Ground System Tutorial

David A. Gell
734.763.6221 (voice) 734.763.7130 (fax)
gellda@umich.edu

Operating the TIDI Ground System
Presentation Outline

- **Instrument Fundamentals**
- **Documentation**
- **Programming the Instrument**
- **Ground System Components**
- **Operating the Engineering Stack**
- **TIDI Real Time Command and Display System**
- **Mission Operations**

Instrument Fundamentals

- Scientific Goals
  - TIMED Mission Objectives
  - TIDI Instrument Objectives
- Measurement Technique
  - Wind Measurements
  - Temperature Measurements
  - Constituent Abundances
- Instrument Hardware
- Instrument Software
- Instrument Operations

Instrument Fundamentals
TIMED Mission Objectives

- **Mission Science Objectives**
  - **To investigate and understand the energetics of the mesosphere and lower thermosphere (MLTI)**
- **Mission Goals**
  - **To determine the pressure, temperature, density, and wind structure in the MLTI region, including seasonal and latitudinal variations.**
  - **To determine the relative importance of various radiative, chemical, electro-dynamical, and dynamical sources and sinks of energy for the thermal structure of the MLTI.**

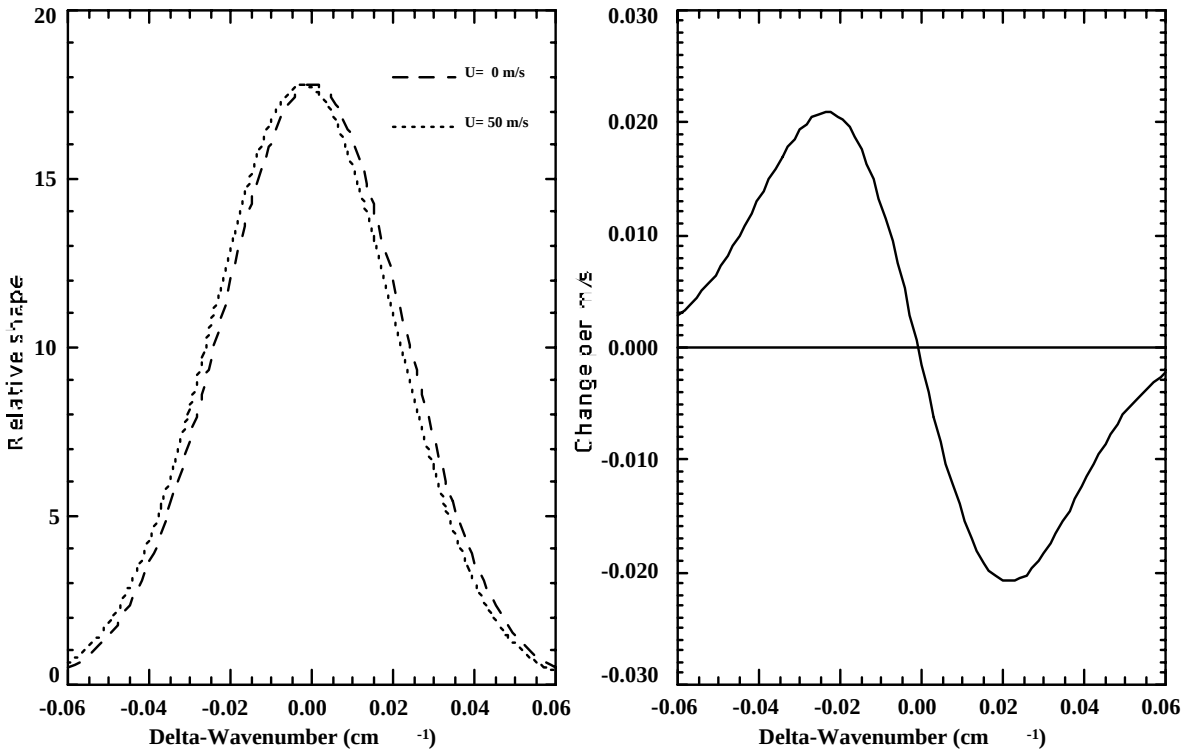
Instrument Fundamentals
TIDI Instrument Objectives

- **Obtain global wind measurements from 60 km to 300 km for at least 2 years with an accuracy of a few m/s.**
- **Obtain global measurements of temperature and species volume emission rates (VER) in the MLTI region.**
- **Derive concentrations of important minor species, such as O, O3, and O(1D).**

Instrument Fundamentals
TIDI Doppler Wind Measurements

- **TIDI measures wind by measuring the Doppler shift of atmospheric emission features.**
- **Doppler shifts are small; 10 m/s is ~5 x 10-4 cm^{-1}, ~ 2 x 10-5 nm, or ~17 MHz**
- **Accurate knowledge of pointing is required to compensate for satellite motion, ~7500 m/s**
- **Wind vectors are formed using 2 approximately orthogonal views of nearly the same volume through forward and aft viewing telescopes.**
- **The telescopes' zenith angles are adjusted to look at different altitudes.**
- **Different emitters are used at different altitudes to cover the regions of interest.**

Instrument Fundamentals
TIDI Wind Measurements



Spectra **Difference**

u = 1 m/s
$\Delta\nu \cong 5 \times 10^{-5}$ cm^{-1}
$\cong 2.2 \times 10^{-6}$ nm
= 1.5 MHz

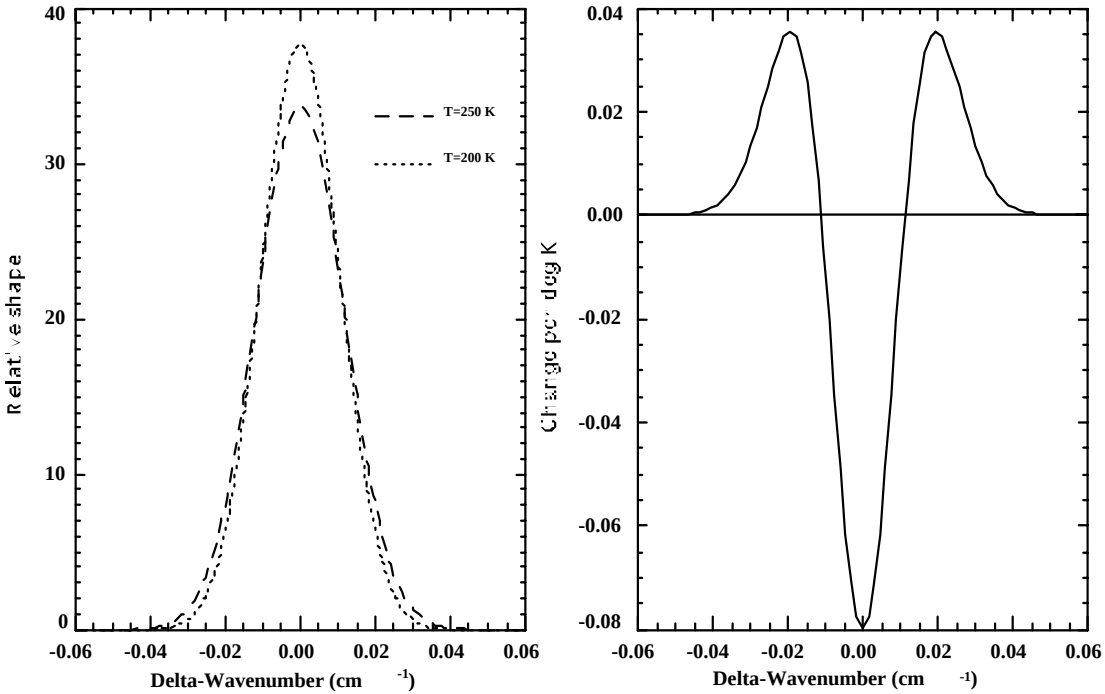
$$\Delta\nu = (\nu_0 u)/c$$

Instrument Fundamentals
TIDI Temperature Measurements

- **Doppler width. The spectral width of emission line is proportional to the square root of the temperature.**

- **Rotational temperature. The ratio of the brightness of two lines in a rotational band of a molecular emission is uniquely related to the temperature.**

Instrument Fundamentals
TIDI Doppler Temperature Measurements



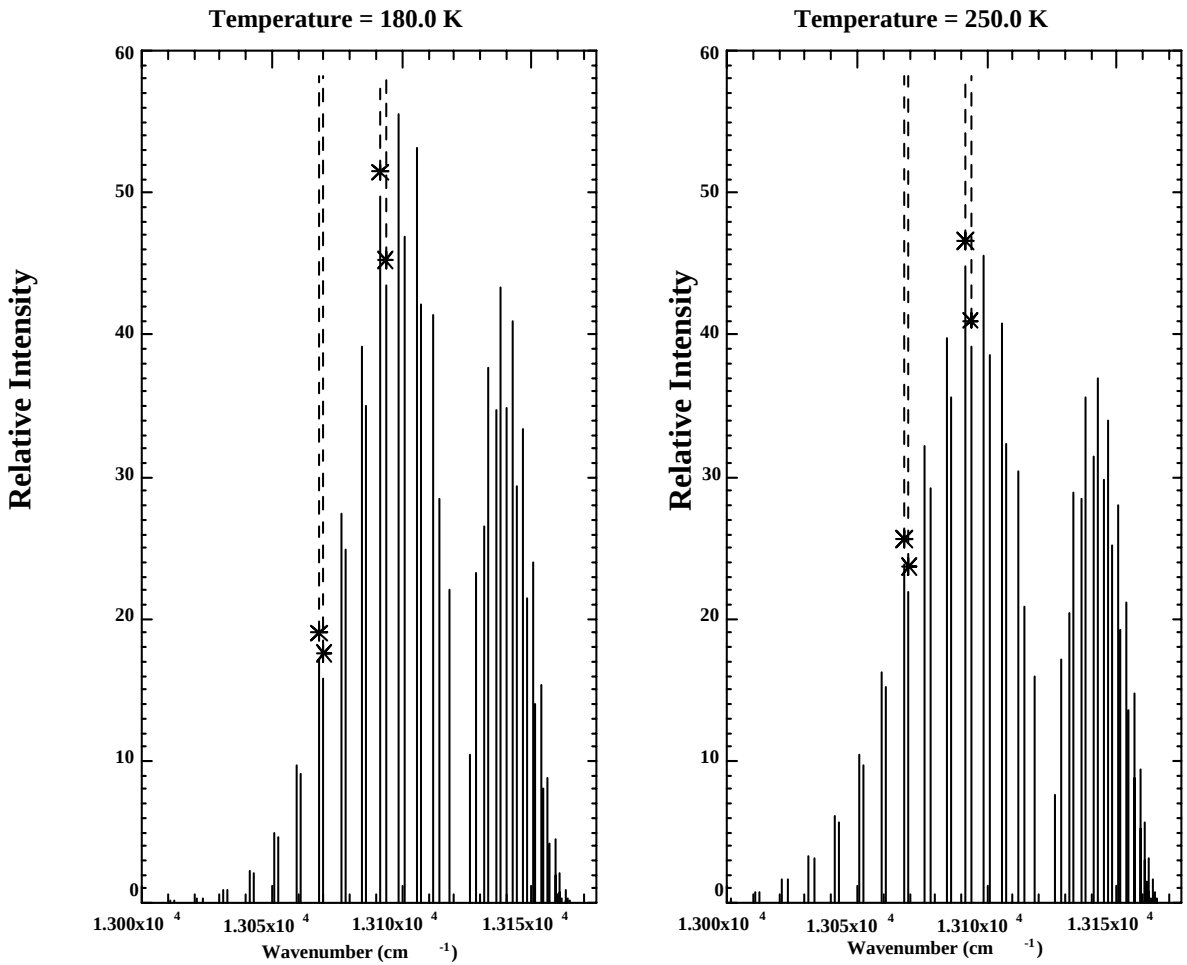
Spectra **Difference**

$$\Delta\nu = 4.30\text{x}10^{-7}\,\nu_0\sqrt{\text{T/M}}$$

Width

Emitter	M	T(K)	(cm^{-1})	(nm)	(MHz)	(m/s)
$O_2(^1\Sigma)$	32	200	0.014	8.2×10^{-4}	420	320
$O(^1S)$	16	200	0.027	8.5×10^{-4}	820	450
$O(^1D)$	16	1200	0.059	2.3×10^{-3}	1765	1100

