Version 1

Description and Manual for the MoPSv1

Rizwan Ahmad

June 9, 2020

Abstract

This document serves as the reference manual for the MoPSv1 chip. MoPSv1 chip is the first prototype submitted with all required components to meet the requirements of the DCS system. TSMC 65nm CMOS technology is used to build the chip. Analog parts of this chip includes On-chip linear voltage regulator, Controller Area Network (CAN) physical layer implementation which includes the driver and the receiver, a power-on-reset circuit and an oscillator. Digital part of this chip includes the CAN controller which is called /"Canakari" and a bridge controller which implements a specific part of the CANopen standard required for our DCS system. In addition, it also includes high voltage regulators and CAN physical layer to test the functionality and radiation effects for this IC process at high radiation levels. This chip has an ADC which is silicon proven and is used in the FrontEnd(FE) chip. This ADC has 40 channels with a resolution of 12-bits.

Acknowledgements

Development of the MoPSv1 chip is carried out by the joint collaboration of the University of Wuppertal and FH Dortmund. I would like to say thanks to all the people involved from both institutes. Michael Karagounis, Tobias Fröse, ALexander walsemann, Aaron beer, Philip ledüc participated from FH Dortmund. Rizwan Ahmad, Susanne Kersten, Peter Kind,Christian Zeitnitz and Ahmed Qamesh were involved from the University of Wuppertal. I also would like to thank Niklaus Lehmann for his valuable suggestions.

Contents

Todo list

1 Introduction

1.1 Overview of the DCS system

ATLAS Pixel Detector is a general purpose detector and one of the machines installed at Large Hadron Collider in CERN,Geneva/Switzerland. An upgrade is planned for the ATLAS Pixel Detector to completely replace its Inner Tracker (ITk) with the new system. A Detector Control System (DCS) has been proposed to prevent damage caused by overvoltage and overheating to the inner modules. It will also help in commissioning. Detailed information about the proposed DCS system is given on this webpage DCS twiki page. DCS system consists of three separate paths which are, Diagnostics, Control and Safety. Control path is defined to monitor and control the individual modules of the Pixel Detector through a user interface which should be available in the main Detector Control Station. Figure 1 shows the DCS Control path where each Pixel module is connected to the MoPS Chip which measures the voltage and temperature. This MoPS chip is also being developed at the University of Wuppertal. One MoPS chip can monitor the temperature and voltage of 16 FE modules in a serial power chain. The MoPS chain then communicates to the main DCS computer over the CAN bus.

1.2 Objective

The objective is to build a radiation hard chip which is able to communicate using CAN and CANopen communication protocol. It will monitor the voltages and temperature across individual modules in the Serial Power (SP) chain and send this data to the main DCS computer. The overall intention is that the MoPS chip shall be as simple as pos-

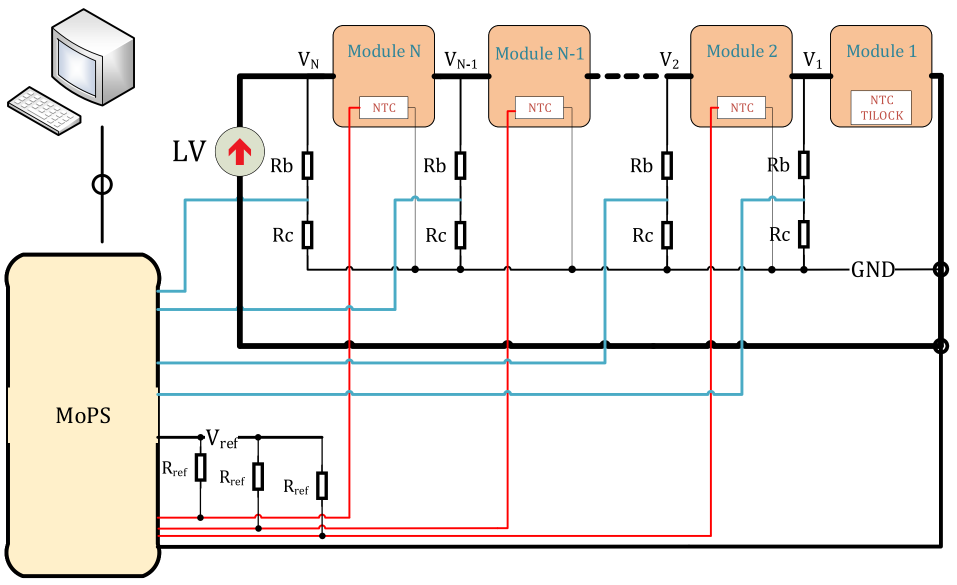


Figure 1: Detector control system control path

sible. It should only send messages or start communicating with the hardware when it has received a request from the control station. Only the most simple protocols with the least overhead have been chosen in order to make the implementation in hardware not too complicated.

The CANopen object dictionary of the MoPS chip reflects this behaviour as is does not allow much configuration and only implements the most important entries that are needed for the intended functionality.

2 MoPSv1 chip

2.1 How to use the chip

This section describes how to use the MOPSv1 chip to read monitoring values without going into any details about the chip. To read the ADC channels of the MOPSv1 chip, things which are required listed below:

- MOPSv1 chip
- CAN level shifter (Provided by Wuppertal DCS group)
- MOPS readout software
- Commercial CAN controller (Anagate / Kvaser (recommended))

CAN level shifter This is a custom made small circuit which translates and isolates the CAN bus signals between the MOPS chip and a standard CAN physical layer. It has D-SUB9 connector on both sides. One end must be connected to the commercial CAN controller and the other end to the CAN bus which is used to communicate with the MOPS chip. This circuit is already developed and provided by the Wuppertal DCS group. The reason to use this circuit is to isolate and translate signals between two CAN physical layers. A standard CAN physical layer works at 3.3 or 5 V while the MOPS physical layer only works at 1.2 V.

MOPS readout software This is a small GUI developed in python by the Wuppertal DCS group to easily readout the MOPS chip. The user does not need specialized knowledge to work with this software. It has the interface to read different ADC channels of the MOPS chip. A GUI button will automatically generate the required CAN message for a specific operation. To figure out which package pin is connected to which ADC channel, Figure 6 can be used. Please keep in mind that the temperature and voltage monitoring lines shown in figure 6 are only for illustration purpose. It does not reflect the actual configuration of each subsystem. The user of the chip has to find out which monitoring value is connected to each ADC channel. A commercial software which is provided with the commercial CAN controllers can also be used to communicate with the MOPS chip but it is not recommended because then the user has to define by himself all details of the CAN message. If it is still required to use a commercial software then further details about the chip are provided in the next sections of this manual.
MOPS readout software can be downloaded from this repository MOPS readout software and the documentation on how to install and use the software can be downloaded from readout software documentation.

