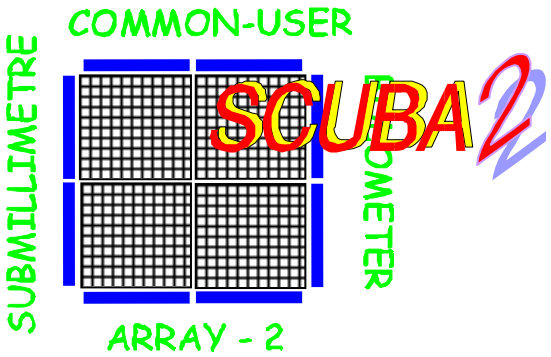

SCUBA-2 FTS Project Office
University of Lethbridge
Physics Department
4401 University Drive
Lethbridge, Alberta
CANADA
T1K 3M4

Tel: 1-403-329-2771
Fax: 1-403-329-2057
Email: brad.gom@uleth.ca
WWW: http://research.uleth.ca/scuba2/

Document Title: FTS-2 to OCS ICD

Document Number: SC2/FTS/SOF/002

Issue: Version 3.1

Date: 1 February 2007

Document Prepared By:	Baoshe Zhang FTS-2 Software Engineer	Signature and Date:	01/02/07
Document Approved By:	D. A. Naylor FTS-2 Project Lead	Signature and Date:	01/02/07
Document Released By:	J. Molnar Canadian Project Manager	Signature and Date:	01/02/07
Document Approved by:	Craig Walther Head of JAC Software	Signature and Date:	31/01/07

Change Record

Issue	Date	Section(s) Affected	Description of Change / Change Request Reference / Remarks
0.1	04/10/04	All	First draft
0.2	13/05/05	All	Revision after restaffing
1.0	20/06/05	All	PDR version
2.0	1/11/06	All	CDR version. No major changes
2.1	2/11/06	5	Updated CONFIGURE XML file
3.0	31/01/07	2.1, 3.2, 3.3, 3.4, 4.4, 4.5, 7	Delta-CDR version. Updated JOS observing actions Observing recipes updated after RTS client tests
3.1	1/02/07	3.2	minor typos

Contents

Change Record 2
Contents 3
1. Introduction 4
  1.1. Interface definition 4
  1.2. Acronyms and Abbreviations 4
  1.3. Applicable and Referenced Documents 4
2. Overview 5
  2.1. JOS to FTS-2 5
  2.2. Translator to FTS-2 5
  2.3. RTS to FTS-2 5
  2.4. FTS-2 to SCUBA-2 DR 5
  2.5. FTS-2 to JOS 6
3. JOS to FTS-2 – Observing Actions 6
  3.1. INITIALISE 6
  3.2. CONFIGURE 6
  3.3. SETUP_SEQUENCE 7
  3.4. SEQUENCE 7
  3.5. END_OBSERVATION 8
  3.6. END_OF_NIGHT 8
4. JOS to FTS-2 – Houskeeping Actions 8
  4.1. LOAD_Aerotech 8
  4.2. INIT_Aerotech 8
  4.3. Home 8
  4.4. INSERT 9
  4.5. RETRACT 9
  4.6. TEST 9
5. FTS-2 CONFIGURE XML 9
6. FTS-2 STATE Structure 11
7. Observing Recipes 12
  7.1. Step and Integrate 12
  7.2. Constant Velocity 15
  7.3. ZPD Scan 17

1. Introduction

1.1. Interface definition

This document defines the interfaces between the JAC Observatory Control System (OCS) and the Fourier Transform Spectrometer (FTS-2) for the SCUBA-2 instrument. The FTS-2 task is a JIT task and conforms to http://www.jach.hawaii.edu/JACdocs/JCMT/OCS/ICD/009/jit.pdf. FTS-2 will also be an RTS DRAMA client. These actions are discussed in detail in the following sections.

