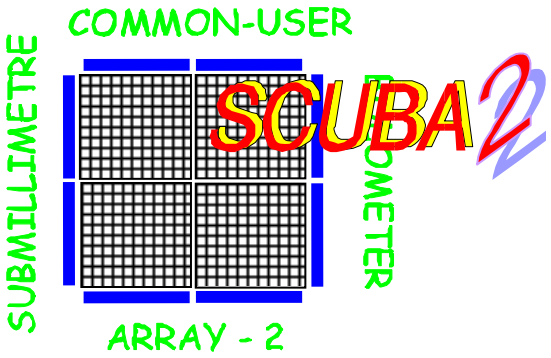

SCUBA-2 FTS Project Office
University of Lethbridge
Physics Department
4401 University Drive
Lethbridge, Alberta
CANADA
T1K 3M4

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 RTS Client Implementation**

Document Number: **SC2/FTS/SOF/xxx**

Issue: **draft**

Date: **9 November 2006**

Document Prepared By:	Baoshe Zhang FTS-2 SW Engineer	Signature and Date:	09/11/06
Document Approved By:	D. A. Naylor FTS Project Lead	Signature and Date:	09/11/06
Document Released By:	J. Molnar Canadian Project Manager	Signature and Date:	09/11/06

Change Record

Issue	Date	Section(s) Affected	Description of Change / Change Request Reference / Remarks
0.1	17/06/02	All	First draft

Contents

Summary

Coordination and synchronization between JCMT instruments, OCS and DAS are performed by the Real Time Sequencer (RTS). Since every RTS device must use the same hardware interface and perform many of the same functions, the JCMT uses a generalized software interface for any new RTS compliant instrument. With this interface, a new RTS compliant instrument only needs to provide 6 event-driven instrument-specific call-back procedures. This significantly simplifies the control software development for new RTS Client instruments.

Currently, the RTS software runs on a VxWorks computer mounted in a VMEBus chassis. Due mainly to budget constraints, the FTS-2 and POL-2 instruments will use the RTAI real-time Linux operating system instead of VxWorks for their RTS client software. At the time of writing however, the real-time Linux version of the RTS software interface was not yet available.

This document details the FTS-2 control system event processing and its RTAI implementation. This implementation follows the RTS requirements tightly, so that when the real-time Linux version of the RTS software interface is available, this implementation can be easily ported to the new interface.

1. Control software structure

A fundamental feature of Linux/Unix is the clear distinction between what occurs in ‘kernel space’ and what occurs in ‘user space’. In order to avoid any kind of latency by the Linux scheduler, the RTS signal processing component of the FTS-2 control software must be a Linux kernel module. Furthermore, the Linux kernel must be a real-time kernel, such as RTAI. However, in order to exploit the functions provided by Linux libraries and system calls by Linux, other parts of the control software must run in the Linux user space. The communication interface between the user-space component and the kernel component is realized by the RTAI FIFO and shared memory mechanism. The control software user-space component uses this interface to set parameters for the kernel component (such as the sequence number of a scan), starts the kernel component real-time handshaking task, and synchronizes the operations between the user-space and kernel components.

2. RTS event sequence

In order to better understand the FTS-2 control software, the RTS signal sequence can be divided into four phases according to function: initialization, handshaking, integration, and last integration. Figure 1 lists various events of these four phases and their relations.

