data transfers to/from the AP are done simultaneously with the AP running. It then calls TRPR, which initializes the memory management, and sets switches in various routines.

TRAL then waits for input.

Input is sent to the AP via two alternating buffers. The format of a buffer depends on the last word in the buffer. The low order 16-bits of this last word determine the action to be taken by the AP, and can be:

- -1: Use data to generate a (weighted) point count for the uniform weighting case.
- -2: Convert part of point count into actual weights.
- +1: Convolve data onto a grid
- +2: Send parts of the convolved data to F1FT.
- +3: Stop AP

The remainder of the buffer contains:

if -1, or +1: A series of visibility records (new format). The end is indicated by a visibility record with U=V=0.

if -2, or +2: A real value in the last but one buffer word indicating how far the process can continue. This value is given as U (grid points).

It is assumed that data is input with a decreasing value of |U| for each successive pigeonhole.

As an example, let us assume that we are sending pigeonholes p_i ($i=0,\ldots,n$) with $U_{i,max}$, and $U_{i,min}$, as its |U| boundaries. We want to have uniform weighting with a width W, and a correlation function with a width C, all in grid points.

In that case we send type -1 data for pigeonholes p_i ($i=o,\ldots,k$) until $U_{k,max}+W/2<U_{0,min}-0.5$; then a type -2 with $U_{k,min}+W/2$; then type +1 data for pigeonhole p_0; then a type +2 with $U_{0,min}+C/2$; etc.

TRAL1 will stop after each +2 type to let the user run FIFT; TRAL2 will stop after a type +2 with a negative value (i.e., the last one).

Handshaking between AP and user is accomplished by:

User-to-AP: By writing code in last word of buffer (AP clears this location after processing buffer).

AP-to-User: A CTL5 interrupt is generated after processing each buffer, and a count of the processed buffers is maintained in the AP-LITES' register.

A filled buffer will initiate the following processing:

Type=-1: TVUR: Unravel visibility record into form more usable. Check U and V (unrotated) against limits; skip record if outside limits; rotate U,V; select INX data from time table; output U, V, W (nsec), weight and flags in output buffer; back to TRAL if buffer empty; else:

TVSL: Check time intervals; skip record if outside limits; include INX flags in data flags; include correlator flags in data flags; undo flags to be bypassed.

TVGN: Update output record ptr to include data.

The visibility record buffer now has records:

0: U (nsec)

1: V (nsec)
2: W (nsec)
3: weight (average time)
4: flags (floated, but definition as in input visibility)
5: data
...

ended with a record with U=V=0.

This buffer is used as input to:

TRCBW: For each record, calls:
TRCVW: For each value table entry: Convert U, V, W in grid points; determine position weight buffer; test skip flag; determine pointers to weighting function: checks U, V, W in buffer-limits; calls TRCVLW once or twice (if wrap around of buffer).
TRCVLW: (in TRCVL module) Add weighted point count to weight buffer.

After processing visibility buffer:

TRAL: Clears buffer type; counts buffer in LITES register; interrupts user; checks and waits for more.

Type=-2: TRMW: From current WUMAX (maximum U-line processed) by step -1 until end value specified in input buffer: convert values in weight buffer into a weight by taking the inverse; update pointers describing the extent of the weight buffer; wrap around if necessary. This is done for each map specified in map list. NOTE: The uniform weight is assumed to be the same for all W-planes. Return to TRAL.

Type=+1: TVUR: see above (Type=-1)
TVSL: see above
TVGN: Determine pair of gain tables from time; interpolate to get correct gains; apply interpolated gain; apply gain corrections from bandpass table.

The output buffer is now used as input to:

TRCB1: 2-AP system, or:
TRCB2: For each visibility record, call TRCV. In addition, TRCB2 tests if an IOP transfer is finished, and tries to start a new transfer a synchronously.

TRCV: For each value table entry: converts U, V, W into grid points; determines correct convolution buffer for W-plane; test flags to skip data; if only antenna pattern wanted: call TRCN; else: use value table to construct complex data point (e.g. I, Q, U, V) from input data; call TRSH.

