# MODULUS Ptolemy

# On Board Software User Manual

Document: RO-LPT-RAL-MA-3102

25th February 2002

Prepared by:

D.L.Drummond

……………………………..

Modulus Team

Approved by:

……………………………..

Authorised by:

……………………………..

CHANGE RECORD

| DATE | CHANGE DETAILS | ISSUE |
|---|---|---|
| 25//02/2002 | First issue | 1 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# CONTENTS

1.1 INTRODUCTION ... 4
1.2 INTENDED READERSHIP ... 4
1.3 APPLICABILITY ... 4
1.4 PURPOSE ... 4
*1.4.1 Of document ... 4*
*1.4.2 Of Software ... 4*
1.5 HOW TO USE THIS DOCUMENT ... 4
1.6 RELATED DOCUMENTS ... 5
1.7 CONVENTIONS ... 5
1.8 PROBLEM REPORTING ... 5

**2 OVERVIEW ... 6**

2.1 CONTEXT ... 6
2.2 TOP LEVEL STRUCTURE AND CONTROL ... 7
2.3 MEMORY USAGE: ... 9

**3 INSTALLATION ... 10**

**4 USING THE SOFTWARE ... 10**

**5 REFERENCE ... 11**

5.1 MODE EVENTS ... 11
*5.1.1 Clippard Valve Mode Event (ID=0): ... 12*
*5.1.2 Injector Control Valve Mode Event (ID=3): ... 12*
*5.1.3 Lindau Valve Mode Event (ID=5): ... 13*
*5.1.4 Reactor Mode Event (ID=6): ... 14*
*5.1.5 Mass Spectrometer Operation Mode Event (ID=9): ... 15*
*5.1.6 Heater Mode Event (ID=10): ... 16*
*5.1.7 SMA heater Mode Event (ID=12) ... 16*
*5.1.8 Pressure Monitoring Mode Event (ID=17): ... 17*
*5.1.9 Temperature Monitoring Mode Event (ID=18): ... 18*
*5.1.10 Auxiliary Data Mode Event (ID=20) ... 19*
*5.1.11 Time Delay Mode Event (ID=24): ... 19*
*5.1.12 Sample Collector Mode Event (ID=33) ... 19*
*5.1.13 Mass Spectrometer Activation Mode Event (ID=34) ... 20*
*5.1.14 WGA Memory Check Mode Event (ID=36) ... 20*
*5.1.15 Timer Mode Event (ID=40): ... 20*
*5.1.16 Set Injector Control Valve Opening Duration Mode Event (ID=48): ... 21*
*5.1.17 RF Frequency Calibration Mode Event (ID=65) ... 21*
*5.1.18 End of Mode Event (ID=65) ... 21*
5.2 TABLES ... 22
*5.2.1 Scan Function ... 22*
5.3 PATCHES ... 23
*5.3.1 SAFE mode HK collection patch ... 23*
*5.3.2 Temperature Servo PID patch ... 23*

## 1.1 Introduction

This is the software user manual for the Ptolemy On-Board Software. This software is installed in fuse-link PROMs in the Ptolemy flight model experiment on the Rosetta Lander.

## 1.2 Intended Readership

The intended readership includes those preparing TCs and procedures for the operation of any Ptolemy model, either in flight or in ground tests. An understanding of the operation of the gas chromatograph and ion-trap mass spectrometer is assumed as well as an appreciation of the context of the Ptolemy instrument on the lander.

## 1.3 Applicability

This manual applies to the FM version of the software dated 06/04/2002 – this is the only release containing both the safe-mode and science-mode software and is the version held in the Ptolemy FM fuse link PROMs.

## 1.4 Purpose

### *1.4.1 Of document*

The purpose of this document is to provide an overview of the Ptolemy on-board software, bringing together information from the documents listed in 1.5

### *1.4.2 Of Software*

The purpose of the Ptolemy on-board software is to provide control and monitoring of the Ptolemy experiment, executing the required experiment sequences on the comet. Its main functions are listed below:

- Collection of experiment Housekeeping data and monitoring instrument safety
- Collection, collation and compression of measurement data
- Handling and execution of telecommands from the lander
- Generation of science and Housekeeping telemetry and transmission to the lander
- Control and sequencing of the experiment science system, of the ion-trap and of the oven docking system.

## 1.5 How to use this document

Detailed information on the Ptolemy On-board software is to be found in the documents named in 1.5 This document is intended as an introduction to the sfotware and its related documentation.

