1. Introduction

The SDSU controller has been adapted to drive a Rockwell HAWAII array as part of the ING Camera known as INGRID.

This document briefly describes both the SDSU electronics and in house designed electronics used for driving the array and the also the software which runs in the controller itself to drive the array and interface with the outside world.. This document does not go into any depth in terms of describing how the electronics works as this is neatly described in the standard SDSU documentation of which there are many copies at ING. It gives more a brief overview of the system. It also does not describe the software implemented in the system since hopefully the software files have been well documented enough to be self explanatory. The fibre servicing code is very similar to that supplied by SDSU and which has been described elsewhere and is in fact already in use at ING.

2. INGRID Array Electronics

The array electronics can be broken down into to four parts:-

1. Power Supply unit
2. SDSU Controller
3. Preamp Box
4. Fanout Board

2.1 Power Supply Unit

The power supply unit supplied with the system is the linear PSU as supplied by SDSU. This is an earlier PSU from SDSU and has now been superceded by at least two variants of switched mode power supplies. The PSU comes in a blue grey box and can be mounted up to a few metres from the SDSU controller itself. The PSU has a push button switch which can be used to reset the controller. The PSU chassis is connected to safety earth. The PSU return lines have also been taken to safety earth at this point. The High Voltage 36V supply is not required for HAWAII array operation.

2.2 SDSU Controller

The SDSU controller has been described many times elsewhere. This document will only try to describe those differences from a standard SDSU controller. The reader is referred to the manual supplied by Irlabs, Tucson or the SDSU web site for more detailed information on the boards briefly described here. Those manuals referenced give information on what links to select when setting up the boards etc.
The controller is made up of the following components and PCBs:-

1. Chassis
2. Power Control Board
3. Timing Board
4. Utility Board
5. Clock Board
6. IR Video Board x 2

2.3 SDSU Chassis

This is the standard SDSU metal chassis with the addition of a fan placed to the side for board cooling. Slots are positioned at the other side of the chassis to allow good air flow. The chassis comes with its own RGO designed front faceplate with is fitted with the relevant connectors for the preamp (50way D-type), for the clocks (26 way alpha) and for services such as temperature monitoring (19 way alpha). The chassis body is grounded via a front panel wire connection.

2.4 Power Control Board

This board is mounted on the back of the chassis. There are many different revisions of this control board supplied by SDSU and it's important to know which board has been fitted to which controller. A Revision 3 board is fitted to the INGRID Array controller. Different revisions of boards control in different ways the switch on and off the +/-15V and +36V supplies. The software in the controller monitors the switch on of the supplies and therefore needs to know which way the supplies come on.

2.5 Timing Board

This is the standard Timing board which comes with SDSU controllers. It is fitted with the 50 MHz DSP and an EPROM running the array specific code.

2.6 Utility Board

Again this is the standard Utility board as fitted with standard SDSU controllers. It is running a version of code specific for the HAWAII array operation.

The user board which mounts to the front of the Utility board has been modified to allow operations specifically for use with INGRID. Three constant current supplies have been added to allow control of 3 temperature diodes, connections have been allowed to make to a heater resistor and LEDs have also been added to show shutter and flash status.

2.7 Clock Board

This is the standard board as supplied by SDSU. It has only been populated enough to drive 6 clock lines. It will have to be fully populated if it is required to drive all four array quadrants separately.

2.8 Video board

This board is a non standard SDSU board designed specifically to be used with IR arrays. Each board consists of two analogue channels each configured with a gain stage and its own ADC. These boards also supply the analogue bias voltages required to drive the array.

2.9 Preamp Board

This board was designed in house for use with INGRID. It is a 4 channel differential design with d.c. coupling between array output and op-amps. Each channel has its own separately adjustable d.c. off set trim circuit which can be software set. The board was originally designed to allow the clocks to be routed via its PCB but because of cross talk problems then this option was not implemented. The clocks go directly from the SDSU front panel to the hermetic connector on the cryostat body. The board is mounted in its own metal box enclosure and fits directly over the hermetic connector on the cryostat body.

2.10 Fanout Board

This board was also designed in house for use with the HAWAII array. The array mounts into the socket in this board which is then mounted into the cryostat. The board is of multi layer construction to allow for good low noise performance and to be able to with stand the thermal cycles associated with INGRID.