Instrument Fundamentals
TIDI Rotational Temperature Method

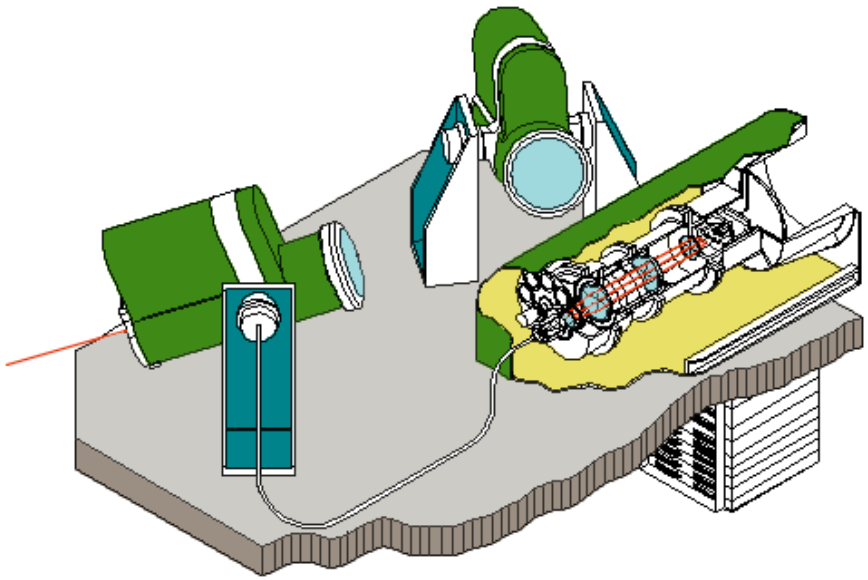


O_2 (0,0) Line Strength Change with Temperature

Instrument Fundamentals
Minor Species Density

- **The volume emission rate is related to concentration of various components.**
- **A knowledge of the chemistry allows minor species density to be retrieved.**
- **The absolute sensitivity of the instrument must be known very well.**
- **Can recover O, $O(^1D)$, and O_3.**

Instrument Fundamentals
Instrument Hardware

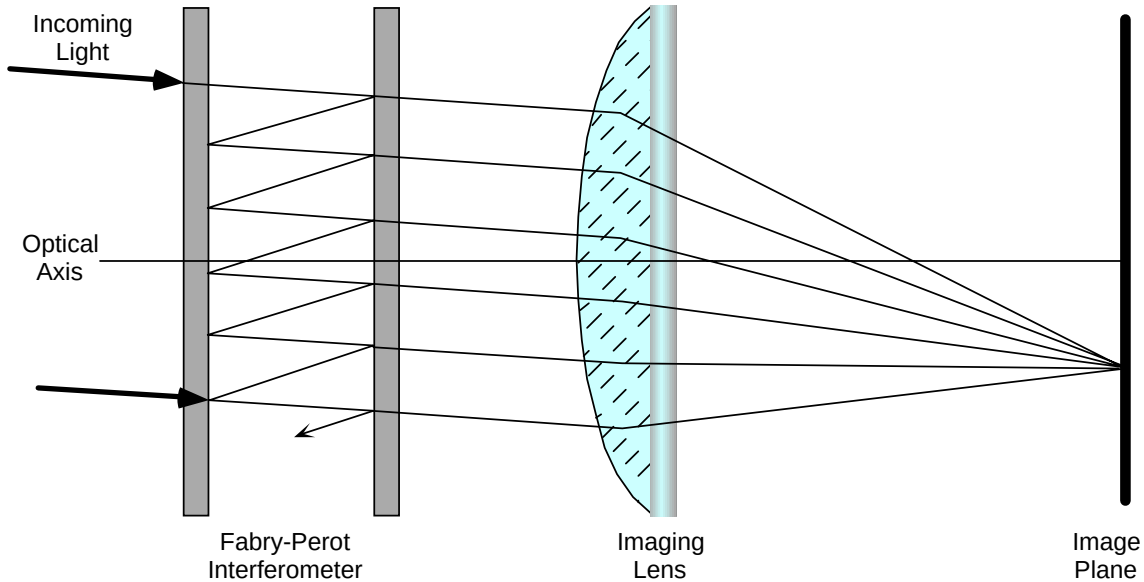


- **Profiler**
  - **Fabry-Perot Interferometer**
  - **All views and calibration analyzed simultaneously**
- **4 telescopes**
  - **Collect light from regions of interest**
  - **Gimbaled to view various altitudes**
  - **Vectors formed by combining pairs of views**
- **Electronics**
  - **Provides interface to spacecraft**
  - **Computer controls instrument**

Instrument Fundamentals
Instrument Hardware

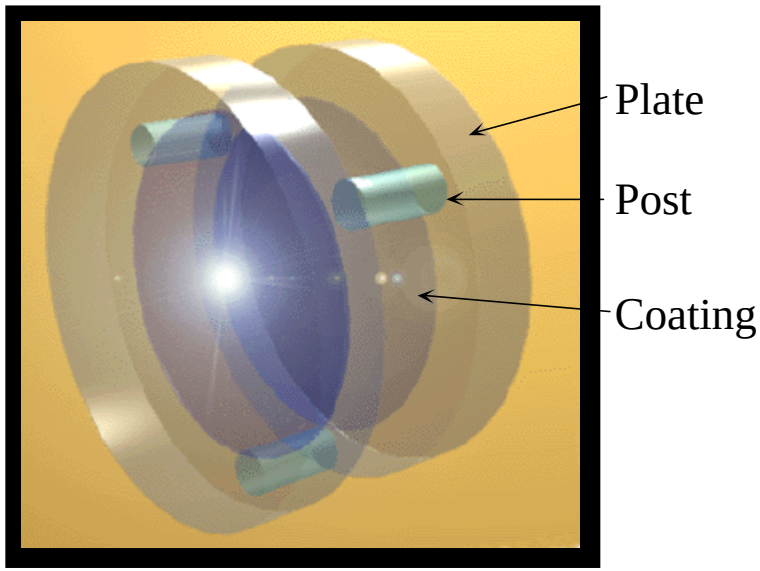
- Fabry-Perot Interferometer
  - All telescopes and calibration source simultaneously observed
  - Fixed gap etalon
  - 2 filter wheels with 8 positions each
  - Conical mirror circle to line converter (clio)
  - CCD detector
- Telescopes
  - 4 Identical Telescopes with 0.05° x 2.5° field of view
  - View directions ±45°, ±135° from the spacecraft x axis
  - Single axis gimbal, 22° travel centered at 20.35° down from horizontal
- Electronics Package
  - Power supplies
  - Data collection
  - Instrument processor - 80C51 8-bit micro-controller

Instrument Fundamentals
Profiler

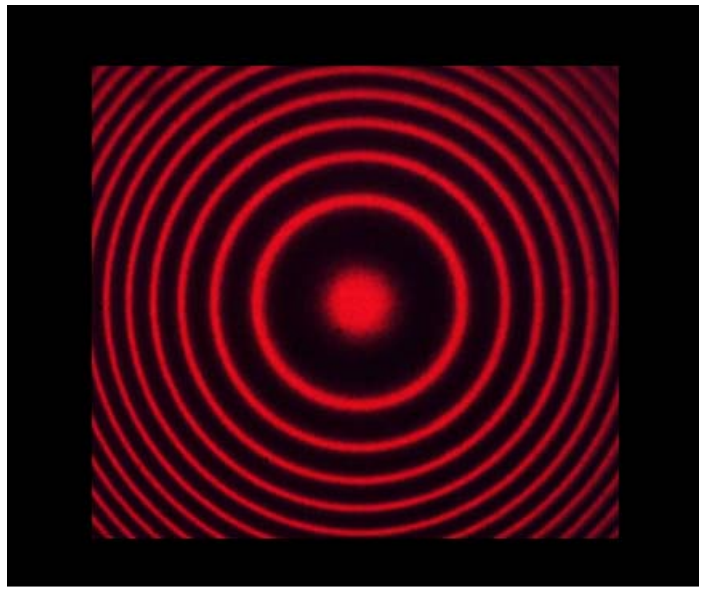


- **Fabry-Perot Interferometer**
  - **Pair of extremely flat reflective surfaces**
  - **Interference forms circular fringes**
  - **Spacing of fringes depends on wavelength, refractive index within the gap, and the width of the gap**

