

Firmware for the Hawaii Muon Beamline Scintillating Tracker Planes and Modular RICH Hodoscope

Author: Nathan Park | park.nathan@gmail.com

1. Introduction

Analyzing the results of a high energy physics experiment is a lot like investigating a crime scene. A physicist must reconstruct the events that took place by analyzing the by-products of the experiment. Common by-products are elementary particles that travel at very high speeds, packed with a lot of energy. Like a crime scene investigator, it is the job of the particle detector to collect data on certain aspects of these particles and send this data back to the physicist. Hawaii Muon Beamline (HMB) and the Modular RICH (mRICH) are two systems of multiple particle detectors that used to reconstruct events. They are similar in design, but serve two different projects. The HMB serves as a testbed for other particle detectors being developed at the UH Instrumentation Development Lab. The mRICH is a proton detector that will be used in an experiment at Fermi Lab.

Both systems have a particle detector that is responsible for triggering the readout of all other subsystems. This triggering sub-system is focus of this document. On the HMB the sub-system is called the Scintillating Tracker Planes, and on the mRICH it is called the Hodoscope. It uses scintillating planes to detect passing particles and monitor their trajectories. The planes emit light as a particle passes through them. This light is picked up by photo-diodes, which convert it into electrical signals. Each plane has an array of 16 photo-diode channels. These channels serve to as a grid to mark the position of the particle. Both systems capture the entrance and exit position of the particle, which is used by a PC to calculate the particle's trajectory. The data acquisition electronics for both systems are essentially the same. Each plane is assigned a daughtercard (DC) that monitor the photo-diodes. On the DC is a TargetX ADC digitizes and stores diode signals and produces a 5-bit trigger signal whenever a particle is detected. Encoded in the signal are the channels that saw the particle. The TargetX is driven by a Spartan6 FPGA. The FPGA, collects data from the TargetX and programs its settings. The DCs are controlled by one Standard Controls and Read Out, and Data (SCROD) master board, which has its own FPGA that reads out data from the DCs and passes it to the user's PC.

The main difference between the HMB Tracker Planes and the Hodoscope, is that the Hodoscope only captures trigger data. The HMB collects digitized waveforms of the diode signals in addition to trigger data. Another difference is the number of planes. The Hodoscope has four planes altogether grouped in pairs. One pair measures the x-y coordinates of the entrance position, and the other measures the coordinates of the exit position. The HMB Tracker Planes has 8 planes grouped into four pairs. Two pairs measure the entrance position, while another two pairs measure the entrance position. The extra two pairs are used to verify the presence of a particle.

This report documents the firmware (FW) on the SCROD and DC FPGAs. Section 2 describes the firmware design and the basic hardware components for each board. Section 3 delves into the experiments used to test the firmware, as well as testbench simulations that can be used to better understand the behavior of firmware modules. This project is still a work in progress. Section 4 goes into the functionalities that still need to be developed. Section 5 provides instructions on how to get started in using the project.

2. Firmware Design

The SCROD Rev A5 master board receives commands from the PC, relays commands to the DCs, and collects data from the DCs for the PC. Communication between PC and SCROD is maintained through a fiber optic Ethernet interface, while communication between the SCROD and DCs are maintained through CAT-6 cables terminated with RJ45 connectors. The main components of the SCROD board include [2]:

1. PCI express optical gigabit transceiver: interfaces with PC, sends and receive packets of information to and from PC.
2. Xilinx Spartan-6 FPGA(1x150T package): parses commands, communicates with Daughtercards, processes data, and sends data to and receives data from the optical transceiver.

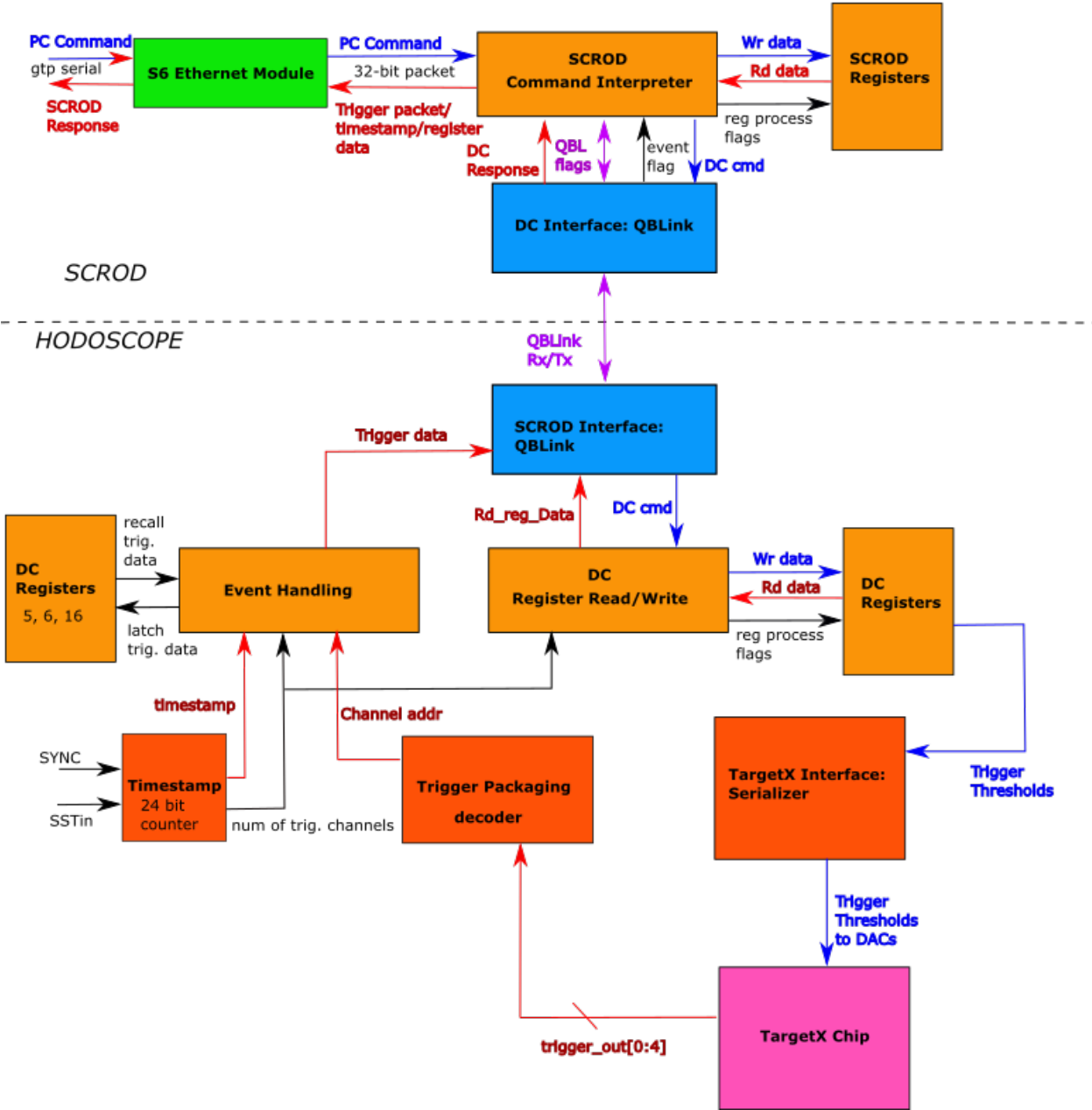
The SCROD is mounted to an interconnect board, called the SCROD to RJ45 Board. The interconnect board contains a Molex power connector for the SCROD and 8 RJ45 ports for connection to 8 tracking plane Daughtercards [3]. The SCROD Rev A5 was designed by Xiaowen Shi and the SCROD to RJ45 Board was designed by Khanh Le.