Most signal lines have been fitted with 1 Mohm resistors and transorb diodes for static protection. The HAWAII array source follower circuits are not used and have been by passed so that external FETs can be used in a source follower configuration. This has been done to reduce FET glow as reported by others. The engineer has the choice of using P type or N type FETs, a P-type J270 has been installed at present. The engineer also has the option to use the on board current load to drive the array cells or to use an external resistor. The board is presently configured to use the on board current source.

The board has also been supplied with a simple diode which allows the board temperature to be checked. A 950nm LED has also been fitted to allow self checking of the array without the use of external lamps etc.

3 INGRID SDSU Software

This section describes the software internal to the INGRID SDSU controller only. This software can be broken down into four separate software areas, namely:-

1. Boot code for Timing Board
2. INGRID Application Code for the Timing Board
3. Boot code for the Utility Board
4. INGRID Application Code for the Utility Board

Each of these areas of software are described in more detail below:-

3.1 Timing Board Boot Code

This is the code which resides on the EPROM and is first executed after a power up or controller reset. To understand how the code functions then one needs to understand the memory mapping for the Timing board. See Memory Maps for details of the memory mapping for the Timing board as implemented here.

This software has been broken down into 6 specific files which are described below:-

Filename:- Compile_timing_bootcode
Type:- UNIX executable script
Description:- this is the file which must be executed to build a version of boot code which can then be programmed into an EPROM.

Filename:- README
Type:- Readme file
Description:- this file gives all the necessary information on the files required for a software build, the checksums expected, how to do the build and whatever other information that I can supply.

Filename:- timing_header.asm
Type:- DSP assembler code
Description:- this file gives all the address information on the TIMING board.

Filename:- timing_initram.asm
Type:- DSP assembler code
Description:- this is a small utility which is used to initialise the RAM areas. It is kept separate to aid program reading only.

Filename:- timing_checksum.asm
Type:- DSP assembler code
Description:- this program is used to calculate a checksum for all the RAM areas in the Timing board. It can be used to verify that the correct boot code is running.

Filename:- timing_bootcode.asm
Type:- DSP assembler code
Description:- this is the main software heart of the timing board. It contains all the code to initialise the timing board and also the code to service the fibres. It cannot operate a HAWAII array without further software being downloaded. On start-up the DSP bootstraps and copies the contents of the EPROM from addresses 0C000 - 0C5FF down to the DSP RAM area 0000-01FF and then jumps to the first instruction at 0000. This has a jump to the initialisation code which starts at 0140 which is executed only once. The initialisation code then jumps to the main Boot code which starts at 0006-013FF.
This software can only execute the following simple commands:-

TDL - test the fibre link
RDM - read a specified memory area
WRM - write to a specified memory area
CHK - run a checksum of the RAM areas

It also allows commands and messages to reroute to the Utility board if required.

3.2 Timing Board Application Code

This is the code which must be downloaded to the SDSU controller. The code is specific to operating a HAWAII array and must not be used for any other detectors. The code can be broken down into 5 specific files as described below:-

Filename:- Compile_timing_application
Type:- UNIX executable script
Description:- this is the file which must be executed to build a version of application code which can then be downloaded to the SDSU Timing board.

Filename:- README
Type:- Readme file
Description:- this file gives all the necessary information on the files required for a software build, the checksums expected, how to do the build and whatever other information that I can supply.

Filename:- timing_testgenerator.asm
Type:- Readme file
Description:- this software is used to produce dummy test data to send to the host system.

Filename:- hawaii_tables.asm
Type:- Readme file
Description:- this file contains all the sequences required by the DSP sequencer to drive the clocks to the HAWAII array. It also contains all the values required to set the voltages to the array.