### 1.6 Related Documents

The following documents contain much of the information required for the operation of the Ptolemy on-board software. References shall be made to sections in these documents where required. Where this is more convenient, information from these documents may be repeated here.

| | | |
|---|---|---|
| RD1 | RO-LPT-RS-3103/EID B | Experiment Interface Document part B -<br>includes an overview of the instrument as well as a detailed description of Ptolemy TCs and TM formats (section 4) |
| RD2 | RO-LPT-OU-DP-3205 | Section 2.8/2.9 give an overview of the Ptolemy electronics and science system. |
| RD3 | RO-LPT-OU-TN-3401 | Ptolemy Hardware/Software Interface Document –<br>describes the Ptolemy electronics hardware from the software viewpoint. |
| RD4 | RO-LPT-OU-RS-3401 | Modulus Ptolemy Software User Requirements Document –<br>High-level overview of user requirements including instrument context. |
| RD5 | RO-LPT-RAL-RS-3403 | Ptolemy Safe Mode Software User Requirements Document –<br>more detailed user requirements analysis for Safe-mode software |
| RD6 | RO-LPT-RAL-RS-3405 | Ptolemy Science Mode Software User Requirements Document –<br>more detailed user requirements analysis for Science mode software |
| RD7 | RO-LPT-OU-PL-3101 | Ptolemy Operations Plan – includes a brief description of the modes for the experiment |
| RD8 | RO-LPT-RAL-TN-3403 | Ptolemy Telecommand and Telemetry Definitions – describes the TC and TM packets in detail – this information is duplicated in RD2 2.8 & 2.9 |
| RD9 | RO-LPT-RAL-LI-3404 | Ptolemy FM EEPROM Memory Map |

### 1.7 Conventions

Data bits are numbered with bit 0 as the least significant bit (the opposite convention is used in TM/TC documents).

The Ptolemy processor is configured with a “big-endian” architecture; words are stored MS byte first and long words are stored MS word first. Intel processors use the opposite (“little-endian”) form.

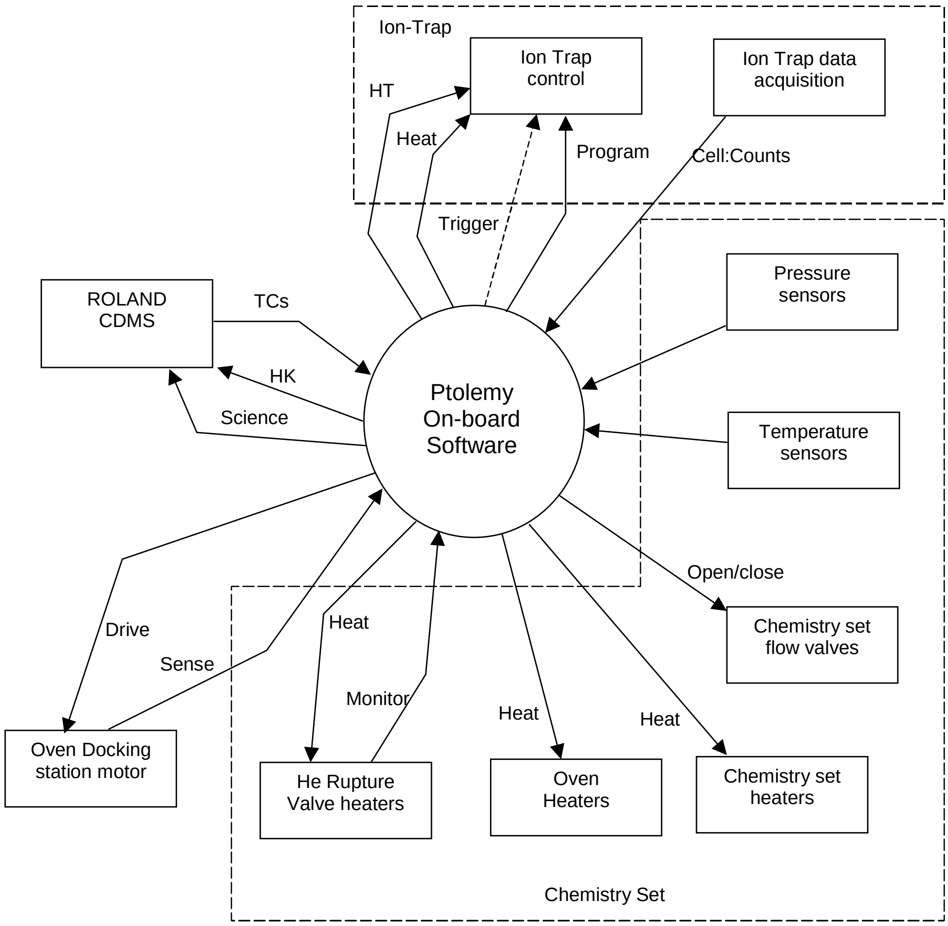
### 1.8 Problem reporting

Problem reporting shall be via the same NCR process used for Ptolemy hardware problems

# 2 OVERVIEW

## 2.1 Context

A context diagram for the Ptolemy On Board Software is shown here:



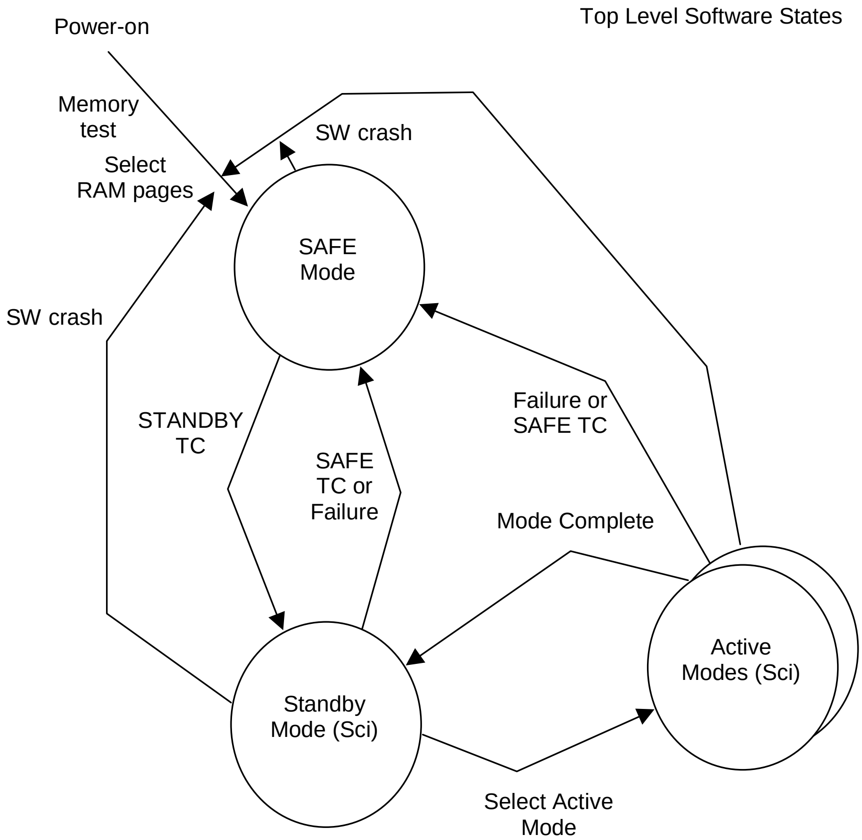

| | |
|---|---|
| Roland CDMS | Forwards TCs to Ptolemy, acquires HK words and receives science data frames from Ptolemy |
| Oven Docking Station Motor | Software drives motor to dock and undock from oven on carousel |
| He Rupture Valves | Valve is heated to open a Helium tank shortly before start of operations |
| Oven Heaters | Pulse width modulated by software |
| Chemistry set Heaters | Various heaters for reactors, pipes etc. PWM by software |
| Chemistry set flow valves | Used to control flow of gas through chemistry set. Two types: Level controlled (Clippard & Injector) and Lindau (controlled by DAC heater) |
| Temperature sensors | All sensors are thermocouples, AD590 sensor monitors reference junction |
| Pressure sensors | Used to monitor pressure in pipes, manifolds etc. |
| Ion Trap data acquisition | Acquires channel:count pairs from Ion Trap and presents them to software in a FIFO. |
| Ion Trap control | Ion trap is controlled by a WGA (waveform generator) which is programmed via I2C interface by software. DACs for level settings must also be programmed. One of the heaters is applied to the Ion Trap |

### 2.2 Top level structure and control

The Ptolemy On-board software has two major components; the Safe Mode software and the Science Mode software:

| Component | Function |
|---|---|
| Safe Mode | • Entered at power-on<br>• Performs power-on checks; locates page of working RAM<br>• Responds to HK requests from CDMS<br>• Science System in safe state with most drives etc. off.<br>• Allows patch/dump TCs and comanded transition to Science Mode (Standby) |
| Science Mode | • Entered by command from Safe mode<br>• Always starts in Standby<br>• Responds to HK requests from CDMS<br>• No patch/dump<br>• Allows commanded execution of any of the science modes or commanded transition to Safe Mode<br>• Sends science data frames when in on of the active science modes |

The top level software states and transitions are shown here:



**Power-on:**

At power-on, the software first performs a test of each RAM page (there are 8 of these, pages 8-15, each of size 64k bytes). The first working page of RAM is selected for the SAFE mode working data memory (this will usually be page 8). Safe mode then begins operation.

**Safe Mode:**

On entering Safe mode, the software generates a "Ptolemy Power-on Start" event packet (RD8 3.6) and then begins normal Safe-mode processing; housekeeping data is provided to the CDMS as requested, filling the 256 word frame with concise HK packets where no event/acknowledge packets need to be transmitted.