The HMB/Hodoscope DC monitors the photo-diode channels for particle events. The TargetX contains comparators for each channel [4]. If one the channel voltages exceed the comparator threshold, the TargetX will send out the trigger signal with the channel number encoded into a 5-bit word [4]. The HMB DC also can readout waveform data, though this feature is still under development. The main hardware components of the HMB are [5]:

1. Ultravolt High Voltage: Provides bias for the MPPCs
2. Trim Digital to analog converter (DAC): Sets Ultravolt bias
3. Amplifier stages: Convert MPPC current signals to amplified voltage signals
4. TARGETX: Advanced analog to digital converter used to collect, store and digitize MPPC signals
5. Xilinx Spartan-6 FPGA (1x4 for Hodoscope, 1x9 for HMB, package 3tqg144): TargetX control, data collection, data readout, and communication with SCROD

The Daughtercard board was designed initially by Khanh Le and later revised by Tommy Lam. The block diagram in figure 1 depicts the major data paths between the SCROD and DC.

SCROD-HODOSCOPE Block Diagram



2.1 SCROD Firmware

Figure 2. shows the top level block diagram of the SCROD Firmware.

SCROD Firmware for Beam Telescope Data Acquisition

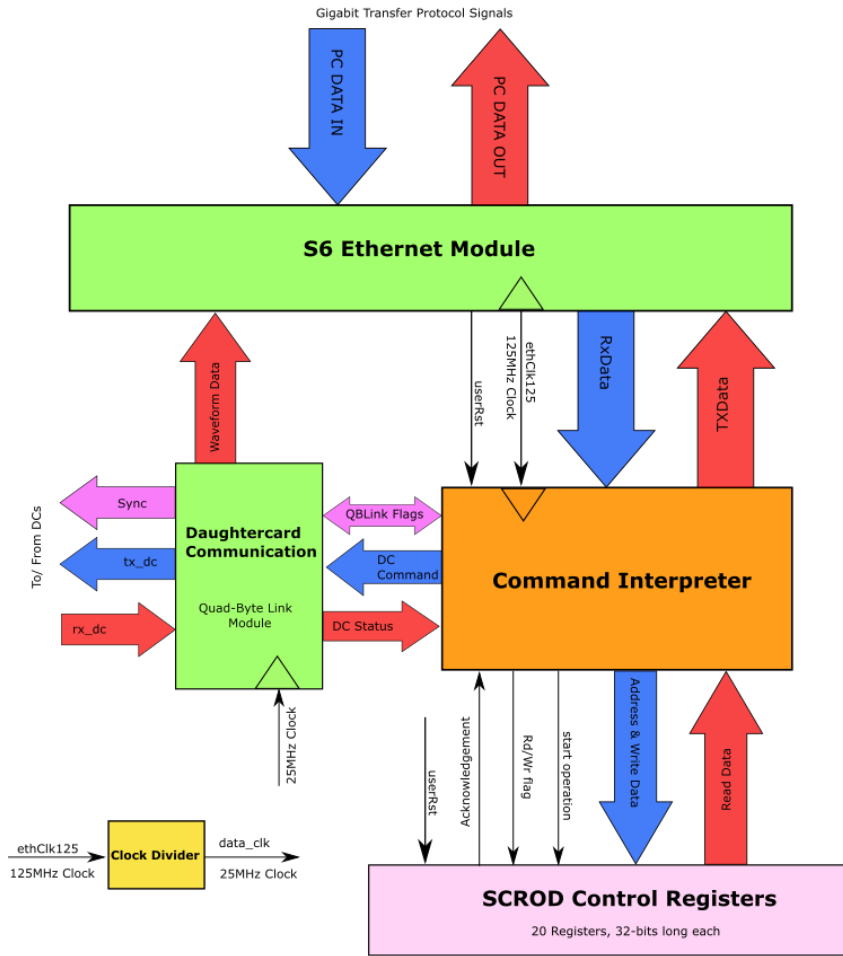


Figure 1. SCROD Top file block diagram. The blue arrows show the paths of the input data from the PC. The red arrows show the paths of the output data going to the PC.