Filename:- hawaii_application.asm
Type:- Readme file
Description:- this is the heart of the software required to drive the HAWAII array. It contains all the clock sequences needed to readout an array. It also has the code to allow all the other operations normally associated with reading out an array such as setting exposure time. The commands available are described below:-

CON - switch power on to the array
COF - switch power off from the array
SET - set the exposure time
DAT - set the data type to dummy test data
TST - put the clocks into test mode
MRA - do a multiple none destructive readout of the array

3.3 Utility Board Boot Code

This is the code which resides on the EPROM and is first executed after a power up or controller reset. To understand how the code functions then one needs to understand the memory mapping for the Utility board. See Maps for details of the memory mapping for the Utility board.
This software has been broken down into 6 specific files which are described below:-

Filename:- Compile_utility_eprom_bootcode
Type:- UNIX executable script
Description:- this is the file which must be executed to build a version of code which can then be used to program the EPROMs on the Utility board.

Filename:- README
Type:- Readme file
Description:- this file gives all the necessary information on the files required for a software build, the checksums expected, how to do the build and whatever other information that I can supply.

Filename:- utility_header.asm
Type:- DSP assembler code
Description:- this file gives all the address information on the Utility board.

Filename:- utility_initram.asm
Type:- DSP assembler code
Description:- this is a small utility which is used to initialise the RAM areas. It is kept separate to aid program reading only.

Filename:- utility_checksum.asm
Type:- DSP assembler code
Description:- this program is used to calculate a checksum for all the RAM areas in the Utility board. It can be used to verify that the correct boot code is running.

Filename:- utility_bootcode.asm
Type:- DSP assembler code
Description:- this is the main software heart of the utility board. It contains all the code to allow communication from the utility board to the outside world. It also contains the same basic set of commands as the timing board boot code, namely TDL, RDM and WRM.

3.4 Utility Board Application Code

This is the code which must be downloaded to the SDSU controller Utility board. The code executes all the house keeping functions associated with operating a HAWAII array, such as temperature checking and controlling shutters etc. The code can be broken down into 5 specific files as described below:-

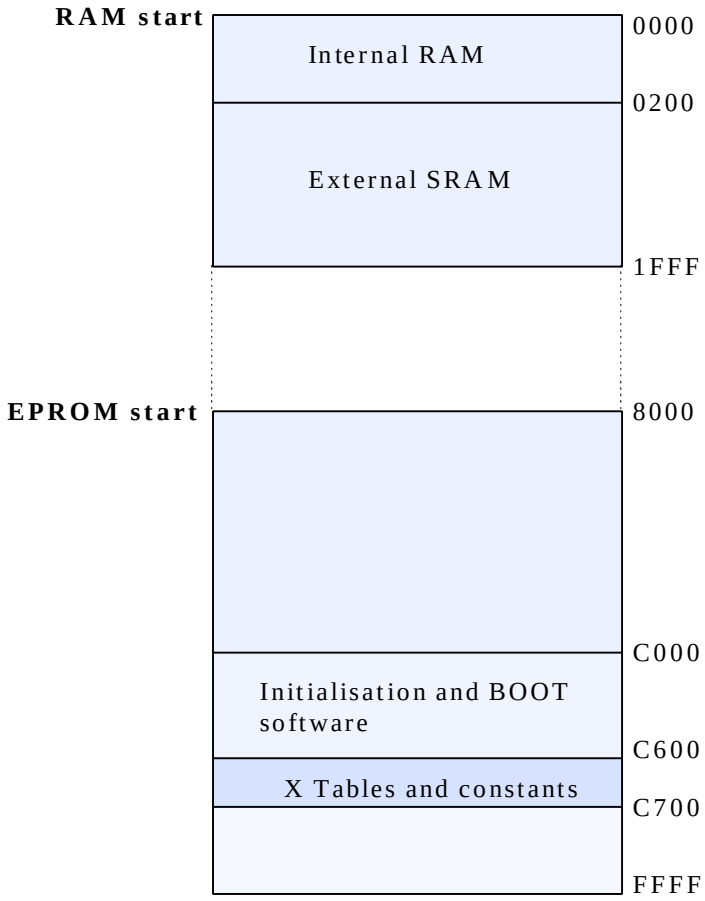
Filename:- Compile_utility_application
Type:- UNIX executable script
Description:- this is the file which must be executed to build a version of application code which can then be downloaded to the SDSU Utility board.

Filename:- README
Type:- Readme file
Description:- this file gives all the necessary information on the files required for a software build, the checksums expected, how to do the build and whatever other information that I can supply.