TRSH: Shift complex data point over (l, m, n) given in value table entry; goto TRSB.

TRSB: Subtract sources (l, m, n) given in corresponding map table entry from data point; goto TRCN.

TRCN: Determine position in appropriate convolution buffer; determine weight * taper function * data; get correct convolution function buffer pointers; call TRCVL once or twice (if wrap around necessary).

TRCVL: Do convolution.

Type=+2: TRWB1: 2-AP-system: save end value given in buffer

TRWB2: 3-AP-system: save end value; for each map, and each W-plane per map, get a line from convolution buffer (call TRUB); initialize IOP-transfer; return to TRAL. NOTE:TRCB and TRAL will continue transfer if more needed.

TRUB: Get lines from convolution buffer into transfer buffer; clear line in convolution buffer (and weight buffer if uniform weight); update pointers to these buffers.

TRAL: Clear buffer type; interrupt user; count LITES' register; wait for more if end value was >0; else: wait for IOP transfer and stop.

2.1. TRAL Timing

Timing is difficult to give, depending on a lot of parameters. However, some approximate times for reasonable assumptions about buffer sizes, gives, for N visibility records, M mapsize, S simultaneous maps, c point convolutions, w point uniform weight.

Natural weight, 2-IF continuum:

50N+17NS+5M+M**2+4NS**2+1.3NS(c+c**2) micro sec.

or, for 12 hours, 10 sec integration, 1 map, 4x4 convolution function; 28 antennas, 2048 mapsize: 160 sec.

4-IF continuum: 195sec (add 22N)

2-IF line,8 channels: 225 sec (add 40N)

Uniform weight, 2 IF continuum:

53N+24NS+10M+2M**2+5NS**2+1.3NS(c+c**2)+NS(w+w**2) micro sec

or, for the above parameters: 220 sec

4-IF continuum: 260 sec (add 11N)
2-IF lie, 8 channels: 275 sec (add 32N)

Times will be less for less filled U, V-planes, simultaneous maps etc. Note that gain corrections are responsible for about 30% of the times. Furthermore, the actual run times depend greatly on transfer time of data to the AP.

3. F1FT

Before running F1FT, the following steps should be taken:

a. Reserve 64 -word work area in AP memory at DPAPTR.
b. Write general memory area (see App. A) into AP-memory at MEMPTR.
c. Write map tables (see App. A) with AP memory at MAPPTR.
d. CALL F1PM (DPAPTR, MEMPTR, MAPPTR).

NOTE: For the 2-AP system DPAPTR should be the same as for TRAL. F1PM should be called only once per run.

The program is started by:

CALL F1FT2 (DPAPTR, TYPE, WEIGHT, RDTM, NPART, PART, OUTSK, BITRV)

or, CALL F1FT1 (...) for the 2 AP case.

where: TYPE, WEIGHT:	see TRAL
RDTM=0:	Write data to transpose memory (TM)
=1:	Add data to TM
NPART :	number of parts in which final output map is output (0, or power of 2)
PART :	Current part number
OUTSK=0:	No output to TM
=1:	Output to TM
BITRV=0:	Straight addressing of TM
=32:	Bit reversed addressing of TM

F1FT starts off by suppressing parity error detection, and calls F1Pp, initializing the memory management, and setting switches in various routines. F1FT uses a set of 1 to 5 buffers to do its work; each buffer has its "progress" indicator. F1FT calls a list of subroutines to see if thing can be done, giving preference to filling the buffers from AP1, or writing them to the TM.

The logical order of routines called per buffer is:

F1RB2 (or F1RB1)	:Start read from AP1
F1FR2 (or F1FR1)	:Finish read from Ap1
F1SB	:Subtract sources
F1OB2 (or F1OB1)	:Output gridded data to user
F1TR	:Fourier transform data
F1WB	:Start output to TM
F1FW	:Finish output to TM

Each buffer is governed by a DPX/DPY indicator pair, containing:

DPX:	Exponent	:	512 + buffer type
	High mantissa	:	W-coordinate of data (Ø,1,...)
	Low mantissa	:	U-cooddinate of data (Ø,1,...)
DPY:	Exponent	:	Map number (Ø,1,...)
	Low mantissa	:	Start of data buffer of length: TYPE (Ø, 1, or 2) * VTLEN (length of full V-line)

