SGP MEMO NO. 6

VLA Computer Memorandum no. 159
GRIDDER SYSTEM
W.N. BROUW
September 15, 1981
Part 2 - PDP 11/44 Programs

1. INTRODUCTION

The GRIDER software has been designed to follow the MAPPER setup as closely as possible. It consists of 8 main programs:

1. PPLCOM - a resident common area
2. GRIDER - communication with DEC10
3. PPMAIN - overall program to produce maps
4. PPSET - set all relevant parameters
5. PPTR - do gridding and other tasks related to AP1
6. PPFT1 - do first part of Fourier transform and all other tasks related to AP2
7. PPFT2 - do second part of Fourier transform and all other tasks related to AP3
8. PPCAT - reformat and catalog output maps.

All of the above programs should be installed, which can be done by:
@[302,1] GRIDER
which will also set the UIC to [302,40], the UIC of the programs.
To start the process:
RUN GRIDER
either from the terminal, or from a program like MAPRCV.
All other programs will be called automatically.

The general logics of the program flow is:

1. PPLCOM - acts as communication area between all programs, and keeps some history.
2. GRIDER - will get information from MAPRCV.CAT, spawns PPMAIN, and will wait for it to finish.
3. PPMAIN - will spawn PPSET to determine map parameters, and wait for it to finish; it then spawns PPTR and PPFT1, wait for them to finish; it then waits for the previous run of PPFT2; starts PPFT2 and stops.
4. PPCAT - is a spooler program. PPFT1 and/or PPFT2 produce output files, which are queued to PPCAT.

2. PROGRAMMING

All source modules and CMD files can be found in [302,20]. All object modules are in [302,21] WNB.OLB.
Each routine has its command file, e.g. PTMEM.FTN is compiled by: @PTMEM.
The command files for the programs GRIDER, PPSET, PPTR, PPFT1, PPFT2 and PPCAT will compile and taskbuild. To just task build them use @GRIDERB, PPSETB etc.
PPMAIN comes in 2 flavors, depending on whether a 2 AP (i.e. gridding and first FFT pass in the same AP) or a 3 AP system is wanted. The routines, and

their commands, are called PPMAN2 and PPMAN3 respectively. They only change in a parameter definition. The main body of their program is included from PPMAIN.DCL.
PPLCOM.DCL defines a lot of parameters used in the programs, and the layout of the resident common area. PPLCOM.FTN defines an initialization program. The resident common area is produced by: @PPLCOM.

If a change is made in PPLCOM.DCL, all routines and programs using it can be rebuild by: @PPLUPD
If a change is made in PPMAIN.DCL run: @PPMAINUPD
If a change is made in the MIX record format, run: @MIXUPD
If a change is made in the INX record format, run: @INXUPD.
If a change is made in PCAT.DCL (see later), run: @PCATUPD.

In addition to the routines described below, some general routines are available in several modules. These modules are:

FILBUF.FTN	to be compiled with	@FILBUF
QMAC.MAC		@QMAC
QFUN.FTN		@QFUN
QAST.MAC		@QAST
QSCALE.MAC		@QSCALE
PPDFIL.FTN		@PPDFIL
GRDLOG.FTN		@GRDLOG
GDTTM.FTN		@GDTTM
PCWZER.FTN		@PCWZER
PCFILN.FTN		@PCFILN

Some other general command files:

@SPOOL	- to print all .LST and .MAP files
@CLUP	- cleans [302,*] areas
@DUP	- duplicates [302,*] areas to DB1:
@CLDUP	- combines CLUP & DUP
@NFTDUP	- duplicates [302,*] areas to GRIDER
@TAPDUP	- dumps [302,*] areas on tape.

Other tasks in [302,20] and [302,40] areas:

MDUMP	- dumps selected blocks of a file (e.g. map) in integer or floating point format. The first block (0) is dumped in 4 formats.
MIX	- allows you to change MIX records
MIX1	- allows you to change map type and size in MY.MIX record

3. AP DEBUGGER:

AP programs can be debugged by including the JTAP(integer) function in your program. If a statement, like:
IF (JTAP(integer).NE.0) STOP is encountered. The debugger is entered, typing at your terminal:

RUNNING	or:	PSA = value
ACTION integer:		ACTION integer:

depending on whether the attached AP is running or stopped.