Commercial CAN controller Two commercial CAN controllers tested to work with the MOPS readout software are:

- AnaGate

- Kvaser

It is recommended to use Kvaser controller which is cheap and easy to configure if the commercial interface is used.

2.2 Overview

MoPSv1 is the first prototype with all components included which are required by the chip for complete functionality. This chip has analog and digital components designed with the TSMC 65nm CMOS technology to check for desired functionality and radiation studies. Below mentioned are the integrated components of this chip.

- 1.2 V Voltage Regulator
- CAN physical layer
- Oscillator
- Power-on-reset circuit
- Digital logic
  - CAN controller
  - Bridge controller
- High voltage CAN physical layer and regulators for radiation studies

2.2.1 Block Diagram

Figure 2 shows the block diagram for the MoPSv1 chip. It includes regulator, CAN transceiver, MoPS controller, ADC, POR and the oscillator. The chip also includes high voltage CAN physical layer and the voltage regulators but that is not relevant for the users. Those extra structures are there only for radiation studies and will not be included in the final version of the chip.

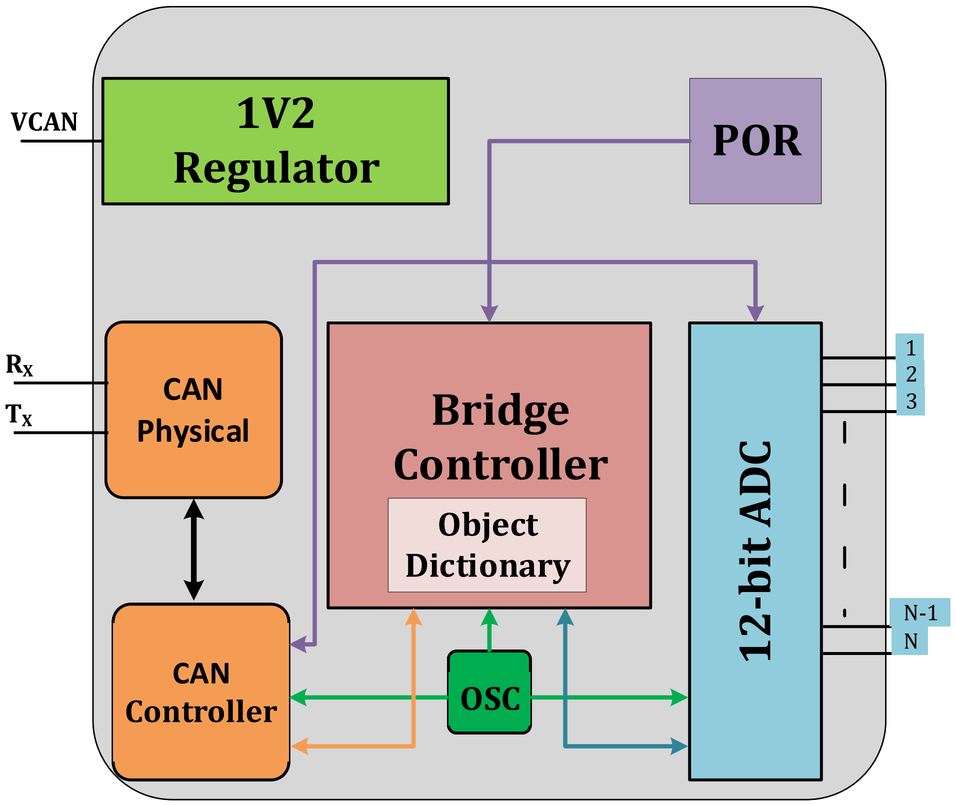


Figure 2: MoPSv1 Block diagram

2.2.2 Chip Layout

Figure 3 show the final layout of the chip. Block diagram is given in the Figure 2. The internal padring is for nonessential structures which are fabricated for radiation studies. They are not relevant for the users. The internal padring will removed in the final version of the chip and it will not effect the functionality of the chip.