1.2. Acronyms and Abbreviations

CFI	-	Canadian Foundation for Innovation
EMI	-	Electromagnetic Interference
FTS	-	Fourier Transform Spectrometer
ICD	-	Interface Control Document
JAC	-	Joint Astronomy Centre
JCMT	-	James Clerk Maxwell Telescope
JOS	-	JCMT Observation Sequencer
OPD	-	Optical Path Difference
OT	-	Observation Template
PC	-	Personal Computer
RTS	-	Real Time Sequencer
SCUBA	-	Submillimetre Common User Bolometer Array
U of L	-	University of Lethbridge
UBC	-	University of British Columbia
UKATC	-	UK Astronomy Technology Centre
ZPD	-	Zero Path Difference

1.3. Applicable and Referenced Documents

Document Number	***Title***	***Number & Issue***
SC2/FTS/SRE/001	FTS-2 Requirements Document	Version 3.0
SCS/FTS/SYS/005	FTS-2 to RTS ICD	Version 2.0
OCS/ICD/009	JAC Instrumentation Task (JIT) Library	

2. Overview

2.1. JOS to FTS-2

The JOS will command FTS-2 with DRAMA obeys. This breaks up into observing actions and housekeeping actions.

Observing Actions

- INITIALIZE - Connect and initialise Aerotech motor controller
- CONFIGURE – Put the pick-off mirror in place if not already, set scan mode, scan velocity, scan direction and scan range or step size.
- SETUP_SEQUENCE - move to first position depending on scan mode
- SEQUENCE – Start moving the stage and run an RTS sequence
- END_OBSERVATION – Restore the FTS to the state immediately after INITIALISE and get ready to accept further commands
- END_OF_NIGHT – Close the FTS-2's shutter, retract the pick-off mirror mount, power off the FTS-2 system.

Housekeeping Actions

- LOAD_AEROTECH - Load any macros into the Aerotech motor controllers flash memory.
- INIT_AEROTECH - Load any constants such as motor acceleration, etc., into the Aerotech volatile memory.
- DATUM - Bring the FTS-2 mirror to DATUM.
- INSERT – Put the pick-off mirror in place.
- RETRACT – Retract the pick-off mirror.
- TEST - Perform a system self test.

Other DRAMA commands

- "kick" of SEQUENCE – Stop the stage, abort the RTS sequence and prepare for a new SEQUENCE

2.2. Translator to FTS-2

The translator prepares an XML file supplying the static parameters for the CONFIGURE action at the start of an observation, based on the description of the observation prepared in the OT.

2.3. RTS to FTS-2

The RTS synchronizes high-speed operations amongst the subsystems participating in a data collecting sequence. FTS-2 is a fully compatible RTS client.

2.4. FTS-2 to SCUBA-2 DR

At the end of every RTS sequence, FTS-2 updates a DRAMA SDS STATE structure which contains configuration information and the position values of the sequence and their corresponding sequence number.

2.5. FTS-2 to JOS

FTS-2 sends error and warning messages to the JOS.

3. JOS to FTS-2 – Observing Actions

3.1. INITIALISE

Arguments:

- INITIALISE → Name of INITIALISE XML file if needed.
- SIMULATE → NONE=0, MIRROR=1

Purpose:

Bring FTS-2 into a known passive state. This will establish any paths with other tasks and hardware. The FTS-2 task will call INIT_AEROTECH. This sets any constants for use during observing. Additionally, this will close and open the Aerotech communications port. FTS-2 will be left in a state waiting for commands.

Comments:

- Normally run once before the first FTS-2 observation of the night.
- This action will not be used during observing but will be used in the observatory start-up scripts.
- In the INITIALIZE action the FTS-2 task will read in any initial constants and values from an initialization XML file. This file name is defined in the start-up script.

3.2. CONFIGURE

Arguments:

- CONFIGURE → The name of the configure XML file

Purpose:

Configure FTS-2 for the appropriate observing mode.

Comments:

- CONFIGURE is run at the start of every observing recipe to configure FTS-2
- See CONFIGURE XML section for various configurations.
- There are 4 possible observing modes for FTS-2. The specific mode used depends on the XML received in the CONFIGURE action. These modes are defined in an XML element called SCAN_MODE and is RAPID_SCAN (mode 0), STEP_AND_INTEGRATE (mode 1), DREAM (mode 2), or ZPD_MODE (mode 3).
  (a) RAPID_SCAN mode configures FTS-2 to use a given scan speed in millimetres/second and scan length in millimetres.
  (b) STEP_AND_INTEGRATE mode configures FTS-2 to use a given step size.
  (c) ZPD_MODE (special hardware may be needed for ZPD_MODE).
- For mode 0, or RAPID_SCAN mode, load the parameters: SCAN_DELAY (in millisecond), SCAN_SPD(millimetres per second), SCAN_ORIGIN and

SCAN_LENGTH (millimetres) and SCAN_DIR (possible values: DIR_LEFT_TO_RIGHT, DIR_RIGHT_TO_LEFT, and DIR_ARBITRARY).