Filename:- utility_application.asm
Type:- Readme file
Description:- this is the heart of the software required to perform the house keeping functions associated with the SDSU controller diving the array. This code is downloaded to the Utility board after reset or power up. The code runs on a 1ms interrupt allowing it to update its inputs and outputs on that timescale. The present implementation allows three separate temperature channels to be read, can send an open/close shutter pulse, can switch an internal LED on/off and also supplies a proportional heater servo loop to allow control of the array temperature. The commands available after this code has been downloaded are as follows:-

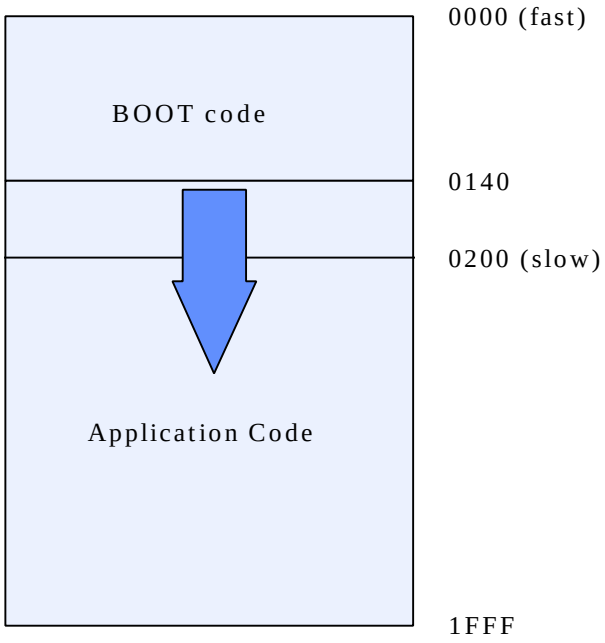
OSH - open shutter
CSH - close shutter
LON - switch LED on
LOF - switch LED off
PON - switch +/- 15V supplies but not supplies to the aray
POF - switch the supplies off

Timing Map 1 - P memory space



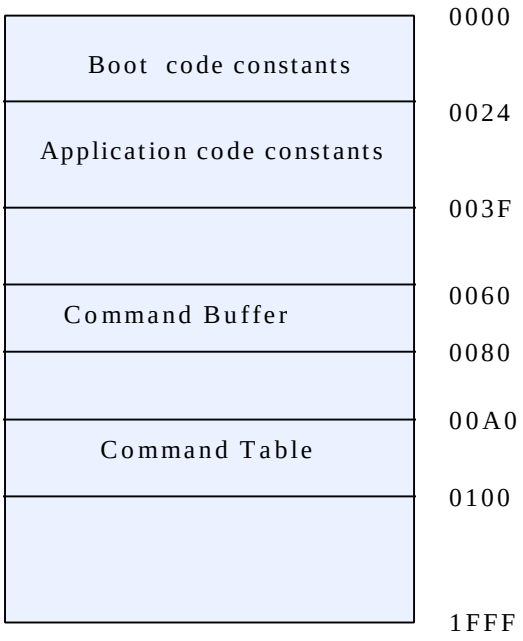
Map shows contents of P memory space in the SDSU timing board for the INGRID application. It is completely different to that supplied by SDSU. On power up the DSP bootstraps and copies the contents of the EPROM into its internal RAM area. It then runs the code from the internal RAM. The intialisation code is copied to P:0140. This is the same area that the application code gets written to when the application program is downloaded to the controller. This means that the initialisation code is “lost” after an application is downloaded but this does not matter since the initialisation code is run only once. The EPROM also contains all the constants and values required for the code to function properly and these are copied to X memory on power up.