Currently, the SCROD firmware for HMB is identical to that of Hodoscope, as there is no waveform data handling. The Waveform Data arrow is just a placeholder. There are four major functional blocks in the SCROD. First is the S6 Ethernet Module. This module was developed by Kurtis Nishimura for the iTOP project. The previous version of the FW used a Xilinx generated IP core ethernet module, which timed out after a period of time. The S6 has the advantage of not timing out, and is a more reliable ethernet module. 32-bit words are transmitted serially over the PCI express gigabit transceiver between the SCROD and PC. The S6 runs on a 125 MHz clock supplied by the transceiver. It then feeds this clock into the Clock Fanout (Clock Divider on the Diagram) to generate the 25 MHz Data clock. Table 1 lists the signals included PC DATA OUT and PC DATA IN groups.

Table 1. Signals between SCROD and Gigabit Transceiver (PC Interface)

<i>PC DATA IN</i>	
Signal	Description
gtRx	Serial PC data in (differential)
gtClk	GTP Clock (differential)
fabClk	Alternative Clock, used in fabric (differential)
<i>PC DATA OUT</i>	
Signal	Description
gtTx	Serial PC Data Out
txDisable	Disable Transceiver Output

The next major block is the Command Interpreter. It is the central block of the SCROD that parses PC commands, programs control registers, and sends Trigger Data or Responses to PC commands. Table 2 shows the signals that move between the Ethernet Module and the Command Interpreter.

Table 2. Signals between S6 Ethernet and Command Interpreter

<i>From Ethernet (RX)</i>	
Signal	Description
userRxData (2 32 bit channels)	Two 32-bit channels of incoming PC data
userRxDataValid(2 bits)	valid incoming word flag
userRxDataLast (2 bits)	last word in incoming packet flag
userRxdataReady (2 bits)	Interpreter ready to receive data flag
<i>To Ethernet (TX)</i>	
Signal	Description
userTxData(2 32 bit channels)	Two 32-bit channels of outgoing PC Data
useTxDataValid (2 bits)	2-bit flag: Valid outgoing word
userTxDataLast (2 bits)	2-bit flag: last word in outgoing packet
userTxdataReady (2 bits)	2-bit flag: Ethernet ready to transmit data

The SCROD has an array of 16-bit control registers that are programmed by a process, SCROD_Ctrl_Reg. This process is represented by the SCROD Control Registers Block. The process reads and writes to the registers. The Command Interpreter starts the process by raising the Start Operation flag (regReq). The process acknowledges the Interpreter by raising the Acknowledgement flag (regAck). The Rd/Wr flag (regOp) sets the operation type. If it is '1' the addressed register will be written to. If it is '0' the addressed register will be read. A description of all signals involved in the process is found in Table 3. A list of all control registers on the SCROD along with their function can be found in Table 4. Most of the registers come from an earlier version of the FW and need to be reimplemented.

Table 3. Command Interpreter-Control Register Interface

<i>SCROD Register Address and Write Data</i>	
Signal	Description
regAddr (16 bits)	register address (acceptable values: 0 to 15)
regWrData(16 bits)	register value
<i>SCROD Register Read Data</i>	
Signal	Description
regRdData(16 bits)	current register value
<i>Register Command Flags</i>	
regReq	begin a register operation
regOp	Read or Read mode
reset	reset command from Ethernet Module
regAck	Register handshake

Table 4. SCROD FPGA Control Registers

Reg #	bits	Value
0	0	Resets state machines
1	0	SCROD ID
2	8, 7* downto 0	Sync Daughtercards, reset QBLink (bit i resets QBLink module i, *7 downto 0 for HMB, 3 downto 0 for HODOSCOPE).
3		<i>not used</i>
4		<i>not used</i>
5	0	<i>software Trigger (not implemented)</i>
6	3 downto 0	<i>Trigger Mode (not implemented)</i>
7	3 downto 0	<i>dc_mask (not implemented)</i>
8	3 downto 0	<i>output mode (not implemented)</i>
9		<i>not used</i>
10		<i>not used</i>
11		<i>not used</i>
12	15, 8 downto 0	enable use of fixed window, fixed start window
13		<i>not used</i>
14		<i>not used</i>
15		<i>not used</i>
16		<i>not used</i>
17		<i>not used</i>
18	4 downto 0	<i>maximum trigger count (not implemented)</i>
19	11 downto 0	<i>trigger count enable(not implemented)</i>

20	15 downto 0	1/2 trigger scalar count (not implemented)
21	0	2/2 trigger scalar count (not implemented)

The fourth major block is the Daughtercard Communication block, which provides communication links between the SCROD and DCs using a custom protocol called QBLink. It passes commands from the Command Interpreter to selected DCs and passes data from the DCs to the Command Interpreter. It also monitors all incoming DC data for particle detection events. If all DCs see a particle a global event flag will be raised to trigger readout of the trigger data and all other detectors. Table 5 lists the signals between the Command Interpreter and DC Communication modules. Table 6 list signals going between the Daughtercard Communication block and the DCs.

Table 6. Signals between SCROD and Daughtercards

<i>To DC</i>	
Signal	Description
tx_dc(8 bits)	QBLink serial output to DCs
Sync (8 bits)	Synchronize all DCs, resets their timestamp counters and QBLink modules.
<i>From DC</i>	
Signal	Description
rx_dc(8 bits)	serial input from DCs

The following sections will go through the 4 blocks in more detail.

2.1.1 Command Interpreter

There are four kinds of commands that the Command Interpreter parses: SCROD Register commands, DC register commands, device pinging, and data readout. Both the SCROD and DC FPGAs have internal registers that store control settings for the peripheral devices, such as the trim DACs. These registers can be written to and read by the user's PC through register commands. The user can check if all communication links are established between the PC, SCROD, and DCs with ping commands. Register and ping commands are sent by the PC through Ethernet packets. The packets consist of 32-bit words that must be sent in a certain order so that the Command Interpreter can parse the command. These words include a header, packet type, device address, command type, and the actual commands themselves. There are scripts included in the project files that handle the construction and sending of command packets.

Table 7 and 8 shows the packet format for a register and ping command. The Interpreter will send a response packet to the PC for each command that it receives. The second and third word in the response packet indicates if the command was parsed successfully or if there was an error. The scripts used to send commands to SCROD will display the response packet to the user. If the register command was successfully parsed, the second word will say x00000006 and the third word will display the constant WORD_ACK_C, x6F6B6179. If there was an error in

parsing the command (i.e. the packet was not correctly ordered), the second word will display x00000005 and the third word will display the constant WORD_ERR_C, x7768613f. The sixth word will be the error flag indicating what went wrong. Table 8 and 9 shows the response packets for successful write and read register commands. Table 10 shows the response to a ping command. Table 11 shows the response packet in the event of an error. Table 12 describes the error flags.