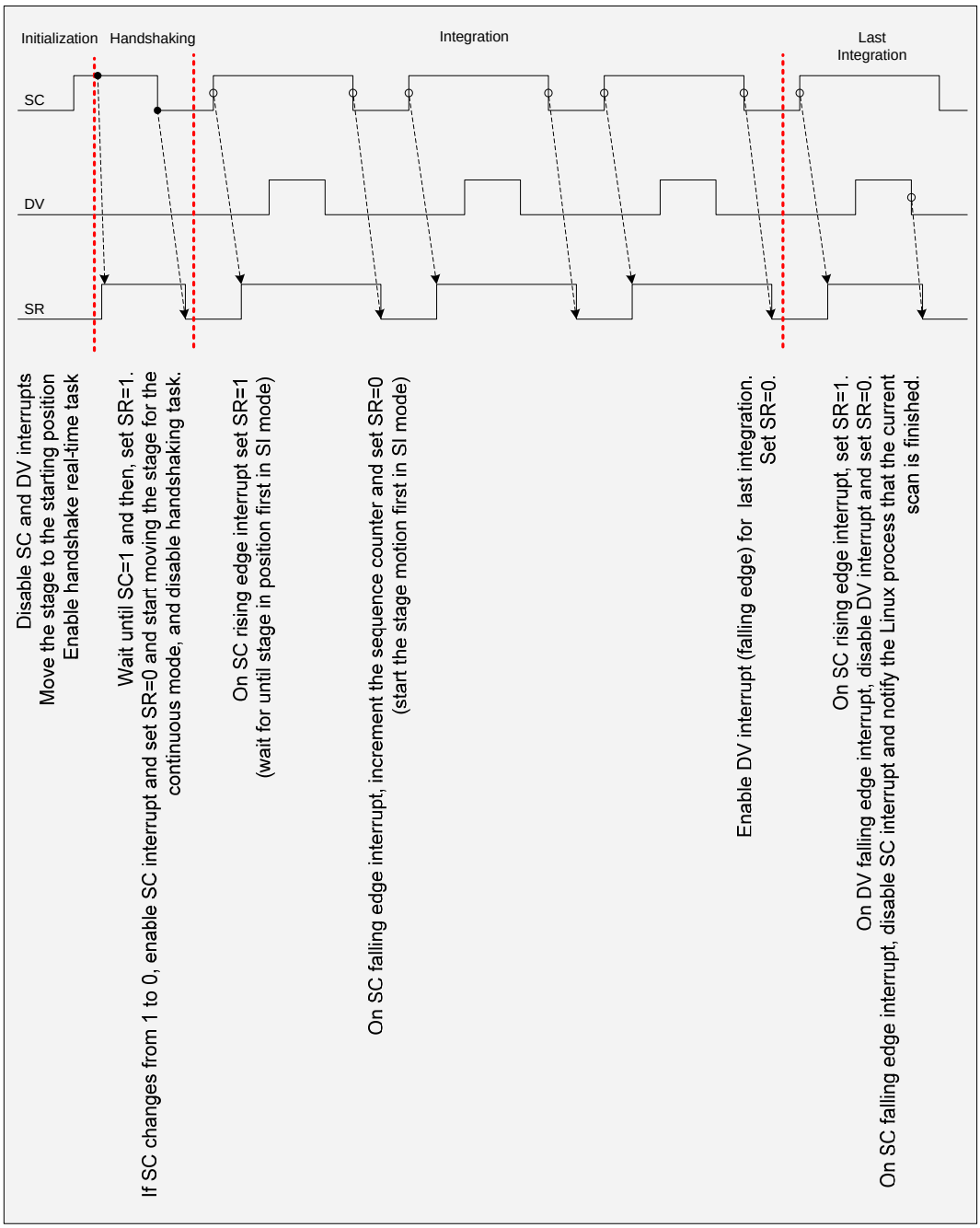

Figure 1. FTS-2 RTS event sequence

3. Standard RTS client interface

In the standard RTS client framework, the function 'rtsClientOpen' is used to obtain an ID pointing to a specific PMC-485 card for all subsequent calls. The function 'rtsClientSetCallback' is used to provide 6 event-driven instrument-specific call-back procedures corresponding to 6 RTS events (or alerts):

- SET_SR_LOW_WHEN_READY
- SET_SR_HIGH_WHEN_READY
- START_INTEGRATION
- STOP_INTEGRATION
- STOP_LAST_INTEGRATION
- ERROR

Their meanings of these events are listed in the following table:

Alert Parameter Value	User Supplied Software Response
SET_SR_LOW_WHEN_READY(0)	Get sequence ready. When ready, respond by calling rtsClientClearSR.
SET_SR_HIGH_WHEN_READ(1)	Get ready for next integration. When ready, respond by calling rtsClientSetSR.
START_INTEGRATION(2)	Begin performing integration, or remain in quiescent state until end of integration.
STOP_INTEGRATION(3)	Stop integration Get ready for next integration.
STOP_LAST_INTEGRATION(4)	Stop integration. Get ready for next command from TODD++.
ERROR(5)	RTS error occurred. Stop everything. Alert the TODD+ of the error. Get ready for next command.

Since all flow control passes to the user supplied software when a callback is performed, care should be taken to only do necessary items in the callback procedures. That way the rtsRealTime task can quickly return to monitoring the RTS.

4. Flowcharts

In this section, in order to clarify the implementation of the FTS-2 RTS Client, the FTS-2 kernel module is divided into five parts according to their functions.

4.1. Initialization of PMC-485 and FTS-2 kernel module

This initialization procedure is only called once, i.e., when the FTS-2 kernel module is loaded into the Linux kernel. Its main function is to set the PMC-485 card and set the communication interface between the kernel module and the Linux user-space process.