Instrument Fundamentals
Profiler

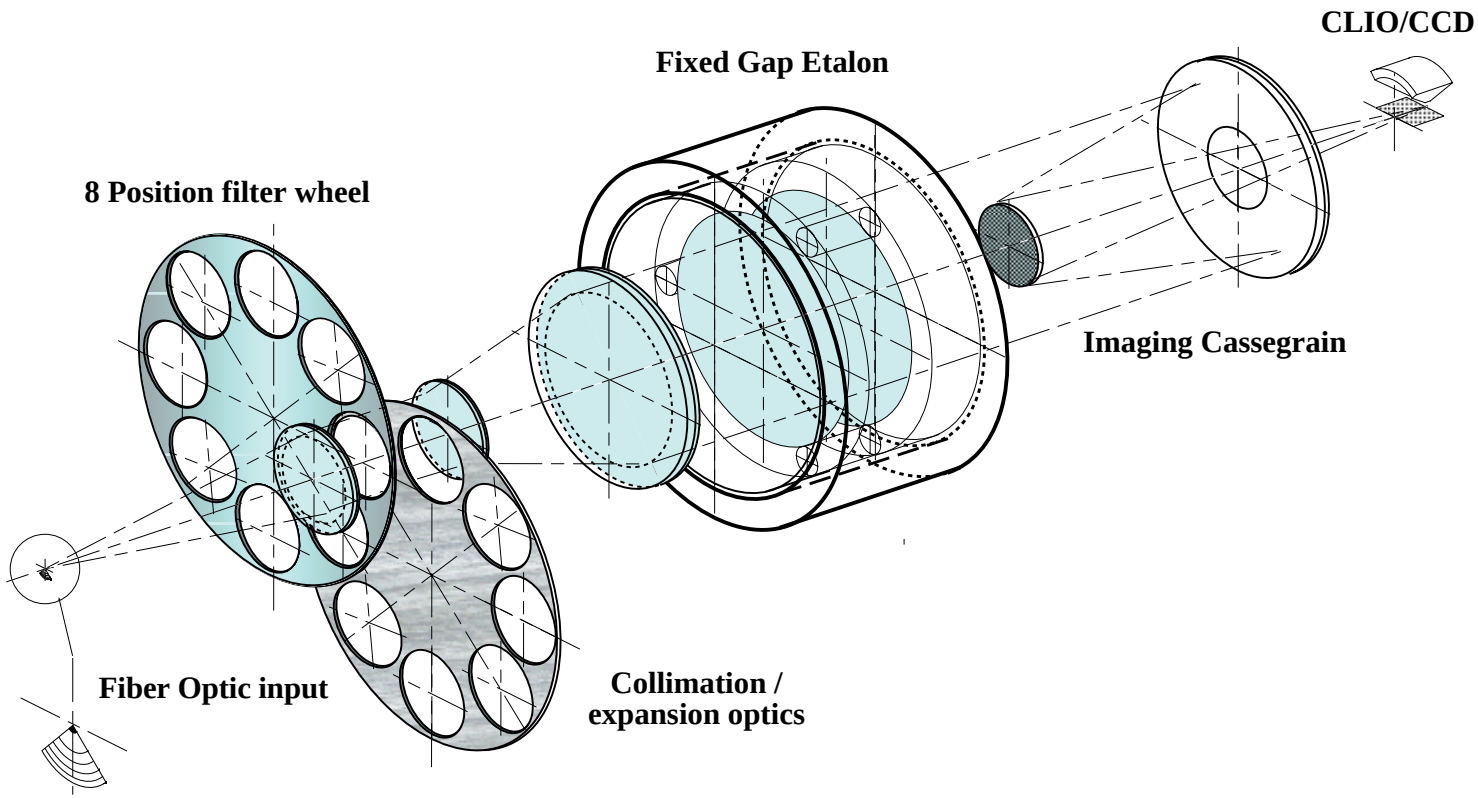


Etalon

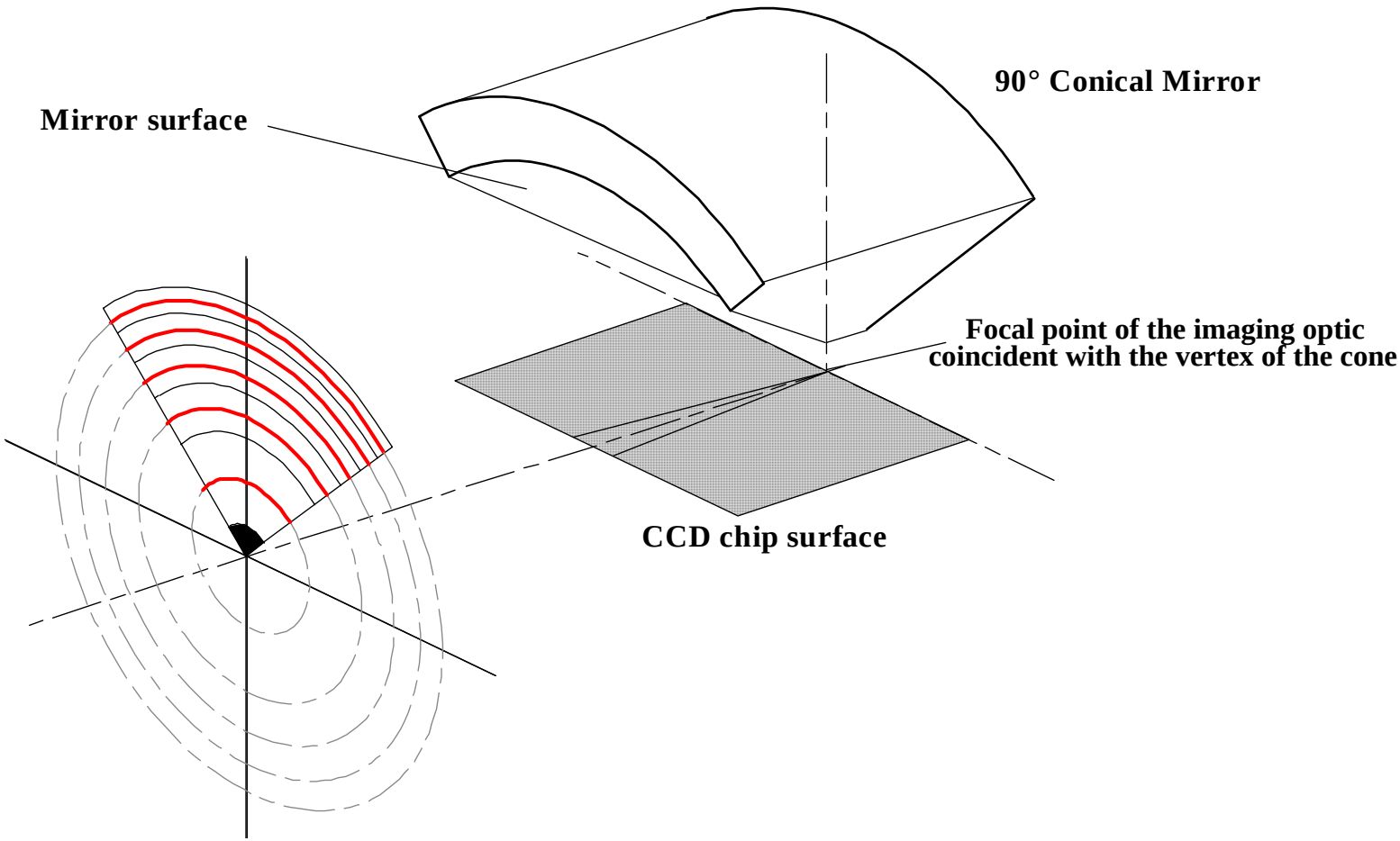


Fabry-Perot Fringe Pattern

Instrument Fundamentals
Profiler

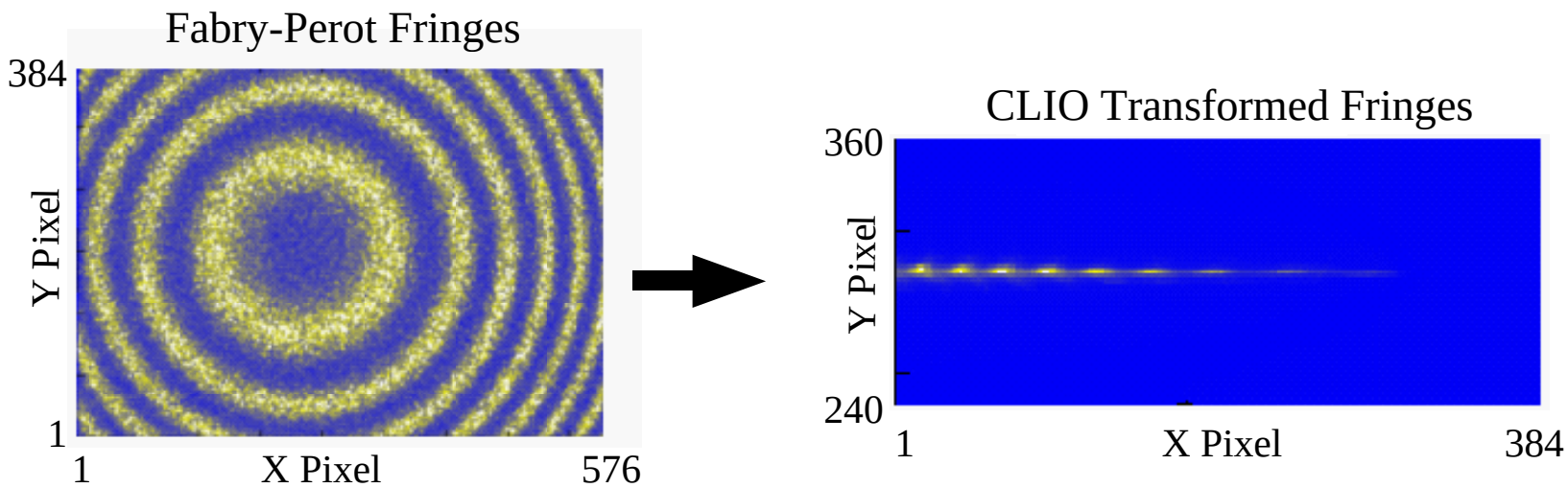


Instrument Fundamentals
Circle to Line Converter

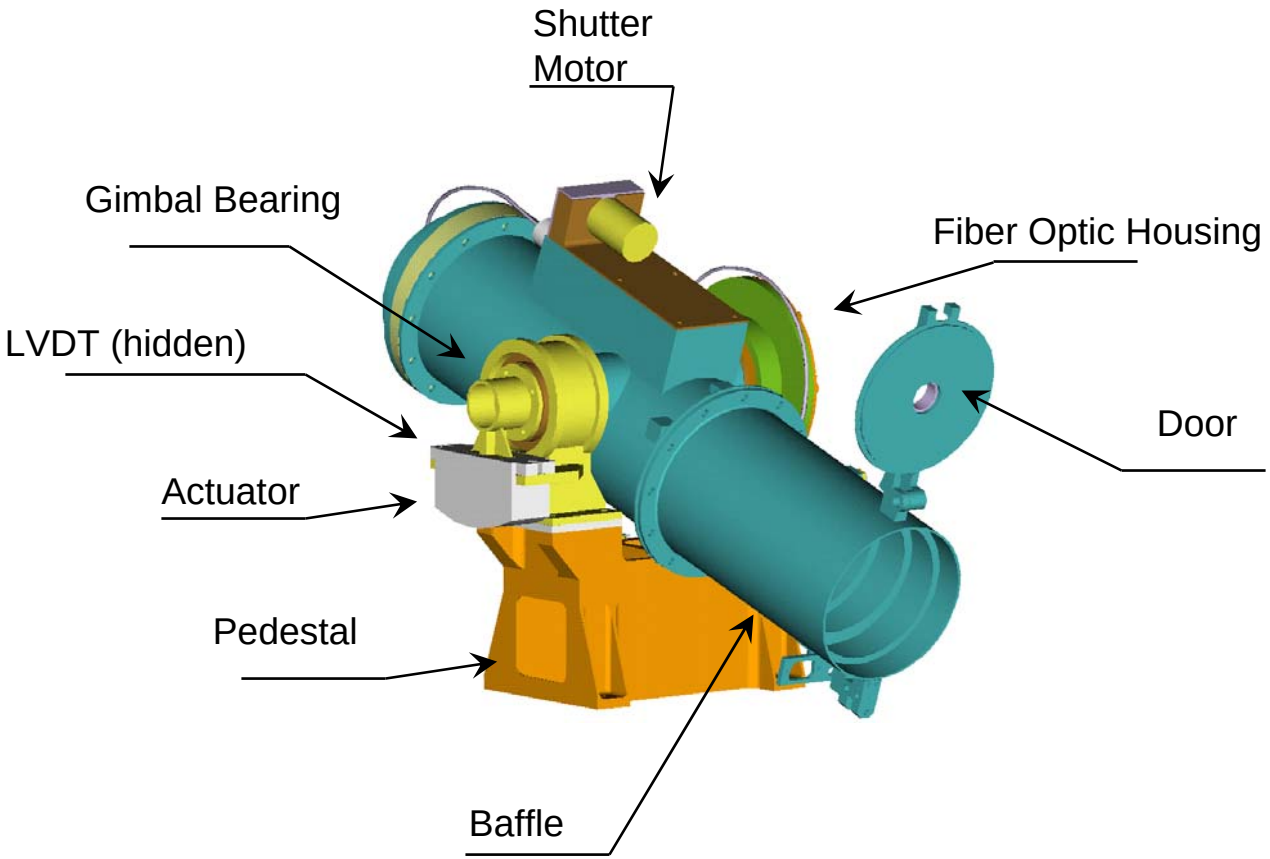


Instrument Fundamentals
Circle to Line Converter

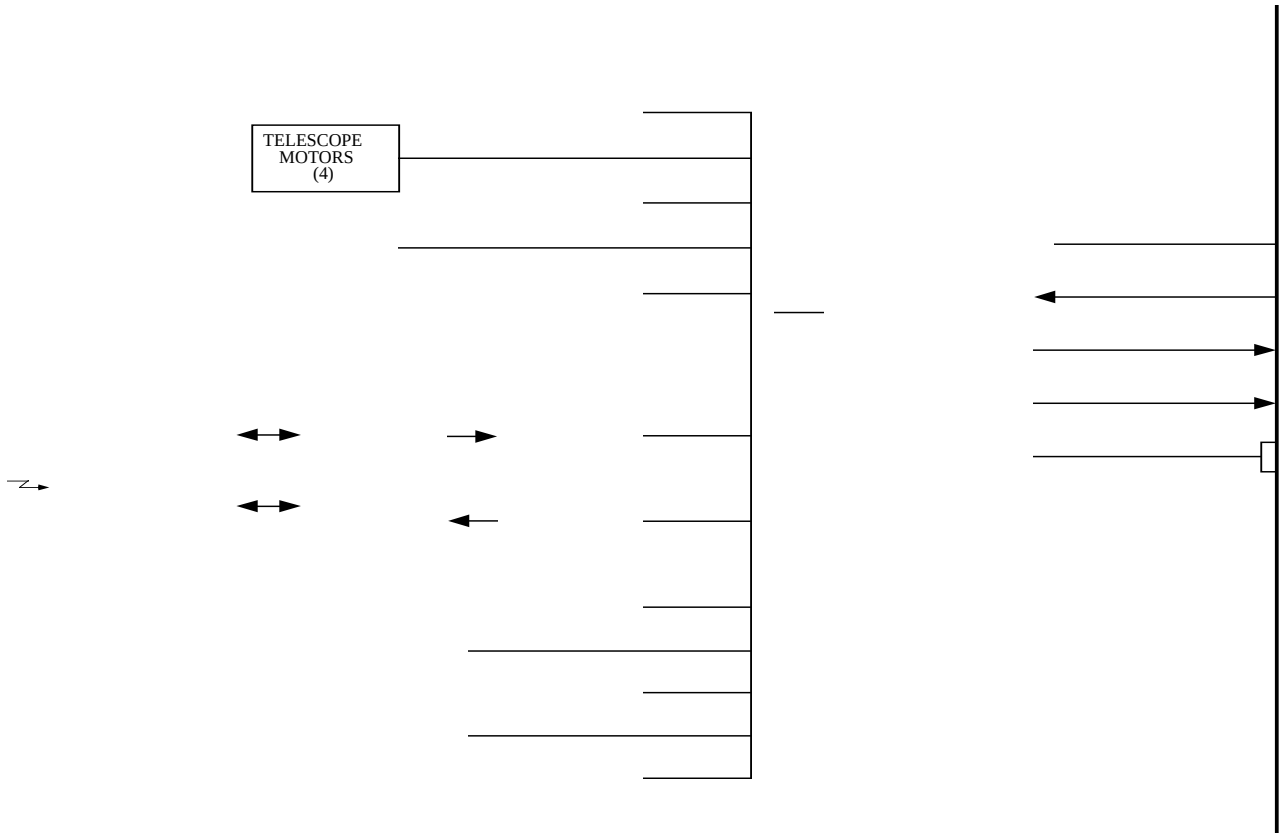
- **Transforms FPI rings into linear pattern**
- **Allows use of linear array detector (e.g. CCD)**