The possible buffer types are:

-1: Buffer not present
Ø: Available for read
1: Read active
2: Antenna pattern read ready (for TYPE=3)
3: Map being read (for TYPE=3)
4: All reading finished
5: Source subtraction finished
6: Output to 11/44 finished
7: Antenna pattern transform finished (for TYPE=3)
8: All transforms finished
9: write to TM active
10: Read from TM active (if RDTM=1)
11: Antenna pattern written (for TYPE=3)
12: Map being written to TM (for TYPE=3)
13: Map being read from TM (for RDTM=1 or TYPE=3)

The routines:

F1RB2 calls F1FM with parameter 2 to check if a TYPE=3 map should be read; if not, it calls F1FM with Ø to find an empty buffer. If an empty buffer is found, it determines the W-, MAP- and U-coordinates of the next data, fills in the buffer description and starts a notification sequence to AP1.

F1RB1 acts as F1RB2, however, the actual read is suppressed, since the data is already in the one buffer allowed.

F1FR2 (in module F1RB2) checks if there is an IOP interrupt from AP1. It determines the cause of the interrupt (handshaking or data transfer) and either initiates a data transfer or handshaking.

F1FR1 (in module F1RB1) sets reading finished.

F1SB calls F1FM to find a buffer ready for subtraction. If one available, it obtains the source list from the current map area, and subtracts the sources from the data if TYPE=3, i.e., both the antenna pattern and map have to be present.

F1OB2 checks if buffer available and output to 11/44 wanted. In that case it checks if the output buffer (one of an alternating pair) is available (i.e., first word of buffer is zero), and transfers data to this output buffer. The output buffer is then scaled to 16-bit integers, and the 11/44 is notified by an interrupt, and the buffer count.

The first 6 words of the output buffer contain (16-bit integers):

```
0:  0:  Buffer empty
    1:  Buffer filled
1:   :  Maximum value in line
2:   :  Position of maximum value (last point=0)
3:   :  minimum value in line
4:   :  Position of minimum value (last point=0)
5:   :  Scale (i.e., power of 2 to multiply data with)
```

F1OB1 is a NOP routine, output is not allowed.

F1TR calls F1FM to check if anything is to be done. It then expands the part of the V-line input from AP1 by adding zeroes at beginning and end and transforms via QSRFFT (antenna pattern) or QSFFT (map). Both routines are in OFFT. They use the same algorithm as the FPs routines, but the output order is bit-reversed.

F1WB checks if anything to be done via F1FM, and if output to TM wanted. It then initiates either a read from (RDTM=1) or write to (RDTM=0) the TM.

F1FW (in module F1WB) checks if there is a TM interrupt; then checks which phase it is in (reading, writing, antenna pattern, map) and either initiates a further read or write, or declares the buffer free.

NOTE: Output to TM is done in 24 bit integers scaled per line. The scale is output as a 10-bit integer in the low order 5-bits of the first two words of a line.

.1. F1FT Timing

The actual program timing is roughly for a mapsize M, with W W-planes:

```
Antenna pattern:        W(120+2M+(2+.2M)2log(M) micro sec.
Map:                    W(100+3M+(2+.45M)2log(M) micro sec.
```

or, about 12 seconds for a 2k antenna pattern
20 seconds for a 2k map.

However, transfer times from AP1 and to TM play a role as well. IOP transfer rates are about 1 micro sec per word, and are concurrent with calculations.

TM transfer times are, as far as I know, about 15 micro sec per word, resulting in about 60 sec for a 2k map. If the gridding is split into several parts n, this amounts to (2n-1)*60 sec.

4. F2FT

Before running F2FT, the following steps should be taken:
- a. Reserve 64-word work area in AP memory at DPAPTR.
- b. Write general memory area (see App. A) into AP memory at MEMPTR
- c. Write map tables (see App. A) into AP memory at MAPPTR.
- d. Call F2PM (DPAPTR, MEMPTR, MAPPTR, WCNVSC)

where WCNVSC: Scale (=number of table points per grid point) of W-convolution correction function.