Timing Map 2 - P RAM space



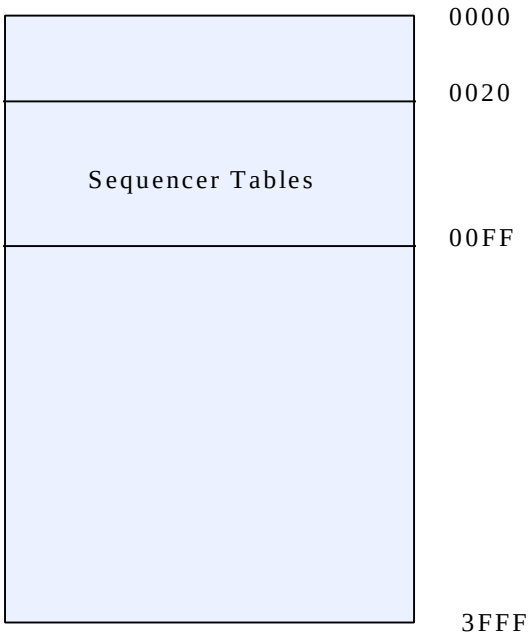
This shows the internal P memory area after the application code has been downloaded. This code overlaps both the internal fast DSP RAM and the slower external RAM. This means that it is important to ensure that the code required to run as fast as possible sits in the internal low memory area. This area is also limited so that very efficent code needs to be written to fit in here. All the sequencer routines for reading out the array are stored from P:0140 to P:01FF.

Timing Map 3 - X RAM space



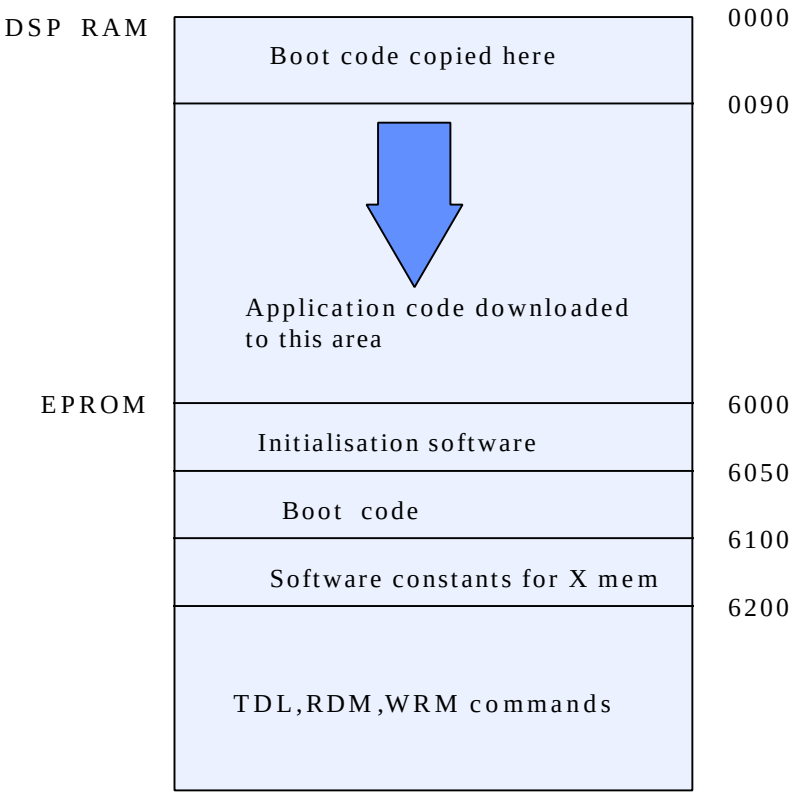
This map shows the contents of the DSP RAM areas and their usage after power up. Even though this area is large there are still constraints on, for example, the area for constants storage. This is because we want to use short addressing modes if possible to save on P memory space. The checksum program only tests above X:0080 because the command ring buffer is stored below this. All command that are received from the fibre go to this ring buffer memory area so that if we were to test this area then the simple act of sending the checksum command would change the checksum each time the checksum command was sent.

Timing Map 4 - Y RAM space



The values used to set the DAC voltages to set the bias voltages for the array are stored here along with sequences themselves which are used to produce the clocking routines to readout the array.

Utility Map 1 - P memory space



The memory map for P memory space is shown above. The A15 address line so that at bootstrap when the DSP jumps to address 0E000 it in fact goes to EPROM at address 06000 where it then executes the initialsation code. The bootcode gets copied to P RAM but the main command code is executed directly from EPROM since it is not time critical.
The X and Y memory spaces are very simple. In X, constants are stored in low memory and two command circular buffers are set up from X:0080-X:00c0.
In Y, in low memory a copy is kept of the status of all the I/O values to the Utility board - look to the SDSU documentation for more detail or to the software programs themselves.