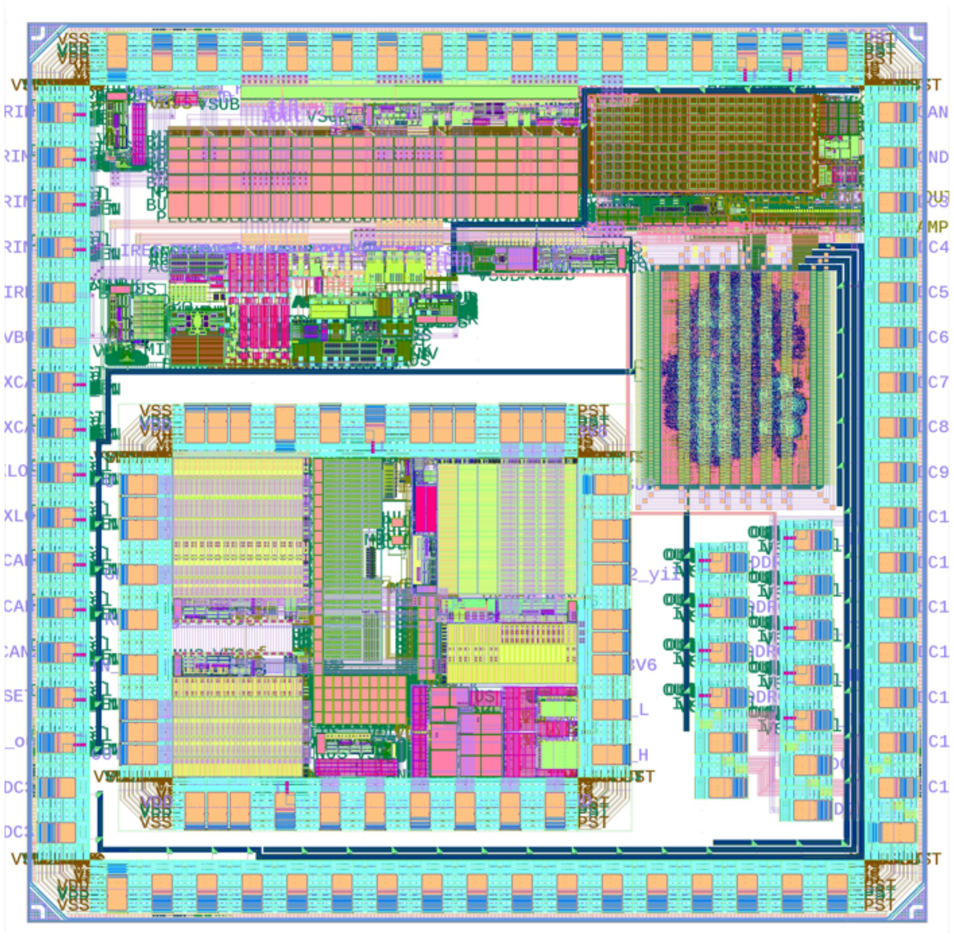


Figure 3: MoPSv1 Layout

2.2.3 Port Layout

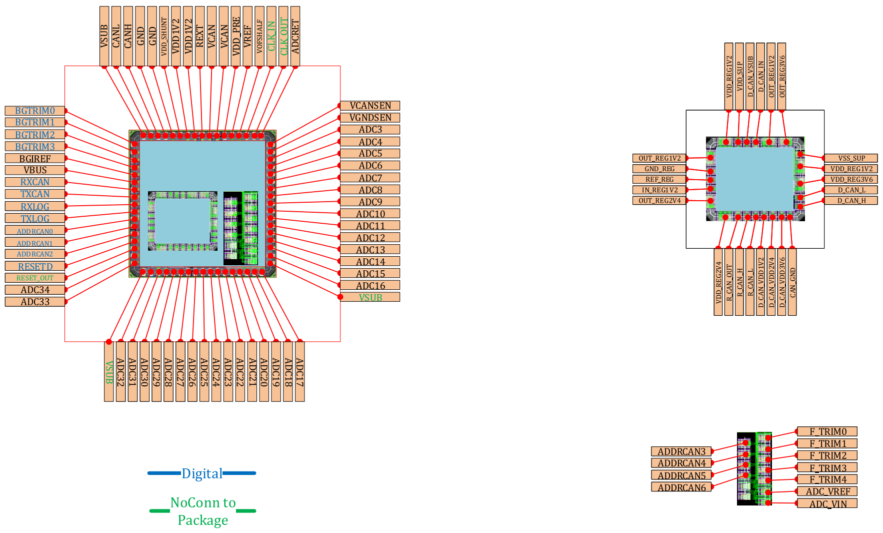


Figure 4: MoPSv1 Port Names

Shown in the Figure 4 are the names of all the pads. There are exactly 17 pads on each side on the outer ring but only 16 pads will be connected to the package pins. The exact connections which are intended for the package are shown in figure 6. In addition there is an internal ring which is used to make connections for the high voltage CAN physical layers and the regulators. These circuits are fabricated to make use of the available space and do the radiation studies for the high voltage circuits. All the pads on the internal ring are also not connected to any package pin in the final chip and are not relevant for the users of the chip. Two small figures on the right side of figure 4 show all the internal pads. Some of these pads are used for debugging purpose e.g F_TRIMx pads are used to trim the oscillator. If not explicitly stated otherwise, all the debug pads will be taken care of inside the package.

CAN Tranceiver CAN transceiver includes both CAN driver and CAN receiver. Two signals CAN high and CAN low to drive and receive messages from the CANbus . The digital signal going in and coming out of the physical layer are routed externally so that the physical layer and digital part of the CAN node can be tested and used independently. e.g on PP3 location some of the chips are only used for the physical layer while others are used solely for the digital logic.

2.2.4 Physical Dimensions

The original design of the chip is 2x2mm. Fabricated size of the chip is still unknown. Pads for this chip are 50μm wide with a pitch of 100μm.
There are at-least 17 pads on each side of the chip. Pads are separated from the corner cells by 34μm. Chip dimensions are shown in the figure

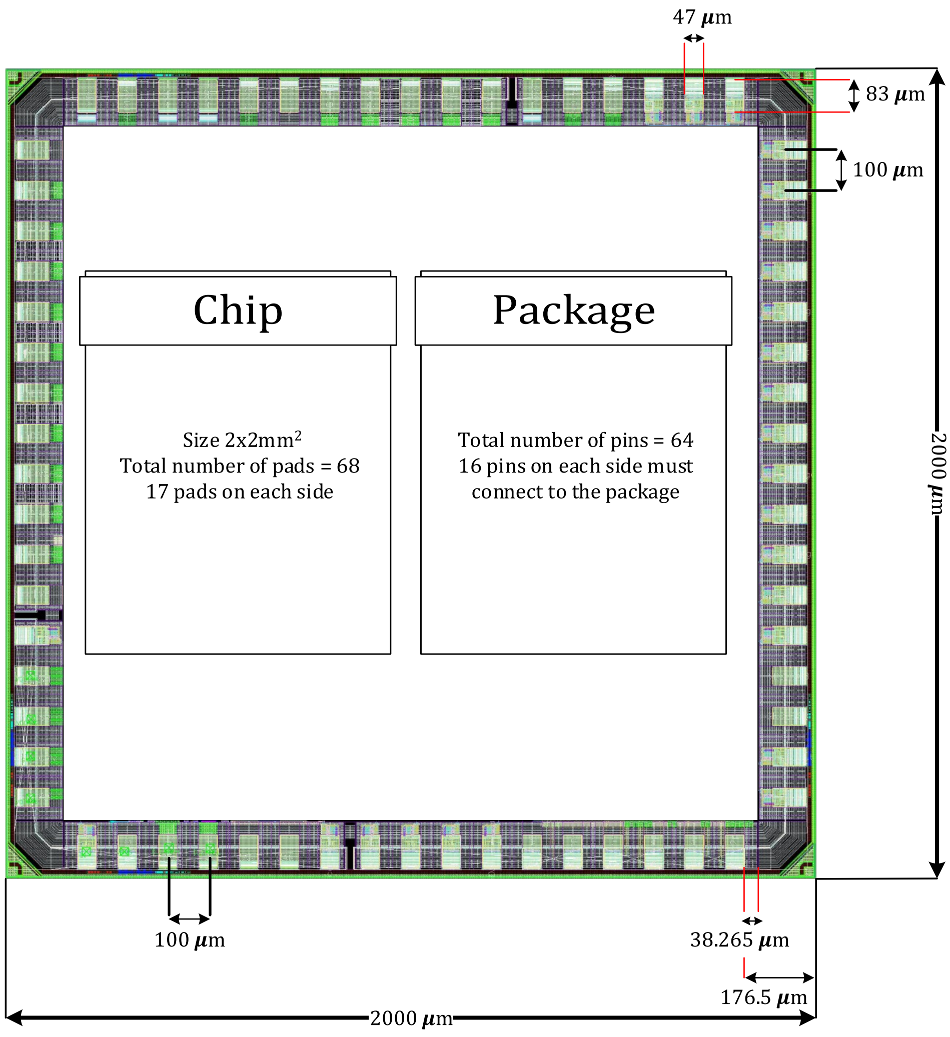


Figure 5: MoPSv1 dimensions

2.2.5 Packaging

The QFN64 9x9 mm^2 package is used to package the chips. Detailed datasheet can be found on this Link Europractice QFN64 datasheet.