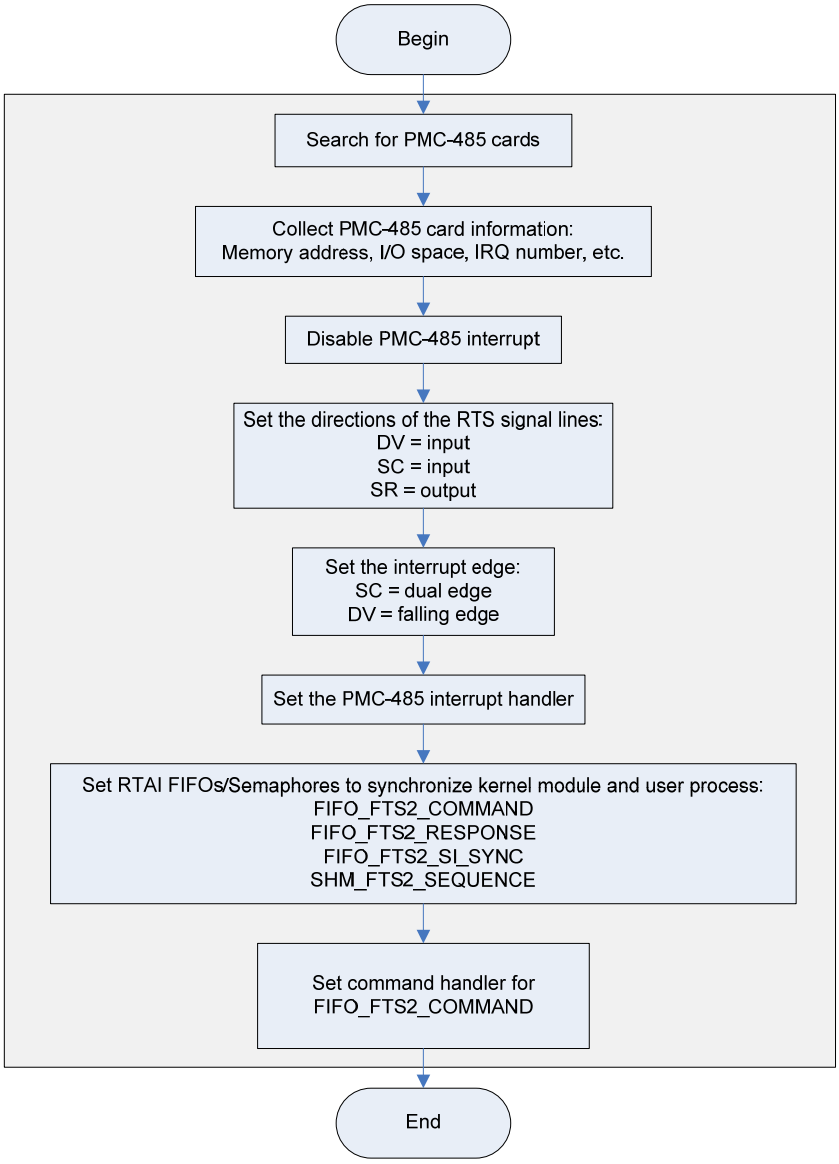

Figure 2. PMC-485 initialization

4.2. FIFO_FTS2_COMMAND command handler

The command handler of FIFO_FTS2_COMMAND provides a communication interface between the FTS-2 kernel module and the Linux user-space process. The two main functions of this command handler are to set scanning parameters and to force the FTS-2 module into its initialization phase.