Table 7. Register Command Packets

Word	High Bytes		Low Bytes		Description
	31:25	24:16	15:8	7:0	
0	0x00BE11E2				Header word
1	packet size				Number of remaining words except packet checksum: 6
2	0x646f6974				Labels packet as a Register Command
3	0x00	Device label	Device #		Device labels: (SCROD) 0x00A5, (DC) 0x00DC Device #: 0x00 (SCROD), 0x01 through 0x08 (DC #)
4	verbosity (31:24)	Command ID(23:0)			Verb(7): suppresses command response. Command ID: unique ID to each command
5	command type				Ping: 0x70696e67, read: 0x72656164, write: 0x72697465
6	Register Value (0x0000 for read operations)	register address			Command data
7	command checksum				sum words 4 through 6
8	packet checksum				sum of entire words 0 to 7

Table 8. Ping Command packets

Word	High Bytes		Low Bytes		Description
	31:25	24:16	15:8	7:0	
0	0x00BE11E2				Header word
1	packet size				Number of remaining words except packet checksum: 6
2	0x646f6974				packet type: device configuration
3	0x00	Device label	Device #		Device labels: (SCROD) 0x00A5, (DC) 0x00DC Device #: 0x00 (SCROD), 0x01 through 0x08 (DC #)
4	verbosity (31:24)	Command ID(23:0)			Verb(7): suppresses command response. Command ID: unique ID to each command

5	0x70696e67	Command Type: Ping
6	command checksum	sum words 4 through 6
7	packet checksum	sum of entire words 0 to 7

Table 9. Register Write Response

Word	Bytes
0	x"00BE11E2"
1	x"00000006"
2	WORD_ERR_C = 0x7768613f.
3	wordScrodRevC = x"0000A500"
4	00 & commandID
5	WORD_WRITE_C =
6	r.regWrData & r.regAddr
7	Checksum

Table 10. Register Read Response

Word	Bytes
0	x"00BE11E2"
1	x"00000006"
2	WORD_ERR_C = 0x7768613f.
3	wordScrodRevC = x"0000A500"
4	00 & commandID
5	WORD_READ_C = x"72656164"
6	r.regWrData & r.regAddr
7	Checksum

Table 11. Error Response Packet

Word	Bytes
0	x"00BE11E2"
1	x"00000005"
2	WORD_ERR_C = x"7768613f"
3	wordScrodRevC = x"0000A500"
4	00 & commandId
5	errFlags
6	Checksum

Table 12. Error Flags

Error Constants (errFlags)
Packet size error: ERR_BIT_SIZE_C = x"00000001"
Packet type error: ERR_BIT_TYPE_C = x"00000002"
Invalid command target: ERR_BIT_DEST_C = x"00000004"
Invalid command type: ERR_BIT_COMM_TY_C = x"00000008"
Command checksum error: ERR_BIT_COMM_CS_C = x"00000010"
Packet checksum error: ERR_BIT_CS_C = x"00000020"
Timeout Error: ERR_BIT_TIMEOUT_C = x"00000040"
connection to DC failed: QBLINK_FAILURE_C: x"00000500"

This module contains several attributes, which can be tweaked in the top file. These are listed in Table 13.

Table 13. Command Interpreter Attributes

Attribute	Description
REG_ADDR_BITS_G	length of register address, integer
REG_DATA_BITS_G	length of register data, integer
TIMEOUT_G	number of clock cycles to wait for register response, integer
GATE_DELAY_G	time (ns)
num_DC	number of DCs

A structural view of the Command Interpreter can be seen in Figure 3. There are two combinatorial processes, the SCROD Combinatorial Process (SCRODRegComb) and the SendTrigger Process (SendTrigger). The former listens to the Ethernet module for PC register command packets, which parses. If the command is intended for the SCROD, it will interact with the Control Registers through the regOut and regIn signals, which include all signals listed in Table 3. If the command is intended for one or more DCs it will pass the command type and command data out to the target device(s) through DC_CMD. The SendTrigger process listens to the global event flag (EVNT_FLAG). If it is '1' it will collect and send trigger packets from all devices to the PC. The processes output a RegType record (rin and tin, color coded brown) that holds data collected in the process (see Table 14.). This information is latched by the Sequential Process by the positive 125 MHz clock edge. The latched data is stored in record signals r (for register process) and t (for trigger process).

In the event of a global trigger, all register commands are dropped and SendTrigger process gains control over the TxData channel to the Ethernet Module. This is ensured through the GTP Tx MUX (gtp_MUX).

Table 14. Signals Included in RegType record

Signal	Description
state	State machine states
regAddr	FPGA register address logic vector(REG_ADDR_BITS_G-1 downto 0)
regWrData	Register data to write logic vector(REG_BIT_DATA_G-1 downto 0)
regRdData	Read register value logic vector(REG_BIT_DATA_G-1 downto 0)
regReq	standard logic bit
regOp	standard logic bit
sendResp	standard logic bit
rxDataReady	standard logic bit
txData	standard logic bit
txDataValid	standard logic bit
txDataLast	standard logic bit
wordsLeft	words left in packet, logic vector(31 downto 0)
wordOutCnt	# of words sent to PC, logic vector(7 downto 0)
checksum	logic vector(31 downto 0)
deviceID	logic vector(31 downto 0)
commandType	logic vector(31 downto 0)
command	logic vector(31 downto 0)
commandID	logic vector(31 downto 0)
noResponse	setting to disable SCROD response to PC, standard logic bit
errFlags	logic vector(31 downto 0)
timeoutCnt	Timer for timeout error, logic vector(31 downto 0)