- For mode 1, or STEP_AND_INTEGRATE mode, load the parameters: SCAN_SPD, SCAN_ORIGIN, STEP_SIZE, and SCAN_DIR.
- For mode 2, or DREAM, load the parameters: SCAN_SPD and DREAM_POS.
- For mode 3, or ZPD_MODE mode, load the parameters: SCAN_SPD, SCAN_ORIGIN, STEP_SIZE, and SCAN_DIR.

3.3. SETUP_SEQUENCE

Arguments:

- INDEX1 – the index of DREAM_POS

Purpose:

For the scans with direction DIR_LEFT_TO_RIGHT or DIR_RIGHT_TO_LEFT, move the stage to the origin (defined by SCAN_ORIGIN). For the scans with direction DIR_ARBITRARY, move the stage to the closer end (defined either by SCAN_ORIGIN or SCAN_ORIGIN+SCAN_LENGTH). For DREAM, move the stage to the position indexed by INDEX1.

Return Values:

The return value of MAX depends on the FTS-2 mode. For STEP_AND_INTEGRATE, MAX = SCAN_LENGTH/STEP_SIZE. For RAPID_SCAN, nothing returned. In DREAM mode, if INDEX1 > POS_NUM, return MAX with 0.
See also: http://docs.jach.hawaii.edu/JCMT/OCS/ICD/009/jit.pdf.)

3.4. SEQUENCE

Arguments:

- START
- END
- DWELL

Purpose:

Run the RTS sequence starting at the START sequence number and ending at the END sequence number. For STEP_AND_INTEGRATE, the stage will stay in the same position for DWELL RTS steps; For RAPID_SCAN and DREAM, DWELL = 1.

Comments:

- A SEQUENCE action can be ended with a kick that will cause the stage to stop and exit immediately.
- During LstInt, download all the position values from Soloist and fill out the STATE structure and publish it.
- During a sequence the following flags are used to keep track of the stage state:

1. POS_NUM An index that counts from 1 to N in a complete scan. Before a new scan, it is reset to 0.
2. LAST_POSITION_FLAG: Before a new scan, it is reset to zero. When the last position value is filled in the state parameter, it is set to 1.

3.5. END_OBSERVATION

Purpose:

Place FTS-2 in a safe mode. Stop the stage and wait for further commands.

Comments:

- Stops the stage.
- Invalidates CONFIGURE, but not INITIALISE.

3.6. END_OF_NIGHT

Purpose:

Retract the pick-off mirror mount to its rest place and place FTS-2 in safe mode.

4. JOS to FTS-2 – Houskeeping Actions

4.1. LOAD_Aerotech

Purpose:

Load any macros into the Aerotech motor controllers flash memory.

4.2. INIT_Aerotech

Purpose:

Load any constants such as motor acceleration, velocity, etc., into the Aerotech volatile memory.

Comments:

The port to the Aerotech controller will always be closed first and then opened. This is to ensure the port is always reset even if the port is currently in an open state. If the port is in a closed state an attempt to close the port will not return a fatal error.

4.3. Home

Purpose:

Bring the FTS-2 mirror to the HOME position.

4.4. INSERT

Purpose:

Bring the pick-off mirror mount into the FTS-2 observation place.

4.5. RETRACT

Purpose:

Put the FTS-2 pick-off mirror mount back to its rest place.

4.6. TEST

Purpose:

Perform simple system tests.

Arguments:

- Mode → SCAN test or just a discrete MOVE test.
  1. SCAN will be given a scan speed (mm/sec) and distance (mm).
  2. MOVE will be given an OPD position in mm.