NOTE: Although the program for this correction is present, I have no clear idea on how to generate this function.

F2PM is called once per run.

The program is started by:

```
CALL F2FT (DPAPTR, TYPE, ABTYP, FPAP, NOCC, BITRV)
```

where:

```
TYPE, ABTYP:See TRAL
FPAP=0      :Full antenna pattern in TM
    =1      :Part of antenna pattern in TM
NOCC=0      :No convolution function correction
    =1      :Convolution function correction
BITRV=0     :Address TM straight
     =32    :Bit-reversed addressing of TM
```

F2FT starts off by suppressing parity error detection (see TRAL ...), and calls F2PR, initializing the memory management and setting tches in the routines called.

F2FT uses a set of from 1 to 4 buffers, arranged and used in the same way as in F1FT. The buffer description in this case is:

DPX:	Exponent	:512 + buffer type
	High mantissa	:W-plane (0, 1, ...)
	Low mantissa	:V-coordinate (0, 1, ...)
DPY:	Exponent	:512 + map number (0, 1, ...)
	Low mantissa	:Address of buffer of length 2* U-dimension

The buffer types are:

-1:	Buffer not present
0:	Buffer empty
1:	Being filled
2:	Filled
3:	Fourier transform ready
4:	W-plane handling active
5:	W-planes ready
6:	Correction for convolution function done
7:	Read TM only for scales

F2FT, as F1FT, calls a set of routines to act on the buffers, giving ority to I/O operations. The routines called, in their logical order are:

F2FB checks if empty buffer available, determine map number, W-plane, V-coordinate to be read. Only the V's necessary for the output size are read, except the first one, which has always to be read for the scales. A TM read is initiated.

F2FR checks if a TM interrupt is present, and finds the buffer being read. If the line being read contains the scales they are saved in a separate scale buffer. The line is then scaled and expanded by inserting zeroes. (Note that input order is bit-reversed.)

F2TR Fourier transforms the data, using QS2FFT (in QFFT).

F2AW checks by calling F2FM if anything to be done. If only 1 W-plane, the real part of the buffer is taken. If there is more than 1 W-plane, the W-planes are all added with the appropriate phase correction in a separate buffer. This buffer is transferred to a standard buffer as soon as the last W-plane is read.

F2CC checks if anything to be done, and then corrects the data for the convolution function.

F2WB checks if anything to be done. It transfers the data to one of an alternating pair of output buffers, if available, scales the line to 16-bit integers, and notifies the 11/44 with an interrupt and buffer count. The format of the output buffer is as for F10B.

4.1. F2FT Timing

Times are approximately for a map of size M, with w W-planes:

M.W(65+20M+(2+.45M).2log(M)) micro sec.

or

100 sec for a full 2K map.

The time will, however, be dominated by the data transfer from the TM and to the 11/44.

Appendix A: Data Formats

A.1. TRAL

A.1.1 General Memory Area of 87 Words

The area consists of 5 parts, all containing floating point numbers. If no description given, a zero should be put into it.

Prolodue:(Ø):

Ø:	CBUF	AP address of start of convolution buffers.
1:	CCNT	Length of convolution buffers.
2:	BUF1	AP address of first input buffer
3:	BUFLEN	Length of one input buffer
4:	WMAX	Maximum W-coordinate (=# of W-planes/2)
5:	WUBLEN	Length full uniform weight buffer.
6:	WLEN	Length full convolution buffer for one W-plane.
7:	ULEN	Length one convolution line (=TYPE*V-length)
8:	WULEN	Length one uniform weight line (=V-length+ convolution width)

W1PT (9):

1:	WULEN	
6:	WULEN	
7:	WOFF	Offset in uniform weight line to V=Ø(=Vmin -FLOOR(.5*conv.width))
8:	WUMAX	Maximum U in uniform weight buffer
10:	WUBLEN	
12:	WUTPBT	# of simultaneous lines in uniform weight buffer.
13:	WUBOT	U at bottom of weight buffer (=WUMAX-WUTPBT)

W2PT (27):