Command Interpreter: Structural View

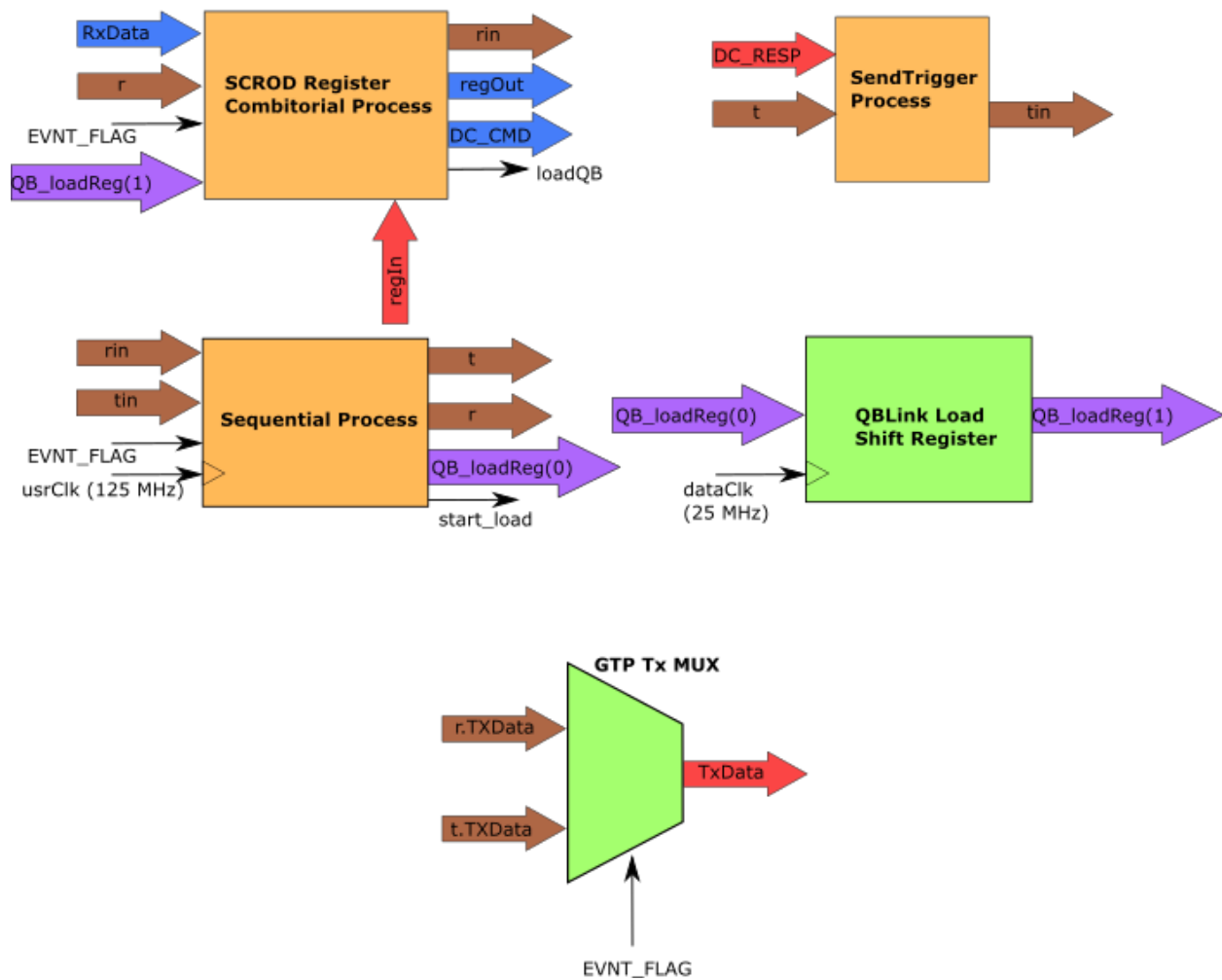


Figure 3.

The state machine for the SCROD Register Combinatorial Process is complicated and long, so it can be referenced in figure 1 in the appendix of this document. Two conditions can reset the state machine back to the IDLE_S state: when `usrRst = '1'` or when `EVNT_FLAG = '1'`.

The state machine of the SendTrigger Process is depicted in Figure 4. While there is no event (`EVNT_FLAG = '0'`), the process is in the IDLE mode and the SCROD Register process has control over the TxData channel. When there is an event it will send the trigger packet of each DC over the Tx channel to the PC.

Command Interpreter: Send Trigger Process

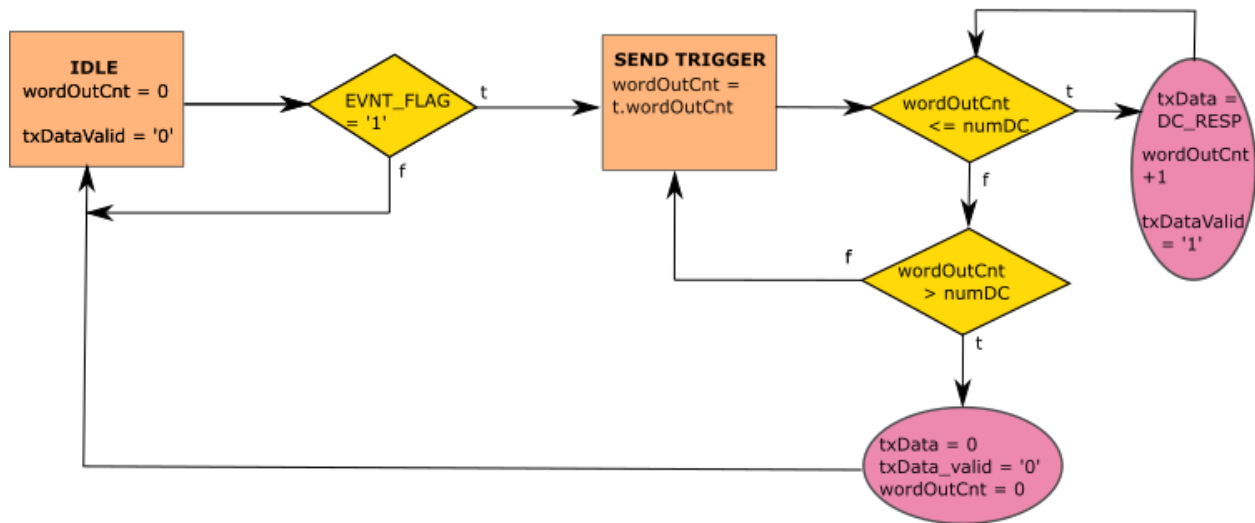


Figure 4.