5. FTS-2 CONFIGURE XML

```
<?xml version="1.0" standalone="no" ?>
<!--  Test file for proposed JCMT FTS XML. Ready for comments! -->
<!--
   This is the definition of FTS-2 configuration that is used for defining the FTS
   motion during an observation. Converted from the ROVER configuration written
   by Mathew Rippa.

  Author: Baoshe Zhang
 -->
<!-- ========================================================  -->
<!--
The FTS_CONFIG element is for controlling the motion of the FTS-2 stage.
 -->
<!--  ELEMENT FTS_CONFIG (FTS_SCAN)  -->
<!--
ELEMENT FTS_SCAN (SCAN_MODE?, SCAN_DIR?, SCAN_SPD?, SCAN_DELAY?,
SCAN_LENGTH?, STEP_SIZE? )
 -->
<!--  ELEMENT SCAN_MODE (#PCDATA)   -->
<!--  ELEMENT SCAN_DELAY (#PCDATA)   -->
```

```
<!--  ELEMENT SCAN_DIR (#PCDATA)   -->
<!--  ELEMENT SCAN_SPD (#PCDATA)   -->
<!--  ELEMENT SCAN_LENGTH (#PCDATA)  -->
<!--  ELEMENT STEP_SIZE (#PCDATA)  -->
<!--
 The FTS_SCAN mode can have the attribute of either "RAPID_SCAN",
where FTS-2 will move at SCAN_SPD millimetres/second for SCAN_LENGTH
millimetres, or "STEP_AND_INTEGRATE" will move STEP_SIZE millimetres at each time,
or "DREAM" (similar to STEP_AND_INTEGRATE, the positions defined by DREAM_POS) ,
or "ZPD_MODE" (similar to STEP_AND_INTEGRATE)
  -->
 <!-- Here is some example XML: Test file for proposed JCMT FTS-2 control. -->
 <FTS_CONFIG>
    <SCAN_MODE VALUE="RAPID_SCAN" />
    <SCAN_DIR VALUE="DIR_ARBITRARY" />
    <SCAN_DELAY unit="millisecond" type="int">3</SCAN_DELAY>
    <SCAN_ORIGIN unit="mm" type="float">30</SCAN_ORIGIN>
    <SCAN_SPD unit="mm/sec" type="float">10</SCAN_SPD>
    <SCAN_LENGTH unit="mm" type="float">170.0</SCAN_LENGTH>
    <STEP_SIZE unit="mm" type="float">0.1</STEP_SIZE>
    <DREAM_POS POS_NUM=9 unit="mm">
        <POS>001</POS>
        <POS>002</POS>
        <POS>003</POS>
        <POS>004</POS>
        <POS>005</POS>
        <POS>006</POS>
        <POS>007</POS>
        <POS>008</POS>
        <POS>009</POS>
    </DREAM_POS>
  </FTS_CONFIG>
```

6. FTS-2 STATE Structure

Parameter Name	Description	Type
POS_NUM	The number of valid position values in this structure.	int
SCAN_MODE	The scan mode. Possible values: RAPID_SCAN (=0), STEP_AND_INTEGRATE (=1), and ZPD_MODE (=2)	int
SCAN_DIR	The direction of the current scan. LEFT_TO_RIGHT = 1, RIGHT_TO_LEFT = -1.	int
LAST_POSITION_FLAG	A last position flag. Before a new scan, it is reset to zero. When the last position value is filled in, it is set to 1.	Int
DWELL	The number of RTS steps for one position.	int
NUMBER	The RTS step number	int
POS	The position value of the stage	float
•	•	•
•	•	•
NUMBER	The last RTS step number	int
POS	The last position value of the stage	float