The following TCs are accepted:

- CONNECTION TEST (RD8 2.5) – responds with an acknowledge packet in HK frame
- DUMP MEMORY (RD8 2.3) – produce dumps of PROM, EEPROM or RAM in the Housekeeping frame.
- LOAD MEMORY (RD8 2.2) – Allows patching to RAM or EEPROM (any page)
- CHECK MEMORY (RD8 2.4) – Responds with checksum over a memory block
- COPY MEMORY (RD8 2.6) – allows copying of data between blocks in PROM, EEPROM or RAM

At this stage, the science mode software may be copied from PROM into RAM (by default, to page 9). After this, patches may be loaded directly into RAM or copied from EEPROM. If no patching is required, the science mode software may be executed in PROM, without copying to RAM.

The Science mode software is started, either in PROM or in RAM, by issuing the STANDBY TC. A mode selection event packet (RD8 3.6) shall be generated for inclusion in the next HK frame

**Standby Mode:**

Standby mode and the active science modes have a separate code page from Safe mode, they share no code but do share the same data structures (in the selected data page). It is in Standby mode that hazardous functions required for subsequent operations must be enabled. From Standby mode, the software may be commanded either back to Safe mode or to one of the 16 active Science modes. TC accepted in Standby are:

- SAFE – return to Safe mode
- PARAMETER UPDATE – update a parameter in a software table
- HAZARDOUS FUNCTION – enable selected hazardous functions
- Select active Science Mode – MODE SELECT TC into any of the 16 active Science modes

Each of the 16 active Science modes are started from Standby mode by the MODE SELECT TC. The selected mode runs to completion and autonomously returns to Standby mode

**Active Modes:**

These active "Science" modes are invoked from Standby mode. For each Science mode, a Mode Event Sequence is defined; a table of Mode Events that determines the sequence of actions (heating, valve operation, ion trap data acquisition etc.) that are to be performed for this mode. Each Mode Event is a high-level instruction to the software, the sequence terminates with a Mode Event that commands the software back to Standby mode. The Mode Event Sequences may be thought of as high-level programs that are interpreted by the Ptolemy on-board software. The tables are stored by default in EEPROM, one for each Science mode.

In Science modes, the only TC accepted is SAFE; this allows a return to Safe mode, aborting the current Science mode.

Each Science mode has a separate set of limits for the analogue parameters. Violation of a limit causes the software to re-enter Safe mode.

The nominal completion of an active science mode occurs when the End-of-Mode mode event is encountered and the software returns to Standby model

A detailed description of each Science mode is to be found in RD7.

### 2.3 Memory Usage:

| Type/Page | Address Range | |
|---|---|---|
| PROM page 0 | 00000-0FFFF | SAFE mode code and initialisation for SAFE/SCI data |
| PROM page 1 | 10000-1FFFF | SCI modes code, may be executed here or relocated |
| I/O page 2 | 20000-2BFFF | Device registers for CPU Card |
| | 2C000-2FFFF | Spacecraft interface (TM/TC) |
| I/O page 3 | 30000-33FFF | not used by Ptolemy |
| | 34000-37FFF | RICA (Ion trap control/acquisition ASCI) |
| | 38000-3FFFF | Other peripheral registers |
| EEPROM page 4 | 40000-4FFFF | Normal Operating limts, Ion Trap tables and Scan functions for the 16 operating modes |
| EEPROM page 5 | 50000-5FFFF | Mode event sequences for the 16 active science modes |
| EEPROM page 6 | 60000-6FFFF | Reserved for software updates |
| EEPROM page 7 | 70000-7FFFF | Tables of PID constants for Science modes |
| RAM page 8 | 80000-8FFFF | Shared data for SAFE and SCI modes |
| RAM page 9 | 90000-9FFFF | SCI mode S/W may be relocated here for patching |
| RAM page 10 | A0000-AFFFF | |
| RAM page 11 | B0000-BFFFF | |
| RAM page 12 | C0000-CFFFF | |
| RAM page 13 | D0000-DFFFF | |
| RAM page 14 | E0000-EFFFF | |
| RAM page 15 | F0000-FFFFF | |

The current usage and contents of the EEPROM pages are documented in RD9

## 3 INSTALLATION

This section is not applicable for the Ptolemy on-board-software as this is an embedded product, installed in fuse-link PROMs on the flight model.

## 4 USING THE SOFTWARE

This is an embedded, real-time component rather than an interactive software product. The use of the software is generally covered by operations or test procedures as outlined in RD7

# 5 REFERENCE

## 5.1 Mode Events

The mode events can have a length of one to six bytes, depending on the number of parameters. The first and second bytes follow a fixed format:

**First byte of Mode Event:**

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | Mode Event Type ID | | | | | | | Sense |

Mode Event Type: 18 mode events are defined, each is given a 7 bit type identifier
Sense: In some mode events this bit is used to specify e.g dock/undock

**Second byte of Mode Event (if present):**

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | Device ID | | | | | | | State |

Device ID: If required, identifies a device (reactor, valve etc.) to which the mode event refers
State: May be used to specify the sate of the device (e.g open/closed)