All AP-registers and PPLCOM can be inspected by commands to ACTIONS:, and all AP-registers can be changed.
Output of JTAP is to YERLUN (defined in PPLCOM), which is assigned to the terminal by the CALL PPASG(0) in each program. Output will be terminal selectable to either terminal or a disk file by changing this call to:

```
CALL PPASG(1,'file name').
```

The debugger commands are (Oi is an octal value, di is a decimal value):

<CR>	:	Return to main program
CO	:	Return to main program
HE	:	Help
SW	:	read SWR
LI	:	read LITES
RS	:	reset AP
ST	:	stop AP
XT	:	exit main program
CM	:	copy input line
ZB	:	empty APEX break point
TD	:	test DMA activity
TR	:	test AP running
TI	:	test AP CTL5 interrupt state
IE	:	enable CTL5 interrupts
ID	:	disable CTL5 interrupts
RP	:	read PSA
RD	:	read DPA
RT	:	read TMA
CC	:	continue AP program from current location
CS	:	execute single AP instruction
SR	:	read STATUS
ME	:	read MAE
MD d1[,d2]	:	dump AP memory from d1 to d2 in floating format
MI d1[,d2]	:	dump AP memory from d1 to d2 in octal format
MV d1,d2	:	set value d1 in AP memory d2
DD d1[,d2]	:	dump D-PAD from d1 to d2
XI d1[,d2]	:	dump DPX in octal from d1 to d2
YI d1[,d2]	:	dump DPY in octal from d1 to d2
SD d1[,d2]	:	dump S-Pad from d1 to d2 in decimal format
SI d1[,d2]	:	dump S-Pad from d1 to d2 in octal format
LD d1	:	set d1 in DPA return 1 + d1 to JTAP
RR d1		function
SO O1	:	
SB O1	:	set APEX breakpoint to O1
PD O1[,O2]	:	dump program memory from O1 to O2
PI O1[,O2]	:	As PD
PS O1,O2,O3,O4	:	set O2,O3,O4 in program memory O1
LP O1	:	load O1 in PSA
LT O1	:	load O1 in TMA
CB O1	:	Continue AP program till breakpoint O1
DA O1[,O2]	:	Read Da from O1 to O2
DS O1,O2	:	Set O2 in DA O1
LM O1	:	Load MA

OD O1[,O2]	:	Dump MP memory from O1 to O2 in float format
OI O1[, O2]	:	Dump MP memory from O1 to O2 in octal format
OS O1,O2,O3,O4	:	Set O2 O3, O4 in MP memory address O1.
EX O1,O2	:	= APEXAM (,O1,O2)
DP O1,O2,O3,O4,O5,O6	:	= APDEP ((O3, O4, O5, O6), O1, O2)
PO O1	:	Use O1 as offset in all input / output of program memory references.
YA O[,O2]	:	type as decimal integer values at offsets O1 to O2 in PPLCOM.
ZA O1[,O2]	:	type as REAL values at offsets O1 to O2 in PPLCOM
SS O1,O2	:	set O2 in S-Pad O1
MS O1,O2,O3,O4	:	set O2, O3, O4 in memory location O1
XS O1,O2	:	set O2 in DPX O1
YS O1, O2	:	Set O2 in DPY O1
XOV O1,O2	:	Set O1 in memory location O2
[302,21] WNB/LB:JTAP	:	in .ODL file for program.

4. GRIDER

GRIDER is called by MAPRCV to start the process. It calls GRDOPN to get an entry from MAPRCV.CAT. If .INX, .GAI and .MIX are found, an entry exists. If also a .VIS file exists, it is assumed later that input data is in the MAPPER format. Else SORTER input is assumed.

If a correct entry is found, GRIDER starts PPMAIN, waits for PPMAIN to finish; and calls GRDOPN for next entry.

A program GTEST exists which basically does the same but assumes all files are called MY.MIX, MY.INX, MY.GAI, MY.VIS instead of getting them from the catalog. Furthermore, the files are not deleted after use, and only one set of MIX records is used.

5. PPMAIN

PPMAIN calls PPINIT, which initializes all AP's if a reboot occurred, and reserves some standard, fixed size area's in all 3 AP's.

PPMAIN then starts PPSET (see 6), which determines all map parameters, and waits for PPSET to finish. It types and logs some information about the maps and its input.

For each pass necessary through the transpose memory (see PPSET for details) it then:

a. For all gridding passes (ie. if the memory in AP1 is too small for one run, see PPSET), and for all data sets, it:

- starts PPFT1
- waits till PPFT1 is properly initialized
- starts PPTR

- waits till PPTR initialized
- waits till PPTR and PPFT1 finish
- adjusts parameters describing gain tables and U, V-plane boundaries.

b. - Waits for PPFT2 to finish
- starts PPFT2
- restart a. for next TM pass.