Instrument Fundamentals
Telescope

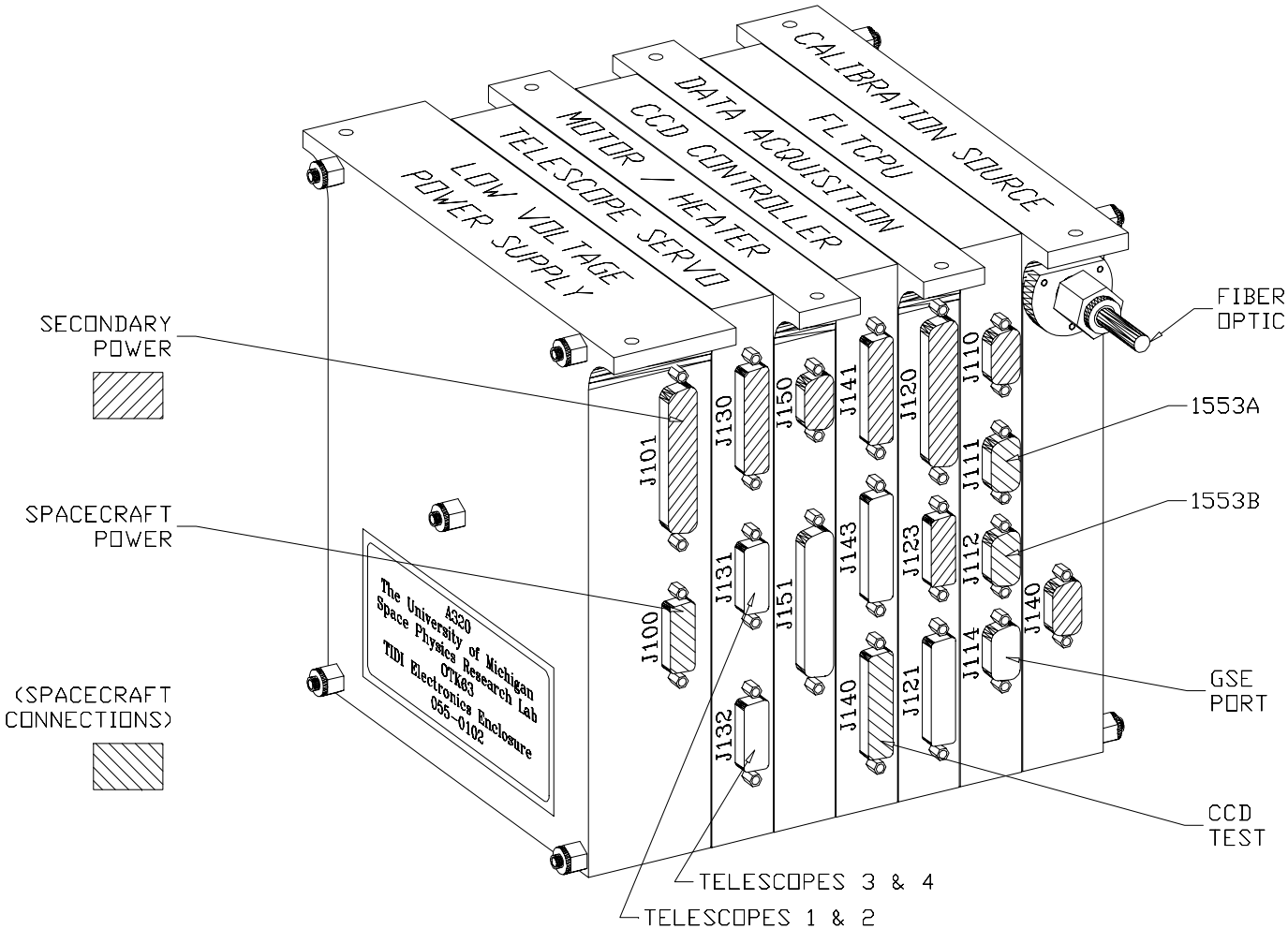


Instrument Fundamentals
Electronics Stack



Instrument Block Diagram

Instrument Fundamentals
Electronics Stack



Instrument Fundamentals Software

- **Instrument Software Modes**
  - **Boot mode**
    - Interprets a limited set of commands
    - Produces housekeeping data only
    - Main function is to initialize instrument hardware and transfer control to instrument software
  - **Instrument Software**
    - Controls observation sequence
    - Produces all data packet types
    - Interprets full set of commands

Instrument Fundamentals Software

- **Mode Transitions**
  - **Boot → Instrument**
    - On receipt of boot_now command
    - After power-up if no boot mode commands are received
  - **Instrument → Boot**
    - When watch dog timer expires by command
    - When watch dog timer expires due to software problem

Instrument Fundamentals
Instrument Software

- **Foreground Activities**
  - **Execution of control programs**
  - **Execution of commands received via 1553 bus**
- **Background Activities**
  - **Execution of scan tables**
  - **Control of heaters for thermal control**
  - **Creation and transmission of telemetry packets**
  - **Management of communications interfaces with spacecraft**

Instrument Fundamentals
Instrument Software - Scan Tables

- Science data is collected only while a scan table is being executed
- Provides list of instrument states
  - Filter wheel positions
  - Integration period
  - Telescope position
  - Calibration lamp state
  - CCD binning and gain
  - Shutter position
- Spectra are collected at each state
- Scan tables execution is controlled by control program

Instrument Fundamentals
TIDI Instrument Command Language (TICL)

- **TICL provides statements for:**
  - **Compiler control**
  - **Arithmetic commands**
  - **Program control**
  - **Control structures**
  - **Instrument control**

- **Control Programs**
  - **Includes program command**
  - **May include subroutines**
  - **Stored in holding buffer on receipt**

- **Command Block**
  - **Specified by .immediate compiler directive**
  - **May not include**
    - program statement
    - control structures
    - Local variables
  - **Executed upon receipt**

```
repeat
   do until date changes
      if on day side
         set up ccd
         load binning tables
         load scan table for day
               measurements
      else
         perform telescope lubrication scan
         execute a calibration segment
         set up ccd
         load binning tables
         load scan table for night
               measurements
      endif
         start scan
         do until the next terminator crossing
            if the date has changed, break
            wait 10 seconds
         end do

         stop scanning at end of table

   end do

until holding buffer is valid
start control program in holding buffer
```

Daily Control Program Structure

Instrument Fundamentals
Instrument Operations Activities

- **Command planning**
  - **Planning aids predict geometry**
  - **Scan tables selected/developed**
  - **Planned timelines delivered to TIDI MDC**
- **Preparation of command messages**
  - **Control program replacements**
  - **Immediate commands**
- **Transmission of command messages**

Instrument Fundamentals
Instrument Operations Activities

- **Instrument Health Assessment**
  - **Command message acknowledgements**
  - **Limits and trends**
- **Production data processing**
  - **Automated scripts for data retrieval and processing**
  - **Data accounting confirms processing**
  - **Assess instrument performance with science products**
  - **Deliver product availability notices (PANS) and actual timelines to MDC**

Instrument Fundamentals
Command Generation

- Command generation requires
  - Development of scan table if required
  - Development of control program or command block
  - Compilation
  - Construction of command messages from compiler output
  - Archival storage of compiler inputs and outputs
  - Encryption and transmission to TIMED MOC
- Programs used include
  - TICL
  - package
  - ticlBuilder
  - ticlArchiveAdj
  - sendCM
  - reportMsg
  - updateMsg
  - updMsgDrv.sh

Instrument Fundamentals
Command Generation Tools

- **Compiler**
  - **Translates TICL statements in instrument commands**
  - **Produces a command block file containing**
    - a series of immediate commands

    or
    - a control program
- **Package**
  - **Builds command messages, adding project specified header information**
  - **Builds a command message history file**
  - **Invoked by the sendCM script**

Instrument Fundamentals
Command Generation Tools

- **ticlBuilder script**
  - **Determines the dependencies of a ticl file**
  - **Forms and invokes a make file to build one or all ticl files**
  - **Compiles control program**
    - All programs in the /tidi/flt_ticl/ticl project
    - One program, archiving all inputs, listing file, and tcmd
  - **Cleans directories**
- **ticlArchiveAdj script**
  - **Duplicates an archive directory**
  - **Used to propagate the archive of a control program through subsequent days until the next upload is performed**

Instrument Fundamentals
Command Generation Tools

- **sendCM script**
  - **Invokes package to form command messages**
  - **Invokes pgp to encrypt and digitally sign the messages**
  - **Invokes ftp to transmit messages to the MOC**
- **reportMsg**
  - **View reports of message history**
- **updateMsg**
  - **Reads a receipt file and updates message history file**
- **updMsgDrv.sh**
  - **Invokes updateMsg for each receipt file found.**

Instrument Fundamentals
Instrument Operations

- **Real time operations**
  - **Used for testing and initial operations**
  - **Commands generated with RCDS**
    - OASIS application
    - Uses the ticl compiler and sendCM for command generation and transmission
  - **Command transmitted by TIDI operator at any appropriate time**
  - **Production data processing scripts used to retrieve and process data**
- **Routine operations**
  - **Used for all science operations in orbit**
  - **Commands generated by direct application of compiler and sendCM**
  - **Command messages are staged at the TIMED MOC until contact**
  - **Real-time data limited to ~10 minutes at 2 contacts per 24 hr**
  - **Playback data retrieved and processed by production scripts**

Documentation

- **TIMED Documents**
  - **GIIS (Section 8 in particular)**
  - **Voice Net Procedures**
- **Instrument Software**
  - **Requirements Specification**
  - **Data Dictionary**
  - **Spreadsheets**
- **Instrument Command Language**
  - **Requirements Specification**
  - **TICL Conventions**
- **Instrument Operations**
  - **Operations Checklists**
  - **Daily Operations Guide**
  - **Special Test Procedures**
- **TIDI RCDS**
  - **OASIS-CC Documentation**
    - CSTOL Reference Manual
    - Database Guide
  - **Design and Maintenance Document**

Documentation
TIMED Documents

- **TIMED Staff, "TIMED Spacecraft General Instrument Interface Specification", APL Document 7363-9050**
  - **Defines all interfaces between the instruments and the spacecraft**
  - **Defines interfaces between the TIMED MOC and the TIDI POC**
- **Grunberger, P., "Telemetry and Command Data Structures for TIMED"**
  - **Provides additional explanatory information about CCSDS packets**
  - **Defines ground receipt headers**
  - **Defines POC Telemetry Packets, PTP**
- **Dove, W., "TIMED Voice Communications System Operations"**
  - **Describes use of voice network**
  - **Specifies channel allocations and call signs**

Documentation
Instrument Software

- **Musko,S., “TIDI Flight Software Requirements Specification”, 055-3320**
  - **Describes the instrument flight computer configuration.**
  - **Describes the capabilities of the boot code**
  - **Describes the capabilities of the instrument code**
  - **Describes the telemetry and command message formats**
- **Musko, S., “TIDI Flight Software Instrument Parameter Dictionary”, 055-3903**
  - **Lists the contents of the TIDI housekeeping TM packet**
  - **Specifies precision and format of each data item**
- **TM Packet definition spreadsheets 055-3519, 3693, 3694, 3695, 3718, 3946, 3947, 3948**

Documentation
Instrument Command Language

- **Gell, D., "TIDI Instrument Command Language Compiler Specification and User's Guide", 055-3564**
  - **Describes the syntax and semantics of the TICL language.**
  - **Includes a TICL design standard (coding standard TBS)**
- **Gell, D., "TIDI Scan Table File Format", 055-3527**
  Gell, D., "Binning Table File Format", 055-3603
  - **Describe the format for tables supplied to the TICL compiler**

Documentation
Instrument Command Language

- **Gell, D., “TICL Conventions”, 055-3765**
  - **Provides numbering conventions for programs and tables**
- **Gell, D., “TIDI Measurement Sequence Specifications”, 055-3431**
  - **Specifies science scan sequences**
  - **Provides design information, filter locations, operational restrictions**