**Mode Event Types:**

The mode event type IDs are listed in the following table. The IDs are numbered so that a single bit change cannot produce another legal ID.

| Mode Event Type ID | Mode Event | |
|---|---|---|
| 0 | Clippard Valve Mode Event | |
| 3 | Injector Control Valve Mode Event | |
| 5 | Lindau Valve Mode Event | |
| 6 | Reactor Mode Event | |
| 9 | Mass Spectrometer Operation Mode Event | |
| 10 | Heater Mode Event | |
| 12 | SMA heater Mode Event | |
| 17 | Pressure monitoring Mode Event | |
| 18 | Temperature monitoring Mode Event | |
| 20 | Auxiliary Data Mode Event | |
| 24 | Time Delay Mode Event | |
| 33 | Sample Collector Mode Event | |
| 34 | Mass Spectrometer Activation Mode Event | |
| 36 | WGA Memory check Mode Event | |
| 40 | Timer Mode Event | |
| 48 | Set Injector Control Valve opening duration Mode Event | |
| 65 | RF Frequency Calibration Mode Event | |
| 127 | End of Mode Mode Event | |

### *5.1.1 Clippard Valve Mode Event (ID=0):*

Length: 2 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 0 | | | | | | | unused |
| 1 | Valve number | | | | | | | Op/Cls |

| | | |
|---|---|---|
| Valve Number | Number of valve to operate:<br><br>This mode event is used to command a Clippard valve open or closed | 0 : V1<br>1 : V2<br>2 : V3<br>3 : V4<br>4 : valve deleted<br>5 : valve deleted<br>6 : V7<br>7 : V8<br>8 : V9<br>9 : V10<br>10 : V11<br>11 : valve deleted<br>12 : V13<br>13 : valve deleted<br>14 : V15<br>15 : V16 |
| Op/Cls | 1 for open, 0 for close | |

### *5.1.2 Injector Control Valve Mode Event (ID=3):*

Length: 2 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 3 | | | | | | | unused |
| 1 | Valve number | | | | | | | Not used |

The indicated injector valve is pulsed open for the duration specified by the Set-injector-control-valve-opening-duration mode event (ID 48).

| | | |
|---|---|---|
| Valve Number | Number of valve to operate: | 16 : VA<br>17 : VB<br>18 : VC |
| L S bit | Not used for this mode event | |

#### *5.1.3 Lindau Valve Mode Event (ID=5):*

Length: 2 bytes for close, 6 bytes for open

| | Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
| 0 | 5 | | | | | | | unused | Always Present |
| 1 | Valve number | | | | | | | Opn/Cls | |
| 2 | Target Pressure (high byte) | | | | | | | | Present only for Open |
| 3 | Target Pressure (low byte) | | | | | | | | |
| 4 | Pressure Sensor | | | | | | | | |
| 5 | Max power | | | | | | | | |

For Close commands, heating of the Lindau valve and monitoring of the pressure are stopped. For Open commands, the software begins heating the Lindau valve, using a PID algorithm to keep the designated pressure sensor reading the target pressure.

| | | |
|---|---|---|
| Valve Number | Number of valve to operate: | 24: LV1<br>25: LV2<br>26: valve deleted<br>27: valve deleted<br>28: LV5<br>29: LV6<br>30: LV7 |
| Opn/Cls | 1:open, 0:close | |
| Target Pressure | The target pressure (in telemetry counts for sensor) | |
| Pressure Sensor | Designates which pressure sensor to monitor: | 32: G1<br>33: G2<br>34: G3<br>35: G4<br>36: G5 |
| Max Power | Maximum Power (in DAC counts) that PID may demand | |

### *5.1.4 Reactor Mode Event (ID=6):*

Length: 2 bytes for end-heating, 6 bytes for start-heating

| | Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
| 0 | 6 | | | | | | | | Always Present |
| 1 | Reactor number | | | | | | | Beg/End | |
| 2 | Target Temperature (high byte) | | | | | | | | Present only for Begin |
| 3 | Target Temperature (low byte) | | | | | | | | |
| 4 | Start of Power Window | | | | | | | | |
| 5 | End of Power Window | | | | | | | | |

For end heating commands, heating of the reactor (or oven) and monitoring of the temperature are stopped. For begin heating commands, the software begins heating the reactor/oven using a PID algorithm to keep the reactor/oven temperature sensor reading the target value.

| | | |
|---|---|---|
| Reactor Number | Number of reactor to heat: | 40: R1<br>41: R2<br>42: reactor deleted<br>43: R4<br>44: R5<br>45: R6<br>46: R7<br>47: R8<br>48: R9<br>49-51: no reactors<br>52: R13<br>53: R14<br>54: R15<br>55: OVEN |
| Beg/End | 1: Begin heating 0: End heating | |
| Target Temperature | The target temperature (in tlm counts for sensor) | |
| Start of power window | In which PWM slot the heating pulse is to start | |
| End of power window | Latest PWM slot in which heating pulse must end | |

The power window specifies the range of (1024Hz) time slots in the 4Hz pulse-width-modulation cycle that the PID algorithm may use for switching power to the reactor/oven heater. The PID varies the width of the power pulse by altering the end slot, keeping the start slot the same. The pulse width is limited to that allowed by the specified “Power Window”.