Ø:	VLEN	Length of one V-line
1:	WULEN	
3:	NW	# of W-planes
5:	OFF	Offset to V=Ø (=TYPE*V_{min})
7:	WCOFF	Offset in uniform weight buffer to V=Ø, U=U_{max} (=(WUMAX-U_{max})*WULEN+WOFF)
8:	UMAX	U_{max}
10:	ULEN	Length one convolution line (=TYPE*VLEN)
12:	WLEN	
15:	UTPBT	# of simultaneous lines in convolution buffer.
16:	WULEN	
20:	ULEN	
21:	WUBLEN	
22:	APBUF	AP address of output buffer for antenna

```
                         pattern (length=VLEN, or 0 if TYPE=2)
      23:      MAPBUF    AP address of output buffer for map.
                         (length =2*VLEN, or 0 if TYPE=1).
      24:      WMOFF     Offset to W    from W=o(=WLEN*(NW-1)/2)
                                    min
      26:      VOFF      Offset to V=0 (=V    *TYPE).
                                          min
      27:      WOFF

MAD (55):
       0:      WUBOT
       1:      UHB       High checking boundary convolution (=U    -
                                                                max
                         NCU/2.+1.)
       3:      UMAX
       4:      VLB       (=V   n+NCV/2.-1.)
                            min
       5:      ULEN
       6:      WLEN
       7:      CVMIN     V
                          min
       8:      CUPTR     AP address of U-convolution function table for
                         U=0.
       9:      TVPTR     AP address of V-taper function table for V=0.
      10:      TUPTR     AP address of U-taper function table for U=0.
      11:      CVPTR     AP address of V-convolution function table
                         for V=0
      12:      VHB       (=V    -NCV/2.+1.)
                            max
      13:      VMAX      V
                          max
      14:      ULB       UMAX+NCU/2.
      15:      UTPBT

WMAD (72):
      -1:      WNU2      Half width uniform weight function in U-
                         direction (nsec).
       0:      WNV2      Half width uniform weight function in V-
                         direction (nsec).
       1:      WVLB      (=WVMIN-1) (WVMIN=V    -FLOOR(NCV/2.))
                                            min
       2:      WUHB      (=WUMAX+1)
       4:      WVMIN
       5:      WUMAX
       6:      WULEN
       7:      WUVSC     # of points in weight function per nsec.
       8:      WUVSC
       9:      WUMAX
      10:      WVPTR     AP address of weight function table for V=0
      11:      WUPTR     AP address of weight function table for U=0
      12:      WVHB      (=WVMAX+1) (WVMAX=WVMIN+WULEN-1)
```

13:	WUTPBT
14:	WVMAX

A.1.2 Gain Correction Memory Area of 54 Words (not specified values should be zero)

0:	8	
1:	SINROT	SIN (Map rotation angle)
2:	COSROT	COS (Map rotation angle)
3:	VLIML	(Low limit to be included in V) **2.
4:	ULIMH	(high U-limit)**2.
5:	ULIML	(low U-limit)**2.
12:	VLIMH	(high V-limit)**2.
14:	BUFDLN	# of read data points per visibility record.
16:	RLIMH	(high (U**2+V**2)**.5)**2.
19:	RLIML	
21:	31	
23:	63	
25:	INXBUF	AP address of gain table distribution table.
26:	TIMLIM	AP address of time limit table.
27:	CORFLP	AP address of correlator flag table.
29:	BIPFL	MOD (3-bypass flag,4).
34:	LCORP	AP address of bandpass correction table.

.3 Value descriptions

The action on the input data is described in a list of value descriptors. Each descriptor has a length VALLEN.VALLEN=29+# of data descriptors, and is defined by user.

The format is:

0:	=0:	End of descriptor list.
	<>0:	AP address of map area describing map to be produced from output data.
1:	FREQ* W-field width (GHz* radians)	
2:	FREQ* U-field width (GHz* radians)	
3:	FREQ* V-field width (GHz* radians)	
4:	(FREQ *U-field width)**-1	
5:	(FREQ *V-field width)**-1	
6:	weight (multiplicative) to be used on data.	
7:	Test bits, describing input data points used in forming output data point. The format is the same as for the flag word in the input.	
8:	offset to real part of data point in visibility record + type * 4096.	