After all TM passes are ready, it spools the file GRID.TMP to PPCAT if gridded output was requested, and finishes.

NOTE: At 10 August 1981 only 2 AP's are available, and also no mechanism exists to read from/write to the TM, and regulate the traffic. Therefore, PPFT2 runs in AP2, and gridding has to wait for PPFT2 to finish.

6. PPSET

PPSET defines all map making parameters by calling:

IDFMIX - Get parameters from MIX record
IDFINX - Get parameters from INX records
IDFPIG - Get pigeon hole parameters from SORTER
IDFCHK - Check parameters for validity
IDFCTP - Define convolution function, taper function etc.
PSETCW - Calculate and set all function tables
PSTVAL - Define value descriptors
IDFUVW - Define (U, V, W) - solid parameters.
PSTSC1 - Set source subtraction list for AP1 (gridding)
PSTSC2 - Set source sutraction list for AP2 (FFT phase 1)
PSTMP1 - Set map descriptions for AP1
PSTMP2 - Set map descriptions for AP2
IDFGN - Prepare gain tables
PPPREP - Calculate the number of TM and gridding loops necessary with the limited TM and AP memory available.

The more detailed description of the above routines:

IDFMIX: Reads a MIX record, and gets all information about the map to be made (size, convolution etc.). The information is converted and stored in PPLCOM for later use. Extension of the MIX record is wanted to be able to set all necessary information. In Appendix B a description of possible MIX extensions is given. After reading a MIX record, all subsequent MIX records belonging to the same "set" (see App. B) are read and the information stored.

IDFINX: Reads first INX record, and fills in some MIX data (coordinates, frequencies etc.), and fills the VLA.LOG. It then reads all other INX records and:

- tests if same "type" (i.e. INX, GAI, VIS record lengths the same, # or IF's the same).

- skips record if different type

- determines the "data set" to which the INX record belongs, based on SORTER choices (source name, meridian passage date, frequency and more, still to be defined in detail).
- If more than 16 datasets, remaining data is ignored.
- save relevant data about INX record, especially where to find it.
- save dataset # in INX record.

IDFPIG: Defines data to be obtained from SORTER. It sets:
- maximum values of U, V and W in the input data sets
- width of one pigeon hole
- map rotation influence on pigeon hole width

IDFCHK: Check parameters for validity (e.g. map size <= 8192, FFT size power of 2 etc.). If invalid data, replace it with default value, and increase the warning count.
IDFCHK also reserves space for some buffers of which the length is now known.

IDFCTP: Using the data from MIX and INX, IDFCTP calculates the parameters for the definition of all functions used (convolution, uniform weight, taper) in an internal format.

PSETCW: Calculates and sets in AP memory the tables for the uniform weight function, convolution function, taper function, convolution correction function.

PSTVAL: Generates and sets into AP-memory the necessary value descriptors (see Part 1) to generate the appropriate maps (I, Q, U, V, ...)from each dataset.

IDFUVW: Calculates the boundaries of the (U,V,W) - convolution solid needed and the boundaries of the (U,V)-uniform weight solid. It reserves space for the intermediate buffers between AP1 and AP2, and for the gridded output data buffers in AP2.

PSTSC1: Makes list of sources to be subtracted in gridding phase.

PSTSC2: Makes list of sources to be subtracted in FFT phase 1
Note: Dummy routine since no data available.

PSTMP1: Put address of source list in map descriptor for gridding.

PSFMP2: Put address of source list in map descriptor for FFT phase 1 (Dummy).

IDFGN: The action taken is:
- open GAIN.TMP file
- read an INX record and save pertinent data (see part 1) in GAIN.TMP
- make sure the INX record and its gain tables fit on 1 AP page.
- define an extra data set if AP memory full.
- Call PSETGN for all gain records

PSETGN - Reorder gains
- Extrapolate if necessary to make all gain records equally distant in time
- Apply all VIS and GAI scales
- If polarization, apply parallactic angle rotation to gain
- If parallactic angle change >15^{0} between table entries, redefine 2 min time interval. If change still >15^{0}, forget data.

PPPREP - Determine number of passes through TM, based on TM size, simultaneous maps, map output sizes.
- Determine number of gridding passes based on memory available in AP1, and minimum size of convolution buffer, based on (U,V,W)-maxima and the pigeon hole width.