### 5.1.5 *Mass Spectrometer Operation Mode Event (ID=9):*

Length: 6 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 9 | | | | | | | 0 |
| 1 | Scan Function Number | | | | | | | |
| 2 | Ion Trap Electrical Characteristics Table Number | | | | | | | |
| 3 | Number of Mass Spectra (high byte) | | | | | | | |
| 4 | Number of Mass Spectra (low byte) | | | | | | | |
| 5 | Spectrum Repeat Period | | | | | | | |

This mode event is used to set up the ion trap and associated electronics for a particular collection mode; operation is started and stopped using the ion-trap activate/deactivate mode event (34)

| | |
|---|---|
| Scan Function Number | The number (0-7) of the scan function to be started. |
| Ion Trap Elec. Char. Table Number | Number (index) of the table of Ion trap set-up values etc. to use |
| Number of Mass Spectra | The number of mass spectra to collect |
| Spectrum Repeat Period | Period (in seconds) at which spectra are to be acquired |

### *5.1.6 Heater Mode Event (ID=10):*

Length: 2 bytes for heater off, 5 bytes for heater on (Lindau valve) 6 bytes for heater on (PWM heater).

*For Lindau valves:*

| Byte | Bit 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 10 | | | | | | | 0 | Always Present |
| 1 | Lindau valve Number | | | | | | | On/Off | |
| 2 | Target Temperature (high byte) | | | | | | | | Present only for On |
| 3 | Target Temperature (low byte) | | | | | | | | |
| 4 | Max power | | | | | | | | |

*For Pulse Width Modulated Heaters:*

| Byte | Bit 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 10 | | | | | | | unused | Always Present |
| 1 | Heater Number | | | | | | | On/Off | |
| 2 | Target Temperature (high byte) | | | | | | | | Present only for On |
| 3 | Target Temperature (low byte) | | | | | | | | |
| 4 | Start of Power Window | | | | | | | | |
| 5 | End of Power Window | | | | | | | | |

| | | |
|---|---|---|
| Lindau Valve Number | Number of Lindau valve to heat: | 24: LV1<br>25: LV2<br>26: valve deleted<br>27: valve deleted<br>28: LV5<br>29: LV6<br>30: LV7 |
| On/Off | 1:Heating on, 0:Heating off | |
| Target Temperature | The target temperature for heater in tlm counts | |
| Max Power | Maximum Power (in DAC counts) for Lindau valves | |
| Start Power Window | First 1024Hz slot to use for PWM heaters | |
| End Power Window | Last 1024Hz slot to use for PWM heaters | |

### *5.1.7 SMA heater Mode Event (ID=12)*

This is a single byte mode event which turns SMA heating on or off:

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | 12 | | | | | | | On/Off |

on/off: 1: Heater on, 0: Heater off

### *5.1.8 Pressure Monitoring Mode Event (ID=17):*

Length: 6 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 17 | | | | | | | unused |
| 1 | Pressure Guage Number | | | | | | | unused |
| 2 | Target Pressure (high byte) | | | | | | | |
| 3 | Target Pressure (low byte) | | | | | | | |
| 4 | Timeout period (high byte) | | | | | | | |
| 5 | Timeout period (low byte) | | | | | | | |

This mode event waits until a target pressure is attained. When this pressure is reached, or the wait exceeds the specified timeout period, the mode event completes and the mode event sequence continues

| | | |
|---|---|---|
| Pressure guage number | Designator for pressure guage to monitor: | 32: G1<br>33: G2<br>34: G3<br>35: G4<br>36: G5 |
| Target Pressure | Pressure (in tlm counts) above which event completes | |
| Timout period | Time in seconds after which event completes regardless of pressure reading | |

### *5.1.9 Temperature Monitoring Mode Event (ID=18):*

Length: 6 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 18 | | | | | | | unused |
| 1 | Temperature sensor Number | | | | | | | unused |
| 2 | Target Temperature (high byte) | | | | | | | |
| 3 | Target Temperature (low byte) | | | | | | | |
| 4 | Timeout period (high byte) | | | | | | | |
| 5 | Timeout period (low byte) | | | | | | | |

This mode event waits until a target temperature is attained. When this temperature is reached, or the wait exceeds the specified timeout period, the mode event completes and the mode event sequence continues

| | | |
|---|---|---|
| Temperature sensor number | Designator for temperature guage to monitor: | 24: LV1<br>25: LV2<br>26-27: none<br>28: LV5<br>29: LV6<br>30: LV7<br>40: R1<br>41: R2<br>42: reactor deleted<br>43: R4<br>44: R5<br>45: R6<br>46: R7<br>47: R8<br>48: R9<br>49-51: no reactors<br>52: R13<br>53: R14<br>54: R15<br>55: OVEN<br>56: ENCL1<br>57: ENCL2<br>58: PIPE<br>59: GC<br>60: ION |
| Target Pressure | Temperature (in tlm counts) above which event completes | |
| Timout period | Time in seconds after which event completes regardless of temperature reading | |

### *5.1.10 Auxiliary Data Mode Event (ID=20)*

This is a single byte mode event which causes an iterm of auxiliary data to be read and added to the current Aux Data record.