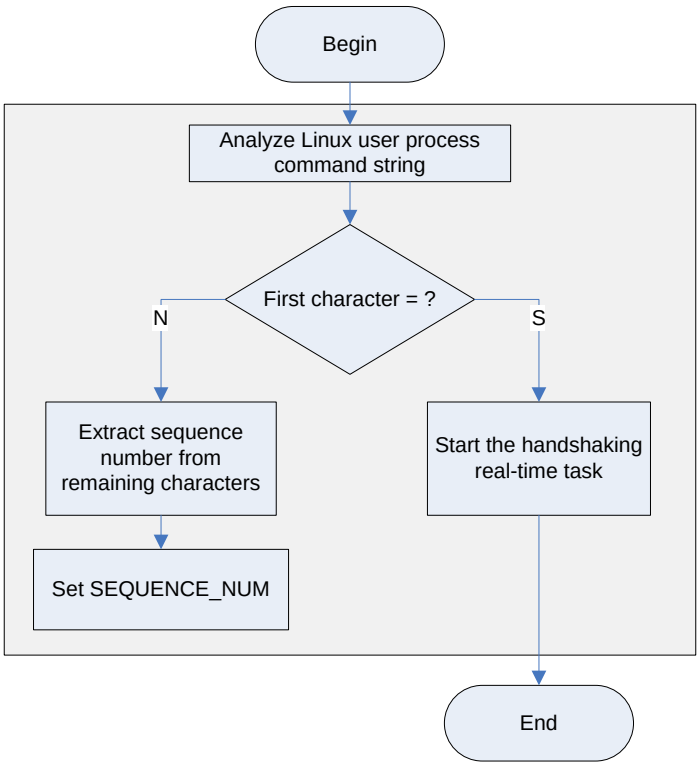

Figure 3. FIFO_FTS2_COMMAND command handler

4.3. Handshaking real-time task

The handshaking task is a real-time periodic task. It uses a polling method to check for SC signal changes, and sets the SR level accordingly.

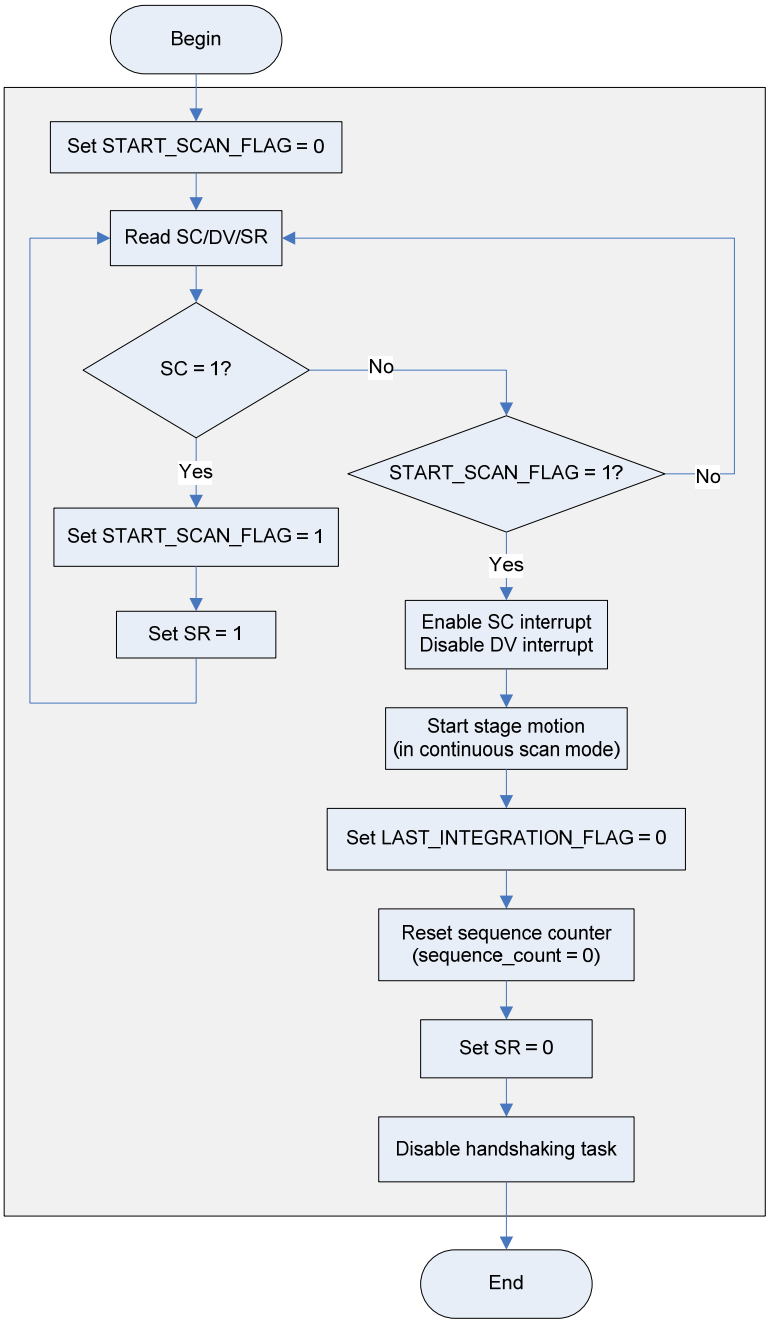

Figure 4. Handshaking real-time task

4.4. Interrupt Handler

The interrupt handler is responsible for processing the changes of SC and DV during the integration phase and the last integration phase.