Documentation
Instrument Operations

- **Gell, D., “Ground System Tutorial”, 055-3796**
  - **This document**
- **Gell, D., “TIDI Operations Checklists” 055-3886**
  - **Describes procedures for major instrument activities**
    - Ground testing
    - Initial Operations
    - Normal command operations
  - **Describes procedures to diagnose and recover from ground system anomalies**

Documentation
Instrument Operations

- **Gell, D., “TIDI Daily Operations”, in preparation**
  - **Describes the procedures for routine command generation and data processing**
- **Gell, D., “TIDI Instrument Activation Scenario” 055-3757**
  - **Provides specifications for activation procedures**
  - **Operational procedures in 055-3886**

Documentation
Real-time Command & Display

- Gell, D., “TIDI RCDS Design and Maintenance”, 055-3957
  - In preparation
  - Describes the implementation of the OASIS-CC application for TIDI
  - Describes the CSTOL procs, macros and utility tasks required for TIDI operation
  - Describes the interaction of OASIS and shell scripts used in commanding
- OASIS-CC Document Set
  - CSTOL Reference Manual
  - Database Guide

Programming the Instrument

- Language Elements
  - identifiers
  - operands
  - statements
- Statements
  - Compiler control
  - Arithmetic commands
  - Program control
  - Control structures
  - Instrument control
- Tables
  - Scan Tables
  - Binning Tables
- Program Organization
  - Immediate Command Blocks
  - Stored Control Programs
  - Default Control Program
- TICL Tools
  - Compiler
  - De-compiler
  - Scripts

Language Elements
Identifiers

- **Specify names of instrument parameters, global variables, local variables, and subroutines**
- **Consist of alphanumeric characters and the underscore**
  - **32 characters maximum length**
  - **Must begin with a letter or an underscore**
  - **Case is not significant**

Valid Identifiers	
sub1	A_mix123ed_Label
sub_1	_1234
_trial	

Invalid Identifiers	
2sub	initial character cannot be a numeral
label@line	invalid character

Language Elements
Operands

- **Constants**
  - **Numeric Constants**
  - **Hexadecimal Constants**
  - **String Constants**
  - **Engineering Unit Constants**
- **Variables**
  - **Instrument Parameters**
  - **Global Variables**
  - **Local Variables**

Language Elements
Operands

- **Numeric constants**
  - **Consist of numerals, one decimal point, optional + or - sign, and optional exponent field**
  - **Numeric constants are converted by the compiler to the appropriate internal representation**

Valid Numbers
-0.123
10.125
+12
1.0E-3

Invalid Numbers	
2ABCE	Alphabetic characters not allowed
1.23.3	Two decimal points

Language Elements
Operands

- **Hexadecimal Constants**
  - **Consist of numerals 0, 1, … 9 and letters A, B,…F with the letter H appended**
  - **Case is not significant**
  - **Maximum length**
    - 5 characters when specifying an address
    - 4 otherwise

Valid Hexadecimal Constants	
AB12H	
aBH	
FfFfh	
A1234H	valid as an address argument, invalid otherwise

Invalid Identifiers	
2AdE	No H suffix
H2345	H must be suffix, no prefix
2AgEH	invalid character ‘g’
123456H	too many characters

Language Elements
Operands

- **String Constants**
  - **Consist of alphanumeric characters**
  - **Case is not significant**
  - **Used as operands for commands**
  - **Each command may have its own set of valid string constants**

Valid String Constants
Off Closed OpEN

Invalid String Constants
Any string not defined for the command

Language Elements
Operands

- **Engineering Unit Constants**
  - **The user must specify both a value and a parameter name**
    - Flight computer performs only fixed point arithmetic
    - Compiler converts from float to fixed point according to the parameters engineering unit to DN conversion
  - **Constants are formed *%instrumentParameterName(value)***
    - **The names are found in**
      Musko, S., “TIDI Flight Software Instrument Parameter Dictionary”, 055-3903
    - **The percent sign is required**

Valid EU Constants
%CCD_Temp(83.5)
%tel1_position(12.35)

Invalid EU Constants	
CCD_Temp(83.5)	no ‘%’ prefix
%CCD_Temp	no value supplied
%CCD_Temp83.5	missing parentheses

Language Elements
Operands

- **Instrument Parameters**
  - **Instrument parameters provide access to the state and configuration of the instrument**
  - **Identified by a mnemonic**
    Musko, S., “TIDI Flight Software Instrument Parameter Dictionary”, 055-3903
  - **CASE is not significant**
  - **May be assigned a value with the *store, inc, dec, add* and *sub* commands**
  - **Examples**
    - TEL_4_Housing_Temp
    - CCD_temp
    - FW_1_Position

Language Elements
Operands

- **Global Variables**
  - **Class of instrument parameters**
  - **Assigned values through *store, inc, dec, add*, and *sub* commands**
  - **Identifier:** global_nn, **where nn is a number from 1 to 32, inclusive**
  - **Conventions for use**
    - global_01: process id
    - global_02 …global_11 messages
    - global_12…global_22 state memory
    - global_23…global_32 subroutine arguments

Language Elements
Operands

- **Local Variables**
  - **Assigned names with the *local* command**
  - **Names must be valid identifiers**
  - **May only be used in subroutines**
  - **Initialized to zero by default, nonzero value may be specified**
  - **May be assigned a value with any of the arithmetic commands**

Example declaration

```
local day_count 10    ; creates variable and initializes to 10
local _private        ; creates variable, default initialization to 0
```

Programming the Instrument Language Elements

- **Compiler Directives**
  - **Compiler Directives consist of a keyword and parameters**

```
keyword  parameter;comment
```

    - Compiler directive keywords begin with a period (.)

- **Statements**
  - **Statements consist of a command field and a comment field**

```
command  ;comment
```

    - Each field is optional
    - Fields are separated by white space, spaces and/or tabs
  - **Commands consist of a keyword followed by operands**

Language Elements
Compiler Directives

- **Directives consist of a keyword prefixed with a period and optional parameters**
- **Directives modify the operation of the compiler or specify information to be placed in the output command block file**
- **Directives**

.include	**includes the text of a file at the location of the directive** .include “/tidi/flt_ticl/ticl_subs/fltCal.ticl”
.define	**specifies a replacement string for a token** .define PID global_01
.purpose	**provides metadata to be included in the output command block file** .purpose “perfom routine hi-beta science data collection”

Language Elements
Compiler Directives

- **Directives**

.immediate specifies that the output commands are to be interpreted immediately upon receipt at the instrument

.scan_table_start
.scan_table_end identifies the beginning and end of an in-line scan table

.bin_table_start
.bin_table_end identifies the beginning and end of an in-line binning table

Language Elements Statements

- Arithmetic
  - permit the assignment of values
  - perform integer arithmetic
- Program Control
  - subroutine definition and invocation
  - local variable declarations
  - delay
- Control Structure
  - conditional expressions
  - logic tests
  - iterations
- Instrument Control
  - scan table control
  - binning table control
  - direct mechanism control
- Miscellaneous
  - boot control
  - load and dump memory
  - poke memory locations
  - checksumming
  - control program holding buffer maintenance
  - watchdog timer

Language Elements
Statements

TICL Operand Classes	
class	description
1	Instrument Parameter Mnemonics
2	String Constants
3	numeric constants
4	hexadecimal constants
5	global variables
6	local variables

Statements
Arithmetic Commands

Arithmetical Commands		
keyword	operands	
	destination	source
Store	variable (1,5,6)	variable or constant (1,3,4,5,6)
Add	variable (1,5,6)	variable or constant (1,3,4,5,6)
Sub	variable (1,5,6)	variable or constant (1,3,4,5,6)
Inc	variable (1,5,6)	
Dec	variable (1,5,6)	

Statements
Arithmetic Command Examples

```
Store CCD_Temp_Setpt %CCD_Temp_Setpt(-83.5)

add global_15 3

sub global_15 global_3

inc global_15

dec a_local_variable
```

Statements
Program Control

Program Control Commands		
keyword	operands	
	destination	source
Wait	Centi-Seconds (3,4,)	
Program	Control Program ID (3,4)	
Call	subroutine name (2)	
Subroutine	subroutine name (2)	
Return		
End		
Local	local variable name (2)	initial value (3,4)

Statements
Program Control Examples

```
subroutine a_routine
;this is a routine called by the main
        local loc_var
        store loc_var global_21
        if ccd_temp .lt. %ccd_temp(-90)
                return
        end_if
        inc global_01
end

program 13005; main starts
;
store global_01 10

wait 3000          ; wait 30 seconds

call a_routine
```

Statements
Control Structures

- **Conditional Expressions**

  rvalue1 <op> rvalue2

- **Operators**
  - **.eq.** **true if rvalue1 equals rvalue2**
    false otherwise
  - **.ne.** **true if rvalue1 does not equal rvalue2**
    false otherwise
  - **.lt.** **true if rvalue1 is less than *or equal* to rvalue2**
    false otherwise
  - **.gt.** **true if rvalue1 is greater than *or equal* to rvalue2**
    false otherwise

Control Structures
Conditional Statement

```
IF <rvalue1> .op. <rvalue2>
        ;statements executed if condition is true
ELSE
        ;statements executed if condition is false
END_IF
```

- **If the conditional expression is true**
  - **Statements in the block following the *if* are executed**
  - **Control transferred to the statement following the *end_if***
- **If the conditional expression is false**
  - **Statements following the else are executed, if the *else* is present**
  - **Control transferred to the statement following the *end_if***

Control Structures
Conditional Statement Example

```
store DAY_SIDE spacecraft_day_night_stat
;;
if DAY_SIDE .eq. 1     ;; =1 for day, 0 for night
  call specSetup
  load_bin_table 1 "../bins/laserBins1.btab"
  load_scan_table "../scans/mltDbasic.scan"
else
  call telLubScan
  call fltCal
  call specSetup
  load_bin_table 1 "../bins/laserBins4.btab"
  load_scan_table "../scans/mltNbasic.scan"
end_if
```

Control Structures
While Iteration

```
WHILE <rvalue1> .op. <rvalue2>
        ;statements to execute
END_WHILE
```

- **Tests the exit condition prior to each iteration of the loop - body of the loop will not be executed if test is initially false**
- **It the condition is true, the body of the loop is executed**
- **If the condition is false, the statement following the *end_while* is executed**
- **The loop can be exited from within using the *break* statement**
- **The remaining portion of the body of the loop can be skipped with the *continue* statement**

Control Structures
While Iteration Example

```
while DAY_SIDE .eq. spacecraft_day_night_stat
        ;;
        if CURRENT_DAY .ne. time_utc_day_number
                break
        end_if
        ;;
        ;; wait 10 seconds and test for crossing again
        wait 1000
end_while
```

Control Structures
Repeat Iteration

```
REPEAT
        ;statements to execute
UNTIL <rvalue1> .op. <rvalue2>
```

- **Tests the exit condition after each iteration - always executes the body of the loop once.**
- **If the condition is false, execute the body of the loop again.**
- **The loop can be exited from within the body with a *break* statement**
- **A continue statement transfers control to the exit condition test.**

Control Structures
Repeat Iteration Example

```
repeat
        wait 50
until sys_expose_down_cnt .lt. 5
stop_scan_end
```

Control Structures
Break Statement

```
REPEAT
        ;statements to execute
        IF <rvalue1> .op. <rvalue2>
          BREAK
        END_IF
        ;more statements to execute
UNTIL <rvalue1> .op. <rvalue2>
        ; statement executed when IF condition is true
```

- **Transfers control out of the body of the innermost *while* or *repeat* loop**
- **Usually used in the body of an *if* statement**

Control Structures
Continue Statement

```
WHILE <rvalue1> .op. <rvalue2>
        ;statements to execute
        IF <rvalue1> .op. <rvalue2>
          CONTINUE
        END_IF
        ;other statements to execute
END_WHILE
```

- **Continue statement**
  - **Causes the remainder of the loop body to be skipped**
  - **In a *while* loop, control is transferred to the top of the loop and the condition tested**
  - **In a *repeat* loop, control is transferred to the bottom of the loop and the condition tested.**
  - **Normal loop processing resumes with the condition test**

Language Elements
Statements

Instrument Control Commands		
keyword	operands	
	first	second
FilterWheel	Filter wheel number (3,4)	Filter wheel encoder position (3,4))
CalLamp	WHITE1 WHITE2 NEON HAK OFF (2)	
Telescope	Telescope ID (3,4)	Telescope Elevation Angle (3,4)
Shutter	Telescope ID (3,4)	Open Closed (2)