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 20 | | | | | | | unused |
| 1 | Aux data channel | | | | | | | unused |

The Aux data channels are as follows:

| | | |
|---|---|---|
| 64: tLV1<br>65: tLV2<br>68: tLV5<br>69: tLV6<br>70: tLV7<br>72: pG1<br>73: pG2<br>74: pG3<br>75: pG4<br>76: pG5 | 80: tR1<br>81: tR2<br>83: tR4<br>84: tR5<br>85: tR6<br>86: tR7<br>87: tR8<br>88: tR9<br>92: tR13<br>93: tR14<br>94: tR15<br>95: tOVEN<br>96: tENC1<br>97: tENC2<br>98: tPIPE<br>99: tGC<br>100:tION | 103: AD590<br>104: 5 Volt Rail<br>105: 28 Volt Bus<br>106: 5 Volt Rail Current<br>107: 28 Volt Bus Current<br>108: Docking Station Potentiomenter<br>112: Nanotip Drive Current<br>113: Detector HT Voltage<br>114: RF Amplitude |

### *5.1.11 Time Delay Mode Event (ID=24):*

Length: 3 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 24 | | | | | | | unused |
| 1 | Time Delay (high byte) | | | | | | | |
| 2 | Time Delay (low byte) | | | | | | | |

This mode event causes the mode event sequence to pause for <Time Delay> seconds

### *5.1.12 Sample Collector Mode Event (ID=33)*

This is a single byte mode event which commands the docking or undocking of the sample collector:

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | 33 | | | | | | | Dck/Und |

Dck/Und: 1: Dock 0: Undock

#### *5.1.13 Mass Spectrometer Activation Mode Event (ID=34)*

Length: 2 bytes (activate), 1 byte (de-activate)

| | Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
| 0 | 34 | | | | | | | Act/Dea | Always pr |
| 1 | Ion Trap Electrical Characteristics Table Number | | | | | | | | Act only |

| | |
|---|---|
| Act/Deact | 1: Activate Mass Spectrometer<br>0: Deactivate Mass Spectrometer |
| Ion Trap Elec. Char. Table Number | Number (index) of the table of Ion trap set-up values etc. to use – present only in activate case |

#### *5.1.14 WGA Memory Check Mode Event (ID=36)*

This is a single byte mode event which initiates a WGA memory check – no parameters

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | 36 | | | | | | | not used |

#### *5.1.15 Timer Mode Event (ID=40):*

Length: 3 bytes (Start Timer), 1 byte (Wait for Timer)

| | Bit | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | |
| 0 | 40 | | | | | | | Strt/Wt | Always |
| 1 | Time Period (high byte) | | | | | | | | Start Timer |
| 2 | Time Period (low byte) | | | | | | | | |

| | |
|---|---|
| Strt/Wt | 1: Start Timer (using time period)<br>0: Wait for Timer to elapse |
| Time Period | Number of seconds for timer |

#### *5.1.16 Set Injector Control Valve Opening Duration Mode Event (ID=48):*

Length: 4 bytes

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 0 | 48 | | | | | | | unused |
| 1 | Pressure Guage Designation | | | | | | | unused |
| 2 | Duration Constant 1 | | | | | | | |
| 3 | Duration Constant 2 | | | | | | | |

Sets the duration of the pulse for opening the injector control valves.

| | |
|---|---|
| Pressure Guage Designation | 32: G1<br>33: G2<br>34: G3<br>35: G4<br>36: G6 |
| Duration Constant 1 | Constant in arbitary units |
| Duration Constant 2 | Constant in arbitary units |

#### *5.1.17 RF Frequency Calibration Mode Event (ID=65)*

This is a single byte mode event initiates an RF frequency calibration:

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | 65 | | | | | | | unused |

#### *5.1.18 End of Mode Event (ID=65)*

This is a single byte mode event marks the end of an active science mode, when this mode event is encountered, the software returns to standby mode.

| Bit | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|
| | 127 | | | | | | | unused |

## 5.2 Tables

### *5.2.1 Scan Function*

The scan function table describes a mode of operation of the RICA and WGA for collection of Spectra from the Ion Trap.