2.1.2 Daughtercard Communication

This module has one attribute, numDC, that denotes the number of DCs. The module creates a QBLink module for each DC. Each QBLink module has a partner on the DC side. On startup, the partners go through a training phase. It has been found that the modules should be reset before beginning the training phase. Otherwise, they do not pass the training phase (Refer to Control Register 2).

The TriggerLogic process listens to the output word of each QBLink module. It searches for the trigger event marker, “111” in the output word, dc_data. If dc_data[i][7 downto 5] has this marker, then DC i had a trigger event and the process raises TrigFlag[i] to ‘1’. The EVNT_FLAG (internally known as evnt_trig) ANDs together all the bits in the TrigFlag logic vector. When the EVNT_FLAG is raised the trigger packets received from the QBLink output words is sent one by one to the Command Interpreter for it to send to the PC on the rising clock edge of the 25 MHz Data_Clk.

2.2. Daughtercard Firmware

The top-level block diagram of the DC FW is shown in Figure 5. Many of the signals shared between the DC and SCROD have a different name on the DC side. Signal mapping between the SCROD and DC can be found in Documentation folder in the project repository (see section 5). The DC has a differential clock input SYSCLK_P/N sourced by the SCROD. The Clock Fanout Module uses the clock input to generate the single ended sysclk (25 MHz) and asic_clk (62.5 MHz). The asic_clk frequency is divided to generate SSTIN (31.25 MHz). The sysclk runs most of the FPGA’s internal processes and modules; most notably, those that interact with the SCROD Communication Module, which contains the QBLink partner. The asic_clk runs processes and modules that interact with the TargetX ASIC, such as the TargetX DAC Controller and DAC Update Process.

HMB/Hodoscope Daughtercard Firmware: Top Level Block Diagram

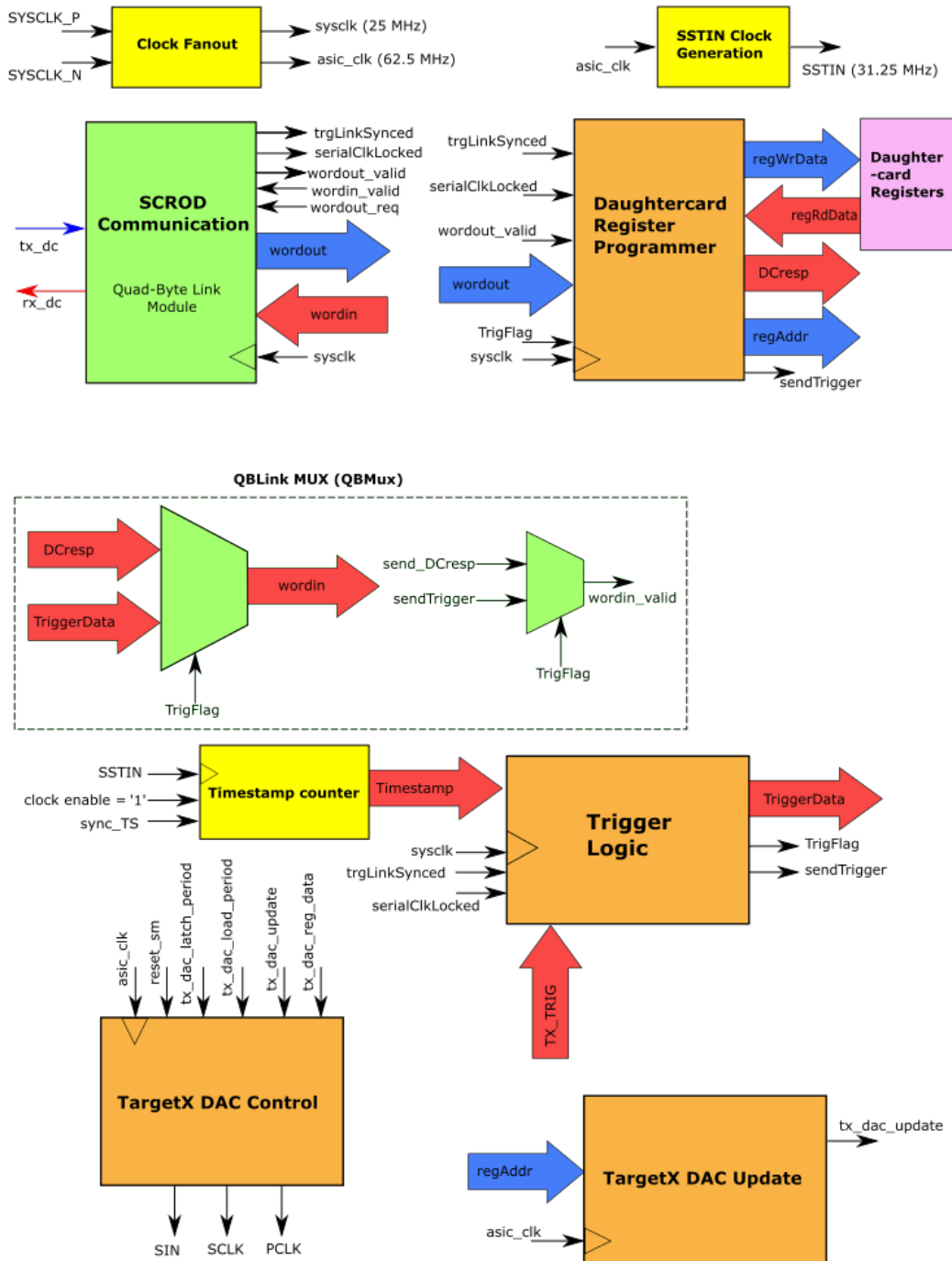


Figure 5.