Type=	1:	-REAL	+IMAG	are used in convolution
	2:	-	-	
	3:	+	-	

```
        4:      +       +

offset to imaginary part of datapoint in visibility record +
type*4Ø96.

Type=   5:      i*(+REAL +IMAG) are used in convolution
        6:      i*(-     +    )
        7:      i*(-     -    )
        8:      i*(+     -    )

=Ø: No more data usage descriptors.
```

```
9:          next input data usage descriptor
            ......

VALLEN-2Ø:          Ø
      -19:          Ø
            ......

      -4:           L-coordinate of shift of input data in
                    fractions of map width.
      -3:           M-coordinate
      -2:           N-coordinate
      -1:           =Ø:No shift requested.
                    =1:Shift requested
```

A.1.4 Map descriptors

Each output map has a 6-word descriptor

```
Ø:      =Ø:No more map descriptors
        =AP address of convolution buffer for this map for
        U=U   , V=Ø, W=Ø
           max
1:      AP address of uniform weight buffer for this map for
        U=WUMAX, V=Ø
2:      Ø (becomes normalization sum)
3:      =AP address of source list to be subtracted.
        =Ø:no sources to be subtracted.
4:       Ø
5:       Ø
```

A.1.5 Source List

Each source has a 4 word entry:

```
Ø:      =Ø: No more sources
        =intensity of source
1:      L-coordinate in fractions of map width
2:      M-coordinate
```

3: N-coordinate

A.1.6 Gain Tables

The input data are split into several datasets according to SORTER rules (meridian transit date, frequency, sourcename, etc.). At any one time the gain tables for one dataset are in AP core. Each INX record has a 9 word entry, followed by the gain tables for this record. The INX format is 32-bit integers.

0: # of line channels
1: Length of one gain record
2: AP address of gain tables for this INX.
3: Time increment (seconds) between gain records.
4: Date (IAD-45000)
5: Flags channel A
6: Flags channel B
7: Flags channel C
8: Flags channel D

Each gain record starts with 2 words:

0: Time (seconds) of this record
1: Time of next record, or >84000, if next record is last.

followed by the gain tables.

A.1.7 D-Pad usage

D-Pad is the main communication between routines. Its usage:

DPX:
- 0: AP address of MAD memory area
- 1: AP address of F1FT memory area in the 2 AP case.
- 2: AP address of W1PT memory area.
- 3: AP address of gain memory area.

DPY:
- 0: AP address of DPA area.
- 1: AP address of WMAD area.
- 2: AP address of W2PT area.

DPX/Y 4-31 are initially filled by TRPM using the data in the memory area prologue.

In TRMW the W1PT area is exchanged with DPX/Y 5-12.

In TRWB the W2PT area is exchanged with DPX/Y 5-18.

A.2 F1FT

A.2.1 General memory area of 50 words, given in 2 consecutive parts; unspecified words should be 0.

MEMLST(0):

0:	WMIN	Minimum W-coordinate (=(NW-1)/2)
2:	VMIN	V_{min}
4:	VMXMN	$V_{max} - V_{min}$
6:	VTLEN	FFT length V-direction
7:	VMOFF	Offset to V_{min} in line (=VMIN+VTLEN/2).
8:	6.	
11:	NW	# of W-planes
13:	UMAX	U_{max}
14:	-1	
15:	VOTLEN	Output length in V-direction (even,<=VTLEN).
16:	UMIN	U_{min} (=CEIL(-NCU/2.))
18:	VMXOFF	Offset to V_{max} in line (VTLEN/2-VMAX-1)
20:	OBUF1	AP address of output buffer to 11/44(or 0)
21:	OBUF2	AP address of output buffer to 11/44(or 0)
24:	=0	This buffer not present
	=1	This buffer present
25:	BUF1	AP address of buffer
26-33:		As 24, 25 for BUF2,...,BUF5

XMEMLST (34)