2.2.6 Required external components for proper functioning

For proper functioning of the chip, some external components are required for proper configuration or trimming.
Figure 6 lists all the required external components and their position.

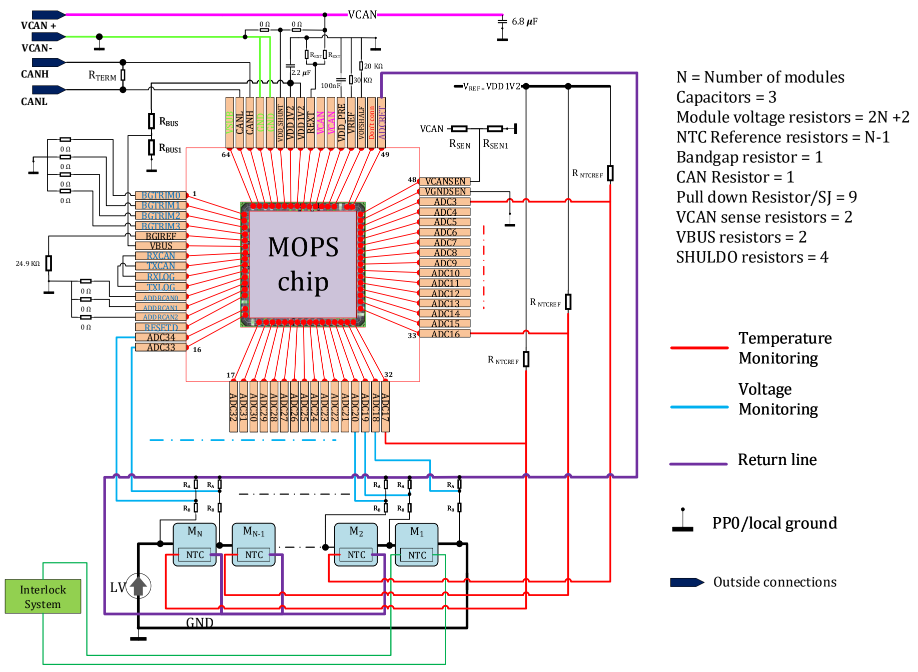


Figure 6: External components required by the chip

Below mentioned is the description about some of the pads and components:

VDD SHUNT This is used to

- enable => connection to VCAN
- disable => connection to ground

shunt-mode of the SLDO amplifier.

REXT The resistor which defines the input impedance of the SLDO in shunt-mode and is connected to VCAN when shunt-mode of the SLDO is enabled.

VDDPRE output of the preregulator and requires a connection to a 100nF blocking capacitor against ground.

VREF Voltage reference for the definition of the regualtor output and the ADC LSB voltage and requires a 30 KOhm resistor connected to ground.

VOFSHALF Offset voltage reference which is used when the SLDO is configured in shunt-mode and requires a 20 KOhm resistor connected to ground.

BGTRIMx These pads are used to trim the bandgap circuit. MSB which is BGTRIM3 is pulled-up internally while rest of the pins are pulled-down internally.

2.3 Power and IO Signal Specifications

In the following section IO and power signals are described.

Table 1 provides description of all the IO signals used in the chip shown in the Figure 4. Please see the footnotes of the table for hints. Further details about different pads are given in table 4 and 3.

Table 1: All pads in the outer ring of the chip

Signal name	Direction	Signal type	Pad type	Description
BGTRIM0	IN	digital	CERN_IO_mod_VSUB_PAD	Trimming bit for the bandgap circuit
BGTRIM1	IN	digital	CERN_IO_mod_VSUB_PAD	Trimming bit for the bandgap circuit
BGTRIM2	IN	digital	CERN_IO_mod_VSUB_PAD	Trimming bit for the bandgap circuit
BGTRIM3	IN	digital	CERN_IO_mod_VSUB_PAD	Trimming bit for the bandgap circuit
BGIREF	-	analog	SF_1V2_FULL_LOCAL	Reference current for the bandgap circuit
VBUS	IN	analog	SF_1V2_FULL_LOCAL	CAN bus voltage in recessive state

Table 1 continued from previous page

Signal name	Direction	Signal type	Pad type	Description
RXCAN	OUT	digital	CERN_IO_ mod_VSUB _PAD	Received bit from CAN Phy. layer
TXCAN	IN	digital	CERN_IO_ mod_VSUB _PAD	Bit to transmit using CAN Phy. layer
RXLOG	IN	digital	CERN_IO_ mod_VSUB _PAD	Received bit to the logic
TXLOG	OUT	digital	CERN_IO_ mod_VSUB _PAD	Bit from the logic to be transmitted
ADDRCAN0	IN	digital	CERN_IO_ mod_VSUB _PAD	CAN node address
ADDRCAN1	IN	digital	CERN_IO_ mod_VSUB _PAD	CAN node address
ADDRCAN2	IN	digital	CERN_IO_ mod_VSUB _PAD	CAN node address
RESETD	IN	digital	CERN_IO_ mod_VSUB _PAD	External reset to reset the digital part
Reset_out	OUT	digital	CERN_IO_ mod_VSUB _PAD	Output of the POR circuit
ADC34	IN	analog	SF_1V2_CDM	ADC channel input pad
ADC33	IN	analog	SF_1V2_CDM	ADC channel input pad
* VSUB	-	-	-	-
ADC32	IN	analog	SF_1V2_CDM	ADC channel
ADC31	IN	analog	SF_1V2_CDM	ADC channel
ADC30	IN	analog	SF_1V2_CDM	ADC channel

Table 1 continued from previous page

Signal name	Direction	Signal type	Pad type	Description
ADC29	IN	analog	SF_1V2_CDM	ADC channel
ADC28	IN	analog	SF_1V2_CDM	ADC channel
ADC27	IN	analog	SF_1V2_CDM	ADC channel
ADC26	IN	analog	SF_1V2_CDM	ADC channel
ADC25	IN	analog	SF_1V2_CDM	ADC channel
ADC24	IN	analog	SF_1V2_CDM	ADC channel
ADC23	IN	analog	SF_1V2_CDM	ADC channel
ADC22	IN	analog	SF_1V2_CDM	ADC channel
ADC21	IN	analog	SF_1V2_CDM	ADC channel
ADC20	IN	analog	SF_1V2_CDM	ADC channel
ADC19	IN	analog	SF_1V2_CDM	ADC channel
ADC18	IN	analog	SF_1V2_CDM	ADC channel
ADC17	IN	analog	SF_1V2_CDM	ADC channel
* VSUB	-	-	-	-
ADC16	IN	analog	SF_1V2_CDM	ADC channel
ADC15	IN	analog	SF_1V2_CDM	ADC channel
ADC14	IN	analog	SF_1V2_CDM	ADC channel
ADC13	IN	analog	SF_1V2_CDM	ADC channel
ADC12	IN	analog	SF_1V2_CDM	ADC channel
ADC11	IN	analog	SF_1V2_CDM	ADC channel
ADC10	IN	analog	SF_1V2_CDM	ADC channel
ADC9	IN	analog	SF_1V2_CDM	ADC channel
ADC8	IN	analog	SF_1V2_CDM	ADC channel
ADC7	IN	analog	SF_1V2_CDM	ADC channel
ADC6	IN	analog	SF_1V2_CDM	ADC channel
ADC5	IN	analog	SF_1V2_CDM	ADC channel
ADC4	IN	analog	SF_1V2_CDM	ADC channel
ADC3	IN	analog	SF_1V2_CDM	ADC channel
GNDSEN	IN	analog	SF_1V2_CDM	ADC channel
VCANSEN	IN	analog	SF_1V2_CDM	ADC channel