7. PPTR

PPTR will call PTVAL to set the appropriate value and map descriptors, based on the current slice of the (U,V,W) - solid. Then it calls PTMEM, which fills the general memory area (see Part 1), and then PPTR will write the general memory to the AP. AP1 is then initialized via calls to TRPM and TRPV. The correct gain tables are loaded in PTLGN; the gain area is prepared in PTVMEM; transferred to the AP and initialized via a call to TVPM. PPMAIN is now notified that initialization is finished. Forgetting about the 2 AP case, TRAL is now started. In the case of natural weight all data from a pigeon hole is treated until no more data available.

In the case of uniform weight data from a pigeon hole is read. If enough data is available to start the convolution, the weight data is transformed into a factor, and convolution is started with data from the first pigeon hole. Alternately more weight data is read and processed and convolution data is processed, until all data is convolved.

8. PPFT1

PPFT1 first determines the number of work buffers (>= 1, <= 5) available in AP memory. It then generates the map descriptors in P1VAL, and the general memory area in P1MEM. The general memory area is transferred to the AP, and initialized via F1PM. PPMA1N is now notified of a successful inizialization. In the case of 2AP's PPFT1 finishes. In the 3AP case the F1FT program is started, and P1GRD is called. P1GRD returns immediately if no gridded output is asked for. If gridded output wanted, a file GRID.TMP is generated and filled.

9. PPFT2

PPFT2 first determines the number of work buffers available in the AP memory (>=1, <=4). It then creates the map descriptors in P2VAL, and the general memory area in P2MEM. After writing the general memory area to AP memory and initializing it by a call to

F2PM,PPMAIN is notified of successful initialization. F2FT is now started, and P2MAP is called to generate the final output map. After the last part of a map has been obtained, P2MAP spools the output to PPCAT for final formatting and cataloging.

10. PPCAT

PPCAT receives file identification for either output grid data, or output maps.

In the case of gridded data, GRID.TMP is opened, and for each output map and W-plane the correct data is read from GRID.TMP and reformatted in PCGRD, after which each map and W-plane is catalogued by a call to PCCAT. In the case of map data the input map data is rescaled from scaled per line to scaled per map via a call to PCMAP, and then catalogued by a call to PCCAT.

11. ADDITIONS

It is obvious that cleaning and self-calibration has to be added to GRIDER, if possible making use of more then 1 AP and the transpose memory. The usage of the transpose memory in these cases is, however,probably limited, due to, at least at present, an access time of about 15 micro sec per word. Since GRIDER produces essentially the same output data as MAPPER, it will be very easy to incorporate the present CLEAN and SELFCAL in the system.
For intermediate use a program PPSORT is added, called by PPMAIN, to sort old DEC 10 visibility files.

Appendix A - Proposed MIX changes

To use all the capabilities of GRIDER extensions to the MIX record have to be made. Space for these extensions can easily be found by trimming existing entries (e.g. bringing back the existing ANTENNAS from 28 words to 28 bits).

Proposed extensions include:

-MSET: A "set" indicator. Since GRIDER can produce similtaneous maps from one input database (e.g. I, Q, U) it would be wasteful not to be able to do it (e.g. 4 maps of 1k can be handled in one run through the input data, greatly reducing the I/O time). In addition, producing COVER etc. can be done at the same time. To be able to use this feature with the existing MIX structure, all MIX records differing only in KIND and / or POLARI should have the same MSET.

-UV-limits: GRIDER has provisions to limit the use of U,V-data to a box or ring by specifying low/high limits for U, V and SQRT (U^2+V^2).

-Passband: For line work bandpass corrections are needed. Provisions are there. It probably is easiest to reference a standard file.

-line work: extensions are necessary to be able to define a set of channels and sums of selected channels from the selected data base. Summation should be possible before or after gridding.

-flags: Correlation based flags are probably helpful, and could be added on a per scan base to the INX record.

-output order: GRIDER produces multiple maps in either map or frequency order.

-third dimension: The third dimension requires the number of planes used and the suppression possibility of the convolution correction, and a "field size".

-output size: Very often the output size of a map can be much smaller than the Fourier transform size. As an interim measure I have assumed that the output size is given by the user, and the transform size is the next higher power of 2. To let this work, the check for a power of 2 should be deleted from the DEC 10. Finally, however, the user should be able to set both sizes with maybe the default output size set to half the transform size.

-uniform weight: the user should be able to set the weight box size in both dimensions in nsec (not in gridpoints). For line channels you want to have the maps in the same coordinate system, not your U,V plane. Furthermore, you want to be able to select your weighting function (in both coordinates). GRIDER has, at the

moment, provisions for Gaussian and Box weighting.