The Format of the Scan Function in memory is as follows:-

| Field | No bytes | |
|---|---|---|
| Number of Bins | 2 | range 0-1024 set to 65535 if no scan function |
| Scan Function Duration | 4 | range 0-4294967295 |
| Transmit Spectrum as Type | 1 | Full=1, One packet=0 |
| TABLE number | 1 | range 0-7 set to 255 if no scan function defined |
| TABLE Start Address (in WGA) | 2 | 01Ch to 1FFh |
| TABLE data | variable | range 1-484 |
| WAVEs Data | variable | |

Wave Data:

| Wave table field | No. bytes | |
|---|---|---|
| WAVE Number | 1 | range 0-9. set to 255 for End of Scan Function |
| WAVE Start Address | 2 | 200h – 5FFh |
| WAVE Stop Address | 2 | 200h – 5FFh |
| WAVE Data | variable | 8 – 1024 bytes |

Wave data:

Each byte represents the Least Significant Byte of the output of the WGA. The onboard software pads out the Most Significant Byte with Zero as this saves memory because this redundant zero is not stored. The total number of states in all 10 WAVES must not exceed 1024. Fewer than 10 WAVES may be specified, this is indicated by placing a byte after the last 'WAVE data' with 255 in it. The format of the bytes that describe the 'WAVE data' is explained in section 2.5.15 of the 'Ptolemy Hardware/Software Interface Document'.

## 5.3 Patches

The following patches have been identified for the Ptolemy On-board Software

### *5.3.1 SAFE mode HK collection patch*

**Problem**

In SAFE mode, the interval at which analogue multiplexer channels are selected for analogue HK collection defaults to a low value (a few µs). For higher impedance channels, this causes an inaccurate value to be returned because of insufficient settling time.

**Resolution**

The delay between HK samples is determined by a timing loop. The count for this loop is held in the constant _MaximumConverterDelayCount at address 0DCEH in the data page (nom. page 8). Patching this address to 600H (exact value not critical) eliminates the problem.

**Patch:**

Patch page 8 address 0DCEH to 0600H

### *5.3.2 Temperature Servo PID patch*

**Problem**

For large temperature errors, the current temperature servo PID gives no change to the heater output (it remains at 0 and so heating never occurs).

**Cause**

Examination of the temperature PID implementation revealed that the heater output update was being checked against the physical limits (0-255) after conversion from floating point to integer. For large temperature errors (as expected at the beginning of heating), the calculated heater update is outside the range of a 16 bit integer and the RTX-C run-time system substitutes 0. If the range check were made before conversion to integer, the large heater demands would be limited to 255.0 which would convert successfully to 255.

**Resolution**

A temporary work-around was to reduce the PID constants to the point at which the integer conversion was successful for the demand encountered. Subsequently a patch was created which converts any heater demands to large to convert to an integer into a number of the same sign with an exponent of 14 and an arbitary mantissa. The exact value of the number is not important as it shall in any case be clipped to the range 0-255.

### Patch

The following patch has been prepared and tested on the Chemistry set simulator:

```
#
# Patch 1 - patch to fix the Heater PID bug.
#
# In the science-mode function ManageHeatingMaintainedModeEvent ( in source
# file Co123333.c), the result of the expression ((float)PowerValue)+DeltaP
# is converted to an integer before range checking is performed. If the value
# is too large to fit a 16 bit integer, as may happen if the PID constants
# are large and there is a substantial difference between measured and
# demanded temperature, 0 shall be returned, with the result that the PID
# algorithm never starts heating.
#
# This patch checks the exponent of the above expression and limits it to
# 14 so that any very large demand is reduced to a number in the ranges
# -16384 to -8191 or 8192 to 16383. Such numbers are in any case clipped to
# the power limit (a much lower value) subsequently so any number is
# suitable that shall result in a large integer of the correct sign.
#
# Link: patch 6810h (call d020h) to address 9164h of code page
#
# org $d020    # patches start at end of baseline code

patch_1::

  dup          # copy exponent
  $e -         # compare to 14
  0< 0=        # if it is less than 14
  cj p_1_1     # skip - is not too big

  drop $e      # else replace with 14

p_1_1:

  call $cfdc   # execute the overwritten instruction
  ;            # return to baseline program
#
# object code shown below
# Disassembler for the RTX 2000 C Cross Compiler
# v1.2g
# COPYRIGHT 1990 HARRIS CORPORATION ALL RIGHTS RESERVED
# 0d020  :  a0c0      1   dup
# 0d022  :  b4ce      1   slit $e -
# 0d024  :  a001      1   0<
# 0d026  :  b008      1   0=
# 0d028  :  8817      1   cj $d02e   d02e
# 0d02a  :  ae40      1   drop
# 0d02c  :  be4e      1   slit $e
# 0d02e  :  67ee      1   call       cfdc
# 0d030  :  a020      1   nop ;
#
```