Table 1 continued from previous page

Signal name	Direction	Signal type	Pad type	Description
ADCRET	IN	analog	SF_1V2_FULL _LOCAL	Return line for the ADC
Clk_out	OUT	digital	CERN_IO_mod _VSUB_PAD	Clock output from the internal oscillator
Clk_in	IN	digital	CERN_IO_mod _VSUB_PAD	Input clock to the digital part
Vofshalf	IN	analog	SF_1V2_OVT _NOB2B	CAN transceiver local ground
Vref	IN	analog	SF_1V2_OVT _NOB2B	Reference voltage for the regulator and the ADC LSB voltage
VDD_PRE	power	analog	SF_1V2_OVT _NOB2B	Output of the preregulator
* VCAN	-	-	-	-
VCAN	power	analog	SF_1V2_OVT _NOB2B	Supply voltage for the chip
Rext	-	analog	SF_1V2_OVT _NOB2B	Config resistor for shuldo
* VDD1V2	-	-	SF_1V2_POWER _CLAMP_CORE _SUPPLY	Output of the regulator
VDD1V2	power/out	analog	SF_1V2_POWER _CLAMP_IO _SUPPLY	Output of the regulator
VDD_SHUNT	power/in	analog	SF_1V2_OVT _NOB2B	enable/disable shunt mode by conn. to VCAN/GND
* GND	-	-	-	-
GND	gnd	analog	SF_1V2_ POWER _CLAMP_IO _GROUND	Ground conn. for the chip
CANH	in/out	analog	SF_1V2_CDM	CAN bus high sig.
CANL	in/out	analog	SF_1V2_CDM	CAN bus low sig.

Table 1 continued from previous page

Signal name	**Direction**	**Signal type**	**Pad type**	**Description**
VSUB	-	gnd	SF_1V2_ POWER _CLAMP_CORE _GROUND	Substrate conn.
* = Internally connected to the pin with similar name				

2.3.1 Maximum Ratings

This section specifies the maximum ratings for inputs.

- All the inputs with the pads which are connected to 1V2 ESD ring must not be supplied more than $\pm 1.3V$. If the input goes above that value than the ESD protection circuit gets activated.
- Inputs using SF_1V2_OVT_NOB2B pad must not be supplied more than 2V in general but must look at individual values given in the table 2:
- To check all the ESD protection ratings of different pads used in the chip, tables 4 and 3 must be consulted.

Table 2: Maximum ratings for different input pads

Name	**Min**	**Typ**	**Max**	**Unit**
VDD_SHUNT	1.2	1.6	2.0	V
VCAN	1.4	1.6	2.0	V

2.3.2 Power pad specifications

Table 3 lists all the power pads used in this chip.

Table 3: This table lists all the pads used for powering of the chip.

Name	**ESD connection**	**Description**
SF_1V2_POWER_CLAMP_IO _SUPPLY	1V2	SF_1V2_POWER_CLAMP _CORE_SUPPLY (Shorted)
SF_1V2_POWER_CLAMP_IO _GROUND	1V2	Supply ground

Table 3 continued from previous page

Name	ESD connection	Description
SF_1V2_POWER_CLAMP _CORE_GROUND	1V2	Substrate connection
SF_1V2_OVT _NOB2B	2V0	Supply voltage

2.3.3 IO pad Specifications

Table 4 lists all the IO pads used in the chip.

Table 4: This table lists all the pads used for IO connections

Name	Type	ESD connection
CERN_IO_mod_VSUB_PAD	digital	1V2 ESD ring
SF_1V2_CDM	analog	1V2 ESD ring
SF_1V2_OVT_NOB2B	analog	OVT*
SF_1V2_FULL _LOCAL	analog	1V2 ESD ring

* Over voltage tolerant i.e can withstand upto 5V.

2.4 Functional Description of the Components

2.4.1 Voltage Regulator

Voltage regulator used in the MoPSv1 chip is the same shunt ldo (shuldo) regulator which is used in the Front End (FE) chip of the ATLAS ITk Pixel Detector.

2.4.2 CAN Physical Layer

A non standard 1.2 V CAN physical layer is developed for the chip to communicate over the CAN bus. The reason to develop non standard low voltage CAN physical layer is the limitation because of TSMC65 nm process and high radiation induced damages due to high voltage levels.

2.5 Implemented communication protocol

The chip communicates using standard CAN protocol but implementation at the application layer using CANopen standard is very limited and specific to DCS monitoring path requirements.
Generally it is to be noted that in CANopen data are transmitted according to the “Little Endian” rule and therefore according to the form for corresponding INTEL processors.

This means that the low value byte is transmitted first. This makes it slightly more difficult for a human to follow a monitored SDO 11 protocol sequence but in the end this is a matter of habituation.

2.5.1 Physical layer for communication on the bus

The chip communicates over the CAN bus using standard CAN bus communication protocol but the electrical signals on the bus are not standard compliant. MoPS communicates using non-standard CAN high and CAN low signals. A level shifter is required to translate and isolate signals from a standard CAN physical layer to MoPS physical layer.

- Recessive state $>$ logic 1 $> CAN_H = CAN_L = 600mV$
- Dominant state $>$ logic 0 $> CAN_H = 1.2V$ and $CAN_L = 0V$

2.5.2 Controller Area Network (CAN) protocol overview

CAN is a serial bus communication protocol where all the nodes are connected to a single bus which is terminated by a termination resistor at both ends. CAN protocol is defined only for the first and second layer in the Open systems interconnection (OSI) model which is physical and data link layer respectively.
Some features of the protocol are:

- Multi master priority based network
- Non destructive message arbitration
- Data is broadcasted to every node on the network
- Robust error detection mechanism
- Differential signaling to provide noise immunity
- Data can be requested remotely by other nodes on the bus
- Automatic re-transmission

2.5.3 Types of CAN message

- Data frame
- Remote frame
- Error frame
- Overflow frame

As the CAN controller in this chip is is compliant to standard 11898 so it implements all four type of CAN messages although we do not use remote messages for communication for the sake of keeping things simple.