There are two main signal groups that connect to the QBLink module: Trigger Data signals and the Register Command signals. The QBLink MUX (QBMux) ensures only one of these signal groups have access to the QBLink module at any given time. By default, the Register Command Signals are active on QBLink until there is a trigger event, in which case the Trigger Data has control over QBLink.

2.2.1 Daughtercard Registers

The DC has internal registers, whose functions are described in Table 15. The Daughtercard Register Programmer process writes and reads the registers. The process receives register command type (either read or write) and register data (such as the target register number) from the SCROD Communication Block.

Table 15. Daughtercard FPGA Control Registers

Reg #	bits	Value
0	0, 4	TargetX control reset, enables calibration
1	6 downto 0	tx_dac_reg_data
2	11 downto 0	tx_dac_reg_data
3	15 downto 0	tx_dac_load_period
4	15 downto 0	tx_dac_latch_period
5	0	<i>software Trigger (not implemented)</i>
6	3 downto 0	<i>Trigger Mode (not implemented)</i>
7	3 downto 0	<i>dc_mask (not implemented)</i>
8	15 downto 0	SCROD acknowledgement wait
9	15, 8 downto 0	offset window direction, number of windows to offset
10	8 downto 0	number of windows to readout
11	15 downto 0	<i>not used</i>
12	15, 8 downto 0	enable use of fixed window, fixed start window
13	15, 3 downto 0	enable pedestal subtraction, number of averages for calculating peds (1-7)
14	15, 1 downto 0	resets window number during sampling, sets timing for sstin and current window count in targetX
15	4 downto 0	Trim DAC address
16	11 downto 0	Trim DAC voltage
17		mppc dac voltage
18	4 downto 0	<i>maximum trigger count (not implemented)</i>
19	11 downto 0	<i>trigger count enable(not implemented)</i>
20	15 downto 0	<i>1/2 trigger scalar count (not implemented)</i>
21	0	<i>2/2 trigger scalar count (not implemented)</i>

After each register command, the DC sends a response (DCresp) to the SCROD that acknowledges the command and sends back any requested data.

2.2.2 TargetX Control and Triggering

In order for the DC to collect event data and trigger reliably, a couple settings must be configured. First, the photodiode biases must be set. A High voltage module can quickly, but roughly tune the biases. In addition, there are trim DACs that more finely tune the biases of the diodes. These 12-bit DACs are programmed by the mppc_DAC module and the DC Registers 15 and 16. This module was designed by Khnah Le in the original HMB/Hodoscope FW. The implementation of this module was not included Figure 5, as it was just recently added, and no testing has been conducted on it. The other setting that must be configured is the comparator threshold on the TargetX. The TargetX has 16 comparators that monitor the 16 photodiode channels on the DC [4]. When one or more of the channel voltages exceed their thresholds, the TargetX will output an encoded 5-bit trigger signal that indicates which channels were triggered. The thresholds are set by the TargetX DACs, which are programmed by the TARGETX DAC Controller. The Controller interacts with the TargetX through a serial protocol described in [4].

The 5-bit trigger output from the TargetX (TX_trig) is monitored by the Trigger Logic unit. When TX_trig takes on any value besides zero, the unit will pack the value along with a timestamp from the Timestamper module into a 32-bit word that will be sent to the SCROD over QBLink. This word is known as TriggerData. Bits 7 through 5 are set to “111”, which is the trigger event marker.

3. Simulation/Testing

The project contains testbenches that simulate the FW. Each testbench file contains documentation of what modules are being simulated as well as the functionalities that have been simulated successfully. The testbenches can be accessed in the appropriate project directory (see section 5.)

After the FW simulates successfully, it was hardware tested. The following functionalities have been successfully hardware tested:

1. PC to SCROD communication
 - a. Fiber optic cable link established.
 - b. Able to read and write to registers without errors
2. PC to DC communication
 - a. QBLink modules pass training state after reset
 - b. Able to read and write registers on DC without errors
3. Simultaneous reset of QBLink modules on SCROD and DC
4. SCROD and DC ping commands.
 - a. When target is DC, get error packet when QBLink training phase is not passed (trg_link_synced = '0').
 - b. Successful ping packet when QBLink passes training phase or when PC and SCROD are connected.

4. Future Work

The next steps are to complete the trigger readout hardware test. The whole system has been simulated. Given that all DCs trigger, each DC can successfully construct the Trigger Data word, send it to the SCROD, trigger a global event and read out all Trigger Data words to the Ethernet Module. In order to test this in hardware, we need to force trigger all DCs. This can be done by replacing a diode with a pulse from a function generator, but first we must correctly configure all the DCs as discussed in section 2.2.2. The values of the diode biases, the addressing scheme of the diode bias DACs, and the optimal values of DC registers 2 through 4.

Once the trigger readout hardware test is successful, the Hodoscope FW is nearly complete. Some functionalities that were in the old FW need to be added, such as the DAC threshold scan, and software triggering [6]. I recommend comparing the new FW with the old and see how much of the code can be recycled. A few of the processes are temporary stand ins for more complete FW. These include the DAC update process and the MPPC (diode) DAC update process. These processes functionality may be sufficiently, but I will leave that up to future users. The last Hodoscope function that needs to be added is sending SiREAD readout triggers to the other SiREAD SCRODs.

To complete the HMB FW, waveform data readout functionality needs to be added in addition to finalizing the trigger readout. This functionality never worked properly in the original FW, but it is still worth checking out the old project's approach.

5. Using the Project

Below is an outline of steps to acquire the project and setup the git repository. This document assumes some familiarity with Git. If there are any questions or issues, please feel free to contact me.

1. Fork all repositories, including submodules
 - a. mRICH-FW: <https://github.com/nathankp/mRICH-FW>
 - b. HMB-FW: <https://github.com/nathankp/HMB-FW>
 - c. Ethernet and Command Interpreter Submodule:
https://github.com/nathankp/SCROD_ETHERNET_V3
 - i. Submodule: <https://github.com/nathankp/firmware-ethernet>
 - ii. Submodule: <https://github.com/nathankp/firmware-general>
 - d. QBLink submodule: <https://github.com/nathankp/QBLink>
2. Clone the main repository into your desktop
3. Run to initialize the submodules: `git submodule update --init --recursive`
 - a. Ensure that all submodules in the main repository are properly referenced in `.gitmodules`
 - b. Follow README instructions in `SCROD_Ethernet_V3` to ensure proper setup of the `SCROD_Ethernet_V3` project
 - c. Add your forked copy of the repository to this list of remotes. I would suggest pushing all changes to your fork, otherwise you will have to submit a pull request to the original author.