7. Observing Recipes

7.1. Step and Integrate

```
#-------------------------------------------------------------------------------------------
# Recipe Name: stepAndIntegrate
# Recipe Purpose: Used to take SCUBA-2 data using either the FTS, or the Polarimeter in
#                  step-and-integrate mode.
#
# Short Recipe Description:
#
# In CONFIGURE :  SCUBA2:  Moves cold shutter into the beam
#                 PTCS:    Slews to the source.
#                 SMU:     Moves to the MIDDLE beam, clears amplitudes, sets mode to ACSIS
#                 RTS:     Sets up the tasks it will hardware handshake with
#                 POL FTS: Move to first position
#
# A loop in entered which is executed NUM_CYCLES times
# The Loop
#     Set src_index to 1;
#     A loop1 is entered which is executed as many times as there are SCIENCEx positions
#     (Loop1 allows us to build up a mosaic of stare panels)
#     The Loop1
#        Form a SOURCE directive made up of the text "SCIENCE" followed by the src_index
#        Ask the PTCS if this source exists, if not stop Loop1
#        A Loop2 is entered which is executed as many times as there are offsets in the target
#        (Loop2 allows for filling in the space between arrays and fully sampling the 450 array)
#        The Loop2
#           Ask the PTCS if this offset exists, if not stop Loop2
#           A Loop3 is entered which is executed as many times as there are step positions
#           in the FTS or polarimeter CONFIGURATION XML
#           The Loop3
#               Move the telescope to the specified SCIENCEx plus offset position
#               Move the FTS or polarimeter to the specified INDEX1 postion. Pol or
#               FTS should return MAX=0 if the offset does not exist.
#               Remove the cold shutter from the beam.(will only happen the first time)
#               Take JOS_MIN number of samples
#
#
# The cold shutter is moved into the beam at the end of the recipe
#
#-------------------------------------------------------------------------------------------

sub SCUBA2_stepAndIntegrate( $$% ) {

  my ( $tasks,                 # Reference to an array of participating tasks -
                               # Typically (PTCS, SCUBA2, SMU, RTS and POL or FTS).
       $configuration,         # Name of configuration XML to use
       $par) = @_;             # Reference to a hash containing the recipe parameters. The hash
                               # must contain values for the following keys
                               #  NUM_CYCLES      => Number of times to repeat entire recipe
                               #  JOS_MIN         => Number of RTS steps to take at each stare
                               #                     position
                               #  STEP_TIME       => RTS step time during an RTS sequence

  my $status = 1;
  my $group = 0;
  my $nsteps = 1;

  configure( $tasks, $configuration, $status );

  my $i = 0;
  my $source = "SCIENCE1";

# Start the outside Loop go through it NUM_CYCLES times

  for ( $i = 1; $i <= $par->{NUM_CYCLES} && $status; $i++ ) {

      my $src_index = 1;

# Start the Loop1
      while ($nsteps) {

# Form the SOURCE directive
```

```
      $source = "SCIENCE" . src_index;
      $src_index += 1;

# Ask the PTCS if this source exists

      $nsteps = 1;

      my ( $nsteps, @dwells ) = setup_sequence("PTCS",
                              { "SOURCE"     => $source,
                                "INDEX"      => 1,
                                "BEAM"       => "MIDDLE",
                                "STEP_TIME"  => $par->{STEP_TIME},
                                "GROUP"      => 0,
                                "LOAD"       => "SKY",
                                "MASTER"     => ""
                              },
                             {},
                              $status);

# If this source does not exist, then we are done

      if( $nsteps == 0) next;

# Start the Loop2

      my $point = 0;
      while ($nsteps) {
        $point++;

# Ask the PTCS if this source exists

        $nsteps = 1;

        my ( $nsteps, @dwells ) = setup_sequence("PTCS",
                               { "SOURCE"     => $source,
                                 "INDEX"      => $point,
                                 "BEAM"       => "MIDDLE",
                                 "STEP_TIME"  => $par->{STEP_TIME},
                                 "GROUP"      => $group,
                                 "LOAD"       => "SKY",
                                 "MASTER"     => ""
                               },
                              {},
                               $status);

# If this source does not exist, then we are done

        if( $nsteps == 0) next;


# Start the Loop3 and step through each of the polorimeter or FTS positions

        my $index1 = 0;
        while ($nsteps) {
           index1++;

# Ask the PTCS to move to $source plus the offset indexed by $point.
# Ask the FTS or Polarimeter to move to the steps position indexed by index1
# If the position does not exist, the FTS or Pol will return a MAX of zero
# causing $nsteps to be zero.
# If $nsteps is not zero, JOS_MIN steps will be taken at this position.
# This will also cause SCUBA2 to remove the cold shutter from the beam (the first time through).

           $nsteps = integrate( $tasks,
                               { "SOURCE"     => $source,
                                 "INDEX"      => $point,
                                 "INDEX1"     => $index1,
                                 "BEAM"       => "MIDDLE",
                                 "STEP_TIME"  => $par->{STEP_TIME},
                                 "GROUP"      => $group,
                                 "LOAD"       => "SKY",
                                 "MASTER"     => ""
                               },
                               $par->{JOS_MIN},
                               1,
                               $status);
```

```
            } # End of while (Loop3), all positions in the Pol's or FTS's XML have been completed
          } # End of while (Loop2), all offsets in the PTCS's XML have been completed
        $group += 1;
       } # End of while (Loop1), all sources in the PTCS's XML have been done
  } # End of for Loop on NUM_CYCLES

# Move the cold shutter into the beam

    setup_sequence( $tasks,
                    { "STEP_TIME"   => $par->{STEP_TIME},
                      "LOAD"        => "DARK",
                      "MASTER"      => "PTCS"
                    },
                    {},
                    $status);

  end_observation( $tasks );
}
```