-taper: Different taper values for U and V should be possible. Tapering functions are grid based in GRIDER to produce the same sky convolution function at different frequencies.

-convolution: separate U and V widths and parameters should be selectable. The present tie between size and function should be broken. Both values should be specifiable.

-map addition: Onee should be able to specify a map that will be weighted and added to the final output map. Usable to subtract e.g. continuum from all line maps. (Note: input logics exist, small additions to P2MAP and F2OB have to be made).

-source subtraction : subtraction should be extended, and preferably be done via a "standard" source file. Furthermore, one should be able to specify subtraction from the gridded U,V-plane. The last possibility exists in GRIDER only if the map and the antenna pattern are produced in the same run.

-four IF's: POLARI should include possibilities for selection. E. g. 1 (use only A and C), 2 (use B and D), B and D.

Appendix B - Error Messages

Error messages are produced by the routine PPERR (in FILBUF module). They give the time, a number, and text. None of the messages give extra information. However, PPERR had provisions for extra information, and it is easy to include it in the call to PPERR whenever necessary.

Errors are a warning or fatal. The type is indicated in the next list by the sign of the number (+=warning, -=fatal).

B.1. GRIDER messages

-900 : Cannot start a process because PPMAIN is not installed or busy. Busy indicates a bug in the rundown process.

+901 : Cannot open catalog
+902 : Cannot open MIX file
+903 : Read error MIX file

B.2. PPMAIN messages

-501 :PPSET not installed or busy
-502 :Error in execution of PPSET
-503 :PPFT1 not installed or busy
-504 :Error found in initializing PPFT1
-505 :PPTR not installed or busy
-506 :Error found in initializing PPTR
-507 :Fatal error in PPTR
-508 :Fatal error in PPFT1
+509 :Fatal error in PPFT2

-510 :Cannot assign AP1
-511 :PPFT2 not installed
-512 :Error found in initializing PPFT2
-513 :Cannot open MIX file
-514 :Read error MIX file
-515 :Cannot open INX file
-516 :Read error INX file
+517 :Too many datasets in input
+518 :Cannot catalog gridding output

B.3 PPTR messages

-101 :No memory in AP1
-102 :No Map asked
-103 :The AP not available
-104 :Cannot open work file GAIN.TMP
+105 :Cannot detail the INX pointers in AP memory
-106 :Read error GAIN.TMP
-107 :No SORTER database program yet
-108 :Cannot open INX file
-109 :Cannot open VIS file
-110 :Read error INX file
-111 :Read error VIS file

B.4. PPFT1 messages

-201 :No memory in AP2
-202 :AP2 not available
-203 :Cannot open workfile GRID.TMP
-204 :Cannot open MIX file
-205 :Read error MIX file
-206 :Write error GRID.TMP file
-207 :Read error GRID.TMP file

B.5. PPFT2 messages

-301 :No memory in AP3
-302 :AP3 not available
+303 :Cannot find a filename for output
+304 :Open error output file
+305 :Write error output file
+306 :Cannot catalog output file (gives name of output file)
+307 :Cannot obtain MIX record

B.6. PPSET messages

-551 :Cannot open GAIN.TMP work file
+552 :Too many INX sets
+553 :GAI file format error
-554 :Write error GAIN.TMP file
-555 :Cannot open MIX file
-556 :Read error MIX file
-557 :Cannot open INX file
-558 :Read error INX file

-559 :Write error INX file
-560 :Cannot open INX file
-561 :Cannot open GAI file
-562 :Read error INX file
-563 :Read error GAI file

B.7 PPCAT messages

+401 :Cannot open GRID.TMP file
-402 :Read error GRID.TMP file
+403 :Disk I/O error
+404 :Cannot open grid output file
+405 :Write error grid output file
+406 :Cannot enqueue on catalog
+407 :Cannot open catalog
+408 :Catalog creation error
+409 :Catalog read error
+410 :Catalog write error
+411 :Cannot open MIX file
+412 :Read error MAP file
+413 :Write error MAP file
+414 :Read error MAP file
+415 :Open error MAP file

Appendix C Timing

Timing depends critically on memory available in the PDP 11/44 and on disk contention.

As an example A 1024*1024 map with 18000 unsorted visibility records required 3 passes through the input data. In the case of no other program running, each pass took about 35 sec. Initialization took 10 sec. and the second phase FFT took about 60 sec. including writing of the map to disk.

However, running concurrently with PPCAT, the times were:

Initialization: 18 sec.
Gridding pass : 70 sec.
2nd phase FFT : 300 sec.

WB/tr