2.5.4 Sengments of a CAN message

- Identifier / Arbitration field
- Control bits
- Datal field
- CRC / error correction field

To use CAN protocol a set of rules must be defined at the application layer to efficiently handle the messages on the bus and provides interface to the user. CANopen standard works at the application layer. CANopen makes use of all the features of the CAN protocol but puts limit to the maximum number of nodes which can be identified on a single network. A maximum of 128 nodes can be connected if one uses CANopen at the application layer.
Below mentioned are the important points about using the CANopen standard:

- An ID is assigned to every node on the bus
- Devices communicate to each other using predefined set of objects called communication objects
- There is a central library / stored values in each node called object dictionary. This object dictionary holds all the values for communication on the bus. This includes configuration values and predefined set of data messages.
- Communication objects have different priorities.
- The highest priority object will get the chance to access bus.

2.5.5 CANopen implementation at application layer

For our application, we have implemented two different types of communication objects in the first version of the chip.

- Network Management Object (NMT) (2.5.7)
- Service Data Object (SDO) (2.5.8)

Even for the above mentioned objects, the chip does not implement the full blown version because of the space and complexity constraints. CANopen is usually implemented in a software so only the very important things which are necessary for our application are implemented in this chip.
A CANopen object dictionary is defined. Object dictionary defined in the MOPSv1 chip is also very small and very specific to the DCS monitoring system to keep things simple.

2.5.6 Object Dictionary (OD)

Table 5 and 6 show communication and manufacturer specific profile area of the object dictionary respectively. All values are hexadecimal if not otherwise stated. The information in the object dictionary is required to communicate with the MOPS chip. The placement of index/subindex and read/write commands inside the CAN message should be done as explained in 2.5.7 and 2.5.8.

Table 5: Communication profile area of OD

Index	SI	Description	Data/Object	Attr	Default	Comment
1000		Device type	U32	RO	191	Meaning: DSP-401 device profile, analogue in- and outputs, digital in- and outputs on device. Mandatory object
1001		Error register	U8	RO	0	Mandatory
1005		COB-ID SYNC	U32	RO	80	
1014		COB-ID EMCY	U32	RO	80 + NodeId	
1018		Identity object	RECORD			Mandatory CANopen object
	0	Number of entries	U8	RO	1	
	1	Vendor Id	U32	RO	12345678	Ambiguous since we will never receive an official vendor Id
1200		Server SDO parameter 0	RECORD			Define standard COB-ID for SDO
	0	Number of entries	U8	RO	2	
	1	COB-ID client to server	U32	RO	600 + NodeId	
	2	COB-ID server to client	U32	RO	580 + NodeId	
1800		TPDO communication parameter 0	RECORD			May be used for confirming a bypass command
	0	Number of entries	U8	RO	6	
	1	COB-ID	U32	RO	180 + NodeId	
	2	Transmission type	U8	RO	FE	Event driven - manufacturer specific
	3	Inhibit time	U16	RO	0	

Table 5 continued from previous page

Index	SI	Description	Data/Object	Attr	Default	Comment
	4	Reserved	U8	RO		Unused, manufacturer specific
	5	Event timer	U16	RO	0	
	6	SYNC start value	U8	RO	0	
1801		TPDO communication parameter 1	RECORD			
	0	Number of entries	U8	RO	6	
	1	COB-ID	U32	RO	280 + NodeId	
	2	Transmission type	U8	RO	FE	Event driven - manufacturer specific
	3	Inhibit time	U16	RO	0	
	4	Reserved	U8	RO		Unused, manufacturer specific
	5	Event timer	U16	RO	0	
	6	SYNC start value	U8	RO	0	
		TPDO mapping parameter 0	RECORD			
1A00	0	Number of entries	U8	RO	1	
	1	PDO mapping entry	U32	RO	21000020	“2100”: Index; “00”: Subindex; “20” : Data length (20=32 bit)
1A01		TPDO mapping parameter 1	RECORD			
	0	Number of entries	U8	RO	1	

Table 5 continued from previous page

Index	SI	Description	Data/Object	Attr	Default	Comment
	1	PDO mapping entry	U32	RO	21010030	“2101”: Index; “00”: Subindex; “30” : Data length (30=48 bit)

Table 6: Manufacturer specific profile area

Index	SI	Description	Data/Object	Attr	Default	Comment
2001		ADC trimming bits	U8	RW		This entry refers to a 6 bit register for ADC trimming which is to be set via SDO when the server starts up. The value should originate from a config file.
2100		Read monitoring values of all FEs	U32	RO		Only to be used with TPDO0. This should return some SDO abort code when polled with SDO.
2310		Other monitoring values	RECORD			This object contains monitoring values which are not connected to a single FE. All values come as raw ADC counts.
	0	Number of entries	U8	RO	3	
	1	VBANDGAP	U16	RO		Bandgap output voltage
	2	VCANSEN	U16	RO		CAN sense line voltage
	3	VGNDSEN	U16	RO		Ground sense line voltage
2400		Internal ADC channels	RECORD			This entry contains all ADC channels which do not have a defined functionality up to now.
	0	Number of entries	U8	RO	40	
	01-20	ADCCh3-ADCCh34	U16	RO		

2.5.7 Network Management Object (NMT)

For this type of communication object there are two things implemented in this chip:

- Remote reset request
- Node guarding

Remote reset request Whenever a chip is required to be reset remotely then an empty message with an ID '0' is sent over the CAN bus. CANopen standard defines this to be node dependent but to make this simpler for our application whenever an ID '0' is detected on the bus then every node will reset itself. After the reset, default configuration values e.g timing etc for the CAN node are reloaded and all the state machines will be brought to a known state.

Node guarding Node guarding is another NMT object which is implemented in this chip. This allows the chip to send one time sign-in message after power-up and also responds whenever the master node wants to check if the node is still alive and operational. Again, this protocol part is simplified to implement what is absolutely necessary. As all of our CAN nodes in the system are either operational or not operational so other status messages such as boot or pre-operational are not implemented.
Whenever there is a message on the bus with Identifier '0x700 + node-id' then the respective node will respond with a single byte of data where the MSB should toggle for every new request. If the master finds no response from the slave node or if the MSB of the response message does not toggle for every new request then there is something wrong about that node. Either it is not working or the configuration values in the registers have been changed. A reset request will then be required to bring back the node to operational state once again. All other status messages are not implemented.
Figure 7 show the format of Node guarding and sign-in message frame.