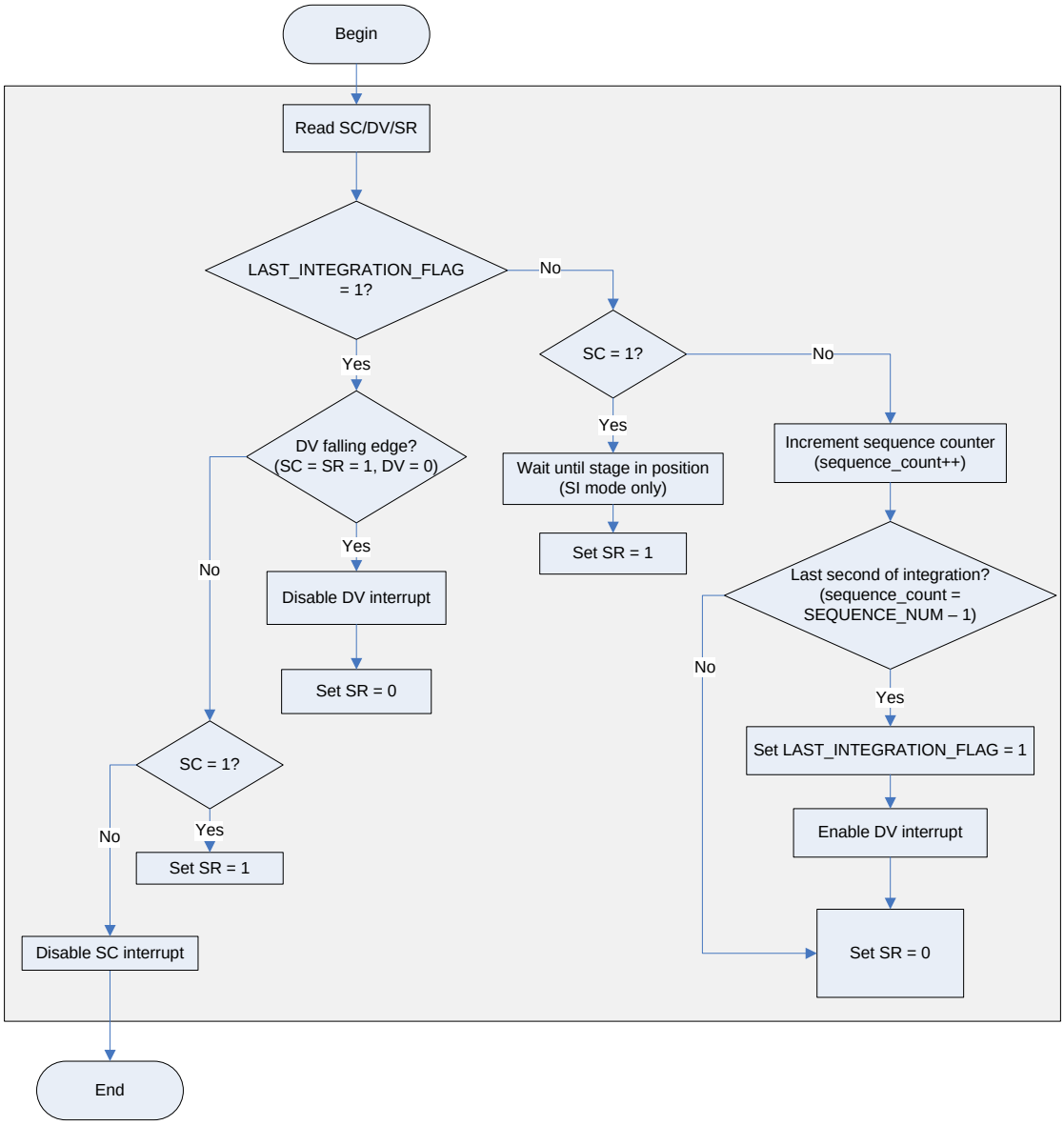

Figure 5. PMC-485 interrupt handler

4.5. Error Handler

The error handler is responsible for processing various errors.

References:

[1] Craig Walther, Ian A Smith, JCMT Real Time Sequencer (RTS) Client Interface Description, RTS/CAW/001, 2002.
[2] B.D.Kelly, Real-Time Sequencer Functional Requirements, RTS/BDK/001.4, 2001.
[3] B.D.Kelly, SCUBA-2 FTS and Polarimeter Coordination, SC2/SOF/S200/026, 2004.