7.2. Constant Velocity

```
#-------------------------------------------------------------------------------------------------
# Recipe Name: constantVelocity
# Recipe Purpose: Used to take SCUBA-2 data using either the FTS, or the Polarimeter in
#                 the constant velocity mode
#
# JOS Parameters: TASKS:       List of participating task names
#                 NUM_CYCLES:  Number of times to repeat entire recipe
#                 STEP_TIME:   RTS step time during a RTS sequence
#                 JOS_MIN:     The number of steps in a sequence
#
# Short Recipe Description:
#
# In CONFIGURE :  SCUBA2: Moves cold shutter into the beam
#                 PTCS:   Slews to the source.
#                 SMU:    Moves to the MIDDLE beam, clears amplitudes, sets mode to ACSIS
#                 RTS:    Sets up the tasks it will hardware handshake with
#                 POL:    Begin spinning waveplate at the requested speed
#                 FTS:    Move to starting position.
#
# A loop in entered which is executed NUM_CYCLES times
# The Loop
#     Set src_index to 1;
#     A loop1 is entered which is executed as many times as there are SCIENCEx positions
#     (Loop1 allows us to build up a mosaic of stare panels)
#     The Loop1
#        Form a SOURCE directive made up of the text "SCIENCE" followed by the src_index
#        Ask the PTCS if this source exists, if not stop Loop1
#        A Loop2 is entered which is executed as many times as there are offsets in the target
#        (Loop2 allows for filling in the space between arrays and fully sampling the 450 array)
#        The Loop2
#           Move the telescope to the specified SCIENCEx plus offset position. The PTCS should
#             return MAX=0 if the offset does not exist.
#           Remove the cold shutter from the beam.(will only happen the first time)
#           The FTS should move back to its starting position and then start moving again at
#              its requested speed. If the FTS has been put in ARBITRARY motion mode in will
#              actual have two different starting positions on either side of its stage and
#              move one direction in one sequence and the opposite direction the next.
#           Take JOS_MIN number of samples
#
#
# The cold shutter is moved into the beam at the end of the recipe
#
#-------------------------------------------------------------------------------------------------

sub SCUBA2_constantVelocity( $$% ) {

  my ( $tasks,                  # Reference to an array of participating tasks -
                                # Typically (PTCS, SCUBA2, SMU, RTS and POL or FTS).
       $configuration,          # Name of configuration XML to use
       $par) = @_;              # Reference to a hash containing the recipe parameters. The hash
                                # must contain values for the following keys
                                #  NUM_CYCLES      => Number of times to repeat entire recipe
                                #  JOS_MIN         => Number of RTS steps to take at each stare
                                #                     position
                                #  STEP_TIME       => RTS step time during an RTS sequence

  my $status = 1;
  my $group = 0;
  my $nsteps = 1;

  configure( $tasks, $configuration, $status );

  my $i = 0;
  my $source = "SCIENCE1";

# Start the outside Loop go through it NUM_CYCLES times

  for ( $i = 1; $i <= $par->{NUM_CYCLES} && $status; $i++ ) {

      my $src_index = 1;

# Start the Loop1
      while ($nsteps) {
```

```
# Form the SOURCE directive

      $source = "SCIENCE" . src_index;
      $src_index += 1;

# Ask the PTCS if this source exists

    $nsteps = 1;

    my ( $nsteps, @dwells ) = setup_sequence("PTCS",
                           { "SOURCE"     => $source,
                             "INDEX"      => 1,
                             "BEAM"       => "MIDDLE",
                             "STEP_TIME"  => $par->{STEP_TIME},
                             "GROUP"      => 0,
                             "LOAD"       => "SKY",
                             "MASTER"     => ""
                           },
                          {},
                           $status);

# If this source does not exist, then we are done

    if( $nsteps == 0) next;

# Start the Loop2

      my $point = 0;
      $nsteps = 1;
      while ($nsteps) {
         $point++;

# Ask the PTCS to move to $source plus the offset indexed by $point. If the offset
# does not exist, the PTCS will return a MAX of zero causing $nsteps to be zero.
# If $nsteps is not zero, JOS_MIN steps will be taken at this position.
# This will also cause SCUBA2 to remove the cold shutter from the beam (the first time through).
# The Polarimeter will already be spinning, the FTS should move to its starting position
# in SETUP_SEQUENCE and start its motion in SEQUENCE

         $nsteps = integrate( $tasks,
                           { "SOURCE"     => $source,
                             "INDEX"      => $point,
                             "BEAM"       => "MIDDLE",
                             "STEP_TIME"  => $par->{STEP_TIME},
                             "GROUP"      => $group,
                             "LOAD"       => "SKY",
                             "MASTER"     => ""
                            },
                            $par->{JOS_MIN},
                            1,
                            $status);

          } # End of while (Loop2), all offsets in the PTCS's XML have been done

      $group += 1;

      } # End of while (Loop1), all sources in the PTCS's XML have been done

  } # End of for Loop on NUM_CYCLES


# Move the cold shutter into the beam

    setup_sequence( $tasks,
                    { "STEP_TIME"  => $par->{STEP_TIME},
                      "LOAD"       => "DARK",
                      "MASTER"     => "PTCS"
                    },
                    {},
                    $status);

  end_observation( $tasks );
}
```