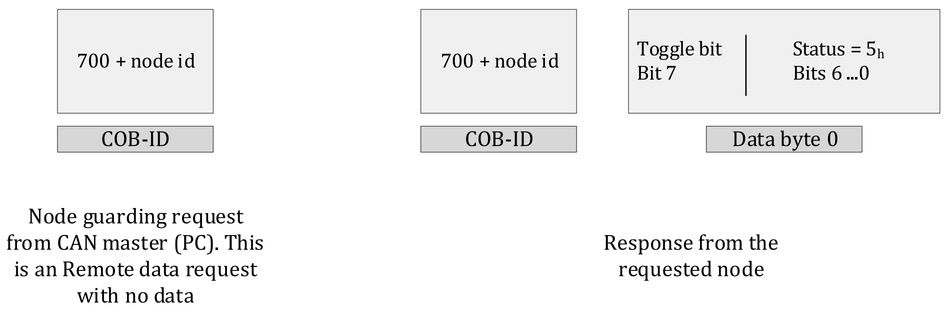

700 + node id
COB-ID
700 + node id
COB-ID
Toggle bit Bit 7
Status = 5h Bits 6 ...0
Data byte 0
Node guarding request from CAN master (PC). This is an Remote data request with no data
Response from the requested node

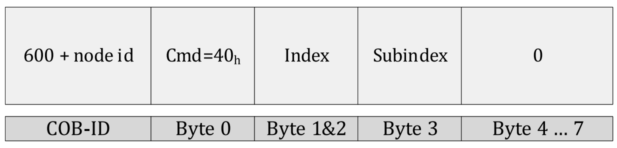
Figure 7: MOPS Node guarding/sign-in frame

Sign-in message [7] A single sign in message is generated by each node on the bus when the power is supplied. The structure of this sign-in message is same as node guarding.

As a matter of fact it is the same node guarding message but sent only once during the start-up by the remote node despite no request from the master node.

2.5.8 Service Data Objects (SDOs)

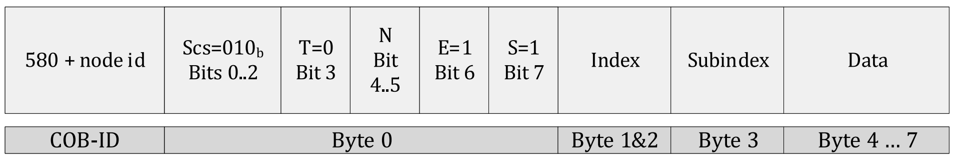
The SDO or Service Data Objects provide access to the object dictionaries in each device. They are particularly useful for configuration of devices as the SDO protocol is allowed in pre-operation mode. But it is also possible to get "process" data values by polling via SDO the appropriate object dictionary entry.
The SDO protocol always confirms the read/write operation. The chip implements service data objects of the CANopen standard but only expedited transfer is implemented. Object dictionary is defined in such a way that its entries do not exceed 4 bytes. All the entries of the object dictionary can be accessed using expedited data transfer of SDOs which allows maximum of 4 bytes of data to be sent in a single CAN message while the rest 4 bytes are used for other things i.e index, sub-index and SDO operation details. Different type of SDO frames which are used to communicate with the MOPS chip are shown in Figure 8 to 12.
Below mentioned is the description of the few terms used in the figures.
scs: Command specifier
n: n = 4 - Data size
t: Toggle bit (unused)
e: Expedited transfer is used
s: Data size is indicated
The table 7 shows the message encoded with each abort code. All values are in hexadecimal.



600 + node id	Cmd=40_h	Index	Subindex	0
COB-ID	Byte 0	Byte 1&2	Byte 3	Byte 4 ... 7

Read Dictionary request from the master to the node

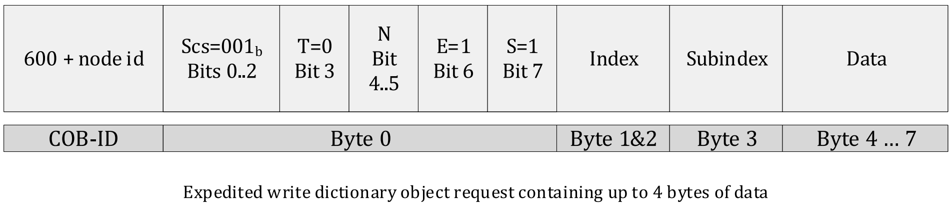
Figure 8: Read MOPS



580 + node id	Scs=010_b Bits 0..2	T=0 Bit 3	N Bit 4..5	E=1 Bit 6	S=1 Bit 7	Index	Subindex	Data
COB-ID	Byte 0					Byte 1&2	Byte 3	Byte 4 ... 7

Response from the node containing up to 4 Bytes of data

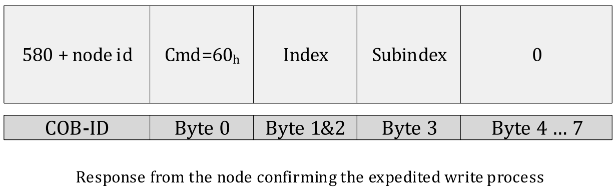
Figure 9: Response to read request



600 + node id	Scs=001$_b$ Bits 0..2	T=0 Bit 3	N Bit 4..5	E=1 Bit 6	S=1 Bit 7	Index	Subindex	Data
COB-ID	Byte 0					Byte 1&2	Byte 3	Byte 4 … 7

Expedited write dictionary object request containing up to 4 bytes of data

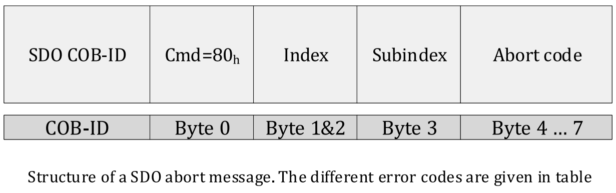
Figure 10: Write MOPS



580 + node id	Cmd=60$_h$	Index	Subindex	0
COB-ID	Byte 0	Byte 1&2	Byte 3	Byte 4 … 7

Response from the node confirming the expedited write process

Figure 11: Response to write request



SDO COB-ID	Cmd=80$_h$	Index	Subindex	Abort code
COB-ID	Byte 0	Byte 1&2	Byte 3	Byte 4 … 7

Structure of a SDO abort message. The different error codes are given in table

Figure 12: MOPS error

2.6 ADC channels

The ADC used in the MoPS chip is the same ADC which was developed for the frontEnd (FE) chip. This ADC has a resolution of 12-bits and the total number of channels are 40. Only 35 channels are utilized in this chip. 32 channels are for reading external analog inputs while 2 channels are defined to sense the supply voltage of the chip. 1 channel is internally connected to the output of bandgap circuit. Figure 4 defines the number and position of all the channels.

Table 7: SDO abort codes. Note that this is only an extract from almost 30 abort codes which represents all abort codes that may be implemented though we will probably not need all of them.