A copy of this document and all figures therein can be found in EIC-Beamtest-FW/Documentation and EIC-Beamtest-FW/Documentation/ Current_diagrams, respectively. To conduct hardware test, you can construct and send packets with UDP_Client_NP.py in EIC-Beamtest-FW/scripts. To listen for trigger packets, a script must be written that receives data from the Gigabit Transceiver continuously in a while loop. This script was written on the mRICH bench computer at the IDLab, and should be included in repository soon.

The latest SCROD FW can be found in EIC-Beamtest-FW/SCROD_A5_RJ45/SCROD_Rev1. The older version is EIC-Beamtest-FW/SCROD_A5_RJ45/ SCROD_A5_RJ45. The latest DC FW can be found in EIC-Beamtest-FW/mRICH_hodo_DC_V1/HMB_QBLink_proto. The older version of the DC FW can be found in EIC-Beamtest-FW/mRICH_hodo_DC_V/ EIC-Beamtest-FW/mRICH_hodo_DC_V.

References:

[1] N. Park, “Hawaii Muon Beamline Scintillating Tracker Plane Data Acquisition System Technical Report 2018-2019.”

[2] SCROD schematic, ID # IDL_15_002, available at:

https://www.phys.hawaii.edu/~idlab/taskAndSchedule/PCBs/PCBs_homepage.html

[3] RJ-45 Board schematic, ID # IDL_18_020, available at:

https://www.phys.hawaii.edu/~idlab/taskAndSchedule/PCBs/PCBs_homepage.html

[4] G. Varner, TargetX Datasheet, available in: Documentation folder of repository

[5] HMB DC schematic, ID# IDL_18_014, available at

https://www.phys.hawaii.edu/~idlab/taskAndSchedule/PCBs/PCBs_homepage.html

[6] T. Lam, “mRICH documentation”, available in: Documentation folder of repository

Appendix

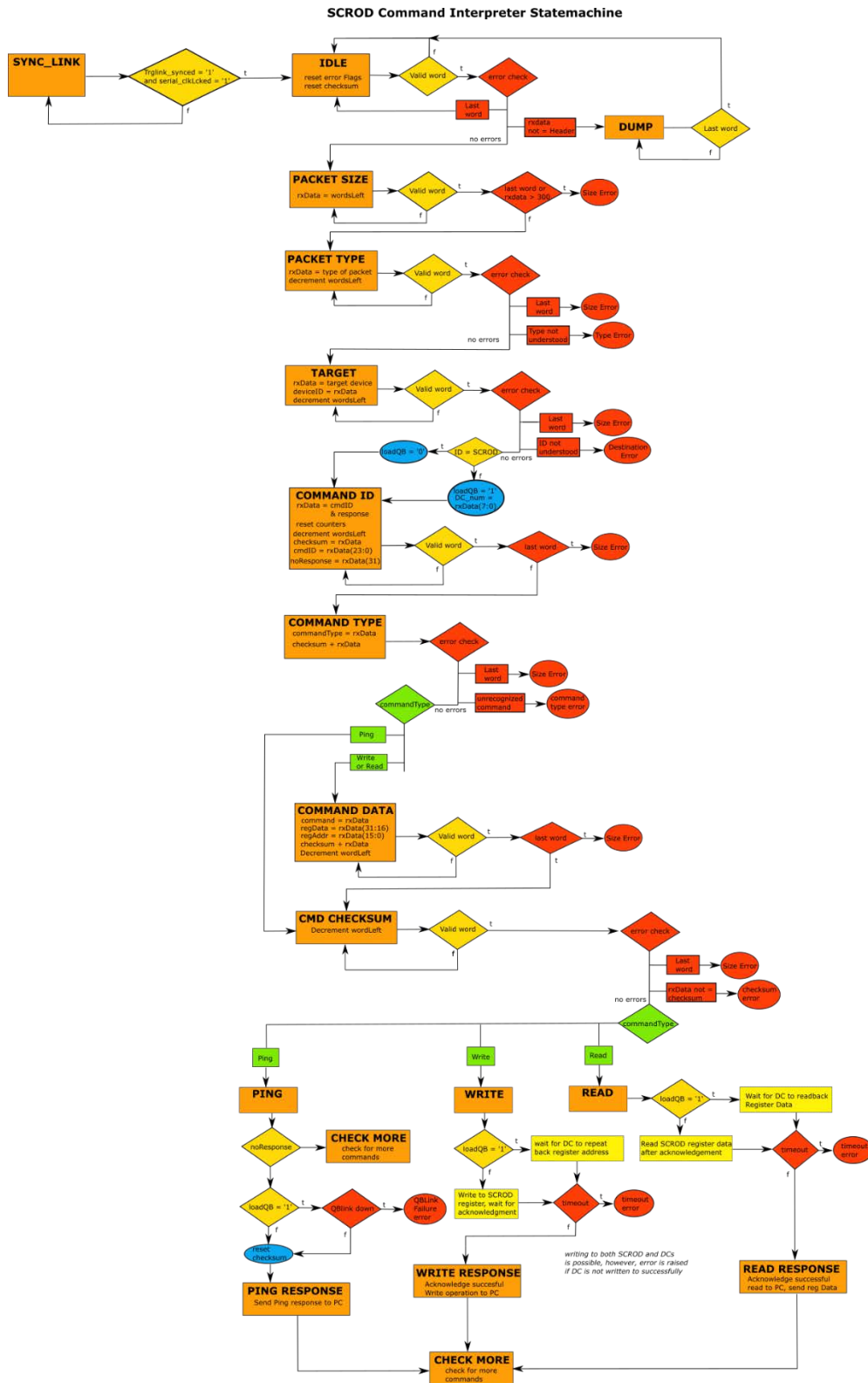


Figure 1.