7.3. ZPD Scan

```
#-----------------------------------------------------------------------------------------------
# Recipe Name: zpd
# Recipe Purpose: To measure the Zero Path Difference of the FTS for each of the SCUBA2 pixels
#
# JOS Parameters: TASKS:       List of participating task names
#                 NUM_CYCLES:  Number of times to repeat entire recipe
#                 STEP_TIME:   RTS step time during a RTS sequence
#                 JOS_MIN:     The number of steps in a sequence
#
# Short Recipe Description:
#
# In CONFIGURE :  SCUBA2: Moves cold shutter into the beam
#                 PTCS:   Slews to the source (which is likely empty sky).
#                 SMU:    Moves to the MIDDLE beam, clears amplitudes, sets mode to ACSIS
#                 RTS:    Sets up the tasks it will hardware handshake with
#                 FTS:    Move to starting position and moves blackbody into 2nd beam.
#
# A loop in entered which is executed NUM_CYCLES times
# The Loop
#           Remove the cold shutter from the beam.(will only happen the first time)
#           The FTS should move back to its starting position and then start moving again at
#              its requested speed
#           Take JOS_MIN number of samples
#
#
# The cold shutter is moved into the beam at the end of the recipe
# The Blackbody will be removed from 2nd beam of the FTS
#
#-----------------------------------------------------------------------------------------------

sub SCUBA2_zpd( $$% ) {

  my ( $tasks,                  # Reference to an array of participating tasks -
                                # Typically (PTCS, SCUBA2, SMU, RTS and FTS).
       $configuration,          # Name of configuration XML to use
       $par) = @_;              # Reference to a hash containing the recipe parameters. The hash
                                # must contain values for the following keys
                                #  NUM_CYCLES      => Number of times to repeat entire recipe
                                #  JOS_MIN         => Number of RTS steps to take.
                                #  STEP_TIME       => RTS step time during an RTS sequence

  my $status = 1;

  configure( $tasks, $configuration, $status );

  my $i = 0;

# Start the outside Loop go through it NUM_CYCLES times

  for ( $i = 1; $i <= $par->{NUM_CYCLES} && $status; $i++ ) {

# The PTCS moves to the source. JOS_MIN steps will be taken at this position.
# This will also cause SCUBA2 to remove the cold shutter from the beam (the first time through).
# The FTS should move to its starting position in SETUP_SEQUENCE and start its motion in SEQUENCE

         $nsteps = integrate( $tasks,
                              { "SOURCE"     => "SCIENCE",
                                "BEAM"       => "MIDDLE",
                                "STEP_TIME"  => $par->{STEP_TIME},
                                "LOAD"       => "SKY",
                                "MASTER"     => ""
                              },
                              $par->{JOS_MIN},
                              1,
                              $status);

  } # End of for Loop on NUM_CYCLES


# Move the cold shutter into the beam

    setup_sequence( $tasks,
                    { "STEP_TIME"  => $par->{STEP_TIME},
```

```
                    "LOAD"        => "DARK",
                    "MASTER"      => "PTCS"
                  },
                  {},
                  $status);

  end_observation( $tasks );
}
```