Language Elements
Example Instrument Control Statements

```
filterwheel 1 50          ; moves fw to position 3
filterwheel 2 37          ; moves fw half way between two
                          ; positions

calLamp neon              ;illuminates neon lamp
calLamp off               ; extingushes all lamps

telescope 1 22.5          ; positions telescope one 22.5
                          ; degrees below the horizon
shutter1 open             ; opens the shutter on telescope
                          ; one
```

Language Elements
Statements

Instrument Control Commands		
keyword	operands	
	first	second
Load_Scan_Table	file description (2)	
Start_Scan		
Stop_Scan_End		
Stop_Scan_Now		
Load_Bin_Table	Bin Table Index (3,4)	file description (2)

Language Elements
Example Instrument Control Statements

```
load_bin_table 1 "../bins/laserBins1.btab"
load_scan_table "../scans/mltDbasic.scan"
start_scan

repeat
        wait 50
until sys_expose_down_cnt .lt. 5
stop_scan_end
```

Language Elements
Statements

Miscellaneous Commands			
keyword	operands		
	first	second	third
NoOp			
Boot			
NoBoot			
Load_Mem	Intel Hex Format file description (2)		
Write_Byte	address (4)	byte value (3,4)	
Write_Word	address (4)	word value (3,4)	
Write_Double	address (4)	word value (3,4)	word value (3,4)
Dump	address (4)	length (3,4)	

Language Elements
Statements

Miscellaneous Commands			
keyword	operands		
	first	second	third
Calculate_crc	address (4)	length (3,4)	
Start_CP			
Stop_CP			
Run	address (4)		
Clear_CPH			
Load_CPH	file description (2)		
Validate_CPH			
Save CP	PRIMARY SECONDARY (2)		
Allow_WD_Expire			

Language Elements
Example Miscellaneous Statements

```
Load_mem "iswload1_ver_31.hex"      ;loads memory from file
write_byte 3000CH 02H               ;writes a byte
write_word 400H 0203H               ;  word
write_double 32000H 0203H 0405H     ; two words
dump 30002H 64                      ;dumps 64 bytes
calculate_crc 30002H 1000H          ;calculates crc
validate_cph                        ;validates the holding
                                    ; buffer

save_cp primary                     ;copies control program to
                                    ; primary eeprom location

allow_wd_expire                     ;forces transition to
                                    ; boot mode
```

Programming the Instrument Tables

- In addition to control programs scan and binning tables are required
- Scan tables define a sequence of measurements to be performed
  - Scan tables are specified in a text file
  - Specifies the scan in terms of altitude or elevation angle
  - Telescope positions may be defined in unison, pairs or individually
  - The load_scan_table command specifies the active scan table
- Binning tables define the size and gain of the wavelength bins
  - Bins are specified by width, starting at the edge of the CCD most distant from the cone apex
  - Each bin may have a gain associated with it

Tables
Scan Table Example

```
.name       mltDbasic3
.id       11300
.description daytime basic science,v3-reduced exposure times, no crossed filters
.approved    15-Feb0-2002
.scan       altitude
;-------------------------------------------------------------
; col  parameter
; 1   wavelength, anotation
; 2   filter wheel one position 1,2,...8
; 3   filter wheel two position 1,2,...8
; 4   exposure time, seconds
; 5   cal lamp state, off, white1, white2, neon, hak
; 6   exposure count
; 7   Science TM Mode,
;       B(Binned according to binning table)
;       I(50X600 image)
; 8   binning table to use
; 9   telescope id
;       A-all,
;       W-warm side, C-cold side,
;       F-forward, B-Backward
;       1,2,3,4 - specific telescope
;10   starting position, altitude (km) or angle (deg)
;11   final position, altitude or angle
;12   step size, altitude or angle
;13   shutter position, open or close
;
;1     2  3   4   5 6   7   8  9   10    11    12   13
  866.23 3   1  0.8 off 1   B   1   A   55.0 80.0   1.25 open
  000.00 3   1  0.8 off 1   B   1   A   80.0 80.0   1.25 close
  763.78 8   1  0.8 off 1   B   1   A   70.0 95.0   1.25 open
```

Tables
Binning Tables

- **Binning tables are text files**
- **Two programs have been developed to produce them**
  - **eqWaveBT - produces a binning table with variable width bins**
  - **eqAreaBT - produces a binning table with equal area bins**
- **Currently one must manually add a line of the form**
  ;.ccd_h_dump_2 value
- **Binning tables do not apply in image collection mode**
  - **Set ccd_cntl_reg_3 to control gain.**
  - **Values may be 0, 9, 11, 12 to get gains of 160, 40, 10 and 5 e^-/cnt respectively.**

Programming the Instrument
Program Organization

- TICL source code can be compiled to produce
  - Immediate Command Blocks
  - Control Program
- Immediate Command Blocks
  - Sequence of commands executed upon receipt
- Control Programs
  - Execution begins on command
  - Basic structure: nested loops
  - Program execution

Program Organization
Immediate Command Blocks

- **Denoted by the presences of the *.immediate* directive**
- **Source file may not contain**

  program statement

  subroutine statement

  call statement

  local statement

  control structures

- **After compilation, the commands contained in command block file will be executed upon receipt by the flight software**
- **Used to make real-time configuration changes**

Program Organization
Control Programs

- **Denoted by the absence of the *.immediate* compiler directive**
- **Requires the presence of the *program* statement, specifying the control program ID**
- **May contain all commands except**

  boot

  noboot

  load_mem

  run

- **Placed in the control program holding buffer upon receipt**
- **Started by the start_cp command**

Program Organization
Control Program Structure

```
repeat
   do until date changes
      if on day side
         set up ccd
         load binning tables
         load scan table for day
               measurements
      else
         perform telescope lubrication scan
         execute a calibration segment
         set up ccd
         load binning tables
         load scan table for night
               measurements
      endif
         start scan

         do until the next terminator crossing
            if the date has changed, break
            wait 10 seconds
         end do

         stop scanning at end of table

   end do
until holding buffer is valid

start control program in holding buffer
```

Daily Control Program Structure

Program Organization
Control Program Execution

- **Control Programs reside in three locations in the flight software**
  - **Execution buffer**
  - **Holding buffer**
  - **Default control program**
- **Destination for the control block file specified by the *.immediate* directive**
- **Provisions exist for starting a control program upon instrument reset**

Program Organization
Control Program Execution

- **Execution buffer**
  - **loaded with the default buffer contents at instrument software startup**
  - **loaded with the holding buffer contents, if valid, upon the *start_cp* command**
  - **if the *start_cp* command is issued while the holding buffer contents are invalid, control program execution halts**
  - **Commands executed in sequence from this buffer. Commands from the 1553 interface are interleaved with these commands.**
- **Holding buffer**
  - **loaded from the 1553 interface if the source code did not contain the *.immediate* directive**
  - **contents transferred to the execution buffer for interpretation**

Program Organization
Control Program Execution

- **Default Control Program**
  - **Located in eeprom**
  - **Loaded from the control program holding buffer by the *save_cp* command**
  - **Copied to the execution buffer during instrument software initialization**
- **Initialization Sequence**
  - **Boot code waits 5 minutes for a command**
  - **If no command is received, copies eeprom to ram and transfers control to instrument software**
  - **Instrument software begins interpreting the control program in the execution buffer**

Programming the Instrument
TICL Tools

- TICL
  - Instrument language compiler
- ticlBuilder
  - maintains make files
  - invokes compiler
- sendCM
  - packages, encrypts and transmits command messages
- ticlArchiveAdj
  - duplicates archive directories
- package
  - Creates upload messages
  - invoked by sendCM
- reportMsg
  - provides status of messages
- updateMsg
  - updates message log from receipt files
- updMsgDrv.sh
  - processes multiple receipt files
- ticlDecomp
  - Decompiler
  - Interprets tcmd files
- Binning table creators
  - eqWaveBT
  - eqAreaBT

TICL Tools Development Cycle

- Specification of scan tables and binning tables
  - What altitudes and features are to be observed?
  - What gain and wavelength resolution is desired?
- Specification of control program
  - When will scan tables start and end?
  - When will calibrations be performed?
  - When will special operations occur?
- Coding
  - Using favorite text editor create scan table using template
  - Create binning tables (eqWaveBT, eqAreaBT)
  - Using favorite text editor create the control program

TICL Tools Development Cycle

- Compile and debug
  - Compile (ticl)
  - verify as needed with ticlDecomp and visualize_scan (idl)
  - test the resulting code on the engineering model electronics
- Place source code files under mkssi configuration control
  - Build project with ticlBuilder
- Prepare control program for execution
  - Determine the upload window
  - use ticlBuilder to create and archive a command message file

TICL Tools Development Cycle

- **Transmit to the instrument**
  - **Use sendCM**
  - **Check for authentication receipts in ~timed**
    - updMsgDrv.sh reads all messages in directory
    - reportMsg may be used to view details
- **Verify operation**

Programming the Instrument Control Program Examples

- **mltDmidbetaNbasic3.ticl**
  - **Example daily control program**
  - **invokes subroutines for calibration**
  - **invokes scanning tables for day and night science**
- **fltCal.ticl**
  - **Calibration Subroutine**
  - **Included in control program files for modularity**
- **Scan tables**
  - **mltDmidbeta.scan, mltNbasic.scan**
  - **used in example control program**

Programming the Instrument
TICL Compiler

ticl <filename>
  Compiles <filename>.ticl to code in <filename>.tcmd and a listing file in <filename>.list
  If <filename> is omitted, takes source from the standard input stream,
  writes the TCMD to standard output and the listing to the standard error stream.

ticl -o tcmdFile -l listFile -i <includePath> <filename>
  Compiles <filename>.ticl to code in tcmdFile and a listing in listFile
  prefixes <includePath> to the filename given for any .include directives.

ticl -v[ersion] displays the compiler version information

ticl -h[elp] displays (this) option/usage information

Programming the Instrument ticlBuilder

purpose:
maintenance of the TIDI instrument command language control programs.

usage:
To prepare a control program for transmission to the instrument and execution on a particular day, set the current directory to the ticl source directory then execute ticlBuilder as follows

```
ticlBuilder -d edate -s ticlFile [-v]
```

where
-d specifies the date of control program execution which defaults to tomorrow, 2002064
-s specifies the ticl source code file
-v verbose, displays additional diagnostic information

Programming the Instrument
ticlBuilder

purpose:
maintenance of the TIDI instrument command language control programs.

usage:
To maintain the control program directories, set the current directory
to the ticl source code directory and execute ticlBuilder as follows

ticlBuilder -h | -a[r] | -c | -i arcDir

where
-h gives this help message
-a creates a make file for all control programs and
invokes it
-c cleans the directories, removing editor backup files
-r forces complete rebuild
-i indexes scanning and binning table files found in arcDir

Programming the Instrument
sendCM

```
sendCM - formats and transmits a tidi command block file
   Usage:
    sendCM options tcmd_file
    options - packaging options
    tcmd_file - file to transmit

options:
  -s <scriptNname>          CSTOL script sending the commands [I&T]
  -l <scriptLineNumber>     line number in script of commands [I&T]
  -n <scriptVersionNumber>  version of script sending commands [I&T]
  -e <deliveryEnableTime>   first date/time (YYYYDOYHHMMSS) that command
                    delivery may be attempted
  -d <windowDuration>       duration of command window (hh:mm:ss)
                    during which command delivery may be attempted
  -i <deliveryInterval>     number of seconds between successive command
                    messages
```

Programming the Instrument
ticlArchiveAdj

purpose:
Maintains the as-flown archives of TICL programs. It copies an archive directory to a directory for a new date. It is usually used to fill in the archive for days during which no program was up-linked to the instrument.

ticlArchiveAdj source_dir dest_date

where
source_dir is the directory to be copied: YYYYdoyR
with YYYY the year, doy the day of year, and R the archive revision letter
dest_date is the date for which a duplicate archive directory is required.
NOTE: the revision letter is determined by the script

example:
ticlArchiveAdj 2002030A 2002031

creates a new archive directory for day 31 of 2002 using the contents of archive directory 2002030A

Programming the Instrument
ticlDecomp

Purpose:

Decompiles a ticl command block file and produces and interpreted listing

ticlDecomp < tcmdFile > listFile

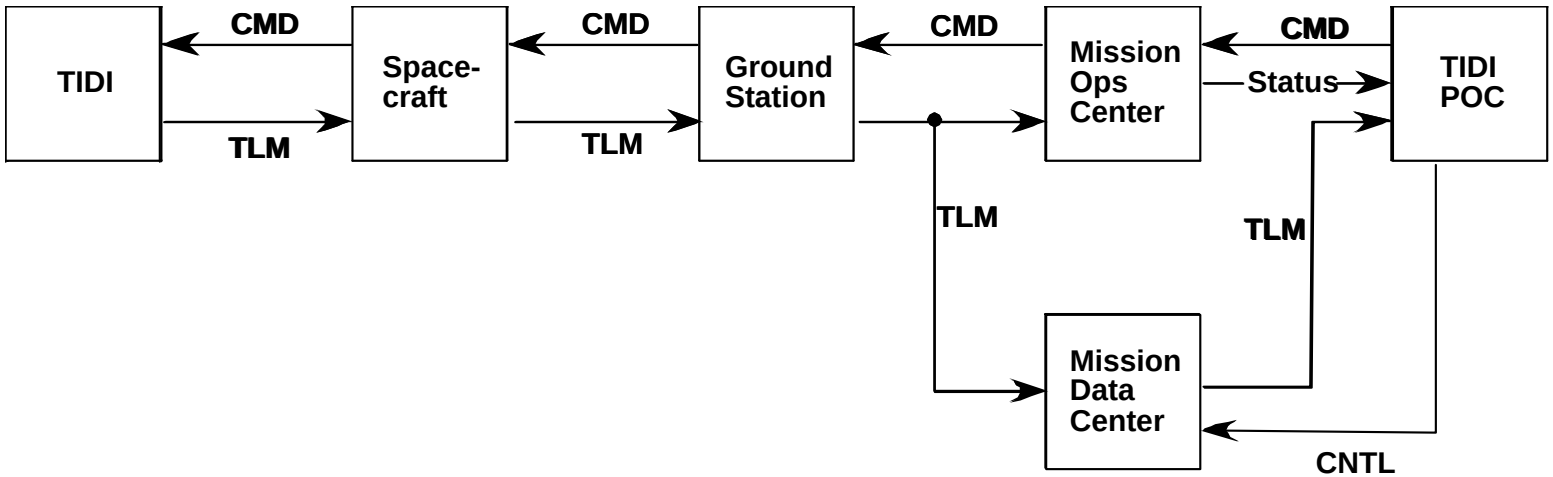
tcmdFile is a command block file produced by the compiler

list file is the interpreted listing. If omitted, the listing is placed on stdout

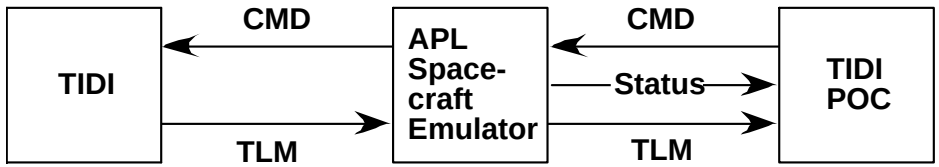
Ground System Components

- **Flight Configuration**
  - **Includes SPRL and APL components**
  - **Geographically dispersed**
- **Engineering Stack Configuration**
- **Component Descriptions**

Ground System Components
Flight Configuration



Ground System Components
Engineering Stack Configuration



Ground System Components

- **EGSE**
  - **Instrument version**
    - Supplies power to instrument when TIDI is separated from the spacecraft
    - Provides 1553 Interface
  - **Engineering Stack**
    - Lab power supply
    - 1553 interface supplied by emulator
- **Spacecraft Emulator (APL Supplied)**
  - **Provides the command and TM path between the engineering stack and the TIDI Ground System**
  - **Emulates**
    - Spacecraft command and TM system
    - TIMED Mission Operations System

Ground System Components

- TIDI Ground System
  - Real-Time Command and Display
  - Telemetry and Data Processing
  - Utility programs
- TIDI Engineering Stack
  - Flight-like electronics
  - Mechanism simulators
  - Dummy Loads

Ground System Components
Real-Time Command and Display (RCDS)

- **The Real-Time Command and Display System is an OASIS-CC application.**
- **Running the TIDI RCDS requires a specific environment specified by the values of a number of environment variables and a specific current directory.**
- **Starting the RCDS requires establishing the environment, starting the OASIS application, and initializing the OASIS session.**
- **Once started, communications with the emulator (or MOC) must be established**
- **Uses the TICL compiler and the sendCM script to create command messages**

Ground System Components
Telemetry and Data Processing

- **tmLogger**
  - **Retrieves telemetry and archives it**
  - **May be started by an RCDS command**
  - **May be started by a UNIX command**
- **Data Processing**
  - **Programs for state of health assessment**
    - trend extractor
    - events
  - **Programs for calibration analysis**
    - analyze
    - epet plotting programs
  - **Programs for science analysis**

Operating the Engineering Stack

- **Uses**
- **Required Components and configuration**
- **Interfaces**
- **Start-up**
- **Emulator Operations**
- **Shutdown**

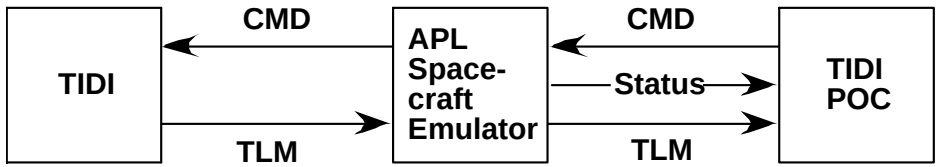
Operating the Engineering Stack
Uses of the Engineering Stack

- **Training**
  - **Once configured, the user interface through RCDS is identical to flight**
  - **Instrument response to commands is flight-like**
  - **Execution time of commands is identical to flight**
- **Testing**
  - **Control programs may be loaded and run**
  - **Spacecraft status message content is controllable**
  - **Orbit simulation is possible**

Operating the Engineering Stack
Required Components

- **Engineering Stack**
- **EGSE - for instrument power and communications**
- **Spacecraft Emulator - to establish the command and TM paths**
- **TM Logger - to archive received telemetry**
- **Real-Time Command and Display System - provides user interface to instrument**
- **Data Processing and Analysis Software as needed**

Engineering Stack Interfaces



Operating the Engineering Stack Interfaces

- From POC to Emulator Command Messages
  - Generated by sendCM
  - Encrypted

  TM Server Commands
  - Generated by TIDI RCDS and tmLogger

- From Emulator to POC Telemetry
  - POC TM Packet format
  - Multiple streams

  Status
  - Authentication Return Receipts
  - Command Return Receipts

- Between Emulator and Stack 1553 Serial Bus
  - Commands
  - Telemetry

Operating the Engineering Stack Start-up

- **Four subsystems are started**
  - **EGSE**
  - **Spacecraft Emulator**
  - **Real-Time Command and Display System**
  - **Instrument Software**
    - On power-up the instrument enters boot-mode
      - Limited command set
      - transitions to instrument operating software after delay or upon specific command
    - Startup complete when instrument software is initialized
      - Instrument software provides full command set
      - TICL programs may be executed

Operating the Engineering Stack Start-up

- **Apply power to engineering stack**
  - **Power supply located on bench below optical bench**
  - **Turn power supply on**
- **Start the spacecraft emulator**
  - **If required, the login ID is *admin* and the domain is *devsys5***
  - **Start the spacecraft emulator by double-clicking the emulator icon on the desktop**
  - **Start the emulation by clicking on the start button (which changes to pause)**
- **These steps may be performed in any order**

Operating the Engineering Stack Start-up

- **Start the Real-Time Command and Display System**
  - **Log into a TIDI workstation as user oasis1(csh) or oasis2 (sh)**
  - **Type the command *go_rcds* to establish the RCDS environment**
  - **Type the command *oasis* to start the RCDS**
  - **Follow the procedures in the Operations Checklist to configure the oasis application**
  - **Establish communications with the emulator by clicking the "Connect to Emulator" item in the "LINKS" menu**

Operating the Engineering Stack Start-up

- **Start the instrument software**
  - **Follow the procedures in the Operations Checklist to boot the instrument software**
  - **When the software status changes from BOOTING to RUNNING, the engineering stack is ready to execute TICL commands and control programs**

Operating the Engineering Stack
Spacecraft Emulator Operations

- **User interface is provided by the emulator program**
  - **To start emulation**
    click on the start button (which changes to pause)
  - **To suspend emulation**
    click on the pause button (which changes to resume)
  - **To resume emulation**
    click on the resume button (which changes to pause)
  - **To terminator emulation**
    click on the terminate button

Operating the Engineering Stack
Spacecraft Emulator Operations

- **Proper Operation is indicated by a flashing green tell-tale**
- **Upon receipt of a command message from the TIDI_RCDS, the terminal “beep” will sound**
- **There may be 1553 errors reported during the instrument software boot-up sequence**
- **The time displayed on the emulator panel is incorrect. The mission time incorporated in the TM packets is derived from the value of the NT system clock, which is set to UTC.**
- **The emulator does not incorporate leap seconds into the mission time recorded in the TM resulting in a 12 second offset.**

Operating the Engineering Stack
Spacecraft Emulator Operations

- **The contents of the spacecraft status message is controlled by check-boxes in the emulator window**
- **Each command message received is stored.**
- **No automatic housekeeping.**
  - **Check the contents of the sub-folders contained in *C:\poc_ftp*.**
    All files in these directories may be deleted at any time. In particular the *arc* folder fills rapidly. Preserve the contents of the *err* sub-folder if anomalous behavior is noted.
  - **Check the contents of the log folder *C:\emulatorrelease2.3A\log_files***
    All files may be deleted at any time. Preserve copies if anomalous behavior is noted.

Operating the Engineering Stack Interrupting Operations - 1

- **It may be desirable to remove power from the stack without shutting down the ground system or causing any ground system component to experience errors**
- **Prior to removing power, place the spacecraft emulator in the pause mode by clicking the pause button on the interface**
- **The emulator will maintain the network connection to the TIDI RCDS and to the TM Logger**

Continued next slide

Operating the Engineering Stack
Interrupting Operations - 2

- **If the interruption will be lengthy, the OASIS “awaiting data” messages can be terminated by selecting the “Disconnect” button from the “Link Control” menu**

- **Operations may be resumed by re-applying power then placing the emulator back into run mode by clicking the resume button.**

- **If required, communications with the emulator are reestablished by selecting the “Connect to Emulator” button from the “Link Control” menu**

Operating the Engineering Stack Shut-down

- **The ground system can be gracefully shut down by placing the emulator in pause mode by clicking on the “pause” button before removing instrument power.**
- **OASIS is shut down by selecting the “QUIT” button from the master panel**
- **The emulator is exited by closing the user interface window, terminating the emulator program and all of its helpers.**

TIDI Real Time Command and Display

- **Introduction to OASIS**
- **CSTOL**
- **TIDI RCDS Panels**
- **Alphanumeric Panels**
- **Graphical Displays**
- **Instrument Commanding**

TIDI Real Time Command and Display Introduction to OASIS

- **The TIDI Real Time Command and Display System is implemented as an OASIS-CC application**
- **OASIS provides telemetry decomposition, engineering unit conversion, limit checking, and a number of display options**
- **OASIS provides a scripting facility (CSTOL) and a graphical user interface for commanding and for operator control of displays**
- **Fundamental Data Types:**
  - **dn_data - data number data or raw TM values**
  - **eu_data - engineering unit data, dn_data converted to engineering units as specified by an analog conversions table**
  - **states - data number data converted to labeled states as specified by a state conversion table.**

TIDI Real Time Command and Display Introduction to OASIS

- **Data item naming convention**
  - **Each global data item is identified by**
    - an external element name (ee)
    - an item name (in)
  - **The flight software mnemonic is formed by concatenation *ee_in***
- **Global data items are extracted from**
  - **spacecraft TM**
  - **instrument housekeeping TM (TIDI type 4 packet)**
  - **science TM (TIDI type 0 packet)**
  - **acknowledgment and error message packets**

TIDI Real Time Command and Display CSTOL

- Command Line User interface for OASIS
- Executes commands, procedures and macros
- Multi-threaded
  - More than one simultaneous procedure may be run
  - Procedures may be aborted, suspended and restarted by user commands
- Commands
  - Control procedures
  - Display data
  - Configure OASIS
  - Evaluate Expressions

TIDI Real Time Command and Display CSTOL

- **Procedure Control Commands**
  - **start proc_name argument_list**
    - Proc_name must be entered in the procedures data table
    - Arguments are separated by commas
  - **wait at line | label**
    - Inserts a breakpoint in an executing procedure
    - Canceled by the go statement
  - **go**
    - Resumes execution of a suspended procedure
  - **return [all]**
    - Stops the executing procedure and returns to its caller
    - All parameter stops all active procedures in thread