Abort code	**Description**
0504 0000	SDO protocol timed out.
0504 0001	Client/server command specifier not valid or unknown.
0601 0000	Unsupported access to an object.
0601 0001	Attempt to read a write only object.
0601 0002	Attempt to write a read only object.
0602 0000	Object does not exist in the object dictionary.
0606 0000	Access failed due to an hardware error.
0606 0007	Timeout of communication occurred.
0607 0010	Data type does not match, length of service parameter does not match
0607 0012	Data type does not match, length of service parameter too high
0607 0013	Data type does not match, length of service parameter too low
0609 0011	Sub-index does not exist.
0609 0030	Invalid value for parameter (download only).
0609 0031	Value of parameter written too high (download only).
0609 0032	Value of parameter written too low (download only).
0800 0000	General error

3 Test setup

Testsetup for the chip is planned in such a way that same set of boards can be used to run all the electrical and functional tests as well as they can also be used for the radiation facilities where the chip is very far from the control room. Same testsetup will be used for later versions of the chip as well with minor updates.
Figure 13 shows the planned testsetup.

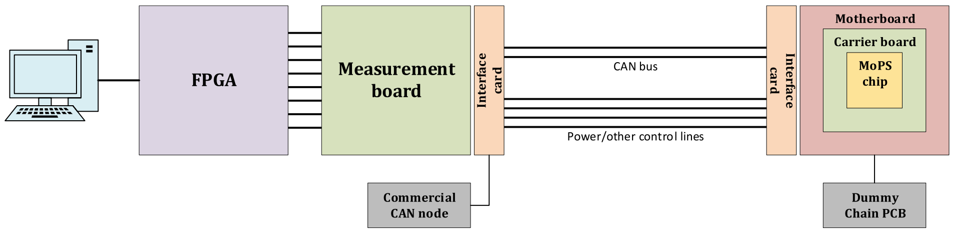


Figure 13: Planned testsetup

Below mentioned are some explanatory points about the testsetup:

- Measurement board includes some ADC which will read the same channels at any given time as the MoPS chip to provide comparison about performance of the chip. It will also include DAC controller power supply for the MOPS chip.
- Interface card provides interface such that we are able to use the CAN node on an FPGA with a seperate physical layer as well as any commercial CAN controller can also be used. For commercial CAN controller an adapter would be required to translate signals between standard CAN bus signals to MoPS specific levels.

The test setup shown in the figure 13 is only interesting for the people working the DCS group at the University of Wuppertal.

3.1 Test boards

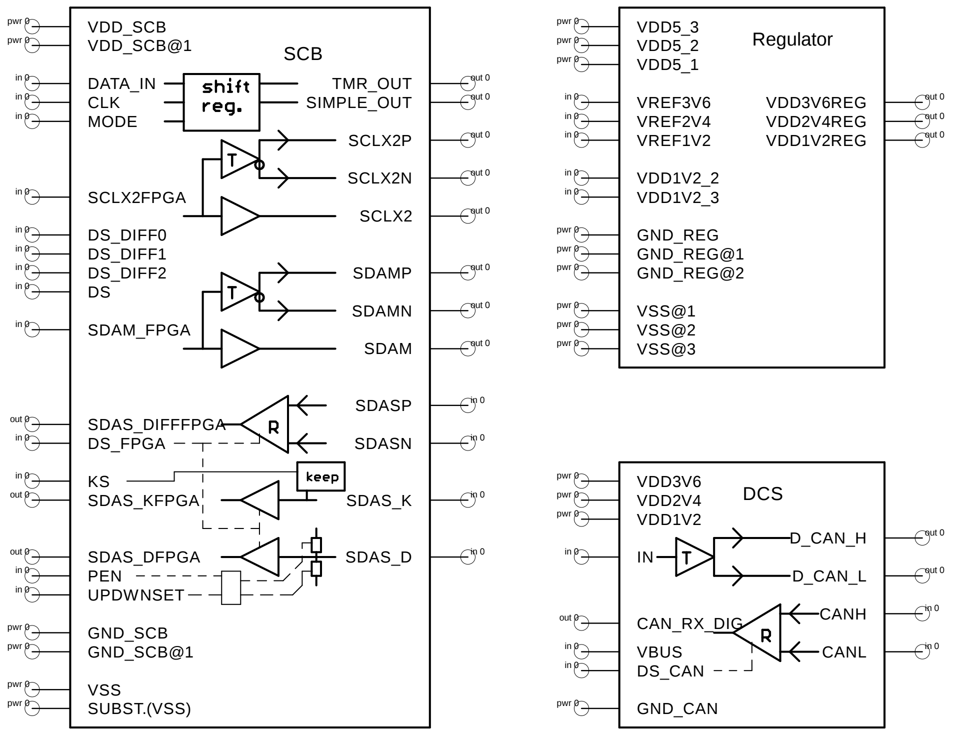


Figure 14: DCS controller chip v1 test board schematic

4 Known bugs and problems in the chip

This section describes all the known problems in the analog components of the chip and also the bugs in the digital part of the chip.

4.1 Problems in the analog components

4.1.1 Power-on-reset

Power-on reset is the circuit which generates a reset signal for the digital part and also starts the oscillator. When the power is supplied to the chip, this circuit applies a reset signal until voltage level of the power supply goes above a certain threshold. The circuit is built using NMOS transistors but PMOS transistors must be used in the later versions of the chip for better radiation performance.

4.2 Bugs in the digital part

4.2.1 Sign- in message

The sign-in message is the first CAN frame sent out by the MOPS chip on the CAN bus when the chip is powered on. The sign-in message is described in 2.5.7. The MSB of the the sign-in message should be 0 as shown in 7 and it toggles every time a new node guarding request is made. The MOPS chip sends out the sign in message with the correct MSB which is 0 and keeps sending this on the bus until it receives an acknowledgment. It is observed during the testing that if MOPS doesn't get an acknowledgment then after couple of sign-in messages on the bus the bit toggles automatically which should not be the case. This is not something which breaks the communication but it should be corrected in the next version of the chip for proper compliance to the specifications.

4.2.2 Reset request

There is a bug within the remote reset request functionality of the MOPS chip. When an ID '0' is sent over the bus then it should cause every MOPS chip connected on the bus to remotely reset itself. After working on the reset request the MOPS chip should respond back with a message which includes no data or ideally a node guarding response explained in 2.5.7 but what is observed during the communication is that the MOPS chip sends back the last message stored in the buffer register. In case the last message sent out by the MOPS chip was a node guarding message then the MSB toggles for every new reset request. In the simulations it was verified that the MOPS chip should reload the configuration values upon reset request but whether the chip is really doing that must be investigated once again.

References

List of Figures

List of Tables