0:	NMWID	Bits in map count (=15.-nearest power of 2 (# of maps))
1:	NMEXP	TM address shift for map number +27.
2:	NWWID	Bits in W-plane (=15.-nearest power of 2 (NW))
3:	NWEXP	TM address shift for W-plane # +27.
4:	NUMSK	(=2**(nearest power of 2(ULEN))-1)
5:	NUEXP	TM address shift for U +27.
7:	INC64	64 * INCAD
9:	INCAD	TM address increment between V-points.
11:	RBFAD	65 word buffer area for TM reads

A.2.2 Map descriptors

Each output map has a 6 word decriptor:

0:	=0	No more maps
	=1	Valid entry
1:	0	
2:	0	
3:	=0	No source subtraction

```
              =          AP address of subtraction source list (see
                         A1.5 for format of list)
        4:         0
        5:         0
```

A.2.3 TM addressing

TM addresses have two formats, based on ABTYP.

	ABTYP=0:	=1:
bit0:	Real/Imaginary	Real/Imaginary
1-a:	Map #	V-coordinate
a-b:	V-coordinate	Map #
b-c:	W-coordinate	W-coordinate
c-d:	U-coordinate	U-coordinate

The values of a, b, c, and d depend on the length of each field.

A.2.4 D-Pad usage

```
DPX:     0:        AP address of XMEMLST area
DPY:     0:        AP address of DPA area
DPX/Y 7-23         are filled from MEMLST area
```

In F1WB DPX/Y 24-31 are exchanged with XMEMLST.

A.3 F2FT

A.3.1 General Memory Area of 64 Words (Unspecified entries should be zero)

MEMLST (0):

4:	VLEN	Length of V-line (=MMAX-MMIN+1)
5:	NW	# of W-planes
7:	NMAP	# of output maps
8:	MMIN	M_{min}
9:		VLEN/2-1
10:	U2LEN	2*output length in U-direction (=2*UOTLN)
11:	NMEXP	TM address (in bits) of map # +27
13:	NWEXP	TM address (in bits) of W-coordinate +27
15:	NVEXP	TM address (in bits) of V-coordinate +27
16:	VOFMIN	Offset to V_{min} in full line
17:	VOFMAX	Offset to V in full line

max

18:	LOTLEN	Output length in L-coordinate (even)
19:	LCPTR	AP address of L convolution correction function for L=0
20:	MMIN	
21:	MCPTR	AP address of M convolution correction function for M=0
24:	OBWF1	AP address of first output buffer to 11/44
25:	OBWF2	
30:	WSCBWF	AP address of list to scale buffer for W-planes
31:	MSCBUF	AP address of list pointing to scale buffers for maps
32:	UILEN	Length of one input U-line
33:	SPUMI	(=UILEN+U_{min})
38:	VTLEN2	LOTLEN/2
39:	WTLEN	U-direction FFT length
40:	UMAX	
41:	UTDUIM	2*(UTLEN/UILEN-1)
42:	VMIN	
43:	NW	
44:	WMIN	
45:	LOTOFF	Offset to start of output (=UTLEN-LOTLEN)
46:	LOTLEN	
47:	WBUF	AP address of W-plane addition buffer.
49:	MFACT	(=0.5*(MFIELD/VTLEN)**2/NFIELD)
50:	LMIN	(=-LOTLEN/2)
51:	WCPTR	AP address of N convolution correction function
52:	LFACT	(=0.5*(LFIELD/UTLEN)**2)/NFIELD)
56:	=0	Buffer 1 not present
	=1	Buffer 1 present
57:	BUF1	AP address of Buffer 1
58-63:		As 56, 57 for BUF2, 3, 4.

A.3.2 Map descriptions

Each output map has a 6 word descriptor:

0:	=0	No more maps
	=1	Valid entry
1:	0	
2:		Normalization factor
3, 4, 5:	0	

A.2.3 D-Pad usage

DPY:0 DPA area AP address

The remainder of DPX/Y is filled from MEMLST.

Appendix B

On August 10, 1981 the status of the AP-software was:

-All routines tested for sizes up to 2048
-No actual third dimension maps made.
-Probably an error in the flagging handler.
-Multiple input datasets not tested.
-Map output in parts not tested.
-Limited number of possible tables tested.
-No line data tested.