TIDI Real Time Command and Display CSTOL

- **Data Display Commands**
  - **write *write_list***
    - Writes a message into the event message log
    - write_list is a comma separated list of expressions to evaluate and write
    - Items in the list may be globals, locals or strings
  - **fwrite global**
    - Writes a message into the event log
    - Message contains both the raw value of the item and its engineering conversion
    - Command is a macro

TIDI Real Time Command and Display
TIDI RCDS Panels

- **Alphanumeric display panels - provide instrument subsystem status**
- **Graphical display panels**
  - **Strip Chart Panels (5) - Operator configurable displays of any instrument parameter as a function of time**
  - **Histogram Spectral Displays (5) - Operator configurable display of science packet data**
- **Instrument command dialog boxes - provide “point & click” interface to many instrument commands**

TIDI Real Time Command and Display
TIDI RCDS Panels

- **Any panel can be printed to the color printer using the print_window function**
  - **Bring the panel to print to the foreground**
  - **Click the print window icon on the workspace dock**
  - **when the cursor switches to a cross-hair, click in the panel to be printed**

TIDI Real Time Command and Display Alphanumeric Panels

- Panels are made visible by selecting the “panel name” button from the “Panels” menu
- The Wildcard panel permits the display of any data item in either raw or converted form
  - Select “Wildcard” button from the “Panels” menu
  - Select the “modify” button on the “Wildcard” page
  - Specify a row number, the external element, and the item name
  - For data type enter either eu_data or dn_data
  - Specify a FORTRAN style display format: Xn, In, Fn.M, En.m for numeric data
  - Specify a FORTRAN style character format to display the eu_data version of a state variable: An

TIDI Real Time Command and Display
Graphical Spectra Display

- Each spectra display displays a histogram of up to 50 columns
- Selected in the “Graphics” sub-panel
- The scale factor and wave length bins to be displayed are configured by the operator
  - The default maximum signal level is 4095 counts
  - Each spectra display displays 50 consecutive bins

TIDI Real Time Command and Display Strip Chart Graphical Display

- Each strip chart displays up to 4 channels of data
- Each strip chart displays a 600 second window of a 1800 second history
- Items for display are selected by the operator
  - Specify the external element and item name
  - Specify the data type, eu_data or dn_data

TIDI Real Time Command and Display
Instrument Commanding

- **Instrument commanding is supported by a series of dialog panels**
- **Functions supported are:**

Function	Panel Name
Control of boot	Boot Code Commands
Loading TCMD files	Stored Commands
Control of mechanisms	Mechanism Control
Control of heaters	Heater Duty Cycle
Memory access, loading & CRC	Memory Commands
Control program operations	Control Program Commands
Scanning & binning operations	Scanning Binning

TIDI Real Time Command and Display Instrument Commanding

- **Pre-compiled command block files**
  - **Select “Stored Commands” from the “Command” Menu**
  - **Select a command block file from the scrolling list**
  - **The selected file will appear in a text box at the bottom of the panel**
  - **Click send**
  - **A blue window will appear reporting the progress of the command transmission**
- **Using the GUI**
  - **Select a category of commands from the “Command” Menu**
  - **Complete the text boxes or select options for the desired command**
  - **When completed, click send**
  - **The blue window will again appear**

Mission Operations

- **Facilities Involved**
- **Interfaces**
- **Activities**
- **Communications Net Protocol**

Mission Operations Facilities Involved

- Ground Station
  - Primary is the APL facility
  - Provides up and down link communications
  - Alternates provided by Universal Space Net (USN)
- Mission Operations Center (MOC)
  - Located at APL
  - Staffed by S/C operations team
  - Operates S/C
  - Operates Ground Station

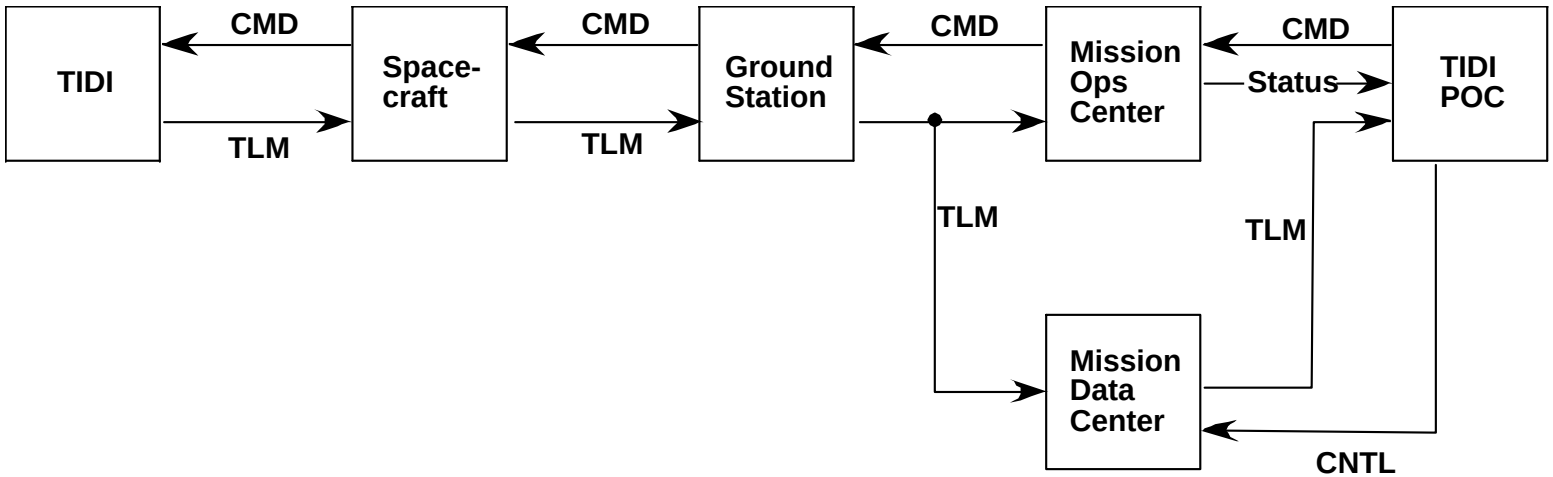
Mission Operations Facilities Involved

- **Mission Data Center**
  - **Located at APL**
  - **Staffed by Project data management staff**
  - **Acquires, archives and serves telemetry**
  - **Produces, archives and serves planning products**
  - **Hosts project data catalog**
- **Test Conductors station**
  - **Located with the spacecraft during ground operations**
  - **Staffed by APL integration and test personnel**
  - **Operates spacecraft GSE**
    - Orbit and attitude simulators
    - Solar array simulators
    - Power supply

Mission Operations Facilities Involved

- **Instrument Payload Operations Centers (POCs)**
  - **One for each instrument remote from APL**
  - **Staffed by instrument team members**
  - **Operates the instrument**
  - **Acquires instrument data from MDC**
  - **Processes data and serves results**

Ground System Components Interfaces



Mission Operations Interfaces

- **From POC to MOC**
  Command Messages
  - **Generated by sendCM**
  - **Encrypted**
  - **Queued at the MOC**
    - staging queue by enable time
    - Execution queue by disable time
- **From POC to MDC**
  TM Server Commands
  - **Generated by TIDI RCDS and tmLogger**
  - **Can select different sources**
    - recorder playback
    - Realtime data
    - RF or Baseband (ground ops)
- **From MDC to POC**
  Telemetry
  - **POC TM Packet format**
  - **Multiple streams**
- **From MOC to POC**
  Status
  - **Authentication Return Receipts**
  - **Command Return Receipts**

Mission Operations Activities

- **Instrument integration and test (complete)**
- **Launch preparations (complete)**
  - **Test spacecraft and instruments to verify their health before launch**
  - **Configure spacecraft and instruments for launch**
  - **Terminal countdown operations**

Mission Operations Activities

- **Initial Operations**
  - **Reconfigure spacecraft and instruments from launch configuration to flight configuration**
  - **Confirm that the spacecraft and instruments survived launch**
- **Routine Operations**
  - **Monitor state of the instrument**
  - **Collect scientifically useful data**
  - **Reduce and analyze telemetry**

Mission Operations Activities

- **"TIDI Operations Checklists" provides procedures for all mission operations functions**
- **All operations are to be logged in the operations logbook**
- **Pre-Launch - completed**
  - **Activities included**
    - spacecraft functionals
    - Stand alone epets
    - Mission simulations
    - Special Tests
- **Initial Operations - completed**

  Rational and plan in "TIDI Instrument Activation Scenario**"**

- **Routine Operations**
  - **Daily contacts**
  - **Activities include**
    - Science data collection
    - Calibrations
    - Routine data processing
    - Delivery of time lines & PANs
- **Real Time Operations**
  - **Support non-routine operations**
  - **Activities include**
    - assess instrument health
    - real-time commanding
    - instrument configuration to support spacecraft operations

Mission Operations
The Three Rules

- **Don’t Panic !**
- **Plan the flight -**
  Fly the plan.
- **If you don’t know what to do -**
  do nothing!

Mission Operations
Routine Operations

- **Uplink**
  - **Time line**
  - **Actions**
- **Downlink**
  - **Time line**
  - **Actions**

Uplink Time Line

Time	Operation
D – 8 weeks D – 4 weeks	Receive predicted orbital ephemeredes
D – 4 weeks	Produce viewing geometry predictions Specify measurements
D – 4 weeks	Determine correlative measurement opportunities Transmit overpass predictions to correlative sites Transmit planned time line to TIMED MDC
D – 3 weeks	Complete specification of instrument control program
D – 2 weeks	Complete coding and simulator verification of control program
D – 1 week	Transfer control program to TIMED control center for upload
D – 1 day	MOCC uploads TIDI control program at any time during the day
0	Operational Day, execute command program currently in instrument

Downlink Time Line

Time	Operation
	Receive near real time (NRT) data Receive previous day's playback (PB)
Data Receipt + 1 hour	Complete automated limit checking completed on NRT data
Data Receipt + 1 hour	Confirm receipt of uploads in NRT data
Data Receipt + 2 hours	Complete automated production of quicklook plots and review
Data Receipt + 12 hours	Complete limit checking on PB data
Data Receipt + 24 hours	Complete routine processing of PB data Confirm production output files Post product availability notices (PANS) Post actual timelines
During initial operations Data Receipt + 24 hours Thereafter, once each week	Examine diagnostic plots: mechanism state plots sample spectra engineering trend data daily wind maps
Once each week	Review calibration results Review summary science products Adjust data quality indicators in production data files

Mission Operations
Routine Operations

Mission Operations
Routine Operations

Mission Operations
Initial Operations

Voice Communications

- Voice network characteristics
  - Multi-channel
  - Party-line
  - All communications are recorded
- Usage
  - Dial into a remote access number
  - Enter passcode
  - Select channel
  - Details may be found in the procedures notebook

Voice Communications
Call Signs

- **TIDI** **us**
- **TIMED MOC** **TIMED test conductor at the MOC**
- **ITC** **TIMED test conductor at the spacecraft**
- **ACOM** **APL Communications Support Engineer (W. Dove)**

Voice Communications Channels

Identifier	description	netcode (T/L)
• S/C1	Launch Operations channel	10/20
• MOPS2	Mission Operations Channel 2	14/24
• MOPS3	Mission Operations Channel 4	15/25
• MOPS4	Mission Operations Channel 4	16/26
	Clear last selection	00
	Clear all selections	99

Voice Communications Procedures

- **Listen before speaking, don't cut into a dialog.**
- **Announce your presence when first joining the network**
  TIMED MOC, TIDI is on the net
- **Identify the party you wish to communicate with**
  TIMED MOC, this is TIDI on S/C1
- **Use the appropriate channel after establishing contact on S/C1**
- **Let the MOCTC know if you will be off the network**
  TIMED MOC, TIDI will be off station for a few minutes

Continued next slide

Voice Communications Procedures

- **Announce the completion of your conversation**

  TIDI is clear

- **Speak slowly, clearly and to the point**

- **Be polite (you never know who may be listening)**

- **Use the phonetic alphabet to indicate letters**
  “alpha”, “bravo”, …, “Quebec”, …, “whiskey”, ‘x-ray”, “zulu”

Continued next slide

Voice Communications Procedures

- **Phraseology**
  - **Affirmative for “yes”**
  - **Negative for “no”**
  - **Stand-by for “hold on”**
  - **Copy for “I understand”**
  - **Correct for “you are right”**
  - **Please repeat for “what did you say?”**
- **The word “HOLD” should never cross your lips (unless you want to delay the launch and become infamous)**